ALGORITMI ŞI STRUCTURI DE DATE 4

Note de Laborator

(uz intern - draft v1.1)

Adrian Răbâea

Cuprins

Capitolul 1

Baraj 2000

1.1 Antene

Autor: prof. Dana Lica, Liceul "Ion Luca Caragiale", Ploieşti

Firmele de telefonie mobilă Dialog şi Connex urmează să-şi instaleze antene satelit într-un spaţiu comun. În acest scop, se realizează o hartă sub forma unui caroiaj dreptunghiular, unde se dau N puncte prin coordonate naturale, reprezentând punctele în care vor fi instalate antene pentru cele două firme.

Managerul Oficiului Concurenţei doreşte ca nici una din firme să nu fie dezavantajată. Astfel, ea stabileşte urmatoarea condiţie care trebuie să fie respectată:

orice dreapta paralelă cu axa Ox sau Oy trasată printr-un punct de coordonate naturale intersectează un numar de antene ale firmei Connex, număr care diferă cu cel mult 1 de numarul de antene ale firmei Dialog intersectate de aceeaşi dreaptă. Altfel spus, modulul diferenţei dintre numărul antenelor Connex şi numarul antenelor Dialog situate pe aceeaşi dreaptă este cel mult 1.

Cerinţă

Cunoscându-se coordonatele celor N puncte, să se realizeze o distribuire a tuturor punctelor între cele două firme, astfel încât restricţia impusă de managerul Oficiului Concurenţei să fie respectată.

Date de intrare:

Prima linie a fişierului de intrare ANTENE.IN conţine numărul de puncte (N).

Pe fiecare din următoarele N linii sunt scrise câte două numere x y, despărţite printr-un spaţiu, reprezentând coordonatele fiecărui punct în care se va instala o antenă.

Coordonatele punctelor sunt numere naturale cuprinse între 0 şi 30000.

Date de iesire

Fişierul de ieşire ANTENE.OUT va conţine N linii:

pe fiecare linie se vor scrie trei numere despňarţite prin câte un spatiu: x y c; primele doua numere (x, y) reprezintă coordonatele unui punct din caroiajul dat, în care este permisă amplasarea unei antene, iar ultimul numar (c) va avea valoarea 1 dacă în acel punct s-a amplasat antena firmei Dialog, respectiv -1 dacă s-a amplasat antena firmei Connex.

Ordinea de scriere a punctelor în fişierul de ieşire poate fi oarecare.

Restricţie

$N \leq 8000$

Exemplu

ANTENE.IN	ANTENE.OUT
6	1 3 1
4 1	2 3 -1
3 2	2 4 1
1 3	3 4 -1
2 3	4 1 1
2 4	3 2 1
3 4	

Timp maxim de executie: 2 secunde/test

1.1.1 Indicaţii de rezolvare - descriere soluţie

1.1.2 Rezolvare detaliată

1.1.3 Codul sursă

1.2 Solitaire

prof. Emanuela Cerchez, Liceul de Informatică, Iaşi

Un jucător solitar a inventat următorul joc. El desenează pe o foaie de hârtie n puncte.

Jocul constă în construirea unor segmente după urmatoarele reguli:

- orice segment uneşte două din cele n puncte;

- orice punct poate să fie extremitatea a cel mult un segment şi nu poate să aparţină interiorului segmentelor;

- oricare două segmente nu se intersectează;

- jocul continuă până când nu se mai poate trasa nici un astfel de segment.

Scopul jocului este de a trasa segmente astfel încât, în final, numărul de puncte rămase izolate (care nu reprezintă extremităţile unor segmente) să fie minim.

Cerinţă

Scrieţi un program care să joace acest joc.

Date de intrare

Din fişierul de intrare JOC.IN se citesc:

n - numărul de puncte;

x_1 y_1 - coordonatele carteziene ale primului punct;

x_2 y_2 - coordonatele carteziene ale celui de-al doilea punct;

...

x_n y_n - coordonatele carteziene ale punctului n.

Date de ieşire

Rezultatele se vor scrie în fişierul JOC.OUT sub forma:

P - numărul de puncte rămase izolate;

K - numărul de segmente construite;

x_{11} y_{11} x_{12} y_{12} - coordonatele carteziene ale extremităţilor primului segment;

x_{21} y_{21} x_{22} y_{22} - coordonatele extremităţilor celui de-al doilea segment;

...

x_{k1} y_{k1} x_{k2} y_{k2} - coordonatele carteziene ale extremităţilor segmentului k.

Restricţie

$n \leq L \leq 200$

bf Timp maxim de executie: 1 secundă/test

1.2.1 Indicaţii de rezolvare - descriere soluţie

1.2.2 Rezolvare detaliată

1.2.3 Codul sursă

1.3 Moşteniri duble

prof. Roxana Tâmplaru, Liceul "Ştefan Odobleja", Craiova

Ministrul de Finanţe din Simot a decis să pună impozit pe moştenirile duble.

În registrul de stare civilă (cu ajutorul căruia se va stabili câte impozite se pot încasa) apar n persoane, numerotate de la 1 la n.

O persoană care nu figurează ca moştenitor al nici unei persoane este denumită *strămoş*.

O persoană care este moştenitor direct al cuiva, dar nu are nici un moştenitor (direct sau indirect) este denumită *terminal*.

Celelalte persoane care moştenesc de la una sau mai multe persoane şi au unul sau mai mulţi moştenitori sunt denumiţi *tranzienţi*.

Numim *succesiune de moştenitori* o secvenţă de persoane $p_1, p_2, ..., p_k$, unde p_1 este strămoş, p_2, ..., p_{k-1} sunt tranzienţi, p_k este terminal, iar p_i este moştenitorul direct al lui p_{i-1}, pentru oricare $i \in \{2, ..., k\}$.

