atbalsts izcilībai

MĀCĪBU PRIEKŠMETU OLIMPIĀŽU
UZDEVUMU UN TO RISINĀJUMU KRĀJUMS

Informātika
(programmēšana)

RĪGA 2021

Valsts izglītības satura centrs
NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020
EIROPAS SAVIENĪBA
Eiropas Sociālais fonds
IEGULDĪJUMS TAVĀ NĀKOTNĒ

Mācību priekšmetu olimpiāžu uzdevumu un to risinājumu krājums ir izstrādāts Valsts izglītības satura centra projekta "Nacionāla un starptautiska mēroga pasākumu īstenošana izglītojamo talantu attīstībai" (projekta Nr. 8.3.2.1/16/I/002) ietvaros un aptver mācību priekšmetu olimpiādēs izstrādāto saturu no 2017. līdz 2020. gadam.

Autoru kolektīvs:

Mārtiņš Opmanis
Rihards Opmanis
Pēteris Pakalns
Jevgēnijs Vihrovs
Normunds Vilciņš

Tehniskais redaktors:
Mārtiņš Opmanis

ISBN 978-9934-24-033-1

Izcilība ir personības iezīme, kas attīstās ilgā darbā, izkopjot savas prasmes un zināšanas. Domājot par sasniegumiem, jāņem vērā arī zināšanu dziļums, kas sekmējis šo izcilo sniegumu ne tikai valsts, bet arī starptautiskā mērogā. Mācību priekšmetu olimpiādes ir patiesi godīga sacensība starp zinošākajiem un izturīgākajiem skolēniem – tiem, kas nepadodas grūtībām, rod iedvesmu un meklē aizvien jaunākus un radošākus risinājumus. Gadi pierādījuši, ka olimpiāžu laureāti un dalībnieki veido talantīgu, kā arī konkurētspējīgu Latvijas zinātnes un uzņēmējdarbības paaudzi ar iespējām radīt nozīmīgas inovācijas un rast risinājumus sabiedrības dzīves kvalitātes uzlabošanai.

Valsts izglītības satura centra vārdā vēlu visiem skolēniem un viņu pedagogiem iedvesmu, aizrautību un izaicinājumu, risinot šo olimpiāžu krājumu uzdevumus.

Līga Lejiņa
Valsts izglītības satura centra vadītāja

Saturs

IEVADS **8**

UZDEVUMI **10**

2016./2017. MĀCĪBU GADS **10**

NOVADA OLIMPIĀDE - 2017 10
Dalītāju summa *10*
Rēķināšanas spēle *10*
Romāns *12*
Garākais fragments *13*
Puzle *14*
Vienreizējie dalītāji *16*
VALSTS OLIMPIĀDE - 2017 17
Cirks *17*
Dēlīši *18*
Alu labirints *20*
Maucis *21*
Mediāna *23*
0 un 1 ..... *25*
Parabola *26*
Punkti taisnstūrī *26*
Puzle *27*
Viktorīna *29*

2017./2018. MĀCĪBU GADS **31**

NOVADA OLIMPIĀDE - 2018 31
Apakštaisnstūri *31*
Ciparu aizvietošana *33*
Nav apakšvirkne *33*
Peļķes *35*
Receptes *36*
VALSTS OLIMPIĀDE - 2018 37
Autosacīkstes-1 ..... *37*
Autosacīkstes-2 ..... *39*
Augļudārza platība *41*
Figūras-1 ..... *42*
Figūras-2 ..... *43*
Minibači *45*
Pārvākšanās *46*
Svara rāvējslēdzējs *47*
Teniss *48*
Skaitļu tornis *49*

2018./2019. MĀCĪBU GADS **51**

NOVADA OLIMPIĀDE - 2019 51
Attālums kokā *51*
Virknes fragments *52*
Kastu tornis *53*
Mazākais taisnstūris *54*
Sēņošanas čempionāts *55*

VALSTS OLIMPIĀDE - 2019 57
Akmens, šķēres, papīrīt's *57*
Nesalasāmie burti *58*
Uzdevumu grāmata *59*
Ingus koeficients *61*
Meklēšana periodiskā virknē *62*
Divi mīnusi *63*
Pāra skaits ciparu *64*
Rūtiņu laukuma sagriešana *64*
Sniega novākšana *65*
Veikals *67*

2019./2020. MĀCĪBU GADS 69

NOVADA OLIMPIĀDE - 2020 69
Cik labo? *69*
Latīņu kvadrāti *70*
Lakatiņa summa *71*
Neder kā summa *72*
"Pēdējās formulas" autosacīkstes *73*
Torņi *75*
VALSTS OLIMPIĀDE - 2020 76
Daudzstūra sadalīšana *76*
Divas iekavas *77*
Formula *78*
Galapunkti *80*
Skandalozā izrāde *80*
Trīsstūru labirints *82*
Laivotāji *84*
Mona Luīze *85*
Kašķīgais parlaments *86*
Teniss *88*
Izslēdzošais VAI *90*

ATRISINĀJUMI 92

2016./2017. MĀCĪBU GADS 92

NOVADA OLIMPIĀDE - 2017 92
Dalītāju summa *92*
Rēķināšanas spēle *92*
Romāns *94*
Garākais fragments *94*
Puzle *95*
Vienreizējie dalītāji *95*
VALSTS OLIMPIĀDE - 2017 96
Cirks *96*
Dēliši *97*
Alu labirints *98*
Maucis *99*
Mediāna *100*
0 un 1 ..... *101*
Parabola *102*
Punkti taisnstūrī *103*

Puzle ... *103*
Viktorīna ... *104*

2017./2018. MĀCĪBU GADS ... 105

NOVADA OLIMPIĀDE - 2018 ... 105
Apakštaisnstūri ... *105*
Ciparu aizvietošana ... *106*
Nav apakšvirkne ... *108*
Peļķes ... *108*
Receptes ... *109*
VALSTS OLIMPIĀDE - 2018 ... 109
Autosacīkstes-1 ... *109*
Autosacīkstes-2 ... *110*
Augļudārza platība ... *111*
Figūras-1 ... *112*
Figūras-2 ... *112*
Minibači ... *112*
Pārvākšanās ... *113*
Svara rāvējslēdzējs ... *113*
Teniss ... *114*
Skaitļu tornis ... *114*

2018./2019. MĀCĪBU GADS ... 117

NOVADA OLIMPIĀDE - 2019 ... 117
Attālums kokā ... *117*
Virknes fragments ... *119*
Kastu tornis ... *119*
Mazākais taisnstūris ... *120*
Sēņošanas čempionāts ... *121*
VALSTS OLIMPIĀDE - 2019 ... 121
Akmens, šķēres, papīrīt's ... *121*
Nesalasāmie burti ... *123*
Ingus koeficients / Uzdevumu grāmata ... *125*
Meklēšana periodiskā virknē ... *129*
Divi mīnusi ... *129*
Pāra skaits ciparu ... *130*
Rūtiņu laukuma sagriešana ... *134*
Sniega novākšana ... *136*
Veikals ... *137*

2019./2020. MĀCĪBU GADS ... 140

NOVADA OLIMPIĀDE - 2020 ... 140
Cik labo? ... *140*
Latīņu kvadrāti ... *140*
Lakatiņa summa ... *140*
Neder kā summa ... *141*
"Pēdējās formulas" autosacīkstes ... *141*
Torņi ... *142*
VALSTS OLIMPIĀDE - 2020 ... 143
Daudzstūra sadalīšana ... *143*
Divas iekavas ... *144*

Formula *146*
Galapunkti *147*
Skandalozā izrāde *148*
Trīsstūru labirints *148*
Laivotāji *151*
Mona Luīze *153*
Kašķīgais parlaments *153*
Teniss *155*
Izslēdzošais VAI *156*

Ievads

Informātikas olimpiāde ir viena no "lielajām" mācību priekšmetu olimpiādēm, kurās Vispasaules olimpiādes IOI (*International Olympiad in Informatics*) notiek kopš 1989. gada. Latvijas valstsvienības IOI piedalās kopš 1992. gada un kopumā Vispasaules olimpiādēs ir izcīnījušas 80 medaļas (septiņas zelta, 23 sudraba un 50 bronzas).

Latvijā informātikas olimpiādes oficiāli notiek kopš 1988. gada, lai gan par pirmo *de-facto* olimpiādi droši var uzskatīt 1986. gada februārī LU Matemātikas un informātikas institūtā (LU MII, toreiz - LVU Skaitļošanas centrā) notikušo atklāto informātikas olimpiādi. Ja olimpiādes pirmsākumos "informātika" un "programmēšana" bija sinonīmi, tad šobrīd tā vairs nav. Olimpiādes nosaukumu ir nācies papildināt ar vārdu "programmēšanas", lai būtu pilnīgi skaidrs, kāda ir olimpiādes būtība. Laika gaitā ir mainījušās olimpiādēs izmantojamās programmēšanas valodas, datortehnikas modeļi un pieejamība, risinājumu testēšanas paņēmieni, bet nemainīgs ir palicis olimpiāžu saturs - efektīvu algoritmu izstrāde. Olimpiāžu saturu un formātu galvenokārt ietekmē "pasaules vēsmas" - tas, kā tiek organizēta un kādas uzdevumu tēmas ir pieļautas Vispasaules olimpiādē. Ne vienmēr LIO žūrijas viedoklis par jaunievedumu kvalitāti sakrīt ar IOI zinātniskās komitejas viedokli.

Tradicionāli Latvijas informātikas olimpiādi (LIO) organizē Valsts Izglītības satura centrs kopā ar LU MII, katru gadu piedaloties citai kādas Latvijas pilsētas pašvaldībai un izglītības iestādei. LIO tradicionāli notiek divās vecuma grupās (jaunākā - 8.-10. klašu un vecākā - 11.-12. klašu) vairākos posmos: novembrī parasti notiek t.s. "iesildīšanās" sacensības, janvārī - Novada olimpiāde, kuras labāko rezultātu ieguvēji tiek uzaicināti uz LIO finālsacensībām - Valsts olimpiādi (parasti notiek februāra beigās).

Valsts olimpiādes medaļu ieguvēji tiek uzaicināti uz atlases sacensībām, pēc kuras rezultātiem tiek atlasīti Baltijas informātikas olimpiādes (BOI, notiek kopš 1995. gada) seši dalībnieki un, ņemot vērā BOI rezultātus, tiek noteikts Latvijas valstsvienības sastāvs dalībai IOI - četri dalībnieki.

Šajā uzdevumu krājumā ir apkopoti no 2017. līdz 2020. gadam notikušo Latvijas informātikas olimpiādes 2. (Novadu olimpiāde) un 3. (Valsts olimpiāde) posma uzdevumi ar atrisinājumiem. Uzdevumi ir gauži atšķirīgi pēc grūtības pakāpes, tāpēc arī atrisinājumi būtiski atšķiras. Ja vienkāršākajiem no tiem ir vienkārši ieskicēta atrisinājuma ideja, tad sarežģītākajiem ir dots detalizēts risinājuma izklāsts. Svarīga ir arī katra rakstnieka izpratne par labāko risinājuma pasniegšanas stilu, kuru neesmu centies lauzt vai unificēt. Tajā pat laikā, reti kura uzdevuma risinājums šeit ir tāds, ka atliek tikai to lasīt un rakstīt datorprogrammu - parasti būs nepieciešams saprast aprakstīto ideju un pašam izvēlēties tās realizēšanai nepieciešamos līdzekļus (piemēram, datu struktūras).

Salīdzinot ar uzdevumu tekstiem, kas tika doti sacensībās, šajā krājumā uzdevumu formulējumos nav sadaļas "Ierobežojumi un prasības", kurā bija dota informācija par tehniskiem nosacījumiem, kas jāievēro, sagatavojot risinājumus iesūtīšanai testēšanas sistēmā (piemēram, kā jānosauc klase valodā Java rakstītā risinājumā).

Līdz 2017. gada olimpiādei ievaddati tika ielasīti no teksta datnes un izvaddati bija jāieraksta teksta datnē. Uzdevumu formulējumi ir pārfrāzēti tā, ka ievaddati jāielasa no standarta ievada un izvaddati jāraksta standarta izvadā.

Uzdevumos ir aprakstīti apakšuzdevumi, kurus atrisinot var iegūt noteiktu punktu skaitu. Parasti apakšuzdevumi ir doti pieaugošā grūtības secībā un pirmo apakšuzdevumu atrisināšanai pietiek ar neefektīva (bet joprojām pareiza!) algoritma realizāciju. Latvijas informātikas olimpiādes "firmas zīme" ir pirmais apakšuzdevums, kura ievaddati ir doti uzdevuma formulējumā un kuru atrisinot (bieži tas iespējams pat bez datora) iespējams iegūt divus punktus. Tiesa, LIO žūrijā valda viedoklis, ka sacensību dalībnieki joprojām nepietiekami labi izmanto apakšuzdevumus, lai iegūtu pēc iespējas lielāku punktu skaitu.

Šajā uzdevumu krājumā apkopotos uzdevumus un to risinājumus olimpiādēm sagatavoja: Eduards Kaļiņičenko, Sergejs Meļņiks, Mārtiņš Opmanis, Rihards Opmanis, Rūdolfs Opmanis, Pēteris Pakalns, Kārlis Seņko, Roberts Leonārs Svarinskis, Artūrs Verza, Andrejs Vihrovs, Jevgēnijs Vihrovs, Normunds Vilciņš un Aleksandrs Zajakins.

Cerams, ka šis uzdevumu krājums palīdzēs gan dalībniekiem, gan viņu skolotājiem un treneriem labāk sagatavoties turpmākajām programmēšanas sacensībām.

M.Opmanis

Uzdevumi

2016./2017. mācību gads

Novada olimpiāde - 2017

Dalītāju summa

Katram naturālam skaitlim K iespējams aprēķināt visu tā dalītāju summu $S(K)$.

Piemēram,

$$S(10) = 1 + 2 + 5 + 10 = 18,$$
$$S(11) = 1 + 11 = 12,$$
$$S(12) = 1 + 2 + 3 + 4 + 6 + 12 = 28.$$

Uzrakstiet programmu, kas dotiem naturāliem skaitļiem A un B (A≤B) aprēķina summu $S(A) + S(A+1) + \cdots + S(B-1) + S(B)$!

Ievaddati

Pirmajā rindā dotas divu naturālu skaitļu A un B vērtības ($A \leq B \leq 10^9$), kas atdalītas ar tukšumzīmi.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis – aprēķinātās summas vērtība.

Piemēri

Ievaddati	Izvaddati
10 12	58

Ievaddati	Izvaddati
6 6	12

1. apakšuzdevuma testu ievaddati

Ievaddati
1 20

Ievaddati
18 30

Ievaddati
6 15

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	B ≤ 1000	10
3.	1000 < B ≤ 10000	20
4.	10000 < B ≤ 10^5	28
5.	Bez papildu ierobežojumiem	40
	Kopā:	100

Rēķināšanas spēle

Rihards ir izdomājis rēķināšanas galvā spēli, kas sastāv no divām darbībām: skaitļu saskaitīšanas un ciparu svītrošanas. Viņš lūdz Ievai saukt veselus skaitļus un rīkojas šādi:

ja skaitlis ir pozitīvs, tad pieskaita to iepriekš aprēķinātajai skaitļu summai (pašā sākumā summa ir 0);

ja skaitlis nav pozitīvs, tad no iepriekš aprēķinātās skaitļu summas izsvītro visus tos ciparus, kas sastopami šajā skaitlī. Atlikušie cipari skaitļa pierakstā tiek "sabīdīti kopā" un atmetot nebūtiskās nulles pieraksta sākumā, iegūstot cita vesela skaitļa pierakstu. Ja tiek izsvītroti visi cipari, tad uzskatām, ka jaunais skaitlis ir 0;

Piemēram, ja Ieva pēc kārtas nosauc skaitļus 723, -54, 717, -54, 0, 2050, -213141, tad summa pēc katra skaitļa nosaukšanas mainās (vai nemainās) šādi:

Skaitlis	Summa	Komentārs
723	723	
-54	723	Summā nav ciparu 5 vai 4, tāpēc tā nemainās
717	1440	
-54	10	Summā bija cipari 4, tie tika izsvītroti
0	1	
2050	2051	
-213141	5	Cipari 1 un 2 ir ievadītajā skaitlī - tie tiek izsvītroti. Papildus tiek atmesta nebūtiskā nulle.

Uzrakstiet programmu, kas nosaka, kāda būs skaitļu summas vērtība pēc visu skaitļu apstrādes!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N($N \leq 10^5$) - Ievas nosaukto skaitļu skaits. Katrā no nākamajām N rindām dots viens vesels skaitlis, kura vērtība pēc moduļa nepārsniedz 10^{12}. Katram i($i \leq i \leq N$) i+1-ajā rindā dots skaitlis, kuru Ieva nosauca kā i-to pēc kārtas.

Izvaddati

Vienīgajā rindā jāizvada vesels skaitlis – aprēķinātās skaitļu summas vērtība pēc visu skaitļu apstrādes.

Piemēri

Ievaddati	Izvaddati
7 723 -54 717 -54 0 2050 -213141	5

Ievaddati	Izvaddati
2 -11 -12	0

1. apakšuzdevuma testu ievaddati

Ievaddati
12 87 -698 7 55 -763 47 37 24 -11 13 0 -11

Ievaddati
12 36 -61 36 -61 36 -61 36 -61 36 -61 36 -61

Ievaddati
11 87654321 -6 -7 543210 -8 -9 987 -1 142 0 17

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Visi nosauktie skaitļi ir robežās no -9 līdz 10000	10
3.	Bez papildu ierobežojumiem	88
	Kopā:	100

Romāns

Žurnāla "Ko tu man stāsti!" redkolēģija ir nolēmusi nākamajos ne vairāk kā N (N - naturāls skaitlis) numuros turpinājumos publicēt talantīga rakstnieka romānu, kas sastāv no K nodaļām. Katrā žurnāla numurā jāpublicē viena vai vairākas romāna nodaļas pēc kārtas (nodaļu secību jaukt nedrīkst!). Katra nodaļa žurnāla numurā jāpublicē pilnībā. Ir zināms, cik žurnāla lappušu aizņem katra no nodaļām. Nepieciešams noteikt, kāds ir mazākais iespējamais lappušu skaits, cik žurnālā aizņems pati apjomīgākā vienā žurnāla numurā publicētā romāna daļa.

Piemēram, ja romāna astoņu nodaļu lappušu skaits ir 10, 1, 2, 3, 4, 5, 6, 8 un to paredzēts publicēt trīs numuros, tad var panākt, ka mazākais lappušu skaits pašā apjomīgākajā vienā žurnāla numurā publicētā romāna daļā ir 14. To var iegūt divos veidos:

a) 1. numurā publicēt 1. un 2. nodaļu (10+1=11 lappuses), 2. numurā publicēt 3. - 6. nodaļu (2+3+4+5=14 lappuses), 3. numurā publicēt 7. un 8. nodaļu (6 + 8 = 14 lappuses).

b) 1. numurā publicēt 1. - 3. nodaļu (10+1+2=13 lappuses), 2. numurā publicēt 4. - 6. nodaļu (3+4+5=12 lappuses), 3. numurā publicēt 7. un 8. nodaļu (6 + 8 = 14 lappuses).

Visos citos gadījumos apjomīgākajā vienā žurnāla numurā publicētās daļas apjoms būs lielāks par 14 lappusēm.

Uzrakstiet programmu, kas dotam žurnālu numuru skaitam un nodaļu aizņemto lappušu skaitam nosaka, kāds ir mazākais iespējamais lappušu skaits, cik žurnālā aizņems pati apjomīgākā vienā žurnāla numurā publicētā romāna daļa!

Ievaddati

Pirmajā rindā doti divi naturāli skaitļi N($N \le 10^5$) un K($K \le 10^5$), kas atdalīti ar tukšumzīmi. N vērtība ir lielākais žurnāla numuru skaits, kādā jāpublicē viss romāns, bet K vērtība ir romāna nodaļu skaits.

Otrajā rindā doti K naturāli skaitļi L_1, L_2, …, L_K. Katram i($1 \le i \le K$) L_i ir žurnāla lappušu skaits, kāds nepieciešams katras nodaļas publicēšanai. Nevienai nodaļai šis skaitlis nepārsniedz 10^9 lappušu. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Pirmajā rindā jāizvada naturāls skaitlis - mazākais iespējamais lappušu skaits, cik žurnālā aizņems pati apjomīgākā vienā žurnāla numurā publicētā romāna daļa.

Piemēri

Ievaddati	Izvaddati
3 8 10 1 2 3 4 5 6 8	14

Ievaddati	Izvaddati
5 2 5 6	6

1. apakšuzdevuma testu ievaddati

Ievaddati
11 16 31 20 48 44 95 23 82 42 58 25 28 90 71 38 76 70

Ievaddati
10 13 619 49 221 407 951 721 259 819 334 819 567 698 977

Ievaddati
3 7 7760 9473 1574 153 8676 7995 261

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Visa romāna kopējais lappušu skaits nepārsniedz 10000	10
3.	K ≤ 100	12
4.	Bez papildu ierobežojumiem	76
	Kopā:	100

Garākais fragments

Dotā naturālu skaitļu virknē nepieciešams atrast pēc iespējas garāku fragmentu (secīgu elementu apakšvirkni), kurā neviens skaitlis neatkārtojas.

Piemēram, virknē 1, 2, 2, 7, 1, 3, 1, 3, 2, 8, 3, 1, 2, 8 garākais fragments satur četrus elementus un šādi fragmenti ir atrodami piecās virknes vietās:

a) 1, 2, 2, 7, 1, 3, 1, 3, 2, 8, 3, 1, 2, 8

b) 1, 2, 2, 7, 1, 3, 1, 3, 2, 8, 3, 1, 2, 8

c) 1, 2, 2, 7, 1, 3, 1, 3, 2, 8, 3, 1, 2, 8

d) 1, 2, 2, 7, 1, 3, 1, 3, 2, 8, 3, 1, 2, 8

e) 1, 2, 2, 7, 1, 3, 1, 3, 2, 8, 3, 1, 2, 8

Uzrakstiet programmu, kas dotai virknei atrod lielāko šādu fragmentu garumu un visas vietas, kur virknē šādi fragmenti ir atrodami!

Ievaddati

Pirmajā rindā dots naturāls skaitlis N($N \le 10^5$) – virknes elementu skaits. Nākamajā rindā doti N naturāli skaitļi – virknes locekļi, kas atdalīti ar tukšumzīmēm. Zināms, ka neviena virknes locekļa vērtība nepārsniedz 10^9.

Izvaddati

Pirmajā rindā jāizvada divi naturāli skaitļi, kas atdalīti ar tukšumzīmi: garākā fragmenta garums G un šādu fragmentu skaits dotajā virknē S. Otrajā rindā augošā secībā jāizvada S naturāli skaitļi – visu to virknes elementu indeksu numuri, kur sākas fragmenti ar aprakstītajām īpašībām. Elementu indeksācija sākas ar 1.

Piemēri

Ievaddati	Izvaddati
14 1 2 2 7 1 3 1 3 2 8 3 1 2 8	4 5 3 7 9 10 11

Ievaddati	Izvaddati
5 4 1 7 4 1	3 3 1 2 3

1. apakšuzdevuma testu ievaddati

Ievaddati
30 16 27 40 54 48 32 36 19 8 4 48 27 16 8 40 36 54 32 19 4 4 32 36 8 48 54 16 27 40 19

Ievaddati
32 9 8 5 2 5 4 1 4 9 6 7 5 6 9 3 8 2 6 8 7 5 1 5 9 8 7 3 9 7 1 1 1

Ievaddati
27 64 11 39 88 67 18 64 11 39 64 11 39 64 11 39 64 11 39 88 67 18 64 11 39 88 67 18

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 3000	16
3.	Neviena virknes elementa vērtība nepārsniedz 10^6	10
4.	Bez papildu ierobežojumiem	72
	Kopā:	100

Puzle

Dacei patīk salikt un izjaukt puzles – mīklas, kas sastāv no gabaliņiem, kas jāsavieno pareizā secībā. Katrs gabaliņš ir unikāls un tas puzlē iederas tikai vienā vietā. Varam uzskatīt, ka katrs gabaliņš ir vienu rūtiņu liels un salikta mīkla veido N rindas un M kolonnas lielu taisnstūri. Kad Dace ir salikusi kādu mīklu pirmoreiz, viņa katram gabaliņam otrā pusē uzraksta šī gabaliņa rindas un kolonnas numurus. Gan rindas, gan kolonnas Dace numurē ar naturāliem skaitļiem pēc kārtas, sākot no 1. Divi puzles gabaliņi ir kaimiņi tikai tad, ja tiem ir kopīga mala - t.i., ja viens no to numuriem (rindas vai kolonnas) sakrīt, bet otrs - atšķiras par 1.

Lai audzinātu savu mazo brāli Kārli, Dace ir izdomājusi šādu spēli: Dace dod Kārlim iepriekš saliktas puzles gabaliņus pa vienam kaut kādā secībā, un Kārlim ir jāveic viena no šādām darbībām:

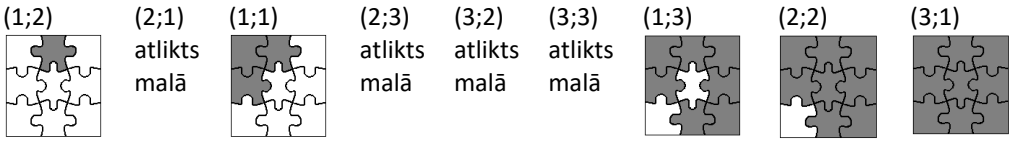
A) ja tas ir pirmais puzles gabaliņš, tad tas jānovieto uz galda un ar šo gabaliņu sākas puzles salikšana, aizvien papildinot iepriekš izveidoto puzles *fragmentu*, līdz puzle būs salikta. Arī viens pats gabaliņš ir puzles fragments;

B) ja iepriekš saliktajā puzles fragmentā atrodas kāds no šī gabaliņa kaimiņiem, tad šis gabaliņš jāpievieno fragmentam un jāpievieno arī visi iepriekš malā atliktie gabaliņi, ko tagad ir iespējams pievienot;

C) ja iepriekš saliktajā puzles fragmentā neatrodas neviens no šī gabaliņa kaimiņiem, tad šis gabaliņš jāatliek malā;

Kārlis ir ļoti uzmanīgs un, izpildot minētās darbības, nekļūdās.

Piemēram, ja N=3, M=3 un gabaliņi tiek doti šādā secībā (iekavās norādīta gabaliņa rinda un kolonna): (1;2), (2;1), (1;1), (2;3), (3;2), (3;3), (1;3), (2;2), (3;1), tad puzle tiek salikta tā, kā redzams 1. zīmējumā.



1. zīm.

Uzrakstiet programmu, kas nosaka, kāds puzles salikšanas laikā ir bijis lielākais skaits malā atlikto kauliņu!

Ievaddati

Pirmajā rindā doti divi naturāli skaitļi – puzles rindu skaits N($N \le 10^5$) un kolonnu skaits M($M \le 10^5$, $N \times M \le 10^5$), kas atdalīti ar tukšumzīmi. Katrā no nākamajām N×M rindām doti divi naturāli skaitļi, kas atdalīti ar tukšumzīmi. Katram i($1 \le i \le N \times M$) i+1-ajā rindā dots i-tā pēc kārtas gabaliņa rindas numurs r_i ($1 \le r_i \le N$) un kolonnas numurs k_i ($1 \le k_i \le M$). Informācija par katru gabaliņu ievaddatos dota vienreiz.

Izvaddati

Vienīgajā rindā jāizvada vesels nenegatīvs skaitlis – lielākais malā atlikto gabaliņu skaits puzles salikšanas laikā.

Piemēri

Ievaddati	Izvaddati
3 3 1 2 2 1 1 1 2 3 3 2 3 3 1 3 2 2 3 1	3

Ievaddati	Izvaddati
1 3 1 2 1 1 1 3	0

1. apakšuzdevuma testu ievaddati

Ievaddati
2 2 1 2 2 1 1 1 2 2

Ievaddati
2 4 2 4 1 3 1 4 2 2 2 3 1 1 2 1 1 2

Ievaddati
4 2 3 2 4 1 2 1 1 2 1 1 3 1 2 2 4 2

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	M = 1	18
3.	N ≤ 1000, M ≤ 1000	60
4.	Bez papildu ierobežojumiem	20
	Kopā:	100

Vienreizējie dalītāji

Katram naturālam skaitlim K iespējams atrast visus tā dalītājus.

Piemēram,

skaitļa 9 dalītāji ir skaitļi 1, 3 un 9;
skaitļa 10 dalītāji ir skaitļi 1, 2, 5 un 10;
skaitļa 11 dalītāji ir skaitļi 1 un 11;
skaitļa 12 dalītāji ir 1, 2, 3, 4, 6 un 12.

Aplūkojot skaitļu segmentu [A;B] (A≤B), varam atrast tos skaitļus, kas ir dalītāji vienam vai vairākiem skaitļiem no šī segmenta, un aprēķināt šo skaitļu summu. Citiem vārdiem – tiek aprēķināta visu segmentā ietilpstošo skaitļu dalītāju summa, katru no tiem pieskaitot vienreiz.

Piemēram, ja A=9 un B=12, tad šādi dalītāji ir skaitļi 1, 2, 3, 4, 5, 6, 9, 10, 11 un 12, un to summa ir 63.

Uzrakstiet programmu, kas dotam naturālu skaitļu segmentam aprēķina tajā ietilpstošo skaitļu dalītāju summu, katru no dalītājiem pieskaitot vienreiz!

Ievaddati

Pirmajā rindā dotas divu naturālu skaitļu A un B vērtības ($A \leq B \leq 10^9$), kas atdalītas ar tukšumzīmi.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis – visu skaitļu segmentā ietilpstošo skaitļu dalītāju summa, katru no dalītājiem pieskaitot vienreiz.

Piemēri

Ievaddati	Izvaddati
`9 12`	`63`

Ievaddati	Izvaddati
`7 7`	`8`

1. apakšuzdevuma testu ievaddati

Ievaddati
`77 260`

Ievaddati
`151 398`

Ievaddati
`365 515`

Apakšuzdevumi un to vērtēšana

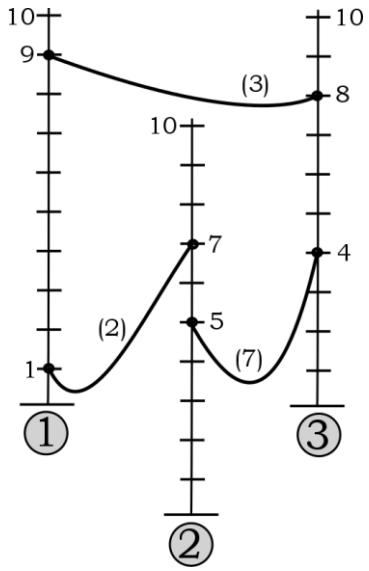
Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	B ≤ 1000	5
3.	B - A ≤ 1000	8
4.	B ≤ 10^6	10
5.	Bez papildu ierobežojumiem	75
	Kopā:	100

Cirks

Vienā no cirka numuriem piedalās N pērtiķi. Šajā numurā arēnā ir izvietotas N vertikālas kāpnes (numurētas pēc kārtas ar skaitļiem no 1 līdz N) ar M pakāpieniem katrā (pakāpieni ir numurēti no lejas uz augšu ar naturāliem skaitļiem no 1 līdz M pēc kārtas). Katru kāpņu galā (pie pēdējā pakāpiena) atrodas banāni.

Starp dažādu kāpņu pakāpieniem var būt novilktas virves. Katra virve saista divus dažādu kāpņu pakāpienus. Pie katra pakāpiena var būt piesieta ne vairāk kā viena virve. Neviena virve nav piesieta pie kāpņu augšējā pakāpiena.

Numura sākumā katrs pērtiķis atrodas pie savām kāpnēm apakšā. Izrādei sākoties, visi pērtiķi sāk rāpties pa trepēm uz augšu, veicot pa vienam pakāpienam sekundē. Ja kāds pērtiķis savā ceļā sastop virvi, tad viņš pa šo virvi noteiktā laikā (šis laiks dažādām virvēm var atšķirties) aiziet līdz otram galam un sāk rāpties augšup pa tām trepēm, uz kurām nonācis.



2. zīm.

Ja uz vienas virves vienlaikus atrodas vairāk par vienu pērtiķi, tad viņi var izmainīties viens otram netraucējot. Kad kāds pērtiķis ir sasniedzis trepju augšējo pakāpienu (un banānus!), viņš sāk tos notiesāt un vairāk nekur nepārvietojas.

Piemēram, ja numurā piedalās trīs pērtiķi, kāpnēm ir pa 10 pakāpieniem un virves savienotas tā, kā redzams 2. zīmējumā (pie katras virves norādīts laiks sekundēs, kādā pa to var tikt līdz otram galam), tad pērtiķi trepju augšējos pakāpienus sasniegs pēc attiecīgi 6, 20 un 28 sekundēm.

Uzrakstiet programmu, kas dotam sekunžu skaitam, kas pagājis no numura sākuma, nosaka, cik pērtiķi šajā brīdī jau ir sasnieguši trepju augšējos pakāpienus!

Ievaddati

Pirmajā rindā dotas veselu nenegatīvu skaitļu N ($1 \le N \le 10^5$), M ($1 \le M \le 10^5$), V (virvju skaits, $0 \le V \le 10^5$) un L (laika momentu skaits, $1 \le L \le 10^5$) vērtības kas atdalītas ar tukšumzīmi. Nākamajās V rindās katrā dots pa vienam virves aprakstam - pieci naturāli skaitļi, kas atdalīti ar tukšumzīmēm: T_1 (pirmo trepju numurs, $T_1 \le N$), P_1 (pirmo trepju pakāpiena numurs, $P_1 < M$), T_2 (otro trepju numurs, $T_2 \le N$, $T_1 \ne T_2$), P_2 (otro trepju pakāpiena numurs, $P_2 < M$), L_v (laiks sekundēs, kas nepieciešams, lai pērtiķis pa šo virvi pārvietotos no viena gala līdz otram, $L_v \le 10^5$). Ievaddatu V+2-ajā rindā doti L dažādi naturāli skaitļi, kuru vērtība nepārsniedz 2×10^9, un kas atdalīti ar tukšumzīmēm - laika momenti sekundēs no numura sākuma augošā secībā.

Izvaddati

Vienīgajā rindā jāizvada L veseli nenegatīvi skaitļi - pērtiķu skaits, kas attiecīgajā laika brīdī ir nonākuši līdz trepju augšējam pakāpienam.

Piemēri

Ievaddati	Izvaddati
3 10 3 6 1 9 3 8 3 3 4 2 5 7 2 7 1 1 2 5 10 15 20 25 30	0 1 1 2 2 3

Ievaddati	Izvaddati
2 100 0 3 70 100 200	0 2 2

1. apakšuzdevuma testu ievaddati

Ievaddati
10 158 5 5 8 56 10 96 53 6 17 8 81 20 3 39 1 151 2 8 75 6 61 78 10 95 9 145 100 122 221 264 298 455

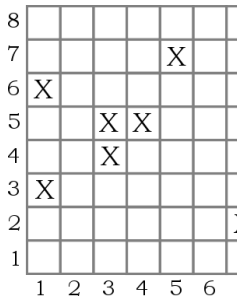
Ievaddati
4 100 6 5 2 50 3 34 70 2 12 3 61 38 1 77 4 36 65 2 17 4 34 70 4 12 2 62 34 3 36 2 1 43 151 156 215 262 284

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie divi testi	2
2.	N ≤ 2	8
3.	N ≤ 100, M ≤ 1000	25
4.	V ≤ 1000	25
5.	Bez papildu ierobežojumiem	40
	Kopā:	100

Dēlīši

Galdnieks Rūdis ir iegādājies dārgas koksnes plāksni, kura ir N cm plata un M cm augsta. Rūdim no šīs plāksnes nepieciešams izzāģēt 1 cm platus un K cm garus ļoti skaistus dēlīšus. Tā kā koka šķiedras virziens ir svarīgs, tad visi dēlīši jāzāģē vienā virzienā - paralēli kādai no plāksnes malām.



3. zīm. Plāksne

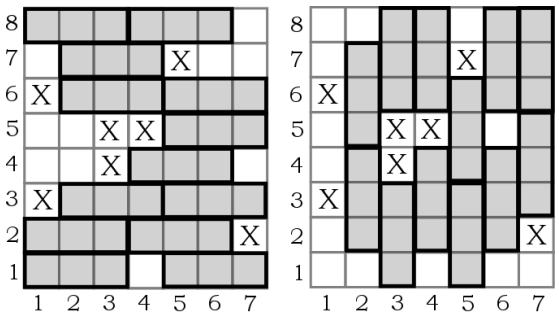
Ieskatoties vērīgāk, Rūdis ir pamanījis, ka plāksnē ir vairāki defekti, kuri nedrīkst būt dēlīšos. Lai zāģēšanas laikā defektus ņemtu vērā, Rūdis ir plāksni sadalījis N×M vienu kvadrātcentimetru lielos kvadrātiņos un ar krustiņiem ir atzīmējis defektiem atbilstošos kvadrātiņus kā neizmantojamus.

Piemēram, ja plāksnes izmēri ir 7 cm × 8 cm, nepieciešams izzāģēt 1 cm × 3 cm garus dēlīšus un septiņi defekti ir atzīmēti tā, kā redzams 3. zīmējumā, tad no šīs plāksnes varēs izzāģēt vai nu 13 dēlīšus, zāģējot horizontāli, vai 12 dēlīšus, zāģējot vertikāli. Viens no dēlīšu izzāģēšanas variantiem ar lielāko iespējamo dēlīšu skaitu katrā no virzieniem parādīts 4. zīmējumā.

Uzrakstiet programmu, kas nosaka, kādu lielāko dēlīšu skaitu iespējams izzāģēt no dotās plāksnes, visus dēlīšus zāģējot vienā virzienā!

Ievaddati

Pirmajā rindā dotas trīs naturālu skaitļu N ($N \le 10^9$), M($M \le 10^9$) un K($K \le 10^9$) vērtības, kas atdalītas ar tukšumzīmi. Otrajā rindā dota vesela nenegatīva skaitļa D (defektu skaits, $D \le 10^5$) vērtība. Nākamajās D rindās doti defektu apraksti - pa vienam katrā rindā. Katra defekta apraksts ir rūtiņas koordinātas - kolonnas numurs k($1 \le k \le N$) un rindas numurs r($1 \le r \le M$), kas atdalīti ar tukšumzīmi. Rindas un kolonnas ir numurētas ar naturāliem skaitļiem, sākot no 1, pēc kārtas.



4. zīm. Dēlīšu izzāģēšana

Izvaddati

Vienīgajā rindā jāizvada vesels nenegatīvs skaitlis - lielākais dēlīšu skaits, kādu iespējams izzāģēt no dotās plāksnes, visus dēlīšus zāģējot vienā virzienā.

Piemēri

Ievaddati	Izvaddati	Piezīme
7 8 3 7 3 5 7 2 3 4 5 7 1 6 4 5 1 3	13	Atbilst uzdevuma tekstā aprakstītajam piemēram.

Ievaddati	Izvaddati
3 3 3 0	3

1. apakšuzdevuma testu ievaddati

Ievaddati
5 5 3 4 1 3 5 2 4 5 4 4

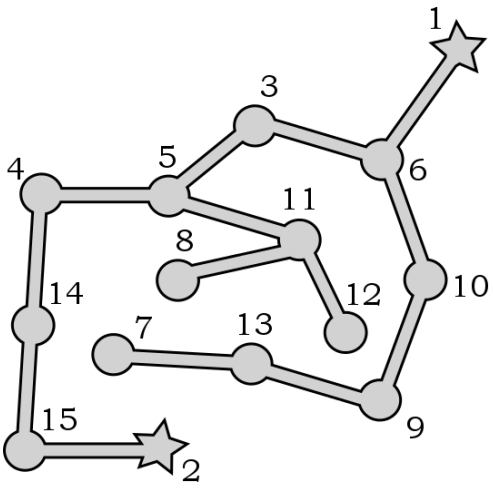
Ievaddati
1000 1 37 4 127 1 808 1 455 1 617 1

Ievaddati
3 600 3 4 3 581 1 211 2 21 3 185

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	D = 0	8
3.	N = M = K, D > 0	9
4.	N ≤ 1000, M ≤ 1000, 0 < D ≤ 100	10
5.	N > 100, M > 100, D > 0, katrā rindā un katrā kolonnā ir ne vairāk kā viens defekts	25
6.	Bez papildu ierobežojumiem	46
	Kopā:	100

Alu labirints



5. zīm.

Alu labirintam, kas sastāv no pazemes zālēm un tuneļiem, ir viena vai vairākas ieejas. Katrs tunelis savā starpā savieno vai nu ieeju un zāli, vai divas zāles. Katra ieeja ir savienota ar tieši vienu zāli. No katras ieejas līdz katrai zālei iespējams tikt tikai pa vienu maršrutu. Viena tuneļa šķērsošana aizņem vienu minūti. Pieņemot, ka katrā zālē atrodas pa vienam apmeklētājam, nepieciešams noteikt mazāko laiku, kādā visi apmeklētāji var pamest labirintu - tikt līdz tuvākajai izejai. Tiek pieņemts, ka zāles šķērsošana papildus laiku neprasa.

Piemēram, 5. zīmējumā attēlotajam alu labirintam mazākais laiks, kāds nepieciešams, lai visi apmeklētāji tiktu līdz kādai no divām izejām (attēlā tās apzīmētas ar 1 un 2), ir piecas minūtes.

Uzrakstiet programmu, kas dotajam alu labirintam atrod mazāko tā pamešanai nepieciešamo laiku!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu Z (zāļu skaits, $Z \le 10^5$) un E (ieeju skaits) vērtības, kas atdalītas ar tukšumzīmi. Ieejas ir numurētas ar naturāliem skaitļiem no 1 līdz E pēc kārtas. Zāles ir numurētas ar naturāliem skaitļiem no E+1 līdz E+Z pēc kārtas. Nākamajās E+Z-1 rindās doti tuneļu apraksti - pa vienam katrā rindā. Katra tuneļa aprakstu veido divi skaitļi, kas atdalīti ar tukšumzīmi - tuneļa galos esošo zāļu vai ieejas un zāles numuri.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis - mazākais laiks minūtēs, kādā visi apmeklētāji var tikt līdz kādai no labirinta izejām.

Piemērs

Ievaddati	Izvaddati
13 2 6 1 8 11 4 5 9 10 5 3 11 5 3 6 10 6 7 13 4 14 2 15 11 12 9 13 15 14	5

1. apakšuzdevuma testu ievaddati

Ievaddati
10 4 13 12 4 9 6 9 13 8 13 14 10 3 2 10 11 7 5 11 5 13 11 10 1 5 11 9

Ievaddati
14 3 5 10 10 15 7 13 8 14 5 8 8 16 8 12 17 6 11 1 8 11 6 13 13 5 2 17 15 3 16 4 9 16

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie divi testi	2
2.	No katras zāles iziet tieši divi tuneļi	10
3.	E = 1	20
4.	E ≤ 100	28
5.	Bez papildu ierobežojumiem	40
	Kopā:	100

Maucis

6. att.

Saskaņā ar Janīnas Kursītes "Vārdeni"[1], *mauči* jeb dūrgaļi ir "adītas manšetes, adīti rakstaini uzroči pie sieviešu tautastērpa." Tos ada spirāles veidā, izadot pavedienu ar dažādas krāsas pērlītēm. Lai veidotos vēlamais raksts, vispirms ir "jādomā lineāri" – visas pērlītes vispirms pareizā secībā jāsaver uz diega. Interesēsimies par vienkāršāko rakstu - pērlīšu kolonnām, kas veidojas perpendikulāri pavedienam (6. attēlā[2] šī daļa atzīmēta ar taisnstūri). Dotai pērlīšu virknei nepieciešams noteikt, no kuras līdz kurai pērlītei izadot, varēs dabūt vēlamā izmēra mauci ar vertikālu atkārtotu rakstu, kas nenobīdās sānis.

Piemēram, ja no 37 pērlīšu virknes (ar "m" apzīmētas melnas, ar "b"-baltas, bet ar "z" - zilas pērlītes)

bmbmmmbzmmbmmmbmmmbmmmbzmmbmmmbmmmbbm

nepieciešams dabūt divas rindas ar 16 pērlītēm katrā, tad jāsāk adīt no trešās pērlītes. Rezultātā tiks iegūts maucis

bmmmbzmmbmmmbmmm
bmmmbzmmbmmmbmmm

kuras visās kolonnās ir tikai vienas krāsas pērlītes.

[1] J.Kursīte Neakadēmiskā latviešu valodas vārdnīca jeb novadu vārdene. ISBN 978-9984-31-457-0 Madris, 2007.

[2] attēls no https://needleb.files.wordpress.com/2011/02/1.jpg

Mazliet mīkstināsim prasības un pieļausim, ka dažu pērlīšu krāsa maucī drīkst atšķirties no pārējo pērlīšu krāsas šajā kolonnā. Katrā kolonnā varam noteikt, kuras krāsas pērlītes tajā ir visvairāk, un uzskatīt, ka visas pārējās pērlītes šajā kolonnā ir *kļūdas*. Ja no iepriekšminētās pērlīšu virknes nepieciešams iegūt sešas rindas pa sešām pērlītēm katrā, tad iespējami tikai divi varianti - sākt adīt ar pirmo vai otro pērlīti:

bmbmmm	mbmmmb
bzmmbm	zmmbmm
mmbmmm	mbmmmb
bmmmbz	mmmbzm
mmbmmm	mbmmmb
bmmmbb	mmmbbm
Kļūdu skaits: 2+1+3+0+3+2=11	Kļūdu skaits: 1+3+0+3+2+3=12

Pirmajā variantā kopējais kļūdu skaits ir mazāks nekā otrajā. Ievērojiet, ka vairākās kolonnās melno un balto pērlīšu skaits ir vienāds, bet šādās kolonnās kļūdainās krāsas izvēle neietekmē kļūdu kopskaitu.

Uzrakstiet programmu, kas dotai pērlīšu virknei un mauča izmēriem nosaka, kādu mazāko kļūdu kopskaitu iespējams iegūt!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu K (pērlīšu skaits virknē, $K \le 10^5$), N (mauča rindu skaits, $N \ge 2$) un M (mauča kolonnu skaits, $M \ge 2$), kas atdalītas ar tukšumzīmi. Zināms, ka $N \times M \le K$. Otrajā rindā dots pērlīšu virknes apraksts - K naturālu skaitļu virkne, kur katrus divus blakus skaitļus atdala tukšumzīme. Katram k($1 \le k \le K$) k-tais skaitlis pēc kārtas apzīmē k-tās pēc kārtas pērlītes krāsu. Visas šajā maucī sastopamās pērlīšu krāsas ir sanumurētas ar naturāliem skaitļiem sākot no 1, pēc kārtas.

Izvaddati

Vienīgajā rindā jāizvada vesels nenegatīvs skaitlis - mazākais iespējamais kļūdu skaits, veidojot noteiktā izmēra mauci.

Piemēri

Ievaddati
37 2 16 1 2 1 2 2 2 1 3 2 2 1 2 2 2 1 2 2 2 1 2 2 2 1 3 2 2 1 2 2 2 1 2 2 2 1 1 2
Izvaddati
0

Ievaddati
37 6 6 1 2 1 2 2 2 1 3 2 2 1 2 2 2 1 2 2 2 1 2 2 2 1 3 2 2 1 2 2 2 1 2 2 2 1 1 2
Izvaddati
11

1. apakšuzdevuma testu ievaddati

Ievaddati
37 4 5 1 2 1 2 2 2 1 3 2 2 1 2 2 2 1 2 2 2 1 2 2 2 1 3 2 2 1 2 2 2 1 2 2 2 1 1 2

Ievaddati
37 10 3 1 2 1 2 2 2 1 3 2 2 1 2 2 2 1 2 2 2 1 2 2 2 1 3 2 2 1 2 2 2 1 2 2 2 1 1 2

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie divi testi	2
2.	Maucī ir tikai divu krāsu pērlītes	20
3.	M*N=K	10
4.	Bez papildu ierobežojumiem	68
	Kopā:	100

Pateicība

Paldies **Aijai Lūsei** par uzdevuma ideju!

Mediāna

Doti divi masīvi X un Y, kuru elementi sakārtoti nedilstošā secībā. Kopējais abu masīvu elementu skaits ir nepāru skaitlis. Ar šiem masīviem iespējams izpildīt šādas operācijas: *Operācija "A"*: visus_viena_masīva_elementus pareizināt ar -1,

Operācija "B(Δ)": visiem_viena_masīva_elementiem pieskaitīt konstanti Δ (Δ - vesels skaitlis),

Operācija "C": pieprasīt noteikt un izvadīt abu masīvu kopējo *mediānu* - elementu, kas sakārtotā abu masīvu elementu apvienotajā virknē, ir vidējais.

Piemēram, ja dotie masīvi ir X = { 2, 6, 12, 14, 14, 18} un Y = { 9, 10, 12, 12, 13}, tad izpildot operāciju "C", tiks izvadīts 12, jo sakārtotā abu masīvu elementu apvienotā virkne ir - 2, 6, 9, 10, 12, 12, 12, 13, 14, 14, 18 un vidējais (sestais pēc kārtas) ir elements 12. Pēc tam masīvam X pielietojot operāciju "A", masīvam Y - operāciju "B(-12)" un, visbeidzot, operāciju "C", tiks izvadīts -3, jo pēc pirmās operācijas izpildes X = { -2, -6, -12, -14, -14, -18}, pēc otrās Y = { -3, -2, 0, 0, 1} un sakārtotā abu masīvu elementu apvienotā virkne ir -18, -14, -14, -12, -6, -3, -2, -2, 0, 0, 1 un vidējais elements ir -3.

Uzrakstiet programmu, kas dotiem masīviem X un Y izpilda doto operāciju virkni!

Ievaddati

Pirmajā rindā doti trīs naturāli skaitļi – masīva X elementu skaits n_X ($n_X \le 10^6$), masīva Y elementu skaits n_Y ($n_Y \le 10^6$) un izpildīto operāciju skaits n_{OP} ($n_{OP} \le 10^5$), kas atdalīti ar tukšumzīmi.

Lai samazinātu ievaddatu apjomu, masīvu X un Y elementu vērtības tiek uzdotas ar rekurentu formulu palīdzību.

Otrajā rindā dota vesela skaitļa X_1, vesela nenegatīva skaitļa a_X, vesela nenegatīva skaitļa c_X un naturāla skaitļa m_X vērtības, kas atdalītas ar tukšumzīmēm. X_1 ir masīva X pirmā elementa vērtība, bet a_X, c_X un m_X ir konstantes, kuru vērtība nepārsniedz 10^9. Katram $k(2 \le k \le n_X)$ masīva elementu X_k aprēķina pēc formulas $X_k = X_{k-1}+d_k$, kur $d_k = (a_X d_{k-1}+c_X) \bmod m_X$, $d_1 = 0$. Ar "P mod Q" apzīmē atlikumu, kādu iegūst dalot veselu skaitli P ar naturālu skaitli Q. Skaitļu X_1, a_X, c_X un m_X vērtības ir tādas, ka neviena masīva X elementa vērtība pēc moduļa nepārsniedz 10^9.

Trešajā rindā dota vesela skaitļa Y_1, vesela nenegatīva skaitļa a_Y, vesela nenegatīva skaitļa c_Y un naturāla skaitļa m_Y vērtības, kas atdalītas ar tukšumzīmēm. Y_1 ir masīva Y pirmā elementa vērtība, bet a_Y, c_Y un m_Y ir konstantes, kuru vērtība nepārsniedz 10^9. Katram $k(2 \le k \le n_Y)$ masīva elementu Y_k aprēķina pēc formulas $Y_k = Y_{k-1}+f_k$, kur $f_k = (a_Y f_{k-1}+c_Y) \bmod m_Y$, $f_1 = 0$. Skaitļu Y_1, a_Y, c_Y un m_Y vērtības ir tādas, ka neviena masīva Y elementa vērtība pēc moduļa nepārsniedz 10^9.

Atlikušajās n_{OP} rindās ir aprakstītas veiktās operācijas - pa vienai katrā rindā. Katras operācijas apraksta rindā pirmais simbols ir operācijas veids - burts "A", "B" vai "C".

Operācijām "A" un "B" kā otrais simbols ir masīva nosaukums - burts "X" vai "Y", kas no operācijas veida ir atdalīts ar tukšumzīmi.

Operācijai “B” kā beidzamā dota operācijas parametra - vesela skaitļa Δ ($-2 \cdot 10^9 \le \Delta \le 2 \cdot 10^9$) vērtība, kas no masīva nosaukuma atdalīta ar tukšumzīmi.

Katram j ($1 \le j \le n_{OP}$) j-tās operācijas pēc kārtas apraksts dots ievaddatu j+3-ajā rindā.

Ir zināms, ka operāciju virknē ir vismaz viena operācija “C” un neviena masīva neviena elementa vērtība pārveidošanas laikā pēc moduļa nepārsniegs 10^9.

Izvaddati

Izvaddatos jābūt tik rindām, cik ievaddatu operāciju virknē ir operāciju “C”. Katrā rindā jāizvada viens vesels skaitlis - operācijas “C” izpildes vērtība. Katram k ($1 \le k \le$ operāciju “C” skaits) k-tās pēc kārtas operācijas “C” rezultāts jāizvada k-tajā rindā.

Piemērs

Ievaddati	Izvaddati	Piezīme
6 5 4 2 3 4 10 9 1 1 3 C A X B Y -12 C	12 -3	Masīvu X un Y elementu vērtības atbilst uzdevuma tekstā dotajām un tiek aprēķinātas tā, kā parādīts nākamajā tabulā

i	di	xi	fi	yi
1	0	2	0	9
2	(3·0+4) mod 10 = 4	2+4= 6	(1·0+1) mod 3 = 1	9+1=10
3	(3·4+4) mod 10 = 6	6+6=12	(1·1+1) mod 3 = 2	10+2=12
4	(3·6+4) mod 10 = 2	12+2=14	(1·2+1) mod 3 = 0	12+0=12
5	(3·2+4) mod 10 = 0	14+0=14	(1·0+1) mod 3 = 1	12+1=13
6	(3·0+4) mod 10 = 4	14+4=18		

1. apakšuzdevuma testu ievaddati

Ievaddati
18 17 6 -16 6 4 8 -155 7 10 14 B Y -186 C A Y C B Y 340 C

Ievaddati
11 6 8 0 2 0 14 -53 46 15 56 B X -466 C B Y 935 C A X C B Y -680 C

Ievaddati
20 11 10 -12 4 3 9 -19 1 2 26 B Y 283 C B X -811 C B Y -315 C B X 582 C B Y -545 C

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$n_X \leq 100$, $n_Y \leq 100$, $n_{OP} \leq 100$	8
3.	$n_X \leq 1000$, $n_Y \leq 1000$, $n_{OP} \leq 5000$	12
4.	$n_X \leq 10^6$, $n_Y \leq 10^6$, $n_{OP} \leq 200$	30
5.	Bez papildu ierobežojumiem	48
	Kopā:	100

0 un 1

Uzrakstiet programmu, kas nosaka, ar kādiem (pēc iespējas mazākā skaitā) simboliem jāpapildina dotā binārā virkne, lai kā apakšvirkni varētu atrast jebkuru bināru virkni garumā K.

Piemēram, ja dotā virkne ir 001101 un K=3, tad šajā virknē var atrast apakšvirknes 000, 001, 010, 011, 101, 110, 111, bet nevar atrast apakšvirkni 100. Doto virkni pietiek papildināt ar vienu simbolu, lai varētu atrast visas binārās apakšvirknes garumā trīs. To var izdarīt vairākos veidos: **1**001101, 0**1**01101, 00110**0**1 vai 0011010.

Ievaddati

Pirmajā rindā dotas divu naturālu skaitļu N (dotās virknes garums, $N \leq 10^5$) un K ($K \leq 10^5$) vērtības, kas atdalītas ar tukšumzīmi. Otrajā rindā dota simbolu virkne garumā N, kas sastāv no 0 un 1.

Izvaddati

Vienīgajā rindā jāizvada ar pēc iespējas mazāk simboliem 0 un 1 papildinātā virkne, kurā kā apakšvirknes iespējams atrast visas binārās virknes garumā K. Ja ir vairākas derīgas virknes, nepieciešams izvadīt jebkuru no tām.

Piemēri

Ievaddati	Izvaddati	Piezīme
6 3 001101	0011001	Atbilst uzdevuma tekstā aprakstītajam piemēram.

Ievaddati	Izvaddati	Piezīme
5 2 10001	10001	Jau dotajā virknē ir atrodamas visas apakšvirknes garumā 2 - virkni papildināt nav nepieciešams.

1. apakšuzdevuma testu ievaddati

Ievaddati
4 3 1100

Ievaddati
4 5 1111

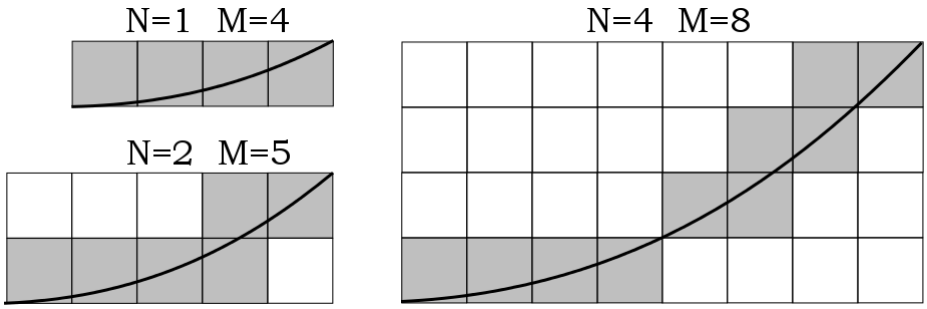
Ievaddati
9 4 101110001

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Visi simboli virknē ir vienādi	8
3.	Papildinātās virknes (atbildes) garums nepārsniedz 10	10
4.	$N \leq 50$, $K \leq 10$, virkni pietiek papildināt ar vienu simbolu	20
5.	$N \leq 100$, $K \leq 100$	20
6.	Bez papildu ierobežojumiem	40
	Kopā:	100

Parabola

Rūtiņu lapā, kurā ir N rindas un M kolonnas, tiek uzzīmēta parabola, kas iet no kreisā apakšējā stūra, kura koordinātas ir (0;0) uz labo augšējo stūri, kura koordinātas ir (M;N). Šīs parabolas vienādojums ir $y = N\left(\frac{x}{M}\right)^2$ un tā šķērso vienu vai vairākas lapas rūtiņas (tikai rūtiņas stūra šķērsošana netiek ieskatīta). Daži parabolu Piemēri ir parādīti 7. zīmējumā:



7. zīm.

Uzrakstiet programmu, kas dotām N un M vērtībām nosaka, cik lapas rūtiņu šķērso parabola!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N ($N \le 10^{18}$) un M ($M \le 10^{9}$) vērtības, kas atdalītas ar tukšumzīmi.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis - šķērsoto rūtiņu skaits.

Piemēri

Ievaddati	Izvaddati
`4 8`	`10`

Ievaddati	Izvaddati
`4 1`	`4`

1. apakšuzdevuma testu ievaddati

Ievaddati
`2401 5`

Ievaddati
`6272 32`

Ievaddati
`611 154`

Apakšuzdevumi un to vērtēšana

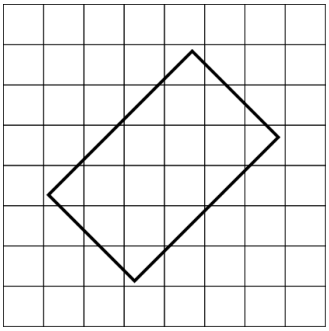
Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$M \le 1000$, $N \le 1000$	8
3.	$M \le 10^6$, $N \le 10^6$	12
4.	$N \le 10^9$	30
5.	Bez papildu ierobežojumiem	48
	Kopā:	100

Punkti taisnstūrī

Ir milzīga rūtiņu lapa, kas sastāv no kvadrātveida rūtiņām. Rūtiņas malas garums ir viena vienība. Šajā lapā uzzīmēts taisnstūris, kura malu garumi ir naturāls skaits vienību, diagonāļu krustpunkts ir kādas rūtiņas virsotne un malas paralēlas rūtiņu diagonālēm.

Nepieciešams noteikt, cik rūtiņu virsotnes atrodas taisnstūra iekšpusē.

Piemēram, ja taisnstūra malu garumi ir 3 un 5, tad taisnstūra iekšpusē ir 17 virsotnes (skat. 8. zīm.):



8. zīm.

Uzrakstiet programmu, kas nosaka, cik virsotnes atrodas taisnstūra iekšpusē!

Ievaddati

Pirmajā rindā doti taisnstūra malu garumi - divi naturāli skaitļi, kas atdalīti ar tukšumzīmi. Malu garumu vērtības nepārsniedz 2×10^9.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis - taisnstūra iekšpusē esošo virsotņu skaits.

Piemēri

Ievaddati	Izvaddati
3 5	17

Ievaddati	Izvaddati
1 1	1

1. apakšuzdevuma testu ievaddati

Ievaddati
95 80

Ievaddati
14 65

Ievaddati
22 50

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Malu garumi nepārsniedz 25	6
3.	Malu garumi nepārsniedz 10^4	16
4.	Malu garumi ir pāru skaitļi	28
5.	Bez papildu ierobežojumiem	48
	Kopā:	100

Puzle

Dacei patīk salikt un izjaukt puzles – mīklas, kas sastāv no gabaliņiem, kas jāsavieno pareizā secībā. Katrs gabaliņš ir unikāls, un tas puzlē iederas tikai vienā vietā. Varam uzskatīt, ka katrs gabaliņš ir vienu rūtiņu liels un salikta mīkla veido N rindas un M kolonnas lielu rūtiņu taisnstūri. Kad Dace ir salikusi kādu mīklu pirmoreiz, viņa katram gabaliņam otrā pusē uzraksta šī gabaliņa rindas un kolonnas numurus. Gan rindas, gan kolonnas Dace numurē ar naturāliem skaitļiem pēc kārtas, sākot no 1. Divi puzles gabaliņi ir kaimiņi tikai tad, ja tiem ir kopīga mala - t.i., ja viens no to numuriem (rindas vai kolonnas) sakrīt, bet otrs - atšķiras par 1.