Se impozitează doar persoanele terminale, beneficiare de moşteniri duble, ştiind că o persoană plăteşte câte un impozit pentru fiecare moştenire dublă. Numim *mostenire dublă* a unui terminal două succesiuni distincte de moştenitori care au acelaşi strămoş. Se consideră că două succesiuni sunt distincte dacă există cel puţin o persoană tranzientă care apare într-o succesiune şi nu apare în cealaltă.

Dacă persoana i este moştenitorul direct sau indirect al persoanei j, nu se poate întâmpla ca persoana j să fie moştenitor direct sau indirect al persoanei i.

Cerinţă

Scrieţi un program care stabileşte numărul total de impozite care pot fi încasate.

Date de intrare

Prima linie a fişierului de intrare MOST.IN conţine numărul de persoane n.

De pe urmatoarele n linii se citesc, pentru fiecare persoană numerele de ordine ale persoanelor de la care persoana respectivă a moştenit direct (de pe linia $i+1$ se citesc numerele de ordine ale persoanelor de la care a moştenit direct persoana i). Fiecare linie se termina cu zero.

Date de ieşire

În fişierul de ieşire MOST.OUT se va scrie numărul total de impozite.

Restricţie

$N \leq L \leq 200$

Exemplul 1

MOST.IN	MOST.OUT
14 0 0 0 0 1 0 5 0 8 0 1 2 0 1 3 0 3 4 0 6 7 12 0 9 0 10 0 7 12 13 0	5

Explicatie

Sunt 5 impozite pe 5 moşteniri duble.

Primul impozit îl plăteşte persoana 11 care a moştenit de la persoanele 6 şi 7 (care au persoana 1 ca strămoş).

Al doilea impozit îl plăteşte tot persoana 11 datorită moştenirilor de la persoanele 6 şi 12 (care au persoana 1 ca strămoş).

Al treilea impozit îl va plati tot persoana 11 datorită moştenirilor primite de la persoanele 7 şi 12 (care au pe persoana 1 ca strămoş).

Al patrulea impozit îl plăteşte persoana 14 care a moştenit de la persoanele 7 şi 12 (care au persoana 1 ca strămoş).

Al cincelea impozit, îl plăteşte tot persoana 14 care moşteneşte de la persoanele 12 şi 13 (care au persoana 3 ca strămoş).

Exemplul 2

MOST.IN	MOST.OUT
14 0 0 0 0 1 0 5 0 8 0 2 0 3 0 4 0 6 7 0 9 0 10 0 12 13 0	0

Exemplul 3

MOST.IN	MOST.OUT
4 0 1 0 1 2 0 3 0	1

Explicaţie

Impozitul este plătit de persoana 4 pentru care cele două succesiuni distincte sunt: $1, 2, 3, 4$ şi $1, 3, 4$, strămoşul comun fiind persoana 1.

Timp maxim de execuţie: 2 secunde/test

1.3.1 Indicaţii de rezolvare - descriere soluţie

1.3.2 Rezolvare detaliată

1.3.3 Codul sursă

1.4 Multiplu

prof. Ovidiu Domşa, Colegiul "Horea, Cloşca şi Crişan", Alba Iulia

Se dă un număr natural n şi k cifre distincte nenule.

Cerinţă

Să se găsească cel mai mic număr natural m, multiplu al lui n, format numai cu cifrele date.

Date de intrare

Prima linie a fişierului de intrare MULTIPLU.IN conţine valorile n şi k separate printr-un spaţiu.

A doua linie conţine cele k cifre separate prin câte un spaţiu.

Date de ieşire

Dacă problema are soluţie, în fişierul MULTIPLU.OUT se va scrie numărul m. În cazul în care nu există soluţie, în fişierul MULTIPLU.OUT se va scrie mesajul NU.

Restricţie

$1 \leq n \leq 10000$, $1 \leq k \leq 9$

Exemplul 1

MULTIPLU.IN	MULTIPLU.OUT
3458 4 1 3 5 2	31122

Exemplul 2

MULTIPLU.IN	MULTIPLU.OUT
15 5 1 3 4 7 9	NU

Timp maxim de execuţie: 1 secundă/test

1.4.1 Indicaţii de rezolvare - descriere soluţie

1.4.2 Rezolvare detaliată

1.4.3 Codul sursă

1.5 Parantezări

Mihai Oltean, doctorand Universitatea Babes Bolyai, Cluj

Se consideră mulţimea şirurilor de N perechi de paranteze rotunde împerecheate corect.

Această mulţime se consideră ordonată lexicografic crescător, ştiind că '(' < ')'.

Cerinţă

Dându-se un şir de paranteze se cere să se determine numărul lui de ordine, ştiind că prima configuraţie

(((...(())...)))

are numarul de ordine 1.

Date de intrare

Prima linie a fişierului de intrare PAR.IN conţine numărul N, iar a doua linie conţine şirul de paranteze.

Date de ieşire

În fiţerul de ieşire PAR.OUT se va scrie un număr întreg. Acest număr trebuie să reprezinte numărul de ordine al parantezării date, în şirul ordonat lexicografic crescător al tuturor parantezărilor corecte formate din N perechi de paranteze.

Restricţie

$1 \leq n \leq 30$

Exemple

PAR.IN	PAR.OUT	PAR.IN	PAR.OUT
3 ((()))	1	4 ()()()()	14

PAR.IN	PAR.OUT	PAR.IN	PAR.OUT
2 ()()	2	4 ()(()())	13

Timp maxim de execuţie: 1 secundă/test

1.5.1 Indicaţii de rezolvare - descriere soluţie

1.5.2 Rezolvare detaliată

1.5.3 Codul sursă

1.6 Vânatorii

Cristian Cadar şi Marius Vlad, studenţi Universitatea Politehnica Bucureşti

Cristi şi Marius s-au hotarât într-o zi să se ducă la vânatoare. Cristi are o puşcă destul de prăpădită cu care poate vâna doar lupi. Marius în schimb, care tocmai şi-a luat bursa, are o puşcă performantă cu care poate vâna şi lupi şi mistreţi. Ajunşi în pădure, ei îşi dau seama că vânatoarea nu e treabă uşoară; ei trebuie să alerge mult după fiecare animal. Având la dispoziţie un număr de doar

T minute alocate pentru vânătoare şi dorind să vâneze în acest timp cât mai multe animale (lupi şi mistreţi) ei au calculat, pe baza unor informaţii date de pădurar, câte minute trebuie să alerge după fiecare animal în parte.

Cerinţă

Scrieţi un program care determină câte animale trebuie să vâneze Cristi şi Marius pentru ca împreună să aducă acasă un număr maxim de animale.

Date de intrare

Prima linie a fişierului de intrare VANATORI.IN conţine numarul T, reprezentând durata maximă a vânatorii.

Pe cea de-a doua linie sunt scrise două numere L şi M, reprezentând numărul de lupi respectiv de mistreţi, despărţite printr-un spaţiu.

Pe cea de-a treia linie se gasesc L numere întregi, fiecare numar t_i $(i = 1, L)$ reprezentând timpul în care poate fi vânat fiecare dintre cei L lupi.

Cea de-a patra linie contine M numere întregi, fiecare numar u_i $(i = 1, M)$ reprezentând timpul în care poate fi vânat fiecare dintre cei M mistreţi.

Date de iesire

Pe prima linie a fişierului de ieşire VANATORI.OUT se va scrie un singur număr, reprezentând numărul maxim de animale pe care le pot vâna Marius şi Cristi.

Restrictii:

$1 \leq T \leq 300$
$0 \leq L \leq 600$
$0 \leq M \leq 600$
$0 \leq t_i, u_i \leq T$

Exemplu

VANATORI.IN	VANATORI.OUT
5 4 2 2 1 1 2 5 4	5

Explicaţie

Cristi vâneaza primul, al doilea şi al patrulea lup, iar Marius al treilea lup şi al doilea mistreţ.

Timp maxim de executie: 10 secunde/test

1.6.1 Indicaţii de rezolvare - descriere soluţie

1.6.2 Rezolvare detaliată

1.6.3 Codul sursă

Capitolul 2

Baraj 2001

2.1 Deşert

Radu Ştefan, Braşov

Se dă imagine alb-negru care reprezintă fotografia, realizată din satelit, a unei zone dintr-un deşert. În această zonă se caută o construcţie secretă de formă dreptunghiulară. Imaginea porţiunii de deşert analizate a fost codificată într-o matrice având dimensiunile maxime de $N \times 255$ elemente. Imaginea construcţiei este codificată într-o matrice având $K \times 32$ elemente. Elementele celor două matrice pot fi numai caracterele '.' şi '#'.

Cerinţă

Determinaţi de câte ori se regăseşte fotografia construcţiei pe hartă.

Date de intrare

Fişier de intrare: DESERT.IN

Linia 1: N K

două numere naturale nenule, N reprezentând numărul de linii al matricei care codifică fotografia deşertului, iar K numărul de linii al matricei pătratice care codifică fotografia construcţiei;

Liniile 2 ... K+1: $c_{i,1}$ $c_{i,2}$ ... $c_{i,32}$

aceste K linii conţin în formă codificată matricea fotografiei construcţiei (32 de caractere '#' şi '.' neseparate prin spaţii);

Liniile K+2 ... K+N+1: $d_{i,1}$ $d_{i,2}$ ... $d_{i,255}$

aceste N linii conţin în formă codificată matricea fotografiei deşertului (255 de caractere '#' şi '.' neseparate prin spaţii).

Date de ieşire

Fişier de ieşire: DESERT.OUT

Linia 1: NR

număr natural, reprezentând numărul de potriviri ale matricei fotografiei construcţiei în matricea fotografiei deşertului.

Restricţii

$3 \leq N \leq 1024$

$1 < K < N$

Fotografia construcţiei nu se va roti şi nu se va oglindi.

Exemplu

DESERT.IN	DESERT.OUT
3 2 #.... (în total 31 de puncte) .... #.... (în total 31 de puncte) .... #.... (în total 254 de puncte) .... #.... (în total 254 de puncte) .... #.... (în total 254 de puncte)	2

Timp de execuţie: 1 secundă/test

2.1.1 Indicaţii de rezolvare - descriere soluţie

2.1.2 Rezolvare detaliată

2.1.3 Codul sursă

2.2 Potrivire

prof. Tudor Sorin, Bucureşti

Se consideră două numere naturale nenule N şi S.

Cerinţă

Determinaţi numerele distincte x_1, x_2, ..., x_N aparţinând mulţimii $\{1, 2, ..., N$ astfel încât

$$1x_1 + 2x_2 + ... + Nx_N = S.$$

Date de intrare

Fişier de intrare: POTRIV.IN

Linia 1: N S

două numere naturale nenule, separate printr-un spaţiu, reprezentând numărul dat, respectiv suma dată.

Date de ieşire

Fişier de ieşire: POTRIV.OUT

Linia 1: x_1 x_2 ... x_N

N numere naturale nenule, separate prin câte un spaţiu, reprezentând soluţia problemei. Dacă nu există soluţie, pe această linie se va scrie numărul 0.

Restricţii

$3 < N \leq 1000$

$0 < S \leq 333834000$

dacă există mai multe soluţii, în fişier se va scrie una singură.