Lai audzinātu savu mazo brāli Kārli, Dace ir izdomājusi šādu spēli: Dace visus puzles gabaliņus sakārto virknē un dod tos Kārlim pa vienam pēc kārtas. Kārlim ir jāveic viena no šādām darbībām:

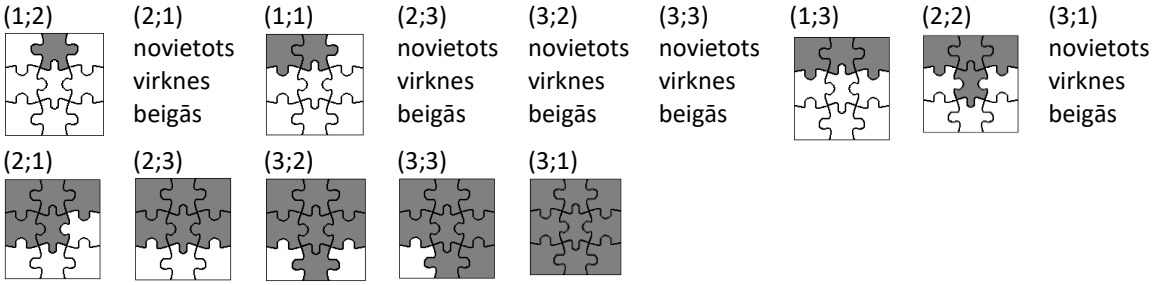
A) ja tas ir pirmais puzles gabaliņš, tad tas jānovieto uz galda un ar šo gabaliņu sākas puzles salikšana, aizvien papildinot iepriekš izveidoto puzles *fragmentu*, līdz puzle būs salikta. Arī viens pats gabaliņš ir puzles fragments;

B) ja iepriekš saliktajā puzles fragmentā atrodas kāds no šī gabaliņa kaimiņiem, tad šis gabaliņš jāpievieno fragmentam;

C) ja iepriekš saliktajā puzles fragmentā neatrodas neviens no šī gabaliņa kaimiņiem, tad šis gabaliņš jāatdod Dacei, kas novieto to virknes beigās.

Kārlis ir ļoti uzmanīgs un, izpildot minētās darbības, nekļūdās.

Piemēram, ja N=3, M=3 un Dace gabaliņus virknē sakārtojusi šādi (iekavās norādīta gabaliņa rinda un kolonna): (1;2), (2;1), (1;1), (2;3), (3;2), (3;3), (1;3), (2;2), (3;1), tad puzle tiek tā, kā redzams 9. zīmējumā.



9. zīm.

Šajā gadījumā Kārlis veica 14 darbības, līdz salika visu puzli.

Uzrakstiet programmu, kas dotiem puzles izmēriem un gabaliņu secībai Daces izveidotajā virknē nosaka, cik darbības Kārlis veica, līdz salika visu puzli!

Ievaddati

Pirmajā rindā doti divi naturāli skaitļi – puzles rindu skaits N un kolonnu skaits M ($N \times M \le 10^5$), kas atdalīti ar tukšumzīmi. Katrā no nākamajām N×M rindām doti divi naturāli skaitļi, kas atdalīti ar tukšumzīmi. Katram i($1 \le i \le N \times M$) ievaddatu i+1-ajā rindā dots i-tā pēc kārtas gabaliņa Daces virknē rindas numurs r_i ($1 \le r_i \le N$) un kolonnas numurs k_i ($1 \le k_i \le M$). Informācija par katru gabaliņu ievaddatos dota vienreiz.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis – Kārļa veikto darbību skaits.

Piemēri

Ievaddati	Izvaddati
3 3 1 2 2 1 1 1 2 3 3 2 3 3 1 3 2 2 3 1	14

Ievaddati	Izvaddati
1 3 1 2 1 1 1 3	3

1. apakšuzdevuma testu ievaddati

Ievaddati
4 4 3 4 1 1 2 2 4 4 1 2 1 3 2 3 3 2 4 1 3 1 3 3 1 4 4 2 2 4 4 3 2 1

Ievaddati
3 6 1 6 3 1 1 2 2 1 2 2 1 3 1 4 2 3 3 4 3 5 3 2 2 5 3 6 2 4 3 3 1 1 1 5 2 6

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie divi testi	2
2.	M = 1	18
3.	N ≤ 3000, M ≤ 3000	60
4.	Bez papildu ierobežojumiem	20
	Kopā:	100

Viktorīna

N dalībnieki piedalās viktorīnā, kurā pēc kārtas atbild uz vadītāja sagatavotajiem jautājumiem. Atbilde uz katru jautājumu ir vai nu pareiza, vai nepareiza. Ja dalībnieks pareizi atbild uz uzdoto jautājumu, tad dalībnieks saņem punktu un šim pašam dalībniekam tiek uzdots nākamais jautājums. Ja dalībnieks pareizi atbild uz K jautājumiem pēc kārtas, tad viņam tiek piešķirts papildpunkts un nākamais jautājums tiek uzdots nākamajam dalībniekam pēc kārtas. Ja dalībnieks uz jautājumu atbild nepareizi, tad šis pats jautājums tiek uzdots nākamajam dalībniekam pēc kārtas. Ja neviens no dalībniekiem nav pareizi atbildējis uz jautājumu, tad neviens dalībnieks punktu nesaņem, un viktorīnas vadītājs spēli turpina uzdodot nākamo jautājumu.

Piemēram, ja viktorīnā piedalās trīs dalībnieki un papildpunkts tiek piešķirts par diviem pareizi atbildētiem jautājumiem pēc kārtas un zināms, ka atbilžu secība bija `npppnnnnnnpppppnpnpnppp` (ar "p" apzīmētas pareizas, ar "n" - nepareizas atbildes), tad pirmais spēlētājs ieguva četrus, otrais - septiņus, bet trešais - sešus punktus.

Uzrakstiet programmu, kas dotai atbilžu virknei nosaka, cik punktu ieguva katrs no dalībniekiem!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N (dalībnieku skaits, $2 \le N \le 10^5$) un K (secīgu pareizu atbilžu skaits, lai saņemtu papildpunktu, $2 \le K \le 5$) vērtības, kas atdalītas ar tukšumzīmi. Lai samazinātu ievaddatu apjomu, vienādās atbildes tiek grupētas - pirmajā grupā ir pēc kārtas pareizo atbilžu skaits, otrajā grupā - nepareizo atbilžu skaits pēc kārtas, trešajā - pareizo, ceturtajā - nepareizo, utt.

Otrajā rindā dota naturāla skaitļa G (vienādo atbilžu grupu skaits, $G \le 10^5$) vērtība.

Trešajā rindā doti G nenegatīvi veseli skaitļi, kas nepārsniedz 10^9 - vienādo atbilžu skaits katrā grupā pēc kārtas, sākot ar pareizajām atbildēm. Vienīgais skaitlis, kas šajā rindā var būt 0, ir pirmais skaitlis, pārējie skaitļi ir pozitīvi.

Izvaddati

Vienīgajā rindā jāizvada N veseli nenegatīvi skaitļi, kur blakus skaitļi atdalīti ar tukšumzīmi. Katram n ($1 \le n \le N$) n-tajam skaitlim jābūt n-tā dalībnieka iegūto punktu skaitam.

Piemēri

Ievaddati	Izvaddati
3 2 11 0 1 3 6 5 1 1 1 1 1 3	4 7 6

1. apakšuzdevuma testu ievaddati

Ievaddati
8 2 **8** **0 13 12 7 9 11 13 16**

Ievaddati
8 2 26 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 3 1 4 1 1

Ievaddati
6 2 12 0 6 1 6 2 7 7 7 6 9 5 4

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Atbilžu kopskaits (skaitļu summa ievaddatu trešajā rindā) $\le 10^7$	10
3.	$N \le 100$	22
4.	Bez papildu ierobežojumiem	66
	Kopā:	100

2017./2018. mācību gads

Novada olimpiāde - 2018

Apakštaisnstūri

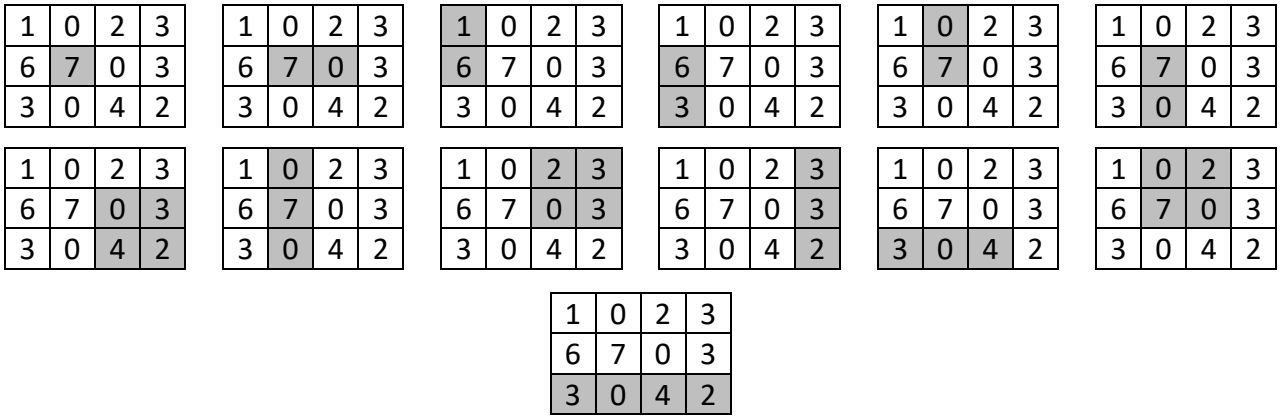
N×M rūtiņas liela tabula aizpildīta ar veseliem nenegatīviem skaitļiem. Nepieciešams noteikt, cik veidos šajā tabulā iespējams atrast taisnstūrveida rūtiņu apgabalu, kura rūtiņās ierakstīto skaitļu summa būtu vismaz A, bet nepārsniegtu B (A un B - veseli nenegatīvi skaitļi, A≤B). Taisnstūrveida apgabals var būt arī viena atsevišķa rūtiņa vai visa tabula.

Piemēram, ja A=7, B=9, un 3×4 rūtiņu tabula aizpildīta kā parādīts 10. zīmējumā,

1	0	2	3
6	7	0	3
3	0	4	2

10. zīm.

, tad taisnstūri var izvēlēties 13 veidos (skat. 11. zīm.)



11. zīm.

Uzrakstiet programmu, kas atrod aprakstīto taisnstūru skaitu!

Ievaddati

Pirmajā rindā dotas veselu nenegatīvu skaitļu N (tabulas rindu skaits, 1≤N≤300), M (tabulas kolonnu skaits, 1≤M≤300), A (mazākā pieļaujamā summas vērtība) un B (lielākā pieļaujamā summas vērtība, A≤B≤2×10^9). Katri divi blakus skaitļi ir atdalīti ar tukšumzīmi.

Nākamajās N rindās dots tabulas rūtiņās ierakstīto skaitļu apraksts.

Katrā no šīm rindām doti M veseli nenegatīvi skaitļi, kur katri divi blakus skaitļi atdalīti ar tukšumzīmi.

Katram n(1≤n≤N) un m(1≤m≤M) skaitlis ievaddatu n+1-ās rindas m-tais skaitlis pēc kārtas ir ierakstīts tabulas n-tās rindas m-tās kolonnas rūtiņā.

Zināms, ka tabulā ierakstīto skaitļu kopsumma nepārsniedz 2×10^9.

Izvaddati

Vienīgajā rindā jāizvada vesels nenegatīvs skaitlis - derīgo taisnstūrveida apgabalu skaits.

Piemēri

Ievaddati	Izvaddati
3 4 7 9 1 0 2 3 6 7 0 3 3 0 4 2	13

Ievaddati	Izvaddati
3 3 2 3 1 0 0 0 1 0 0 0 1	7

1. apakšuzdevuma testu ievaddati (jaunākajai grupai)

Ievaddati
3 8 310 310 3 34 69 33 1 34 69 3 36 2 68 35 66 67 69 35 2 1 3 3 34 36 3 66

Ievaddati
3 4 377 415 76 0 2 26 50 26 25 75 2 76 25 51

Ievaddati
8 6 1063 1592 82 1 1 21 0 21 60 41 40 42 60 1 1 0 0 42 40 22 60 80 61 2 21 42 40 40 41 61 0 22 21 20 42 41 42 40 60 1 61 0 20 80 82 21 60 41 41 62

Apakšuzdevumi un to vērtēšana (jaunākajai grupai)

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	2 ≤ N, M ≤ 30	26
3.	2 ≤ N, M ≤ 100	54
4.	Bez papildu ierobežojumiem	18
	Kopā:	100

1. apakšuzdevuma testu ievaddati (vecākajai grupai)

Ievaddati
4 5 120 415 3 68 67 68 2 3 68 35 34 35 2 34 34 34 35 68 34 3 69 66

Ievaddati
6 8 1272 2029 26 77 75 51 52 77 50 52 2 50 0 0 25 27 51 75 0 26 50 26 51 52 76 50 27 75 51 50 2 25 27 0 52 25 77 25 2 52 76 51 2 2 76 27 77 52 75 2

Ievaddati
3 4 37 75 0 82 21 61 21 1 21 40 21 80 82 82

Apakšuzdevumi un to vērtēšana (vecākajai grupai)

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	2 ≤ N, M ≤ 30	20
3.	2 ≤ N, M ≤ 100	33
4.	Bez papildu ierobežojumiem	45
	Kopā:	100

Ciparu aizvietošana

Ja naturāla skaitļa decimālajā pierakstā katru ciparu aizvieto ar tā bināro pierakstu, kā redzams 1. tabulā, tad iegūst nullīšu un vieninieku virkni.

Piemēram, no 431 iegūst 100111, bet no 1022 - 101010.

Var gadīties, ka vienu un to pašu nullīšu un vieninieku virkni var iegūt no vairākiem skaitļiem. Tā, virkni 100111 var iegūt no 14 skaitļiem: 100111, 10013, 10031, 1007, 20111, 2013, 2031, 207, 4111, 413, 431, 47, 911, 93. No šiem skaitļiem mazākais ir 47.

Uzrakstiet programmu, kas dotai nullīšu un vieninieku virknei nosaka dažādo skaitļu skaitu no kuras to var iegūt, kā arī atrod mazāko no šiem skaitļiem!

Decimālais cipars	Binārais pieraksts
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

1.tabula Ciparu aizvietošana

Ievaddati

Pirmajā rindā dots naturāls skaitlis G($G \le 10^5$) - nullīšu un vieninieku virknes garums. Otrajā rindā dota ciparu virkne garumā G, kas var saturēt tikai ciparus 0 un 1. Zināms, ka virkne sākas ar 1.

Izvaddati

Pirmajā rindā jāizvada naturāls skaitlis - naturālo skaitļu skaits, no kuriem iegūst ievaddatos doto virkni. Skaits jāizvada pēc moduļa 10^9+9 - t.i., ja dažādo skaitļu skaits ir S, tad jāizvada atlikums, kādu iegūst S dalot ar 1000000009.

Otrajā rindā jāizvada mazākā no šiem skaitļiem decimālais pieraksts.

Piemēri

Ievaddati	Izvaddati
6 100111	14 47

Ievaddati	Izvaddati
36 111111111111111111111111111111111111	82876085 777777777777

1. apakšuzdevuma testu ievaddati

Ievaddati
12 100111101100

Ievaddati
13 1110100001111

Ievaddati
8 10000001

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	G ≤ 8	8
3.	G ≤ 32	10
3.	G ≤ 1000, vieninieku skaits otrajā ievaddatu rindā nepārsniedz 8	14
4.	Bez papildu ierobežojumiem	66
	Kopā:	100

Nav apakšvirkne

Latviešu alfabētā ir 22 mazie burti bez diakritiskajām zīmēm:

a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, r, s, t, u, v, z.

Burtu virkne X ir veidota, izmantojot pirmos N no šiem burtiem (katru burtu var izmantot vairākkārt).

Nepieciešams noskaidrot, kāda ir īsākā burtu virkne, kas nav X apakšvirkne. Ja tādas virknes ir vairākas, tad interesēsimies par *leksikogrāfiski mazāko* no tām. No divām vienāda garuma virknēm leksikogrāfiski

mazākā ir tā, kurai pirmais atšķirīgais burts, skaitot no virknes sākuma, alfabētā atrodas pirms otrā virknē atbilstošajā pozīcijā esošā burta.

Piemēram, ja N=4 un X = “dabacbadac”, tad kā apakšvirknes ir atrodamas visas virknes, kas sastāv no burtiem ‘a’, ‘b’, ‘c’ un ‘d’ garumā 1 un 2, bet nav atrodamas trīsburtu virknes ”adb”, “add”, “bbb”, “bdb”, “bdd”, “cab”, “cbb”, “cca”, “ccb”, “ccc”, “ccd”, “cdb”, “cdd”, “ddb” un “ddd”. No visām šīm virknēm leksikogrāfiski mazākā ir virkne “adb”.

Uzrakstiet programmu, kas dotām N un X vērtībām atrisina šo uzdevumu!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N (alfabēta burtu skaits, N≤22) un burtu virknes garums G($G \leq 10^5$), kas atdalītas ar tukšumzīmēm.

Otrajā rindā dota virkne X - G burtu virkne bez atdalošajām tukšumzīmēm.

Izvaddati

Vienīgajā rindā jāizvada īsākā burtu virkne, kas nav X apakšvirkne. Ja ir vairākas virknes ar šo garumu, tad jāizvada leksikogrāfiski mazākā no tām.

Piemēri

Ievaddati	Izvaddati
4 10 dabacbadac	adb

Ievaddati	Izvaddati
3 2 ca	b

1. apakšuzdevuma testu ievaddati

Ievaddati
2 7 baaabab

Ievaddati
3 15 cbbcaabacbcaaca

Ievaddati
4 24 dbcbcaabdadcddabacbacbcd

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$G \leq 4N$	12
3.	$N = 2$	14
4.	Bez papildu ierobežojumiem	72
	Kopā:	100

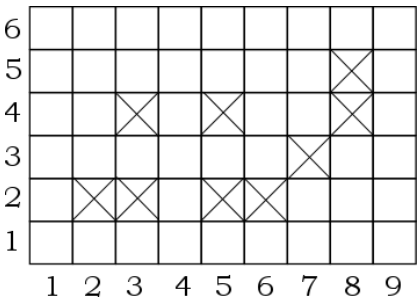
Peļķes

Milzīgs kādas pilsētas galvenais laukums ir noklāts ar kvadrātveida bruģa plāksnītēm. Laika gaitā, dažas plāksnītes ir sadrupušas un to vietās pēc lietus veidojas nesimpātiskas peļķes. Tuvojoties pilsētas svētkiem, pilsētas mērs ir uzdevis apzināt visas sadrupušo plāksnīšu vietas un izgatavot divas vienādas plāksnes, kuras uzklājot bojātajām vietām paralēli plāksnīšu malām, tās tiktu pilnībā noklātas. Plāksnes nedrīkst pārklāties. Līdzekļu taupīšanas dēļ, mērs vēlas, lai plākšņu laukums būtu mazākais iespējamais.

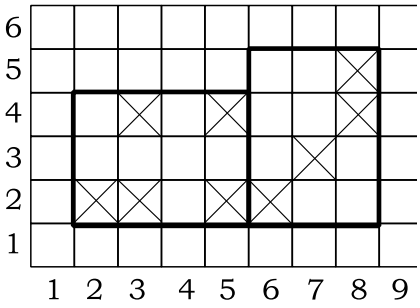
Pilsētas laukumu var attēlot kā rūtiņu laukumu, kur katra rūtiņa atbilst vienai bruģa plāksnītei.

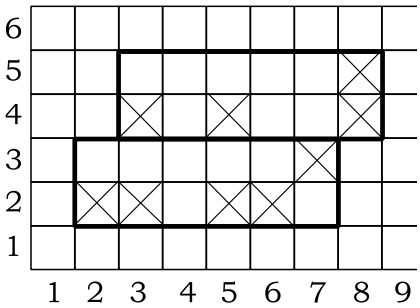
Piemēram, ja peļķes (bojātās plāksnītes) ir ar krustiņiem atzīmētajās vietās kā redzams 12. zīmējumā, tad mazākās plāksnes, kas der visu šo vietu noklāšanai ir 12 (3×4 vai 2×6) rūtiņas lielas (skat. 13. zīm.).

Uzrakstiet programmu, kas dotām peļķu koordinātām nosaka mazāko iespējamo plākšņu laukumu!



12. zīm. Peļķes.





13. zīm. Peļķu noklāšana.

Ievaddati

Pirmajā rindā dots bojāto plāksnīšu skaits - naturāls skaitlis N ($N \le 10^5$).

Katrā no nākamajām N rindām dotas vienas bojātās plāksnītes koordinātas - divi naturāli skaitļi robežās no 1 līdz 10^9, kas atdalīti ar tukšumzīmi - bojātās plāksnītes rindas un kolonnas numurs. Gan rindas, gan kolonnas ir numurētas pēc kārtas ar skaitļiem no 1 līdz 10^9 pēc kārtas.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis - mazākais tādas plāksnes laukums, lai ar divām vienādām šī izmēra plāksnēm varētu pārklāt visas bojātās plāksnītes.

Piemērs

Ievaddati	Izvaddati
9 2 2 4 3 5 8 3 7 2 6 4 5 2 5 2 3 4 8	12

1. apakšuzdevuma testu ievaddati

Ievaddati
3 1 4 3 9 4 2

Ievaddati
5 7 6 7 8 9 9 2 7 2 2

Ievaddati
10 81 90 3 96 33 62 39 24 5 90 73 70 75 8 29 66 91 20 55 80

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 50	8
3.	N ≤ 1000 un visas bojātās plāksnītes atrodas pirmajā kolonnā	10
4.	N ≤ 10^5 un visas bojātās plāksnītes atrodas pirmajā kolonnā	20
5.	N ≤ 1000	30
6.	Bez papildu ierobežojumiem	30
	Kopā:	100

Receptes

Slavenais konditors Mārtiņš Vakariņš ir savācis lielu toršu recepšu kolekciju. Pēdējā laikā Vakariņš ir pamanījis, ka ir grūti atcerēties vai atrast noteiktas tortes recepti, tādēļ ir izveidojis datorsistēmu, kurā ievadījis visas zināmās receptes, katrai no tortēm norādot nepieciešamo izejvielu daudzums kā N veselu nenegatīvu skaitļu virkni $a_1\ a_2\ \ldots . a_N$. Katram i (1≤i≤N) a_i norāda i-tās sastāvdaļas daudzumu noteiktās vienībās (piemēram, cukura vai miltu mērvienība būs izteikta gramos, olu dzeltenumi - gabalos, utt.). Ja konkrētai receptei attiecīgā sastāvdaļa nav nepieciešama, tad atbilstošā a_i vērtība ir 0.

Vakariņš vēlas izgatavot pēc iespējas vairāk toršu pēc vienas receptes un zina, cik katra veida izejvielas šobrīd ir noliktavā. Viņš vairākām receptēm vēlas noskaidrot, kādu lielāko daudzumu toršu iespējams izgatavot pēc katras receptes.

Piemēram, ja N=4 un izejvielu daudzumi ir tādi, kā norādīts 2. tabulā:

	1. izejviela	2. izejviela	3. izejviela	4. izejviela
Noliktavā	8	7	10	6
1.recepte	1	0	3	2
2.recepte	2	1	2	1
3.recepte	3	4	5	0

2. tabula Izejvielu daudzumi

, tad būs iespējams izgatavot vai nu trīs tortes pēc pirmās receptes, vai četras tortes pēc otrās receptes, vai arī vienu torti pēc trešās receptes.

Uzrakstiet programmu, kas dotam izejvielu daudzumam noliktavā un recepšu aprakstiem aprēķina toršu skaitu!

Ievaddati

Pirmajā rindā dotas divu naturālu skaitļu N (izejvielu veidu skaits, $N \le 10^5$) un R (dažādo recepšu skaits, $1 < R \le 10$) vērtības, kas atdalītas ar tukšumzīmi.

Nākamajā rindā doti N nenegatīvi veseli skaitļi, kur katri divi blakus skaitļi atdalīti ar tukšumzīmi. Katram i(1≤i≤N) i-tais skaitlis norāda i-tās izejvielas daudzumu noliktavā.

Nākamajās R rindās katrā dots vienas tortes receptes apraksts - tortes pagatavošanai nepieciešamais katra veida izejvielu daudzums.

Katrā rindā doti N nenegatīvi veseli skaitļi, kur katri divi blakus skaitļi atdalīti ar tukšumzīmi. Katram i(1≤i≤N) i-tais skaitlis norāda receptē nepieciešamo i-tās izejvielas daudzumu. Zināms, ka katras tortes receptei vismaz vienas izejvielas daudzums ir pozitīvs skaitlis.

Nevienai izejvielai tās daudzums noliktavā vai receptē nepārsniedz 10^9 vienību.

Izvaddati

Izvaddatiem jāsatur tieši R rindas. Katram r(1≤r≤R) r-tajā rindā jāizvada vesels nenegatīvs skaitlis - lielākais toršu skaits, kādu iespējams izgatavot pēc šīs receptes.

Piemēri

Ievaddati	Izvaddati
4 3 8 7 10 6 1 0 3 2 2 1 2 1 3 4 5 0	3 4 1

Ievaddati	Izvaddati
5 2 6 4 0 3 5 1 1 1 1 1 1 1 0 1 2	0 2

1. apakšuzdevuma testu ievaddati

Ievaddati
4 2 251 433 109 77 11 7 8 5 5 2 6 3

Ievaddati
4 2 999 999 999 999 8 7 6 5 11 12 13 14

Ievaddati
4 2 765 432 987 789 8 7 9 8 76 43 98 78

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 3	10
3.	Nevienai izejvielai tās daudzums noliktavā vai receptē nepārsniedz 1000	28
4.	Bez papildu ierobežojumiem	60
	Kopā:	100

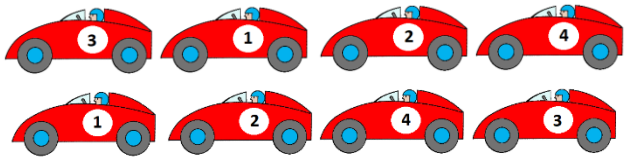
Valsts olimpiāde - 2018

Autosacīkstes-1

Autosacīkstēs piedalās N automašīnas, kas sanumurētas ar naturāliem skaitļiem no 1 līdz N pēc kārtas. Automašīnas pa šauru ceļu joņo viena aiz otras, veikdamas apdzīšanas retajās vietās, kur tas ir iespējams. "Apdzīšana" nozīmē, ka viena automašīna, kas atradās tieši aiz citas, aizbrauca tai garām - katras apdzīšanas rezultātā tieši divas blakus mašīnas samainās vietām.

Tālivaldis ir nokavējis sacīkšu sākumu un tagad ar interesi seko sacīkstēm, uzmanīgi pierakstot visas apdzīšanas pēc kārtas. Tālivaldis nekļūdās - viņa pierakstos visas apdzīšanas ir pierakstītas pareizā secībā, nav lieku vai kļūdaini pierakstītu apdzīšanu. Kāda ir bijusi automašīnu secība, pirms Tālivaldis sācis veikt pierakstus, nav zināms.

Piemēram, ja N=4, 1. auto apdzinis 4. un 2. auto apdzinis 4., tad iespējamās divas automašīnu secības pēc šīm apdzīšanām ir parādītas 14. zīmējumā.



14. zīm.

Uzrakstiet programmu, kas atrod vienu iespējamo automašīnu secību pēc Tālivalža novērotajām apdzīšanām!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N (automašīnu skaits, $N \le 10^5$) un A (pierakstīto apdzīšanu skaits, $A \le 10^5$) vērtības, kas atdalītas ar tukšumzīmi. Ievaddatu nākamajās A rindās doti apdzīšanu apraksti to pierakstīšanas secībā. Katru apdzīšanu apraksta divi atšķirīgi naturāli skaitļi robežās no 1 līdz N, kas atdalīti ar tukšumzīmi. Pirmais skaitlis ir tās automašīnas numurs, kas veica apdzīšanu, bet otrais - tās mašīnas numurs, kas tika apdzīta.

Izvaddati

Vienīgajā rindā jāizvada N naturāli skaitļi - automašīnu numuri tādā secībā, kādā tie var būt pēc Tālivalža novērotajām apdzīšanām. Starp katriem diviem blakus skaitļiem ievaddatos jābūt tukšumzīmei. Ja iespējamas vairākas automašīnu secības, tad jāizvada jebkura no tām.

Piemēri

Ievaddati	Izvaddati
4 2 1 4 2 4	3 1 2 4

Ievaddati	Izvaddati
5 2 2 3 1 4	1 4 5 2 3

1. apakšuzdevuma testu ievaddati

Ievaddati
13 13 5 8 8 5 12 9 1 6 5 8 6 1 8 5 9 12 4 11 11 4 1 6 13 7 13 12

Ievaddati
8 10 7 8 8 7 7 8 2 8 2 7 8 7 8 2 5 4 1 6 6 1

Ievaddati
11 21 10 11 11 10 10 11 11 10 10 11 11 10 10 11 4 2 8 11 8 10 5 6 11 10 6 5 2 4 11 8 4 2 2 4 4 2 8 11 4 5 2 5

Apakšuzdevumi un to vērtēšana

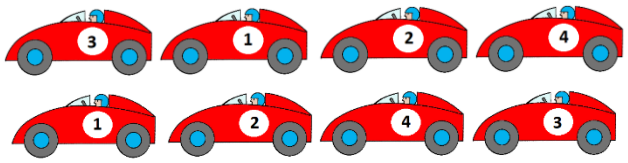
Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$N \le 10$	8
3.	$N \le 1000$, $A \le 1000$	10
4.	$A \le 1000$	20
5.	Bez papildu ierobežojumiem	60
	Kopā:	100

Autosacīkstes-2

Autosacīkstēs piedalās N automašīnas, kas sanumurētas ar naturāliem skaitļiem no 1 līdz N pēc kārtas. Automašīnas pa šauru ceļu joņo viena aiz otras, veikdamas apdzīšanas retajās vietās, kur tas ir iespējams. "Apdzīšana" nozīmē, ka viena automašīna, kas atradās tieši aiz citas, aizbrauca tai garām - katras apdzīšanas rezultātā tieši divas blakus mašīnas samainās vietām.

Ferdinands ir nokavējis sacīkšu sākumu un tagad ar interesi seko sacīkstēm, pierakstot visas apdzīšanas pēc kārtas. Kāda ir bijusi automašīnu secība, pirms Ferdinands sācis veikt pierakstus, nav zināms.

Piemēram, ja N=4, 1. auto apdzinis 4. un 2. auto apdzinis 4., tad iespējamās divas automašīnu secības pēc šīm apdzīšanām ir parādītas 15. zīmējumā.



15. zīm.

Diemžēl, savos pierakstos Ferdinands var būt kļūdījies un kādu apdzīšanu izlaidis vai ierakstījis nepareizi.

Piemēram, ja N=5 un ierakstīts, ka 2. auto apdzinis 3. un 3. auto apdzinis 4., tad šīs apdzīšanas nevar sekot tieši viena aiz otras -- pēc pirmās apdzīšanas 3. auto atrodas tieši aiz 2. auto un nevar apdzīt 4. auto. Tātad šajā gadījumā nav iespējama neviena automašīnu secība.

Uzrakstiet programmu, kas aprēķina dažādo iespējamo automašīnu secību pēc Ferdinanda novērotajām apdzīšanām!

Ievaddati

Ievaddatu pirmajā rindā dotas naturālu skaitļu N (automašīnu skaits, $N \le 10^5$) un A (pierakstīto apdzīšanu skaits, $A \le 10^5$) vērtības, kas atdalītas ar tukšumzīmi. Ievaddatu nākamajās A rindās doti apdzīšanu apraksti tādā secībā, kā tās pierakstījis Ferdinands. Katru apdzīšanu apraksta divi atšķirīgi naturāli skaitļi robežās no 1 līdz N, kas atdalīti ar tukšumzīmi. Pirmais skaitlis ir tās automašīnas numurs, kas veica apdzīšanu, bet otrais - tās mašīnas numurs, kas tika apdzīta.

Izvaddati

Vienīgajā rindā jāizvada vesels nenegatīvs skaitlis - dažādo iespējamo automašīnu secību pēc Ferdinanda novērotajām apdzīšanām skaits pēc moduļa 10^9+9.

Piemēri

Ievaddati	Izvaddati
4 2 1 4 2 4	2

Ievaddati	Izvaddati
5 2 2 3 3 4	0

Ievaddati	Izvaddati
100 1 100 1	863172927

1. apakšuzdevuma testu ievaddati

Ievaddati
8 12 2 4 4 2 2 4 1 3 1 4 1 2 6 3 4 2 6 2 3 2 2 3 2 6

Ievaddati
15 29 1 13 13 1 1 13 13 1 1 13 13 1 5 7 1 13 7 5 13 1 1 13 13 1 1 13 13 1 5 7 2 12 7 5 1 13 13 1 12 2 3 4 5 7 1 13 7 5 11 14 2 12 14 11 5 7 6 8

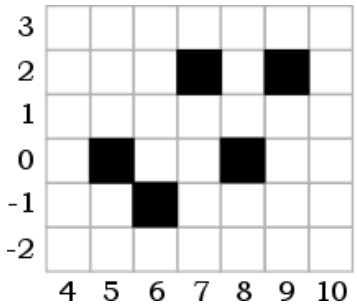
Ievaddati
13 7 1 11 10 4 10 12 13 2 13 8 4 12 7 11

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 10	8
3.	N ≤ 1000, A ≤ 1000	10
4.	A ≤ 1000	20
5.	Bez papildu ierobežojumiem	60
	Kopā:	100

Augļudārza platība

Augļudārzā ietilpstošie N koki plānā atzīmēti kā rūtiņas, kur vienas rūtiņas malai dabā atbilst divi metri. Dārza plāna piemērs parādīts 16. zīmējumā.



16. zīm. Augļudārza plāns

Dārzam ir četri īpašnieki, un katram no tiem ir atšķirīgs skatījums, kādai jābūt žoga formai un orientācijai dabā.

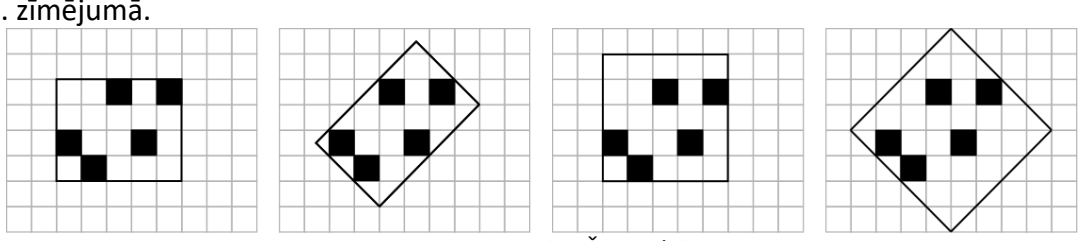
Pirmais īpašnieks uzskata, ka žogam jābūt taisnstūrveida un tā malām jābūt paralēlām rūtiņu režģa malām. Arī otrais īpašnieks uzskata, ka žogam jābūt taisnstūrveida, bet tā malām ar rūtiņu režģa malām jāveido 45° liels leņķis.

Trešais īpašnieks uzskata, ka žogam jābūt kvadrātveida un tā malām jābūt paralēlām rūtiņu režģa malām. Arī ceturtais īpašnieks uzskata, ka žogam jābūt kvadrātveida, bet tā malām ar rūtiņu režģa malām jāveido 45° liels leņķis.

Plānā žogs drīkst iet caur rūtiņu, kurās atrodas koki, virsotnēm vai malām.

Visi īpašnieki ir vienisprātis, ka žoga ierobežotajai teritorijai jābūt pēc iespējas mazākai.

16. zīmējumam atbilstošajam plānam pēc katra īpašnieka vēlmēm izveidoto žogu plāni parādīti 17. zīmējumā.


17. zīm. Žoga izbūves varianti.

Pēc pirmā un otrā īpašnieka plāna žoga iekšpusē būtu 80 m^2, pēc trešā - 100 m^2, pēc ceturtā - 128 m^2.

Uzrakstiet programmu, kas dotām koku atrašanās vietu koordinātām aprēķina žoga ierobežotās teritorijas lielumu kvadrātmetros katra īpašnieka iecerētajā variantā!

Ievaddati

Ievaddatu pirmajā rindā dota naturāla skaitļa N (koku skaits augļudārzā, $N \le 10^5$). Katrā no nākamajām N ievaddatu rindām doti divi veseli skaitļi, kas atdalīti ar tukšumzīmi - viena koka atrašanās rūtiņu režģī koordinātas - kolonnas numurs x un rindas numurs y.

Zināms, ka $-5\ 10^8 \le x, y \le 5\ 10^8$.

Izvaddati

Izvaddatos jābūt četrām rindām, katrā no kurām jāatrodas naturālam skaitlim. Katram i ($1 \le i \le 4$) izvaddatu i-tajā rindā jāizvada žoga iekšpusē esošās teritorijas lielums kvadrātmetros, ja žogs tiktu uzcelts pēc i-tā īpašnieka ieceres.

Piemērs (atbilst tekstā dotajam zīmējumam)

Ievaddati	Izvaddati
5 7 2 8 0 6 -1 9 2 5 0	80 80 100 128

1. apakšuzdevuma testa ievaddati

Ievaddati
5 12 -18 -19 7 -15 -18 -13 0 2 -11

Apakšuzdevumi un vērtēšana

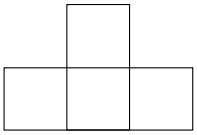
Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotais tests	2
2.	Bez papildu ierobežojumiem	98
	Kopā:	100

Ja pareizi būs aprēķināta dārza platība atsevišķam īpašniekam, tad kopumā varēs iegūt:

Īpašnieks	Punkti
1	20
2	40
3	15
4	25

Figūras-1

Armands interesējas par plaknes figūrām, ko veido saistītu rūtiņu apgabals. Katras figūras *aprakstu* veido kā rūtiņu apstaigāšanas virkni:

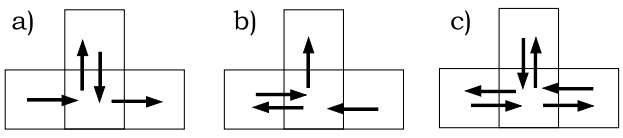


18. zīm.. Figūras piemērs

- izvēlas kādu figūras rūtiņu, no kuras sākt figūras rūtiņu apstaigāšanu,
- norāda, kādā virzienā (←,↑,→ vai ↓) jādodas uz nākamo rūtiņu,
- apstaigāšanu turpina tikmēr, līdz visās figūras rūtiņas ir apciemotas vismaz vienreiz.

Rūtiņu apstaigāšana var sākties un beigties jebkurā rūtiņā. Vienai un tai pašai figūrai var būt atšķirīgi apraksti.

Piemēram, 18. zīmējumā redzamajai figūrai iespējamas vairākas rūtiņu apstaigāšanas virknes - dažas virknes un tām atbilstošie apraksti ir parādīti 19. zīmējumā.



19. zīm. Figūras apstaigāšanas virkņu piemēri:
a) →, ↑, ↓, →; b) ←, ←, →, ↑; c) ←, →, →, ←, ↑, ↓.

Ja viena no vienādām figūrām ir pagriezta un/vai atspoguļota attiecībā pret otru, tad uzskatīsim, ka šīs figūras ir atšķirīgas.

Uzrakstiet programmu, kas vairākiem dotiem figūru aprakstiem atrod visas savā starpā atšķirīgās figūras!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (figūru skaits, 2≤N≤40) vērtība.

Nākamajās N ievaddatu rindās doti figūru apstaigāšanas virkņu apraksti - pa vienam katrā rindā. Katram i(1≤i≤N) i-tās figūras apraksts dots ievaddatu i+1-ajā rindā.

Katrs apraksts ir ciparu virkne bez atdalošiem tukšumiem. “1” atbilst iešanai pa kreisi (←), “2” - uz augšu (↑), “3” - pa labi (→), “4” - uz leju (↓). Katras figūras apraksta beigas norāda cipars “0”.

Zināms, ka nevienas figūras apraksta garums nepārsniedz 50000 simbolus.

Izvaddati

Izvaddatiem jāsatur tieši N rindas. Katram i(1≤i≤N) i-tajā rindā jāizvada vai nu tikai skaitlis i, ja šī figūra nav vienāda ar nevienu no figūrām ar mazāku numuru, vai arī skaitlis i un mazākais figūras numurs ar kuru i-tā figūra ir vienāda. Starp blakus skaitļiem izvaddatos jāizvada tukšumzīme.

Piemērs

Ievaddati	Izvaddati	Piezīme
7 32430 11320 0 4110 1331240 3320 444210	1 2 1 3 4 5 1 6 4 7	Apraksts "0" nozīmē, ka figūra sastāv no vienas rūtiņas. Septītā figūra nav vienāda ar pirmo, jo ir pagriezta.

1. apakšuzdevuma testu ievaddati

Ievaddati
8 223213414410 3432340 1444132230 413223210 432414410 2144410 433213411210 23432412110

Ievaddati
15 3410 3410 41210 1230 2340 4320 4120 4120 1230 1230 2340 2140 4120 4320 1430

Ievaddati
11 4112223442214140 214431222322110 1434133321230 122321322113344440 211233344221114430 3211414333111220 4332334132111210 1334414341430 1112321333410 2343341113332210 32342112234214140

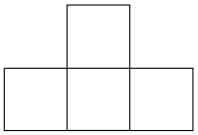
Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 12, katras figūras apraksta garums ≤ 100	16
3.	Katras figūras apraksta garums ≤ 2000	30
4.	Bez papildu ierobežojumiem	52
	Kopā:	100

Figūras-2

Armands interesējas par plaknes figūrām, ko veido saistītu rūtiņu apgabals. Katras figūras *aprakstu* veido kā rūtiņu apstaigāšanas virkni:

- izvēlas kādu figūras rūtiņu, no kuras sākt figūras rūtiņu apstaigāšanu,
- norāda, kādā virzienā (←,↑,→ vai ↓) jādodas uz nākamo rūtiņu,
- apstaigāšanu turpina tikmēr, līdz visās figūras rūtiņas ir apciemotas vismaz vienreiz.



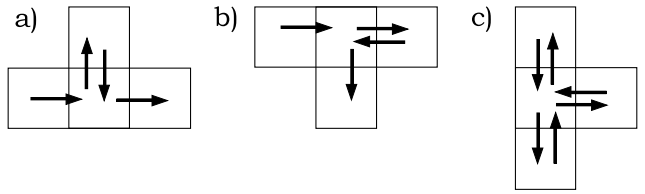
20. zīm.*Figūras piemērs*

Rūtiņu apstaigāšana var sākties un beigties jebkurā rūtiņā. Vienai un tai pašai figūrai var būt atšķirīgi apraksti.

Piemēram, 20. zīmējumā redzamajai figūrai iespējamas vairākas rūtiņu apstaigāšanas virknes - dažas virknes un tām atbilstošie apraksti ir parādīti 21. zīmējumā.

Ja viena no vienādām figūrām ir pagriezta un/vai atspoguļota attiecībā pret otru, tad uzskatīsim, ka šīs figūras ir vienādas.

Uzrakstiet programmu, kas vairākiem dotiem figūru aprakstiem atrod visas savā starpā atšķirīgās figūras!



21. zīm. Figūras apstaigāšanas virkņu piemēri:

a) →, ↑, ↓, →; b) →, →, ←, ↓; c) →, ←, ↑, ↓, ↓, ↑.

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (figūru skaits, 2≤N≤40) vērtība.

Nākamajās N ievaddatu rindās doti figūru apstaigāšanas virkņu apraksti - pa vienam katrā rindā. Katram i(1≤i≤N) i-tās figūras apraksts dots ievaddatu i+1-ajā rindā.

Katrs apraksts ir ciparu virkne bez atdalošiem tukšumiem. “1” atbilst iešanai pa kreisi (←), “2” - uz augšu (↑), “3” - pa labi (→), “4” - uz leju (↓). Katras figūras apraksta beigas norāda cipars “0”.

Zināms, ka nevienas figūras apraksta garums nepārsniedz 50000 simbolus.

Izvaddati

Izvaddatiem jāsatur tieši N rindas. Katram i(1≤i≤N) i-tajā rindā jāizvada vai nu tikai skaitlis i, ja šī figūra nav vienāda ar nevienu no figūrām ar mazāku numuru, vai arī skaitlis i un mazākais figūras numurs, ar kuru i-tā figūra ir vienāda. Starp blakus skaitļiem izvaddatos jāizvada tukšumzīme.

Piemērs

Ievaddati	Izvaddati	Piezīme
6 32430 33140 0 3320 3124420 4110	1 2 1 3 4 5 1 6 4	Apraksts “0” nozīmē, ka figūra sastāv no vienas rūtiņas.

1. apakšuzdevuma testu ievaddati

Ievaddati
8 32214234440 2214440 442222340 2343121140 12321341440 343241132110 432414440 4442223210

Ievaddati
15 22340 213340 243410 123110 14312330 3410 122430 4320 4324140 2142320 12330 1231410 244120 343110 14340

Ievaddati
11 1322222331421140 32342233331111134110 141112331141333330 1431124322333330 34113333213340 2221233421113434444430 412231244443410 32333341110 23431211114330 234434112433212220 234414232333320

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 12, katras figūras apraksta garums ≤ 100	16
3.	Katras figūras apraksta garums ≤ 2000	30
4.	Bez papildu ierobežojumiem	52
	Kopā:	100

Minibači

Par *Minibači virkni* sauksim tādu augošu naturālu skaitļu virkni, kuras pirmie divi locekļi a_1 un a_2 ($a_1<a_2$) ir doti, bet katrs nākamais ir vienāds ar vairāku secīgu iepriekšējo (ne obligāti pēdējo) elementu summu, turklāt katrs no šiem locekļiem ir mazākais iespējamais.

Piemēram, ja $a_1=2$ un $a_2=3$, tad virknes pirmie locekļi ir 2, 3, 5(=2+3), 8(=3+5), 10(=2+3+5), 13(=5+8), 16(=3+5+8), 18(=2+3+5+8=8+10), 23(=5+8+10=10+13), 26(=3+5+8+10), 28(=2+3+5+8+10), 29(=13+16) ,....

Uzrakstiet programmu, kas dotām a_1, a_2 un N vērtībām atrod a_N vērtību!

Ievaddati

Vienīgajā rindā dotas naturālu skaitļu a_1, a_2 ($a_1<a_2<1000$) un N ($N \le 10^5$) vērtības, kas atdalītas ar tukšumzīmēm.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis - a_N vērtība.

Piemēri

Ievaddati	Izvaddati
2 3 12	29

Ievaddati	Izvaddati
1 6 5	14

1. apakšuzdevuma testu ievaddati

Ievaddati
2 3 19

Ievaddati
1 6 11

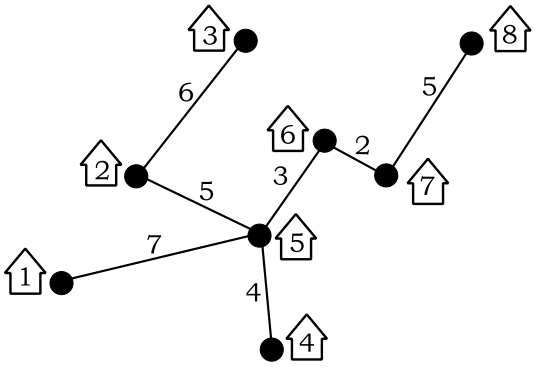
Ievaddati
7 17 17

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 10	10
3.	10 < N ≤ 5000	38
4.	Bez papildu ierobežojumiem	50
	Kopā:	100

Pārvākšanās

Rūķu ciemā dzīvo N rūķi, kas sanumurēti ar naturāliem skaitļiem no 1 līdz N pēc kārtas. Katrs rūķis dzīvo savā mājiņā, kas atrodas kādā ceļu krustojumā vai ceļa galā. No katras mājiņas līdz jebkurai citai iespējams aiziet tikai vienā vienīgā veidā. Katrs ceļš rūķu ciemā ir taisnes nogrieznis, kura garums rūķu garuma vienībās ir naturāls skaitlis.



22. zīm. *Rūķu ciema piemērs.*

Rūķiem ir apnicis dzīvot savās mājiņās un viņi ir nolēmuši visi vienlaicīgi veikt pārvākšanos uz kādu citu mājiņu tā, lai kopējais visu rūķu veiktais attālums starp veco un jauno mājiņu būtu mazākais iespējamais.

Piemēram, 22. zīmējumā redzamajā ciema kartē mazākais kopējais attālums starp vecajām un jaunajām mājiņām ir 48 rūķu garuma vienības. Tas ir iespējams, ja rūķi pārvācas, piemēram, šādi: 1→5, 2→3, 3→2, 4→1, 5→4, 6→7, 7→8, 8→6.

Uzrakstiet programmu, kas dotam rūķu ciema ceļu aprakstam nosaka mazāko kopējo attālumu starp vecajām un jaunajām mājiņām!

Ievaddati

Pirmajā rindā dots rūķu skaits - naturāls skaitlis N ($2 \le N \le 10^5$).

Nākamajā N-1 ievaddatu rindā dots rūķu ciema ceļu apraksts - pa vienam ceļam katrā rindā. Katra ceļa aprakstu veido trīs naturāli skaitļi: rūķu, kuru mājiņas atrodas ceļa galos, numuri un ceļa garums rūķu garuma vienībās. Starp katriem diviem blakus skaitļiem ir viena tukšumzīme.

Zināms, ka neviena ceļa garums nepārsniedz 10^9 rūķu garuma vienībās.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis - mazākais kopējo attālumu starp vecajām un jaunajām mājiņām rūķu garuma vienībās.

Piemēri

Ievaddati	Izvaddati
8 1 5 7 6 7 2 8 7 5 5 4 4 5 6 3 2 3 6 5 2 5	48

Ievaddati	Izvaddati
2 1 2 5	10

1. apakšuzdevuma testu ievaddati

Ievaddati
12 2 3 1 9 5 1 9 1 2 10 4 1 9 10 3 11 4 2 9 7 3 7 3 3 12 6 2 9 6 1 4 8 2

Ievaddati
9 5 7 9 2 5 5 7 3 5 8 1 1 3 9 3 6 5 3 5 4 7 4 8 6

Ievaddati
14 7 13 1 2 9 1 1 12 1 10 9 1 3 8 1 2 1 1 6 2 1 8 5 1 11 7 1 1 4 1 7 10 1 6 14 1 10 8 1

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 12	10
3.	12 < N ≤ 1000	16
4.	Visi ceļi ir vienu vienību gari	24
5.	Bez papildu ierobežojumiem	48
	Kopā:	100

Svara rāvējslēdzējs

Kādā pilsētas krustojumā ar intensīvu satiksmi saplūst divas vienvirziena ielas - *kreisā* un *labā*, kas turpinās kā viena vienvirziena iela. Šajā krustojumā darbojas īpašs - *svara rāvējslēdzēja* princips: kā kārtējā brauc automašīna no tās ielas, no kuras līdz šim gada laikā (skaitot no 1. janvāra plkst. 0.00) izbraukušo automašīnu kopējais svars, skaitot veselos kilogramos, ir mazāks. Ja no abām ielām izbraukušo automašīnu kopējais svars ir vienāds, tad nākamā automašīna brauc no kreisās ielas. Tāpēc arī kā pirmā gada sākumā (kad no katras ielas izbraukušo automašīnu kopējais svars ir 0 kg) izbrauc automašīna no kreisās ielas.

Ilggadīgi novērojumi rāda, ka automašīnu svaru secība katrā ielā cikliski atkārtojas, Piemēram, ja kreisajā ielā automašīnu svaru secība ir 21, 17, 5, 5, 11, bet labā - 18, 5, 15, 3, tad tās krustojumu izbrauks šādā secībā (pasvītrotās brauc no kreisās, bet nepasvītrotās - no labās):

21, 18, 5, 17, 15, 5, 3, 18, 5, 11, 21, 5, 15, 3, 17, …

Uzrakstiet programmu, kas nosaka, no kuras puses izbrauca un cik svēra automašīna, kas izbrauca krustojumam kā i-tā pēc kārtas!

Ievaddati

Ievaddatu pirmajā rindā dotas naturālu skaitļu K (automašīnu skaits kreisās ielas ciklā, $K \le 10^5$), L (automašīnu skaits labās ielas ciklā, $L \le 10^5$) un V (vaicājumu skaits, $V \le 10000$) vērtības. Ievaddatu otrajā rindā doti K naturāli skaitļi - to automašīnu svari kilogramos, kas krustojumā iebrauc no kreisās ielas. Ievaddatu trešajā rindā doti L naturāli skaitļi - to automašīnu svari kilogramos, kas krustojumā iebrauc no labās ielas. Zināms, ka nevienas automašīnas svars kilogramos nepārsniedz 10^5.

Tālāk ievaddatos seko V rindas ar vaicājumiem. Katram i($1 \le i \le V$) i-tais vaicājums v_i ir dots ievaddatu 3+i-tajā rindā un ir naturāls skaitlis - krustojumu izbraukušās automašīnas kārtas numurs. Vaicājumos

automašīnu kārtas numuri ir doti augošā secībā un nevienam vaicājumam automašīnas kārtas numurs nepārsniedz $2 \cdot 10^9$. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatos jābūt tieši V rindām. Katram i($1 \le i \le V$) izvaddatu i-tajā rindā jāizvada atbilde uz vaicājumu v_i - burts 'K' (ja vi-tā automašīna izbrauca no kreisās ielas) vai burts 'L' (ja v_i-tā automašīna izbrauca no labās ielas), kam seko tukšumzīme un naturāls skaitlis - šīs automašīnas svars kilogramos.

Piemēri

Ievaddati	Izvaddati
5 4 3 21 17 5 5 11 18 5 15 3 1 5 10	K 21 L 15 K 11

Ievaddati	Izvaddati
1 2 2 1 1 1 100 777	L 1 K 1

1. apakšuzdevuma testu ievaddati

Ievaddati
1 1 4 3 5 6 16 61 100

Ievaddati
3 5 6 10 20 30 1 2 3 4 5 7 17 27 37 47 57

Ievaddati
4 3 6 3 2 10 6 8 4 2 5 8 13 21 34 55

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Visiem i $v_i \le 10^6$	30
3.	Bez papildu ierobežojumiem	68
	Kopā:	100

Teniss

Tenisists Ernests sezonas laikā piedalās vienā vai vairākos tenisa turnīros. Katrs turnīrs sastāv no N izslēgšanas kārtām un katrā turnīrā piedalās tieši 2^N spēlētāji.

Visi spēlētāji turnīru sāk no pirmās kārtas. Katras kārtas zaudētājs no turnīra izstājas, bet uzvarētājs turnīru turpina nākamajā kārtā (vai arī beidz to kā uzvarētājs, ja tā bija pēdējā kārta).

Par piedalīšanos turnīrā dalībnieks saņem vienu punktu, bet par katru uzvaru vēl punktus papildus. Par uzvaru pirmajā kārtā dalībnieks saņem divus punktus, par uzvaru otrajā - četrus, par uzvaru trešajā - astoņus, …, katrā kārtā par uzvaru divreiz vairāk punktus kā iepriekšējā. Jeb, citiem vārdiem, katram i=1,…,N dalībnieks par uzvaru i-tajā kārtā saņem papildus 2^i punktus.

Ir zināms, cik spēlēs Ernests ir piedalījies sezonas laikā un ir zināma Ernesta zaudējumu un uzvaru virkne tādā secībā, kā spēles spēlētas.

Piemēram, ja N=7 un Ernests piedalījies 16 spēlēs, kuru rezultāti ir `uzuuzzuuuuuuuuuz` (ar "u" tiek apzīmēta uzvara, bet ar "z" - zaudējums), tad Ernests ir piedalījies piecos turnīros un nopelnījis 273 punktus (pa turnīriem: 3, 7, 1, 255, 7).

Uzrakstiet programmu, kas aprēķina, cik turnīros Ernests ir piedalījies un cik punktus ir nopelnījis sezonas laikā!

Ievaddati

Ievaddatu pirmajā rindā dotas naturālu skaitļu N (katra turnīra kārtu skaits, N ≤ 50) un K (spēļu skaits, K ≤ 10^5) vērtības, kas atdalītas ar tukšumzīmi. Otrajā rindā dota K burtu virkne bez atdalošajām tukšumzīmēm. Katram k(1≤k≤K) k-tais burts virknē ir 'u', ja sezonas k-to spēli pēc kārtas Ernests uzvarēja, vai 'z' - ja zaudēja.

Izvaddati

Vienīgajā rindā jāizvada divi naturāli skaitļi, kas atdalīti ar tukšumzīmi - turnīru skaits, kuros Ernests sezonas laikā ir piedalījies, un tajos nopelnīto punktu skaits.

Piemēri

Ievaddati	Izvaddati
7 16 uzuuzzuuuuuuuuuz	5 273

Ievaddati	Izvaddati
4 4 zzzz	4 4

1. apakšuzdevuma testu ievaddati

Ievaddati
4 13 uzuuzuuuzuuuu

Ievaddati
5 13 zzuuuuuzuuuuu

Ievaddati
3 13 uuzuuzuzuzuuz

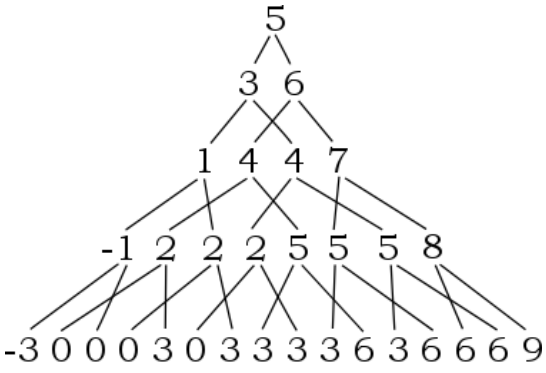
Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 4	11
3.	4 < N ≤ 17	34
4.	Bez papildu ierobežojumiem	53
	Kopā:	100

Skaitļu tornis

Skaitļu torni veido šādi: Sākumā izvēlas trīs veselus skaitļus n, a un b (starp tiem var būt arī vairāki vienādi skaitļi). Skaitli n uzraksta lapas augšpusē un zem tā blakus uzraksta skaitļus n+a un n+b, kas veido torņa pirmo rindu.

Tālāk procesu turpina veidojot otro rindu. Vispirms katram pirmās rindas skaitlim s_i zem tā pa kreisi uzraksta skaitli $s_i + a$ un pa labi - skaitli $s_i + b$. Pēc tam, neiesaistot pirmo un pēdējo no rindā uzrakstītajiem skaitļiem, skaitļus rindā secīgi sadala pa pāriem un katrā pārī skaitļus samaina vietām - t.i., otro skaitli samaina ar trešo, ceturto - ar piekto, utt. Pēc tam tāpat veido nākamās rindas. Skaitļu tornis, kuram n=5, a=-2, b=1 parādīts 23. zīmējumā.



23. zīm.

Uzrakstiet programmu, kas nosaka, kāds skaitlis atrodas noteiktā torņa vietā!

Ievaddati

Ievaddatu pirmajā rindā dotas veselu skaitļu n ($-10^5 \le n \le 10^5$), a ($-10^5 \le a \le 10^5$), un b ($-10^5 \le b \le 10^5$) vērtības. Ievaddatu otrajā rindā dota naturāla skaitļa v (vaicājumu skaits, $v \le 10^5$) vērtība. Katrā no nākamajām v ievaddatu rindām doti divi naturāli skaitļi rindas numurs r($r \le 10^9$) un kārtas numurs k ($k \le \min(2^r, 2^{63} - 1)$) rindā r.

Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatos jābūt tieši v rindām. Katram i ($1 \le i \le v$) izvaddatu i-tajā rindā jāizvada atbilde uz i-to vaicājumu ievaddatos pēc kārtas - skaitli, kas atrodas norādītās rindas norādītajā vietā.

Piemēri

Ievaddati	Izvaddati
5 -2 1 4 4 5 4 6 2 4 3 1	3 0 7 -1

Ievaddati	Izvaddati
0 1 1 3 11 1 21 273 33 5000	11 21 33

1. apakšuzdevuma testu ievaddati

Ievaddati
0 2 -2 5 5 20 2 1 6 43 4 6 6 50

Ievaddati
117 14 22 2 4 11 2 1

Ievaddati
-11 10 100 5 2 4 4 5 4 1 5 10 4 2

Apakšuzdevumi un to vērtēšana

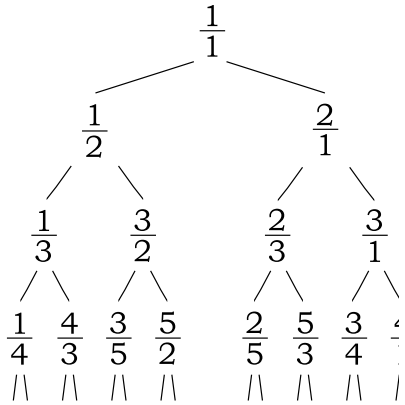
Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$r \le 16$	14
3.	$r \le 63$	14
4.	$r \le 10^4$, $v \le 1000$	14
5.	Bez papildu ierobežojumiem	56
	Kopā:	100

2018./2019. mācību gads

Novada olimpiāde - 2019

Attālums kokā

Datorzinātniekiem ir labi pazīstams Kalkina-Vilfa koks, kurā sakārtoti visi racionālie pozitīvie skaitļi. Šī binārā koka saknē atrodas skaitlis 1, bet katrs racionāls skaitlis kā nesaīsināma daļa $\frac{x}{y}$ atrodas kādā šī koka virsotnē un tās divās bērnu virsotnēs atrodas skaitļi $\frac{x}{x+y}$ un $\frac{x+y}{y}$. Katrs no racionāliem pozitīviem skaitļiem šajā kokā ir atrodams vienreiz. Koka sākuma fragments parādīts 24. zīmējumā.



24. zīm. Kalkina - Vilfa koka sākuma fragments.

Attālumu starp divām daļām definē kā attālumu starp atbilstošajām virsotnēm Kalkina - Vilfa kokā. Piemēram, attālums starp $\frac{1}{4}$ un $\frac{15}{6}$ (jeb $\frac{5}{2}$) ir 4, bet starp $\frac{9}{3}$ (jeb $\frac{3}{1}$) un $\frac{9}{15}$ (jeb $\frac{3}{5}$) ir 5.

Uzrakstiet programmu, kas nosaka attālumu starp diviem dotiem racionāliem skaitļiem!

Ievaddati

Pirmajā rindā dotas četru naturālu skaitļu P, Q, R, S ($P, Q, R, S \le 4 \times 10^{18}$) vērtības, kas atdalītas ar tukšumzīmēm.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada vesels nenegatīvs skaitlis - attālums starp skaitļiem $\frac{P}{Q}$ un $\frac{R}{S}$.