Exemplu

POTRIV.IN	POTRIV.OUT
4 26	3 2 1 4

Explicaţie

$13 + 22 + 31 + 44 = 26$

Timp de execuţie: 1 secundă/test

2.2.1 Indicaţii de rezolvare - descriere soluţie

2.2.2 Rezolvare detaliată

2.2.3 Codul sursă

2.3 Tort

Mihai Stroe, Bucureşti

Gigel a primit de ziua lui un tort dreptunghiular pe care mama lui a desenat un caroiaj. În mijlocul anumitor pătrăţele sunt aşezate bezele, una într-un pătrăţel.

Gigel ar vrea să mănânce cât mai multe bezele, dar tatăl lui a inventat o şmecherie prin care speră să reducă pofta de bezele a lui Gigel.

Gigel a primit şi o mulţime de bucăţi de ciocolată. Tatăl lui îi cere să înlocuiască bezelele pe care vrea să le mănânce cu bucăţi de ciocolată, formate din două pătrăţele. Gigel trebuie să înlocuiască întotdeauna câte două bezele (vecine pe orizontală sau verticală), fără a suprapune bucăţile de ciocolată.

Cerinţă Ştiind că există suficiente bucăţi de ciocolată, ajutaţi-l pe Gigel să înlocuiască cât mai multe bezele, respectând regulile impuse de tatăl lui.

Date de intrare

Fişier de intrare: TORT.IN

Linia 1: M N

două numere naturale nenule, separate printr-un spaţiu, reprezentând dimensiunile caroiajului de pe tort;

Liniile 2...M+1: t_{i1} t_{i2} ... t_{iN}

aceste M linii conţin codificarea caroiajului de pe tort, linie după linie: valoarea 1 reprezintă o bezea, 0 înseamnă că în pătrăţelul respectiv nu există bezea; între două valori numerice există un singur spaţiu.

Date de ieşire

Fişier de ieşire: TORT.OUT

Linia 1: K

numr natural, reprezentând numărul bucăţilor de ciocolată (o bucată este formată din două pătrăţele, deci va înlocui două bezele); acest număr trebuie să fie cel mai mare posibil, respectând cerinţele problemei; o bucată de ciocolată, formată din două pătrăţele nu se poate rupe;

Liniile 2 ... K+1: X_i Y_i D_i

aceste K linii conţin câte două numere naturale nenule şi un caracter, separate prin câte un spaţiu, care descriu amplasarea unei bucăţi de ciocolată; X_i şi Y_i sunt coordonatele pătrăţelului stânga sus, iar D_i identifică poziţia celuilalt pătrăţel al bucăţii astfel: este 'E'(Est) sau 'S'(Sud), după cum al doilea pătrăţel se află la dreapta primului sau sub primul.

Restricţii

$1 < M, N \leq 100$

o bucată de ciocolată trebuie să înlocuiască exact două bezele

bucăţile de ciocolată nu se pot suprapune

ordinea în fişierul de ieşire a descrierii bucăţilor de ciocolată poate fi oarecare.

Exemplu

TORT.IN	TORT.OUT
4 3	3
1 0 1	2 2 S
0 1 0	3 1 S
1 1 0	4 2 E
1 1 1	

Timp de execuţie: 1 secundă/test

2.3.1 Indicaţii de rezolvare - descriere soluţie

2.3.2 Rezolvare detaliată

2.3.3 Codul sursă

2.4 Descompuneri

şef lucrări Stelian Ciurea, Sibiu

Se dau numerele naturale nenule N, K şi X. Numim o K-*descompunere* a lui N un şir strict crescător de K numere naturale nenule, a căror sumă este N.

Cerinţă

Dintre toate K-*descompunerile* lui N, să se determine în câte apare numărul X.

Date de intrare

Fişier de intrare: DESCOMP.IN

Linia 1: N K X

trei numere naturale nenule, separate prin câte un spaţiu, având semnificaţia din enunţ.

Date de ieşire

Fişier de ieşire: DESCOMP.OUT

Linia 1: NR

număr natural, reprezentând numărul descompunerilor respectând cerinţele problemei.

Restricţii

$2 \leq N \leq 350$

Exemplu

DESCOMP.IN	DESCOMP.OUT
12 3 2	3

Explicaţii

$12 = 1 + \mathbf{2} + 9$

$12 = \mathbf{2} + 3 + 7$

$12 = \mathbf{2} + 4 + 6$

Timp de execuţie: 1 secundă/test

2.4.1 Indicaţii de rezolvare - descriere soluţie

2.4.2 Rezolvare detaliată

2.4.3 Codul sursă

2.5 James Bond

Autor: Mihai Stroe, Bucureşti

Un grup de terorişti s-a ascuns într-un sistem de canalizare. James Bond, având la dispoziţie o hartă care descrie configuraţia sistemului de canalizare şi poziţiile în care teroriştii au amplasat bombe, vrea să anihileze grupul de terorişti. Conductele din acest sistem sunt desenate pe hartă sub forma unor segmente de dreaptă în plan, astfel încât oricare două segmente au cel mult un punct comun, care este capăt pentru amândouă segmentele.

Explozia unei bombe afectează o zonă circulară (interior şi frontieră) şi este instantanee pe întreaga zonă afectată. Bombele pot să difere una de cealaltă prin raza de acţiune sau prin timpul la care explodează.

În urma unei explozii, canalele care trec prin zona de acţiune a bombei devin impracticabile pe segmentul afectat, dar, dacă Bond se află într-un astfel de canal şi a trecut deja de zona afectată de explozie, poate să-şi continue drumul.

Dacă James Bond se află în raza de acţiune a unei bombe la momentul exploziei, el moare.

La momentul iniţial (momentul 0), James Bond porneşte dintr-un punct al sistemului de canalizare (capăt al unui segment) şi se deplaseaz cu viteză constantă, egală cu 1. James Bond trebuie să ajungă în punctul în care se află teroriştii (capăt al unui segment) în cel mai scurt timp posibil.

Cerinţă

Determinaţi calea pe care trebuie să o urmeze James Bond astfel încât să ajungă viu la teroriştii în cel mai scurt timp.

Date de intrare

Fişier de intrare: BOND.IN

Linia 1: M N

• două numere întregi, pozitive, separate printr-un spaţiu, reprezentând numărul de conducte şi numărul de bombe

Liniile 2 ... M+1: X_1 Y_1 X_2 Y_2

• aceste M linii conţin fiecare câte patru numere întregi, separate prin câte un spaţiu, reprezentând descrierea segmentelor prin coordonatele extremităţilor;

Liniile M+2 ... M+N+1: X_C Y_C R T

• aceste N linii conţin fiecare câte patru numere întregi, separate prin câte un spaţiu, reprezentând descrierea bombelor prin coordonatele punctului unde sunt amplasate, raza cercului pe care acţionează şi momentul de timp la care acţionează;

Linia M+N+2: X_P Y_P

• două numere întregi, separate printr-un spaţiu, reprezentând coordonatele punctului de plecare, capăt al unuisegment;

Linia M+N+3: X_D Y_D

• două numere întregi, separate printr-un spaţiu, reprezentând coordonatele punctului în care se află teroriştii, capăt al unui segment.

Date de ieşire

Fişier de ieşire: BOND.OUT

Linia 1: $TMIN$

• număr real pozitiv, reprezentând timpul minim în care James Bond ajunge la terorişti;

Liniile 2 ...: C_1 C_2

• pe următoarele linii, până la sfârşitul fişierului, se vor scrie câte două numere întregi, separate printr-un spaţiu, reprezentând coordonatele capetelor de segmente prin care va trece James Bond pentru a ajunge la grupul de terorişti.

Restricţii

• $1 \leq M \leq 100$

• $0 \leq N \leq 50$

• $1 \leq R \leq 30000$

• $0 \leq T \leq 30000$

• coordonatele capetelor segmentelor şi ale poziţiilor bombelor aparţin intervalului $[-30000, 30000]$.

Exemplu

BOND.IN	BOND.OUT
3 1	20
0 0 10 0	0 0
0 0 10 10	10 0
10 0 10 10	10 10
6 4 2 1	
0 0	
10 10	

Observaţii

• numerele din fişierul de ieşire se vor afişa cu o precizie de două zecimale;

• se garantează existenţa soluţiei;

• datele de intrare sunt corecte.

Timp de execuţie: 1 secundă/test

2.5.1 Indicaţii de rezolvare - descriere soluţie

2.5.2 Rezolvare detaliată

2.5.3 Codul sursă

2.6 Şiruri

Autor: Radu Ştefan, Braşov

Se dau două şiruri având ambele câte N numere naturale, cuprinse între 1 şi $N/2$ (N este par). Fiecare număr între 1 şi $N/2$ apare de exact două ori în fiecare dintre cele două şiruri.

Cerinţă

Determinaţi subşirul comun având lungimea maximă.

Date de intrare

Fişier de intrare: SIRURI.IN

Linia 1: N

• număr natural nenul având semnificaţia din enunţ;

Linia 2: $a_1\ a_2\ a_3\ ...\ a_N$

• N numere naturale nenule, separate prin câte un spaţiu, reprezentând elementele primului şir;

Linia 3: $b_1\ b_2\ b_3\ ...\ b_N$

• N numere naturale nenule, separate prin câte un spaţiu, reprezentând elementele celui de-al doilea şir.

Date de ieşire

Fişier de ieşire: SIRURI.OUT

Linia 1: NR

• numărul de elemente ale subşirului comun;

Linia 2: $c_1\ c_2\ c_3\ ...\ c_{NR}$

• NR numere naturale nenule, reprezentând subşirul comun de dimensiune maximă.

Restricţii

• $0 < N \leq 15000$

• N este număr par

Exemplu

SIRURI.IN	SIRURI.OUT
10 3 5 4 4 2 5 2 3 1 1 1 4 2 2 3 4 5 5 3 1	5 4 2 2 3 1

Timp de execuţie: 1 secundă/test

2.6.1 Indicaţii de rezolvare - descriere soluţie

2.6.2 Rezolvare detaliată

2.6.3 Codul sursă

Capitolul 3

Baraj 2002

3.1 EQS

Autor: Mihai Pătraşcu, Craiova

Se consideră ecuaţii de forma:

$$a_1 \cdot x^3 + a_2 \cdot y^3 + a_3 \cdot z^3 + a_4 \cdot u^3 + a_5 \cdot v^3 = 0.$$

Se consideră soluţie un pentuplu (x, y, z, u, v) care verifică ecuaţia. Să se determine numărul de soluţii ale ecuaţiei, pentru care valorile necunoscutelor sunt numere întregi nenule din intervalul $[-50, 50]$.

Date de intrare

Fişierul de intrare EQS.IN conţine o singură linie pe care se află cei cinci coeficienţi ai ecuaţiei, separaţi prin câte un spaţiu.

Date de ieşire

Fişierul de ieşire EQS.OUT va conţine o singură linie pe care se va afla numărul soluţiilor ecuaţiei.

Precizare

• valorile coeficienţilor, precum şi cele ale necunoscutelor, sunt numere întregi cuprinse în intervalul $[-50, 50]$.