Piemēri

Ievaddati	Izvaddati
1 4 15 6	4

Ievaddati	Izvaddati
3 2 6 4	0

Ievaddati	Izvaddati
9 3 9 15	5

1. apakšuzdevuma testu ievaddati

Ievaddati
130 84 6 13

Ievaddati
195 832 68 221

Ievaddati
14132 3632 616 2976

Ievaddati
204522 206040 6443 25522

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie četri testi	2
2.	$P, Q, R, S \le 10^6$	9
3.	$\frac{P}{Q}$ un $\frac{R}{S}$ ir nesaīsināmi daļskaitļi	33
4.	Bez papildu ierobežojumiem	56
	Kopā:	100

Virknes fragments

Ir dota N veselu skaitļu virkne un vesels skaitlis S. Dotajā skaitļu virknē jāatrod pēc iespējas garāks fragments (pēc kārtas sekojoši virknes locekļi), kuru summa ir tieši S, vai arī jāsecina, ka šāda fragmenta dotajā virknē nav.

Ja N=11, S=3 un dotā virkne ir **2, 4, -2, 3, -5, 3, -2, 2, -5, 3, 1,** tad šajā virknē var atrast dažāda garuma fragmentus ar summu 3. Piemēram:

1) garumā 1: **2, 4, -2, 3, -5, 3, -2, 2, -5, 3, 1;**
2) garumā 3: **2, 4, -2, 3, -5, 3, -2, 2, -5, 3, 1;**
3) garumā 7: **2, 4, -2, 3, -5, 3, -2, 2, -5, 3, 1;**
4) garumā 10: **2, 4, -2, 3, -5, 3, -2, 2, -5, 3, 1;**

Uzrakstiet programmu, kas dotajā skaitļu virknē atrod garāko fragmentu ar norādīto summu!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N (virknes locekļu skaits, $1 \le N \le 2\times10^5$) un S (fragmenta summa, $|S| \le 4\times10^{15}$) vērtības, kas atdalītas ar tukšumzīmi.

Katrā no nākamajām N rindām dots pa vienam virknes loceklim - veselam skaitlim robežās no -2×10^9 līdz 2×10^9 (ieskaitot). Katram i($1\le i\le N$) virknes i-tais loceklis dots ievaddatu i+1-ajā rindā.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada lielākā iespējamā fragmenta garums un pirmā, fragmentā ietilpstošā, virknes locekļa indekss. Ja ir vairāki fragmenti ar summu S un lielāko garumu, tad jāizvada informācija par to, kuram pirmā locekļa indekss ir vismazākais. Starp abiem skaitļiem izvaddatos jābūt tukšumzīmei. Ja šāda fragmenta virknē nav, tad jāizvada "0 0".

Piemēri

Ievaddati	Izvaddati
11 3 2 4 -2 3 -5 3 -2 2 -5 3 1	10 1

Ievaddati	Izvaddati
10 -5 1 0 -1 1 -2 2 -3 3 -4 4	0 0

1. apakšuzdevuma testu ievaddati

Ievaddati
10 -3 1 0 -1 1 -2 2 -3 3 -4 4

Ievaddati
9 -6 -3 -1 4 -1 -5 -9 2 -3 6

Ievaddati
11 983 1026 -2061 -18 2727 -1 -1755 -18 117 864 -18 999

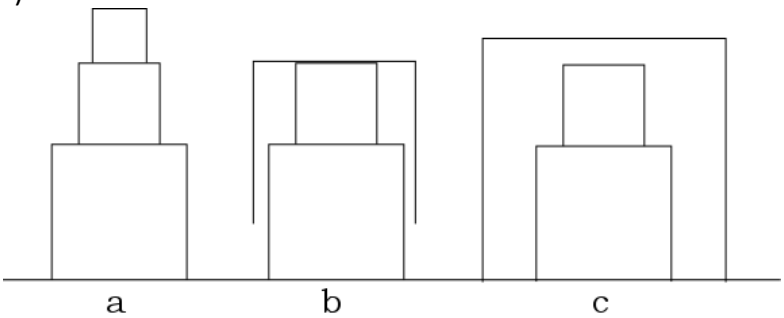
Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 3000	8
3.	Visi virknes locekļi ir pozitīvi	12
4.	S > 0	24
5.	Bez papildu ierobežojumiem	54
	Kopā:	100

Kastu tornis

Ir N atšķirīga izmēra kubveida kastes, kuru apakšējās skaldne ir vaļā. Kastu malu garumi ir izsakāmi veselā skaitā centimetru un sienu malu biezumu var neņemt vērā. Kastes tiek liktas pēc kārtas viena virs otras ar tukšo pusi uz leju, veidojot simetrisku kastu torni.

Katra nākamā kaste vai nu *uzsēžas* uz iepriekšējās kastes ar pamatu (skat. 25. a) zīm.), *uzkaras* uz iepriekšējās kastes ar augšējo skaldni (skat. 25. b) zīm.) vai *nosedz* vienu vai vairākas no iepriekšējām kastēm (skat. 25. c) zīm.).



25. zīm. Dažādi kastu torņa 5-3 turpināšanas varianti: a) kastes 2 uzsēšanās, b) kastes 6 uzkāršanās, c) divu iepriekšējo kastu torņa nosegšana ar kasti 9.

Uzrakstiet programmu, kas dotai dažāda izmēra kastu virknei nosaka, kāds ir kastu torņa kopējais augstums centimetros!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (kastu skaits, $1 \le N \le 2 \times 10^5$) vērtība.

Otrajā rindā doti kastu malu garumi centimetros - N atšķirīgi naturāli skaitļi, kuru vērtība nepārsniedz 2×10^9. Katram i ($1 \le i \le N$) i-tās kastes malas garums dots kā i-tais pēc kārtas. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada naturāls skaitlis - kastu torņa kopējais augstums centimetros.

Piemēri

Ievaddati	Izvaddati
3 5 3 2	10

Ievaddati	Izvaddati
3 5 3 6	8

Ievaddati	Izvaddati
3 5 3 9	9

1. apakšuzdevuma testu ievaddati

Ievaddati
20 15 1 17 12 7 6 9 3 13 5 14 19 20 2 10 11 4 16 18 8

Ievaddati
15 67 63 62 58 30 52 46 95 49 54 84 36 29 43 28

Ievaddati
10 1759 1088 1265 1715 547 1696 946 1534 757 136

Apakšuzdevumi un to vērtēšana

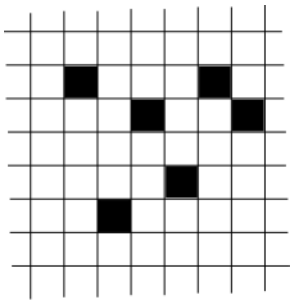
Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 2000, nevienas kastes malas garums nepārsniedz 2000	18
3.	N ≤ 2000	30
4.	Bez papildu ierobežojumiem	50
	Kopā:	100

Mazākais taisnstūris

Plaknē uzzīmēts kvadrātveida rūtiņu režģis un N rūtiņas ir iekrāsotas. Nepieciešams atrast taisnstūri ar pēc iespējas mazāku laukumu, kura malas sakrīt ar režģa līnijām un kura iekšpusē atrastos vismaz N-1 no iekrāsotajām rūtiņām.

Piemēram, 26. zīmējumā parādītajam rūtiņu iekrāsojumam mazākais šāds taisnstūris ir 24 (4×6) rūtiņas liels un tā iekšpusē atrodas piecas rūtiņas.

Uzrakstiet programmu, kas dotam rūtiņu iekrāsojumam atrod mazākā iespējamā aprakstītā taisnstūra laukumu!



26. zīm. Rūtiņu režģa piemērs.

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N (iekrāsoto rūtiņu skaits, $3 \le N \le 2 \times 10^5$). Katrā no nākamajām N rindām dots pa vienas iekrāsotās rūtiņas koordinātām - rindas un kolonnas numuram, kas atdalīti ar tukšumzīmi. Rindu un kolonnu numuri ir veseli skaitļi robežās no -10^9 līdz 10^9 (ieskaitot). Rindas un kolonnas ir numurētas ar veseliem skaitļiem pēc kārtas. Visas iekrāsotās rūtiņas ir atšķirīgas.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada naturāls skaitlis - mazākā iespējamā taisnstūra, kura iekšpusē atrodas vismaz N-1 no dotajām rūtiņām, laukums.

Piemēri

Ievaddati	Izvaddati
6 1 -1 0 1 -2 3 0 4 1 3 -3 0	24

Ievaddati	Izvaddati
4 5 -5 5 5 -5 -5 -5 5	121

1. apakšuzdevuma testu ievaddati

Ievaddati
7 14 15 9 26 5 35 8 9 79 3 23 8 46 26

Ievaddati
8 43 -3 -8 32 -7 -9 50 -28 8 41 -9 71 -6 -9 39 9

Ievaddati
9 175 10 58 209 149 44 59 225 181 64 20 199 162 80 34 225 34 211

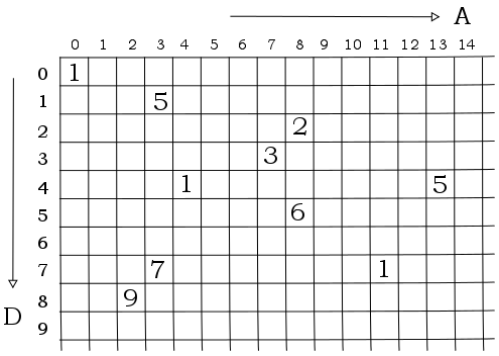
Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N = 3	14
3.	4 ≤ N ≤ 1000	33
4.	Bez papildu ierobežojumiem	51
	Kopā:	100

Sēņošanas čempionāts

Šīgada sēņošanas čempionāts notiks Lielajā mežu masīvā. Gatavojoties šim notikumam, Lielais mežu masīvs ir sadalīts joslās ziemeļu - dienvidu un rietumu - austrumu virzienā, veidojot kvadrātveida nogabalus. Katrā virzienā joslas ir sanumurētas pēc kārtas ar veseliem nenegatīviem skaitļiem sākot no 0.

Vienā vai vairākos nogabalos atrodas sēnes, kuras čempionāta dalībniekiem jāsasēņo, pārejot no viena nogabala uz blakus nogabalu tikai dienvidu vai austrumu virzienā. Katrā nogabalā esošais sēņu skaits ir zināms un nepieciešams aprēķināt, kādu lielāko sēņu skaitu čempionātā iespējams sasēņot, ja čempionāts sākas nogabalā (0;0).



27. zīm. Sēņu izvietojuma piemērs. Skaitlis norāda sēņu skaitu katrā nogabalā. Ja skaitlis nav norādīts, attiecīgajā nogabalā sēņu nav.

Piemēram, ja sēnes izvietotas nogabalos tā, kā parādīts 27. zīmējumā, tad lielākais sēņu skaits, ko iespējams sasēņot, ir 16, ko var iegūt sasēņojot sēnes no nogabaliem (0;0)→(3;1)→(7;3)→(8;5)→(11;7).

Uzrakstiet programmu, kas dotajam sēņu izvietojumam aprēķina lielāko sasēņojamo sēņu skaitu!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (netukšo nogabalu skaits, $1 \le N \le 2 \times 10^5$) vērtība.

Katrā no nākamajām N rindām dots viena nogabala, kurā atrodas sēnes, apraksts - trīs veseli skaitļi A (joslas numurs rietumu - austrumu virzienā, $0 \le A \le 4 \times 10^{18}$), D (joslas numurs ziemeļu - dienvidu virzienā, $0 \le D \le 4 \times 10^{18}$) un S (sēņu skaits nogabalā, $1 \le S \le 2 \times 10^9$), kas atdalīti ar tukšumzīmēm. Katrs nogabals datos ir minēts vienreiz.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada naturāls skaitlis - lielākais iespējamais sasēņojamo sēņu skaits.

Piemēri

Ievaddati	Izvaddati
10 2 8 9 3 1 5 0 0 1 13 4 5 11 7 1 7 3 3 8 5 6 8 2 2 4 4 1 3 7 7	16

Ievaddati	Izvaddati
5 1 8 100 2 5 112 3 3 101 4 2 107 5 0 111	112

1. apakšuzdevuma testu ievaddati

Ievaddati
8 8 7 16 8 8 13 0 8 11 6 1 1 4 3 18 2 4 20 7 7 11 3 4 1

Ievaddati
8 12 0 12 4 13 6 11 0 6 5 6 19 1 11 12 2 0 13 0 8 7 10 11 11

Ievaddati
8 4 22 1 1 24 2 11 17 1 7 19 6 0 25 5 5 25 1 3 25 1 3 24 3

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$D \leq 1$	18
3.	$N \leq 2000$	18
4.	$A, D \leq 2000$	18
5.	Bez papildu ierobežojumiem	44
	Kopā:	100

Akmens, šķēres, papīrīt's

Dace un Kārlis spēlē populāro spēli, kurā abiem pēc noskaitīšanas "Akmens, šķēres, papīrīt's - viens, divi, trīs", vienlaicīgi ar plaukstu jāattēlo viens no priekšmetiem - akmens, šķēres vai papīrs. Akmens uzvar šķēres, jo šķēres, mēģinot sagriezt akmeni, salūst. Šķēres uzvar papīru, jo var to sagriezt. Papīrs uzvar akmeni, jo papīrā akmeni var ietīt.

Dace un Kārlis šo spēli spēlē daudz reižu. Katru šādu reizi sauksim par *izspēli*.

Ja bērni ir attēlojuši dažādus priekšmetus, tad viens no viņiem izspēlē ir uzvarējis, ja vienādus, tad šī izspēle ir beigusies neizšķirti.

Dace un Kārlis ir pierakstījuši, kādu priekšmetu katrs ir attēlojis kārtējā izspēlē - attiecīgi burtu "A" (akmens), "S" (šķēres) vai "P" (papīrs). Tagad viņi savus pierakstus ir pārveidojuši saglabāšanai datorā, pārveidojot simbolu virknes saspiestā formātā. Katrs vienādu burtu fragments, kas garāks par 1, jāaizstāj ar skaitli, kas apzīmē šī burta parādīšanās reižu skaitu pēc kārtas, kam seko pats burts. Vienmēr tiek aizstāts viss vienādo burtu fragments. Piemēram, virkne "AAAAPSSPPSASSS" tiek pārveidota par "4AP2S2PSA3S", bet virkne "ASPASPASP" arī saspiestajā formātā ir tāda pati.

Tagad bērni pēc datorizētajiem pierakstiem vēlas noskaidrot, cik reizes uzvarējusi Dace, cik reizes - Kārlis, un cik reizes bijis neizšķirts. Piemēram, ja Daces attēlotos priekšmetus apraksta virkne "3A3SPAS2AP", bet Kārļa - "SA5P2A3S", tad piecas reizes ir uzvarējusi Dace, četras - Kārlis, bet trīs reizes izspēle beigusies neizšķirti.

Uzrakstiet programmu, kas dotām izspēļu rezultātu virknēm nosaka, cik reizes katrs no bērniem uzvarējis un cik reizes izspēle beigusies neizšķirti!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (Daces pierakstītās izspēļu rezultātu virknes saspiestā formātā garums, $N \le 2 \times 10^5$). Nākamajā rindā dota Daces pierakstītā izspēļu rezultātu virkne saspiestā formātā - N simbolu virkne, kas var saturēt tikai ciparus un lielos burtus A, S, P.

Nākamajās divās rindās tādā pat formātā ir aprakstīta Kārļa pierakstītā izspēļu rezultātu virkne saspiestā formātā.

Ir zināms, ka Daces un Kārļa izspēļu rezultātu virknes apraksta vienādu izspēļu skaitu, kas nepārsniedz 4×10^{18}.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada trīs veseli nenegatīvi skaitļi - Daces uzvarēto izspēļu skaits, Kārļa uzvarēto izspēļu skaits un neizšķirto izspēļu skaits. Starp katriem diviem blakus skaitļiem izvaddatos jābūt tukšumzīmei.

Piemēri

Ievaddati	Izvaddati
10 10A10SA10P 12 5P10A5PA3S7P	5 13 13

Ievaddati	Izvaddati
10 3A3SPAS2AP 8 PA5P2A3S	5 4 3

1. apakšuzdevuma testu ievaddati

Ievaddati
20 2P2S2A2P2S2A2P2S2A2S 20 ASASASPSPSASASPAPASA

Ievaddati
28 3ASA2S2A2SP2AS2PAPA2SASPA2PA 24 3P3A3P2A3P2A2S2A2S2A2P3A

Ievaddati
45 95P57A23S51A43P18A56S37P19A24S33A14P15S29A33P 39 18S70A57P47S72P66A15P41A28S58P35A14S26A

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Neviens priekšmets nav attēlots vairākas reizes pēc kārtas	10
3.	Neviens priekšmets nav attēlots vairāk kā deviņas reizes pēc kārtas	14
4.	Bez papildu ierobežojumiem	74
	Kopā:	100

Nesalasāmie burti

Dace un Kārlis spēlē populāro spēli, kurā abiem pēc noskaitīšanas "Akmens, šķēres, papīrīt's - viens, divi, trīs", vienlaicīgi ar plaukstu jāattēlo viens no priekšmetiem - akmens, šķēres vai papīrs. Akmens uzvar šķēres, jo šķēres, mēģinot sagriezt akmeni, salūst. Šķēres uzvar papīru, jo var to sagriezt. Papīrs uzvar akmeni, jo papīrā akmeni var ietīt.

Dace un Kārlis šo spēli spēlē daudz reižu. Katru šādu reizi sauksim par *izspēli*.

Ja bērni ir attēlojuši dažādus priekšmetus, tad viens no viņiem izspēlē ir uzvarējis, ja vienādus, tad šī izspēle ir beigusies neizšķirti.

Dace un Kārlis ir pierakstījuši, kādu priekšmetu katrs ir attēlojis kārtējā izspēlē - attiecīgi burtu "A" (akmens), "S" (šķēres) vai "P" (papīrs). Tagad viņi savus pierakstus ir pārveidojuši saglabāšanai datorā, pārveidojot simbolu virknes saspiestā formātā. Katrs vienādu burtu fragments, kas garāks par 1, jāaizstāj ar skaitli, kas apzīmē šī burta parādīšanās reižu skaitu pēc kārtas, kam seko pats burts. Vienmēr tiek aizstāts viss vienādo burtu fragments. Piemēram, virkne "`AAAAPSSPPSASSS`" tiek pārveidota par "`4AP2S2PSA3S`", bet virkne "`ASPASPASP`" arī saspiestajā formātā ir tāda pati.

Tagad bērni pēc datorizētajiem pierakstiem vēlas noskaidrot, cik reizes uzvarējusi Dace, cik reizes - Kārlis, un cik reizes bijis neizšķirts. Piemēram, ja Daces attēlotos priekšmetus apraksta virkne "`3A3SPAS2AP`", bet Kārļa - "`SA5P2A3S`", tad piecas reizes ir uzvarējusi Dace, četras - Kārlis, bet trīs reizes izspēle beigusies neizšķirti.

Diemžēl, nenoskaidrotu iemeslu dēļ, datorizētajos pierakstos daži burti var būt nesalasāmi un aizvietoti ar jautājuma zīmēm. Par laimi, ir zināms, ka visi cipari (ja tādi bija) ir saglabājušies un arī virkņu garumi nav izmainīti.

Tāpēc ir iespējams, ka attēloto priekšmetu virkne tagad izskatās šādi: "`?2P?3S4?`" un tā atbilst kādai no virknēm "`A2PA3S4A`", "`A2PA3S4P`", "`S2PA3S4A`" vai "`S2PA3S4P`".

Uzrakstiet programmu, kas dotām izspēļu rezultātu virknēm nosaka, kādu lielāko reižu skaitu katrs no bērniem var būt uzvarējis un kādu lielāko reižu skaitu izspēles var būt beigušās neizšķirti!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (Daces pierakstītās izspēļu rezultātu virknes saspiestā formātā garums, $N \le 2 \times 10^5$). Nākamajā rindā dota Daces pierakstītā izspēļu rezultātu virkne saspiestā formātā - N simbolu virkne, kas var saturēt tikai ciparus, jautājuma zīmes un lielos burtus `A`, `S`, `P`.

Nākamajās divās rindās tādā pat formātā ir aprakstīta Kārļa pierakstītā izspēļu rezultātu virkne saspiestā formātā.

Ir zināms, ka Daces un Kārļa izspēļu rezultātu virknes apraksta vienādu izspēļu skaitu, kas nepārsniedz 4×10^{18}.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada trīs veseli nenegatīvi skaitļi - lielākais iespējamais Daces uzvarēto izspēļu skaits, lielākais iespējamais Kārļa uzvarēto izspēļu skaits un lielākais iespējamais neizšķirto izspēļu skaits. Starp katriem diviem blakus skaitļiem jābūt tukšumzīmei.

Piemēri

Ievaddati	Izvaddati
10 10A10SA10P 12 5P10A5PA3S7P	5 13 13

Ievaddati	Izvaddati
3 ?A? 3 ??S	1 3 3

Ievaddati	Izvaddati
6 2?A?2P 5 3S??S	3 4 3

1. apakšuzdevuma testu ievaddati

Ievaddati
6 3?7?8? 7 2?10?6?

Ievaddati
12 ?A?A?A?A?A?A 12 P???S???A???

Ievaddati
10 2?AS3?S5?A 4 7P7?

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Izspēļu virknēs nav jautājuma zīmju	30
3.	Neviens priekšmets nav attēlots vairākas reizes pēc kārtas	12
4.	Neviens priekšmets nav attēlots vairāk kā deviņas reizes pēc kārtas	16
5.	Bez papildu ierobežojumiem	40
	Kopā:	100

Uzdevumu grāmata

Ingus savas zināšanas, prasmes un iemaņas jebkurā brīdī māk raksturot ar naturālu skaitli - *Ingus koeficientu* jeb *IK*. Ingum piemīt arī maģiskas spējas jebkuram uzdevumam jebkurā nozarē uzreiz noteikt tā grūtības līmeni - mazāko IK vērtību, lai Ingus šo uzdevumu spētu atrisināt. Jebkurš uzdevums, kuru Ingum izdevies atrisināt, paaugstina IK vērtību par vienu vienību.

Ingus ir lejuplādējis jaunu programmēšanas uzdevumu grāmatu ar N(N≥1) uzdevumiem un ir nolēmis atrisināt visus tajā esošos uzdevumus. Viņš rīkojas šādi: katru dienu Ingus pēc kārtas cenšas atrisināt ne vairāk kā pirmos M (M≥1) pagaidām vēl neatrisinātos uzdevumus. Ja uzdevuma atrisināšanai spēju pietrūkst, tad to drīkst izlaist un tā risināšanu atstāt uz citu dienu, bet nedrīkst mainīt uzdevumu risināšanas secību. Ja kādu dienu neizdodas atrisināt nevienu uzdevumu, tad uzdevumu risināšana apstājas.

Piemēram, ja grāmatā ir septiņi uzdevumi, kuru grūtības pakāpes pēc kārtas ir `3, 1, 7, 2, 4, 3` un `9`, M=3, un pirms šo uzdevumu risināšanas IK=1, tad Ingus pa dienām uzdevumus risinās šādi:

Diena	Aplūkotie uzdevumi	Atrisinātie uzdevumi un IK izmaiņas
1	3, 1, 7	1 (IK = 2)
2	3, 7, 2	2 (IK = 3)
3	3, 7, 4	3 (IK = 4), 4 (IK = 5)
4	7, 3, 9	3 (IK = 6)
5	7, 9	---

Kā redzams, šajā gadījumā Ingus nevar atrisināt visus uzdevumus - piektajā dienā IK vērtība ir nepietiekama kāda no kārtējo uzdevumu atrisināšanai.

Uzrakstiet programmu, kas aprēķina, kādai mazākai sākotnējai IK vērtībai Ingus spēs atrisināt visus grāmatā esošos uzdevumus, un cik dienas pie šīs sākotnējās IK vērtības tam būs nepieciešamas!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N (uzdevumu skaits grāmatā, $N \le 5\times10^5$) un M (lielākais vienā dienā aplūkojamo uzdevumu skaits, $M \le 5\times10^5$) vērtības, kas atdalītas ar tukšumzīmi. Otrajā rindā dota uzdevumu grūtība - N naturāli skaitļi, kur katram i($1 \le i \le N$) i-tais skaitlis pēc kārtas norāda i-tā pēc kārtas uzdevuma grūtību. Neviena uzdevuma grūtība nepārsniedz 2×10^9. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada divi naturāli skaitļi, kas atdalīti ar tukšumzīmi: mazākā sākotnējā IK vērtība, kas ļauj atrisināt visus grāmatā esošos uzdevumus, un mazākais visu uzdevumu atrisināšanai nepieciešamais dienu skaits ar šo IK vērtību.

Piemēri

Ievaddati	Izvaddati
7 3 3 1 7 2 4 3 9	3 3

Ievaddati	Izvaddati
5 9 7 6 5 4 3	3 5

1. apakšuzdevuma testu ievaddati

Ievaddati
7 2 5 3 1 6 3 5 3

Ievaddati
8 4 28 59 87 19 19 17 69 72

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie divi testi	2
2.	M = 1	10
3.	M ≤ 100	20
4.	Bez papildu ierobežojumiem	68
	Kopā:	100

Ingus koeficients

Ingus savas zināšanas, prasmes un iemaņas jebkurā brīdī māk raksturot ar naturālu skaitli - *Ingus koeficientu* jeb *IK*. Ingum piemīt arī maģiskas spējas jebkuram uzdevumam jebkurā nozarē uzreiz noteikt tā grūtības līmeni - mazāko IK vērtību, lai Ingus šo uzdevumu spētu atrisināt. Jebkurš uzdevums, kuru Ingum izdevies atrisināt, paaugstina IK vērtību par vienu vienību.

Ingus ir lejuplādējis jaunu programmēšanas uzdevumu grāmatu ar N (N≥1) uzdevumiem un ir nolēmis atrisināt visus tajā esošos uzdevumus. Viņš rīkojas šādi: katru dienu Ingus aplūko ne vairāk kā pirmos M (M≥1) pagaidām vēl neatrisinātos uzdevumus un cenšas tos atrisināt. Risināšanas secību šīs uzdevumu porcijas ietvaros Ingus var izvēlēties patvaļīgi. Ja kādu dienu neizdodas atrisināt nevienu uzdevumu, tad uzdevumu risināšana apstājas.

Piemēram, ja grāmatā ir seši uzdevumi, kuru grūtības pakāpes pēc kārtas ir `3, 1, 7, 2, 6` un `3`, M=3, un pirms šo uzdevumu risināšanas IK=1, tad Ingus pa dienām uzdevumus risinās šādi:

Diena	Aplūkotie uzdevumi	Atrisinātie uzdevumi un IK izmaiņas
1	3, 1, 7	1 (IK = 2)
2	3, 7, 2	2 (IK = 3), 3 (IK = 4)
3	7, 6, 3	3 (IK = 5)
4	7, 6	---

Kā redzams, šajā gadījumā Ingus nevar atrisināt visus uzdevumus - ceturtajā dienā IK vērtība ir nepietiekama kāda no kārtējo uzdevumu atrisināšanai.

Uzrakstiet programmu, kas aprēķina, kādai mazākai sākotnējai IK vērtībai Ingus spēs atrisināt visus grāmatā esošos uzdevumus, un cik dienas pie šīs sākotnējās IK vērtības tam būs nepieciešamas!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N (uzdevumu skaits grāmatā, $N \le 5\times10^5$) un M (lielākais vienā dienā aplūkojamo uzdevumu skaits, $M \le 5\times10^5$) vērtības, kas atdalītas ar tukšumzīmi. Otrajā rindā dota uzdevumu grūtība - N naturāli skaitļi, kur katram i($1\le i\le N$) i-tais skaitlis pēc kārtas norāda i-tā pēc kārtas uzdevuma grūtību. Neviena uzdevuma grūtība nepārsniedz 2×10^9. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada divi naturāli skaitļi, kas atdalīti ar tukšumzīmi: mazākā sākotnējā IK vērtība, kas ļauj atrisināt visus grāmatā esošos uzdevumus, un mazākais visu uzdevumu atrisināšanai nepieciešamais dienu skaits ar šo IK vērtību.

Piemēri

Ievaddati	Izvaddati
6 3 3 1 7 2 6 3	2 3

Ievaddati	Izvaddati
5 9 7 6 5 4 3	3 1

1. apakšuzdevuma testu ievaddati

Ievaddati
7 2 5 3 1 6 3 5 3

Ievaddati
8 4 28 59 87 19 19 17 69 72

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie divi testi	2
2.	M = 1	10
3.	M ≤ 100	20
4.	Bez papildu ierobežojumiem	68
	Kopā:	100

Meklēšana periodiskā virknē

Simbolu virkne A_N ir veidota no simbolu virknes A to N(N≥1) reizes atkārtojot un uzrakstot pašu sev galā. Kādai mazākajai N vērtībai virkne B ir A_N apakšvirkne? Tas ir, B sastāv no viena vai vairākiem A_N simboliem tādā pat secībā, kādā tie bija virknē A_N, bet šiem simboliem nav noteikti jābūt A_N pēc kārtas.

Piemēram, ja A = ”Atspere” un B = ”Asteres”, tad N = 3, jo “Asteres” ir A_3 = “AtspereAtspereAtspere” apakšvirkne (izcelti simboli, kas veido B), bet nav A_1 vai A_2 apakšvirkne.

Uzrakstiet programmu, kas, dotām virknēm A un B, atrod mazāko iespējamo N vērtību!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu G_A (virknes A garums, $G_A \le 2\times10^5$) un G_B (virknes B garums, $G_B \le 2\times10^5$) vērtības, kas atdalītas ar tukšumzīmi. Otrajā rindā dota virkne A - G_A simboli bez atdalošajām tukšumzīmēm. Trešajā rindā dota virkne B - G_B simboli bez atdalošajām tukšumzīmēm. Virknes var saturēt tikai angļu alfabēta lielos un mazos burtus, un ciparus. Lielie un mazie burti tiek uzskatīti par atšķirīgiem.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada vesels nenegatīvs skaitlis - mazākā iespējamā N vērtība, kurai B ir A_N apakšvirkne. Ja nevienai N vērtībai B nevar būt A_N apakšvirkne, tad izvaddatu vienīgajā rindā jāizvada skaitlis 0.

Piemēri

Ievaddati	Izvaddati
7 7 Atspere Asteres	3

Ievaddati	Izvaddati
7 6 TSUKUBA KABATA	3

Ievaddati	Izvaddati
7 6 Tsukuba Kabata	0

1. apakšuzdevuma testu ievaddati

Ievaddati
30 33 000111222333444555666777888999 314159265358979323846264338327950

Ievaddati
24 32 BUTCWKHSOPEFNJZGQLAIYDRX GCBWFEIDRNLTYHPQJKOSUZXASKEZLIAT

Ievaddati
40 40 agcttttcattctgactgcaacgggcaatatgtctctgtg tggattaaaaaaagagtgtctgatagcagcttctgaactg

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$G_A, G_B \leq 1000$	6
3.	$G_A, G_B \leq 30000$	20
4.	Visi simboli ir cipari	30
5.	Bez papildu ierobežojumiem	42
	Kopā:	100

Divi mīnusi

Dota veselu skaitļu virkne. Katra skaitļa priekšā, atšķirībā no jau ierastā veselu skaitļu pieraksta, ciparu virknes priekšā (ja tā nav 0) var būt gan viena, gan divas mīnuszīmes. Viena mīnuszīme joprojām apzīmē negatīvu skaitli, bet divas mīnuszīmes pozitīva skaitļa A pieraksta priekšā nozīmē $A - 1$. Piemēram, pieraksts "--7" apzīmē skaitli 6.

Ja virknes `1 10 3 -5 0 -3 6` skaitļu pieraksti tiek papildināti ar tieši trim mīnuszīmēm, tad mazākā iespējamā skaitļu summa (`-26`) ir virknei `1 -10 -3 -5 0 -3 -6`, bet lielākā (`24`) - virknei `-1 10 3 --5 0 --3 6`.

Uzrakstiet programmu, kas dotai veselu skaitļu virknei nosaka, kādu mazāko un kādu lielāko virknes skaitļu summu var izveidot, ja skaitļu pieraksts tiek papildināts ar tieši K papildu mīnuszīmēm?

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N (virknes locekļu skaits, $N \leq 2 \times 10^5$) un K (mīnuszīmju skaits ar kādu jāpapildina virknes skaitļu pieraksts, $K \leq 2N$) vērtības, kas atdalītas ar tukšumzīmi. Otrajā rindā doti N veseli skaitļi, kuru vērtības pēc moduļa nepārsniedz 2×10^9. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme. Zināms, ka visiem testiem ir iespējams skaitļu pierakstu papildināt ar K mīnuszīmēm.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada divi veseli skaitļi, kas atdalīti ar tukšumzīmi - virknes skaitļu summas mazākā un lielākā iespējamā vērtība.

Piemēri

Ievaddati	Izvaddati
7 3 1 10 3 -5 0 -3 6	-26 24

Ievaddati	Izvaddati
2 1 1 1	0 0

1. apakšuzdevuma testu ievaddati

Ievaddati
5 2 0 0 -495 149 769

Ievaddati
8 2 0 85 47 -74 82 0 -10 -72

Ievaddati
10 3 -1 -3 5 2 -6 7 -5 -1 8 -8

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$N \le 10$	8
3.	$N \le 1000$	20
4.	Bez papildu ierobežojumiem	70
	Kopā:	100

Pāra skaits ciparu

Aplūkojam tos naturālos skaitļus, kuru decimālajā pierakstā visi izmantotie cipari ir pāra skaitā. Augošā secībā pirmie šīs virknes skaitļi ir 11, 22, 33, 44, 55, 66, 77, 88, 99, 1001, 1010, 1100, 1111, 1122, 1133, 1144, 1155, 1166, 1177, 1188, 1199, 1212, 1221, 1313, 1331,…

Uzrakstiet programmu, kas ievadītai naturāla skaitļa N vērtībai nosaka, kurš skaitlis šajā virknē ir pēc kārtas N-tais?

Ievaddati

Vienīgajā rindā dota naturāla skaitļa N vērtība.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis, kurš virknē ir pēc kārtas N-tais. Zināms, ka visiem ievaddatiem šī skaitļa vērtība nepārsniedz 10^{18}.

Piemēri

Ievaddati	Izvaddati
10	1001

Ievaddati	Izvaddati
39	2020

Ievaddati	Izvaddati
55	2442

1. apakšuzdevuma testu ievaddati

Ievaddati
19

Ievaddati
101

Ievaddati
2945

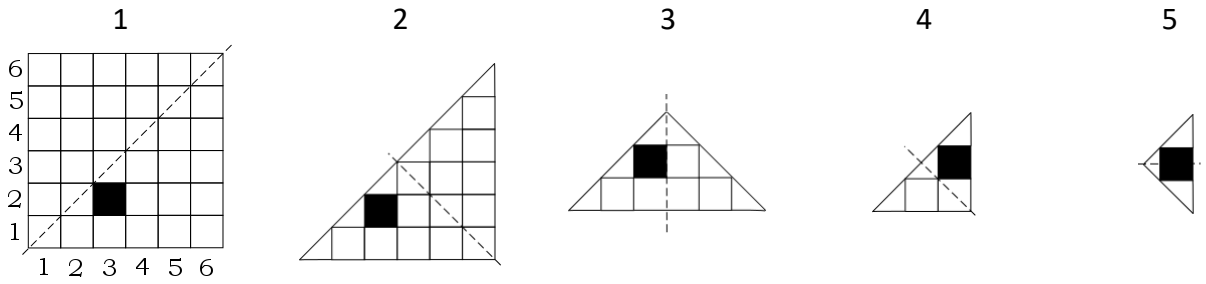
Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$N \le 10^6$	7
3.	$N \le 2 \times 10^9$	28
4.	Bez papildu ierobežojumiem	63
	Kopā:	100

Rūtiņu laukuma sagriešana

N×N rūtiņu laukuma rindas un kolonnas ir sanumurētas pēc kārtas ar skaitļiem no 1 līdz N. Viena laukuma rūtiņa ir aizkrāsota. Laukums tiek sagriezts vairākās daļās. Pirmo griezienu izdara, laukumu sagriežot pa diagonāli (jebkuru). Ja aizkrāsotā rūtiņa ir pārgriezta, tad process beidzas. Ja nē, tad tiek ņemta tā daļa (taisnleņķa vienādsānu trīsstūris), kurā ir aizkrāsotā rūtiņa, un pārgriezta uz pusēm pa augstumu, kas vilkts no taisnā leņķa virsotnes. Ja aizkrāsotā rūtiņa ir pārgriezta, tad process beidzas. Ja nē, tad iepriekš aprakstītajā veidā griešana tiek turpināta līdz kamēr aizkrāsotā rūtiņa tiek pārgriezta. Ja griezuma līnija iet tikai pa rūtiņas malu vai caur virsotni, tad rūtiņa neskaitās pārgriezta.

Piemēram, ja N=6 un aizkrāsota otrās rindas trešās kolonnas rūtiņa, tad tā tiks pārgriezta, izdarot pēc kārtas piekto griezienu, kā parādīts 28. zīmējumā.



28. zīm.

Uzrakstiet programmu, kas dotam laukuma izmēram un aizkrāsotās rūtiņas koordinātām nosaka mazāko griezienu skaitu, kāds nepieciešams, lai pārgrieztu aizkrāsoto rūtiņu!

Ievaddati

Vienīgajā rindā dota naturāla skaitļa N (rūtiņu laukuma malas garums, $N \le 10^{18}$), aizkrāsotās rūtiņas rindas numurs R ($1 \le R \le N$) un kolonnas numurs K ($1 \le K \le N$). Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada naturāls skaitlis - mazākais grieziena numurs pēc kārtas, kura rezultātā tiek sagriezta aizkrāsotā rūtiņa.

Piemēri

Ievaddati	Izvaddati
6 2 3	5

Ievaddati	Izvaddati
100 97 4	1

1. apakšuzdevuma testu ievaddati

Ievaddati
15 1 14

Ievaddati
20 20 4

Ievaddati
30 15 24

Apakšuzdevumi un to vērtēšana

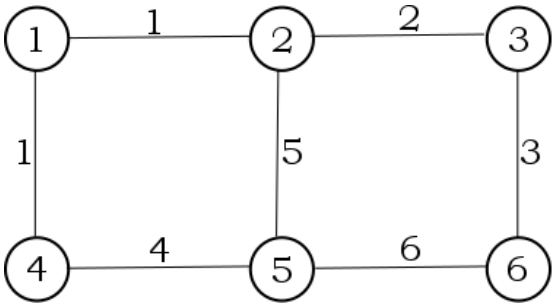
Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$N \le 100$	28
3.	$N \le 10^9$	40
4.	Bez papildu ierobežojumiem	30
	Kopā:	100

Sniega novākšana

Kādā valstī ir N (N≥1) pilsētas, kuras savieno M starppilsētu ceļu tīkls. Katrs ceļš savieno divas pilsētas un starp jebkurām divām pilsētām ir ne vairāk kā viens ceļš. No katras pilsētas var aizbraukt uz jebkuru citu vai nu tieši, vai braucot cauri citām pilsētām. Pilsētas ir sanumurētas ar naturāliem skaitļiem no 1 līdz N pēc kārtas.

Negaidīti ir uzsnigusi bieza sniega kārta un nepieciešams organizēt sniega novākšanu no visiem starppilsētu ceļiem. Lai neizraisītu domstarpības pilsētu starpā, valdība ir nolēmusi katrai pilsētai uzticēt vismaz viena tajā ieejoša ceļa tīrīšanu. Lai to nodrošinātu, nepieciešams visiem ceļiem piešķirt vienas pilsētas, kas atrodas tā galos, numuru tā, lai ar katras pilsētas numuru būtu vismaz viens ceļš.

Viens ceļiem piešķirto numuru variants parādīts 29. zīmējumā.



29. zīm. *Ceļu numuru piešķiršana (N=6, M=7)*

Uzrakstiet programmu, kas dotam starppilsētu ceļu tīkla aprakstam nosaka, kāds numurs jāpiešķir katram no ceļiem!

Ievaddati

Pirmajā rindā dotas naturālu skaitļu N (pilsētu skaits, $N \le 2\times10^5$) un M (starppilsētu ceļu skaits, $M \le 2\times10^5$) vērtības, kas atdalītas ar tukšumzīmi. Katrā no nākamajām M ievaddatu rindām dots viena starppilsētu ceļa apraksts - tā savienojošo pilsētu numuri, kas atdalīti ar tukšumzīmi. Katrs ceļš ievaddatos ir aprakstīts vienreiz.

Izvaddati

Ja ceļiem numurus aprakstītajā veidā piešķirt ir iespējams, tad izvaddatiem jāsatur tieši M rindas. Katram i(1≤i≤M) izvaddatu i-tajā rindā jābūt ceļam, kas aprakstīts ievaddatu i+1-ajā rindā, piešķirtais numurs - naturāls skaitlis no 1 līdz N. Ja iespējami vairāki numuru piešķiršanas varianti, jāizvada jebkurš no tiem.

Ja ceļiem numurus aprakstītajā veidā piešķirt nav iespējams, tad izvaddatu vienīgajā rindā jāizvada skaitlis 0.

Piemēri

Ievaddati	Izvaddati	Piezīme
6 7 1 2 5 2 6 5 3 2 3 6 1 4 4 5	1 5 6 2 3 1 4	Der arī citi izvaddatu varianti. Piemēram, 2 1 1 5 2 2 6 5 5 2 3 3 3 6 6 1 4 4 4 5 4

Ievaddati	Izvaddati
2 1 2 1	0

1. apakšuzdevuma testu ievaddati

Ievaddati
8 11 4 3 7 3 5 1 5 7 4 7 5 8 7 1 3 6 8 4 4 6 2 8

Ievaddati
10 9 10 2 2 3 9 2 8 9 5 8 8 4 3 7 7 1 6 1

Ievaddati
12 12 12 7 11 8 4 10 11 6 3 11 5 3 9 10 7 10 4 7 2 7 5 9 1 7

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	M ≤ 25	6
3.	N ≤ 1000	14
4.	Bez papildu ierobežojumiem	78
	Kopā:	100

Veikals

Nesen savu darbu ir uzsācis jauns veikals, kura darbību iespējams raksturot ar četru veselu nenegatīvu skaitļu A, B, C un D palīdzību:

- veikala atvēršanas brīdī tā krājumā bija A preces.
- katru dienu veikalā pircēji vēlas iegādāties B preces. Ja veikalā dienas sākumā ir mazāk par B precēm, tad šī ir pēdējā veikala darbības diena.
- ja kādas dienas vakarā veikalā paliek C preces vai mazāk, tad nākamajā dienā uz veikala atvēršanu tiek papildus piegādātas D preces.

Piemēram, ja `A=22, B=7, C=4, D=9`, tad veikals varēs darboties sešas dienas, preču daudzumam veikalā pa dienām mainoties šādi:

Diena	Preču daudzums veikalā dienas sākumā	Preču daudzums veikalā dienas beigās
1	22	15
2	15	8
3	8	1
4	10	3
5	12	5
6	5	---

Uzrakstiet programmu, kas dotām A, B, C un D vērtībām nosaka, vai veikals var darboties neierobežoti ilgi!

Ievaddati

Vienīgajā rindā dotas veselu nenegatīvu skaitļu A, B, C un D vērtības (A, B, C, D ≤ 2×10^{18}). Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada vesels skaitlis. Ja veikals var darboties neierobežoti ilgi, jāizvada skaitlis 1, bet ja nevar - skaitlis 0.

Piemēri

Ievaddati	Izvaddati
22 7 4 9	0

Ievaddati	Izvaddati
1 0 10 1	1

Ievaddati	Izvaddati
99 4 2 100	0

1. apakšuzdevuma testu ievaddati

Ievaddati
79210579 5467152 5467148 9767586

Ievaddati
163316941 10651311 10651290 19808239

Ievaddati
37872689 7288661 7288659 11324912

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	A, B, C, D ≤ 2×10^6	10
3.	Bez papildu ierobežojumiem	88
	Kopā:	100

2019./2020. mācību gads

Novada olimpiāde - 2020

Cik labo?

Spēles "Cik labo?" sākumā uz spēles galda atrodas N žetoni. Uz katra žetona ir uzrakstīts kāds naturāls skaitlis no 1 līdz N un visi uzrakstītie skaitļi ir atšķirīgi - t.i., nav divu žetonu, uz kuriem uzrakstītie skaitļi būtu vienādi. Žetonu sauc par *labu*, ja uz tā uzrakstītais skaitlis dalās ar vismaz vienu no naturāliem skaitļiem M_1 vai M_2.

Spēles gaitā spēlētāji pēc kārtas nosauc divus naturālus skaitļus: a_i un b_i. Tad spēles vadītājs no spēles galda noņem visus tos žetonus, uz kuriem uzrakstītajam skaitlim z ir spēkā sakarība $a_i \le z \le b_i$. Iespējams, ka kāds no šiem žetoniem jau ir noņemts iepriekšējo gājienu laikā. Spēles beigās vadītājam jāpasaka, cik labo žetonu joprojām atrodas uz spēles galda.

Piemēram, ja N=50, M_1=2, M_2=3, tad spēles sākumā uz galda atrodas 33 labie žetoni (2, 3, 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 26, 27, 28, 30, 32, 33, 34, 36, 38, 39, 40, 42, 44, 45, 46, 48 un 50).

Ja pirmais spēlētājs nosauc 23 un 32, tad seši labie žetoni tiek noņemti, bet 27 labie žetoni vēl paliek uz galda.

Ja tagad otrais spēlētājs nosauc 10 un 28, tad vēl deviņi žetoni tiek noņemti, un uz galda paliek 18 labie žetoni.

Uzrakstiet programmu, kas pēc spēlētāju izdarītajiem gājieniem - skaitļu intervālu apraksta, nosaka uz spēles galda atlikušo labo žetonu skaitu!

Ievaddati

Pirmajā rindā doti četri naturāli skaitļi - N($N \le 10^{18}$), M_1($M_1 \le 10^9$), M_2($M_2 \le 10^9$) un spēlētāju izdarīto gājienu skaits G($G \le 10^5$). Katrā no nākamajām G rindām dots viena spēlētāja izdarītā gājiena apraksts - divi naturāli skaitļi a_i un b_i ($a_i \le b_i \le 10^{18}$). Katram i($1 \le i \le G$) i-tā gājiena apraksts ir dots ievaddatu i+1-ajā rindā.

Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatu vienīgajā rindā jābūt veselam nenegatīvam skaitlim - uz spēles galda atlikušo labo žetonu skaitam spēles beigās.

Piemēri

Ievaddati	Izvaddati	Piezīme
50 2 3 2 23 32 10 28	18	Atbilst uzdevuma tekstā dotajam piemēram

Ievaddati	Izvaddati
450 21 30 2 375 500 20 345	2

1. apakšuzdevuma testu ievaddati

Ievaddati
300 2 7 10 290 293 23 54 111 132 33 33 122 144 276 284 6 45 128 143 60 71 250 259

Ievaddati
2999 13 11 9 333 444 2992 3000 1234 1432 76 145 873 900 23 25 238 338 999 1011 1666 1685

Ievaddati
999 12 18 10 765 768 567 678 675 786 984 1003 489 523 849 948 121 138 211 318 112 112 1 23

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	G = 1	8
3.	$M_1 = M_2$	20
4.	Visiem i ($1 \le i < G$) $a_{i+1} > b_i$	10
5.	Visiem i ($1 \le i < G$) $a_{i+1} > a_i$	20
6.	$N \le 10000$, $G \le 1000$	10
7.	$G \le 1000$	10
8.	Bez papildu ierobežojumiem	20
	Kopā:	100

Latīņu kvadrāti

Latīņu kvadrāts ir $n \times n$ rūtiņu laukums, kurā ir izmantoti n atšķirīgi simboli, turklāt katrs no šiem simboliem katrā laukuma rindā un kolonnā parādās tieši vienreiz.

Ja $n = 3$, tad latīņu kvadrāta piemērs ir

a	1	U
U	a	1
1	U	a

Uzrakstiet programmu, kas $m \times k$ rūtiņu laukumā nosaka, cik vietās tajā ir atrodami latīņu kvadrāti noteiktai n vērtībai!

Ievaddati

Pirmajā rindā doti trīs naturāli skaitļi m($2 \le m \le 5 \times 10^5$), k($2 \le k \le 5 \times 10^5$, $m \times k \le 10^6$) un n($2 \le n \le 62$), kas atdalīti ar tukšumzīmēm.

Katrā no nākamajām m ievaddatu rindām doti k simboli, kur katrs simbols ir vai nu angļu alfabēta burts vai cipars. Katram i($1 \le i \le m$) un j($1 \le j \le k$) simbols, kas atrodas laukuma i-tās rindas j-tajā kolonnā, ir dots kā j-tais ievaddatu i+1-ajā rindā.

Izvaddati

Izvaddatu vienīgajā rindā jābūt veselam nenegatīvam skaitlim - tādu vietu skaits dotajā tabulā, kurā iespējams atrast $n \times n$ lielu latīņu kvadrātu.

Piemēri

Ievaddati	Izvaddati
4 10 4 Kartupelis traKtieris rtKa12Aste aKtriseUna	1

Ievaddati	Izvaddati
5 6 2 a7a7a7 7a7a7a a7a8a7 7a7a7a a7a7a7	16

Ievaddati	Izvaddati
5 4 3 KurT uTec iKur Tute ciga	0

1. apakšuzdevuma testu ievaddati

Ievaddati
11 11 10 01234567890 12345678901 23456789012 34567890123 56789012345 67890123456 78901234567 89012345678 90123456789 45678901234 01234567891

Ievaddati
12 11 2 qpowtqopwui fuqooijfqoi jiethqwoihi ojodijweueh tqwoidjfioe whfohoidjwe iofhohthwei oijdoweihfe ioweiofdjie iojewodjiio djjodoejdoe ifhfodjsjeo

Ievaddati
12 13 3 ABCABACABCABA BACACBACABACA BCABACBABCABA CABCABAABCABA CABACABCABAAB CABCABABACACB ACABACABCABAC BABCABCABACAB CABABACACBACA BACABCABACBAB CABACABCABAAB CABAABAABCABA

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Rūtiņu laukumā ir tikai cipari, n ≤ 10	22
3.	k = n vai m = n	26
4.	Bez papildu ierobežojumiem	50
	Kopā:	100

Lakatiņa summa

Jebkuram n-ciparu naturālam skaitlim $A = \overline{a_n a_{n-1} \dots a_2 a_1}$ iespējams aprēķināt lakatiņa summu S(A) kā n skaitļu summu $S(A) = \overline{a_n a_{n-1} \dots a_2 a_1} + \overline{a_n a_{n-1} \dots a_2} + \overline{a_n a_{n-1} \dots a_3} + \dots + \overline{a_n a_{n-1}} + a_n$.

Citiem vārdiem sakot, pirmais saskaitāmais ir pats skaitlis A, bet katru nākamo saskaitāmo iegūst atmetot pēdējo iepriekšējā saskaitāmā ciparu.

Piemēram, S(1024) = 1024 + 102 + 10 + 1 = 1137, S(91) = 91+9 = 100, S(3) = 3.

Uzrakstiet programmu, kas dotam naturālu skaitļu intervālam [S_{maz}, S_{liel}] atrod mazāko un lielāko skaitli, kuru lakatiņu summas atrodas norādītajā intervālā!

Ievaddati

Pirmajā rindā doti divi naturāli skaitļi, kas atdalīti ar tukšumzīmi - S_{maz} un S_{liel} ($S_{maz} \le S_{liel} \le 10^{18}$).

Izvaddati

Izvaddatu vienīgajā rindā jābūt diviem skaitļiem A_{maz} un A_{liel}, kas atdalīti ar tukšumzīmi. A_{maz} jābūt mazākajam skaitlim, kuram ir spēkā sakarība $S_{maz} \le S(A_{maz}) \le S_{liel}$ vai 0, ja šāda skaitļa nav. A_{liel} jābūt lielākajam skaitlim, kuram ir spēkā sakarība $S_{maz} \le S(A_{liel}) \le S_{liel}$ vai 0, ja šāda skaitļa nav.

Piemēri

Ievaddati	Izvaddati
3 10	3 9

Ievaddati	Izvaddati
10 10	0 0

Ievaddati	Izvaddati
1136 1137	1023 1024

1. apakšuzdevuma testu ievaddati

Ievaddati
454442 638418

Ievaddati
1200536107 1234394373

Ievaddati
6455305608503 12266276192360

Ievaddati
9204850104885846 12294164966663901

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie četri testi	2
2.	$S_{liel} \le 10^5$	8
3.	$S_{liel} - S_{maz} \le 2 \times 10^6$	12
4.	$S_{liel} \le 2 \times 10^9$	18
5.	Bez papildu ierobežojumiem	60
	Kopā:	100

Neder kā summa

Jebkuram n-ciparu naturālam skaitlim $A = \overline{a_n a_{n-1} \dots a_2 a_1}$ iespējams aprēķināt *lakatiņa summu S(A)* kā n skaitļu summu

$$S(A) = \overline{a_n a_{n-1} \dots a_2 a_1} + \overline{a_n a_{n-1} \dots a_2} + \overline{a_n a_{n-1} \dots a_3} + \dots + \overline{a_n a_{n-1}} + a_n.$$

Citiem vārdiem sakot, pirmais saskaitāmais ir pats skaitlis A, bet katru nākamo saskaitāmo iegūst atmetot pēdējo iepriekšējā saskaitāmā ciparu.

Piemēram, S(1024) = 1024 + 102 + 10 + 1 = 1137, S(91) = 91+9 = 100, S(3) = 3.

Ne visi naturāli skaitļi var būt kāda skaitļa lakatiņa summa. Piemēram, nav tāda A, kuram S(A)=10.

Uzrakstiet programmu, kas dotam naturālu skaitļu intervālam [S_{maz}, S_{liel}] nosaka, cik šajā intervālā ir tādi skaitļi, kas nevar būt neviena skaitļa lakatiņa summa!

Ievaddati

Pirmajā rindā doti divi naturāli skaitļi, kas atdalīti ar tukšumzīmi - S_{maz} un S_{liel} ($S_{maz} \le S_{liel} \le 10^{18}$).

Izvaddati

Izvaddatu vienīgajā rindā jābūt veselam nenegatīvam skaitlim N - tādu skaitļu P skaitam, kuriem vienlaikus ir spēkā sakarības: $S_{maz} \le P \le S_{liel}$ un neeksistē tāds naturāls skaitlis A, ka P = S(A).

Piemēri

Ievaddati	Izvaddati
3 10	1

Ievaddati	Izvaddati
10 10	1

Ievaddati	Izvaddati
1136 1137	0

1. apakšuzdevuma testu ievaddati

Ievaddati
594385 1234342

Ievaddati
985740838 1228016108

Ievaddati
3293515909748 11657666999437

Ievaddati
3932798889795660 9399455997453946

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie četri testi	2
2.	$S_{liel} \le 10^5$	8
3.	$S_{liel} - S_{maz} \le 2 \times 10^6$	12
4.	$S_{liel} \le 2 \times 10^9$	18
5.	Bez papildu ierobežojumiem	60
	Kopā:	100

"Pēdējās formulas" autosacīkstes

Sezonas laikā S autosportisti piedalās "Pēdējās formulas" N sacensības un startē ar savu numuru (unikāls naturāls skaitlis robežās no 1 līdz S). Katrs sportists Sezonas sākumā katram sportistam ir 0 punktu. Katrās sacensībās desmit labāko rezultātu ieguvējiem tiek piešķirti punkti saskaņā ar 3. tabulu.

Ja vairāki sportisti finišē vienlaicīgi, un to rezultāts ir starp 10 labākajiem, tad visi šie sportisti sacensībās saņem vienādu punktu skaitu. Sezonas beigās katram sportistam tiek aprēķināta visās sacensībās iegūto punktu kopsumma.

Piemēram, katrās sacensībās iegūto vietu un kopvērtējuma punktu tabula sezonas nobeigumā var izskatīties tā, kā parādīts 4. tabulā. Kā redzams, lielāko punktu kopsummu - 45 punktus ieguvuši trīs sportisti.

Uzrakstiet programmu, kas pēc atsevišķās sacensībās iegūtajām vietām nosaka lielāko kāda sportista sezonas laikā iegūto punktu kopsummu, un izvada tā sportista numuru (vai to sportistu numurus, ja tādi ir vairāki), kas šo lielāko kopsummā ieguvis!

3.tabula. Punktu skaits par sacensībās iegūtu vietu

Vieta	Punkti
1.	25
2.	18
3.	15
4.	12
5.	10
6.	8
7.	6
8.	4
9.	2
10.	1

4. tabula. Sezonas nobeiguma tabula

Nr.	Sportists	1. sac.	2. sac.	3. sac.	4. sac.	Punkti
1	Miķelis Riepa	10.	Nepiedalījās	1.	2.	44
2	Antons Kardāns	9.	7.	Nepiedalījās	6.	16
3	Kaspars Zobrats	12.	8.	5.	Nefinišēja	14
4	Kristaps Stūre	8.	1.	10.	3.	**45**
5	Aleksis Ritenis	7.	1.	5.	Nepiedalījās	41
6	Pēteris Lukturis	6.	4.	Nepiedalījās	Nepiedalījās	20
7	Smuidris Vāks	5.	4.	2.	Nefinišēja	40
8	Zigis Sūknis	4.	1.	8.	Nepiedalījās	41
9	Anatolijs Bampers	3.	Nepiedalījās	4.	5.	37
10	Sandris Sprausla	2.	Nefinišēja	3.	4.	**45**
11	Juris Svece	1.	6.	7.	7.	**45**
12	Kārlis Haube	10.	9.	9.	1.	30

Ievaddati

Pirmajā rindā doti divi naturāli skaitļi - $S(S \le 10^5)$ un $N(N \le 10^5, S \times N \le 10^6)$. Katrā no nākamajām S rindām dots pa N veseliem nenegatīviem skaitļiem. Katram $s(1 \le s \le S)$ un $n(1 \le n \le N)$ n-tais skaitlis pēc kārtas s+1-ajā rindā ir s-tā sportista iegūtā vieta n-tajās sacensībās vai 0, ja sportists šajās sacensībās nav piedalījies vai nav finišējis. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatu pirmajā rindā jābūt diviem veseliem nenegatīviem skaitļiem - lielākajai kāda sportista iegūtajai punktu kopsummai sezonas laikā M un šo kopsummu ieguvušo sportistu skaitam S_M. Otrajā rindā jābūt S_M naturāliem skaitļiem - to sportistu numuriem augošā secībā, kuri kopsummā ieguvuši M punktus. Starp katriem diviem blakus skaitļiem izvaddatos jābūt tukšumzīmei.

Piemērs (atbilst uzdevuma tekstā dotajam piemēram)

Ievaddati	Izvaddati
12 4 10 0 1 2 9 7 0 6 12 8 5 0 8 1 10 3 7 1 5 0 6 4 0 0 5 4 2 0 4 1 8 0 3 0 4 5 2 0 3 4 1 6 7 7 10 9 9 1	45 3 4 10 11

1. apakšuzdevuma testu ievaddati

Ievaddati
17 3 1 17 16 2 16 15 3 15 12 4 14 11 5 13 8 6 12 7 7 11 4 8 10 2 9 9 1 10 8 3 11 7 5 12 6 6 13 5 9 14 4 10 15 3 13 16 2 14 17 1 17

Ievaddati
14 4 1 0 1 11 1 13 1 11 3 10 1 11 3 10 1 11 5 10 5 6 5 7 5 6 7 7 5 6 7 7 5 6 9 4 9 6 9 4 9 1 11 4 9 1 11 1 9 1 13 1 13 1 13 1 13 1

Ievaddati
12 11 1 2 3 4 5 6 7 8 9 10 11 2 3 4 5 6 7 8 9 10 11 12 3 4 5 6 7 8 9 10 11 12 1 4 5 6 7 8 9 10 11 12 1 2 5 6 7 8 9 10 11 12 1 2 3 6 7 8 9 10 11 12 1 2 3 4 7 8 9 10 11 12 1 2 3 4 5 8 9 10 11 12 1 2 3 4 5 6 9 10 11 12 1 2 3 4 5 6 7 10 11 12 1 2 3 4 5 6 7 8 11 12 1 2 3 4 5 6 7 8 9 12 1 2 3 4 5 6 7 8 9 10

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$S \leq 10$	25
3.	$S > 10$, $S_M = 1$	30
4.	Bez papildu ierobežojumiem	43
	Kopā:	100

Torņi

Eduardam patīk gan torņi, gan statistika un viņš kāri tver informāciju gan par uzbūvētiem, gan nojauktiem torņiem, un var katrā brīdī pateikt, cik augsts šobrīd ir n-tais augstākais tornis.

Piemēram, ja ir uzbūvēti 100, 70, 75, 83 un 91 metrus augsti torņi, tad ceturtā augstākā torņa augstums ir 75 metri. Ja 83 metrus augstais, iepriekš uzbūvētais, tornis tiek nojaukts, tad ceturtais augstākais tornis ir 70 metrus augsts.

Uzrakstiet programmu, kas dotai informācijai par uzbūvētajiem un nojauktajiem torņiem var atbildēt uz jautājumiem par tobrīd n-to augstāko torni!

Ievaddati

Pirmajā rindā dots naturāls skaitlis - ziņu un jautājumu skaits N($N \leq 2 \times 10^5$).

Katrā no nākamajām N rindām dots vienas ziņas vai jautājuma apraksts formā <burts><tukšumzīme><naturāls_skaitlis>.

Ja <burts> ir "U", tad ziņa apraksta uzbūvēta torņa augstumu un <naturāls_skaitlis> norāda uzbūvētā torņa augstumu garuma vienībās. Ja <burts> ir "N", tad ziņa apraksta nojaukta torņa augstumu un <naturāls_skaitlis> norāda nojauktā torņa augstumu garuma vienībās. Tiek garantēts, ka Ievaddati ir korekti - eksistē iepriekš uzbūvēts, nenojaukts tornis ar šādu augstumu. Ja <burts> ir "C", tad tiek aprakstīts vaicājums "Kāds ir n-tais augstākais tornis", un <naturāls_skaitlis> norāda n vērtību.

Zināms, ka katrā testā būs vismaz viens vaicājums. Neviena torņa augstums ziņās nepārsniedz 10^{18} garuma vienības. Vaicājumos n vērtība nepārsniedz N vērtību.

Izvaddati

Izvaddatos jābūt tik rindām, cik ievaddatos ir vaicājumu. Katrā rindā jābūt atbildei uz vienu jautājumu - attiecīgā torņa augstumam vai 0, ja torņu skaits ir mazāks nekā n vērtība vaicājumā. Atbildes uz vaicājumiem jāizvada tādā secībā, kādā vaicājumi doti ievaddatos.