Exemplu

EQS.IN	EQS.OUT
37 29 41 43 47	654

Timp de execuţie: 2 secunde/test

3.1.1 Indicaţii de rezolvare - descriere soluţie

3.1.2 Rezolvare detaliată

3.1.3 Codul sursă

3.2 Labirint

Autor: Mihai Stroie, Bucureşti

Romeo şi *Julieta* sunt prinşi într-un labirint, descris printr-o matrice cu M linii şi N coloane, având elemente din mulţimea $\{0, 1\}$. Un element cu valoarea 1 reprezintă un zid, în timp ce un element cu valoarea 0 reprezintă un spaţiu liber. *Romeo* se află iniţial în colţul din stânga-sus al labirintului $(1, 1)$, iar *Julieta* în colţul din dreapta-jos (M, N).

În poziţiile iniţiale ale celor doi nu pot exista ziduri, iar ei se pot deplasa doar în spaţiile libere, fără a părăsi amtricea.

Pentru a evada din labirint, *Romeo* trebuie să ajungă în poziţia (i_1, j_1), iar *Julieta* în poziţia (i_2, j_2).

Dumneavoastră dispuneţi de un şir de mutări pe care cei doi le pot efectua. Şirul este format din K mutări, codificate prin literele N, E, S şi V, reprezentând deplasări de un pătrăţel spre $Nord$, Est, Sud, respectiv $Vest$. *Romeo* şi *Julieta* trebuie să efectueze *toate* mutările din acest şir, *în ordinea dată*. Dumneavoastră sunteţi cel care decide dacă o anumită mutare va fi efectuată de *Romeo* sau de *Julieta*.

Cerinţă

Scrieţi un program care va decide care dintre mutări vor fi efectuate de *Romeo* şi care de Julieta pentru a-i ajuta pe cei doi să ajungă la destinaţii.

Date de intrare

Fişierul de intrare LABIRINT.IN are următoarea structură:

• pe prima linie numerele M şi N;

• pe următoarele M linii descrierea labirintului; ficare linie conţine N valori, separate prin câte un spaţiu;

• pe următoarea linie se află patru numere întregi, separate prin câte un spaţiu, care reprezintă valorile i_1 j_1 i_2 şi j_2;

• pe următoarea linie numărul K de mutări care trebuie efectuate;

• pe ultima linie se află un şir de K litere care descriu direcţiile în care trebuie efectuate mutările.

Date de ieşire

În fişierul de ieşire LABIRINT.OUT se va scrie o singură linie care conţine K caractere. Al i-lea caracter este R dacă *Romeo* va efectua a i-a mutare sau J dacă mutarea va fi efectuată de *Julieta*.

Restricţii şi precizări

• $3 \leq M \leq 20$;

• $3 \leq N \leq 60$;

• $1 \leq K \leq 200$.

• există întotdeauna cel puţin o soluţie; în cazul în care există mai multe, se va scrie doar una dintre ele;

• prin deplasare la $Nord$ se înţelege deplasarea pe linia precedentă, o deplasare la Sud înseamnă deplasare pe linia următoare, deplasare la $Vest$ înseamnă deplasare pe coloana precedentă, iar deplasarea la Est înseamnă deplasarea pe coloana următoare.

Exemplu

LABIRINT.IN	LABIRINT.OUT
5 5 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 2 2 4 4 6 SNEEVV	RJRRJR

Timp maxim de executare: 1 secundă/test

3.2.1 Indicaţii de rezolvare - descriere soluţie

3.2.2 Rezolvare detaliată

3.2.3 Codul sursă

3.3 ATP

Autor: şef lucrări Stelian Ciurea, Sibiu

Se ştie că jucătorii profesionişti de tenis sunt clasaţi, în funcţie de rezultatele obţinute, într-un clasament al Asociaţiei Tenismanilor Profesionişti (ATP). Studiindu-se rezultatele ultimilor ani, s-a constatat că într-o partidă în care se întâlnesc doi jucători, dacă diferenţa de locuri în clasament între cei doi e mai mare decât o valoare k, atunci câştigă în mod sigur cel mai bine clasat; în schimb dacă diferenţa de poziţii în clasament este de cel mult k poziţii atunci, teoretic, poate să câştige oricare dintre ei.

La un concurs care urmează să se desfăşoare în curând după sistemul eliminatoriu, şi-au anunţat participarea primii n jucători din clasament (unde n este o putere a lui 2).

Cerinţă Să se determine care este cel mai slab jucător care ar putea, teoretic, să câştige concursul, precum şi schema de desfăşurare a partidelor din turneu şi câştigătorul fiecărei partide, pentru ca jucătorul respectiv să câştige turneul.

Date de intrare

Fişierul de intrare ATP.IN conţine o singură linie pe care se află valorile n şi k, separate prin spaţii.

Date de ieşire

Fişierul de ieşire ATP.OUT va avea următoarea structură:

• pe prima linie, poziţia în clasament a celui mai slab clasat jucător care, teoretic, ar putea câştiga concursul;

• pe a doua linie, întâlnirile din primul tur al turneului; o întâlnire va fi descrisă prin două numere, corespunzătoare poziţiilor din clasament ale jucătorilor care o susţin, iar primul jucător dintr-o pereche este câştigătorul; aşadar pe prima linie vor fi $n/2$ perechi de numere separate prin minim un spaţiu;

• pe a treia linie, întlnirile din al doilea tur, descrise la fel ca cele din prima linie; pe această linie vor fi $n/4$ perechi;

• ...

• pe ultima linie partida din finală.