Piemēri

Ievaddati	Izvaddati
8 U 100 U 70 U 75 U 83 U 91 C 4 N 83 C 4	75 70

Ievaddati	Izvaddati
8 C 2 U 123 N 123 C 1 U 123 C 1 U 124 C 2	0 0 123 123

Ievaddati	Izvaddati
7 U 12 U 15 U 761 U 178 U 8 U 34 C 5	12

1. apakšuzdevuma testu ievaddati

Ievaddati
10 U 17 C 1 U 2 U 17 C 2 C 2 N 17 C 2 U 93 C 3

Ievaddati
10 U 59 U 44 N 59 U 27 U 62 C 2 U 3 U 6 U 29 C 3

Ievaddati
10 C 3 U 25 C 1 U 80 C 2 C 1 U 56 U 17 C 3 C 2

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 3000, neviena torņa augstums nepārsniedz 3000	20
3.	N ≤ 3000	15
4.	Neviena torņa augstums nepārsniedz 2×10^5	20
5.	Bez papildu ierobežojumiem	43
	Kopā:	100

Valsts olimpiāde - 2020

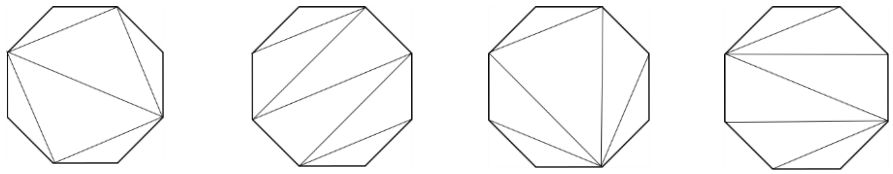
Daudzstūra sadalīšana

Regulārs N-stūris ar diagonāļu palīdzību ir sadalīts trīsstūros. Diagonāles nedrīkst krustoties un kopīgi tām drīkst būt tikai galapunkti - daudzstūra virsotnes.

Tad, kad n-stūris šādi ir sadalīts, tajā var atrast ne tikai trīsstūrus ar virsotnēm sākotnējā daudzstūra virsotnēs un malām, ko veido novilktās diagonāles un sākotnējā daudzstūra malas, bet arī četrstūrus, piecstūrus, …, N-1-stūrus.

Četri astoņstūra sadalīšanas Piemēri un attiecīgo daudzstūru skaits ir parādīts 5. tabulā:

5. tab. Daudzstūru skaits.



Četrstūri	5	5	5	5
Piecstūri	6	4	5	4
Sešstūri	6	3	5	3
Septiņstūri	4	2	3	2

Uzrakstiet programmu, kas dotam N-stūra sadalījumam trīsstūros nosaka, cik dažādus četrstūrus, piecstūrus, …, N-1-stūrus ar virsotnēm sākotnējā daudzstūra virsotnēs tajā iespējams atrast!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (regulārā daudzstūra virsotņu skaits, 5 ≤ N ≤ 500) vērtība.

Pieņemsim, ka virsotnes tiek numurētas pēc kārtas pulksteņrādītāja virzienā ar skaitļiem no 0 līdz N-1

Katrā no nākamajām N-3 rindām dots vienas novilktās diagonāles apraksts - sākuma un beigu virsotnes numurs, kas atdalīts ar tukšumzīmi.

Izvaddati

Izvaddatiem jāsatur N-4 rindas. Katram i(1≤i≤N-4) izvaddatu i-tajā rindā jābūt veselam nenegatīvam skaitlim - i+3-stūru skaits dotajā daudzstūrī. Daudzstūru skaits jāizvada pēc moduļa 10^9+9.

Piemēri

Ievaddati	Izvaddati
8 1 7 7 3 1 3 5 7 5 3	5 6 6 4

Ievaddati	Izvaddati
8 1 7 6 1 6 2 2 5 3 5	5 4 3 2

Ievaddati	Izvaddati
5 0 2 0 3	2

1. apakšuzdevuma testu ievaddati

Ievaddati
9 3 1 5 8 1 4 5 7 5 0 5 1

Ievaddati
10 8 2 2 4 2 6 0 2 8 6 2 5 9 2

Ievaddati
11 8 10 2 6 2 0 8 2 8 6 6 4 10 2 2 4

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 20	13
3.	N ≤ 100	25
4.	Bez papildu ierobežojumiem	60
	Kopā:	100

Divas iekavas

Korektu iekavu izteiksmi definē šādi:

- `()` ir korekta iekavu izteiksme;
- ja A ir korekta iekavu izteiksme, tad (A) arī ir korekta iekavu izteiksme;
- ja A un B ir korektas iekavu izteiksmes, tad AB arī ir korekta iekavu izteiksme.

Tā, `(()())` un `()(()(()))` ir korektas iekavu izteiksmes, bet `)(` un `)((`-nav.

Iespējams, ka iekavu izteiksmē, apgriežot divas dažādas iekavas uz pretējām, iespējams iegūt korektu iekavu izteiksmi. Dažreiz tas ir iespējams pat vairākos veidos.

Piemēram, virknē `(((((`, apgriežot divas dažādas iekavas uz pretējām, korektu iekavu virkni var iegūt piecos veidos (izceltas apgrieztās iekavas): `())()`, `()(())`, `(())()`, `(()())`, `((()))`.

Uzrakstiet programmu, kas dotai iekavu virknei nosaka, cik veidos no tās iespējams iegūt korektu iekavu virkni, apgriežot divas dažādas iekavas uz pretējām!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (iekavu virknes garums, N ≤ 10^6) vērtība.

Otrajā ievaddatu rindā dota iekavu virkne garumā N.

Izvaddati

Izvaddatu vienīgajā rindā jāizvada vesels nenegatīvs skaitlis - cik veidos no dotās iekavu virknes iespējams iegūt korektu iekavu no virknes, apgriežot divas iekavas uz pretējām.

Piemēri

Ievaddati	Izvaddati
6 ((((()	5

Ievaddati	Izvaddati
4)())	0

Ievaddati	Izvaddati
4))((	1

1. apakšuzdevuma testu ievaddati

Ievaddati
10 ()()()()()

Ievaddati
10 ())((((((

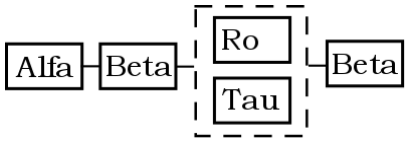
Ievaddati
12 ((())(()(())

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$N \leq 100$	20
3.	$N \leq 10^4$	36
4.	Bez papildu ierobežojumiem	42
	Kopā:	100

Formula

Lai izgatavotu kādu absolūti slepenu ķīmisku produktu, nepieciešams izpildīt noteiktu *operāciju virkni*. Dažas operācijas jāizpilda stingri noteiktā secībā, bet citas var veikt patvaļīgā secībā.



30. zīm. *Operāciju virknes piemērs.*

Piemēram, 30. zīmējumā attēlotā operāciju virkne nozīmē, ka vispirms jāizpilda operācija *Alfa*, pēc tam - operācija *Beta*, tad operācijas *Ro* un *Tau* patvaļīgā secībā, un, visbeidzot, vēlreiz operācija *Beta*. Tātad, ķīmisko produktu var izgatavot divos veidos: izpildot operāciju virkni *Alfa-Beta-Ro-Tau-Beta* vai *Alfa-Beta-Tau -Ro -Beta*.

Lai pierakstītu operāciju virknes, tiek izmantotas šādas vienošanās:

- Katru operāciju apzīmē angļu alfabēta burtu virkne, kuras garums nepārsniedz četrus simbolus.
- Operācijas, kuras var izpildīt patvaļīgā secībā, veido *operāciju grupu*. Operācijas vienas grupas ietvaros ir atšķirīgas, un pierakstot tiek liktas iekavās, atdalot ar komatiem. Operāciju grupa nevar saturēt citu operāciju grupu.
- Operācijas un operāciju grupas virknē tiek rakstītas, atdalot ar komatiem tādā secībā, kādā tās jāizpilda.
- Operāciju virknes beigās ir punkts.

1. zīmējumā attēloto operāciju virkni pieraksta kā "`Alfa,Beta,(Ro,Tau),Beta.`"

Raivis ir izgatavojis iekārtu, kas spēj izpildīt noteiktas operāciju virknes, ja nepieciešams, dažas operācijas tajā izlaižot. Nepieciešams noteikt, vai ar Raivja iekārtu iespējams izgatavot absolūti slepeno ķīmisko produktu.

Piemēram, ja Raivja iekārta spēj izpildīt virkni "`Jota,(Alfa,Beta),Tau,(Ro,Kapa,Beta).`", tad ar šo iekārtu var izgatavot produktu, kura izgatavošanai nepieciešams realizēt 1. zīmējumā attēloto operāciju virkni, jo iekārta var izpildīt operāciju virkni: *Alfa-Beta-Tau-Ro-Beta*, izlaižot un neizpildot šoreiz liekās operācijas *Jota* un *Kapa*.

Uzrakstiet programmu, kas nosaka, vai ar iekārtu var izgatavot produktu, un, ja var, izvada produkta izgatavošanai nepieciešamo operāciju virkni!

Ievaddati

Pirmajā rindā dota simbolu virkne, kuras garums nepārsniedz 10^6 simbolus - produkta izgatavošanai nepieciešamās operāciju virknes pieraksts. Operāciju virknē ir vismaz viena operācija.

Otrajā rindā dota simbolu virkne, kuras garums nepārsniedz 10^6 simbolus - iekārtas izpildāmās operāciju virknes pieraksts. Operāciju virknē ir vismaz viena operācija.

Simbolu virknes var saturēt tikai angļu alfabēta lielos un mazos burtus, komatus, atverošās un aizverošās iekavas un punktus. Katra simbolu virkne beidzas ar punktu.

Operāciju nosaukumos lielo un mazo burtu izmantošana ir svarīga.

Katras operācijas nosaukums ir burtu virkne, kurā ir vismaz viens un ne vairāk kā četri burti.

Iekavas nevar būt iekļautas.

Izvaddati

Vienīgajā rindā jāizvada simbolu virkne - iekārtas realizētā produkta izgatavošanas operāciju virknes pieraksts. Operāciju virknei jāapraksta stingri noteikta secīgu operāciju virkne - operāciju virknes pieraksts nedrīkst saturēt iekavas.

Ja iespējamas vairākas šādas operāciju virknes, jāizvada jebkura no tām.

Ja ar iekārtas palīdzību produktu izgatavot nevar, vienīgajā rindā jāizvada tikai punkts (“.”).

1. piemērs (atbilst piemēram uzdevuma tekstā)

Ievaddati	Izvaddati
`Alfa,Beta,(Ro,Tau),Beta.` `Jota,(Alfa,Beta),Tau,(Ro,Kapa,Beta).`	`Alfa,Beta,Tau,Ro,Beta.`

2. piemērs

Ievaddati	Izvaddati
`(A,B,C),(C,D,E),(E,F,G).` `(C,B,E),(C,A,G),E,(E,D,G),F.`	`B,C,A,C,E,D,E,G,F.`

3. piemērs

Ievaddati	Izvaddati
`(A,B,C),(C,F,E),(E,F,G).` `(C,B,E),(C,A,G),E,(E,D,G),F.`	`.`

1. apakšuzdevuma testu ievaddati

Ievaddati
`(a,A),(b,B),c,C,(d,D),e,E,(f,F,g,G).` `(A,B,C,a,b,c),D,d,(E,F,G,e,f,g),H,h.`

Ievaddati
`Al,Be,Al,Be,Ga,Al,Be,Ga,Alfa,Al,Be,Ga,Al,Be,Al.` `(Be,Al,Ga),(Be,Al,Ga),(Be,Al,Ga),Alfa,(Be,Al,Ga),(Be,Al,Ga).`

Ievaddati
`(d,Re,Mi,Fa,S,La,Si),d,Re,Mi,Fa,S,La,Si,(d,Re,Mi,Fa,S,La,Si).` `Si,La,(S,Fa,Mi,Re),d,d,(S,Fa,Mi,Re),(Si,La),(S,Fa,Mi,Re),d,La,(S,La,Si).`

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	Visu operāciju nosaukumi ir vienu burtu gari	42
3.	Bez papildu ierobežojumiem	56
	Kopā:	100

Galapunkti

Veselu skaitļu virknē $\{a_i\}$ nepieciešams noteikt, cik ir tādi fragmenti [i,j] (i<j), kuros neviena vērtība fragmenta iekšpusē (a_k, kur i≤k≤j) nepārsniedz tā lielākā galapunkta ($\max(a_i,a_j)$) vērtību.

Piemēram, ja virkne ir `-1, 5, 6, -32, 4, -1`, tad ir astoņi fragmenti ar minēto īpašību (elementu indeksācija sākas ar 1): `[1,2], [1,3], [2,3], [3,4], [3,5], [3,6], [4,5], [5,6]`).

Uzrakstiet programmu, kas ievadītai veselu skaitļu virknei aprēķina uzdevumā aprakstīto fragmentu skaitu!

Ievaddati

Pirmajā rindā dots virknes garums - naturāls skaitlis N ($N \le 2\times10^5$). Otrajā rindā dota skaitļu virkne - N veseli skaitļi, kas pēc absolūtās vērtības nepārsniedz 10^{15}. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Vienīgajā rindā jāizvada vesels nenegatīvs skaitlis - aprakstīto fragmentu skaits virknē.

Piemēri

Ievaddati	Izvaddati	Piezīme
6 -1 5 6 -32 4 1	8	Atbilst piemēram uzdevuma tekstā.

Ievaddati	Izvaddati	Piezīme
6 1 5 26 26 26 3	13	Der fragmenti `[1,2], [1,3], [1,4], [1,5], [2,3], [2,4], [2,5], [3,4], [3,5], [3,6], [4,5], [4,6], [5,6]`

1. apakšuzdevuma testu ievaddati

Ievaddati
5 -463499243011160 -970389778668892 -240824112132889 475513124938705 -596377894032737

Ievaddati
10 7 3 5 5 6 5 4 5 2 10

Ievaddati
10 0 1 1 0 1 0 0 0 1 0

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 500	18
3.	N ≤ 5000	25
4.	Bez papildu ierobežojumiem	55
	Kopā:	100

Skandalozā izrāde

Vecajā Rīgas teātrī 13. rindā ir N skatītāju vietas, kas numurētas ar naturāliem skaitļiem no 1 līdz N pēc kārtas. Šīs rindas skatītāji ir visai prasīga un izvēlīga publika, kas ierodas uz visām izrādēm un izrādes sākumā aizņem visas N vietas.

Ja teātra izrādes saturs, režisora veikums vai aktierspēle kādu no šajā rindā sēdošajiem skatītājiem neapmierina, viņš izrādes laikā (nesagaidot starpbrīdi vai izrādes beigas!) pieceļas un dodas uz tuvāko izeju, pa ceļam likdams piecelties citiem 13. rindas skatītājiem, kuriem spraucas garām. “Uz tuvāko izeju” nozīmē spraukties uz to rindas galu, kur kopējais iztraucēto skatītāju kopskaits ir mazāks. Ja uz abiem

rindas galiem iztraucēto skatītāju kopskaits ir vienāds, tad neapmierinātais skatītājs spraucas uz rindas galu, kurā atrodas vieta ar numuru 1. Iztraucētie skatītāji pēc piecelšanās apsēžas savās vietās un turpina vērot izrādi.

Neapmierināto skatītāju skaits var būt visai liels un tad dažiem skatītājiem izrādes laikā nākas piecelties vairākkārt, lai palaistu garām projām ejošos skatītājus. Nekad divi neapmierinātie skatītāji izrādi nepamet vienlaicīgi.

Īpaši skandalozā izrādē visi 13. rindas skatītāji pamet savas vietas pirms izrādes beigām.

Piemēram, ja N=14 un izrādi (tieši šādā secībā!) ir pametuši skatītāji, kas sēdēja 3., 7., 11., 6., 5., 14., 2., 4., 9., 1., 8., 10., 12. un 13. vietā, tad situācija izrādes laikā mainījās šādi (ar < apzīmēta tā vieta no kuras skatītājs pamet izrādi virzienā uz rindas sākumu, ar > apzīmēta tā vieta no kuras skatītājs pamet izrādi virzienā uz rindas beigām, ar ^ apzīmētas tās vietas, kur skatītājiem nācās piecelties, ar _ - jau tukšās vietas, ar o – vietas, kur skatītāji netika iztraucēti):

Kuras vietas skatītājs pamet zāli	Situācija 13. rindā	Piezīmes
3.	^^<oooooooooooo	Piecēlās skatītāji no 1., 2. un 3. vietas.
7.	^^_^^^<ooooooo	Piecēlās skatītāji no 1., 2., 4., 5., 6. un 7. vietas.
11.	oo_ooo_ooo>^^^	Piecēlās skatītāji no 11., 12., 13. un 14. vietas.
6.	^^_^^<_ooo_ooo	Piecēlās skatītāji no 1., 2., 4., 5. un 6. vietas.
5.	^^_^<__ooo_ooo	Piecēlās skatītāji no 1., 2., 4. un 5. vietas.
14.	oo_o___ooo_oo>	Piecēlās skatītājs no 14. vietas.
2.	^<_o___ooo_oo_	Piecēlās skatītāji no 1. un 2. vietas.
4.	^__<___ooo_oo_	Piecēlās skatītāji no 1. un 4. vietas.
9.	^_______^<o_oo_	Piecēlās skatītāji no 1., 8. un 9. vietas.
1.	<_______o_o_oo_	Piecēlās skatītājs no 1. vietas.
8.	________<_o_oo_	Piecēlās skatītājs no 8. vietas.
10.	__________<_oo_	Piecēlās skatītājs no 10. vietas.
12.	____________<o_	Piecēlās skatītājs no 12. vietas.
13.	_____________<_	Piecēlās skatītājs no 13. vietas.

Tātad šoreiz skatītājam no katras vietas ir nācies piecelties šādu reižu skaitu:

Vietas numurs	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Reižu skaits	8	5	1	4	3	2	1	2	1	1	1	2	2	2

Uzrakstiet programmu, kas dotam īpaši skandalozo izrādi pametušo 13. rindas skatītāju vietu numuriem nosaka, cik reizes katram no skatītājiem ir nācies piecelties!

Ievaddati

Pirmajā rindā dots naturāls skaitlis – vietu skaits 13. rindā N($N \le 2\times10^5$). Otrajā rindā doti N atšķirīgi naturāli skaitļi robežās no 1 līdz N – izrādi pametušo skatītāju vietu numuri tādā secībā, kā skatītāji pameta izrādi. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Vienīgajā rindā jāizvada N naturāli skaitļi. Katram i($1 \le i \le N$) i-tajam skaitlim pēc kārtas jābūt reižu skaitam, cik reizes nācās piecelties skatītājam, kas atradās i-tajā vietā.

Piemērs (atbilst uzdevumā tekstā dotajam)

Ievaddati	Izvaddati
14 3 7 11 6 5 14 2 4 9 1 8 10 12 13	8 5 1 4 3 2 1 2 1 1 1 2 2 2

1. apakšuzdevuma testu ievaddati

Ievaddati
16 15 7 8 9 2 10 11 6 1 16 5 4 12 13 3 14

Ievaddati
25 1 3 5 7 9 11 13 24 22 20 18 16 14 12 10 8 6 4 2 15 17 19 21 23 25

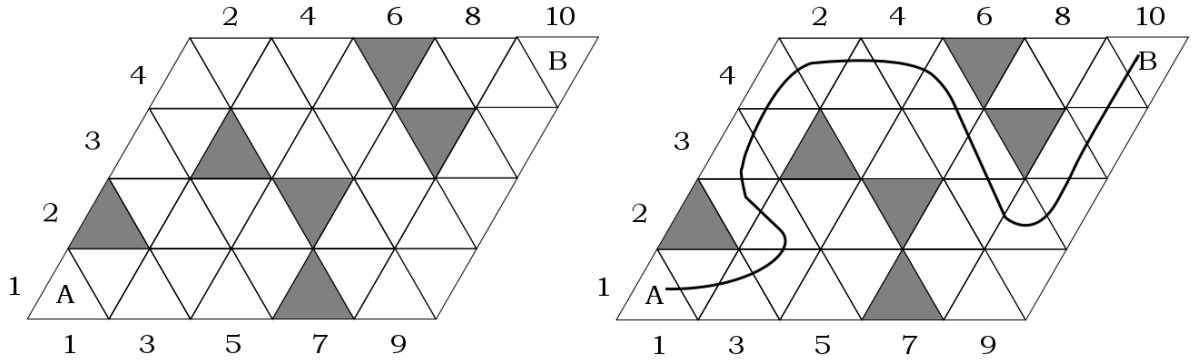
Ievaddati
24 7 13 4 14 6 15 8 16 2 17 9 18 1 19 10 20 3 21 11 22 5 23 12 24

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N ≤ 2000	36
3.	N ≤ 40000	20
4.	Bez papildu ierobežojumiem	42
	Kopā:	100

Trīsstūru labirints

Pēdējā laikā Ingus interesi ir piesaistījuši trīsstūru labirinti, kurus var attēlot kā rindās izvietotus regulārus trīsstūrus. Trīsstūru rindas un trīsstūri katrā rindā ir numurēti ar naturāliem skaitļiem, sākot no 1, pēc kārtas. Katrs trīsstūris var būt vai nu *tukšs*, vai *iekrāsots*. Kustība ir atļauta tikai uz un caur tukšajiem trīsstūriem. Viens *solis* ir pāreja no viena tukša trīsstūra uz tam kaimiņos esošu (kuram ar šo ir kopīga mala) tukšu trīsstūri. Katrā šādā labirintā ir jācenšas aizkļūt no pirmās rindas pirmā trīsstūra (A) līdz pēdējās rindas pēdējam trīsstūrim (B), veicot pēc iespējas mazāk soļu. Zināms, ka gan A, gan B trīsstūri ir tukši. Viena šāda labirinta piemērs, kurā četrās rindās katrā izvietoti 10 trīsstūri, parādīts 31. zīmējumā.



31. zīm. Labirinta piemērs un viens maršruts no A uz B ar mazāko iespējamo soļu skaitu (21).

Uzrakstiet programmu, kas dotam labirinta aprakstam atrod mazāko soļu skaitu s_{AB} maršrutā no A līdz B, kā arī dažādo maršrutu skaitu no A līdz B, kas izmanto tieši s_{AB} soļus!

Ievaddati

Pirmajā rindā dotas divu naturālu skaitļu R (labirinta rindu skaits) un T (trīsstūru skaits katrā rindā) vērtības, kas atdalītas ar tukšumzīmi. T vērtība vienmēr ir pāra skaitlis.

Otrajā rindā dota vesela nenegatīva skaitļa N (netukšo trīsstūru skaits) vērtība.

Skaitļu R, T un N vērtību ierobežojumus skatīt sadaļā "Apakšuzdevumi un to vērtēšana".

Katrā no nākamajām N rindām dotas viena iekrāsotā trīsstūra koordinātas. Katram i ($1 \le i \le N$) ievaddatu i+2-ajā rindā doti divi naturāli skaitļi r_i ($1 \le r_i \le R$) un t_i ($1 \le t_i \le T$), kas atdalīti ar tukšumzīmi.

Izvaddati

Vienīgajā rindā jāizvada divi naturāli skaitļi – mazākais soļu skaits s_{AB} maršrutā no A līdz B, un dažādo maršrutu skaits no A līdz B ar soļu skaitu s_{AB}. Maršrutu skaits jāizvada pēc moduļa 10^9+9. Ja maršruts no A uz B neeksistē, tad izvaddatu vienīgajā rindā jāizvada "0 0". Izvaddatos starp skaitļiem jābūt tukšumzīmei.

Piemēri

Ievaddati	Izvaddati
4 10 6 3 3 2 6 4 6 1 7 2 1 3 8	21 2

Ievaddati	Izvaddati
3 4 2 2 3 2 2	0 0

Ievaddati	Izvaddati
76 20 0	169 136781704

1. apakšuzdevuma testu ievaddati

Ievaddati
5 10 9 3 3 2 5 1 4 3 7 5 5 2 8 4 9 1 10 4 4

Ievaddati
7 14 11 6 3 3 1 4 4 6 10 4 8 4 5 7 2 6 6 2 9 5 7 4 12

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie divi testi	2
2.	$R \le 1000$, $T \le 2000$, $N \le 10^6$, nav iekrāsoti trīsstūri vienā no maršrutiem, kas iet gar labirinta malām (pirmajās divās kolonnās un pēdējā rindā, vai arī pirmajā rindā un pēdējās divās kolonnās)	20
3.	$R \le 1000$, $T \le 2000$, $N \le 10^6$	40
4.	$R \times T \le 2 \times 10^6$, $N \le 10^6$	20
5.	$R \le 10^6$, $T \le 2 \times 10^6$, $N \le 2000$, nav iekrāsoti trīsstūri vienā no maršrutiem, kas iet gar labirinta malām (pirmajās divās kolonnās un pēdējā rindā, vai arī pirmajā rindā un pēdējās divās kolonnās)	18
	Kopā:	100

Laivotāji

N laivotāji (N – nepāra skaitlis, N≥3) ir nolēmuši vienā trīsvietīgā un $\frac{N-3}{2}$ divvietīgās laivās braukt pa upi. Ir zināms katra laivotāja svars, un laivotājiem jāsadalās pa laivām tā, lai smagākās laivas kopējais svars būtu pēc iespējas mazāks. Visas tukšas laivas sver vienādi, tāpēc interesēsimies tikai par vienas laivas laivotāju kopējo svaru, pieņemot, ka tukšas laivas svars jebkurās smaguma mērvienībās ir 0.

Piemēram, ja laivotāju svars ir 53, 53, 100, 72 un 85 kg, tad mazākais iespējamais smagākās laivas svars ir 185 kg. Tas iespējams, ja trīsvietīgajā laivā ir trīs vieglākie, bet divvietīgajā - divi smagākie laivotāji. Sadaloties pa laivām jebkurā citā veidā, smagākās laivas svars būs lielāks.

Uzrakstiet programmu, kas dotiem laivotāju svariem nosaka mazāko iespējamo smagākās laivas svaru!

Ievaddati

Pirmajā rindā dots laivotāju skaits - naturāls nepāra skaitlis N(N<2000). Otrajā rindā doti N naturāli skaitļi – laivotāju svari kaut kādās smaguma mērvienībās. Katra laivotāja svars nepārsniedz 10^{15}.

Izvaddati

Vienīgajā rindā jāizvada naturāls skaitlis – mazākais iespējamais smagākās laivas svars.

Piemēri

Ievaddati	Izvaddati	Piezīme
5 53 53 100 72 85	185	Atbilst piemēram uzdevuma tekstā.

Ievaddati	Izvaddati
9 1 2 3 1 2 2 1 2 3	5

Ievaddati	Izvaddati
9 1 2 3 1 2 2 1 1 3	4

1. apakšuzdevuma testu ievaddati

Ievaddati
7 31 158 65 3 57 95 80

Ievaddati
11 57 48 9 18 5 1 10 1 10 50 71

Ievaddati
15 1 18 27 25 6 24 2 3 19 1 27 9 15 5 1

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	N < 10	15
3.	N < 200	30
4.	Bez papildu ierobežojumiem	53
	Kopā:	100

Mona Luīze

32. att. Mona Liza

Tiek uzskatīts, ka pati slavenākā glezna pasaulē ir Leonardo da Vinči "Mona Liza" jeb Džokonda (32. att.). Lai gan šī glezna iekļauta Ginesa rekordu grāmatā kā glezna, kas apdrošināta par vislielāko naudas summu, tās vērtību nav iespējams noteikt, jo, saskaņā ar Francijas kultūras mantojuma likumu, šo gleznu nedrīkst ne pārdot, ne pirkt.

Šī situācija nedod mieru slavenajai mākslas ekspertei Luīzei (paziņu lokā sauktai arī par *Monu Luīzi*), kuru regulāri aicina novērtēt dažādu gleznu vērtību un izteikt to naudas izteiksmē. Luīze uzskata, ka katras gleznas cena ir izsakāma veselos eiro un ir vismaz 1€. Džokondas cenu Luīze ir pieņēmusi vienādu ar 2×10^9€ un uzskata, ka neviena cita glezna to nevar sasniegt vai pārsniegt.

Parasti Luīze nenosauc precīzu gleznas cenu, bet pasaka, kura no divām gleznām ir vērtīgāka. Ja kāda no šīm gleznām tiek pārdota izsolē, tad vienas gleznas pārdošanas cena ļauj novērtēt arī otras gleznas cenu. Luīzes vērtējums nekad nenonāk pretrunā ar izsoles cenu - pēc Luīzes domām lētākā glezna nekad netiek pārdota dārgāk kā pēc Luīzes domām dārgākā.

Piemēram, ja Luīze ir novērtējusi, ka glezna ① ir lētāka nekā glezna ②, glezna ② ir lētāka nekā glezna ③, un zināms, ka glezna ② izsolē ir pārdota par 10430€, tad gleznas ① vērtība ir robežās no 1€ līdz 10429€, bet gleznas ③ vērtība - no 10431€ līdz 1999999999€

Uzrakstiet programmu, kas dotai informācijai par gleznu savstarpējo vērtību un izsolē pārdoto gleznu cenu nosaka katras gleznas vērtības, izteiktas eiro, diapazonu!

Ievaddati

Pirmajā rindā dots trīs veselu nenegatīvu skaitļu G (gleznu skaits, $1 \le G \le 10^5$), S (salīdzinošo novērtējumu skaits, $0 \le S \le 10^5$) un C (zināmo cenu skaits, $0 \le C \le G$) vērtības. Gleznas ir sanumurētas ar naturāliem skaitļiem no 1 līdz G pēc kārtas.

Katrā no nākamajām S rindām ir doti divi naturāli skaitļi g_i un g_j($1 \le g_i,g_j \le G$, $g_i \ne g_j$), kas nozīmē, ka glezna g_i Luīzes vērtējumā ir lētāka nekā glezna g_j.

Katrā no nākamajām C rindām ir doti divi naturāli skaitļi g_i un c_i ($1 \le g_i \le G$, $1 \le c_i < 2\times10^9$), kas nozīmē, ka glezna g_i ir pārdota izsolē par c_i eiro. Katra glezna izsolē var būt pārdota ne vairāk kā vienreiz.

Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme. Zināms, ka dati ir nepretrunīgi - katrai gleznai ir iespējams noteikt vērtību diapazonu.

Izvaddati

Izvaddatos jābūt tieši G rindām. Katram i($1 \le i \le G$) izvaddatu i-tajā rindā jābūt diviem naturāliem skaitļiem - i-tās gleznas mazākajai un lielākajai vērtībai, kas atdalītas ar tukšumzīmi.

Piemēri

Ievaddati	Izvaddati
3 2 1 1 2 2 3 2 10430	1 10429 10430 10430 10431 1999999999

Ievaddati	Izvaddati
7 5 0 2 1 3 1 5 4 1 4 7 6	2 1999999998 1 1999999997 1 1999999997 3 1999999999 1 1999999998 2 1999999999 1 1999999998

1. apakšuzdevuma testu ievaddati

Ievaddati
6 7 2 2 1 4 2 6 4 3 1 6 3 5 3 6 5 1 2000 6 1000

Ievaddati
7 8 2 1 2 2 3 2 4 3 4 4 5 4 6 5 6 6 7 5 17 3 13

Ievaddati
6 5 0 4 1 1 5 4 2 2 6 1 6

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	G ≤ 10	14
3.	G ≤ 1000, S ≤ 5000	23
4.	Bez papildu ierobežojumiem	61
	Kopā:	100

Kašķīgais parlaments

Kašķīgā parlamentā deputāti sēžu laikā uzstājoties mēdz necienīgi izteikties par deputātiem - gan citiem, gan sevi, gan klātesošiem, gan klāt neesošiem. Vienā runā par vienu un to pašu deputātu var necienīgi izteikties arī vairākkārt. Tomēr arī šajā parlamentā pastāv formālas pieklājības robežas. Tiklīdz deputāts savas runas laikā ir trīsreiz necienīgi izteicies par deputātiem, viņš tiek izraidīts no sēžu zāles. Deputātu runu saturs (tajā skaitā pirmie trīs deputāti, par kuriem būs necienīgie izteikumi) ir zināms iepriekš. Kādā secībā sēdes vadītājam jādod vārds deputātiem, lai neviens deputāts, kurš atrodas sēžu zālē (vēl nav izraidīts), par sevi kopumā nedzirdētu vairāk kā trīs necienīgus izteikumus?

Piemēram, ja sēdes sākumā zālē atrodas pieci deputāti, kas sanumurēti ar skaitļiem no 1 līdz 5, un pirmie trīs deputāti, par kuriem būs necienīgi izteikumi, katram ir tādi, kā parādīts tabulā

Deputāts	Pirmie trīs deputāti, par kuriem runā būs necienīgi izteikumi
1	3, 3, 5
2	2, 3, 5
3	1, 4, 5
4	1, 1, 1
5	1, 2, 3

, tad derīga uzstāšanās secība ir, piemēram, 4 2 1 3 5.

Pēc ceturtā deputāta runas zālē pirmais deputāts būs dzirdējis trīs necienīgus izteikumus, bet otrais, trešais, ceturtais un piektais deputāts vēl nebūs dzirdējuši sev adresētus necienīgus izteikumus. Ceturtais deputāts tiks izraidīts no zāles.

Pēc otrā deputāta runas pirmais būs dzirdējis trīs, bet otrais, trešais un piektais - pa vienam necienīgam izteikumam. Otrais deputāts tiks izraidīts no zāles.

Pēc pirmā deputāta runas pirmais un trešais deputāts būs dzirdējuši pa trim, bet piektais - divus necienīgus izteikumus. Pirmais deputāts tiks izraidīts no zāles.

Pēc trešā deputāta runas trešais un piektais deputāts būs dzirdējuši pa trim necienīgiem izteikumiem. Trešais deputāts tiks izraidīts no zāles.

Pēc piektā deputāta runas piektais deputāts būs dzirdējis trīs necienīgus izteikumus. Piektais deputāts tiks izraidīts no zāles.

Uzrakstiet programmu, kas dotai informācijai par necienīgajiem izteikumiem atrod derīgu deputātu uzstāšanos secību!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (deputātu skaits, $N \le 2 \times 10^5$) vērtība.

Deputāti ir sanumurēti ar naturāliem skaitļiem no 1 līdz N pēc kārtas.

Katram i ($1 \le i \le N$) ievaddatu i+1-ajā rindā doti trīs naturāli skaitļi - to deputātu numuri, par kuriem i-tais deputāts kā pirmajiem necienīgi izteiksies savā runā. Starp katriem diviem blakus skaitļiem ievaddatos ir tukšumzīme.

Izvaddati

Izvaddatu vienīgajā rindā jābūt N atšķirīgu skaitļu virknei - deputātu numuriem tādā secībā, kādā tiem jāuzstājas sēdes laikā.

Ja iespējamas vairākas korektas virknes, izvadiet vienu no tām.

Ja korekta virkne neeksistē, izvaddatu vienīgajā virknē izvadiet skaitli 0.

Piemēri

Ievaddati	Izvaddati	Piezīme
5 3 3 5 2 3 5 1 4 5 1 1 1 1 2 3	4 2 1 3 5	Atbilst piemēram uzdevuma tekstā.

Ievaddati	Izvaddati	Piezīme
4 1 1 1 2 2 2 3 3 3 4 4 4	3 1 4 2	Der jebkura deputātu secība.

1. apakšuzdevuma testu ievaddati

Ievaddati
5 4 5 4 1 1 4 1 1 5 2 1 2 3 3 2

Ievaddati
7 4 4 6 3 3 5 2 6 6 7 1 1 6 2 2 5 1 1 6 2 6

Ievaddati
9 5 2 8 8 4 9 5 5 5 9 9 4 2 1 3 2 2 8 8 2 8 5 6 7 4 4 9

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$N \le 2000$	10
3.	$N \le 20000$	28
4.	Bez papildu ierobežojumiem	60
	Kopā:	100

Teniss

Tenisists Ernests ir atkopies pēc traumas, kas lika izlaist visu sezonu, un atkal regulāri piedalās tenisa turnīros, kuros iespējams nopelnīt reitinga *punktus*. Tenisa turnīri katru sezonu notiek vienā un tajā pašā secībā un kārtējā turnīra sākumā tenisists zaudē iepriekšējā sezonā šajā turnīrā iegūtos reitinga punktus (ja tādi bija).

Ernesta turnīros iegūto punktu izmaiņa trīs secīgu sezonu laikā parādīta 6. tabulā. Izcelti turnīri, kuros Ernests ir piedalījies, bet pārsvītroti tie, kuros nav.

Vislielākais punktu skaits, kāds Ernestam bijis pēc kāda turnīra beigām, ir 96.

Uzrakstiet programmu, kas dotam turnīru sarakstam un tajos nopelnīto punktu daudzumam aprēķina lielāko punktu skaitu, kāds Ernestam varēja būt pēc kāda turnīra beigām!

6. tab. Turnīros iegūtie un zaudētie punkti.

Turnīrs	Nopelnītie punkti	Zaudētie punkti	Punktu summa turnīra beigās
StPetersburg	19	0	**19**
USOpen	22	0	**41**
Wimbledon	20	0	**61**
~~Halle~~	0	0	61
FrenchOpen	34	0	**95**
~~Geneva~~	0	0	95
~~AustralianOpen~~	0	0	95
~~StPetersburg~~	0	19	76
~~USOpen~~	0	22	54
Wimbledon	8	20	**42**
Halle	10	0	**52**
FrenchOpen	13	34	**31**
Geneva	7	0	**38**
AustralianOpen	15	0	**53**
StPetersburg	20	0	**73**
USOpen	21	0	**94**
Wimbledon	9	8	**95**
Halle	11	10	**96**
FrenchOpen	12	13	**95**
~~Geneva~~	0	7	88
AustralianOpen	21	15	**94**

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (turnīru skaits, $N \le 10^6$) vērtība.

Katrā no nākamajām N rindām dots viena turnīra, kurā Ernests ir piedalījies, apraksts - turnīra nosaukums (angļu alfabēta burtu virkne, ne garāka par 20 burtiem), kam seko tukšumzīme un šajā turnīrā iegūto punktu skaits (naturāls skaitlis P, $P \le 10^9$). Dati par turnīriem ir doti hronoloģiskā secībā.

Turnīru nosaukumos lielo un mazo burtu izmantošana ir svarīga.

Zināms, ka visos testos kopējā turnīra nosaukumu garumu summa nepārsniedz 2×10^6.

Ņemiet vērā, ka dotais turnīru saraksts atklātā veidā neapraksta visu turnīru secību gada ietvaros, kā arī to, kurā gadā minētajā turnīrā Ernests ir piedalījies!

Izvaddati

Izvaddatu vienīgajā rindā jāizvada naturāls skaitlis - lielākais teorētiski iespējamais punktu skaits pēc kāda turnīra beigām.

Piemēri

Ievaddati	Izvaddati
15 StPetersburg 19 USOpen 22 Wimbledon 20 FrenchOpen 34 Wimbledon 8 Halle 10 FrenchOpen 13 Geneva 7 AustralianOpen 15 StPetersburg 20 USOpen 21 Wimbledon 9 Halle 11 FrenchOpen 12 AustralianOpen 21	96

Ievaddati	Izvaddati
9 Pirmais 10 OtRais 12 Pirmais 8 Otrais 15 Otrais 13 OtRais 12 Pirmais 11 Otrais 9 Pirmais 13	36

1. apakšuzdevuma testu ievaddati

Ievaddati
19 StPetersburg 19 USOpen 22 Wimbledon 20 FrenchOpen 34 Geneva 7 AustralianOpen 15 StPetersburg 17 USOpen 25 Wimbledon 15 Halle 10 FrenchOpen 35 Geneva 1 AustralianOpen 17 StPetersburg 15 USOpen 28 Wimbledon 10 Halle 11 FrenchOpen 36 AustralianOpen 20

Ievaddati
15 Ab 123 Cd 234 Ef 345 Gh 456 Ij 567 Gh 678 Cd 789 Ab 891 Ef 912 Ij 123 Ef 234 Cd 345 Ab 456 Gh 567 Ij 678

Ievaddati
20 Te 12 Tur 13 Te 11 Tur 15 Te 10 Tur 16 Tur 18 Te 6 Tur 13 Te 13 Te 20 Nekur 5 Te 17 Tur 4 Te 19 Tur 8 Te 18 Tur 4 Nekur 4 Te 13

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie trīs testi	2
2.	$N \leq 10$	14
3.	$10 < N \leq 100$, visu turnīru nosaukumi ir vienu burtu gari	14
4.	$100 < N \leq 10^4$	20
5.	Bez papildu ierobežojumiem	50
	Kopā:	100

Izslēdzošais VAI

Programmēšanā bieži nākas izmantot aritmētisku operāciju "izslēdzošais VAI" jeb *xor*. Šīs operācijas operandi ir divi veseli nenegatīvi skaitļi, un operācijas rezultāts tiek aprēķināts pēc kārtas salīdzinot bitus šo skaitļu binārajā pierakstā. Ja tieši viens no šiem bitiem ir 1, tad attiecīgais bits operācijas rezultātā ir 1, un 0 - pretējā gadījumā.

Piemēram, lai aprēķinātu 43 *xor* 13, nepieciešams izteikt abus skaitļus binārajā skaitīšanas sistēmā: $43 = 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$, $13 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$, izpildīt bitu salīdzināšanu, un rezultātu izteikt atpakaļ decimālajā skaitīšanas sistēmā:

43	⇒	1	0	1	0	1	1	
13	⇒	0	0	1	1	0	1	
		1	0	0	1	1	0	⇒ **38**

Tādējādi 43 *xor* 13 = 38.

Ir iespējams aprēķināt arī garāku izteiksmju, kurās lietota operācija *xor*, vērtību.

Piemēram,

13 *xor* 23 *xor* 33 *xor* 43 = ((13 *xor* 23) *xor* 33) *xor* 43 = (26 *xor* 33) *xor* 43 = 59 *xor* 43 = 16.

Šajā uzdevumā interesēsimies par vēl sarežģītākām izteiksmēm, kur izmantota gan operācija *xor*, gan saskaitīšana (ar X un k apzīmēti naturāli skaitļi):

X *xor* (X+1) *xor* (X+2) *xor* … *xor* (X+k) = A (*)

Piemēram, ja X=7 un k=3, tad

A = 7 *xor* 8 *xor* 9 *xor* 10 = 15 *xor* 9 *xor* 10 = 6 *xor* 10 = 12

Uzrakstiet programmu, kas dotai vesela nenegatīva skaitļa A vērtībai atrod tādus naturālus skaitļus X un k ($X \le 10^{18}$, $k \le 10^{18}$), lai izteiksme (*) būtu patiesa!

Ievaddati

Pirmajā rindā dota naturāla skaitļa N (dažādo A vērtību skaits, N ≤ 10). Katrā no nākamajām N rindām dots vesels nenegatīvs skaitlis - viena A ($A \le 10^{15}$) vērtība.

Izvaddati

Izvaddatos jābūt tieši N rindām. Katram i($1 \le i \le N$) i-tajā rindā jāizvada divu naturālu skaitļu X un k vērtības ($X \le 10^{18}$, $k \le 10^{18}$), kas atdalītas ar tukšumzīmi un atbilst A vērtībai, kas dota ievaddatu i+1-ajā rindā.

Ja dotajai A vērtībai iespējami vairāki atrisinājumi, attiecīgajā rindā nepieciešams izvadīt jebkuru derīgu X un k vērtību pāri. Ja dotai A vērtībai atbilstošas X un k vērtības atrast nav iespējams, attiecīgajā rindā nepieciešams izvadīt "**0 0**".

Piemēri

Ievaddati	Izvaddati
3 12 7 15	7 3 99 1 8 6

Ievaddati	Izvaddati
3 1 11 42	1 8 1 9 43 6

Ievaddati	Izvaddati
1 0	2 7

1. apakšuzdevuma testu ievaddati

Ievaddati
3 37 75 99

Ievaddati
3 1811 432 416

Apakšuzdevumi un to vērtēšana

Nr.	Testu apraksts	Punkti
1.	Uzdevuma tekstā dotie divi testi	2
2.	Eksistē atrisinājums, kur X ≤ 1000 un k ≤ 1000	28
3.	Eksistē atrisinājums, kur X = 1	28
4.	Bez papildu ierobežojumiem	42
	Kopā:	100

Atrisinājumi

2016./2017. mācību gads

Novada olimpiāde - 2017

Dalītāju summa

Katra naturāla skaitļa N dalītājus d varam iedalīt divās grupās: "mazie" dalītāji, kuru vērtības nepārsniedz $\sqrt{N}$ un "lielie" dalītāji, kas lielāki par $\sqrt{N}$.

Ciklā apstrādāsim visus mazos dalītājus, un ievērosim, ka:

Mazo dalītāju skaits fiksētam d ir to skaitļu skaits segmentā $[A; B]$, kas dalās ar d. Šādu skaitļu skaits ir $s_{mazie} = \left[\frac{B}{d}\right] - \left[\frac{A-1}{d}\right]$. Šo skaitļu summa ir $s_{mazie} * d$.

Lielo dalītāju vērtības fiksētam d mainās lineāri ar soli 1 no $\left[\frac{A-1}{d}\right] + 1$ līdz $\left[\frac{B}{d}\right]$. Tātad šie skaitļi veido aritmētisko progresiju ar diferenci 1 un to summu var aprēķināt kā $\frac{\left(\left[\frac{B}{d}\right] - \left[\frac{A-1}{d}\right]\right)\left(\left[\frac{A-1}{d}\right] + 1 + \left[\frac{B}{d}\right]\right)}{2}$. Var ievērot, ka pirmais reizinātājs skaitītājā sakrīt ar s_{mazie} vērtību.

Šādi var rīkoties tikai tik ilgi, kamēr d vērtība ir mazāka par $\sqrt{A}$. Turpinot aprēķinus lielākām d vērtībām, iestāsies situācija, ka mazā dalītāja d vērtība pārsniedz lielā dalītāja $\left[\frac{A-1}{d}\right] + 1$ vērtību un aprēķinātās vērtības nebūs pareizas, jo viens un tas pats dalītājs gala summā būs ieskaitīts vairākkārt.

Ja $d = \sqrt{A}$, tad dalītāju summai (vienreiz!) jāpieskaita d un kā apakšējā robeža iepriekš aprakstītajos aprēķinos $\left[\frac{A-1}{d}\right]$ vietā jāņem d - t.i., jāatrisina sākotnējais uzdevums segmentam $[d^2 + 1; B]$ visiem dalītājiem, kas lielāki vai vienādi ar d.

Risinājuma izpildes laika sarežģītība ir $O(\sqrt{B})$.

Rēķināšanas spēle

Algoritms nepieciešamās datorprogrammas uzrakstīšanai jau ir dots uzdevuma tekstā, un vienīgā problēma ir tā realizēšana.

Tā kā algoritms ir saistīts ar skaitļu summas aprēķināšanu, tad skaidrs, ka aktuālā summas vērtība visu laiku būs jāsaglabā. Dotie ierobežojumi skaidri norāda, ka summas saglabāšanai būs nepieciešams vismaz 64 bitu vesela skaitļu tipa mainīgais.

Pirmajā gadījumā - "Skaitļu saskaitīšana" (kārtējais ielasītais skaitlis ir pozitīvs) izpildāmā darbība ir elementāra - summa jāpalielina par ievadītā skaitļa vērtību.

Otrajā gadījumā - "Ciparu svītrošana" (kārtējais ielasītais skaitlis ir 0 vai negatīvs) jāveic sarežģītākas darbības, kuras var sadalīt divos posmos:

A) jānosaka, kuri cipari būtu jāizsvītro;

B) ja summas vērtība satur kādu no izsvītrojamajiem cipariem, tad šī izsvītrošana jāveic.

Lai realizētu A), nepieciešams sadalīt skaitli pa cipariem un saglabāt informāciju par tiem. Informācijas saglabāšanai var izmantot 10 bitus - pa vienam katram decimālajam ciparam.

Skaitļa sadalīšanu ciparos var veikt, ciklā meklējot atlikumu, skaitli dalot ar 10 (atrod pēdējo ciparu) un pēc tam pašu skaitli dalot ar 10 (jau atrastais cipars tiek atmests).

```
vajag_izsvītrot ≔ (0000000000)
ja skaitlis = 0
    Atzīmē_bitu(vajag_izsvītrot,0)
citādi
    skaitlis = -skaitlis  //ielasītais skaitlis nebija pozitīvs
    kamēr skaitlis > 0
          cipars ≔ Atlikums(skaitlis,10)
          Atzīmē_bitu(vajag_izsvītrot,cipars)
          skaitlis ≔ skaitlis / 10
```

Piemēram, skaitlis 4743 tiktu apstrādāts šādi:

Skaitlis	Atlikums, dalot ar 10	Uzkrātā bitu informācija par jau atrastajiem cipariem (atbilst cipariem no 0 līdz 9)
4743	3	0001000000
474	4	0001100000
47	7	0001100100
4	4	0001100100
0		

Ja skaitļa vērtība kļūst 0, process jāpārtrauc. Izņēmums - ja 0 ir pats ievadītais skaitlis, kas nozīmē, ka no summas jāizsvītro visas 0. Šis gadījums ir jāapstrādā īpaši.

Algoritma B) (pati izsvītrošana) realizācija ir sarežģītāka, jo tādas aritmētikas operācijas "izsvītrot" nav. Var, protams, mēģināt pārveidot skaitli uz ciparu virkni un darboties ar to, bet pēc tam nāksies rezultātu pārveidot atpakaļ uz skaitli (jo, iespējams, būs jāizpilda summēšanas darbība), kas kopumā procesu padarīs visai sarežģītu.

Izrādās, ka arī izsvītrošanu var realizēt ar tīri aritmētiskām darbībām.

Piemēram, ja no skaitļa 4743 būtu jāizsvītro visi četrinieki, tad to varētu veikt, ielasot ciparus pēc kārtas, sākot no jaunākā, katru reizi noskaidrojot, vai kārtējais cipars nav vai ir jāizsvītro - t.i., vai jāiekļauj rezultātā, vai nē.

Pareizo decimālo pozīciju nodrošinās atbilstošā desmitnieka pakāpe (sākumā tā ir 1), kas tiek palielināta katru reizi, kad cipars tiek paņemts (netiek izsvītrots). Skaitļa dalīšanai ciparos var izmantot algoritmu, kas tika izmantots izsvītrojamo ciparu noteikšanai iepriekš.

Skaitlis	Kārtējais cipars	Desmitnieka pakāpe	Vajag izsvītrot?	Rezultāts
		1		0
4743	3	1	Nē	0+3×1=3
474	4	10	Jā	3
47	7	10	Nē	3+7×10=73
4	4	100	Jā	73
0				

Pseidokoda formā algoritms izskatītos šādi:

```
desmitnieka_pakāpe ≔ 1
rezultāts ≔ 0
kamēr skaitlis > 0
    cipars ≔ Atlikums(skaitlis,10)
    skaitlis ≔ skaitlis / 10
    ja Bits_nav_atzīmēts(vajag_izsvītrot,cipars)
          rezultāts ≔ rezultāts + cipars * desmitnieka_pakāpe
          desmitnieka_pakāpe ≔ desmitnieka_pakāpe * 10
```

Romāns

Lai atrisinātu šo uzdevumu, pamēģināsim atrisināt vienkāršāku uzdevumu: Cik numuri būtu nepieciešami, ja vienā numurā drīkst publicēt ne vairāk kā L lappuses?

Šo uzdevumu var risināt rijīgi - ņemt doto K nodaļu lappušu skaitu pēc kārtas un summēt to, kamēr netiek pārsniegta L vērtība. Šī uzdevuma izpildes laika sarežģītība ir $O(K)$.

Tagad šo uzdevumu varam risināt dažādām L vērtībām segmentā $[0; \sum_i L_i]$. Segmenta augšējā robeža izvēlēta šāda, jo tas ir mazākais lappušu skaits, kāds nepieciešams, lai visu romānu nopublicētu vienā numurā. Ar segmenta dalīšanu uz pusēm ir jāatrod mazākā L vērtība, kurai romāna publicēšanai nepieciešami ne vairāk kā N numuri. Šī L vērtība arī ir tā, kas uzdevuma formulējumā apzīmēta ar "mazākais iespējamais lappušu skaits, cik žurnālā aizņems pati apjomīgākā vienā žurnāla numurā publicētā romāna daļa". Tā kā maksimālais romāna lappušu skaits nepārsniedz $10^5 \cdot 10^9 = 10^{14}$, tad dažādās L vērtības būs nepieciešams pārbaudīt ne vairāk kā $\log_2 10^{14} < 47$ reizes.

Garākais fragments

Risināsim šo uzdevumu, izmantojot *divu norāžu* metodi. Pieņemsim, ka dotais masīvs ir $A[1 \dots N]$. Divas norādes L un R norāda uz pozīcijām masīvā tā, ka apakšmasīvs $A[L \dots R]$ satur tikai dažādus elementus. Uzturēsim arī datu struktūru, kas satur visus $A[L \dots R]$ elementus, un kurā iespējams ielikt un izņemt vienu elementu $O(\log N)$ laikā. Tāda datu struktūra ir *pašbalansējošs binārs koks*, un ir iebūvēts daudzās valodās (piemēram, C++ tas ir `set<int>`).

Pašā algoritmā pārvietosim norādi L no N līdz 1, un R sākumā ir $N + 1$ (norāda ārpus masīva). Kad L samazinās par 1, pārvieto R pa kreisi, un ņemam ārā no koka elementus $A[R]$, līdz koks vairs nesatur skaitli $A[L]$. Kad tas nesatur, ieliekam kokā skaitli $A[L]$. Tā kā pirms tam kokā nebija vienādu elementu, arī tagad visi elementi ir atšķirīgi. Turklāt ievērojam, ka R ir lielākā pozīcija masīvā tāda, ka apakšmasīvs $A[L \dots R]$ satur tikai atšķirīgus skaitļus.

Pašreizējā divu norāžu ierobežotā masīva garums ir $R - L + 1$. Mēs šo garumu varam saglabāt masīvā $G[L]$, kas būs lielākais apakšmasīva, kas sastāv tikai no atšķirīgiem skaitļiem, un sākas pozīcijā L, garums. Kad esam izskrējuši cauri visam masīvam, tad zinām arī atbildi. Tā kā esam garumus saglabājuši masīvā G, lai izvadītu sākumus visiem fragmentiem ar lielāko garumu, izskrienam cauri šim masīvam un izvadām tos L, kuriem $G[L]$ ir vislielākais.

Risinājuma izpildes laika sarežģītība: Ievērosim, ka katra no norādēm tiek pabīdīta ne vairāk kā N reizes, un ar katru vienas norādes nobīdi kokā tiek vai nu izņemts, vai arī ielikts viens elements. Līdz ar to izpildes laika sarežģītība ir $O(N \log N)$.

Puzle

Mēģināsim efektīvi nosimulēt uzdevumā aprakstīto procesu. Liksim gabaliņus klāt puzlei pēc kārtas un uzturēsim masīvu izmērā $N \times M$. Katrā no masīva rūtiņām vienmēr būs ierakstīta viena no trim vērtībām:

- ja vērtība ir 0, tad attiecīgo puzles gabaliņu Dace vēl nav iedevusi Kārlim;
- ja vērtība ir 1, tad šis puzles gabaliņš ir atlikts malā;
- ja vērtība ir 2, tad šo puzles gabaliņu Kārlis ir pielicis klāt puzlei.

Sākumā visu masīvu aizpildām ar 0. Kad Dace iedod jaunu gabaliņu, tad aplūkojam, vai puzlē nav jau pielikts kāds no šī gabaliņa (ne vairāk par 4) kaimiņiem. Ja tāda nav, tad ierakstām rūtiņā 1 (atliekam gabaliņu malā).

Ja tāds ir, tad ierakstām rūtiņā 2 (pieliekam puzlei gabaliņu). Tagad mums vajag pielikt arī visus gabaliņus, kas ir atlikti malā, un ir kaimiņi vai nu jaunajam gabaliņam, vai nu ir kaimiņi kādam no šādiem gabaliņiem, ko mēs tagad pieliksim klāt. To var efektīvi izdarīt, lietojot *meklēšanu plašumā* vai *meklēšanu dziļumā*. Ar vienu no šiem algoritmiem mēs varam apstaigāt visas rūtiņas, kurās ir ierakstīts 1 un kuras vajag pielikt dēļ jaunā gabaliņa.

Piemēram, 33. zīmējumā dotajā tabulā pēc iekrāsotā 2-nieka pievienošanas visi iekrāsotie 1-nieki tiks pārvērsti par 2:

2	2	1			
	2			1	1
			1	1	
2		1	2	1	1

33. zīm.

Risinājuma izpildes laika sarežģītība: Tā kā meklēšana plašumā vai dziļumā patērēs lineāru laiku (katru rūtiņu apmeklēs ne vairāk kā vienu reizi), tad kopējā risinājuma izpildes laika sarežģītība arī ir lineāra (attiecībā pret rūtiņu skaitu) - $O(NM)$.

Implementācija: Ievērosim, ka gan N, gan M vērtības var sasniegt pat 10^5. Līdz ar to masīvs izmērā $10^5 \times 10^5$ patērēs pārāk daudz atmiņas. Ir vairāki veidi, kā šo problēmu var apiet.

- Valodā C++ var izveidot N mainīga izmēra masīvus (vector<int>), katru izmērā M.
- Var izveidot vienu masīvu garumā NM, un darboties ar to kā ar divdimensiju masīvu. To var izdarīt, interpretējot elementu pozīcijā i kā elementu pozīcijā $\left(\frac{i}{M}, i \bmod M\right)$ divdimensiju masīvā.

Vienreizējie dalītāji

Ja mums būtu uzdevums, ka jāsaskaita dalītāju summa vienam skaitlim, nevis skaitļu intervālam, tad šo uzdevumu varētu risināt šādi:Ja ir dots skaitlis x kurš dalās ar d, tad arī $\frac{x}{d}$ ir x dalītājs. Tātad mēs varētu apskatīt visas d vērtības no 1 līdz $\sqrt{x}$ un rezultātam pieskaitīt d un $\frac{x}{d}$, ja x dalās ar d.

Ja ar šo pašu domu risinām doto uzdevumu, tad jāpārbauda visas dalītāja d vērtības no 1 līdz $\sqrt{B}$ un katram d atbilstošie dalītāji veido skaitļu intervālu no $\frac{A+d-1}{d}$ līdz $\frac{B}{d}$.

Tātad mums ir izveidojušies vairāki pēc kārtas esošu skaitļu intervāli, kuros esošie skaitļi ir dalītāji kādam no skaitļiem no dotā skaitļu intervāla. Intervālu skaits ir ne vairāk kā $2\sqrt{B}$, jo katram potenciālajam dalītājam (no 1 līdz $\sqrt{B}$) atbilst ne vairāk kā viens skaitļu intervāls un potenciālie dalītāji

no 1 līdz $\sqrt{B}$ arī sadalās ne vairāk kā $\sqrt{B}$ intervālos (intervāls sadalās vairākos, ja izrādās, ka neviens skaitlis no A līdz B nedalās ar konkrēto dalītāju).

Vienīgais uzdevums, kas atlicis, ir atrast elementu summu šiem intervāliem, katru skaitli ieskaitot tikai vienreiz. Tā kā šie skaitļu intervāli var savā starpā pārklāties, tad nevar apstrādāt katru intervālu atsevišķi un rezultātus sasummēt.

Sakārtosim visus intervālus pēc to sākuma augošā secībā. Tad varam caurskatīt visus intervālus un noskaidrot, vai tas pārklājas ar nākamo intervālu. Šādā veidā mēs varam atrast visu intervālu apvienojumu un izveidot savā starpā nešķeļošos skaitļu intervālus. Pēc tam jau drīkst katru intervālu apstādāt atsevišķi un aprēķināt tā skaitļu summu pēc aritmētiskās progresijas summas formulas: skaitļu summa intervālā $[k..l]$ ir $\frac{(l+k)(l-k+1)}{2}$.

Valsts olimpiāde - 2017

Cirks

Šajā uzdevumā izrādās, ka katram pērtiķim pietiek vienkārši simulēt viņa ceļu līdz augšai. Sākumā aprakstīsim algoritmu, kā to izdarīt, un pēc tam pamatosim, kāpēc tas strādā korekti un pietiekoši ātri.

Algoritms. Uzdevuma pamatdaļa ir katram pērtiķim izrēķināt, cik ilgi tam prasa nonākt līdz augšai (pēc tam pamatosim, ka katrs pērtiķis nevar nekur "iecikloties"). To var izdarīt, efektīvi nosimulējot viņa ceļu.

Katrām kāpnēm aplūkosim visas tās pozīcijas, no kurām iziet kāda virve, un sakārtosim tās. Tad paņemam vienu no pērtiķiem un izsekojam tā ceļojumam. Sākumā pērtiķis atrodas savu kāpņu apakšā. Kad sekojam tā ceļam, uzturam kāpņu numuru un pērtiķa pozīciju uz tām. Pieņemsim, ka pērtiķis ir kaut kādā pocīzijā savā ceļojumā. Lai izrēķinātu nākamo punktu, ar *bināro meklēšanu* atrodam nākamo **zemāko** pozīciju **virs** pašreizējās pozīcijas, no kurās iziet virve. Tad pārvietojam pērtiķi uz to pozīciju un tad gar šo virvi uz jauno pozīciju citās kāpnēs. Tā turpinām, līdz pērtiķis nonāk kādu kāpņu galā. Paralēli ceļojumam skaitām arī, cik laika pērtiķis jau ir noceļojis.

Pēc tam mums jāatbild uz dotajiem pieprasījumiem, cik pērtiķu ir galā konkrētajos laika brīžos. Lai to izdarītu, var izmantot *divu norāžu metodi*. Paņemam laikus, kad pērtiķi nonāca galā, un sakārtojam tos. Pieprasījumi mums jau ir doti sakārtotā secībā. Uzturam divas norādes katrā no šiem diviem masīviem, sākumā norādes rāda pirms masīvu sākumiem. Tad pa vienai pozīcijai bīdām pieprasījumu norādi; kad aplūkojam jauno pieprasījumu, tad pabīdām gala laiku norādi tik tālu, kamēr pērtiķu gala laiki ir ne lielāki par šo pieprasījumu. Tad atbilde uz šo pieprasījumu ir tieši tik pērtiķu, cik esam aplūkojoši ar gala laiku norādi.

Korektība. Tagad pierādīsim divas lietas:

1. Visi pērtiķi kādā brīdī nonāk kādu kāpņu augšā.
2. Gar katru virvi kopā noiet tieši divi pērtiķi.

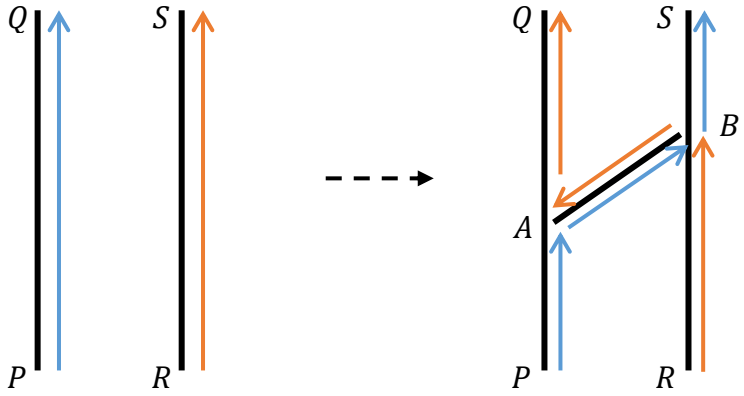
Pierādīsim abus apgalvojumus ar *matemātisko indukciju* pa virvju skaitu.

Ja virvju skaits ir 0, tad visi pērtiķi uziet augšā katrs pa savām kāpnēm. Gan 1., gan 2. apgalvojums izpildās. Šī ir indukcijas bāze.

Tagad pierādīsim indukcijas pāreju. Pieņemsim, ka virvju skaits ir k un abi apgalvojumi 1., 2. izpildās. Pierādīsim, ka tie joprojām izpildās, ja konfigurācijai pievienojam vēl vienu virvi papildus.

Aplūkosim situāciju zīmējumā, kur starp divām kāpnēm tiek pievienota jauna virve. Abas bultiņas norāda divu pērtiķu ceļojumu kāpņu posmos, kuros tiek pievienota jauna virve. Ievērosim, ka pērtiķi kustās uz augšu. Viens kustās no punkta P uz Q, bet otrs no punkta R uz S. Tad pievienojam virvi, kas savieno punktus A un B. Pērtiķu ceļojumi tad šajā posmā mainās no $P \to Q$ uz $P \to A \to B \to S$, un no $R \to S$ uz $R \to B \to A \to Q$.

Kāpēc apgalvojumi joprojām izpildās? Ievērosim, ka visu pērtiķu maršruti saglabājas, izņemot divus tikko aplūkotos pērtiķus. Pieņemsim, ka pirmā pērtiķa maršruts bija $[X] \to P \to Q \to [Y]$, un otrā pērtiķa maršruts bija $[Z] \to R \to S \to [T]$ (ar [] iekavām apzīmēta daļa no ceļojuma maršruta). Tad šo divu pērtiķu ceļojumi kļūst attiecīgi par $[X] \to P \to A \to B \to S \to [Y]$ un $[Z] \to R \to B \to A \to Q \to [T]$ (skat. 34. zīm.) Tātad kopā šo divu pērtiķu nostaigātie nogriežņi paliek tādi paši, plus abos virzienos tiek nostaigāts nogrieznis AB. Līdz ar to katrs pērtiķis joprojām nonāk augšā, un arī katra virve joprojām tiek nostaigāta tieši divas reizes.



34. zīm.

Risinājuma izpildes laika sarežģītība: Lai sakārtotu virvju galapunktus uz katrām kāpnēm, nepieciešams $O(V \log V)$ laiks. Kopā visu pērtiķu ceļojumu simulēšanā tiek veikti $O(V)$ soļi (jo katrā solī simulēšanā mēģinām noiet gar kārtējo virvi, un kopā pērtiķi gar virvēm noiet $2V$ reizes). Lai vienam pērtiķim konkrētā pozīcijā atrastu, pa kuru virvi tam tālāk ir jāiet, vajadzīgs $O(\log V)$ laiks. Ņemam arī vērā, ka simulēšanas laiks prasa vismaz laiku $O(N)$, jo ceļojuma gala brīdis jāizrēķina katram no N pērtiķiem. Tāpēc simulēšanas daļa prasa $O(N + V \log V)$ laiku.

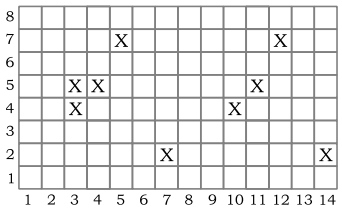
Lai atbildētu uz pieprasījumiem, sākumā vajag sakārtot pērtiķu gala brīžu masīvu, kas prasa laiku $O(N \log N)$. Pati divu norāžu metode prasa lineāru laiku, $O(N + L)$.

Līdz ar to kopējā izpildes laika sarežģītība ir $O(N \log N + V \log V + L)$.

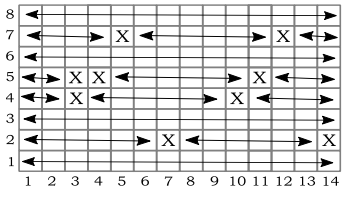
Dēlīši

Vispirms aplūkosim risinājumu gadījumam, kad visi dēlīši novietoti horizontāli.

Ievērosim, ka nav nepieciešams domāt par visas plāksnes sazāģēšanu - pēc uzdevuma nosacījumiem, visu plāksni var aplūkot kā vienu centimetru (vai vienu rūtiņu) augstu un N cm (vai N rūtiņu) platu rindu kopumu. Šādu rindu skaits ir M. Turklāt katras rindas sazāģēšana notiek neatkarīgi no citu rindu sazāģēšanas. Tas nozīmē, ka nepieciešams mācēt atrisināt uzdevumu vienai rindai jebkuram iespējamajam defektu izvietojumam.



35. zīm. Plāksnes piemērs.



36. zīm. Fragmenti.

Pats vienkāršākais gadījums ir rinda bez defektiem (35. zīmējumā dotajam plāksnes piemēram tādas ir 1., 3., 6. un 8. rinda). No šādas rindas iespējams izzāģēt lielākais $\left\lfloor \frac{N}{K} \right\rfloor$ dēlīšus (katra dēlīša platums - K cm).

Ja rindā ir viens vai vairāki defekti, tad visu rindu varam uztvert kā divu vai vairāku fragmentu (vai īsāku rindu) bez defektiem kopumu. Šāda fragmenta garums var būt arī 0 (ja divi defekti atrodas blakus). Rindu bez defektiem arī varam uzskatīt par vienu lielu fragmentu. Visi

fragmenti aplūkotajam piemēram ir parādīti 36. zīmējumā. Līdzīgi kā rindai bez defektiem, no katra fragmenta var izzāģēt $\left\lfloor \frac{fragmenta\_garums}{K} \right\rfloor$ dēlīšus.

Pieņemsim, ka rindā ir P defekti, kuru koordinātas (kolonnu numuri) ir sakārtotas augošā secībā: $d_1 < d_2 < \cdots < d_P$. Šiem defektiem ir vērts pievienot divus fiktīvus "defektus" - vienu pirms rindas sākuma ($d_0 = 0$) un vienu - pēc beigām ($d_{P+1} = N + 1$). Tad katram $i = 0, \ldots, P$ fragmenta garumu var aprēķināt kā $d_{i+1} - d_i - 1$ un no rindas iespējams izzāģēt lielākais $\sum_{i=0}^{P} \left\lfloor \frac{d_{i+1} - d_i - 1}{K} \right\rfloor$ dēlīšus.

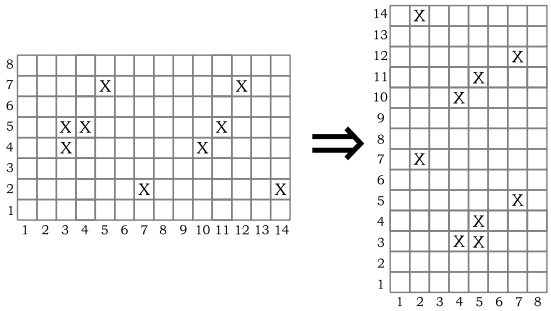
Piemērā dotā laukuma piektajai rindai $P = 3, d_1 = 3, d_2 = 4, d_3 = 11$ un fragmentu garumi ir attiecīgi 2,0,6 un 3.

Aprēķinus ir izdevīgi organizēt tā, ka īpaši tiek aplūkotas tikai tās rindas, kurās ir kaut viens defekts - par rindām, kurās defektu nav, atbilde jau ir zināma (skat. iepriekš).

Tāpēc nepieciešams vispirms sakārtot visus defektus vispirms augoši pa rindām un tad katrai rindai augoši pa kolonnām.

Pēc tam rūtiņas jāaplūko pēc kārtas, aprēķinot fragmentu garumus un no tiem izzāģējamo dēlīšu skaitu pēc kārtas, kā aprakstīts iepriekš. Papildus nepieciešams korekti veikt pāreju no iepriekšējās rindas uz nākamo, apstrādājot pēdējo iepriekšējās rindas un pirmo nākamās rindas fragmentu. Svarīgi neaizmirst pareizi apstrādāt fragmentu rindas beigās pēc paša pēdējā defekta. Papildus var uzskaitīt, cik rindās ir defekti un atlikušām rindām pieskaitīt vienreiz aprēķināto dēlīšu, ko iespējams izzāģēt no bezdefektu rindas, skaitu.

Kad aprēķināts maksimālais izzāģējamo dēlīšu skaits horizontālā virzienā, plāksni var "pagriezt", samainot vietām laukuma dimensijas un visu defektu koordinātas (skat. 37. zīm.), un jau šai plāksnei pielietot aprakstīto algoritmu.



37. zīm. Plāksnes pagriešana.

Alu labirints

Šī uzdevuma risināšanā ir izdevīgi izmantot grafa apstaigāšanu plašumā sākot no ieejām. Acīmredzami, ka izkļūšanai no kādas zāles līdz tuvākajai ieejai būs nepieciešams tāds pats laiks (šķērsojamo tuneļu skaits), kāds šī maršruta veikšanai pretējā virzienā - no ieejas līdz konkrētajai zālei. Papildus varam izmantot informāciju, kas zināma par zāļu un ieeju numerāciju. Varam glabāt informāciju tikai par zālēm un, jau datu ielasīšanas laikā atzīmēt, ka, ja tunelis savieno ieeju ar zāli, tad no šīs zāles līdz ieejai var tikt vienā laika vienībā. To zāļu numurus, par kuriem izkļūšanas no labirinta laiks jau ir zināms, varam likt rindā un, apstrādājot šīs zāles pēc kārtas to ielikšanas secībā, arī tiks veikta labirinta apstaigāšana plašumā - t.i., pirmās tiks aplūkotas zāles no kurām izkļūšanas laiks būs mazāks.

Atrisinājuma izpildes laika sarežģītība ir $O(Z + E)$.

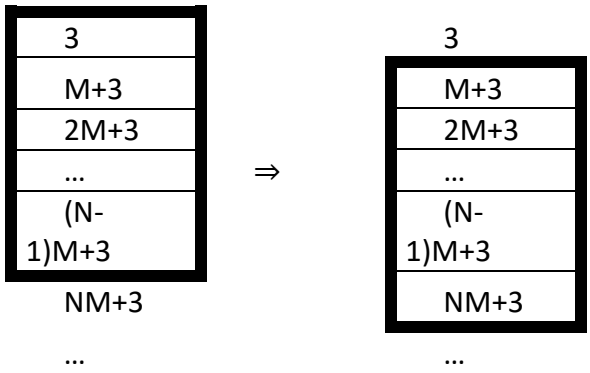
Maucis

Dotās pērlītes sakārtosim M kolonnas platā joslā:

1	2	3	…	M
M+1	M+2	M+3	…	2M
2M+1	2M+2	2M+3	…	3M
…	…	…	…	…
(N-1)M+1	(N-1)M+2	(N-1)M+3	…	NM
NM+1	NM+2	NM+3	…	…
…	…	…	K	

Katrā kolonnā aprēķināsim, cik liels ir kļūdu skaits, ja šajā kolonnā ir paņemtas tieši N secīgas pērlītes - N pērlīšu stabiņš. Lai to noskaidrotu, nepieciešams zināt, kādas un cik pērlītes ir pārstāvētas starp šīm N pērlītēm. Ja būs zināms katras krāsas pērlīšu skaits, tad varēs aprēķināt kļūdu skaitu, kāds ir šim N pērlīšu stabiņam. Turklāt ievērosim, ka, ja šis katras krāsas pērlīšu skaits ir zināms iepriekšējam N pērlīšu stabiņam, tad, aplūkojot N pērlīšu stabiņu, kas sākas vienu pērlīti zemāk, būs mainījušās tikai divas pērlītes - pērlīte ar iepriekš mazāko indeksu vairs nav starp aplūkojamajām, bet tās vietā ir nākusi klāt jauna pērlīte ar lielāko indeksu (par M lielāks, salīdzinot ar iepriekš lielāko).

Piemēram, trešajā kolonnā, nomainot trešo pērlīti pret NM+3-šo, no N pērlīšu stabiņa, kas sākās ar trešo pērlīti, tiks iegūts N pērlīšu stabiņš, kas sākas ar M+3-šo pērlīti:



Tagad doto pērlīšu virkni uzlūkosim kā joslu N pērlīšu augstumā, pirmajā rindā liekot visas tās pērlītes, kuras var būt izvēlētā laukuma pirmajā rindā. Šo pērlīšu indeksi ir robežās no 1 līdz K-(N-1)M. Ievērojiet, ka viena un tā pati pērlīte šajā tabulā var būt atzīmēta vairākās vietās. Piemēram, ja K>MN, tad M+1-ā pērlīte maucī var atrasties gan pirmajā, gan otrajā rindā, gan nebūt vispār, atkarībā no tā, kura pērlīte būs izvēlēta kā mauča sākums:

1	2	3	…	M	M+1	M+2	M+3	…	K-(N-1)M
M+1	M+2	M+3	…	2M	2M+1	2M+2	2M+3	…	…
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
(N-1)M+1	(N-1)M+2	(N-1)M+3	…	NM	NM+1	NM+2	NM+3	…	K

Būtiska šīs tabulas īpašība ir, ka, ja izvēlēsimies tieši M secīgas kolonnas, tad iegūsim kādu sākotnējās pērlīšu virknes fragmentu, kur pērlītes sakārtotas pa M kolonnām un N rindām un atbilst vienam iespējamam maucim.

Aplūkojot šo tabulu, var redzēt, ka visās kolonnās ir N secīgu pērlīšu stabiņi, kuriem kļūdu skaits ir sarēķināts iepriekš un atliek atrast starp tiem M secīgus stabiņus, kuros kļūdu summa ir mazākā iespējamā. To var paveikt, sasummējot kļūdu skaitu pirmajās M kolonnās un tad šo skaitu pārrēķinot: pieskaitot nākamās kolonnas kļūdu skaitu un atņemot iepriekš pirmās kolonnas kļūdu skaitu.

Aprakstītā risinājuma izpildes laika sarežģītība ir $O(\big(K - M(N-1)\big)(2 + logN))$.

Mediāna

Uzdevuma būtība: Dotas skaitļu virknes $X(x_1, x_2, \dots, x_{n_1}), Y(y_1, y_2, \dots, y_{n_2})$. Zināms, ka $n_1 + n_2 \equiv 1\ (mod\ 2)$.

Uz dotajām skaitļu virknēm tiek izpildīti V vaicājumi:

Skaitļu virknē X vai Y visas vērtības pareizināt ar -1.

Skaitļu virknē X vai Y visām vērtībām pieskaitīt vērtību δ.

Nepieciešams noteikt virkņu X, Y apvienojuma mediānas vērtību.

Risinājums:

Skaitļu virknes X, Y sakārto nedilstošā secībā.

Lai ātri apstrādātu operācijas, izmanto šādus mainīgos:

$$X_{minus} = 1; Y_{minus} = 1; X_{add} = 0; Y_{add} = 0$$

$minus$ mainīgie glabā informāciju par to, ar ko attiecīgās virknes elementu vērtības ir jāpareizina, add mainīgie glabā informāciju par to, cik liela vērtība ir jāpieskaita attiecīgās virknes elementu vērtībām.

Tad operācijas var efektīvi izpildīt sekojoši:

Skaitļu virknē X vai Y visas vērtības pareizināt ar -1.

Lai izpildītu šo operāciju, izmaina papildus pieglabāto mainīgo vērtības:

Ja skaitļu virknei X:

$$X_{minus} = X_{minus} \times (-1)$$
$$X_{add} = -X_{add}$$

Ja skaitļu virknei Y:

$$Y_{minus} = Y_{minus} \times (-1)$$
$$Y_{add} = -Y_{add}$$

Izpildes laika sarežģītība: $O(1)$, konstanta.

Skaitļu virknē X vai Y visām vērtībām pieskaitīt vērtību δ.

Lai izpildītu šo operāciju, izmaina papildus pieglabāto mainīgo vērtības:

Ja skaitļu virknei X:

$$X_{add} = X_{add} + \delta$$

Ja skaitļu virknei Y:

$$Y_{add} = Y_{add} + \delta$$

Izpildes laika sarežģītība: $O(1)$, konstanta.

Noteikt mediānas vērtību virkņu X, Y apvienojumam.

Lai efektīvi realizētu šo operāciju, ir nepieciešams ātri iegūt i-tā mazākā elementa vērtību kādā no virknēm. To var izdarīt sekojoši:

Ja virknes $minus$ mainīgā vērtība:

ir 1, tad paņem sākotnēji sakārtotās virknes i-tā elementa vērtību,

Ir -1, tad paņem sākotnēji sakārtotās virknes i-tā no beigām elementa vērtību.

Vērtība jāņem no virknes beigām, jo virkni pareizinot ar -1 pirmajā operācijā virknes elementu secība mainās uz pretējo, savukārt pieskaitīšanas operācija virknes elementu secību nemaina.

Iegūto vērtību pareizina ar virknes $minus$ papildus pieglabātā mainīgā vērtību.

Iegūto vērtībai pieskaita virknes add papildus pieglabātā mainīgā vērtību.

Izpildot šos soļus ir iegūta i-tā mazākā virknes elementa vērtība ar līdz šajam brīdim izpildīto operāciju izmaiņām.

Pieņemsim, ka mediānas elements eksistē virknē X.

Lai realizētu operāciju, jāveic binārā meklēšana pār virknes X elementiem meklējot mediānas elementu:

Binārajā meklēšanas iterācijā, apskatot virknes X i-to mazāko elementu, ir zināma tā vērtība $value_{Xi}$.

Ar bināro meklēšanu virknē Y meklē elementu skaitu (pozīciju), kas ir **mazāki** par $value_{xi}$, atrasto elementu skaitu (pozīciju) sauksim par j_{less},

Ar bināro meklēšanu virknē Y meklē elementu skaitu (pozīciju), kas ir **mazāki vai vienādi** ar $value_{\text{xi}}$, atrasto elementu skaitu (pozīciju) sauksim par j_{lesseq}.

Tad varam secināt, vai virknes X i-tais elements ir mediānas elements, pārbaudot vai $i + j_{less} \leq \frac{n_1+n_2+1}{2} \leq i + j_{lesseq}$. Ja nevienādība neizpildāts, tad turpina bināro meklēšanu kreisajā, labajā diapazonā atkarībā no tā, kura nevienādība neizpildās.

Ja mediānu neizdevās atrast, tad pieņēmums, ka mediānas elements eksistē virknē X bija aplams, tas eksistē virknē Y, tad meklē mediānas elementu virknē Y izmantojot analoģisku algoritmu.

izpildes laika sarežģītība: $O(\log n_1 \log n_2)$

Kopējā izpildes laika sarežģītība: $O(V \log n_1 \log n_2)$

0 un 1

Novērojumi. Sākumā pierādīsim sekojošu apgalvojumu: visas bināras virknes garumā K kā apakšvirknes var atrast tad un tikai tad, ja dotajā virknē ir K divu dažādu secīgu simbolu fragmenti (t.i., vai nu 01, vai nu 10), kas nepārklājas. Piemēram, virknē 01001100011 ir sastopama jebkura binārā apakšvirkne garumā 4, jo mēs varam atrast komplektu ar 4 tādiem fragmentiem: [01]0[01][10]0[01]1.

Pierādījumu veiksim divos virzienos.

Pirmkārt, pieņemsim, ka dotajā virknē ir K tādi fragmenti. Tad, ja mums vajag atrast bināru apakšvirkni $\overline{b_1 b_2 \dots b_K}$, tad no i-tā pēc kārtas fragmenta mēs varam izvēlēties ciparu b_i. Līdz ar to jebkuru bināru apakšvirkni garumā K mēs atrast varam.

Tagad pierādīsim otro virzienu, t.i., ka obligāti vajag K tādus fragmentus, lai varētu atrast jebkuru bināru apakšvirkni garumā K. Pierādīsim to ar *matemātisko indukciju* pēc K.

Indukcijas bāze ir $K = 1$. Tad skaidrs, ka virknē noteikti jābūt vismaz vienai 0 un vismaz vienam 1. Vismaz vienā vietā 0 un 1 tad ir blakus, paņemam to kā fragmentu.

Pieņemsim, ka pierādīts pie K, un pierādīsim pie $K + 1$. Pieņemsim, ka mums ir virkne, kurā kā apakšvirknes atrodas visas binārās virknes garumā $K + 1$.

Aplūkosim virknes pēdējo simbolu; pieņemsim, ka tas ir 0 (gadījums, kad tas ir 1, ir analoģisks). Tad aplūkosim pēdējo 1 šajā virknē; tā kā tas nav pēdējais simbols, tad nākamais simbols pēc tā ir 0. Iegūstam šādu situāciju:

[... virknes sākums ...] 10 ... 0

Bet tā kā šajā virknē var atrast jebkuru bināru virkni garumā $K + 1$, tad virknē [... virknes sākums ...] var atrast jebkuru bināru virkni garumā K kā apakšvirkni (lai kopā ar pēdējo vieninieku varētu sastādīt jebkuru bināru apakšvirkni garumā $K + 1$ ar pēdējo simbolu 1).

Tagad izmantojam indukcijas pieņēmumu, un secinām, ka virknē [... virknes sākums ...] ir K nepārklājošies fragmenti no diviem dažādiem simboliem. Kā $(K + 1)$-o fragmentu paņemam dotās virknes pēdējo 1 un nākamo 0. Apgalvojums ir pierādīts.

Algoritms. No pierādītā seko arī efektīvs algoritms.

Sākumā izejam cauri masīvam un atrodam maksimālo skaitu fragmentu no diviem dažādiem simboliem, kuri nepārklājas. Atzīmējam visas pozīcijas tajās kā "izmantotās". Šo soli var darīt *alkatīgi*: Ejam masīvam no kreisās puses uz labo, un, ja sastopam tādu fragmentu, uzreiz atzīmējam tā abas pozīcijas kā izmantotās. Pieņemsim, ka šajā solī atrasto fragmentu skaits ir S.

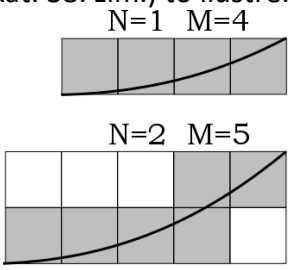
Ja $S \geq K$, tad virkne jau atbilst prasībām un nekas nav jādara. Pretējā gadījumā mums vēl jadabū $R = K - S$ fragmenti. Aplūkojam virknes neizmantotos simbolus, pieņemsim, ka to skaits ir T). Paņemam $\min(R, T)$ no tiem un katram pierakstām blakus pretējo simbolu. Ja $T < R$, tad mums vēl jādabū $R - T$ fragmenti. Tos vienkārši varam pierakstīt virknes galā.

Risinājuma izpildes laika sarežģītība. Risinājumā tikai vairākas reizes izskrienam cauri dotajai virknei. Līdz ar to risinājuma izpildes laika sarežģītība ir $O(N)$.

Parabola

Novērojumi. Sākumā veiksim šādu novērojumu: ja parabola nešķērsotu nevienu rūtiņas stūri lapas iekšienē, tad atbilde būtu $N + M - 1$. Pamatojums tam ir šāds: parabola virzas tikai uz augšu un pa labi. Kad parabola šķerso malu starp divām rūtiņām, parabola sāk "apmeklēt" jaunu rūtiņu. Savukārt lai tiktu no stūra $(0,0)$ uz stūri (N, M), tai ir jākrusto $N - 1$ horizontālas malas starp rūtiņām un $M - 1$ vertikālas malas starp rūtiņām. Turklāt sākumā parabola "apmeklē" rūtiņu $(0,0)$. Līdz ar to kopējais apmeklēto rūtiņu skaits ir $(N - 1) + (M - 1) + 1 = N + M - 1$.

Abi piemēri uzdevuma formulējumā (skat. 38. zīm.) to ilustrē:



38. zīm.

Savukārt, ja parabola šķērso kādu stūri lapas iekšienē, tad tā vienlaicīgi krusto gan vienu horizontālu, gan vienu vertikālu malu. Līdz ar to, vispārīgajā gadījumā no $N + M - 1$ ir jāatņem lapas iekšējo stūru skaitu, kurus šķērso parabola.

Ar ko ir vienāds šis skaits? Tie ir visi tādi punkti (x, y), ka $N\left(\frac{x}{M}\right)^2 = \frac{N}{M^2} \cdot x^2$ ir vesels skaitlis, kur $x \in [1, M - 1]$. Mēģināsim izrēķināt šo lielumu.

Pieņemsim, ka p ir kāds pirmskaitlis, kas dala M. Tad pieņemsim, ka maksimālā p pakāpe, kas dala M, ir a, bet maksimālā p pakāpe, kas dala N, ir b. Ja $\frac{N}{M^2} \cdot x^2$ ir vesels skaitlis, tad x^2 ir jādalās ar $\frac{p^{2a}}{p^b} = p^{2a-b}$. Tas nozīmē, ka x ir jādalās ar $p^{\left\lceil \frac{b-2a}{2} \right\rceil}$.

Apzīmēsim ar A skaitli, ko iegūst, sareizinot visus tādus $p^{\left\lfloor \frac{b-2a}{2} \right\rfloor}$ (kur p ir visi pirmskaitļi, ar kuriem dalās M). Tad $\frac{N}{M^2} \cdot x^2$ ir vesels skaitlis tad, ja x dalās ar A. Savukārt intervālā $[1, M - 1]$ ir tieši $\left\lfloor \frac{M-1}{A} \right\rfloor$ skaitļi, kuri dalās ar A: tie ir A, $2A$, ..., $\left\lfloor \frac{M-1}{A} \right\rfloor A$. Līdz ar to atbilde uz uzdevumu ir $N + M - 1 - \left\lfloor \frac{M-1}{A} \right\rfloor$.

Algoritms. Uzdevums reducējas uz to, lai izrēķinātu skaitli A. Lai to izdarītu, pielāgosim klasisko algoritmu, kas sadala doto skaitli pirmreizinātājos.

Sākumā uzstādām $A = 1$. Tad pārlasām skaitli p no 1 līdz $\lfloor \sqrt{M} \rfloor$, un pārbaudām, vai tas dala M. Ja tas dala M, tad dalām M ar p, līdz tas vairs nedalās. Dalīšanas reižu skaits ir tieši p pakāpe a. Līdzīgi dalām N ar p, lai iegūtu pakāpi b. Pareizinām A ar $p^{\left\lceil \frac{b-2a}{2} \right\rceil}$.

Ievērojam, ka M var dalīties ar maksimums vienu pirmskaitli, kas ir lielāks par $\lfloor\sqrt{M}\rfloor$. Tiešām, ja būtu divi tādi pirmskaitļi p un q (iespējams, vienādi), tad M dalītos ar $pq > M$, pretruna. Līdz ar to tas, kas palika no M, ir vai nu 1, vai nu viens pirmskaitlis, jo ar visiem pirmskaitļiem, kas nav lielāki par $\lfloor\sqrt{M}\rfloor$, mēs M jau esam izdalījuši.

Tātad, ja palikušais M nav 1, tad tas ir kaut kāds pirmskaitlis p. Līdzīgi kā iepriekš, atrodam vajadzīgās pakāpes a un b, un piereizinām A skaitli $p^{\left\lfloor\frac{b-2a}{2}\right\rfloor}$.

Risinājuma ātrdarbība. Risinājumā dominē cikls, kurā mēs pārlasām skaitļus no 1 līdz $\lfloor\sqrt{M}\rfloor$. Līdz ar to kopējā risinājuma ātrdarbība ir $O(\sqrt{M})$.

Punkti taisnstūrī

Ja iekrāsosim visas rūtiņu virsotnes pamīšus baltā un melnā krāsā (katrai baltajai virsotnei visas blakus virsotnes ir melnas, un katrai melnajai virsotnei visas blakus virsotnes ir baltas), tad var ievērot, ka visi baltie punkti, kas atrodas dotā taisnstūra iekšpusē, veido taisnstūra režģi un arī visi melnie punkti veido taisnstūra režģi. Tā kā rūtiņas ir ar izmēru 1*1 tad katras rūtiņas diagonāles garums ir $\sqrt{2}$ – un tas ir arī attālums starp blakus punktiem katrā no vienkrāsainajiem režģiem. Tad atlicis aprēķināt katra vienkrāsainā režģa izmēru. To var izdarīt ar šādām formulām – režģim kuram pieder arī centrālais punkts:$2\left\lfloor\frac{a}{\sqrt{8}}\right\rfloor + 1$ un $2\left\lfloor\frac{b}{\sqrt{8}}\right\rfloor + 1$, un otram režģim: $2\left\lfloor\frac{a+\sqrt{2}}{\sqrt{8}}\right\rfloor$ un $2\left\lfloor\frac{b+\sqrt{2}}{\sqrt{8}}\right\rfloor$.

Risinājuma izpildes laika sarežģītība ir $O(1)$.

Puzle

Lai aprēķinātu, cik kopā veiktas darbības, aprēķināsim cik darbības ir veiktas ar katru no gabaliņiem. Ar katru gabaliņu veikto darbību skaits ir vienāds ar to, cik reizes tiks iziets cauri visiem palikušajiem kauliņiem, līdz tiks paņemts šis kauliņš. Mēģināsim noskaidrot, kādā gadījumā gabaliņš tiks pievienots jau saliktajam puzles fragmentam. Gabaliņš tiek pievienots, ja kāds no tā kaimiņu gabaliņiem jau iepriekš ir pievienots saliktajam puzles fragmentam. Tas nozīmē, ka mēs šo uzdevumu varam pārformulēt kā grafa uzdevumu. Grafa virsotnes ir puzles gabaliņi un orientēta šķautnes ir starp diviem gabaliņiem, ja tie ir blakus viens otram. Šķautnes (a, b) svars ir 0, ja gabaliņš a atrodas virknē pirms b (tas nozīmē ka b tiks pievienots saliktajam fragmentam agrāk vai tajā pašā gabaliņu apstrādes reizē kad a tiks pievienots saliktajam fragmentam). Šķautnes (a, b) svars ir 1, ja gabaliņš a atrodas virknē pēc b (b tiks pievienots saliktajam puzles fragmentam nākamajā gabaliņu apstrādes reizē). Tātad mums ir orientēts grafs kur visu šķautņu svari ir 0 vai 1, zināms sākuma gabaliņš (kurš tiks pielikts pirmajā apstrādes reizē) un nepieciešams aprēķināt īsākos attālumus līdz visām virsotnēm, kam var lietot kādu klasisku grafu algoritmu. Lai aprēķinātu šīs vērtības, var lietot arī mazliet modificētu grafa apstaigāšanas plašumā algoritmu.

Risinājuma izpildes laika sarežģītība ir $O(NM)$.

Viktorīna

Šī uzdevuma risināšanas metodi vienkārši var raksturot kā “gudra simulācija” - t.i., secīgi jālasa dati, tie optimāli jāsaglabā un beigās jāizvada prasītais rezultāts.

Pirms katras grupas datu ievades un apstrādes ir jāzina, kāds ir nākamā dalībnieka, kurš atbildēs uz kārtējo jautājumu, indekss. Skaidrs, ka, ja grupā ir daudz jautājumu, tad vienkārša skaitīšana pa vienai atbildei būs pārāk laikietilpīga. Tādēļ jāizmanto veselo skaitļu dalīšana, nosakot, cik pilnus ciklus visi dalībnieki ir atbildējuši vienādi (pareizi vai nepareizi). Pareizo atbilžu gadījumā iegūtie punkti līdz ar papildpunktiem veidos kopējo punktu “fonu”, kas beigās būs jāpieskaita visiem dalībniekiem.

Nepareizo atbilžu gadījumā pilno ciklu skaitu var atmest, jo nepareizo atbilžu gadījumā ir svarīgi tikai atrast nākamo dalībnieku, ar kuru sāksies pareizo atbilžu josla.

Tagad atliek noskaidrot, kā vislabāk saglabāt informāciju par secīgām pareizām atbildēm. Raksturīgi, ka secīgu pareizu atbilžu skaits var būt visai liels, kas var novest pie tā, ka lielam skaitam dalībnieku ir piešķirti punkti par K pareizām atbildēm un papildpunkts, bet šādu dalībnieku apstrāde pa vienam atkal var būt laikietilpīga.

Tādēļ labāk ir saglabāt punktu skaita izmaiņu salīdzinot ar iepriekšējo dalībnieku.

Piemēram, ja zināms, ka visi dalībnieki no trešā līdz devītajam ir saņēmuši pa M punktiem, tad masīvā varam palielināt trešā elementa vērtību par M, bet desmitā elementa vērtību par M samazināt. Ja punktu skaits mainījies vienam dalībniekam, tad par šo punktu skaitu palielinām attiecīgā masīva elementa vērtību, bet nākamā elementa vērtību par šo lielumu samazinām. Jāņem vērā, ka dalībnieku indeksi var šķērso robežu pēc N-tā dalībnieka uz pirmo.

Visbeidzot, lai aprēķinātu katra dalībnieka iegūto punktu summu, jāsasummē visas punktu izmaiņas no pirmā līdz šim dalībniekam un jāpieskaita iepriekš minētie fona punkti.

Aprakstītā risinājuma izpildes laika sarežģītība ir $O(G + N)$.

2017./2018. mācību gads

Novada olimpiāde - 2018

Apakštaisnstūri

Šim uzdevumam eksistē acīmredzams un neefektīvs risinājums - ciklā izvēlamies taisnstūrveida apgabala kreiso augšējo un labo apakšējo rūtiņu, aprēķinām skaitļu summu šajā apgabalā un, ja tā atrodas vajadzīgajās robežās, tad pieskaitām derīgo taisnstūru skaitam 1.

Efektīvam risinājumam pārlase jāorganizē gudrāk.

Vispirms, jau ielasot, glabāsim nevis pašas tabulas vērtības, bet skaitļu summu sākot no rindas sākuma. Uzdevumā dotajam piemēram ielasītā tabula būtu šāda:

1	1	3	6
6	13	13	16
3	3	7	9

Pēc tam saturīgi notiek tas, kas tika izmantots neefektīvajā risinājumā - izvēlamies tabulas sākuma rindu (cikls no 1 līdz N) un aplūkojam visas iespējamās tabulas beigu rindas (cikls no sākuma rindas līdz N).

Šī pārlase tiek organizēta izmantojot viendimensiju masīvu v, kurā sākumā ielasa un sākuma rindas vērtības (kolonnu numerācija no 1, bet $v_0 = 0$), bet pēc tam šim masīvam pakāpeniski pieskaita nākamo rindu saturu.

Ciklā tiek aplūkota sākuma kolonna k un tiek noskaidrots, cik elementiem šajā viendimensiju masīvā summas vērtība ir derīga.

Katrā brīdī tiek izmantoti divi rādītāji: r_A - pirmās kolonnas indekss, kur elementu summa ir vismaz $A + v_{k-1}$, un r_B - pirmās kolonnas indekss, kur elementu summa ir lielāka par $B + v_{k-1}$. Tad derīgo apgabalu skaits sākuma kolonnai k ir vienāds ar $r_B - r_A$. Svarīgi ir pamanīt, ka palielinot sākuma kolonnas k vērtību r_A un r_B vērtības nevar samazināties. Tas nozīmē, ka masīva v elementi (kopskaitā M) ar rādītāju palīdzību tiek caurskatīti vienreiz un šajā mehānismā nav ieslēpti papildus cikli.

Aplūkosim, kā šis algoritms darbojas uzdevuma tekstā dotajā piemērā (N=3, M=4):

1) `sākuma_rinda=1, beigu_rinda=1`

v[0]	v[1]	v[2]	v[3]	v[4]
0	1	1	3	6

$k = 1 \Rightarrow r_A = 5, r_B = 5$ Tā kā $r_A > M$, tad šai beigu rindai derīgu taisnstūru nebūs - ejam ārā no kolonnu cikla.

2) `sākuma_rinda=1, beigu_rinda=2`

v[0]	v[1]	v[2]	v[3]	v[4]
0	7	14	16	22

$k = 1 \Rightarrow r_A = 1, r_B = 2$ Derīgo variantu skaits 2-1=1.

$k = 2 \Rightarrow r_A = 2, r_B = 4$ Derīgo variantu skaits 4-2=2.

$k = 3 \Rightarrow r_A = 4, r_B = 5$ Derīgo variantu skaits 5-4=1.

$k = 4 \Rightarrow r_A = 5, r_B = 5$ Tā kā $r_A > M$, izeja no kolonnu cikla.

3) `sākuma_rinda=1, beigu_rinda=3`

v[0]	v[1]	v[2]	v[3]	v[4]
0	10	17	23	31

$k = 1 \Rightarrow r_A = 1, r_B = 1$ Derīgo variantu skaits 1-1=0.

$k = 2 \Rightarrow r_A = 2, r_B = 3$ Derīgo variantu skaits 3-2=1.

$k = 3 \Rightarrow r_A = 4, r_B = 4$ Derīgo variantu skaits 4-4=0.
$k = 4 \Rightarrow r_A = 4, r_B = 5$ Derīgo variantu skaits 5-4=1.

4) `sākuma_rinda=2, beigu_rinda=2`

v[0]	v[1]	v[2]	v[3]	v[4]
0	6	13	13	16

$k = 1 \Rightarrow r_A = 2, r_B = 2$ Derīgo variantu skaits 2-2=0.
$k = 2 \Rightarrow r_A = 2, r_B = 4$ Derīgo variantu skaits 4-2=2.
$k = 3 \Rightarrow r_A = 5, r_B = 5$ Tā kā $r_A > M$, izeja no kolonnu cikla.

5) `sākuma_rinda=2, beigu_rinda=3`

v[0]	v[1]	v[2]	v[3]	v[4]
0	9	16	20	25

$k = 1 \Rightarrow r_A = 1, r_B = 2$ Derīgo variantu skaits 2-1=1.
$k = 2 \Rightarrow r_A = 2, r_B = 3$ Derīgo variantu skaits 3-2=1.
$k = 3 \Rightarrow r_A = 4, r_B = 5$ Derīgo variantu skaits 5-4=1.
$k = 4 \Rightarrow r_A = 5, r_B = 5$ Tā kā $r_A > M$, izeja no kolonnu cikla.

6) `sākuma_rinda=3, beigu_rinda=3`

v[0]	v[1]	v[2]	v[3]	v[4]
0	3	3	7	9

$k = 1 \Rightarrow r_A = 3, r_B = 5$ Derīgo variantu skaits 5-3=2.
$k = 2 \Rightarrow r_A = 5, r_B = 5$ Tā kā $r_A > M$, izeja no kolonnu cikla.

Aprakstītā algoritma izpildes laika sarežģītība ir $O(N^2M)$.

Ciparu aizvietošana

Risinājumam ir divas relatīvi neatkarīgas daļas - A) dažādo skaitļu ar doto pierakstu skaita atrašana un B) mazākā skaitļa ar šo pierakstu atrašana.

Vispirms atrisināsim uzdevumu A.

Lai to paveiktu, caurskatīsim simbolu virkni no beigām un katrā pozīcijā saglabāsim masīvu garumā 10, atzīmējot, vai šajā vietā var sākties katrs no desmit decimālajiem cipariem. Apstrāde no beigām ir izdevīga, jo var izmantot iepriekš apstrādāto informāciju.

Ja kārtējā pozīcijā ir 0, tad šajā pozīcijā var sākties tikai 0. Visu pārējo ciparu pierakstu apstrādei jau zinām, ka pirmais cipars ir 1. Piemēram, lai noteiktu, vai šajā pozīcijā var sākties decimālā cipara 7 pieraksts, pietiek pārbaudīt, ka nākamajā pozīcijā var sākties 3.

Uzdevuma tekstā dotajai virknei 100111 šo masīvu vērtības būtu šādas:

	1	0	0	1	1	1
0	nē	jā	jā	nē	nē	nē
1	jā	nē	nē	jā	jā	jā
2	jā	nē	nē	nē	nē	nē
3	nē	nē	nē	jā	jā	nē
4	jā	nē	nē	nē	nē	nē
5	nē	nē	nē	nē	nē	nē
6	nē	nē	nē	nē	nē	nē
7	nē	nē	nē	jā	nē	nē
8	nē	nē	nē	nē	nē	nē
9	jā	nē	nē	nē	nē	nē

Zinot šo informāciju, varam pakāpeniski aprēķināt dažādo variantu skaitu, atkal sākot apstrādi no rindas beigām. Ar s_i apzīmēsim dažādo skaitļu skaitu, kas sākas virknes pozīcijā $i (1 \leq i \leq G)$. Atļaujam skaitļiem virknes vidū sākties arī ar 0 (no dotā ir zināms, ka visas virknes pirmais simbols būs 1). Uzskatām, ka tukšo virkni varam izveidot vienā veidā ($s_{G+1} = 1,\ s_{G+2} = s_{G+3} = s_{G+4} = 0$).

Tālāk varam izmantot sakarības:

- $S_G = S_{G+1}$
- ja pozīcijā i var sākties 2 vai 3 pieraksts, tad $S_G = S_G + S_{G+2}$
- ja pozīcijā i var sākties 4, 5, 6 vai 7 pieraksts, tad $S_G = S_G + S_{G+3}$
- ja pozīcijā i var sākties 8 vai 9 pieraksts, tad $S_G = S_G + S_{G+4}$

Mūsu piemēra virknei dažādo skaitļu skaita vērtības mainītos šādi:

indekss	1	2	3	4	5	6	7	8	9	10
cipars	1	0	0	1	1	1				
s_i	14	4	4	4	2	1	1	0	0	0

Jāatceras, ka summa jāaprēķina pēc moduļa, tāpēc katrā solī vienkāršas saskaitīšanas vietā jāsaskaita un jāpaņem atlikums pēc norādītā moduļa.

Līdz ar to uzdevums A ir atrisināts.

Uzdevuma B atrisināšanai vispirms ievērosim, ka decimālam ciparam ar lielāku vērtību atbilst tikpat gara vai garāka bināro ciparu virkne. Tāpēc mazāko iespējamo skaitli varam meklēt no pieraksta beigām katru reizi ņemot lielāko iespējamo decimālo ciparu - to, kura binārais pieraksts ir visgarākais. Tas nodrošinās, ka skaitļa decimālajā pierakstā ir mazākais iespējamais ciparu skaits un pats skaitlis ir ar vismazāko vērtību.

Lai šo procesu varētu ērti organizēt, nepieciešams papildināt iepriekš aplūkoto masīva "vai šeit var sākties attiecīgā cipara pieraksts" aizpildīšanu ar viendimensiju masīva "kāda ir lielākā decimālā cipara vērtība, kas beidzas šajā pozīcijā, vērtība" aizpildīšanu.

To ir vienkārši realizēt - brīdī, kad zinām, ka pozīcijā i var sākties decimālā cipara c pieraksts, varam atzīmēt, ka lielākais cipars, kas beidzas pozīcijā $i + c\_garums - 1$ ir c. Ņemot vērā to, ka binārās virknes apstrāde noris no beigām uz sākumu, lielākā cipara vērtība noteiktā pozīcijā var tikai palielināties. Svarīgi, ka noteiktā pozīcijā ar vienu un to pašu garumu var beigties tikai viena decimālā cipara pieraksts un nav tādas pozīcijas, kurā nebeidzas neviens cipars.

Mūsu piemērā masīva "kāda ir lielākā decimālā cipara vērtība, kas beidzas šajā pozīcijā, vērtība" elementi būtu šādi:

indekss	1	2	3	4	5	6
binārais cipars	1	0	0	1	1	1
lielākais decimālais cipars, kas šeit beidzas	1	2	4	9	3	7

Decimālā skaitļa pēdējais cipars ir 7 (lielākais, kas beidzas 6. pozīcijā). Šis decimālais cipars aizņem trīs binārās pozīcijas, tāpēc kā nākamā jāaplūko 3. pozīcija. Tātad priekšpēdējais decimālais cipars ir 4. Esam "iztērējuši" visus bināros ciparus, tāpēc visi decimālā skaitļa cipari ir atrasti un esam ieguvuši, ka meklētā skaitļa pieraksts ir 47.

Algoritma sarežģītība ir $O(G)$. No realizācijas detaļām interesanti atzīmēt, ka masīva "vai šeit var sākties attiecīgā cipara pieraksts" realizācijai pietiek ar 5×10 elementiem, kuru vērtības cikliski tiek atjauninātas.

Nav apakšvirkne

Šis ir klasisks dinamiskās programmēšanas uzdevums.

Izveidosim $N \times G$ elementu masīvu P, kur $P[burts, i]$ elementa vērtība nozīmēs "virknes, kas sākas ar $burts$, un kam līdz dotās virknes X beigām var atrast visas iespējamās apakšvirknes, lielākais garums".

Šo masīvu aizpildīsim no beigām un ievērosim, ka, ja $burts \neq X[i]$, tad $P[burts, i] = P[burts, i+1]$, bet, ja $burts = X[i]$, tad $P[burts, i] = 1 + \min\limits_{\text{burts}} P[burts, i+1]$.

Lai atvieglotu pirmās neatrodamās apakšvirknes vērtību, katrai i vērtībai ir vērts saglabāt mazāko $burts$ vērtību, kurai atbilstošā $P[burts, i]$ vērtība sakrīt ar mazāko vērtību starp visām $P[j, i]$.

Aplūkosim, kā izskatās masīva P aizpildījums uzdevuma tekstā dotajā piemērā:

	d	a	b	a	c	b	a	d	a	c
a	2	2	2	2	1	1	1	1	1	0
b	2	2	2	1	1	1	0	0	0	0
c	2	2	2	2	2	1	1	1	1	1
d	2	1	1	1	1	1	1	1	0	0
min	**2**	**1**	1	1	1	1	**0**	0	0	0
kurš	**a**	**d**	d	b	a	a	**b**	b	b	a

Aplūkojot vērtības, kas atbilst X pirmajam burtam, varam pārliecināties, ka virknē X var atrast visas apakšvirknes garumā 2, un pirmā neatrodamā virkne būs garumā trīs. Lai to atrastu, ir jāmeklē pēc kārtas pirmās kolonnas, kur mainās kolonnas minimuma vērtība un jāņem alfabēta pirmais atbilstošais burts, kas nosaka šo minimumu.

Aprakstītā risinājuma izpildes laika sarežģītība ir $O(NG)$.

Ja programmā tiek izmantota pāreja no burta uz tā kārtas numuru alfabētā, tad jāatceras, ka šajā uzdevumā tiek izmantots latviešu, nevis angļu alfabēts.

Peļķes

Tā kā plāksnes nedrīkst pārklāties, tad, lai kā arī tās būtu novietotas, būs iespējams (domās) novilkt vertikālu vai horizontālu līniju tā, ka viena no plāksnēm būs viena puse un otra plāksne otra puse.

Turpmāk pieņemsim, ka iedomātā dalījuma līnija būs vertikāla un atradīsim labāko atbildi šajā gadījumā.

Sakārtosim visas peļķu koordinātas (punktus) pēc x koordinātas (ja x koordinātas ir vienādas, y koordināšu secība ir patvaļīga).

Ciklā iesim cauri sakārtoto punktu masīvam un aprēķināsim, kāda būtu atbilde, ja šī peļķe atrastos pie kreisākās plāksnes labās malas.

Lai to aprēķinātu, mums vajag atrast minimālo plāksnes augstumu un platumu, lai pārklātu punktus pa kreisi no iedomātās līnijas un minimālo augstumu un platumu, lai pārklātu punktus pa labi no iedomātās līnijas.

To mēs varam aprēķināt laikā $O(peļķu\ skaits)$ – pieņemsim, ka mēs jau zinām mazāko taisnstūri, kāds nepieciešams, lai pārklātu pirmos i punktus (sakārtotajā peļķu masīvā), tad lai aprēķinātu mazāko taisnstūri kāds nepieciešams, lai pārklātu pirmos $i+1$ punktus ir nākusi klāt tikai viena peļķe un iepriekš zināmajam taisnstūrim varam noskaidrot, vai ir jāmaina kreisā, labā, augšējā vai apakšējā mala, lai pārklātu arī šo punktu un to var izdarīt laikā $O(1)$. Esam noskaidrojuši taisnstūra izmērus kas nepieciešams, lai pārklātus visus punktus kas atrodas pa kreisi no iedomātās līnijas. Līdzīgā veidā varam aprēķināt taisnstūra, kas nepieciešami lai pārklātu punktus pa labi no iedomātās līnijas tikai rēķinot no masīva beigām, izmērus.

Tad katrai iedomātajai līnijai, kurai tagad zinām minimālo kreiso un minimālo labējo taisnstūri, jānoskaidro minimālais kopējais taisnstūris, lai varētu pārklāt abas daļas – jāatceras, ka taisnstūri var būt arī pagriezti.

Risinājuma izpildes laika sarežģītība ir $O(NlogN)$.

Receptes

Vispirms jāielasa pieejamo izejvielu daudzumi un tad ciklā pēc kārtas jāapstrādā visas receptes. Tālāk iztirzāsim vienas receptes apstrādi.

Aplūkosim tikai tās izejvielas, kuras nepieciešamas receptē - t.i., kuru nepieciešamais daudzums nav 0. Lielāko iespējamo toršu skaitu nosaka tā sastāvdaļa, kura starp visām sastāvdaļām pietiek **mazākajam** toršu daudzumam. Tas nozīmē, ka katrai receptei jāatrod un jāizvada mazākā lieluma $\left[\frac{pieejams_i}{receptē_i}\right]$ vērtība, kur $pieejams_i$ ir pieejamais i-tās izejvielas daudzums, bet $receptē_i$ - receptē nepieciešamais šīs izejvielas daudzums.

Risinājuma izpildes laika sarežģītība ir $O(nk)$, kur n - dažādo izejvielu, bet k - dažādo recepšu skaits.

Valsts olimpiāde - 2018

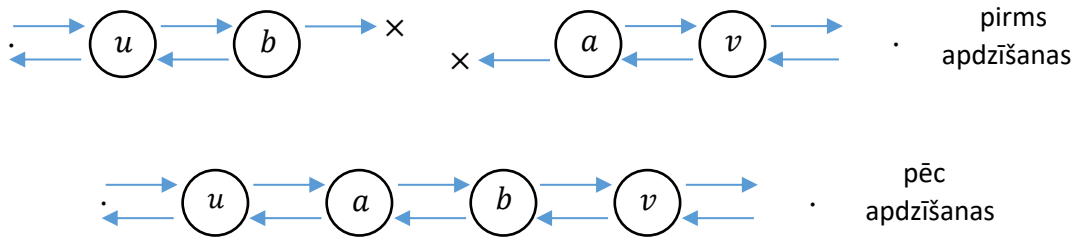
Autosacīkstes-1

Risinājuma ideja ir pakāpeniski simulēt apdzīšanas, tādā veidā būvējot daļēju mašīnu secību. Algoritma gaitā uzturēsim datu struktūru, kas glabās informāciju par visiem mašīnu pāriem, par kuriem mēs noteikti zinām, ka atbilstošās mašīnas dotajā brīdī ir kaimiņi.

Izmantosim datu struktūru *saistītais saraksts*, kur katram elementam papildus glabājas divas norādes - viena uz kreiso kaimiņu, bet otra uz labo kaimiņu. Sākumā katrai no mašīnām no 1 līdz N izveidojam atsevišķu elementu, kura abas norādes ir tukšas (norāda uz "neko"). Tas, ka norādes uz neko nenorāda, attēlo to, ka sākumā neko nezinām par mašīnu secību. Implementācijā izveidojam arī papildus masīvu, kas katrai mašīnai pasaka, kurš elements no saraksta tai pieder.

Tad aplūkojam visas apdzīšanas no sākuma līdz beigām. Pieņemsim, ka kārtējā apdzīšanā mašīna a apdzina mašīnu b. Tad ir iespējami divi gadījumi:

- Mašīnu a un b elementi saistītajā sarakstā nenorāda viena uz otru. Tā kā apdzīšana ir korekta, tas nozīmē, ka gan a elementa kreisā norāde, gan b elementa labā norāde abas norāda uz tukšumu. Tad piešķiram a elementu kā kreiso kaimiņu b elementam, un b elementu kā labo kaimiņu a elementam. Pēc tam apmainām a un b elementus: a tagad piederēs vecais b elements, bet b piederēs a vecais elements (skat. 39. zīm.).



39. zīm.

- Mašīnas a un b saistītajā sarakstā jau norāda viena uz otru. Tajā gadījumā vienkārši samainām a un b elementu piederību.

Pēc visu apdzīšanu apskatīšanas tad mums ir saistītais saraksts, kurā ir vairākas saistītas komponentes. Katra no šīm komponentēm gala secībā definē vienu secīgu mašīnu virkni. Ievērosim, ka šādas virknes

varam brīvi mainīt vietām. Piemēram, ja saistītajā sarakstā ir divas komponentes 143 un 52, tad der gan secība 14352, gan 52143, jo tiek ievēroti visi kaimiņu nosacījumi.

Kā tad izvadīt vienu derīgu secību? Aplūkojam visas mašīnas pēc kārtas no 1 līdz N. Ja kādas mašīnas a elementam nav kreisā kaimiņa (norāde norāda uz "neko"), tad tas nozīmē, ka ar šo mašīnu sākas viena saistīta komponente. Tā kā komponentes varam izvadīt jebkurā secībā, tad apstaigājam visus saistītā saraksta elementus, sākot no a un sekojot norādēm pa labi, paralēli arī izdrukājot mašīnu numurus, kurām šie elementi pieder.

Ātrdarbība. Simulācijas viena soļa izpildes laika sarežģītība ir $O(1)$, pavisam ir M soļi, un beigās secības konstruēšana aizņem $O(N)$. Tāpēc ātrdarbība ir $O(N + M)$.

Autosacīkstes-2

Šajā risinājumā izmantosim to pašu simulāciju, kā uzdevumā "Autosacīkstes-1". Uzdevumā ir iespējami divi gadījumi:

- Ferdinanda pieraksti ir pretrunīgi. Tādā gadījumā atbilde ir 0.
- Ferdinanda pieraksti nav pretrunīgi. Tad aplūkojam saistīto sarakstu simulācijas beigās. Pieņemsim, ka dažādu saistīto komponenšu skaits tajā ir S. Tad atbilde uz uzdevumu ir S dažādu fragmentu permutāciju skaits, kas ir $S!$ (S faktoriāls).

Līdz ar to ir vienīgi pareizi jānosaka, vai pieraksti nav pretrunīgi. Pieņemsim, ka simulācijā aplūkojam apdzīšanu $a\ b$, un līdz tai pieraksti nebija pretrunīgi. Šajā brīdī mums ir saistītajā sarakstā jāsavieno a un b elementi ar norādēm, un jāapmaina to elementu piederība vietām. Tad ir tikai daži gadījumi, kad pieraksts $a\ b$ var izraisīt pretrunu:

1) Ja a un b ir blakus elementi, tad b jābūt pa kreisi no a, jo pirms apdzīšanas b ir priekšā a.
2) Ja a un b nav blakus elementi, tad a kreisajai norādei un b labajai norādei abām jānorāda uz tukšumu, lai šos divus elementus pēc tam varētu savienot. Turklāt abiem a un b jābūt atšķirīgās komponentēs.

Ja kāds no šiem nosacījumiem neizpildās, tad ir pretruna. Nosacījumu 1) var pārbaudīt viegli, aplūkojot a un b kaimiņus.

Ar nosacījumu 2) ir mazliet sarežģītāk. Ja a un b nav blakus elementi, tad ir viegli pārbaudīt, vai a kreisā norāde un b labā norāde norāda uz tukšumu. Bet ir vēl viens pretrunas gadījums – ja a ir vienas komponentes sākums, un b ir tās pašas komponentes beigas. Tādā gadījumā ir pretruna, jo, savienojot a un b, iegūstam ciklu, kas nav iespējams mašīnu secībā.

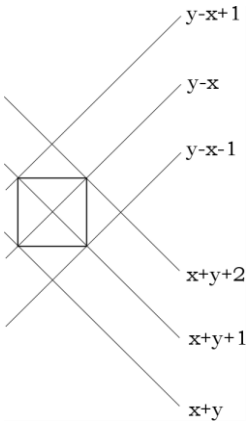
Kā efektīvi pārbaudīt, vai a un b ir vienā komponentē? Priekš tam var izmantot datu struktūru *nešķeļošos kopu sistēma* (angliski *disjoint set union*). Šī struktūra uztur N elementu nešķeļošās apakškopas. Ar $K(a)$ apzīmēsim kopu, kurai pieder a. Tad šai struktūrai ir šādas īpašības:

- Diviem elementiem a un b var laikā $O(\log N)$ pārbaudīt, vai $K(a) = K(b)$.
- Diviem elementiem a un b var apvienot kopas $K(a)$ un $K(b)$ vienā laikā $O(\log N)$.

Mūsu uzdevumā kopas $K(a)$ apzīmē atsevišķas komponentes saistītajā sarakstā. Kad simulācijā a apdzen b, tad mēs apvienojam $K(a)$ ar $K(b)$. Un lai pārbaudītu, vai kārtējā apdzīšana $a\ b$ nesavieno vienu komponenti ciklā, pārbaudām, vai $K(a)$ nav vienāds ar $K(b)$.

Ātrdarbība. Simulācijas ātrdarbība ir $O(N + M)$, un katrā apdzīšanā ir ne vairāk kā divi pieprasījumi kopu sistēmas struktūrai, līdz ar to kopējā izpildes laika sarežģītība ir $O((N + M) + M \log N) = O(N + M \log N)$.

Augļudārza platība



40. zīm.

Vispirms nepieciešams tikt galā ar divām koordinātu sistēmām - ievaddatos ir dotas rūtiņu koordinātas, bet, meklējot ierobežojošos taisnstūrus un kvadrātus, ērtāk darboties ar punktu koordinātām. Vienkāršākais variants ir uzskatīt, ka rūtiņas ar koordinātām (x, y) virsotnes atrodas punktos (x, y), $(x + 1, y)$, $(x + 1, y + 1)$ un $(x, y + 1)$.

Uzdevuma atrisināšana balstās uz ierobežojošo taisnstūru atrašanu katrā no virzieniem - paralēli rūtiņu režģim vai 45° leņķī pret to. Attiecīgā virziena kvadrāti tiks iegūti par to malu garumiem ņemot atrasto taisnstūru garākās malas.

Taisnstūru atrašanas ideja ir atrast pirmo attiecīgā virziena taisni, kas šķērso kādas rūtiņas virsotni - sauksim to par robežtaisni. Pavisam ir astoņas dažādas robežtaisnes, jeb četri paralēlu robežtaišņu pāri. Pēc tam atliek noteikt attālumu starp paralēlajām robežtaisnēm tādejādi iegūstot taisnstūra vienas malas garumu un, visbeidzot, abu malu garumus sareizināt, iegūstot taisnstūra laukumu. Nedrīkst aizmirst, ka vienas vienības garums šoreiz ir divi (nevis viens!) metri - tas bija nepieciešams, lai garantētu, ka taisnstūra laukums vienmēr ir izsakāms kā naturāls skaitlis.

Vispirms aplūkosim gadījumu, ka taisnstūra malas ir paralēlas rūtiņu režģa malām. Šajā gadījumā varam rīkoties vienkārši - atrast mazākās un lielākās x un y koordinātas starp visām kvadrātu virsotnēm: $x_{min}, x_{maks}, y_{min}, y_{maks}$.

Tad taisnstūra laukums ir aprēķināms kā $(x_{maks} - x_{min}) \times (y_{maks} - y_{min}) \times 4$.

Mazliet sarežģītāks ir gadījums, ja taisnstūris ir pagriezts 45° leņķī pret rūtiņu režģi. Lai noteiktu, kuras taisnes veido rūtiņu apgabala robežu, nepieciešams veikt papildu aprēķinus.

Aplūkosim, kā rūtiņu virsotnēs mainās vērtības $y - x$ un $x + y$ (skat. 40. zīm.) Šīs vērtības ir konstantas uz attiecīgā virziena taisnēm un mainās par 1, pārejot uz blakus taisni. Attālums starp divām blakus taisnēm ir vienāds ar $\sqrt{2}$. Tātad, ja būs zināmas $y - x$ (vai $x + y$) vērtības uz ierobežojošajām taisnēm, tad taisnstūra, kas pagriezts 45° leņķī pret rūtiņu režģi, laukumu varēs aprēķināt šādi: $((y - x)_{maks} - (y - x)_{min}) \times ((x + y)_{maks} - (x + y)_{min}) \times 2$

Kā tad atrast šīs ierobežojošās taisnes"?

Vienkāršs veids ir rūtiņu koordinātu ielasīšanas laikā pārbaudīt, vai kārtējā rūtiņa "neizlien" no līdz šim atrasto, taišņu ierobežotā apgabala.

Pirmajai ielasītajai rūtiņai nepieciešams saglabāt robežtaisnes raksturojošās vērtības: x (minimālā vertikāle), $x + 1$ (maksimālā vertikāle), y (minimālā horizontāle), $y + 1$ (maksimālā horizontāle), $y - x - 1$ (minimālā vērtība taisnei 45° leņķī), $y - x + 1$ (maksimālā vērtība taisnei 45° leņķī), $x + y$ (minimālā vērtība taisnei 135° leņķī), $x + y + 2$ (maksimālā vērtība taisnei 135° leņķī).

Katrai nākamajai ielasītajai rūtiņai nepieciešams pārbaudīt, vai attiecīgā tās vērtība nav lielāka par iepriekš zināmo maksimālo vai mazāka par minimālo. Ja tā ir, tad šī pēdējā rūtiņa vienā vai vairākos virzienos atrodas pie robežas (caur tās virsotni vai virsotnēm iet kāda no robežtaisnēm) un jānomaina attiecīgā virziena robežtaisnes skaitliskā vērtība. Visām astoņām robežtaisnēm atbilstošās vērtības iespējams aprēķināt uzreiz datu ielasīšanas gaitā, pašas koordinātas nesaglabājot.