Restricţii

• $2 \leq n \leq 4.96$

• n este o putere a lui 2

Exemplu

ATP.IN	ATP.OUT
16 3	11 3 1 5 2 6 4 7 16 8 13 9 14 10 15 11 12 5 3 8 6 9 7 11 10 8 5 11 9 11 8

Timp maxim de executare: 1 secundă/test

3.3.1 Indicaţii de rezolvare - descriere soluţie

3.3.2 Rezolvare detaliată

3.3.3 Codul sursă

3.4 Gard

Autor: Mugurel-Ionuţ Andreica, Bucureşti

O echipă de K muncitori a fost angajată să vopsească un gard format din N scânduri numerotate de la 1 la N, de la stânga spre dreapta. Fiecare muncitor i ($1 \leq i \leq K$) se aşează în faţa scândurii S_i şi poate vopsi numai un interval compact (numerele de ordine ale scândurilor din interval sunt consecutive) având maxim L_i scânduri, interval care trebuie să conţină scândura S_i. Pentru fiecare scândură vopsită, acesta este plătit cu suma P_i. Din motive de eficienţă, oricare 2 muncitori din echipă trebuie să vopsească intervale de scânduri disjuncte.

Fiind conducătorul echipei de muncitori, dumneavoastră doriţi să determinaţi, pentru fiecare membru al echipei, intervalul de scânduri pe care acesta va trebui să îl vopsească, astfel încât câştigul total să fie maxim. Câştigul total este egal cu suma câştigurilor realizate de fiecare membru al echipei. Câştigul realizat de fiecare muncitor este egal produsul dintre numărul de scânduri vopsite de acesta şi suma P_i corespunzătoare muncitorului.

Cerinţă Scrieţi un program care determină câştigul maxim obţinut de cei K muncitori.

Date de intrare

• Pe prima linie a fişierului de intrare GARD.IN se află numărul N al scândurilor şi numărul K al muncitorilor, separate printr-un spaţiu.

• Fiecare dintre următoarele K linii va conţine trei numere l, p şi s cu semnificaţia: muncitorul corespunzător liniei poate vopsi cel mult l scânduri, venitul primit pentru fiecare scândură vopsită este p şi, iniţial, el se află în faţa scândurii s.

Date de ieşire

În fişierul de ieşire GARD.OUT se va scrie câştigul maxim care poate fi obţinut de echipa de muncitori.

Restricţii şi?i precizări

• $1 \leq N \leq 16000$
• $1 \leq K \leq 100$
• $1 \leq P_i \leq 10000$
• $1 \leq L_i, S_i \leq N$
• iniţial, doi muncitori nu se pot afla ñ faţa aceleiaşi scânduri;
• nu trebuie vopsite neapărat toate cele N scânduri ale gardului;
• este posibil ca unul sau mai mulţi muncitori să nu vopsească nici o scândură, caz în care scândura în faţa căreia s-au aşezat iniţial poate fi vopsită, eventual, de către alt muncitor;

Exemplu

GARD.IN	GARD.OUT
8 4 3 2 2 3 2 3 3 3 5 1 1 7 17	17

Explicaţie

Muncitorul 1 vopseşte intervalul de scânduri $[1, 2]$; muncitorul 2 vopseşte intervalul de scânduri $[3, 4]$; muncitorul 3 vopseşte intervalul de scânduri $[5, 7]$; muncitorul 4 nu vopseşte nici o scândură.

Timp maxim de execuţie: 1 secundă/test

3.4.1 Indicaţii de rezolvare - descriere soluţie

3.4.2 Rezolvare detaliată

3.4.3 Codul sursă

3.5 Monede

Autor: asist. univ. Iuliu Vasilescu, Bucureşti

Se consideră N monede identice ca formă şi culoare, numerotate de la 0 la $N-1$. Printre ele există exact o monedă falsă. Toate monedele adevărate au aceeaşi greutate, cea falsă fiind mai grea sau mai uşoară decât celelalte. Pentru găsirea monedei false se foloseşte o balanţă cu două talere identice, pe care încap oricâte monede. O cântărire folosind această balanţă constă în a pune pe cele două talere ale balanţei câte un număr egal de monede. În urma cântăririi se determină dacă greutăţile de pe talere sunt egale sau talerul care conţine o greutate mai mare.

Cerinţă Scrieţi un program care să comande cântăririle care trebuie efectuate astfel încât să se determine moneda falsă şi dacă este mai grea sau mai uşoară decât cele adevărate, în număr minim de cântăriri. Pentru a afla numărul de monede şi rezultatul cântăririlor programul va trebui să folosească un modul extern.

Instrucţiuni pentru programatorii în C/C++

Programatorii în C/C++ au la dispozitie *header*-ul **mon.h**. În acest fişier sunt declarate următoarele tipuri şi funcţii:

```
typedef char taler[1000];
int init();
int cantarire(taler stanga, taler dreapta);
void rezultat(int moneda, int tip);
```

Instrucţiuni pentru programatorii în Pascal

Programatorii în Pascal au la dispozitie *unit*-ul **mon**. În acest *unit* sunt declarate următoarele tipuri, funcţii şi proceduri:

```
type taler=array[0..999] of byte;
function init: integer;
function cantarire(var stanga, dreapta: taler): integer;
procedure rezultat(moneda, tip: integer);
```

Instrucţiuni pentru utilizarea modulului

Tipul `taler` este folosit pentru a reprezenta monedele de pe un taler al balanţei. Un vector de tipul `taler` va avea elemente de mulţimea $\{0, 1\}$. Un vector de acest tip conţine pe o poziţie valoarea 1 dacă şi numai dacă moneda cu numărul corespunzător se află n balanţă pe talerul reprezentat prin vector şi 0 altfel.

Prima funcţie care trebuie apelată este **init**, funcţie care returnează numărul de monede.

În continuare va fi apelată, de câte ori este nevoie, funcţia `cantarire`, având ca parametri configuraţiile celor două talere. Funcţia returnează 0 dacă talerele sunt echilibrate, -1 dacă talerul din stânga este mai uşor, respectiv 1, dacă talerul din stânga este mai greu.