Risinājuma izpildes laika sarežģītība ir lineāra no rūtiņu skaita.

Figūras-1

Tā kā vienai un tai pašai figūrai var atbilst vairāki pieraksta veidi, tad sākumā nepieciešams pierakstus normalizēt. Katrai pierakstītajai figūrai aprēķinām visas rūtiņas (katru vienu reizi), kuras tai pieder un sakārtojam šī rūtiņas sākumā pēc rūtiņu kolonnas numura augošā secībā un, ja tie ir vienādi, tad pēc rindas numura augošā secībā, kā arī pārvietojam figūru tādā veidā, lai katras figūras normalizētais pieraksts sāktos rūtiņā (0,0). Šādā veidā mēs esam ieguvuši figūras pieraksta veidu kas nodrošina īpašību: "figūras ir vienādas savā starpā tad un tikai tad, ja to normalizētie pieraksti (rūtiņu virknes) ir vienādi savā starpā".

Tātad sākumā katrai dotajai figūrai aprēķina tās normalizēto pieraksta veidu, un pēc tam katru normalizēto pieraksta veidu salīdzina ar visiem līdz šim sastaptajiem, lai noteiktu, vai tie ir savā starpā vienādi.

Risinājuma izpildes laika sarežģītība ir $O(50000 \times (N^2 + Nlog50000))$.

Figūras-2

Tā kā vienai un tai pašai figūrai var atbilst vairāki pieraksta veidi, tad sākumā nepieciešams pierakstus normalizēt. Katrai pierakstītajai figūrai aprēķinām visas rūtiņas (katru vienu reizi), kuras tai pieder un sakārtojam šī rūtiņas sākumā pēc rūtiņu kolonnas numura augošā secībā un, ja tie ir vienādi, tad pēc rindas numura augošā secībā, kā arī pārvietojam figūru tādā veidā, lai katras figūras normalizētais pieraksts sāktos rūtiņā (0,0). Izveidojam tādus pašus normalizētos pierakstus arī tad, ja figūra ir pagriezta un / vai atspoguļota. Šādā veidā mēs esam ieguvuši figūras pieraksta veidu, kas nodrošina šādu īpašību – figūras ir vienādas savā starpā tad un tikai tad, ja kādi no to normalizētajiem pierakstiem (rūtiņu virknēm) savā starpā ir vienādi. Ar figūru normalizāciju var iet vēl soli tālāk – salīdzināt visus astoņus normalizētos pierakstus un salīdzināt tos savā starpā un teikt, ka figūras normalizētais pieraksts ir mazākais no tiem visiem tad mēs iegūstam īpašību, ka figūras ir vienādas savā starpā tad un tikai tad, ja to mazāki normalizētie pieraksti savā starpā ir vienādi.

Tātad sākumā katrai dotajai figūrai aprēķina tās normalizēto pieraksta veidu un pēc tam katru normalizēto pieraksta veidu (vai veidus) salīdzina ar visiem līdz šim sastaptajiem un salīdzina, vai tie ir savā starpā vienādi.

Lai padarītu salīdzināšanu ātrāku (lai tas vienmēr nav jādara pa vienai rūtiņai), tad var figūru normalizētajiem pierakstiem pielietot jaucējfunkciju, un sākumā salīdzināt to rezultātus. Pilno salīdzināšanu nepieciešams veikt tikai tad, ja jaucējfunkciju rezultāti ir vienādi.

Risinājuma izpildes laika sarežģītība ir $O(50000 \times (N^2 + Nlog50000))$.

Minibači

Katrs skaitlis, kas parādās virknē, ir kāda šīs virknes secīgu iepriekšējo elementu summa. Varam teikt, ka katrs skaitlis atbilst kādam virknes elementu segmentam. Ja ir fiksēts segmenta labais galapunkts, tad šo segmentu summas virknē var parādīties tikai intervālu garumu pieaugšanas secībā - t.i., vispirms virknē būs skaitlis, kurš atbilst intervālam $[kreisais; labais]$ $(1 < kreisais < labais)$ un tikai pēc tam var parādīties skaitlis, kurš atbilst intervālam $[kreisais - 1; labais]$.

Šajā uzdevumā ļoti svarīga ir datu struktūra, kurā glabāt vēl nepaņemtos skaitļus - virknes locekļu kandidātus. Bez pašas skaitļa (segmenta summas) vērtības ir jāsaglabā arī atbilstošā segmenta kreisā gala indekss $kreisais$. Šīs struktūras īpašībai jābūt spējai no tās ātri dabūt mazāko elementu - virknes nākamo locekli, kā arī relatīvi ātrai jābūt jaunu elementu pievienošanai. Šāda īpašība piemīt *kaudzei*.

Tikko ir zināma kārtējā virknes locekļa a_n $(n > 2)$ vērtība, tā kaudzē tiek pievienots elements ar summu $a_n + a_{n-1}$ un segmenta kreisā gala indeksu $n - 1$. Tad, kad kāda segmenta summa kaudzē būs mazākā un tāpēc būs virknes nākamais loceklis, attiecīgais elements no kaudzes tiks izņemts. Ja

segmenta kreisā gala indekss nebija 1, tad kaudzei jāpievieno elements ar vērtību $vecā\_segmenta\_summa + a_{kreisais-1}$, samazinot segmenta kreisā gala indeksu par 1. Jāņem vērā, ka, ja ar vienu un to pašu summu ir vairāki elementi, tad tie visi no kaudzes ir jāizņem un kaudzē jāsaliek elementi, kas izveidoti pēc aprakstītā algoritma.

Risinājuma izpildes laika sarežģītība ir $O(NlogN)$.

Pārvākšanās

Mēs varam ceļu koku "pakārt" aiz kādas no virsotnēm un iedomāties, ka visi koka ceļi "karājas" uz leju. Virsotni aiz kuras koks ir "pakārts", sauksim par koka sakni, līdzīgi arī apakškokos to augšējās virsotnes sauksim par apakškoka saknēm. Šo ceļu koku apstrādāsim sākot no lapām, tās (vismaz domās) izdzēsīsim no koka un tad jaunas koka virsotnes kļūs par lapām, apstrādāsim tās u.t.t. kamēr būsim apstrādājuši visu koku.

Katrai virsotnei aprēķināsim divas vērtības:

- X ir mazākais kopējais ceļu kopgarums, ko rūķi ir veikuši šajā apakškokā, ja visi rūķi ir pārvākušies, vērtību un
- Y ir mazākais kopējais ceļu kopgarums, ko rūķi ir veikuši šajā apakškokā, ja visi rūķi, izņemot rūķi, kas dzīvo šī apakškoka saknē, ir pārvākušies.

Apstrādājot kārtējo virsotni, nepieciešams šīs divas vērtības aprēķināt. Virsotnes X vērtību varam aprēķināt šādi: rūķim no saknes virsotnes kaut kur ir jāpārvācas (tātad jāpārvācas uz kādu no apakškokiem kas atrodas zem saknes virsotnes). Tam apakškokam, uz kuru viņš pārvāksies, ņemam mazāko no šī apakškoka X un Y vērtībām, bet pārējos apakškokos ņemam X vērtību, jo tajos nekāda kustība nenotiek, un jāņem mazākā vērtība, kas mums jau ir aprēķināta. Aplūkojam visus variantus - uz kurām virsotnēm rūķis no saknes var pārvākties, un izvēlamies mazāko no visiem. Virsotnes Y vērtību varam aprēķināt līdzīgi, tikai visos apakškokos, kas atrodas zem šīs virsotnes, ņemam X vērtības un saskaitām, jo tajos nekādas rūķu kustības nenotiek. Pēdējai apstrādātajai virsotnei būsim aprēķinājuši X vērtību, kas arī ir meklētā vērtība - mazākais kopējais rūķu pārvietojums, lai visi rūķi nonāktu jaunā mājā.

Risinājuma izpildes laika sarežģītība ir $O(N)$.

Svara rāvējslēdzējs

Atmiņā glabāsim nevis katras automašīnas svaru, bet pirmo $i(0 \leq i \leq cikla\_garums)$ automašīnu kopsvaru kop_i katram no virzieniem (attiecīgi $cikla\_garums = K$ kreisajai ielai vai $cikla\_garums = L$ - labajai).

Zinot, ka automašīnu svari katrā ielā cikliski atkārtojas, patvaļīgam indeksam $j(j > 0)$ varam aprēķināt pirmo j automašīnu kopsvaru kā $\left\lfloor \frac{j-1}{cikla\_garums} \right\rfloor * svars[cikla\_garums] + svars[j \bmod cikla\_garums + 1]$.

Tad, meklējot, kura automašīna un no kuras ielas iebrauca krustojumā kā j-tā, darīsim to ar segmenta dalīšanu uz pusēm. Mazākais automašīnu skaits, kas varēja iebraukt no kreisās puses, ir 0, bet lielākais - j (ja visas j automašīnas iebrauca no kreisās puses). Atradīsim to no kreisās puses iebraukušo automašīnu skaitu j_{kr}, kuram atbilstošais no labās puses iebraukušo automašīnu kopsvars $kop_{j_{la}} \leq kop_{j_{kr}}$ un $j_{la} + j_{kr}$ vērtība ir maksimāli tuvu j vērtībai (j vai $j-1$).

Ja $j = j_{kr} + j_{la}$, tad kā j-tā izbrauks j_{kr} automašīna no kreisās puses. Pretējā gadījumā kā j-tā automašīna izbrauks $j - j_{kr}$ automašīna no labās puses. Attiecīgās automašīnas svaru var iegūt atņemot kopsvaru virknē divu blakuselementu vērtības.

Aprakstītā atrisinājuma izpildes laika sarežģītība ir $O(VlogK + K + L)$.

Teniss

Viss algoritms jau ir aprakstīts uzdevuma tekstā un atliek to tikai realizēt.

Ir tikai divas lietas, kur jāuzmanās:

1) turnīra beigu apstrāde, ja Ernests vienu turnīru ir uzvarējis un ar uzvaru sāk nākamo turnīru (ievaddatos par to liecina secīgu 'u' fragments, kas garāks par N);

2) punkta par piedalīšanos turnīrā pieskaitīšana.

Programmas mainīgajos saglabāsim par uzvarām iegūto punktu skaitu (`punkti`), turnīru skaitu (`turnīru_skaits`) un kārtas numurs turnīrā (`kārta`). Sākotnēji visu šo lielumu vērtības ir 0.

Punktu skaitu par uzvaru noteiktā kārtā var aprēķināt vienreiz programmas sākumā un saglabāt masīvā `punkti_par_uzvaru`.

Uzdevuma saturs ļauj visu nepieciešamo aprēķināt uzreiz pēc informācijas par pārtējās spēles rezultātu ielasīšanas - nav nepieciešams saglabāt visu rezultātu virkni.

Katras spēles rezultāta apstrādi var veikt šādi:

```
ja Zaudējums
   turnīru_skaits ≔ turnīru_skaits + 1
   kārta ≔ 0
citādi    // Uzvara
   kārta ≔ kārta + 1
   punkti := punkti + punkti_par_uzvaru[kārta]
   ja (kārta = N)
         turnīru_skaits ≔ turnīru_skaits + 1
         kārta ≔ 0
```

Gan zaudējuma, gan N secīgi uzvarētu spēļu virkne nozīmē kārtējā turnīra beigas, tādēļ abas šīs situācijas tiek apstrādātas vienādi.

Dotajā algoritma fragmentā punktu kopsummai tiek pieskaitīti tikai par uzvarām iegūtie punkti, bet netiek pieskaitīti punkti par piedalīšanos turnīrā. Tā kā par katru turnīru tiek piešķirts viens punkts, tad visu šo punktu kopsumma sakrīt ar turnīru skaitu, un kā programmas rezultāts jāizvada summa `punkti+turnīru_skaits`.

Algoritma izpildes laika sarežģītība ir lineāri atkarīga no kopējā spēļu skaita jeb rezultātu virknes garuma.

Skaitļu tornis

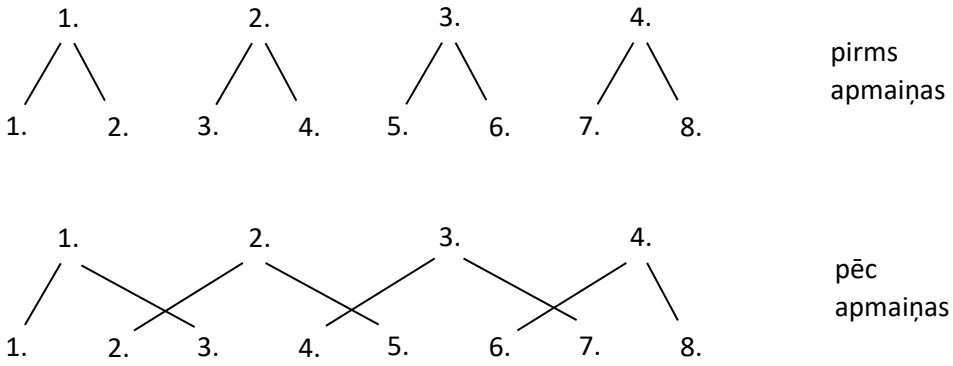
Vecāku numuri. Uzdevumā dotais tornis ir binārs koks. Katrai virsotnei ir divi bērni, un katrai virsotnei, kas nav koka sakne, ir viens vecāks. Aplūkosim, kā ir saistīti bērnu un vecāku indeksi.

Paņemsim r-to rindu, tajā kopā ir tieši 2^r virsotnes. Šajā rindā aplūkosim virsotni pozīcijā k un noteiksim, kāds numurs ir tās vecākam (rindā $r-1$). Šķirojam vairākus gadījumus:

- $k = 1$. Šīs virsotnes vecāka numurs ir 1.
- $k = 2^r$. Šīs virsotnes vecāka numurs ir 2^{r-1}, pēdējā virsotne iepriekšējā rindā.
  Šis un iepriekšējais gadījums ir spēkā, jo pirmo un pēdējo virsotni mēs nemainām vietām ar blakus virsotni.
- $k > 1$ un k ir nepāra. Ja virsotnes vispār netiktas mainītas vietām, tad, tā kā katram vecākam ir tieši pa diviem bērniem, tad pirms k-tās virsotnes ir $k-1$ bērni $\frac{k-1}{2}$ vecākiem. Tātad pašai k-tai virsotnei vecāks būtu ar numuru $\frac{k+1}{2}$. Bet tā kā virsotnes ar numuriem $k-1$ un k tika samainītas vietām, tad pozīcijā k tagad ir vecāka ar numuru $\frac{k-1}{2}$ bērns.

- $k < 2^r$ un k ir pāra. Līdzīgi kā iepriekšējā punktā, ja nebūtu mainīšanās vietām, tad vecāka numurs būtu $\frac{k}{2}$. Bet, tā kā virsotnes ar numuriem k un $k+1$ tiek mainītas vietām, tad pozīcijā k tagad būs vecāka ar numuru $\frac{k}{2}+1$ bērns.

Aplūkosim šo rindu $r = 3$ (41. zīm.)



41. zīm.

Redzam, ka virsotņu vecāki numuri ir:

- 1. virsotnes vecāks ir 1.
- 2. virsotnes vecāks ir $\frac{2}{2}+1=2$.
- 3. virsotnes vecāks ir $\frac{3-1}{2}=1$.
- 4. virsotnes vecāks ir $\frac{4}{2}+1=3$.
- 5. virsotnes vecāks ir $\frac{5-1}{2}=2$.
- 6. virsotnes vecāks ir $\frac{6}{2}+1=4$.
- 7. virsotnes vecāks ir $\frac{7-1}{2}=3$.
- 8. virsotnes vecāks ir $2^{3-1}=4$.

Algoritms. Sākumā aplūkosim neefektīvu algoritmu. Sāksim ar doto virsotni un virzīsimies uz augšu pa vecākiem, līdz nonāksim saknē. Aplūkosim divus secīgus līmeņus $r-1$ un r, un ievērosim:

- Skaitlis a tiek pieskaitīts uz tām šķautnēm, kas ved uz virsotnēm ar numuriem 1 un k, kur k ir pāra un $k < 2^r$.
- Skaitlis b tiek pieskaitīts uz tām šķautnēm, kas ved uz virsotnēm ar numuriem 2^r un k, kur k ir nepāra un $k > 1$.

Līdz ar to, zinot virsotnes rindu r un tās pozīciju tajā k, mēs varam pāriet uz tās vecāku, un arī pieskaitīt atbildei šķautnes skaitli (a vai b). Nonākot saknē, būsim saskaitījuši visus a un b, kas tiek pieskaitītas ceļā uz doto virsotni. Lai iegūtu atbildi, pieskaitām beigās n un izvadām.

Optimizācija. Diemžēl iepriekš aprakstītais algoritms ir pārāk lēns, jo r var būt pat līdz 10^9, un turklāt mums jāatbild uz v pieprasījumiem, kur v var būt līdz 10^5. Tas rezultētos algoritmā, kam jāizpilda 10^{14} operācijas, kas ir ļoti tālu no efektīvā.

Lai paātrinātu algoritmu, ņemsim vērā, ka:

Ja $k = 1$, tad vecāka numurs vienmēr ir 1, un uz šķautnes uz vecāku vienmēr tiek pieskaitīts a. Tas nozīmē, ka summa šajā virsotnē būs $r \cdot a$.

Ja $k = 2$, tad vecāka numurs vienmēr ir 2, izņemot pirmo rindu, kad vecāka numurs ir 1. Uz šķautnēm savukārt vienmēr būs skaitlis a, izņemot šķautni no saknes, kur skaitlis būs b. Līdz ar to summa šajā virsotnē būs $(r-1) \cdot a + b$.

Ja $k > 2$, tad vecāka numurs ir vai nu $\frac{k-1}{2}$ (ja k ir nepāra), vai nu $\frac{k}{2} + 1$ (ja k ir pāra un mazāks par 2^r), vai nu $\frac{k}{2}$ (ja $k = 2^r$). Tātad ar katru soli uz augšu virsotnes numurs samazinās gandrīz divreiz (un vienmēr ir stingri mazāks), un pēc $O(\log k)$ soļiem mēs nonāksim vecākā ar numuru 1 vai 2.

Visbeidzot, ievērosim, ka $k < 2^{63}$ pēc dotā. Līdz ar to kopā veiksim aptuveni $\log_2 k \leq 63$ operācijas uz katru pieprasījumu.

Ātrdarbība. Izpildes laika sarežģītība ir vienāda ar $O(v \log k)$.

2018./2019. mācību gads

Novada olimpiāde - 2019

Attālums kokā

Eiklīda algoritms. Šis uzdevums ir cieši saistīts ar plaši pazīstamo *Eiklīda algoritmu*. Šis algoritms atrod divu naturālu skaitļu lielāko kopīgo dalītāju (LKD). Vienkāršā algoritma versija ir šāda: ja dotie skaitļi ir x un y, tad vienā solī algoritms no lielāka skaitļa atskaita mazāko. Neizbēgami abi skaitļi kādā brīdī kļūst vienādi, kurā brīdī abu vērtība ir tieši vienāda ar $\mathrm{LKD}(x,y)$.

Šī algoritma ātrdarbība sliktākajā gadījumā var būt $O(\max(x,y))$. Tas var notikt, piemēram, ja $x=1$. Tad algoritms izpildīs $y-1$ soli, līdz abi skaitļi kļūst vienādi ar 1. Par laimi, Eiklīda algoritmam ir zināms paātrinājums: pieņemsim, ka $x<y$; tad tā vietā, lai y aizvietotu ar $y-x$, y aizvieto ar $y \pmod x$. Tādā rezultātā ar vienu operāciju tiek izpildīti visi soļi, kuros x nemaina savu vērtību. Var pierādīt, ka Eiklīda algoritms ar šādu paātrinājumu strādā laikā $O(\log(\max(x,y)))$.

Vispārīgie apsvērumi. Aplūkosim vienu skaitli kokā $\frac{x}{y}$, kas nav saknē, un tā vecāku. Skaitli, kas ir ierakstīts vecāka virsotnē, iegūst sekojošā veidā:

- ja $x>y$, tad vecāka skaitlis ir $\frac{x-y}{y}$;
- ja $x<y$, tad vecāka skaitlis ir $\frac{x}{y-x}$.

Ievērojam, ka naturāli skaitļi, kurus iegūstam vecāka skaitītājā un saucējā, ir tieši tie skaitļi, kurus mēs būtu ieguvuši, ja izpildītu vienu *Eiklīda algoritma* vienkāršās versijas soli uz skaitļiem x un y. Šis algoritms ir plaši pazīstama procedūra, kas diviem skaitļiem atrod to abu lielāko kopīgo dalītāju.

Līdz ar to uzdevums reducējas uz sekojošo uzdevumu: vajag atrast tādus naturālus skaitļus U un V, ka skaitlis $\frac{U}{V}$ ir zemākais priekštecis abiem skaitļiem $\frac{P}{Q}$ un $\frac{R}{S}$ Kalkina-Vilfa kokā; tad atbilde uzdevumam ir attālumu no $\frac{P}{Q}$ līdz $\frac{U}{V}$ un no $\frac{R}{S}$ līdz $\frac{U}{V}$ summa.

Atrisinājums. Pašā sākumā izdalīsim P un Q ar $\mathrm{LKD}(P,Q)$, un R un S ar $\mathrm{LKD}(R,S)$. Tādā veidā strādāsim ar nesaīsināmām daļām mūsu risinājumā.

Tad nesaīsināmai pozitīvai daļai $\frac{A}{B}$ definējam *īpašu priekšteču virkni*. Par $\frac{A}{B}$ *īpašu priekšteci* sauksim tādu priekšteci $\frac{A'}{B'}$, ka:

- ja $A=1$ vai $B=1$, tad $A'=B'=1$;
- citādi, ja $A>B$, tad $A'=A \pmod B$ un $B'=B$;
- citādi, ja $A<B$, tad $A'=A$ un $B'=B \pmod A$.

Par $\frac{A}{B}$ īpašu priekšteču virkni sauksim skaitļu virkni, ko iegūst, atkārtoti ņemot īpašos priekštečus (ieskaitot sākotnējo skaitli). Turklāt katram priekštecim arī izrēķinām, cik Eiklīda vienkāršā algoritma soļus vajag veikt, lai iegūtu šo priekšteci. Īpašo priekšteču virkne simulē Eiklīda paātrināta algoritma darbību (izņemot pēdējo soli, kad pieprasām, lai pēdējais pāris būtu $(1,1)$, nevis $(0,\mathrm{LKD}(A,B))$ vai $(\mathrm{LKD}(A,B),0)$, kā notiktu Eiklīda algoritma beigās – tas ir vajadzīgs, jo Kalkina-Vilfa kokā koka sakne ir $\frac{1}{1}$).

Piemēram, skaitlim $\frac{7}{31}$ īpašo priekšteču virkne (un soļu skaiti) būtu

$$\frac{8}{31} \xrightarrow{4} \frac{8}{3} \xrightarrow{2} \frac{2}{3} \xrightarrow{1} \frac{2}{1} \xrightarrow{1} \frac{1}{1}$$

Ievērosim, ka pēdējais skaitlis šajā virknē vienmēr būs $\frac{1}{1}$. Ievērosim arī, ka Eiklīda algoritma darbības laikā abu skaitļu lielākais kopīgais dalītājs nemainās; un tā kā $\mathrm{LKD}(A,B)=1$, tad katrs īpašais priekštecis arī būs nesaīsināmā daļa.

Tālāk pierādīsim sekojošo apgalvojumu: ja $\frac{P}{Q}$ un $\frac{R}{S}$ ir divas nesaīsināmās daļas, tad to zemākais kopējais priekštecis $\frac{U}{V}$ Kalkina-Wulfa kokā ir īpašais priekštecis kādai no šīm divām daļām.

Pierādījums: pieņemsim no pretējā, ka $\frac{U}{V}$ nav īpašais priekštecis nevienai no abām daļām. Daļa $\frac{U}{V}$ nevar būt vienāda ar $\frac{1}{1}$, jo tas ir īpašs priekštecis abās virknēs. Tāpēc ir vismaz viens skaitlis, kas ir priekštecis pašai daļai $\frac{U}{V}$, nosauksim to par $\frac{X}{Y}$. Paņemsim augstākos īpašos $\frac{P}{Q}$ un $\frac{R}{S}$ priekštečus tādus, kuriem $\frac{X}{Y}$ vēl ir priekštecis. Pieņemsim, ka $X < Y$ (gadījums $X > Y$ ir analoģisks). Tad pēc īpašo priekšteču definīcijas, jāizpildās $Y = Q$ un $X = P \pmod Y$, kā arī $Y = S$ un $X = R \pmod Y$. Bet tad vai nu $\frac{P}{Q}$ ir priekštecis $\frac{R}{S}$, vai nu $\frac{R}{S}$ ir priekštecis $\frac{P}{Q}$. Tā ir pretruna, jo sanāk, ka vai nu $\frac{U}{V} = \frac{P}{Q}$, vai nu $\frac{U}{V} = \frac{R}{S}$, un tā kā abi skaitļi $\frac{P}{Q}$ un $\frac{R}{S}$ ir arī īpaši priekšteči savās virknēs, tad $\frac{U}{V}$ ir vienāds ar kādu no īpašiem priekštečiem.

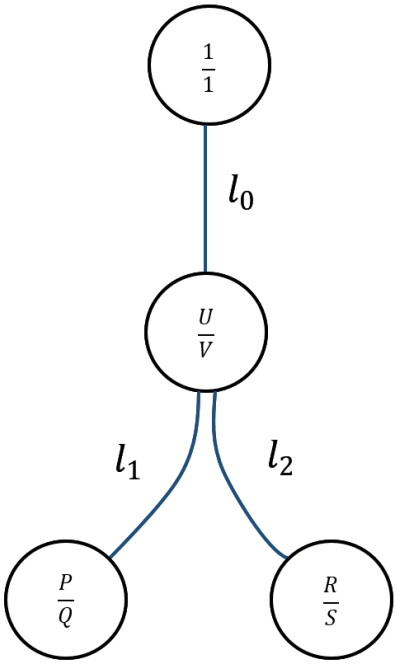
No šī apgalvojuma seko arī algoritms uzdevumam.

1. Atrodam abiem skaitļiem $\frac{P}{Q}$ un $\frac{R}{S}$ īpašo priekšteču virknes.
2. Katram no viena skaitļa īpašiem priekštečiem pārbaudām, vai tas var būt arī otrā skaitļa īpašais priekštecis. Kā to izdarīt? Ja tā ir tiesa, tad tam vai nu jābūt arī otrā virknē, vai nu tam jābūt kāda otrās virknes skaitļa priekštecim. Piemēram, kādiem skaitļiem $\frac{8}{3}$ vai būt priekštecis? Tie ir vai nu skaitļi formā $\frac{8}{3+8k}$, vai nu skaitļi formā $\frac{8+3k}{3}$ (kur k ir naturāls). Piemēram, ja $\frac{P}{Q} = \frac{8}{31}$ un $\frac{R}{S} = \frac{25}{3}$, tad šo skaitļu īpašo priekšteču virknes ir

$$\frac{8}{31} \xrightarrow{4} \frac{8}{3} \xrightarrow{2} \frac{2}{3} \xrightarrow{1} \frac{2}{1} \xrightarrow{1} \frac{1}{1}$$

$$\frac{17}{3} \xrightarrow{5} \frac{2}{3} \xrightarrow{1} \frac{2}{1} \xrightarrow{1} \frac{1}{1}$$

   Tad $\frac{8}{3}$ ir priekštecis $\frac{17}{3}$, jo $\frac{17}{3} = \frac{8+3\cdot3}{3}$. Šajā gadījumā $\frac{8}{3}$ ir arī zemākais kopējais priekštecis.
3. Izvēlamies no visiem īpašajiem priekštečiem, kuri ir priekšteči abiem skaitļiem $\frac{P}{Q}$ un $\frac{R}{S}$, zemāko – tas arī ir meklētais $\frac{U}{V}$. Attālumu no saknes var izrēķināt, izmantojot soļu skaitu virkni īpašo priekšteču virknēs.
4. Lai iegūtu attālumu starp $\frac{P}{Q}$ un $\frac{R}{S}$, saskaitām visu soļu skaitu abās virknēs, un atņemam divreiz attālumu no saknes līdz $\frac{U}{V}$. To ilustrē 42. zīmējums:



42. zīm.

Attālums starp $\frac{P}{Q}$ un $\frac{R}{S}$ ir vienāds ar

$$l_1 + l_2 = (l_1 + l_0) + (l_2 + l_0) - 2l_0$$

Risinājuma izpildes laika sarežģītība. Atkarībā no realizācijas izpildes laika sarežģītība ir vai nu $O(\log(\max(x,y)))$, vai arī $O(\log(\max(x,y)))^2)$. Abi varianti ir pietiekoši ātri.

Virknes fragments

Uzdevuma risinājumā izmantosim asociatīvu masīvu (piemēram, *map* valodā C++) A, kura elementu atslēgas ir veseli skaitļi un vērtības - nenegatīvi veseli skaitļi. Pievienosim A elementu $A[0] = 0$. Aprēķināsim pirmo $k(1 \leq k \leq N)$ virknes locekļu vērtību summu s_k un katrai no tām pārbaudīsim, vai A neatrodas elements $A[s_k - S]$.

Ja šāds elements atrodas, tad esam atraduši derīgu fragmentu, kas sākas ar virknes locekli, kura indekss ir $A[s_k - S] + 1$, bet beidzas ar k-to virknes locekli. Tātad attiecīgā fragmenta garums ir $k - A[s_k - S]$. Ja šis fragments ir garāks par iepriekš atrastajiem, tad jāsaglabā informācija par to.

Ja A nesatur elementu $A[s_k]$, tad pievienojam to: $A[s_k] = k$.

Elementa esamības asociatīvajā masīvā pārbaude un jauna elementa pievienošana notiek logaritmiskā laikā, tādēļ kopējā risinājuma izpildes laika sarežģītība ir $O(NlogN)$.

Kastu tornis

Lai gan uzdevuma noteikumos ir minēti trīs atšķirīgi kastes novietošanas veidi (uzsēšanās, uzkāršanās un nosegšana), varam būtiski atšķirīgo veidu skaitu samazināt uz diviem - vai kastes pamats sasniedz kādu no iepriekšējām kastēm (uzsēšanās vai nosegšana) vai - nē (uzkāršanās).

Ja aplūkosim tās kastes uz kurām ir iespējams uzlikt vēl kādu kasti, tad redzēsim, ka to malu garumi vienmēr ir sakārtoti augošā secībā no lejas uz augšu (varam pieņemt, ka sākotnējais pamats ir milzīgas kastes augša ar malas garumu, kas pārsniedz lielākās kastes malas garumu). Tās kastes, kas ir nosegtas ar lielāku kasti, vai arī kuras vairs nav pieejamas lielākas kastes uzkāršanās dēļ, nekādi vairs nevar ietekmēt nākamo kastu novietošanu.

Saglabāsim informāciju par kastēm, uz kurām vēl iespējams uzlikt vēl kādu kasti, saglabājot kastes malas garumu un tās augšējās skaldnes augstumu, skaitot no sākotnējā pamata. Pēc katras kastes uzlikšanas torņa augstums vai nu pieaug, vai arī saglabājas iepriekšējais - tas nevar samazināties kastes uzlikšanas rezultātā.

Ciklā ielasām kārtējās kastes malas garumu $kastes\_mala$ un meklējam augstāko kasti, kura derētu pēc izmēra, lai uz tās šo kasti uzliktu. Meklējot šo kasti varam dzēst informāciju par kastēm, kas ir mazākas par $kastes\_mala$, jo, neatkarīgi no novietošanas veida, šīs kastes vairs nebūs pieejamas kā pamats kādai no nākamajām kastēm. Kad pirmā derīgā kaste atrasta, aprēķinām teorētisko torņa augstumu, ja šī kaste būtu novietota: $augstums_{teorētiskais} = augstums_{derīgs\ kā\ pamats} + kastes\_mala$. Ja šī teorētiskā torņa augstums ir lielāks vai vienāds ar torņa augstumu pirms kastes novietošanas, tad ir notikusi uzsēšanās vai nosegšana un teorētiskais augstums ir reālais torņa augstums. Papildinām ierakstu par šobrīd augstāko kasti (ar malas garumu $kastes\_mala$) ar informāciju par torņa augstumu $augstums_{teorētiskais}$.

Ja teorētiskā torņa augstums ir mazāks par torņa augstumu pirms kastes novietošanas, tas nozīmē, ka ir notikusi uzkāršanās un ieraksts par šobrīd augstāko kasti (ar malas garumu $kastes\_mala$) ir jāpapildina ar informāciju par to torņa augstumu, kāds tas bija pirms kastes novietošanas.

Pēc visu kastu apstrādes atliek izvadīt aprēķināto augstākās kastes līmeni, kas sakrīt ar torņa kopējo augstumu.

```
raksts: {mala: naturāls_skaitlis; augstums: naturāls_skaitlis}
tornis: raksts masīvs
tornis[0] ≔ {∞, 0}
skaits ≔ 0
visa_torņa_augstums ≔ 0
cikls
    ielasa kastes_mala
    cikls
          ja tornis[skaits].mala > kastes_mala
                pārtraukt
          citādi
                skaits ≔ skaits - 1
    teorētiskais_augstums ≔ tornis[skaits].augstums + kastes_mala
    ja teorētiskais_augstums > visa_torņa_augstums
          visa_torņa_augstums ≔ teorētiskais_augstums
    skaits ≔ skaits + 1
    tornis[skaits] ≔ {kastes_mala, visa_torņa_augstums}
izvada visa_torņa_augstums
```

Algoritma pierakstā izmantotā masīva vietā šajā uzdevumā dabīgi ir lietot datu struktūru steks.

Algoritma izpildes laika sarežģītība ir $O(N)$ - lineāra atkarībā no kastu skaita.

Mazākais taisnstūris

Saglabāsim informāciju par divām rūtiņām, kurām koordinātas x vērtības starp visām rūtiņām ir lielākās - t.i., rūtiņu ar vislielāko un otru lielāko x vērtību. Iespējams, ka šīs vērtības sakrīt - t.i., ir divas vai vairāk rūtiņas ar lielāko x vērtību. Šādā gadījumā saglabā informāciju par jebkurām divām rūtiņām ar šo īpašību.

Līdzīgi saglabā informāciju par divām "malējām" rūtiņām ar mazāko x vērtību, ar lielāko y vērtību un ar mazāko y vērtību.

Kad visos virzienos saglabāts pa divām "malējām" rūtiņām, varam aprēķināt, cik liels būtu atlikušās rūtiņas ierobežojošais taisnstūris, ja pašu malējo rūtiņu (tikai vienu!) mēģinātu atstāt ārpusē.

Aplūkosim gadījumu, ka gribam nogriezt rūtiņu ar maksimālo x vērtību. Varam iedomāties, ka griezienu veicam ar vertikālas taisnes palīdzību, ārpusē atstājot tieši vienu rūtiņu.

Ir iespējami vairāki gadījumi:

1) ja ar lielāko x vērtību ir vairākas rūtiņas, tad ar šāda grieziena palīdzību nevar nogriezt tikai vienu rūtiņu. Ierobežojošā taisnstūra laukums nemainās.

2) ir viena rūtiņa ar lielāko x vērtību un

a) šī rūtiņa ir malējā y virzienā (ar lielāko vai mazāko y vērtību). Ja šī ir vienīgā rūtiņa y virzienā ar lielāko (mazāko) vērtību, tad, nogriežot šo rūtiņu, tiks nogriezta arī malējā rūtiņa y virzienā un attiecīgi samazināsies ierobežojošā taisnstūra izmēri abos virzienos - gan x, gan y. Taisnstūra robežas attiecīgajos virzienos noteiks rūtiņas ar otru lielāko (mazāko) koordinātas vērtību.

b) šī rūtiņa nav malējā y virzienā - taisnstūra izmērs mainās tikai x virzienā un lielākā x vērtība ir otrai malējai rūtiņai.

Pēc tam analoģiski aprēķina ierobežojošā taisnstūra izmērus, ja tiek nogriezta viena rūtiņa katrā no atlikušajiem trim virzieniem.

Risinājuma izpildes laika sarežģītība ir $O(N)$.

Sēņošanas čempionāts

Par *s-nogabalu* sauksim nogabalu, kurā ir sēnes.

Sākumā "saspiedīsim" mežu masīvu - pārnumurēsim visas rindas un kolonnas tā, lai nebūtu neviena pilnīgi tukša rinda un kolonna (šo soli var arī nedarīt, bet, to izdarot, pēc tam ir vieglāk domāt un risināt uzdevumu). To darām sakārtojot visus s-nogabalus augošā secībā pēc kolonnas numura un piešķirot tām kolonnu numurus pēc kārtas (protams tiem s-nogabaliem, kam pirms tam bija vienādi numuri, arī pēc piešķiršanas ir jābūt vienādiem numuriem).

Lietosim datu struktūru, ko sauc par segmentu koku. Katra no koka lapām atbilst vienai kolonnai, un tajā glabāsim lielāko skaitu sēņu, kādu var salasīt, ja pēdējais apmeklētais s-nogabals atrodas šajā kolonnā. Segmentu koka virsotnēs glabāsim lielāko vērtību, kāda ir sastopama šīs virsotnes apakškokā.

Sakārtojam visus s-nogabalus pēc rindām augošā secībā un, ja rindas ir vienādas, tad pēc kolonnām augošā secībā. Tagad apstrādāsim nogabalus pa vienam pēc kārtas. Apstrādājot kārtējo s-nogabalu $[x, y]$, mums nepieciešams noskaidrot, cik ir lielākais iespējamais salasāmo sēņu skaits kolonnās $[1, x]$. To noskaidro ar segmentu koka palīdzību, jo no visiem šiem un tikai šiem s-nogabaliem mēs varam nokļūt s-nogabalā $[x, y]$. Pie iegūtās atbildes pieskaitām sēņu skaitu, kas ir s-nogabalā $[x, y]$ un iegūto vērtību ierakstām nogriežņu koka lapā kas atbilst x kolonnai un atjaunojam nogriežņu koka virsotņu vērtības.

Beigās noskaidrojam kāda ir lielākā ierakstītā vērtība nogriežņu koka lapās un tik sēnes arī būs iespējams savākt, ievērojot uzdevuma nosacījumus.

Risinājuma izpildes laika sarežģītība ir $O(NlogN)$.

Valsts olimpiāde - 2019

Akmens, šķēres, papīrīt's

Vispirms aplūkosim vienkāršāko gadījumu, ka izspēļu pierakstos nav ciparu - tas nozīmē, ka ne Dace, ne Kārlis nav attēlojuši vienu priekšmetu vairākas reizes pēc kārtas.

Šādā gadījumā atliek ciklā analizēt katru izspēli (salīdzināt attiecīgos simbolus abās virknēs) un noteikt, kurš no bērniem uzvarējis vai izspēle beigusies neizšķirti. Šos rezultātus summējot, iegūst gala rezultātus.

Tagad aplūkosim vispārīgo gadījumu, ka izspēļu pierakstos var būt arī cipari.

Aplūkosim uzdevuma tekstā doto piemēru, kur pierakstītās izspēļu virknes ir **`3A3SPAS2AP`** un **`PA5P2A3S`**. Tas atbilst šādām izspēlēm:

Dace	A	A	A	S	S	S	P	A	S	A	A	P
Kārlis	P	A	P	P	P	P	P	A	A	S	S	S

Šo izspēļu virkni var sadalīt secīgos fragmentos, kur katrā fragmentā ir viens vai vairāki vienādi izspēļu pāri:

Dace	A	A	A	S	S	S	P	A	S	A	A	P
Kārlis	P	A	P	P	P	P	P	A	A	S	S	S

Šajā gadījumā ir deviņi fragmenti, kuru garumi ir attiecīgi 1, 1, 1, 3, 1, 1, 1, 2 un 1.

Šo fragmentu rezultāti (uzvarētājs vai neizšķirts) ir: Kārlis, neizšķirts, Kārlis, Dace, neizšķirts, neizšķirts, Kārlis, Dace, Kārlis. Kopumā Dace ir uzvarējusi 3+2=5 izspēlēs, Kārlis - 1+1+1+1=4 izspēlēs, bet trīs izspēles ir beigušās neizšķirti. Diemžēl skaitliskie ierobežojumi ir tādi, ka izvērst virknes atsevišķu izspēļu virknē nav iespējams - tās būtu pārāk garas. Tāpēc jāatrod veids, kā apstrādāt izspēļu pierakstus, neizvēršot tos pilnībā.

Visas izspēles sadalīsim tādu rakstu virknē, kur katrs raksts satur vienādu secīgu burtu skaitu (naturāls skaitlis) un pašu burtu. Pēc tam nepieciešams salīdzināt šīs virknes. Iepriekš aplūkotajā rakstu piemērā šīs virknes ir:

Dace	3A	3S	1P	1A	1S	2A	1P
Kārlis	1P	1A	5P	2A	3S		

Salīdzināsim šīs virknes, ņemot rakstus pēc kārtas, izvēloties īsāko no tiem un, ja raksti ir atšķirīga garuma, garākā raksta garumu saīsinot par īsākā raksta garumu. Aplūkotajā piemērā rakstu virknes mainītos šādi:

Dace	2A	3S	1P	1A	1S	2A	1P
Kārlis	1A	5P	2A	3S			

Paņemts: 1A, 1P — Kārlis +1

Dace	1A	3S	1P	1A	1S	2A	1P
Kārlis	5P	2A	3S				

Paņemts: 1A, 1A — Neizšķ. +1

Dace	3S	1P	1A	1S	2A	1P
Kārlis	4P	2A	3S			

Paņemts: 1A, 1P — Kārlis +1

Dace	1P	1A	1S	2A	1P
Kārlis	1P	2A	3S		

Paņemts: 3S, 3P — Dace +3

Dace	1A	1S	2A	1P	
Kārlis	2A	3S			

Paņemts: 1P, 1P — Neizšķ. +1

Dace	1S	2A	1P	
Kārlis	1A	3S		

Paņemts: 1A, 1A — Neizšķ. +1

Dace	2A	1P	
Kārlis	3S		

Paņemts: 1S, 1A — Kārlis +1

Dace	1P	
Kārlis	1S	

Paņemts: 2A, 2S — Dace +2

Dace
Kārlis

Paņemts: 1P, 1S — Kārlis +1

Ieskatoties, kā mainās rezultāts, var pamanīt, ka rakstu virknes tiek apstrādātas tieši tāpat, kā iepriekš tika apstrādāti fragmenti, tikai fragmenti netiek izvērsti līdz atsevišķam simbolu pārim.

Tehniski interesants ir veids kā no dotās izspēļu apraksta virknes ***s*** sākot ar simbolu, kura indekss ir ***ind***, iespējams iegūt kārtējā raksta saturu.

To var realizēt ar šāda algoritma palīdzību:

```
raksts: {sk: naturāls_skaitlis; burts: simbols}
cikls
    c = s[ind]
    ind = ind + 1
    ja (c≥'0') un (c≤'9')
          raksts.sk := raksts.sk * 10 + (c-'0')
    citādi
          raksts.burts := c
          ja (raksts.sk = 0)
                raksts.sk ≔ 1
          pārtraukt
```

Nesalasāmie burti

Ja virknēs nav jautājuma zīmju, tad uzdevums sakrīt ar uzdevumu “Akmens, šķēres, papīrīt’s”.

Katrā brīdī kāds izspēļu pāris ir bijis pēdējais. Saglabāsim informāciju par to, kāds lielākais Daces uzvaru, Kārļa uzvaru un neizšķirtu skaits ir iegūstams, ja pēdējais pāris ir kāds no deviņiem izspēļu pāriem: `A-A`, `A-S`, `A-P`, `S-A`, `S-S`, `S-P`, `P-A`, `P-S`, `P-P`. Turklāt neinteresēsimies par virkņu turpmāko saturu - interesēsimies tikai par izpēļu nepretrunību no virknes sākuma līdz kārtējai vietai.

Aplūkosim vienkāršu piemēru, ka izspēļu virknes ir ar jautājuma zīmēm, bet bez ciparies: **`AS???`** un **`?ASP?`**.

Pirms virkņu apstrādes varam uzskatīt, ka iedomātā “iepriekšējā” izspēle varēja būt jebkura no iespējamām izspēļu kombinācijām un lielākais kāda uzvaru vai neizšķirtu skaits sākumā ir 0:

Dace	Kārlis	Lielākais Daces uzvaru skaits	Lielākais Kārļa uzvaru skaits	Lielākais neizšķirtu skaits
A	**A**	**0**	**0**	**0**
A	**S**	**0**	**0**	**0**
A	**P**	**0**	**0**	**0**
S	**A**	**0**	**0**	**0**
S	**S**	**0**	**0**	**0**
S	**P**	**0**	**0**	**0**
P	**A**	**0**	**0**	**0**
P	**S**	**0**	**0**	**0**
P	**P**	**0**	**0**	**0**

Aplūkojam pirmo izspēļu pāri no dotajām virknēm: `A-?`. Jautājuma zīmes vietā var būt jebkurš no trim simboliem, un, atkarībā no simbola, mainās rezultāts:

Dace	Kārlis	Lielākais Daces uzvaru skaits	Lielākais Kārļa uzvaru skaits	Lielākais neizšķirtu skaits
A	**A**	**0**	**0**	**1**
A	**S**	**1**	**0**	**0**
A	**P**	**0**	**1**	**0**
S	A			
S	S			
S	P			
P	A			
P	S			
P	P			

Pārējie seši salikumi nav iespējami un nākamajā solī nav jāanalizē.

Nākamais izspēļu pāris `S-A` jāsalāgo ar iepriekšējo situāciju. Ņemot vērā, ka Kārļa simbols nevar būt A, šis pāris var sekot tikai pāriem `A-S` un `A-P`. Tātad pēc šī pāra izspēles situācija ir:

Dace	Kārlis	Lielākais Daces uzvaru skaits	Lielākais Kārļa uzvaru skaits	Lielākais neizšķirtu skaits
A	A			
A	S			
A	P			
S	**A**	**1**	**2**	**0**
S	S			
S	P			
P	A			
P	S			
P	P			

Ievērosim, ka lielākais katra spēlētāja uzvaru vai neizšķirtu skaits ir dažādām virknēm. Bet uzdevumā netiek prasīts izvadīt konkrēto izspēļu pāru virkni, kas sasniedz maksimālo vērtību.

Nākamais izspēļu pāris `?-S` jāsalāgo ar `S-A`. Tātad ? vietā nevar būt S, bet var būt gan A, gan P:

Dace	Kārlis	Lielākais Daces uzvaru skaits	Lielākais Kārļa uzvaru skaits	Lielākais neizšķirtu skaits
A	A			
A	**S**	**2**	**2**	**0**
A	P			
S	A			
S	S			
S	P			
P	A			
P	**S**	**1**	**3**	**0**
P	P			

Nākamais izspēļu pāris `?-P` jāsalāgo ar `A-S` un `P-S`. Tagad ? vietā var būt jebkurš simbols, bet ne katrs salikums ir saderīgs ar katru iepriekšējo. Aprēķinot rezultātu, katrā iespējamajā variantā jāizvēlas maksimālā no vērtībām.

Dace	Kārlis	Lielākais Daces uzvaru skaits	Lielākais Kārļa uzvaru skaits	Lielākais neizšķirtu skaits
A	A			
A	S			
A	**P**	**1**	**4**	**0**
S	A			
S	S			
S	**P**	**3**	**3**	**0**
P	A			
P	S			
P	**P**	**2**	**2**	**1**

Pēdējais pāris `?-?` jāsalāgo ar šiem trim pāriem. Vienīgais ierobežojums, ka šajā pārī Kārļa simbols nevar būt P. Tātad ir iespējami seši salikumi:

Dace	Kārlis	Lielākais Daces uzvaru skaits	Lielākais Kārļa uzvaru skaits	Lielākais neizšķirtu skaits
A	**A**	**3**	**3**	**2**
A	**S**	**4**	**3**	**1**
A	P			
S	**A**	**2**	**5**	**1**
S	**S**	**2**	**4**	**2**
S	P			
P	**A**	**4**	**4**	**0**
P	**S**	**3**	**5**	**0**
P	P			

Kad visi simboli ir apstrādāti, nepieciešams katrā no tabulas uzvaru un neizšķirtu kolonnām paņemt lielāko skaitli un iegūstam, ka lielākais iespējamais Daces uzvaru skaits ir 4, Kārļa - 5, bet neizšķirtu - 2.

Ja virknēs ir cipari (skaitliski koeficienti), tad to apstrāde jāveic tā, kā aprakstīts uzdevuma "Akmens, šķēres, papīrīt's" risinājumā - sadalot izspēļu aprakstu virknes rakstos un tad apstrādājot vienāda garuma fragmentus.

Ingus koeficients / Uzdevumu grāmata

Šie uzdevumi olimpiādē tika doti vienā dienā ("Ingus koeficients" jaunākajai un "Uzdevumu grāmata" - vecākajai grupai) un saturā ir ļoti līdzīgi:

Eksistē uzdevumu grāmata ar N lapaspusēm. Grāmatas i-tajā lapaspusē ir i-tais uzdevums ar grūtību $\mathrm{a_i}$. Ingus risina uzdevumus no dotās grāmatas, viņa risināšanas prasmi var izteikt ar naturālu skaitli q. Viņš var atrisināt visus tādus uzdevumus, kuru grūtība ir mazāka vienāda par Ingus risināšanas prasmi q. Kad Ingus atrisina kādu uzdevumu, viņā risināšanas prasme q pieaug par 1.

Katrā dienas sākumā Ingus apskata pirmos M neatrisinātos uzdevumus no grāmatas un cenšas tos atrisināt:

- uzdevumā "Ingus koeficients": patvaļīgā secībā
- uzdevumā "Uzdevumu grāmata": uzdevumu numuru pieaugšanas secībā.

Ja Ingus apskata uzdevumu, ko viņš var atrisināt ar tā brīža risināšanas prasmi, viņš to noteikti atrisina un lapaspusi ar uzdevumu izplēš no grāmatas.

Uzdevums: Noteikt minimālo nepieciešamo risināšanas prasmi q_{min} ar kādu Ingus var atrisināt visus uzdevumus grāmatā un kāds mazākais dienu skaits tam būs nepieciešams ar risināšanas prasmi q_{min}.

Piemēri:

Grāmata: $N = 7, M = 3$, a: 3,1,7,2,4,3,9. Pieņem, ka $q = 1$

"Ingus koeficients":

Diena	Aplūkotie uzdevumi	Atrisinātie uzdevumi	Prasme dienas beigās	Atlikušie neatrisinātie uzdevumi grāmatā
0			1	3, 1, 7, 2, 4, 3, 9
1	3, **1**, 7	1	2	3, 7, 2, 4, 3, 9
2	**3**, 7, **2**	2, 3	4	7, 4, 3, 9
3	7, **4**, **3**	3, 4	6	7, 9
5	7, 9	---	---	---

“Uzdevumu grāmata”:

Diena	Aplūkotie uzdevumi	Atrisinātie uzdevumi	Prasme dienas beigās	Atlikušie neatrisinātie uzdevumi grāmatā
0			1	3, 1, 7, 2, 4, 3, 9
1	3, **1**, 7	1	2	3, 7, 2, 4, 3, 9
2	3, 7, **2**	2	3	3, 7, 4, 3, 9
3	**3**, 7, **4**	3, 4	5	7, 3, 9
4	7, **3**, 9	3	6	7, 9
5	7, 9	---	---	---

Savukārt, ja $q = 2$ visus uzdevumus ir iespējams atrisināt.

Lai atrisinātu uzdevumu, uzdevumu sadala divās daļās:

- Pirmā daļa: noteikt q_{min},
- Otrā daļa: atrast mazāko dienu skaitu ar atrasto q_{min}.

Risinājumu ir izdevīgi apskatīt sākot ar otro daļu.

Mazākais dienu skaits, lai atrisinātu visus uzdevumus pie fiksētas sākuma risināšanas prasmes q_s.

“Ingus koeficients”:

Dienā aplūkojamos neatrisinātos uzdevumus glabā kaudzē.

Algoritms:

1. Dienu uzsākot, papildina kaudzi līdz tā satur pirmos M neatrisinātos uzdevumus, vai kamēr grāmatā uzdevumi ir beigušies.
2. Kamēr kaudzē ir uzdevumi, kuru grūtība ir mazāka, vienāda par risināšanas prasmi. Tikmēr tos risina, izņemot tos no kaudzes.
3. Ja grāmatā ir vēl neatrisināti uzdevumi, tad uzsāk jaunu dienu un turpina algoritma izpildi sākot no 1. punkta.

Katrs uzdevums kaudzē tiks ievietots un no tās izņemts vienu reizi.

Izpildes laika sarežģītība: $O(N\ log(N))$.

Vingrinājums lasītājam: $O(N)$ sarežģītības realizācija.

“Uzdevumu grāmata”

Katru dienā aplūkojamo uzdevumu un tā numuru saglabā *tieši vienā* no trim kaudzēm:

Nosaukums	Paskaidrojums	Kārto pēc
Neatrisināmo uzdevumu kaudze	Uzdevumi, kurus Ingus pašreiz nevar atrisināt	Grūtības
Šodien atrisināmo uzdevumu kaudze	Uzdevumi, kurus šodien Ingus noteikti atrisinās	Numura
Rītdien atrisināmo uzdevumu kaudze	Uzdevumi, kurus Ingus rītdien noteikti atrisinās	Numura

Trīs kaudzes ir nepieciešamas, jo dienas ietvaros uzdevumi tiek pildīti uzdevuma numuru pieaugšanas secībā. Var gadīties, ka uzdevumus, ko “Ingus koeficienta” versijā varēja pildīt uzreiz pēc prasmes pieauguma, “Uzdevumu grāmatas” versijā jāatliek uz nākamo dienu.

Algoritms:

1. Dienu uzsākot, papildina kaudzes, kamēr tās kopā satur pirmos M grāmatas neatrisinātos uzdevumus vai arī grāmatā uzdevumi ir beigušies.

   Šie uzdevumi tiks ievietoti vai nu šodien noteikti atrisināmo vai neatrisināmo uzdevumu kaudzēs.

2. Kamēr ir noteikti šodien atrisināms uzdevums:
    1. Risina uzdevumu ar mazāko numuru.
    2. Palielinoties risināšanas prasmei, pārbauda, vai nerisināmo uzdevumu starpā nav uzdevumi, kurus tagad var atrisināt. Kamēr ir šādi uzdevumi:
        - Pārvieto šādu uzdevumu no neatrisināmo uz šodien vai rītdien atrisināmo uzdevumu kaudzēm atkarībā no tā, vai tas uzdevumu sarakstā atrodas attiecīgi pēc vai pirms tikko atrisinātā uzdevuma.
3. *Šodien neviens uzdevums vairs nav atrisināms*
4. Ja visi grāmatas uzdevumi atrisināti, tiek pabeigta algoritma izpilde.
5. Pārvieto rītdien atrisināmos uzdevumus uz šodien atrisināmiem. Pāriet uz jaunu dienu un atkārtoti izpilda algoritmu no 1. soļa.

Izpildes laika sarežģītība: $O(N\ log(N))$.

Mazākās nepieciešamās risināšanas prasmes q_{min} noteikšana

Izpildās šāda īpašība:

- Ja pie q_{good} vērtības Ingus spēs atrisināt visus grāmatas uzdevumus, tad viņš spēs atrisināt visus uzdevumus arī pie visiem q', kur $q' > q_{good}$.

Secinājums: Var veikt bināro meklēšanu pa q pēc tā vai Ingus var atrisināt visu grāmatu *(Atrisināmību pārbauda izmantojot simulāciju)*. Kopējā risinājuma izpildes laika sarežģītība: $O(log(2 \times 10^9)\ N\ log(N))$. Šāds risinājums var pārsniegt izpildes laika limitu. Optimālā risinājumā var izvairīties no binārās meklēšanas veikšanas.

Optimālā stratēģija:

Viegli ievērot, ka visus grāmatas uzdevumus ir iespējams atrisināt tad un tikai tad, ja visus grāmatas uzdevumus var atrisināt katrā dienā no pirmajiem M neatrisinātajiem uzdevumiem atrisinot *tieši vienu* uzdevumu.

Piemērs:

Grāmata: $N = 7, M = 3$, a: 3,1,7,2,4,3,9. Pieņemam, ka $q = 1$

Pēc jaunās stratēģijas:

Diena	Aplūkotie uzdevumi	Atrisinātie uzdevumi	Prasmes dienas beigāsq	Atlikušie neatrisinātie uzdevumi grāmatā
0			1	3, 1, 7, 2, 4, 3, 9
1	3, **1**, 7	1	2	3, 7, 2, 4, 3, 9
2	3, 7, **2**	2	3	3, 7, 4, 3, 9
3	**3**, 7, 4	3	4	7, 4, 3, 9
5	7, 4, **3**	3	5	7, 4, 9
6	7, **4**, 9	4	6	7, 9
7	7, 9	---	---	---

Pie $q = 1$ nevar atrisināt visus grāmatas uzdevumus.

Ja sasniedz dienu, kurā nevar atrisināt nevienu aplūkoto uzdevumu.

Diena	Aplūkotie uzdevumi	Atrisinātie uzdevumi	Prasmes dienas beigāsq	Atlikušie neatrisinātie uzdevumi grāmatā
6	7, **4**, 9	4	6	7, 9
*	7, 9	---	---	---

Viegli redzēt, ka arī ar dotajām "Ingus koeficients", "Uzdevumu grāmata" uzdevumu risināšanas stratēģijām nevarēs atrisināt visu grāmatu, jo neeksistēs uzdevumu virkne, ko atrisināt, lai sasniegtu nepieciešamo prasmes līmeni.

Lai turpinātu risināt grāmatu, pašreizējam prasmes līmenim ir jabūt vismaz tik lielam, cik vieglākajam uzdevumam no aplūkotajiem uzdevumiem. *Šajā gadījumā prasme* 6*, vieglākais uzdevums* 7. Palielinot Ingus sākuma prasmi q par $7 - 6 = 1$, Ingus pašreizējā prasme pieaugs par tādu pašu vērtību, šajā piemērā tā būs 7 un uzdevumu risināšana varēs turpināties.

Algoritms:

1. Pieņem, ka $q_{min} = 1$
2. Kamēr var izpildīt uzdevumu no pirmajiem M neatrisinātajiem uzdevumiem, tikmēr to izpilda.
3. Ja kādā dienā nevienu no aplūkojamajiem uzdevumiem nevar atrisināt, tad ir jāpalielina pašreizējais prasmes q un sākotnējais q_{min} prasmes līmenis:
   - No aplūkotajiem neatrisinātajiem uzdevumiem atrod uzdevumu ar minimālo sarežģītības līmeni z.
   - Ja šajā dienā Ingus prasmju līmenis ir q, tad uzstāda:
     - $q_{min} := q_{min} + z - q$
     - $q := z$.
   - Pāriet pie algoritma 2. punkta. *(Turpina risināt uzdevumus ar jauno prasmes līmeni)*

Pēc jaunās stratēģijas:

Diena	Aplūkotie uzdevumi	q_{min}	q	Atlikušie neatrisinātie uzdevumi grāmatā	Paskaidrojums
0			1	3, 1, 7, 2, 4, 3, 9	
1	3, **1**, 7	1	2	3, 7, 2, 4, 3, 9	
2	3, 7, **2**	1	3	3, 7, 4, 3, 9	
3	**3**, 7, 4	1	4	7, 4, 3, 9	
5	7, 4, **3**	1	5	7, 4, 9	
6	7, **4**, 9	1	6	7, 9	
7	7, 9	---	---	---	Nevar atrisināt nevienu uzdevumu. $z = 7$. Prasmju līmenis q ir par vienu vienību pa zemu. Palielina q par 1 (jeb $q = z$). $q_{min} := q_{min} + 1$
		2	7		
7	**7**, 9	2	8	9	
8	9	---	---	---	Nevar atrisināt nevienu uzdevumu. $z = 9$. Prasmju līmenis q ir par vienu vienību pa zemu. Palielina q par 1 (jeb $q = z$). $q_{min} := q_{min} + 1$
		3	9		
8	9	3	10		Visi uzdevumi atrisināti. $q_{min} = 3$

Šo algoritmu var realizēt ar kaudzi un rādītāju, pirmos M neatrisinātos uzdevumus tur kaudzē. Lai kaudzi papildinātu, tur rādītāju uz pirmo grāmatas uzdevumu, kas nav kaudzē. Katrā dienā aplūko no kaudzes visvieglāko uzdevumu. Izpildes laika sarežģītība ir $O(N\,log(N))$. Kad atrod q_{min} vērtību, tad dienu skaitu atrod izmantojot simulāciju.

Izpildes laika sarežģītība

1. q_{min} atrašana: $O(N\,log(N))$.
2. Nosimulēt optimālu grāmatas uzdevumu risināšanu: $O(N\,log(N))$.

Kopējā sarežģītība: $O(N\,log(N))$.

Meklēšana periodiskā virknē

Ņemam visus G_A simbolus un izveidojam pārus <simbols, vieta>, kas nozīmēs, ka attiecīgais simbols virknē G_A atrodas norādītajā vietā. Sakārtojam visus pārus pēc simbola, un, ja simboli ir vienādi, tad vietu pieaugšanas secībā. Šādā veidā mēs esam ieguvuši masīvu, kurā ar binārās meklēšanas palīdzības palīdzību jebkuram simbolam varam uzzināt tuvāko vietu, kur tāds atrodas.

Tagad iesim cauri meklētajai simbolu virknei G_B un katram nākamajam simbolam noskaidrosim, kura ir tuvākā vieta virknē G_A, kurā šādu simbolu var atrast. Ja līdz virknes G_A beigām šāds simbols vairs nav atrodams, tad tas nozīmē, ka nepieciešama vēl viena G_A kopija (atbilde jāpalielina par 1) un meklējam pirmo vietu, kur atrodas meklējamais simbols. Ja doto simbolu virknē atrast neizdodas, tad tas nozīmē, ka atbilde ir 0, jo virknē G_B ir simbols kāda nav virknē G_A.

Risinājuma izpildes laika sarežģītība ir $O(G_A log G_A + G_B log G_B)$.

Divi mīnusi

Sākumā varam izņemt no dotā masīva visas 0, jo pirms tām nevar likt mīnusu. Tad apzīmējam ar neg un pos attiecīgi negatīvo un pozitīvo skaitļu skaitu masīvā.

Minimāla summa. Galvenais novērojums: nekad nav izdevīgi ielikt papildus mīnusu negatīvam, nevis pozitīvajam skaitlim. Pieņemsim, ka optimālajā atbildē mums ir divi skaitļi $--a$ un b (kur $a, b \geq 1$). Šo skaitļu summa ir vienāda ar $a - 1 + b \geq 1$. Savukārt, ja pārliktu pirmo mīnusu pie b, tad būtu skaitļi $-a$ un $-b$, kuru summa būtu $-a - b \leq -2$. Līdz ar to nav izdevīgi atstāt pozitīvus skaitļus bez mīnusiem.

Vēl ievērosim, ka divus mīnusus ir izdevīgāk piešķirt dažādiem pozitīviem skaitļiem, nevis vienam. Pirmajā gadījumā mums ir skaitļi $-a$ un $-b$, kuru summa ir $-a - b \leq -2$, bet otrajā gadījumā mums ir skaitļi $--a$ un b, kuru summa ir $(a - 1) + b \geq 1$.

No šī fakta seko arī algoritms. Šķirojam divus gadījumus:

- $pos > K$. Tad mēs varam pielikt pa vienam mīnusam tieši K no pozitīvajiem skaitļiem. Ieliekot vienu mīnusu skaitlim a, mēs samazinām summu par $a - (-a) = 2a$. Līdz ar to ir izdevīgāk piešķirt mīnusus lielākiem pozitīviem skaitļiem. Tāpēc izvēlāmies k lielākos no tiem un pierakstām klāt katram pa mīnusam.

Piemēram, ja ir dots masīvs $[-10, -4, -3, 2, 7, 9]$ un $K = 2$, tad mēs to pamainām uz $[-10, -4, -3, 2, -7, -9]$.

- $pos \leq K$. Tad pirms katra pozitīvā skaitļa jāpieliek pa mīnusam. Tad visi N skaitļi kļūst negatīvi (katrs ar vienu mīnusu). Atliek noskaidrot kā sadalīt atlikušos $K - pos$ mīnusus. Ja pieliekam papildus mīnusu negatīvam skaitlim $-a$, tad jaunajam skaitlim būs vērtība $--a = a - 1$. Tāpēc kopējā summa palielinās par $a - 1 - (-a) = 2a - 1$. Līdz ar to ir jēga pierakstīt atlikušos mīnusus pie vislielākajiem skaitļiem (ar vismazāko a). Sakārtojam visus skaitļus un pierakstām atlikušos mīnusus pie vislielākajiem $K - pos$ no tiem.

Piemēram, ja ir dots masīvs $[-10, -4, -3, 2, 7, 9]$ un $K = 6$, tad pamainām to uz $[-10, --4, --3, --2, -7, -9]$.

Maksimālā summa. Šķirosim divus gadījumus:

- $neg > K$. Ņemot vērā iepriekšējās sadaļas novērojumu, secinām, ka izdevīgāk ir pielikt mīnusus negatīviem, nevis pozitīviem skaitļiem. Pieliekot mīnusu negatīvam skaitlim $-a$, iegūstam $--a$ un kopējā summa palielinās par $(a-1)-(-a)=2a-1$. Līdz ar to izvēlamies K mazākos negatīvos skaitļus un piešķiram tiem pa mīnusam.
  Piemēram, ja ir dots masīvs $[-10,-4,-3,2,7,9]$ un $K=2$, tad pamainām to uz $[--10,--4,-3,2,7,9]$.
- $neg < K$. Atkal šķirojam divus gadījumus. Apzīmējam $m := neg + K$.
  - m ir pāra skaitlis. Aplūkosim, kādu lielāko summu mēs potenciāli varam sasniegt. Skaidrs, ka visizdevīgāk būtu tad, ja $\frac{m}{2}$ skaitļiem priekšā katram būtu pa diviem mīnusiem, un pārējie skaitļi būtu pozitīvi. Tad kopējā summa būtu vienāda ar $|A_1|+|A_2|+\cdots+|A_N|-\frac{m}{2}$, jo katrs mīnusu pāris atņem pa 1 no kopējās summas. Savukārt tādu summu mēs arī varam sasniegt: piešķiram neg mīnusus pirms visiem negatīvajiem skaitļiem (to varam izdarīt, jo $neg < K$), un atlikušos $K-neg$ mīnusus piešķiram patvaļīgiem $\frac{K-neg}{2}$ pozitīvajiem skaitļiem.
    Piemēram, ja ir dots masīvs $[-10,-4,-3,2,7,9]$ un $K=5$, tad pamainām to uz $[--10,--4,---3,2,---7,9]$.
  - m ir nepāra skaitlis. Tad labākajā gadījumā optimālā atbildē mums būs $\frac{m-1}{2}$ skaitļi ar diviem mīnusiem pirms katra, un vienam skaitlim ar mazāko absolūto vērtību $|A_i|$ viens mīnuss priekšā. Tātad lielākā iespējamā summa būtu $|A_1|+|A_2|+\cdots+|A_N|-\frac{m-1}{2}-2|A_i|$,
    To arī varam sasniegt; aplūkojam skaitli ar vismazāko absolūto vērtību $|A_i|$.
    Ja $A_i > 0$, tad piešķiram neg mīnusus visiem negatīviem skaitļiem, vienu mīnusu skaitlim A_i, un $\frac{k-neg-1}{2}$ patvaļīgiem pozitīviem skaitļiem piešķiram pa diviem mīnusiem.
    Piemēram, masīvam $[-10,-4,-3,2,7,9]$ un $K=6$, tad izmainām uz $[--10,--4,---3,-2,---7,9]$.
    Ja $A_i < 0$, tad atstājam šo skaitli nemainītu. Piešķiram $neg-1$ pārējiem negatīviem skaitļiem pa vienam mīnusam, un $\frac{k-neg+1}{2}$ patvaļīgiem pozitīviem skaitļiem piešķiram pa diviem mīnusiem.
    Piemēram, masīvam $[-10,-4,-1,2,7,9]$ un $K=6$, izmainām uz $[--10,--4,-1,2,---7,---9]$.

Ātrdarbība. Izpildes laika sarežģītība ir $O(N \log N)$ dēļ skaitļu masīva kārtošanas.

Pāra skaits ciparu

Risinājuma plāns:

1. Nodefinēt funkciju $f(k, par)$, kuras vērtība ir visu skaitļu, kuru garums ir k un ciparu paritātes par, skaits. Un nodefinēt algoritmu, kas aprēķina šo funkciju pietiekami ātri.

2. Izmantojot $f(k, par)$ atrast meklētā N-tā skaitļa garumu K, un pēc tam visus ciparus, sākot ar pirmo ciparu c_K, tad c_{K-1}, .., c_1.

3. Novērtēt risinājuma sarežģītību.

Jebkuru skaitli mēs varam raksturot ar tā ciparu skaitiem $[x_0, x_1, .., x_9]$. Piemēram, skaitlim 1270277 tie ir $[1,1,2,0,0,0,0,3,0,0]$, bet skaitlim 38331831 - $[0,2,0,4,0,0,0,0,2,0]$.

Līdzīgi, katru skaitli mēs varam raksturot ar tā ciparu skaita paritātēm $[p_0, p_1, .., p_9]$, kur katram ciparam i paritāte $p_i = 0$, ja skaits x_i ir pāra, un $p_i = 1$, ja nepāra. Tātad, iepriekš minēto skaitļu paritātes būtu: 1270277 - $[1,1,0,0,0,0,0,1,0,0]$, 38331831 - $[0,0,0,0,0,0,0,0,0,0]$.

Uzdevumā dotās virknes skaitļi ir tādi, ka visi cipari skaitlī ir pāra skaitu reižu. Tātad tiem visiem ciparu paritātes ir $[0,0,0,0,0,0,0,0,0,0]$. Arī skaidrs, ka jebkurš skaitlis ar visām paritātēm 0 ir šīs virknes loceklis.

Tātad mēs meklējam N-to skaitli, kura ciparu paritātes ir $[0,0,0,0,0,0,0,0,0,0]$.

1. Nodefinēsim funkciju $f(k, par)$, un atradīsim metodi, kā to aprēķināt skaitliski.

1.1. Definēsim funkciju $f(k, par)$ kā visu skaitļu, kuru garums ir tieši k un ciparu paritātes $par = [p_0, p_1, .., p_9]$, skaits. Turklāt ieskaitām arī skaitļus, kuru pieraksts sākas ar 0, proti, 011 tiek uzskatīts par trīsciparu skaitli, kura pieraksts sastāv no vienas 0 un diviem 1.

Daži piemēri:

• $f(1, [0,0,0,0,0,0,0,0,0,0]) = 0$ (nav tādu viencipara skaitļu, kuru pieraksta visi cipari būtu pāra skaita reižu)

• $f(1, [0,0,0,0,0,0,0,1,0,0]) = 1$ (vienīgais šāds skaitlis ir 7)

• $f(2, [0,0,0,0,0,0,0,0,0,0]) = 10$ (divciparu skaitļi ar abiem vienādiem cipariem ir 00, 11, 22, 33, 44, 55, 66, 77, 88, 99)

• $f(2, [0,1,0,0,0,1,0,0,0,0]) = 2$ (15 un 51)

• $f(3, [0,1,0,0,0,0,0,0,0,0]) = 28$ (001, 010, 100, 111, 221, 212, 122, ..)

1.2. Kā aprēķināt patvaļīgu $f(k, par)$? To var izdarīt ar dinamiskās programmēšanas metodi.

Pieņemsim, ka skaitļa pirmais cipars ir 2. Cik ir tādu skaitļu garumā k, kuru pirmais cipars ir 2 un paritātes ir $par = [p_0, p_1, .., p_9]$? Aplūkosim to skaitļa fragmentu, kas seko pēc pirmā cipara (kurš ir 2). Pirmkārt, skaidrs, ka tā garums ir $k - 1$. Otrkārt, šī fragmenta paritātes par' ir tādas pašas kā par, izņemot cipara 2 paritāti, proti, $p_2 = 0$ (pāra), tad $p_2' = 1$ (nepāra), vai otrādi (visu pārējo ciparu skaits paliek nemainīgs). Tas tāpēc, ka, lai cik reizes cipars 2 parādītos šajā fragmentā (ko raksturo paritāte p_2'), lai iegūtu sākumā doto skaitli, mēs fragmenta priekšā pievienojam vēl vienu ciparu 2, tātad - palielinām ciparu 2 skaitu par viens, jeb mainām cipara 2 paritāti. Apzīmēsim "paritātes maiņu" ar $flip(p_i)$.

Līdz ar to, ja mēs zinām, ka skaitļa garumā k ar paritātēm par pirmais cipars ir 2, mēs šādu skaitļu skaitu varam izteikt ar $f(k-1, par'), par' = [p_0, p_1, flip(p_2), .., p_9]$. Citiem vārdiem, lai aprēķinātu skaitu garumam k, esam izmantojuši funkcijas f vērtību pie $k - 1$. Protams, analoģiskā veidā skaitu varētu aprēķināt, ja pirmais fiksētais skaitļa cipars būtu cits, nevis 2.

Šo ideju var vispārināt, lai aprēķinātu $f(k, par)$ (pirmais cipars nav zināms). Izmantojot iepriekš atrasto metodi, varam aprēķināt $f(k, par)$, sasummējot skaitus, kad pirmais cipars ir 0, kad pirmais cipars ir 1, utt.

Proti:

Skaidrs, ka pirmais cipars noteikti ir kāds no cipariem $0, 1, \ldots, 9$. Izmantojot iepriekš atrasto metodi, varam izteikt:

• pirmais cipars ir 0, un tad seko $k - 1$ nezināmi cipari ar paritātēm $par_0' = [flip(p_0), p_1, .., p_9]$.

• pirmais cipars ir 1, un tad seko $k - 1$ nezināmi cipari ar paritātēm $par_1' = [p_0, flip(p_1), .., p_9]$.

• ...

• pirmais cipars ir 9, un tad seko $k - 1$ nezināmi cipari ar paritātēm $par_9' = [p_0, p_1, .., flip(p_9)]$.

Tātad, $f(k, par) = f(k-1, par_0') + f(k-1, par_1') + .. + f(k-1, par_9')$.

Ja mēs zinām funkcijas f vērtības garumam $k - 1$ un visām paritāšu kombinācijām, mēs varam aprēķināt funkcijas $f(k, par)$ vērtību jebkurām paritātēm par. Tātad, zinot visas vērtības pie $k - 1$, mēs

varam aprēķināt visas vērtības pie k, pēc tam $k+1$ utt. Tā arī ir dinamiskās programmēšanas būtība - reducēt uzdevumu uz vienkāršākiem (šajā gadījumā - samazināt skaitļa garumu), un izteikt rezultātu ar jau zināmam vērtībām.

1.3. Mums atliek atrast "bāzes" gadījumu, jeb sākuma vērtības, uz kurām balstīsies visu pārējo vērtību aprēķināšana pēc iepriekš minētās formulas. Šajā gadījumā par "bāzi" ņemsim $k = 1$, jeb viencipara skaitļus. Ir tieši 10 viencipara skaitļi (ieskaitot "0"):

- $0, par = [1,0,0,0,0,0,0,0,0,0]$
- $1, par = [0,1,0,0,0,0,0,0,0,0]$
- ...
- $9, par = [0,0,0,0,0,0,0,0,0,1]$

Citu iespēju nav. Līdz ar to:

- $f(1,[1,0,0,0,0,0,0,0,0,0]) = 1$
- $f(1,[0,1,0,0,0,0,0,0,0,0]) = 1$
- ...
- $f(1,[0,0,0,0,0,0,0,0,0,1]) = 1$
- un $f(1,par) = 0$ visām citām ciparu paritātēm, jo tādu (garumā 1) skaitļu nav.

Esam atraduši vērtības $f(1,par)$ visam iespējamām paritātēm par, līdz ar to varam aprēķināt $f(k,par)$ izmantojot iepriekš atrasto formulu.

2. Meklējamā skaitļa X (virknes N-tā locekļa) konstrukcija.

2.1. Definēsim vēl vienu funkciju: $f_2(k,par)$ - skaitļu, kuru garums ir k un ciparu paritātes par, un kas nesākas ar 0, skaits. Viegli pārliecināties, ka $f_2(k,[p_0,p_1,..,p_9]) = f(k,[flip(p_0),p_1,..,p_9])$. Tie ir visi skaitļi garumā k ar atbilstošajām paritātēm par, izņemot tos, kuru pieraksts sākas ar 0. Tos, kas sākas ar 0, apraksta $f(k-1,[flip(p_0),p_1,..,p_9])$, jeb, citiem vārdiem, cik ir tādu skaitļu garuma $k-1$, kuriem sākumā pievienojot 0 iegūsim par.

2.2. Vispirms atradīsim X garumu. Mazākais iespējamais garums ir 1. Tādu derīgu skaitļu (kas nesākas ar 0) ir $n_1 = f_2(1,[0,0,0,0,0,0,0,0,0,0])$.

- Ja $N > n_1$, tas nozīmē, ka N-tais loceklis noteikti būs garāks par 1 ciparu. Tas tāpēc, ka visu virknes locekļu, kuru garums ir 1, skaits ir mazāks par meklētā skaitļa X kārtas numuru.
- Citādi (ja $N \leq n_1$), tad mēs zinām, ka meklējamais X būs 1 ciparu garš.

Pieņemsim, ka $N > n_1$. Tad mes zinām, ka X ir vismaz divus ciparus garš. Mēs varam pārfrāzēt uzdevumu kā "derīgo vismaz divu ciparu garo skaitļu virknē meklējam $N - n_1$-to locekli". Citiem vārdiem, mēs zinām, ka viencipara skaitļi (kuru skaits ir n_1) mums neder, tāpēc varam apskatīt virkni bez viencipara skaitļiem, un meklēt $N - n_1$-to locekli. Esam no iespējām izmetuši visus skaitļus garumā $k = 1$, un ieguvuši jaunu $N \to N_{jaunais} = N - n_1$.

Tālāk apskatām divciparu skaitļu skaitu $n_2 = f_2(2,[0,0,0,0,0,0,0,0,0,0])$. Ja $N > n_2$, tad atkal secinām, ka divciparu skaitļi mums noteikti neder, un tālāk apskatam tikai trīs un vairāk ciparu skaitļus, un $N \to N_{jaunais} = N - n_2$.

Šādi turpinot, mēs noteikti kaut kādā brīdī atradīsim k, kur $N \leq n_k = f_2(k,[0,0,0,0,0,0,0,0,0,0])$. Esam atraduši meklējamā skaitļa garumu k, kā arī zinām kāds k-ciparu derīgo skaitļu virknē ir X kārtas numurs: $N_{jaunais} = N - n_1 - n_2 - .. - n_k - 1$.

2.3. Tālāk atradīsim X ciparus. Metode līdzīga kā meklējot garumu:

- Apskatīsim X potenciālos sākuma fragmentus (pirmais vai vairāki pirmie cipari).
- Fragmentus apskatīsim augošā secībā - sāksim ar pirmo ciparu, tad pirmajiem diviem, utt.; fiksēta garuma fragmentiem sāksim ar mazākiem cipariem un pāriesim uz lielākiem.
- Ja dotai konfigurācijai (garums + sākuma fragments) atbilstošo virknes locekļu skaits $n_{k_{frag}}$ ir mazāks par meklējamo N, atjaunojam $N \to N_{jaunais} = N - n_{k_{frag}}$, un apskatam nākamo fragmentu.

• Kad atrodam konfigurāciju ar $N \leq n_{k_{frag}}$, mēs zinām, ka meklētajam X atbilst šobrīd fiksētais sākuma fragments; turpinām meklēšanu, paplašinot fragmentu par vienu ciparu.

Šādi mēs kaut kāda brīdī būsim atraduši X atbilstošu fragmentu tieši garumā k - tātad visus ciparus, jeb būsim atraduši X.

Ilustrēsim ar piemēru. Pieņemsim, ka esam atraduši X garumu $k = 4$, un tagad meklējam N-to derīgo virknes locekli garumā 4.

• Nofiksējam sākuma fragmentu $1xxx$, jeb pirmais cipars ir 1. To skaits ir $n_{4_1} = f(3, [0,1,0,0,0,0,0,0,0,0])$. Pieņemsim, ka $N > n_{4_1}$. Tātad pārejam uz nākamo fragmentu, izmantojot $N \to N_{jaunais} = N - n_{4_1}$.

• Nofiksējam sākuma fragmentu $2xxx$. To skaits ir $n_{4_2} = f(3, [0,0,1,0,0,0,0,0,0,0])$. Pieņemsim, ka $N > n_{4_2}$. Tātad pārejam uz nākamo fragmentu, izmantojot $N \to N_{jaunais} = N - n_{4_2}$.

• Tā turpinām ar $3xxx4, 4xxx$, utt.

• Pieņemsim, ka pie fragmenta $6xxx$, skaits ir $n_{4_6} = f(3, [0,0,0,0,0,0,1,0,0,0])$, un $N \leq n_{4_6}$. Tātad esam atraduši X pirmo ciparu. Turpinām ar divciparu fragmentiem.

• Nofiksējam sākuma fragmentu $60xx$. To skaits it $n_{4_{60}} = f(2, [1,0,0,0,0,0,0,1,0,0,0])$. Pieņemsim, ka $N > n_{4_{60}}$. Tātad pārejam uz nākamo fragmentu, izmantojot $N \to N_{jaunais} = N - n_{4_{60}}$.

• Tā turpinām ar $61xx, 62xx$, utt.

• Pieņemsim, ka pie fragmenta $64xx$, skaits ir $n_{4_{64}} = f(2, [0,0,0,0,1,0,1,0,0,0])$, un $N \leq n_{4_{64}}$. Tātad esam atraduši X pirmos divus ciparus. Turpinām ar trīsciparu fragmentiem.

Šādi turpinām, līdz kamēr būsim atraduši visus k ciparus.

3. Sarežģītības novērtējums

3.1. Apskatīsim, cik sarežģīti ir atrast funkcijas $f(k, par)$ vērtības.

Pieņemsim, ka gala rezultātā X pierakstā ir K cipari. Tas nozīmē, ka mums vajadzēt aprēķināt $f(k, par)$ tikai $k \leq K$.

Paritāšu iespējamās vērtības ir 0 vai 1 katram ciparam (kopā 10), tātad dažādo par vērtību skaits ir $2^{10} = 1024$.

Tātad kopā mums vajadzēs aprēķināt funkcijas f vērtību ne vairāk kā $K \times 1024$ argumentu (k, par) pāriem. Kā jau iepriekš apskatījām, lai aprēķinātu $f(k, par)$, mums jāsaskaita 10 dažādas jau zināmas $f(k-1, par')$ vērtības. Tātad, lai aprēķinātu visas nepieciešamās vērtības, nepieciešamas $K \times 1024 \times 10$ operācijas. No uzdevuma nosacījumiem zināms, ka $K \leq 18$, tātad nepieciešamas ~ 180000 operācijas. Visas šīs vērtības var aprēķināt pirms skaitļa X konstrukcijas, un tad konstantā laikā izmantot jau atrastās vērtības.

3.2. Apskatīsim, kāda ir X konstrukcijas procesa sarežģītība.

• Atkal, pieņemsim, ka skaitļa garums ir K.

• Lai atrastu X garumu, mēs esam apskatījuši potenciālos garumus $1, 2, .., K-1, K$, jeb veikuši K darbības.

• Pirmā cipara meklēšanā mēs apskatam ciparus $1, 2, \ldots$ Sliktākajā gadījumā meklētais pirmais cipars ir 9, un tātad esam veikuši 9 pārbaudes.

• Līdzīgi otrā cipara meklēšanā - sliktākajā gadījumā apskatīsim visus ciparus līdz 9, tātad 10 pārbaudes (10, jo, sākot ar otro pozīciju, kā iespējamo kandidātu mēs apskatām arī ciparu 0).

Un tieši tāpat visu pārējo ciparu meklēšanā - slikākajā gadījumā mēs katrā pozīcijā veiksim 10 pārbaudes. Tātad kopā būsim veikuši ne vairāk kā $K + 9 + 10(K-1)$ pārbaudes, jeb ne vairāk kā $11K$ pārbaudes.

3.3. Ņemot vērā, ka visu funkcijas $f(k, par)$ vērtību aprēķināšana (3.1.) ir krietni sarežģītākā par meklējamā skaitla X konstrukciju (3.2.), kopējā risinājumam nepieciešamas ~ 180000 operācijas, jeb < 200000.

Vispārīgā gadījumā sarežģītību var novērtēt kā $O(KA\,2^A)$, kur K - maksimālais meklējamā skaitļa garums, un A - dažādo izmantojamo ciparu skaits.

Rūtiņu laukuma sagriešana

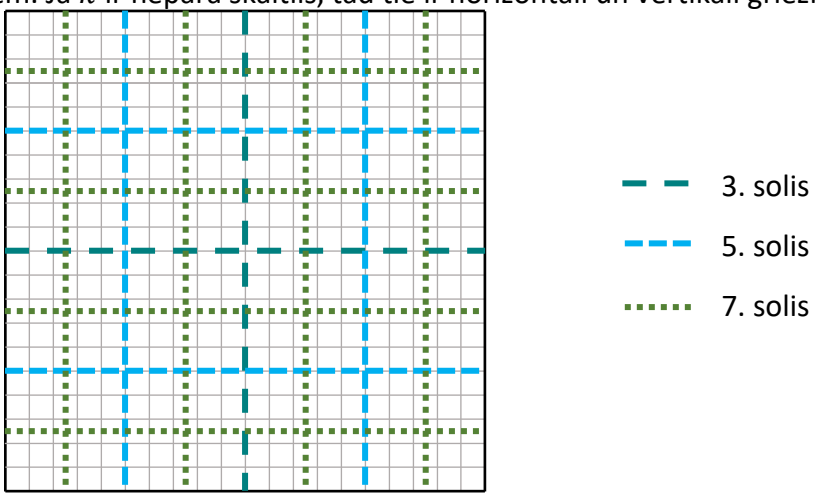
Sākumā ievērosim, ka ar katru griezienu palikušās figūras laukums samazinās divas reizes. Tā kā dotās rūtiņas laukums ir 1, tad to noteikti pārgriezīsim pēc $O(\log N)$ soļiem. Tāpēc uzdevums slēpjas tajā, lai veikli nosimulētu šo griešanu.

Ieviesīsim koordinātu sistēmu, kur punkts $(0,0)$ ir kvadrāta kreisais apakšējais stūris, X koordināta pieaug uz augšu, Y koordināta pieaug pa labi. Tad jebkuru rūtiņu aprakstām ar tās labējā apakšējā punkta koordinātām (tad arī dotajai rūtiņai paliek tādas pašas koordinātas, kā ievadā). Kvadrāta labējā augšējā punkta koordinātas ir (N, N).

Apzīmēsim ar x un y dotās rūtiņas koordinātas. Ievērosim, ka ar 1. griezienu varam pārgriezt tādas (un tikai tādas) rūtiņas, kas atrodas uz vienas no kvadrāta diagonālēm. Tās ir rūtiņas, kurām $x = y$ vai $x + y = N + 1$.

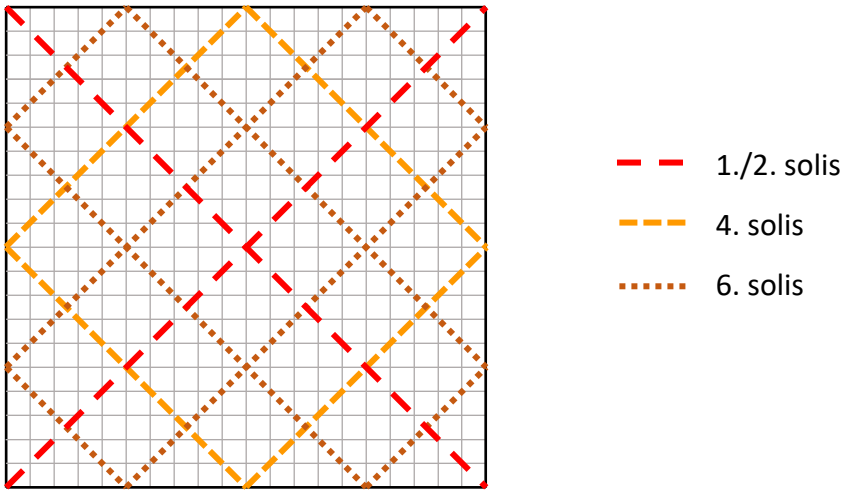
Kuras rūtiņas mēs varam sagriezt ar 2. griezienu? Ievērosim, ka 2. grieziens iet pa otru kvadrāta diagonāli. Bet tādas rūtiņas mēs jau varējām sagriezt 1. griezienā, izvēloties otru diagonāli. Līdz ar to ar 2. griezienu mēs nekad nepārgriezīsim doto rūtiņu.

Apzīmēsim ar k grieziena kārtas numuru. Tagad atsevišķi aplūkosim, kādi veidi ir nepāra un pāra kārtas numura griezieniem. Ja k ir nepāra skaitlis, tad tie ir horizontāli un vertikāli griezieni (43. zīm.):



43. zīm.

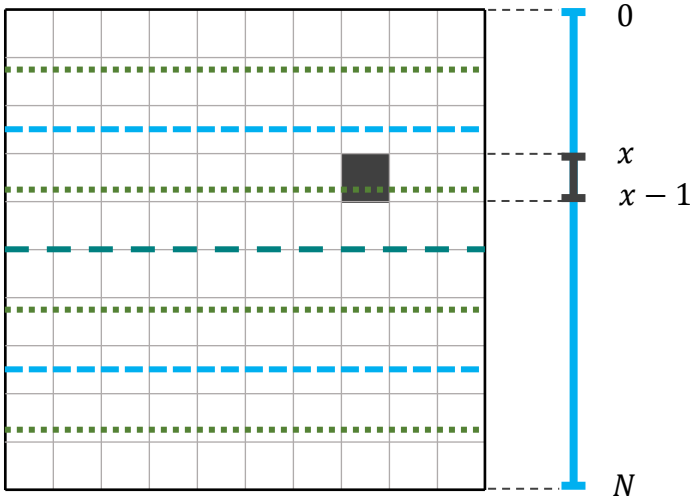
Ja k ir pāra skaitlis (vai 1), tad tie ir diagonāli ↘ un ↗ griezieni (44. zīm.):



44. zīm.

Pieeja būs šāda: katram no šiem četriem virzieniem (horizontāls, vertikāls, diagonāls ↘ vai ↗) neatkarīgi izrēķināsim, ar kuru šāda virziena griezienu sagriezīsim doto rūtiņu. Tad atbilde būs minimālais no šo četru griezienu kārtas numuriem.

Sākumā aplūkosim horizontālus griezienus. Šādiem griezieniem ir svarīga tikai X koordināta (45. zīm.). Līdz ar to varam reducēt šo apakšuzdevumu uz šādu: Ir dots nogrieznis $[0, N]$, kuru ar vienu griezienu varam sagriezt uz pusēm, un vienu pusi atmest. Cik griezienu vajag, lai pārgrieztu nogriezni $(x-1, x)$ (neieskaitot galapunktus)?



45. zīm.

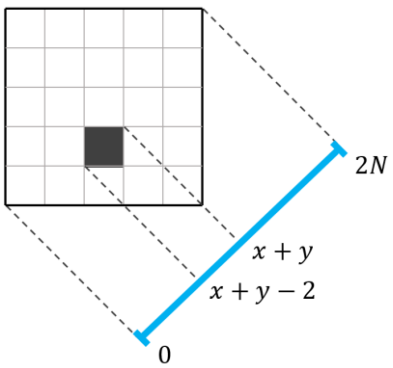
Atrisināsim vēl vispārīgāku uzdevumu: cik griezienus vajag, lai pārgrieztu nogriezni (L, R), patvaļīgiem veseliem $0 \le L < R \le N$? Šo uzdevumu risināsim rekursīvi: uzrakstīsim funkciju $\text{get}(N, L, R)$, kura atgriež vajadzīgā grieziena kārtas numuru:

- Ja $L < N/2$ un $N/2 < R$, tad atgriež 1.
- Ja $R \le N/2$, atgriež $1 + \text{get}(N/2, L, R)$.
- Ja $L \ge N/2$, atgriež $1 + \text{get}(N/2, L - N/2, R - N/2)$.

Viegli pārliecināties, ka funkcija vienkārši simulē griešanu. Funkcija strādā laikā $O(\log N)$, jo ar vienu griezienu nogrieznis tiek samazināts divreiz, bet $R - L \ge 1$.

Tagad pielāgosim šo funkciju, lai atrastu katram no četriem virzieniem tā grieziena kārtas numuru (starp visu virzienu griezieniem), kas sagrieztu doto rūtiņu.

- Horizontāli griezieni: kārtas numurs ir $2 \cdot \text{get}(N, x-1, x) - 1$.
- Vertikāli griezieni: kārtas numurs ir $2 \cdot \text{get}(N, y-1, y) - 1$.
- Diagonāli ↘ griezieni: ievērosim, ka visiem punktiem, kas pieder šādam griezienam, ir konstanta summa $x + y$ (skat. *46*. zīm.). Līdz ar to varam atkal noreducēt šo apakšuzdevumu uz funkciju get, projicējot visu uz taisnes, kas iet ↗ virzienā. Varam ievērot, ka tad vajadzīgais kārtas numurs ir $2 \cdot \text{get}(2N, x + y - 2, x + y)$.
- Diagonāli ↗ griezieni: ievērojam, ka, atspoguļojot visu konfigurāciju pret Y asi, ↗ virziens pamainās uz ↘. Šādā atspoguļošana izpaužas koordināšu maiņā $x' = n + 1 - x$, $y' = y$. Līdz ar to esam noreducējuši šo gadījumu uz iepriekšējo (↘ griezieni). Tātad šajā gadījumā atbilde ir



46. zīm.

$$2 \cdot \text{get}(2N, x' + y' - 2, x' + y') = 2 \cdot \text{get}(2N, (N + 1 - x) + y - 2, (N + 1 - x) + y) =$$
$$= 2 \cdot \text{get}(2N, N - x + y - 1, N - x + y + 1).$$

Implementācijas detaļas. Ievērosim, ka funkcijas get rekursīvajos izsaukumos bieži dalām skaitli ar 2. Lai izvairītos no reālo skaitļu tipa un atstātu visu rēķināšanu veselajos skaitļos (kas izslēgtu iespēju apaļošanas kļūdai), varam pamainīt implementāciju šādi (pareizinot visus skaitļus visur ar 2):

- Ja $2L < N$ un $N < 2R$, tad atgriež 1.
- Ja $2R \leq N$, atgriež $1 + \text{get}(N, 2L, 2R)$.
- Ja $2L \geq N$, atgriež $1 + \text{get}(N, 2L - N, 2R - N)$.

Ievērosim, ka neviens no skaitļiem nekad nepārsniedz $2N$, tāpēc tie nepārsniedz $4 \cdot 10^{18}$, un "ielien" garo veselo skaitļu tipā.

Ātrdarbība. Izpildes laika sarežģītība ir $O(\log N)$.

Sniega novākšana

Uzdevuma būtība: Dots grafs, katrai šķautnei nepieciešams izvēlēties tieši vienu no šķautnes galapunktu virsotnēm tā, lai katrai virsotnei būtu piešķirta vismaz viena šķautne.

Risinājums:

Vispirms piešķir šķautnes virsotnēm, kurām to var izdarīt viennozīmīgā veidā:

- Kamēr eksistē virsotne v, kurai pievienoto šķautņu daudzums ir ne lielāks kā 1, tikmēr:
  - Ja pievienoto šķautņu daudzums ir 0, tad virsotnei u nevar piešķirt šķautni. Risinājums neeksistē!
  - Ja pievienoto šķautņu daudzums ir 1, tad šķautni piešķir virsotnei. Turpmāk šo virsotni un šķautni, risinot uzdevumu, neapskata (izņem no grafa).

Izpildot šos soļus, atlikušajā grafā visām virsotnēm pievienoto šķautņu skaits būs vismaz 2.

Tad var secināt, ka katrā grafa komponentē (savienota grafa daļa, kurā no katras virsotnes var nokļūt uz katru citu virsotni caur vienu vai vairākām šķautnēm) eksistēs cikls.

Tad, lai atrastu korektu šķautņu piešķiršanas veidu, katrai grafa komponentei izpilda sekojošo algoritmu:

- Izmantojot meklēšanu dziļumā (DFS) atrod ciklu (DFS izpildoties sasniedz virsotni, kas jau ir apmeklēta DFS izpildes laikā).
- Izvēlas vienu šķautni no cikla. Piešķir to vienam no šķautnes galapunktiem, kuru apzīmēsim ar v. Izņem šķautni no grafa. Virsotnei v ir jau piešķirta šķautne, visas pārejās virsotnes grafa komponentē ir sasniedzamas caur vienu vai vairākām šķautnēm no virsotnes v, jo tā bija ciklā.
- Tad visus atlikušās šķautnes grafa komponentē varam piešķirt pēc šāda algoritma:
- Izpilda meklēšanu dziļumā, sākot no virsotnes v:
  - Katrā algoritma iterācijā izpildās invariants, ka iterācijā apskatāmajai virsotnei u jau ir piešķirta šķautne.
  - Kamēr virsotnei u ir šķautne uh (starp virsotnēm u un h), kas vēl nav piešķirta nevienai virsotnei, tad:
  - piešķir šķautni uh virsotnei h,
  - izsauc rekursīvi algoritmu virsotnei h, ja uz tās vēl algoritms (meklēšana dziļumā) nav tikusi izsaukta.

Izpildot meklēšanu dziļumā tiks apmeklētas visas grafa komponentes virsotnes un viegli var secināt, ka katrai virsotnei tiks piešķirta vismaz viena šķautne - šķautne no kuras tika ieiets virsotnē, lai izsauktu meklēšanu dziļumā -, izņemot sākuma virsotnei v, kurai manuāli tika piešķirta šķautne no cikla.

Sarežģītība:

$O(|V| + |E|)$, lai apstrādātu virsotnes, kurām pievienoto šķautņu skaits ir ne vairāk kā viena. To var realizēt katrai virsotnei saglabājot un atjaunojot pievienoto šķautņu skaitu, uzturot rindu ar virsotnēm, kurām šķautņu skaits ir ne vairāk kā viena.

$O(|V| + |E|)$, lai katrā grafa komponentē atrastu ciklu un pēc tam komponentē katru šķautni piešķirtu virsotnei.

Kopējā izpildes laika sarežģītība $O(|V| + |E|)$ - lineāra atkarībā no grafa izmēra.

Veikals

Uzdevuma būtība ir šķirot būtiskos (A, B, C, D) gadījumus, un katram saprast, vai process būs bezgalīgs. Tehniski risinājums ir secīgi pārbaudīt analīzē atrastos nosacījumus un atgriezt atbilstošo rezultātu. Kamēr pilna gadījumu analīze no matemātiskās puses ir diezgan laikietilpīga, implementācija kodā ir ļoti īsa un jāizmanto tikai vienkāršas operācijas, nav jārealizē nekādi algoritmi (izņemot divu skaitļu lielāka kopīgā dalītāja atrašanas funkcija).

1. $A < B$

Jau uzreiz pirmajā dienā ir nepietiekams preču skaits: veikalā ir A preces, bet pircēji grib pirkt $B > A$. Tātad veikala darbība beidzas - process ir galīgs.

Turpmāk apskatām gadījumu $A \geq B$.

2. $B > D$

Piegādāt var mazāk, nekā katru dienu pircēji grib nopirkt. Tātad pat, ja piegādātu katru dienu, preču skaits veikalā vienmēr samazinātos (par $B - D > 0$). Tas nozīmē, ka kaut kad veikalā paliks mazāk par B precēm, tātad veikala darbība beigsies - process ir galīgs.

Turpmāk apskatām gadījumu, ka $B \leq D$.

3. $B \leq C$ (un $A \geq B, B \leq D$)

3.1. Pirmkārt, pierādīsim, ka pie dotajiem nosacījumiem, noteikti pienāks tāda diena i, ka preču skaits dienas sākumā ir $a_i > C$.

• Ja $A > C$, tad tāda ir jau pirmā diena $a_1 = A > C$.

• Ja $a_1 = A < C$, tad skaidrs, ka dienas beigās notiks piegāde (ievērojiet, ka $A \geq B$, tāpēc pirmā diena noritēs veiksmīgi, un piegāde notiks). Tātad otrās dienas sākumā veikalā būs $a_2 = A - B + D$ preces. Ja $B < D$, tad $a_2 > a_1$, jeb otrās dienas sākumā būs vairāk preču nekā pirmās. Līdzīgi spriežot iegūstam $a_1 < a_2 < a_3 < .. < a_i$, kamēr būsim atraduši $a_i > C$.

• Atlikušais īpašais gadījums ir $B = D$ (un $a_1 < C$): šajā situācijā $a_2 = a_1 - B + D = a_1$, un līdzīgi $A = a_1 = a_2 = a_3 = \ldots$, jeb katras dienas sākumā veikalā ir vienāds preču skaits; šajā apakšgadījumā uzreiz skaidrs, ka process ir bezgalīgs.

3.2. Otrkārt, pierādīsim, ka, ja mums kādu dienu ir $a_i > C$ preces (3.1.), tad arī $a_{i+1} > C$.

• $a_i - B > C$. Pēc preču pirkšanas i-tajā dienā pāri paliek vairāk par C precēm. Piegāde nenotiek un nākamā diena sākas ar $a_{i+1} = a_i - B > C$ precēm.

• $a_i - B \leq C$. Dienas beigās notiks piegāde. Tā kā $B \leq D$, tad pēc piegādes otrās dienas sākumā būs $a_{i+1} = a_i - B + D \geq a_i > C$ preces.

3.3. Līdz ar to esam pierādījuši, ka (3.1.) pienāks diena ar $a_i > C$ precēm, šī nebūs pēdējā diena (jo $B \leq C < a_i$), un (3.2.) nākamajā dienā arī būs $a_{i+1} > C$ preces. No tā izriet, ka process ir bezgalīgs.

4. Turpinām ar $B > C,\ A \geq B,\ B \leq D$.

Turklāt reducēsim uzdevumu. Skaidrs, ka, ja $A >> B, C$, sākumā būs vairākas dienas, kurās būs iespējams nopirkt B preces, bet piegāde vēl nenotiks. Atrodam pirmo "interesanto dienu", proti, tādu $k \geq 0$, ka $A - kB > C$ bet $A - (k+1)B \leq C$. Citiem vārdiem, pirmo dienu, kurā pēc pirkšanas (ja iespējama) notiks piegāde. Divi gadījumi:

4.1. $A - (k+1)B < 0$. Tā kā $A - kB > C$, tad pēc k-tās pirkšanas reizes (vai arī pirmajā dienā, ja $k = 0$) veikalā paliek vairāk par C precēm, tātad piegāde nenotiek. Līdz ar to nākamās dienas sākumā ir $A - kB$ preces. Bet, tā kā $A - kB - B = A - (k+1)B < 0$, nākamajā dienā vairs nav iespējams nopirkt B preces, tātad veikals beidz darbību - process ir galīgs.

4.2. $A - (k+1)B \geq 0$. Tātad pēc $k+1$-ās pirkšanas veikalā paliek $0 \leq A - (k+1)B \leq C$ preces. Tā kā visi iespējamie gadījumi līdz šim ir izanalizēti, mēs varam pieņemt, ka šis ir sākotnējais preču skaits veikalā. Un tā kā šis skaits iegūts pēc $k+1$-ās pirkšanas, pieņemsim, ka diena sākas ar piegādi (ja $a_i \leq C$). Tātad uzdevums reducēts uz:

piegāde notiek dienas sākumā

$$a_1 = A' = A - (k+1)B$$

tātad: $0 \leq a_1 = A' \leq C < B \leq D$

5. $0 \leq a_1 = A' \leq C < B \leq D$

5.1. Pieņemsim, ka process ir bezgalīgs. Ja process ir bezgalīgs, skaidrs, ka bezgalīgi daudziem a_i izpildīsies $a_i \leq C$. Tātad noteikti sakritīs vērtības kādu divu dienu sākuma preču skaitiem, jeb $i \neq j: a_i = a_j$. Tā kā katrs a_i unikāli nosaka a_{i+1}, secinām, ka process ir ciklisks: $.. \to a_{i_1} \to a_{i_2} \to .. \to a_{i_n} \to a_{i_1} \to ..$

5.2. Šķirojam gadījumus:

• Visi $a_i \geq B$. Tātad vienmēr varēs nopirkt B preces. Tātad process bezgalīgs.

• Eksistē $i: a_i < B$. Pieņemsim, ka tad $a_i \leq C$. Tā kā a_i ir preču daudzums dienas sākumā, un $a_i \leq C$, notiks piegāde un pirms pirkšanas būs $a_i + D$ preces. Tā kā $B \leq D$, preces būs iespējams nopirkt ($a_i + D \geq a_i + B \geq B$). Tātad šajā gadījumā process ir bezgalīgs.

• Atliek gadījums, kad eksistē $i: C < a_i < B$. Tā kā $a_i > C$, piegāde nenotiek. Tā kā $a_i < B$, nav iespējams nopirkt B preces. Tātad šajā gadījumā process ir galīgs.

5.3. Esam ieguvuši, ka (pie $0 \leq a_1 = A' \leq C < B \leq D$) process vienmēr ir bezgalīgs, izņemot tad, ja eksistē $i: C < a_i < B$!

6. Apskatam kaut kādu a_i. Pierādīsim, ka $0 \leq a_i < D$.

$0 \leq a_1 < D$ pēc definīcijas

• Ja $0 \leq a_1 < D$ un piegāde nenotiek, tad $a_{i+1} = a_i - B < a_i$un tātad $0 \leq a_i + 1 < D$

• Ja $0 \leq a_1 < D$ un piegāde notiek, tad $a_{i+1} = a_i + D - B \leq a_i$ un tātad $0 \leq a_{i+1} < D$

Tā kā $a_i = a_1 - nB + mD$ (n pirkšanas un m piegādes), un $0 \leq a_i < D$, varam izteikt: $a_i = a_1 - nB \ (mod\ D)$ ($mod\ D$ - atlikums dalot ar D)

7. Mēs zinām, ka $a_i = a_1 - nB \ (mod\ D)$, un gribam noskaidrot, vai var gadīties $C < a_i < B$.

7.1. Pieņemsim, ka B un D ir savstarpēji pirmskaitļi.

Apskatam visus n starp 0 un $D-1$, ieskaitot.

Pieņemsim, ka diviem dažādiem $n_1 \neq n_2$ izpildās $n_1 B = n_2 B \ (mod\ D)$. Tad $n_1 B - n_2 B = 0 \ (mod\ D) \Rightarrow (n_1 - n_2)B = 0 \ (mod\ D) \Rightarrow n_1 - n_2 = 0 \ (mod\ D) \Rightarrow n_1 = n_2$ - pretruna. Tātad, ja n pieņem visas vērtības $0..D-1$, tad $nB \ (mod\ D)$ pieņem visas vērtības $0..D-1$.

Līdz ar to arī $a_1 - nB \ (mod\ D)$ pieņem visas vērtības $0..D-1$.

Tātad a_i pieņem visas vērtības $0..D-1$.

Atgriežamies pie jautājuma, vai var gadīties $C < a_i < B$?

• Ja $C + 1 = B$, skaidrs, ka tāds a_i neeksistē, tātad šajā gadījumā process ir bezgalīgs.

• $C + 1 < B$. Tā kā a_i pieņem visas vērtības $0..D-1$ un $B \leq D$, var secināt, ka noteikti būs tāds $a_i: C < a_i < B$. Tātad šajā gadījumā process būs galīgs.

7.2. Atliek gadījums, kad B un D nav savstarpēji pirmskaitļi.

Tātad $lkd := lkd(B, D) > 1$ (lielākais kopīgais dalītājs), un $B = lkd * B'$, $D = lkd * D'$ kaut kādiem veseliem skaitļiem $B', D' \geq 1$.

$$a_i = a_1 - nB + mD = a_1 - nB' * lkd + mD' * lkd = a_1 + lkd * (mD' - nB')$$

Citiem vārdiem: visi a_i atšķiras par lkd daudzkārtņiem: $a_i = a_1 + k * lkd \ (mod\ D)$

$$k = (mD' - nB') = -nB' \ (mod\ D')$$

Analoģiski (7.1.) spriedumam varam secināt, ka k pieņem visas vērtības $0..D'-1$

Tātad a_i pieņem visas vērtības $0..D-1$, kas vienādas ar $a_1(mod\ lkd)$

Atgriežamies pie jautājuma, vai var gadīties $C < a_i < B$?

• Ja $C + lkd < B$, tad skaidrs, ka tāds a_i eksistē, jo a_i pieņem visas vērtības, kas vienādas ar $a_1(mod\ lkd)$, un intervāls starp C un B (neieskaitot) ir garumā vismaz lkd, tātad kāds no a_i tajā atradīsies. Tātad šajā gadījumā process ir galīgs.

• Turpinām ar $C + lkd \geq B$.

$$C < a_i < B \Rightarrow C < a_1 + k * lkd < B$$

$B = lkd * B'$ un $C = lkd * C' + r,\ r < lkd$

Tā kā $C + lkd \geq B$, skaidrs, ka $B' = C' + 1$

Jautājumu var pārveidot formā: $lkd * C' + r < a_1 + k * lkd < lkd * C' + lkd.$

Kas savukārt ir ekvivalents $r < a_1(mod\ lkd)$

Tātad, ja $C\ (mod\ lkd) = r < a_1(mod\ lkd)$, process ir galīgs.

Citādi (ja $C\ (mod\ lkd) = r \geq a_1(mod\ lkd)$), process ir bezgalīgs.

2019./2020. mācību gads

Novada olimpiāde - 2020

Cik labo?

Sākumā vajag noskaidrot, kuri skaitļi ir palikuši pēc visu gājienu izdarīšanas. No dotā skaitļu intervāla mēs izņemam vairākus skaitļu intervālus, kas savā starpā var pārklāties. Lai noskaidrotu, kuri skaitļi ir palikuši, sakārtojam visu gājienu intervālu galapunktus augošā secībā (zinot, vai konkrētais skaitlis ir intervāla sākums vai beigas) un tad, ejot cauri šim sakārtotajam masīvam, var uzturēt skaitītāju, kas nosaka vai šajā brīdī visi gājienu intervāli ir beigušies. Ja visi gājienu intervāli ir beigušies, tad tas nozīmē, ka šajā brīdī sākas skaitļu intervāls, kuri ir palikuši.

Tātad atlicis noskaidrot cik labo skaitļu ir dotā skaitļu intervālā $[a, b]$. Sākumā noskaidrosim, cik ir labo skaitļu intervālā $[1, c]$. Tā kā labi ir tie skaitļi, kas dalās ar M_1 vai M_2, tad labo skaitļu skaits ir $L(c) = \frac{c}{M_1} + \frac{c}{M_2} - \frac{c}{MKD(M_1,M_2)}$. To skaitļu, kas dalās gan ar M_1, gan M_2, skaits ir jāatņem, jo tas ir ieskaitīts divas reizes - abos iepriekšējos saskaitāmajos. Tad, lai noskaidrotu cik ir labo skaitļu intervālā $[a, b]$, jāaprēķina $L(b) - L(a-1)$.

Risinājuma izpildes laika sarežģītība ir $O(GlogG)$.

Latīņu kvadrāti

Ielasot dotā rūtiņu laukuma vērtības, pārkodēsim simbolus par skaitļiem. Piemēram, simbolus, kas atbilst cipariem, par vienciparu skaitļiem no 0 līdz 9, lielos burtus - par skaitļiem no 10 līdz 35, bet mazos - par skaitļiem no 36 līdz 61. Tagad katrā laukuma rūtiņā varam izveidot 62 bitus garu masīvu, kur katrā bitā glabājas informācija vai iepriekšējās secīgās n šīs rindas rūtiņās attiecīgais simbols bija pārstāvēts, vai nē. Turklāt, lai savāktu šo informāciju, katrā pozīcijā pietiek paņemt ierakstu no iepriekšējās pozīcijas un veikt divas izmaiņas - pievienot informāciju par jauno bitu un nodzēst par to, kas tagad "izbrauc no redzamības apgabala".

Tagad varam caurskatīt rindas un katrā pozīcijā atzīmēt vērtību A_{ij} - cik ir tādas rindas pēc kārtas, kuru secīgo n rūtiņu segmenta, kas beidzas rūtiņā $(i; j)$, saturs (vērtību komplekts) ir vienāds.

Tādā pat veidā varam aprēķināt šos lielumus, veicot laukuma caurskati pa kolonnām. Finālā iegūsim vērtības B_{ij} - cik ir tādas kolonnas pēc kārtas, kuru secīgo n rūtiņu segmenta, kas beidzas rūtiņā $(i; j)$, saturs (vērtību komplekts) ir vienāds.

Beigās atliek katrai rūtiņai $(i; j)$ izanalizēt saistībā ar to uzkrāto informāciju. Ja bitu masīvu ieraksts abos virzienos sakrīt, ja katrā no virzieniem ir vismaz n vienādi ieraksti un atzīmēto bitu skaits ir tieši n, tad šādā rūtiņā beidzas derīgs latīņu kvadrāts.

Atrisinājuma izpildes laika sarežģītība ir $O(mk)$.

Lakatiņa summa

Reducēšana. Sākumā reducēsim uzdevumu uz vieglāku. Pieņemsim, ka mums ir funkcija $F(T)$, kas dotam nenegatīvam skaitlim T izrēķina lielāko skaitli, kura lakatiņa summa nepārsniedz T (ja $T = 0$, tad $F(T) = 0$).

Tad atradīsim $X = F(S_{\text{maz}} - 1)$ un $Y = F(S_{\text{liel}})$:

- Ja $X = Y$, tad lielākais skaitlis, kura lakatiņa summa nepārsniedz S_{liel}, ir stingri mazāks par S_{maz}, un atbilde ir (0, 0), jo nav neviena skaitļa, kura lakatiņa summa atrodas intervālā $[S_{\text{maz}}, S_{\text{liel}}]$.
- Pretējā gadījumā atbilde ir $(X + 1, Y)$, jo $X + 1$ ir pirmais skaitlis, kura lakatiņa summa ir vismaz S_{maz}.

Risinājums. Nepieciešams implementēt funkciju F. Aplūkosim vēlreiz skaitļa lakatiņa summas definīciju

$$S(A) = \overline{a_n a_{n-1} \dots a_2 a_1} + \overline{a_n a_{n-1} \dots a_2} + \overline{a_n a_{n-1} \dots a_3} + \dots + \overline{a_n a_{n-1}} + a_n$$

Ievērosim, ka varam to pārrakstīt citā formā:

$$S(A) = \underbrace{\overline{a_n a_n \dots a_n a_n}}_{n \text{ cipari}} + \underbrace{\overline{a_{n-1} \dots a_{n-1}}}_{n-1 \text{ cipars}} + \underbrace{\overline{a_{n-2} a_{n-2} \dots a_{n-2}}}_{n-2 \text{ cipari}} + \dots + \overline{a_2 a_2} + a_1$$

Tāpēc, ja gribam izrēķināt $F(T)$, tad varam rīkoties pēc alkatīga principa. Uzbūvēsim tādu A, ka $F(T) = S(A)$. Apzīmēsim $A = \overline{a_n a_{n-1} \dots a_2 a_1}$. Lai A būtu pēc iespējas lielāks, pirmkārt, ciparu skaitam n jābūt pēc iespējas lielākam. Savukārt, ja zināms n, tad vērtībai a_n jābūt pēc iespējas lielākai. Līdz ar to varam pārlasīt n (no 0 līdz 17, jo S_{liel} nav lielāks par $\underbrace{11\dots1}_{17 \text{ cipari}}$) un a_n (no 1 līdz 9), un izvēlamies tādus, ka

$$a_n \cdot \underbrace{11\dots1}_{n \text{ cipari}} \le T$$

un n, a_n ir pēc iespējas lielāki. Tad atkārtojam šo procesu skaitlim $T - a_n \cdot \underbrace{11\dots1}_{n \text{ cipari}}$, tādā veidā atrodot visus skaitļa A ciparus.

Piemēram, izrēķināsim $F(5436)$. Lielākais skaitlis, kas sastāv no visiem vienādiem cipariem un nepārsniedz 5436, ir 4444. Tad paliek skaitlis $5436 - 4444 = 992$. Lielākais skaitlis no vienādiem cipariem, kas nepārsniedz 992, ir 888. Paliek skaitlis $992 - 888 = 104$. Nākamais skaitlis no vienādiem cipariem tad ir 99. Pēdējais skaitlis ir $104 - 99 = 5$. Līdz ar to $F(5436) = 4895$.

Ātrdarbība. Tiek divreiz izsaukta funkcija F, kuras izpildes laika ātrdarbība atkarībā no realizācijas ir vai nu $O(\log(S_{\text{liel}})^2)$, vai arī $O(\log(S_{\text{liel}}))$.

Neder kā summa

Aplūkosim uzdevuma "Lakatiņa summa" risinājumu un tajā definēto funkciju $F(T)$ – lielāko skaitli, kura lakatiņa summa nepārsniedz T.

Atkal izrēķināsim divus skaitļus $X = F(S_{\text{maz}} - 1)$ un $Y = F(S_{\text{liel}})$. Visi skaitļi, kuru lakatiņa summa atrodas starp S_{maz} un S_{liel}, ieskaitot, ir visi skaitļi intervālā $(X, Y]$, jo var pamanīt, ka skaitļa lakatiņa summa pieaug, ja pieaug skaitlis. Līdz ar to atbilde uz uzdevumu ir skaitļu skaits intervālā $[S_{\text{maz}}, S_{\text{liel}}]$, no kura ir atņemts skaitļu skaits intervālā $(X, Y]$. Tātad atbilde uz uzdevumu ir

$(S_{\text{liel}} - S_{\text{maz}} + 1) - (Y - X)$.

Ātrdarbība. Līdzīgi kā uzdevumā "Lakatiņa summa", izpildes laika ātrdarbība ir vai nu $O(\log(S_{\text{liel}})^2)$, vai arī $O(\log(S_{\text{liel}}))$.

"Pēdējās formulas" autosacīkstes

Šis uzdevums ir visai tehnisks - nepieciešams precīzi realizēt uzdevuma tekstā aprakstīto. Pēc kārtas pa rindām ir jāievada dati par kārtējo sportistu, sacensību rezultāti (iegūtā vieta vai informācija par nepiedalīšanos/nefinišēšanu) korekti jāpārvērš punktos un, ja punktu skaits šobrīd ir lielākais, informācija par šo sportistu un iegūtajiem punktiem jāsaglabā.

Saprotams, ka atsevišķā mainīgā jāglabā šobrīd lielākais kāda sportista iegūtais punktu skaits P_{maks}.

Uzdevumu mazliet sarežģī nepieciešamība izvadīt **visu** maksimālo punktu skaitu ieguvušo sportistu numurus. Ja viena sportista gadījumā tas būtu vienkāršs veselu skaitļu tipa mainīgais, tad tagad numuru saglabāšanai nepieciešams izmantot kādu datu struktūru - piemēram, masīvu.

Tad, atkarībā no aprēķinātā punktu skaita P vai nu šim masīvam jāpievieno sportista numurs (ja $P = P_{maks}$), vai arī masīvs jānotīra, tam jāpievieno sportista numurs un P kļūst par jauno P_{maks}.

Ja $P < P_{maks}$, tad nekas nav jādara - šis sportists nevar būt starp labākajiem kopvērtējumā.

Pēc visu sportistu datu apstrādes nepieciešams izvadīt P_{maks}, masīva elementu (sportistu numuru) skaitu un pēc tam arī pašus sportistu numurus.

Torņi

Uzdevuma būtība: Ir V vaicājumi, kur katrā var būt viena no darbībām:

- Uzbūvē torni augstumā h,
- Nojauc vienu no jau uzbūvētiem torņiem augstumā h,
- Noskaidro, kāds ir k-tā augstākā torņa augstums.

Risinājums:

Binārā meklēšana segmentu kokā vai *treap* datu struktūrā.

Apskatīsim risinājumu, izmantojot segmentu koku.

Lai nebūtu jāraksta dinamisks segmentu koks un samazinātu izpildes laika sarežģītību binārajā meklēšanā, vispirms veiksim torņu augstumu vērtību kompresiju:

- No ievaddatiem nolasām visus dažādos torņu augstumus, kas parādās vaicājumos, tos sakārtojam nedilstošā secībā izslēdzot atkārtojumus (nav svarīgi, cik torņi ir ar vienādu augstumu). Tad turpmāk torni ar augstumu h uzdevumā aizvietojam ar tā pozīciju šajā sakārtotajā masīvā. Jauno torņa augstumu var efektīvi atrast izmantojot bināro meklēšanu vai iebūvētās datu struktūras, kā `std::map`. Pārveidojot augstumus uz intervālu $[1; V]$, tiek saglabātas savstarpējās īpašības par to, kuri torņi ir lielāki, mazāki par citiem, padarot torņu augstumu vērtības mazākas un daudz vieglāk apstrādājamas.

Lai atrisinātu uzdevumu:

1. Uzkonstruē segmentu koku intervālā $[1; V]$, kas glabās uzbūvēto torņu skaitu.
2. Apstrādā pieprasījumu:

- Uzbūvē torni augstumā h:
  - Segmenta kokā pozīcijā h pieskaita 1. Izpildes laika sarežģītība: $O(logV)$
- Nojauc torni augstumā h:
  - Segmenta kokā pozīcijā h atņem 1. Izpildes laika sarežģītība: $O(logV)$
- Atrod k-to šobrīd augstāko uzbūvēto torni:
  - Ja intervālā $[1; V]$ ubūvēto torņu skaits ir mazāks par k, tad atbilde neeksistē.
  - Citādi veic bināro meklēšanu uz uzbūvēto torņu skaitu:
  - Risinājums ar sarežģītību $O(\log^2 V)$
    - Binārās meklēšanas iterācijā pārbaudot vērtību h no segmentu koka noskaidro uzbūvēto torņu skaitu intervālā $[1; h]$ un izdara secinājumus.

  Var pamanīt, ka var izmantot segmentu koka uzbūves binārās īpašības, lai optimizētu bināro meklēšanu.
  - Risinājums ar sarežģītību $O(\log V)$

    Apstaigā segmentu koku sākot ar saknes intervālu $[1; V]$:

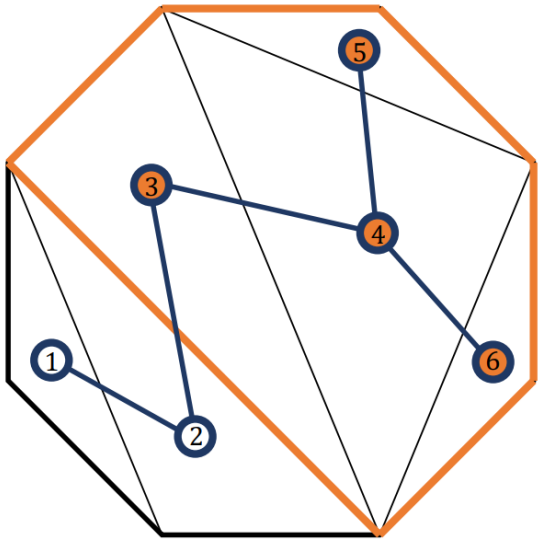
    - Ja intervāla izmērs ir 1 (tam var piederēt torņi ar augstumu h', tad vaicājuma atbilde atrasta, tā ir h').
    - Citādi apskata intervāla "bērnu" intervālus:
    - Ja no "bērniem" intervālā pa labi uzbūvēto torņu skaits cnt_R ir lielāks, vienāds par meklēto skaitu k, tad rekursīvi apstaigā labējo intervālu.
    - Citādi rekursīvi apstaigā intervālu pa kreisi, vērtību k aizstājot ar $k - cnt_R$, jo tika noskaidrots, ka cnt_R torņi ir augstāki, un tie vairs netiks apskatīti intervālā pa kreisi.

Izpildes laika sarežģītība:
Koordinašu kompresija: $O(V \times \log V)$
Segmentu koka uzkonstruēšana: $O(V \times \log V)$
Katrs pieprasījums: $O(\log V)$
Visu pieprasījumu apstrāde: $O(V \times \log V)$
Kopējā risinājuma izpildes laika sarežģītība: $O(V \times \log V)$

Valsts olimpiāde - 2020

Daudzstūra sadalīšana

Izveidosim šādu *grafu*: virsotnes atbilst trijstūriem, kuros ir sadalīts daudzstūris, un šķautne starp divām virsotnēm ir tad, ja attiecīgiem trijstūriem ir kopīga mala (diagonāle dotajā daudzstūrī). Šajā grafā ir $N-2$ virsotnes un $N-3$ šķautnes. Tā kā šis grafs ir sakarīgs, un šķautņu skaits ir par 1 mazāks nekā virsotņu skaits, tad tas ir *koks*.

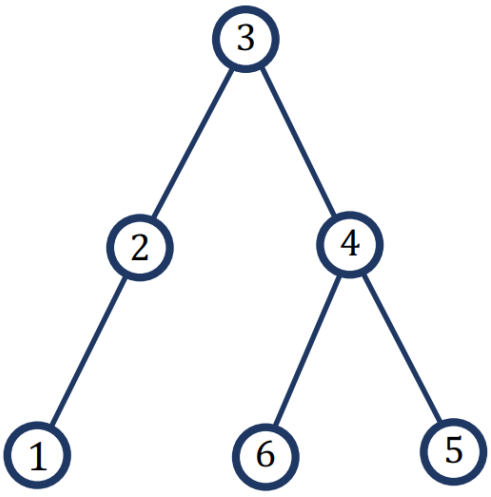


47. zīm.

Tagad aplūkosim, kam atbilst viens k-stūris šajā kokā. Ievērosim, ka katrs k-stūris sastāv no $k-2$ trijstūriem. Līdz ar to viens k-stūris atbilst vienam apakškokam no $k-2$ virsotnēm mūsu izveidotajā kokā.

Piemēram, 47. zīmējumā parādītajā sadalījumā iezīmētajam sešstūrim atbilst apakškoks no četrām iekrāsotajām virsotnēm (3., 4., 5., 6. virsotnes).

Lai izrēķinātu k-stūru skaitu visiem k, lietosim *dinamiskās programmēšanas* metodi. Sākumā pārveidosim mūsu koku par *sakņotu* – izvēlamies vienu no virsotnēm un pasludinām tās kaimiņus par tās *bērniem*, bet sakni par to *vecāku*. Tālāk rekursīvi, sākot no saknes bērniem, izejam cauri virsotnēm v un pasludinām tās kaimiņus par tās *bērniem*, izņemot pašu v vecāku. Savukārt v bērniem pasludinām v kā to vecāku. Šādu koka interpretāciju dažreiz sauc arī par "pakarināšanu". Papildus lieta, specifiska šim uzdevumam – kā sakni izvēlēsimies tādu virsotni, kur atbilstošajam trijstūrim vismaz viena mala ir dotā daudzstūra mala (iemeslu tam redzēsim vēlāk).



48. zīm.

Piemēram, pakarinot aplūkoto koku aiz 3. virsotnes, iegūstam šādu sakņotu koku (48. zīm., katrai šķautnei vecāks ir augšā, un bērns apakšā):

Ar virsotnes *pēcteci* apzīmēsim jebkuru virsotni, kas ir vai nu šī pati virsotne, vai nu jebkura tās bērna pēctecis. Par *virsotnes apakškoku* sauksim tādu sakņota koka apakškoku, kas satur tikai šīs virsotnes pēctečus. Piemēram, 4. virsotnes apakškoks sastāv no virsotnēm ar numuriem 4., 5. un 6., un šie ir visi tās pēcteči.

Izveidosim 2-dimensiju masīvu dp, kur $\mathrm{dp}[v][i]$ apzīmē, cik ir apakškoku izmērā i, kas pilnībā atrodas v apakškokā, un satur arī pašu virsotni v. Aprēķināsim šī masīva elementu vērtības, sākot no koka lapām un ejot uz augšu.

Pašām lapām uzstādām:

- $\mathrm{dp}[v][1] = 1$, kas atbilst apakškokam, kas sastāv no vienas pašas virsotnes v.
- $\mathrm{dp}[v][i] = 0$ visiem $i > 1$.

Tagad aplūkosim patvaļīgu virsotni v, kas nav lapa. Ievērosim, ka tai ir ne vairāk kā divi kaimiņi, jo:

- Katrai virsotnei kokā ir ne vairāk kā trīs kaimiņi, jo viena virsotne atbilst trijstūrim, kuram ir trīs malas.
- Ja v nav sakne, tad viena no šķautnēm savieno to ar savu vecāku; tas nozīmē, ka v var būt ne vairāk par diviem bērniem.
- Ja v ir sakne, tad tā kā v izvēlējāmies tā, lai tai atbilstu trijstūris, kuram viena mala sakrīt ar dotā daudzstūra malu, tad v arī var būt ne vairāk kā divi bērni.

Pieņemam, ka bērniem esam izrēķinājuši masīvu dp, jo rēķinām kokā no lejas uz augšu. Tagad mums ir divi gadījumi, atkarībā no tā, cik ir bērnu.

- Virsotnei v ir tieši viens bērns u. Tad uzstādām
  - $\mathrm{dp}[v][1] = 1$, tas atbilst apakškokam no vienas virsotnes v.
  - $\mathrm{dp}[v][i] = \mathrm{dp}[u][i-1]$ visiem $i > 1$, jo jebkurš apakškoks, kas atrodas v apakškokā un satur v, sastāv no virsotnes v un u apakškoka, kas satur u.
- Virsotnei v ir tieši divi bērni u un w. Tad sākumā uzstādam $\mathrm{dp}[v][i] = 0$ visiem i, kā arī $\mathrm{dp}[u][0] = \mathrm{dp}[w][0] = 1$. Tad, ja ir kāds apakškoks, kas ir v apakškokā un satur v, tad tas sastāv no v, i virsotnēm no u apakškoka (iespējams, 0), un j virsotnēm no w apakškoka (arī, iespējams, 0). Līdz ar to pārlasam i no 0 līdz $N-3$, j arī no 0 līdz $N-3$, un pieskaitām iekš $\mathrm{dp}[v][1+i+j]$ skaitu $\mathrm{dp}[u][i] \cdot \mathrm{dp}[w][j]$. Reizināšana šeit ir, jo mums ir $\mathrm{dp}[u][i]$ izvēles paņemt apakškoku izmērā i no u apakškoka, un $\mathrm{dp}[w][j]$ izvēles paņemt apakškoku izmērā j no w apakškoka.

Tādā veidā izrēķinām masīva dp elementu vērtības. Lai beigās aprēķinātu, cik ir apakškoku izmērā k, mums ir jāsaskaita visas vērtības $\mathrm{dp}[v][k]$ visām virsotnēm v, jo katrs apakškoks izmērā k ir tieši viena "visaugstākā" virsotne kokā v, un tas tiek pieskaitīts tieši vienā $\mathrm{dp}[v][k]$.

Ātrdarbība. Dinamiskajai programmēšanai ir $O(N)$ soļi, un viena soļa izpildes laika sarežģītība veidojas no i un j pārlases, kas ir $O(N^2)$ iespējas. Līdz ar to kopējā izpildes laika sarežģītība ir $O(N^3)$.

Divas iekavas

Korektu iekavu izteiksmi pārvērtīsim iekavu bilanču (veselu skaitļu) virknē pēc šādiem noteikumiem:

- pirms izteiksmes apstrādes iekavu bilance ir 0;
- atverošai iekavai iekavu bilances vērtība tiek palielināta par 1;
- aizverošai iekavai iekavu bilances vērtība tiek samazināta par 1.

Piemēram, iekavu izteiksme `()(())` tiks pārvērsta iekavu bilanču virknē `1, 0, 1, 2, 1, 0`.

Protams, nekas mums netraucē šīs bilances aprēķināt arī patvaļīgai virknei, kas sastāv tikai no atverošām un aizverošām iekavām.

Jebkurai iekavu bilanču virknei (pēc konstrukcijas) ir spēkā īpašība, ka divi tās blakus elementi atšķiras tieši par 1 un pirmais virknes elements ir vai nu 1, vai -1.

Ja iekavu izteiksme, no kuras tika iegūta iekavu bilanču virkne, bija korekta, tad papildus ir spēkā vēl divas īpašības:

- visi virknes elementi ir nenegatīvi, un
- pēdējais virknes elements ir 0.

Viegli saprast, ka no iekavu bilanču virknes viennozīmīgi iespējams atjaunot iekavu virkni.

Kā mainās iekavu bilanču virkne, ja kāda iekava tiek apgriezta otrādi? Ja atverošā iekava tiek nomainīta ar aizverošo, tad, sākot no šīs vietas, visi iekavu bilanču virknes elementi tiek samazināti par divi. Piemēram, ja `()(())` → `())())`, tad `1, 0, 1, 2, 1, 0` → `1, 0, -1, 0, -1, -2`.

Līdzīgi, nomainot aizverošo iekavu uz atverošo, sākot no šīs vietas, visi iekavu bilanču virknes elementi tiek palielināti par divi. Piemēram, ja `()(())` → `((()))`, tad `1, 0, 1, 2, 1, 0` → `1, 2, 3, 4, 3, 2`.

Tā kā mēs vēlamies doto iekavu virkni pārveidot par korektu iekavu virkni, tad skaidrs, ka tas var būt iespējams tikai tad, ja atbilstošās iekavu bilanču pēdējais elements ir -4, 0 vai 4. Ievērojiet, ka tas ir tikai nepieciešamais, bet ne pietiekamais nosacījums. Piemēram, iekavu virkni `)))(((` ar divu iekavu apgriešanu nevar pārveidot par korektu, lai gan tās iekavu bilanču virknes pēdējais elements ir 0.

Uzdevumu risināsim šādi:

Pēc kārtas ielasīsim iekavas, aprēķināsim iekavu bilances vērtību b un noteiksim, cik veidos varam līdz šim ielasīto iekavu izteiksmes sākuma fragmentu pārveidot par korektas iekavu izteiksmes sākuma fragmentu. Acīmredzami, ka korektu iekavu izteiksmi neizdosies iegūt no nekorekta sākumfragmenta. Tāpēc visu laiku jānodrošina izteiksmes (vai virknes) sākumdaļas nepretrunība, ja vien tas ar divu iekavu nomainīšanu uz pretējo ir panākams.

Šai vajadzībai ieviesīsim sešus skaitītājus:

- v_0 - korekto sākuma fragmentu skaits, ja neviena iekava līdz šim nav tikusi apgriezta;
- v_1 - korekto sākuma fragmentu skaits, ja līdz šim tieši viena atverošā iekava ir pārveidota par aizverošo (pārveidotajā virknē bilances vērtība būtu $b-2$);
- v_2 - korekto sākuma fragmentu skaits, ja līdz šim tieši viena aizverošā iekava ir pārveidota par atverošo (pārveidotajā virknē bilances vērtība būtu $b+2$);
- v_3 - korekto sākuma fragmentu skaits, ja līdz šim tieši divas atverošās iekavas ir pārveidotas par aizverošām (pārveidotajā virknē bilances vērtība būtu $b-4$);
- v_4 - korekto sākuma fragmentu skaits, ja līdz šim pārveidotas tieši divas iekavas - viena atverošā un viena aizverošā (pārveidotajā virknē bilances vērtība sakristu ar b);
- v_5 - korekto sākuma fragmentu skaits, ja līdz šim tieši divas aizverošās iekavas ir pārveidotas par atverošām (pārveidotajā virknē bilances vērtība būtu $b+4$);

Pirms izteiksmes apstrādes v_0 vērtība ir 1, bet $v_1, \dots, v_5$ vērtības ir 0. Apstrādājot izteiksmē ietilpstošās iekavas, atjaunosim šo mainīgo vērtības pēc šāda algoritma:

```
bilance = 0
cikls
    Ielasa_kārtējo_iekavu(c)
    ja (c='(')
          bilance := bilance + 1
          ja (bilance >= 4) v3 += v1 citādi v3 = 0
          ja (bilance >= 0) v4 += v2 citādi v4 = 0
          ja (bilance >= 2) v1 += v0 citādi v1 = 0
    citādi // aizverošā iekava
          bilance := bilance - 1
          ja (bilance >= 0) v4 += v1 citādi v4 = 0
          ja (bilance >= -4) v5 += v2 citādi v5 = 0
          ja (bilance >= -2) v2 += v0 citādi v2 = 0
          ja (bilance < 0) v0 = 0
          ja (bilance < 2) v1 = 0
          ja (bilance < 4) v3 = 0
```

Saturīgi, mainīgo izmaiņas atspoguļo atbildes uz jautājumu: "Vai šo iekavu apgriežot uz pretējo mēs varam iegūt korektas iekavu izteiksmes sākuma fragmentu un, ja varam, tad cik veidos?"

Piemēram, rinda

```
          ja (bilance >= 4) v3 += v1 citādi v3 = 0
```

nozīmē:

“Ja iekavu bilances vērtība ir pietiekami liela (tā, ka divu atverošo iekavu nomaiņa ar aizverošajām, jeb iekavu bilances vērtības samazināšana par 4, izteiksmi nepadarīs par nekorektu), tad varam nomainīt šo atverošo iekavu par aizverošo un kopējais variantu skaits jāpalielina par v1 - šī iekava kā otrā pārī ar visām tām, ko iepriekš drīkstēja nomainīt kā pirmo atverošo uz pirmo aizverošo.”

Aplūkosim, kā mainīgo vērtības mainās apstrādājot iekavu izteiksmi `())))(()((`:

	Pirms apstr.	(	)	)	)	(	)	(	(
bilance	0	1	0	−1	−2	−1	−2	−1	0
v0	1	1	1	0	0	0	0	0	0
v1	0	0	0	0	0	0	0	0	0
v2	0	0	1	2	2	2	2	2	2
v3	0	0	0	0	0	0	0	0	0
v4	0	0	0	0	0	0	0	0	**2**
v5	0	0	0	1	3	3	5	5	5

Atkarībā no bilances beigu vērtības, jāizvada mainīgā v_3 (ja bilance ir 4), v_4 (ja bilance ir 0) vai v_5 (ja bilance ir -4) vērtība. Šoreiz jāizvada v_4 vērtība (2).

Atbilstošās korektās iekavu virknes būtu `(())()()` un `()()()()`. Patiesībā, sekojot vērtību izmaiņām tabulā, ir iespējams atrast visas nepieciešamās virknes, bet tas būtu tehniski sarežģītāk, un šajā uzdevumā tas nav prasīts.

Ko nozīmē pārējās beigu vērtības?

$v_0 = 0$ nozīmē, ka izteiksme nav korekta (var redzēt, ka tā kļuva nekorekta jau sākot ar trešo iekavu no sākuma).

$v_1 = 0$ nozīmē, ka šajā iekavu izteiksmē nevarēja nomainīt vienu atverošo iekavu pret aizverošo tā, lai pārveidotā izteiksme būtu korektas iekavu izteiksmes sākuma fragments.

$v_2 = 2$ nozīmē, ka šajā iekavu izteiksmē vienu aizverošo iekavu pret atverošo nomainīt tā, lai pārveidotā izteiksme būtu korektas iekavu izteiksmes sākuma fragments, varēja divos atšķirīgos veidos - nomainot otro vai trešo iekavu.

$v_3 = 0$ nozīmē, ka šajā iekavu izteiksmē nevarēja nomainīt divas atverošās iekavas pret divām aizverošajām tā, lai pārveidotā izteiksme būtu korektas iekavu izteiksmes sākuma fragments.

$v_5 = 5$ nozīmē, ka šajā iekavu izteiksmē divas aizverošās iekavas pret divām atverošajām nomainīt tā, lai pārveidotā izteiksme būtu korektas iekavu izteiksmes sākuma fragments, varēja piecos atšķirīgos veidos:

`(`**`((`**`)()((`, `(`**`(`**`)`**`(`**`()((`, `(`**`(`**`))(`**`(`**`((`, `()`**`((`**`()((` un `()`**`(`**`)(`**`(`**`((`.

Vēl var ievērot, ka mainīgo vērtību atjaunošana notiek tā, ka nav nepieciešams ieviest papildus mainīgos. Risinājuma izpildes laika sarežģītība ir $O(N)$ - lineāra atkarībā no iekavu izteiksmes garuma.

Formula

Uzdevumu iespējams risināt rijīgi, pēc kārtas apstrādājot operāciju virkņu daļas.

Sadalīsim visus virknes elementus *porcijās*. Katra porcija ir vai nu atsevišķa operācija vai operāciju grupa, ja virknes pierakstā tā bija iekļauta iekavās.

Piemēram, virknes pieraksts “`Jota,(Alfa,Beta),Tau,(Ro,Kapa,Beta).`” satur četras porcijas: “`Jota`”, “`Alfa,Beta`”, “`Tau`” un “`Ro,Kapa,Beta`”.

Varam uzskatīt, ka katrā brīdī mums ir divas operāciju kopas: kas_ir un ko_vajag. Kopā ko_vajag glabāsim aktuālo porciju no produkta izgatavošanai nepieciešamās operāciju virknes un nākamo porciju ielasīsim tikai tad, ja iepriekšējās porcijas operācijas būs izdevies realizēt. Ja nebūs iespējams izpildīt

kārtējās porcijas operāciju(vai operācijas), tad nebūs iespējams izpildīt visu operāciju virkni. Kopā kas_ir glabāsim tās operācijas, kuru izpildi šobrīd varam nodrošināt un šo kopu "uzpildīsim" ar porcijām no Raivja iekārtas.

Tātad pieņemsim, ka šobrīd ko_vajag satur tikai operāciju o_x - t.i., kā kārtējo ir nepieciešams realizēt operāciju o_x, bet pieejamo operāciju kopā ir operācijas o_y un o_z. Ja $o_x = o_y$vai $o_x = o_z$ tad nepieciešamo operāciju esam realizējuši, no kopām kas_ir un ko_vajag izmetam vienādos elementus, un varam ielasīt nākamo realizējamo operāciju. Ja $o_x \neq o_y$ un $o_x \neq o_z$, tad šī Raivja iekārtas operācija mums neder un arī tālāk nebūs izmantojama. Varam visus kas_ir elementus dzēst un ielasīt nākamo porciju no Raivja iekārtas operāciju virknes. Ja porcijā ietilpst vairāk nekā viena operācija, rīkojamies līdzīgi - katrā brīdī cenšamies realizēt ***jebkuru*** operāciju no ko_vajag, un kas_ir saturu dzēšam un jaunu porciju ielasām tikai tad, ja nevienu ko_vajag operāciju vairs nevaram realizēt.

Process beidzas veiksmīgi tikai tad, ja visas nepieciešamās operācijas ir realizētas - t.i., ko_vajag ir tukša un vairs nav ko tajā ielasīt.

Uzdevuma tekstā dotajā piemērā nepieciešams realizēt operāciju virkni "`Alfa,Beta,(Ro,Tau),Beta.`", bet Raivja iekārta var realizēt virkni "`Jota,(Alfa,Beta),Tau,(Ro,Kapa,Beta).`"

Kopu saturs tad mainās šādi:

ko_vajag	kas_ir	Komentārs
`Alfa`	`Jota`	Vajadzīgo operāciju realizēt nevar - dzēš kas_ir saturu un ielasa nākamo porciju.
`Alfa`	`Alfa, Beta`	`Alfa` var realizēt - vienādie elementi tiek izmesti. ko_vajag tiek ielasīta nākamā porcija.
`Beta`	`Beta`	`Beta` var realizēt - vienādie elementi tiek izmesti. Abās kopās tiek ielasītas nākamās porcijas.
`Ro, Tau`	`Tau`	`Tau` var realizēt - vienādie elementi tiek izmesti. kas_ir tiek ielasīta nākamā porcija.
`Ro`	`Ro, Kapa, Beta`	`Ro` var realizēt - vienādie elementi tiek izmesti. ko_vajag tiek ielasīta nākamā porcija.
`Beta`	`Kapa, Beta`	`Beta` var realizēt - vienādie elementi tiek izmesti. Kopā ko_vajag vairs nav ko ielasīt - process sekmīgi pabeigts - ir realizēta operāciju virkne "`Alfa,Beta,Tau,Ro,Beta.`"

No risinājuma ātrdarbības viedokļa svarīgi ir mācēt ātri atrast, vai kopā kas_ir atrodas nepieciešamā operācija.

Galapunkti

Šo uzdevumu var atrisināt ar datu struktūras, ko sauc par steku, palīdzību. Sadalīsim uzdevumu divās daļās – katram derīgam fragmentam $[i, j]$, teiksim, ka tas ir pirmā tipa segments, ja $a_i \geq a_j$ un otrā tipa, ja $a_i < a_j$.

Skaitīsim atsevišķi pirmā veida fragmentus.

Apstrādāsim virknes elementus pēc kārtas un stekā glabāsim neaugošu elementu virkni. Atnākot jaunam elementam, mēs vēlamies noskaidrot, cik derīgi pirmā veida fragmenti var beigties ar šo elementu. Tas nozīmē, ka visiem fragmenta elementiem ir jābūt mazākiem par fragmenta pirmo elementu. Lai to aprēķinātu, ar steku darbojamies sekojošā veidā – izmetam no steka visus tos elementus, kas ir mazāki par jauno elementu, jo tie vairs nevar būt derīgu pirmā veida fragmentu pirmie elementi. Visi stekā palikušie elementi (un tikai tie) ir derīgi pirmā veida fragmentu pirmie elementi, tātad to skaits arī ir meklētā vērtība. Un tad pievienojam stekam mūsu elementu.

Otrā veida fragmentus var saskaitīt līdzīgā veidā kā pirmā veida fragmentus, tikai apstrādājot virkni no otra gala un uzmanoties, lai derīgie fragmenti, kuriem abi galējie elementi ir vienādi, netiktu ieskaitīti divas reizes.

Risinājuma izpildes laika sarežģītība ir $O(N)$.

Skandalozā izrāde

Šo uzdevumu var atrisināt ar datu struktūru, ko sauc par segmentu koku – šajā uzdevumā var lietot divus atsevišķus segmentu kokus. Pirmajā glabāsim, kuri skatītāji vēl skatās izrādi, lai varētu noskaidrot, uz kuru pusi izies katrs no skatītājiem. Otrajā segmentu kokā katram skatītājam glabāsim cik reizes tas līdz šim tika iztraucēts.

Pirmajā segmentu kokā sākumā inicializējam, ka visi skatītāji ir atnākuši un segmentu koka virsotnēs glabājam cik attiecīgajā apakšsegmentā vēl ir sēdošu skatītāju. Kad skatītājs iet prom, tad noskaidrojam cik vēl sēdoši skatītāji sēž pa kreisi no tā, cik pa labi, un tādā veidā varēsim noteikt, uz kuru rindas galu šis skatītājs izies. Un pēc tam atzīmējam, ka šis skatītājs ir aizgājis un atjaunojam visas attiecīgo apakšsegmentu vērtības.

Otrajā segmentu kokā sākumā inicializējam, ka neviens skatītājs nav iztraucēts. Tad, kad kāds skatītājs iet prom, tad visam apakšegmentam no viņa vietas līdz tam galapunktam uz kuru viņš iet ārā, pieskaitām viens, jo visas šīs vietas tika iztraucētas – uzskatīsim ka arī tukšās vietas tiek traucētas. Tajā brīdī, kad skatītājs iet prom no izrādes, noskaidrojam, cik reizes viņa vieta tika traucēta, un pieskaitām šo vērtību atbildei. Visas nākamās šīs vietas traucēšanas reizes mums neinteresē, jo skatītājs jau ir aizgājis.

Risinājuma izpildes laika sarežģītība ir $O(NlogN)$.

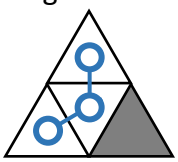
Trīsstūru labirints

Šis uzdevums sastāv no diviem atšķirīgiem apakšuzdevumiem ar dažādiem risinājumiem:

- Pirmajā apakšuzdevumā laukuma izmērs ir mazs ($R \times T \leq 10^6$), bet iekrāsoto trīsstūru skaits ir liels ($N \leq 10^6$). Tas atbilst pirmajiem četriem apakšuzdevumiem.
- Otrajā apakšuzdevumā laukuma izmērs ir liels ($R \leq 10^6$, $T \leq 2 \cdot 10^6$), bet iekrāsoto trīsstūru skaits ir mazs ($N \leq 2000$). Turklāt ir dots, ka nav iekrāsoti trīsstūri vienā no maršrutiem, kas iet gar labirinta malām. Tas atbilst piektajam apakšuzdevumam.

Pirmais apakšuzdevums.

Izveidosim *grafu* šādā veidā: katram no neiekrāsotajiem trīsstūriem atbilst viena grafa virsotne, iekrāsotie trīsstūri grafā neparādās. Divas virsotnes ir savienotas ar šķautni, ja atbilstošajiem trīsstūriem ir kopīga mala. Piemēram, 49. zīmējumā parādīts grafs četriem savienotiem trīsstūriem.



49. zīm.

Tad, lai atrastu mazāko soļu skaitu S_{AB}, varam lietot algoritmu *meklēšana plašumā* šajā grafā, sākot no virsotnes A. Šis algoritms mums sniegs masīvu d, kur katrai virsotnei v skaitlis $d[v]$ ir vienāds ar īsāko ceļu šajā grafā no A līdz v. Tad $S_{AB} = d[B]$.

Tagad aplūkosim, kā aprēķināt īsāko ceļu skaitu. Paralēli meklēšanai plašumā rēķināsim arī ar *dinamisko programmēšanu* - masīvu s, kur $s[v]$ būs vienāds ar dažādo ceļu skaitu no virsotnes A uz virsotni v.

Sākumā uzstādām $s[A] = 1$, no virsotnes A pašai uz sevi ir tieši viens ceļš (palikt uz vietas). Tad, pieņemsim, ka aplūkojam konkrētu virsotni v. Tā kā virsotnes aplūkojam meklēšanā plašumā, tad jau

tika apskatītas visas virsotnes u tādas, ka $d[u] < d[v]$. Varam pieņemt, ka visām tādām virsotnēm u vērtība $s[u]$ jau ir izrēķināta.

Tad, lai izrēķinātu $s[v]$, aplūkojam visus v kaimiņus w tādus, ka $d[w] + 1 = d[v]$. Visi ceļi uz v garumā $d[v]$ iet caur vienu no šīm virsotnēm w. Līdz ar to $s[v]$ ir vienāds ar visu šo virsotņu $s[w]$ summu.

Ātrdarbība. Meklēšana plašumā strādā laikā $O(\text{grafa virsotņu skaits} + \text{grafa šķautņu skaits})$. Virsotņu skaits nepārsniedz kopējo trīsstūru skaitu, kas ir RT. Katram no trīsstūriem var būt ne vairāk kā trīs šķautnes, un katra šķautne savieno divus trīsstūrus, līdz ar to kopējais šķautņu skaits nepārsniedz $\frac{3RT}{2}$. Tāpēc algoritms strādā laikā $O\left(RT + \frac{3RT}{2}\right) = O(RT)$.

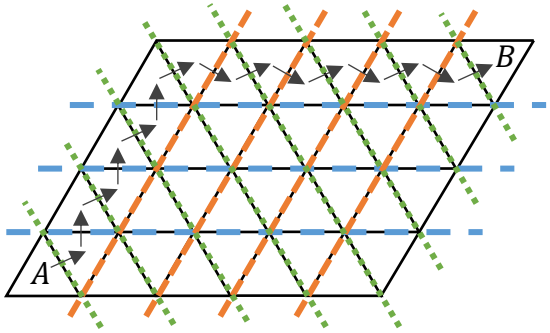
Otrais apakšuzdevums.

Svarīgs nosacījums šajā uzdevumā – tā kā nav iekrāsoti trīsstūri vienā no maršrutiem, kas iet gar labirinta malām, tad eksistē ceļš ar minimālo iespējamo garumu, kas ir vienāds ar $2R + T - 3$. Pierādīsim, ka tas ir mazākais iespējamais gājienu skaits.

- Lai nokļūtu no punkta A līdz B, ir noteikti kaut kādā brīdī jāšķērso visas robežas, kas atdala dažādas rindas. Tādu robežu skaits ir $R - 1$.
- Tāda paša iemesla dēļ ir jāšķērso visas diagonālas robežas, kas iet slīpi uz augšu pa labi. Tādu skaits ir $\frac{T}{2} - 1$.
- Jāšķērso arī visas diagonālas robežas, kas iet slīpi no augšas uz leju. Tādu skaits ir $R - 1 + \frac{T}{2}$.

Ar vienu gājienu mēs varam šķērsot tikai vienu no šīm robežām. Līdz ar to minimālais gājienu skaits, kas vajadzīgs, lai tiktu no A uz B, ir $(R - 1) + \left(\frac{T}{2} - 1\right) + \left(R - 1 + \frac{T}{2}\right) = 2R + T - 3$.

Savukārt ceļš, kas iet gar labirinta malām, arī šķērso katru no šīm robežām tieši vienu reizi, un līdz ar to tam ir minimāls garums. Tāda ceļa piemēru un kā tas krusto robežas var redzēt 50. zīmējumā.



50. zīm.

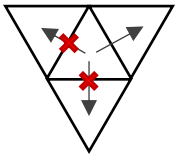
Līdz ar to $S_{AB} = 2R + T - 3$. Atliek izrēķināt, cik dažādi ceļi ir ar šādu garumu; nosauksim tādus par "minimāliem" ceļiem.

Ievērosim, ka jebkuram minimālam ceļam ir jākrusto katru no robežām tieši vienu reizi. Līdz ar to varam secināt sekojošus divus apsvērumus:

- Minimāls ceļš nevar iet uz leju.
- Minimāls ceļš nevar iet uz leju pa kreisi.

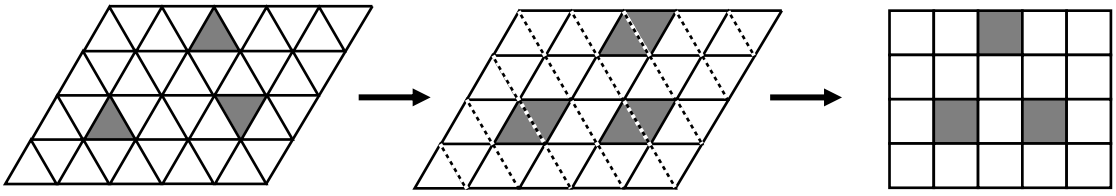
Ja kāds no šiem nosacījumiem neizpildītos, tad ceļš ietu "atpakaļ", un noteikti krustotu kādu no robežām vismaz divas reizes, tātad nebūtu minimāls.

Savukārt, no šiem novērojumiem varam tālāk izsecināt, ka, ja ceļš nokļūst tādā trīsstūrī, kurš ir orientēts tādā pašā veidā, kā A trīsstūris, tad nākamais trīsstūris ceļā var būt tikai tā kaimiņš pa labi (skat. 51. zīm.).



51. zīm.

Tas nozīmē, ka šādā rūtiņā gājiens ir viennozīmīgs, un mēs varam sagrupēt visus trīsstūrus pa divām virsotnēm. Ja kāds no trīsstūriem bija iekrāsots, tad attiecīgajā pārī tagad iekrāsojam abus trīsstūrus (jo nevienā no tiem nevar nokļūt minimāls ceļš)(skat. 52. zīm.).



52. zīm.

Tagad varam ievērot, ka īstenībā esam ieguvuši $R \times \frac{T}{2}$ rūtiņu taisnstūri, kurā dažas no rūtiņām ir iekrāsotas, kuras mēs nevaram apmeklēt. Līdz ar to esam reducējuši mūsu uzdevumu uz sekojošu: mums ir dots $R \times K$ rūtiņu režģis (kur $K = \frac{T}{2}$), vajag saskaitīt, cik ir dažādu ceļu, kas ved no kreisās apakšējās rūtiņas uz labējo augšējo, un iet tikai uz augšu un pa labi.

Savukārt, lai atrisinātu šo uzdevumu, mums vajadzēs papildus apsvērumus.

Ceļu skaits taisnstūrī. Mums noderēs zināt atbildi vienkāršākam uzdevumam. Pieņemsim, ka ir dots taisnstūris ar izmēriem $P \times Q$ (bez iekrāsotām rūtiņām). Cik veidos no apakšējā kreisā stūra var nokļūt augšējā labējā stūrī, ja ir atļauts veikt gājienus tikai pa labi un uz augšu?

Šo uzdevumu var atrisināt kombinatoriski. Ievērosim, ka, lai nokļūtu beigās, mums ir jāšķērso $P - 1$ horizontālas malas un $Q - 1$ vertikālas malas. No citas puses, jebkurš ceļš šādā taisnstūrī, kurā $P - 1$ reizes ejam uz augšu un $Q - 1$ reizes ejam pa labi, ir korekts ceļš. Tātad vienu korektu ceļu uzdod viena virkne, kur ir $P - 1$ simboli "↑" un $Q - 1$ simboli "→", un mums vajag saskaitīt, cik kopā ir šādu virkņu.

Lai saskaitītu šo skaitu, varam iztēloties, ka virknē no $P + Q - 2$ simboliem "↑" izvēlamies $Q - 1$, un nomainām tos uz "→". Tādu izvēļu skaits ir *binomiālais koeficients*

$$\binom{P+Q-2}{Q-1} = \frac{(P+Q-2)!}{(P-1)!\,(Q-1)!},$$

kur $k!$ apzīmē k faktoriālu. Nosaucam šo izteiksmi par $F(P, Q)$.

Dinamiskā programmēšana. Tagad aplūkosim, kā varam atrisināt oriģinālo uzdevumu taisnstūrī $R \times K$, izmantojot iepriekšējo faktu. Pievienosim vienu fiktīvu iekrāsotu rūtiņu f koordinātās (R, K), un aprēķināsim katrai iekrāsotai rūtiņai c skaitli $dp[c]$ – cik ir tādu ceļu no kreisā apakšējā stūra (koordinātās $(1,1)$) līdz rūtiņai c, kas iet tikai uz augšu un pa labi, un neiet caur nevienu citu iekrāsoto rūtiņu. Tad atbilde būs tieši $dp[f]$.

Pieņemsim, ka iekrāsotās rūtiņas mums ir $c_1 = (x_1, y_1)$, $c_2 = (x_2, y_2)$, …, $c_{N+1} = (R, K)$. Sakārtosim visas šīs rūtiņas augošā secībā pēc x (rindas), un vienādu x gadījumā sakārtosim augošā secībā pēc y (kolonnas). Aplūkosim visas šīs rūtiņas šādā secībā. Katrai no šīm rūtiņām c izrēķināsim $dp[c]$, pieņemot, ka visām iepriekšējām esam dp vērtību jau izrēķinājuši.

Aplūkojam konkrētu rūtiņu $c = (x, y)$.

- Sākumā piešķiram $dp[c] = F(x-1, y-1)$, kopējo ceļu skaitu no sākuma rūtiņas līdz c ieskaitot.
- Tagad ir jāatņem visu tādu ceļu skaits no sākuma līdz c, kas pirms tam iet caur kādu citu iekrāsotu rūtiņu. Pārlasām visas iekrāsotās rūtiņas $c' = (x', y')$, kurām $x' \le x$ un $y' \le y$. Šīs rūtiņas ir tās, kas var trāpīties ceļā no sākuma līdz c. Tad ievērojam, ka katram ceļam, kas iet no sākuma līdz c

un pirms tam apmeklē kādu citu iekrāsotu rūtiņu, mēs varam identificēt pašu pirmo apmeklēto iekrāsotu rūtiņu c'. Tāpēc katrai rūtiņai c' izrēķināsim tādu ceļu skaitu, un atņemsim no $dp[c]$. Tātad nofiksējam tādu rūtiņu c'. Visu ceļu skaits, kas iet no sākuma līdz c', un neapmeklē nevienu citu iepriekš iekrāsotu rūtiņu, ir $dp[c']$. Savukārt visu ceļu skaits, kas iet no c' līdz c, ir vienāds ar $F(x - x' + 1, y - y' + 1)$. Tāpēc kopējais ceļu skaits, kur pirmā iekrāsotā rūtiņa ir c', ir vienāds ar $dp[c'] \cdot F(x - x' + 1, y - y' + 1)$.
Attiecīgi atņemam no $dp[c]$ šādu vērtību visiem c'.

Visbeidzot atbilde oriģinālam uzdevumam tad būs atrodama $dp[f]$.

Apgrieztais elements pēc moduļa. Lai pilnībā atrisinātu uzdevumu, ir vēl nepieciešams ātri realizēt aprakstīto risinājumu. Lēna vieta šajā algoritmā ir binomiālā koeficienta $\binom{n}{k}$ rēķināšana.

Pēc definīcijas,
$\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Tā kā n un k mums var būt līdz pat $2 \cdot 10^6$, tad vienkārša šīs vērtības rēķināšana ir gan pārāk lēna, gan var neietilpt izvēlētajā skaitļu tipā. Bet izmantosim to, ka atbildi mums vajag sarēķināt pēc moduļa $p = 10^9 + 9$. Ir zināms, ka šis skaitlis ir pirmskaitlis, līdz ar to visu skaitļošanu risinājumā mēs varam veikt pēc moduļa p.

Lai efektīvi rēķinātu binomiālos koeficientus, algoritma sākumā veiksim *prekalkulāciju*. Izskaitļosim faktoriālu vērtības masīvā $f[k] = k! \pmod p$ visiem k no 1 līdz $L = R + K$, ieskaitot. Izrēķināsim to laikā $O(L)$, izmantojot formulu $k! = (k-1)! \cdot k$ (izmantojam iepriekšējo sarēķināto vērtību, lai sarēķinātu nākamo).

Lai sarēķinātu pašu faktoriāli $\binom{n}{k}$, mums ir jāatrod vērtība $\frac{f[n]}{f[k] \cdot f[n-k]} \pmod p$.

Izrādās, ka jebkuram skaitlim a pēc moduļa p (izņemot 0) eksistē tieši viens tāds skaitlis b pēc moduļa p, ka $a \cdot b = 1 \pmod p$. Šādu elementu sauc par *a apgriezto elementu* un apzīmē ar $a^{-1} \pmod p$ vai $\frac{1}{a} \pmod p$. Līdz ar to mēs pēc moduļa varam dalīt ar nenulles skaitļiem, un meklētā vērtība ir vienāda ar $f[n] \cdot f[k]^{-1} \cdot f[n-k]^{-1} \pmod p$.

Inversā elementa aprēķināšanai ir zināmi efektīvi algoritmi. Piemēram, *binārās kāpināšanas metode* vai *paplašinātais Eiklīda algoritms* ļauj mums izrēķināt apgriezto elementu pēc moduļa p laikā $O(\log p)$.

Ātrdarbība. Izpildes laika sarežģītība veidojas no vairākiem soļiem:

- Paša algoritma sākumā mēs taisām prekalkulāciju, kuras izpildes laika sarežģītība ir $O(R + K) = O(R + T)$.
- Dinamiskajā programmēšanā mēs aplūkojam $N + 1 = O(N)$ iekrāsotas rūtiņas $c = (x, y)$. Katram no šiem punktiem mēs pārlasām $O(N)$ iekrāsotas rūtiņas $c' = (x', y')$. Katram pārim c, c' mēs izrēķinam $F(x - x' + 1, y - y' + 1)$. Funkcijas F rēķināšanā mēs divas reizes rēķinām inverso elementu, tāpēc F izsaukums strādā laikā $O(\log p)$. Līdz ar to dinamiskās programmēšanas daļa aizņem $O(N^2 \log p)$.

Kopējā izpildes laika sarežģītība līdz ar to ir vienāda ar $O(R + T + N^2 \log p)$.

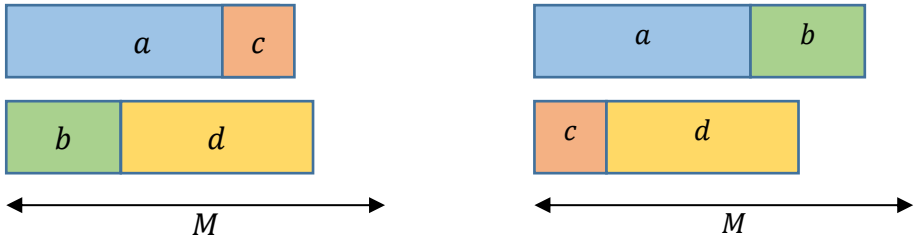
Laivotāji

Veiksim *bināro meklēšanu* pēc atbildes intervālā $[3{,}3 \cdot 10^{15}]$ (šīs robežas ir apakšējais un augšējais novērtējums lielākajam iespējamām laivas svaram). Aplūkosim, kā fiksētam atbildes kandidātam M mēs varam pārbaudīt, vai var salikt laivās cilvēkus tā, lai katras laivas svars nepārsniegtu M.

Mūsu algoritms balstīsies uz šāda apgalvojuma: pieņemsim, ka a, b ir tādi divi no svariem, ka $a + b \leq M$ un $a + b$ vērtība ir maksimālā iespējamā. Tad, ja eksistē korekts cilvēku sadalījums, tad noteikti eksistē arī tāds korekts sadalījums, kur vienā laivā sēž divi cilvēki ar svariem a un b.

Pieņemsim no pretējā, ka nav laivas no diviem cilvēkiem ar svariem a un b. Aplūkosim patvaļīgu korektu sadalījumu un aplūkosim, kur var atrasties svari a un b. Šķirosim trīs gadījumus:

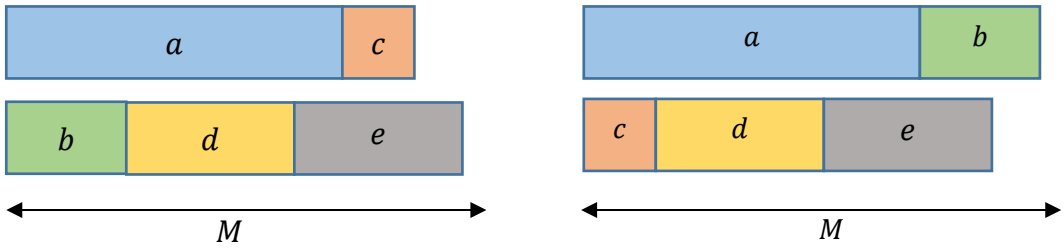
- a un b atrodas dažādās laivās no tieši diviem cilvēkiem. Pieņemsim, ka vienā laivā atrodas cilvēki ar svariem a un c, bet otrā laivā atrodas cilvēki ar svariem b un d (53. zīm.).



53. zīm.

Tad ievērojam, ka $b \geq c$, jo citādi summa $a + c$ būtu vislielākā, kas nepārsniedz M. Līdz ar to mēs varam apmainīt vietām b ar c, un saglabāt abu laivu svaru zem M. Tiešām, $a + b \leq M$ pēc dotā, un $c + d \leq b + d \leq M$. Līdz ar to esam uzbūvējuši tādu korektu sadalījumu, ka a un b ir kopā.

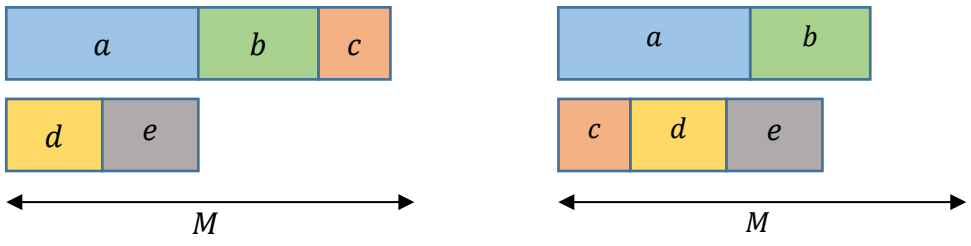
- a atrodas vienā laivā ar diviem cilvēkiem, un b atrodas laivā ar trīs cilvēkiem. Pieņemsim, ka pirmajā laivā ir cilvēki ar svariem a un c, un otrajā ir cilvēki ar svariem b, d, e (54. zīm.).



54. zīm.

Līdzīgi kā iepriekš, varam secināt, ka $c \leq b$. Līdz ar to atkal varam apmainīt vietām b un c un iegūt korektu sadalījumu ar a un b vienā laivā.

- a un b atrodas kopā laivā no trīs cilvēkiem. Pieņemsim, ka trešajam cilvēkam tajā laivā ir svars c. Paņemsim jebkuru citu laivu, pieņemsim, ka cilvēku svari tajā ir d un e (55. zīm.).



55. zīm.

Pēc a un b definīcijas ir spēkā $d + e \leq a + b$. Līdz ar to mēs varam pārvietot svaru c laivā ar d un e, iegūstot korektu sadalījumu, kur a un b paliek divi laivā.

Algoritms. Izmantosim šo apgalvojumu, lai atrisinātu uzdevumu. Algoritms būs šāds: $\frac{N-3}{2}$ reizes atrodam pāri ar svariem a, b, ka $a + b \leq M$ un $a + b$ ir lielākais. Tad izņemam abus šos svarus no kopējā masīva, un atkārtojam, līdz paliek trīs cilvēki, kurus beigās sasēdinām vienā laivā.

Parādīsim, kā implementēt a un b atrašanu $O(N)$ laikā masīvā no N elementiem. Apzīmēsim doto masīvu ar $w[1..N]$, kur visi svari ir sakārtoti augošā secībā. Izdarīsim vajadzīgo, izmantojot *divu norāžu* metodi. Uzturēsim divas norādes: kreisā (L) sākumā norāda pirms masīva pirmā elementa ($L = 0$), un labējā (R) sākumā norāda uz masīva pēdējo elementu ($R = N$). Vienkāršībai piešķiram $w[0] = 0$.

Tad bīdām L pa labi tik tālu, kamēr $L < R$ un $w[L] + w[R] \leq M$, un visu laiku saglabājam lielāko summu starp apskatītajiem pāriem. Kad vairs tādā veidā nevaram pabīdīt L, pabīdām R vienu elementu pa kreisi.

Tagad skaidrs, ka summa $w[L] + w[R]$ joprojām ir mazāka par M, jo $w[R]$ nevarēja palielināties, pabīdot R pa kreisi. Tagad atkārtojam visu algoritmu ar jauno R, un tādā veidā aplūkojam visus L kandidātus, kur $w[L] + w[R] \leq M$.

Turpinām algoritmu līdz brīdim, kad $L = R$. Ja neesam atraduši nevienu tādu pāri, ka $w[L] + w[R] \leq M$, tad binārajā meklēšanā aplūkotajai vērtībai M nav korekta sadalījuma. Ievērojam, ka katra no norādēm pabīdījās ne vairāk kā par N pozīcijām, līdz ar to ātrdarbība ir $O(N)$.

Ātrdarbība. Algoritma sākumā mums ir vienreiz jāsakārto skaitļi, kas ir $O(N \log N)$, bet pašā algoritmā veicam bināro meklēšanu pāri atbildei M, kas strādā laikā $O(\log \max w[i])$. Tad vienai fiksētai M vērtībai algoritmam ir $\frac{N-3}{2}$ iterācijas, katrā no kurām atrodam vienu svaru pāri, un izmetam no masīva. Katra iterācija aizņem $O(N)$ laiku. Līdz ar to kopējā izpildes laika sarežģītība ir $O(N \log N + \log(\max w[i]) \cdot N^2) = O(\log(\max w[i]) \cdot N^2)$.

Mona Luīze

Viegli pamanīt, ka gleznu skaitliskie novērtējumi veido orientētu grafu bez cikliem, kur grafa virsotnēs atrodas attiecīgo gleznu raksturojošs cenu diapazons, bet katra šķautne savieno divas virsotnes tad un tikai tad, ja ievaddatos dota informācija, ka viena ir lētāka par otru. Uzskatīsim, ka šķautne iziet no virsotnes, kas atbilst dārgākajai gleznai, bet ieiet virsotnē, kas atbilst lētākajai gleznai. Katru grafa virsotni papildināsim ar diviem skaitliskiem atribūtiem maz un $liel$ - attiecīgi gleznas mazāko un lielāko iespējamo cenu, izteiktu eiro.

Lai aprēķinu laikā neizmainītu šos atribūtus virsotnēm, kas atbilst gleznām, kurām cena jau ir zināma (dota ievaddatos), virsotnes ir vērts papildināt ar loģiskā tipa atribūtu $fiksēta$, kura vērtība ir "patiess", ja cena ir precīzi zināma, vai "aplams" - ja nav.

Tagad varam uzdevumu risināt divos posmos - pirmajā katrai gleznai noteikt lielāko iespējamo cenu, bet otrajā - mazāko.

Grafa caurskatīšanu sāksim no tām virsotnēm, no kurām iziet kāda šķautne, bet kurās neviena šķautne neieiet - attiecīgā gleznā nevienā salīdzināšanā nav bijusi lētāka par kādu citu. Ja šādai virsotnei atribūta $fiksēta$ vērtība ir "aplams", tad tai $liel = 1999999999$. Varam atzīmēt, ka šo virsotni jau esam apstrādājuši, un aplūkot tās virsotnes u, kurās ieiet šķautne no jau apstrādātās virsotnes v. $u_{liel} = \min(u_{liel}, v_{liel} - 1)$. Svarīgi ir uzmanīgi apstrādāt gadījumus, kad vienā virsotnē ienāk vairākas šķautnes - turpināt nākamo virsotņu apstrādi ar tām, kurās ieiet šķautnes no šīs virsotnes var tikai tad, kad apstrādātas visas šajā virsotnē ienākošās šķautnes.

Kad šādi apstrādātas visas virsotnes, visām būs zināmas $liel$ vērtības.

Pēc tam līdzīgi var atrast arī maz vērtības - jāsāk ar virsotnēm, kurās šķautnes ieiet, bet no kurām neiziet. Visām tām maz vērtība ir 1.

Atrisinājuma izpildes laika sarežģītība ir $O(G + S)$.

Kašķīgais parlaments

Risinājuma ideja ir parādīt, ka pie jebkādas N deputātu un viņu izteikumu kopas var izvēlēties vienu deputātu kā secībā pēdējo, un korekti reducēt uzdevumu uz $N - 1$ deputātiem. Tad vienmēr būs iespējams uzkonstruēt derīgu virkni, sākot ar pēdējo deputātu, tad pirmspēdējo, utt., un atbilde nekad nebūs 0 ("derīga virkne neeksistē").

Galvenais novērojums: ja apskatām visu N deputātu izteikumus kopumā, noteikti eksistē vismaz viens deputāts, par kuru ir veikti ne vairāk kā trīs izteikumi.

Tas izriet no Dirihlē principa. Pieņemsim pretējo, proti, ka par visiem deputātiem ir veikti vairāk nekā trīs izteikumi - tātad vismaz četri. Tā kā kopā ir N deputāti, tas nozīmē, ka kopā jābūt vismaz $4N$ izteikumiem. Bet, tā kā katrs deputāts veic tieši trīs izteikumus, kopā ir $3N$ izteikumi. Tātad pieņēmums ir nepareizs, no kā seko, ka eksistē kāds (vismaz viens) deputāts, par kuru ir ne vairāk kā trīs izteikumi.

Ja mēs atrodam šādu deputātu D, par kuru kopā ir ne vairāk kā trīs izteikumi, mēs varam viņu likt uzstāšanās secības beigās.

1. Tā kā par D kopā veikti ne vairāk kā trīs izteikumi, neatkarīgi no visu pārējo deputātu secības, viņš nevienā brīdī par sevi nebūs dzirdējis vairāk par trim izteikumiem.

2. Tā kā D uzstājas pēdējais, viņa veiktos izteikumus nedzirdēs neviens cits, līdz ar to mēs D izteikumus attiecībā uz pārējiem $N-1$ deputātiem varam neņemt vērā.

Tātad esam izvēlējušies secībā pēdējo deputātu D un reducējuši uzdevumu uz tādu pašu, bet jau par mazāku deputātu skaitu - atlikušajiem $N-1$ deputātiem: doti $N-1$ deputāti, kur katrs veic trīs izteikumus par citiem, un jāatrod tāda secība, kurā neviens no deputātiem par sevi nedzird vairāk kā trīs izteikumus. (Piebilde: apskatot atlikušo $N-1$ deputātu kopu, deputātu izteikumi par deputātu D netiek ņemti vērā, tāpēc patiesībā katrs no viņiem dzird "ne vairāk kā" trīs izteikumus šīs kopas ietvaros, bet varam pieņemt sliktāko gadījumu, proti, ka katrs veic tieši trīs izteikumus - gluži kā oriģinālajā uzdevumā)

Līdz ar to mēs analoģiski iepriekšējam spriedumam varam atrast pēdējo deputātu starp atlikušajiem $N-1$ deputātiem (tātad - priekšpēdējo starp visiem N) un tālāk reducēt uzdevumu uz $N-2$ deputātiem. Šādi atkārtojot mēs katrā solī garantēti atrodam pēdējo deputātu no atlikušajiem, līdz ir palicis tieši viens deputāts, kurš tad arī tiek izvēlēts kā pirmais visu N deputātu secībā.

Risinājumu tehniski var implementēt precīzi atbilstoši šiem spriedumiem:

1. apstrādājam visus N' atlikušos deputātus un izskaitām, cik izteikumi ir veikti par katru no viņiem
2. atrodam deputātu $D_{N'}$, par kuru veikti ne vairāk kā trīs izteikumi (tāds noteikti eksistē)
3. atzīmējam viņu kā pēdējo no šiem N' deputātiem
4. izmetam $D_{N'}$ no kopas un atkārtojam (1)-(4) atlikušajiem $N'-1$ deputātiem.

Šāda risinājuma izpildes laika sarežģītība ir $O(N^2)$, kas iegūst tikai daļējus punktus.

Algoritmu var paātrināt šādi:

1. glabājam katram deputātam i pret viņu veikto izteikumu skaitu $pret[i]$; sākumā tos vienkārši saskaitām, apstrādājot visus N deputātus
2. glabājam sarakstu $max3$ ar deputātiem par kuriem kopā veikti ne vairāk kā trīs izteikumi; zināms ka pašā sākumā šajā sarakstā būs vismaz viens deputāts
3. tā kā $max3$ ir vismaz viens deputāts, izvēlamies kādu no tiem un atzīmējam kā pēdējo secībā (skaidrs, ka, ja tādi ir vairāki, varam izvēlēties vienalga kuru no tiem)
4. "izmetam" šī izvēlētā deputāta izteikumus par pārējiem (jo, tā kā viņš ir pēdējais secībā, viņa izteikumi neietekmē pārējos); proti, samazinām $pret[i]$ visiem deputātiem i par kuriem izvēlētais deputāts ir izteicies
5. ja operācijas (4) rezultātā $pret[i]$ tiek samazināts līdz 3 (vai mazāk) kādam i, mēs zinām, ka starp atlikušajiem deputātiem pret deputātu D_i ir veikti ne vairāk kā trīs izteikumi; tāpēc pievienojam i sarakstam $max3$
6. atkārtojam darbības (3)-(5), kamēr esam uzkonstruējuši visu N deputātu secību; katrā solī mēs apstrādājam reducētu deputātu kopu atbilstoši teorētiskajam spriedumam, līdz ar to zināms, ka $max3$ vienmēr būs kāds deputāts ko izvēlēties kā pēdējo, un tāpēc process neapstāsies pirms visiem deputātiem nebūs piešķirta vieta secībā (atlikusī deputātu kopa būs tukša)

Šī risinājuma izpildes laika sarežģītība ir $O(N)$.

Teniss

Lasot uzdevuma formulējumu, ir vērts pievērst uzmanību teikumam *"Ņemiet vērā, ka dotais turnīru saraksts atklātā veidā neapraksta visu turnīru secību gada ietvaros, kā arī to, kurā gadā minētajā turnīrā Ernests ir piedalījies!"*

Netieši ar to ir mēģināts pateikt, ka datos **nav iekodēta** Ernesta piedalīšanās turnīros gada laikā un aprēķinātā lielākā punktu summa būs tikai vienas sezonas laikā **teorētiski sasniedzamā**.

Nav nepieciešams mēģināt noteikt precīzu turnīru kalendāru sezonas laikā, jo to nemaz nav iespējams izdarīt. Piemēram, ja Ernests katru sezonu piedalītos tikai vienā turnīrā, tad vienas sezonas ietvaros turnīru secība varētu būt **jebkura**.

Pēc uzdevuma nosacījumiem secīgo turnīru fragmentā nedrīkst būt vienādi turnīri - tas arī ir vienīgais būtiskais nosacījums. Jebkurai turnīru secībai ar šo īpašību atlikušos turnīrus var sadalīt pa gadiem tā, ka pretrunu nav. Vienkāršākais - uzskatīt, ka visi pārējie turnīri ir notikuši atsevišķos gados un līdz ar to neveido pretrunu ar atrasto fragmentu. Tātad uzdevums reducējas uz nepieciešamību aplūkot visus dažādu turnīru fragmentus un atrast starp tiem fragmentu ar lielāko iespējamo punktu summu.

Programmā mainīgajā ***summa*** saglabāsim visu turnīru punktu summu, sākot ar pirmo turnīru un neraizējoties par atkārtotajiem turnīriem. Mainīgajā ***zaudētie*** saglabāsim iepriekšējo gadu turnīros jau zaudēto punktu summu. Mainīgo ***summa*** un ***zaudētie*** starpība būs lielākā tāda fragmenta, kurš beidzas ar šo turnīru, punktu summa. Mainīgajā ***maksis*** saglabāsim šobrīd lielāko dažādos secīgos turnīros iegūto punktu kopsummu. Visu nosaukto mainīgo vērtības sākumā ir 0.

Kolekcijā ***k*** katram turnīra nosaukumam tiks saglabāts punktu skaits kāds bija līdz šim turnīram (ieskaitot to) formā *<turnīra nosaukums, punktu summa līdz šim turnīram>.*

```
cikls
  Ielasa turnīra nosaukumu t un punktu skaitu tajā p
  summa := summa + p
  ja k satur rakstu <t, summa_t>
        ja zaudētie < summa_t
              zaudētie = summa_t
        summa_t := summa
  citādi
        <t, summa> pievieno k
  ja summa – zaudētie > maksis
        maksis := summa - zaudētie
cikla beigas
```

Aplūkosim programmas darbības piemēru:

t	*p*	*summa*	*k*	*zaudētie*	*summa-zaudētie*	*maksis*
		0	{}	0	0	0
Te	12	12	<Te,12>	0	12	**12**
Tur	13	25	<Te,12>,<Tur,25>	0	25	**25**
Te	11	36	<Te,36>,<Tur,25>	12	24	25
Tur	15	51	<Te,36>,<Tur,51>	25	26	**26**
Nekur	10	61	<Te,36>,<Tur,51>,<Nekur,61>	25	36	**36**
Tur	16	77	<Te,36>,<Tur,77>,<Nekur,61>	51	26	36
Tur	18	95	<Te,36>,<Tur,95>,<Nekur,61>	77	18	36
Te	6	101	<Te,101>,<Tur,95>,<Nekur,61>	77	24	36
Nekur	13	114	<Te,101>,<Tur,95>,<Nekur,114>	77	37	**37**
Tur	4	118	<Te,101>,<Tur,118>,<Nekur,114>	95	23	37

Risinājuma izpildes laika sarežģītība ir *O(turnīru_skaits * log(dažādo_turnīru_skaits))*.

Izslēdzošais VAI

Šajā uzdevumā izrādās, ka vienmēr pietiek ar k, ne lielāku par 4. Skaitlis X ir jāizvēlas atkarībā no dotā skaitļa A atlikuma, dalot ar 8. Katrā no šiem gadījumiem eksistē viegli definējams komplekts ar secīgiem skaitļiem, kuru izslēdzošais VAI ir vienāds ar A.

Sākumā aplūkosim gadījumu, kad $A \equiv 0 \pmod 4$, tas ietver atlikumus 0 un 4, dalot ar 8. Aplūkosim skaitli A binārajā pierakstā,

$A = a_n \cdot 2^{n-1} + a_{n-1} \cdot 2^{n-2} + \cdots + a_3 \cdot 4 + a_2 \cdot 2 + a_1 \cdot 1 = \overline{a_n a_{n-1} \ldots a_3 a_2 a_1}_2$.

Tā kā A dalās ar 4, izpildās $a_2 = a_1 = 0$. Paņemsim tad kā X skaitli $A + 1$, un $k = 2$. Pierādīsim, ka $(A+1)\ xor\ (A+2)\ xor\ (A+3) = A$.

Skatīsimies atsevišķi pa bitu pozīcijām i. Ja $i > 2$, tad tajā pozīcijā būs cipars $a_i\ xor\ a_i\ xor\ a_i = a_i$, līdz ar to visām šādām pozīcijām vērtība sakrīt ar A vērtību šajās pozīcijās. Savukārt, 1. un 2. pozīcijā vērtība ir $01_2\ xor\ 10_2\ xor\ 11_2 = 00_2$.

Tāpēc arī labējās divās pozīcijās vērtības sakrīt ar A.

No šī piemēra redzam vispārīgu šādas konstrukcijas principu:

- Paņemam kā X kādu skaitli, kas ir tuvs A, un kam kreisie biti, izņemot trīs labējos, sakrīt ar skaitli A.
- Izvēlamies k pāra skaitli, lai kopā summā būtu nepāra skaits saskaitāmo (kopā ir $k+1$ saskaitāmais); tā kā nepāra reizes veicot *xor* ar vienu bitu, rezultāts ir tāds pats bits, tad gandrīz visi summas vērtības biti sakritīs ar A, izņemot dažus pēdējos.
- Izvēlāmies X un k tādu, lai labējos bitos summa sanāktu vienāda ar A atlikumu pēc 8.

Tālāk atsevišķi aplūkosim katru no pārējiem gadījumiem.

$A \equiv 1 \pmod 8$.

Šajā gadījumā labējie trīs biti ir vienādi ar 001_2. Paņemsim $X = A$ un $k = 4$. Tad summas labējie trīs biti būs vienādi ar $001_2\ xor\ 010_2\ xor\ 011_2\ xor\ 100_2\ xor\ 101_2 = 001_2$.

$A \equiv 2 \pmod 8$.

Šajā gadījumā labējie trīs biti ir vienādi ar 010_2. Paņemsim $X = A + 1$ un $k = 2$. Tad summas labējie trīs biti būs vienādi ar $011_2\ xor\ 100_2\ xor\ 101_2 = 010_2$.

$A \equiv 5 \pmod 8$.

Šajā gadījumā labējie trīs biti ir vienādi ar 101_2. Paņemsim $X = A - 3$ un $k = 2$. Tad summas labējie trīs biti būs vienādi ar $010_2\ xor\ 011_2\ xor\ 100_2 = 101_2$.

$A \equiv 6 \pmod 8$.

Šajā gadījumā labējie trīs biti ir vienādi ar 110_2. Paņemsim $X = A - 4$ un $k = 4$. Tad summas labējie trīs biti būs vienādi ar $010_2\ xor\ 011_2\ xor\ 100_2\ xor\ 101_2\ xor\ 110_2 = 110_2$.

Atliek tikai gadījumi, kad $A \equiv 3 \pmod 4$.

Šajā gadījumā labējie divi biti ir vienādi ar 11_2. Ja $A > 3$, paņemsim $X = A - 3$ un $k = 2$. Tad summas labējie divi biti būs vienādi ar $00_2\ xor\ 01_2\ xor\ 10_2 = 11_2$.

Visbeidzot, ja $A = 3$, tad varam paņemt $A = 1$ un $k = 1$, jo $01_2\ xor\ 10_2 = 11_2 = 3$.

Ātrdarbība. Uz vienu gadījumu ātrdarbība ir $O(1)$. Tā kā kopā ir N gadījumi, ātrdarbība ir $O(N)$.