Procedura/funcţia `rezultat` va fi apelată la sfârşit pentru a anunţa moneda falsă. Primul parametru este numărul monedei false, iar al doilea trebuie să fie -1 dacă moneda falsă este mai uşoară decât celelalte, respectiv 1 dacă moneda falsă este mai grea decât celelalte.

Nu vor fi realizate operaţii de intrare/ieşire cu fişiere.

Exemplu de succesiune de apeluri

```
init();    întoarce valoarea 5, sau în Pascal   init;
cantarire([1,1,0,0,0,0,...], [0,0,1,1,0,0,...]);   întoarce 0
cantarire([0,0,0,0,1,0,...], [1,0,0,0,0,0,...]);   întoarce −1
rezultat(4, -1);
```

Restricţii

• $3 \leq N \leq 1000$

Timp de executare: 1 secundă pe test.

3.5.1 Indicaţii de rezolvare - descriere soluţie

3.5.2 Rezolvare detaliată

3.5.3 Codul sursă

3.6 Banana

Autori: prof. Roxana Tâmplaru şi Mihai Pătraşcu, Craiova

Se consideră o pădure tropicală, reprezentată sub forma unui caroiaj dreptunghiular. Celula din colţul stânga sus al caroiajului are coordonatele $(1, 1)$, iar coordonatele celorlalte celule sunt determinate de linia şi coloana pe care se află. În anumite celule ale caroiajului sunt plasaţi bananieri; o celulă conţine cel mult un bananier. Mai mulţi bananieri care se învecinează pe orizontală sau verticală formează o zonă de bananieri. Într-o astfel de zonă, *Cekili* se deplasează uşor, cu agilitatea-i cunoscută, de la un bananier la altul.

Maimuţa *Cekili* este lacomă şi nu îi ajung bananele dintr-o singură zonă. *Tarzan* vrea să-şi ajute prietena. Pentru aceasta, el ar putea conecta exact K zone de bananieri înnodând mai multe liane şi astfel *Cekili* s-ar putea deplasa de la o zonă la alta utilizând lianele.

Evident, *Tarzan* trebuie să aleagă zonele astfel încât numărul total de bananieri din cele K zone să fie maxim.

Cerinţă Determinaţi numărul maxim de bananieri care se poate obţine prin conectarea a exact K zone.

Date de intrare

Pe prima linie a fişierului de intrare BANANA.IN se află numărul Nr al bananierilor şi numărul K al zonelor care pot fi conectate, despărţite printr-un

spaţiu. Pe fiecare dintre următoarele Nr linii se află câte două numere naturale, despărţite printr-un spaţiu, care reprezintă coordonatele unuia dintre bananieri.

Date de ieşire

Fişierul de ieşire BANANA.OUT va conţine o singurălinie pe care se va afla numărul maxim de bananieri care se poate obţine prin conectarea zonelor.

Restricţii şi precizări

• $1 \leq Nr \leq 16000$

• coordonatele bananierilor sunt numere întregi cuprinse între 1 şi 10000;

• numărul de zone este cel puţin egal cu K;

• două poziţii se învecinează pe orizontală dacă sunt pe aceeaşi linie şi pe coloane consecutive, respectiv pe verticală dacă sunt pe aceeaşi coloană şi pe linii consecutive.

Exemplu

BANANA.IN	BANANA.OUT
10 3 7 10 1 1 101 1 2 2 102 1 7 11 200 202 2 1 3 2 103 1	9

Timp maxim de executare: 1 secundă/test

3.6.1 Indicaţii de rezolvare - descriere soluţie

3.6.2 Rezolvare detaliată

3.6.3 Codul sursă

Capitolul 4

Baraj 2003

4.1 Excursie

În una dintre zile, la Olimpiada de Informatică se organizează P excursii atractive. La aceste excursii participă în total N persoane. Pentru simplitate, persoanele au fost numerotate de la 1 la N, primele K persoane fiind ghizii. O persoană se poate înscrie la exact una dintre cele P excursii organizate.

Pentru a evita surprizele neplăcute (insuficiente mijloace de transport, insuficiente locuri la restaurant, etc) organizatorii intenţionează să studieze toate configuraţiile ce pot să apară în urma înscrierilor participanţilor, considerând totuţ că în fiecare excursie va exista cel puţin un participant.

Cerinţă

Scrieţi un program care să determine numărul de configuraţii distincte ce se pot obţine după înscrierea celor N persoane la cele P excursii organizate, astfel încât cei K ghizi să fie înscrişi în excursii diferite.

Date de intrare

Fişierul de intrare se numeşte **ex.in** şi conţine o singură linie pe care se află

3 numere naturale separate prin câte un spaţiu: N K P (reprezentând numărul de persoane, numărul de ghizi şi respectiv numărul de excursii).

Date de ieşire

Fişierul de ieşire **ex.out** conţine o singură linie pe care se află numărul de configuraţii distincte.

Restricţii

• $1 \leq K \leq P \leq N \leq 100$

• Într-o configuraţie nu contează ordinea excursiilor sau ordinea în care se înscriu persoanele la o excursie.

Exemplu

ex.in	**ex.out**	Explicaţie
5 3 4	7	Cele 7 configuraţii distincte sunt: (1,4)(2)(3)(5) (1)(2,4)(3)(5) (1)(2)(3,4)(5) (1)(2)(3)(4,5) (1,5)(2)(3)(4) (1)(2,5)(3)(4) (1)(2)(3,5)(4)

Timp maxim de execuţie: 0.1 secunde/test sub Linux şi 0.5 secunde sub Windows.

4.1.1 Indicaţii de rezolvare - descriere soluţie

4.1.2 Rezolvare detaliată

4.1.3 Codul sursă

4.2 Robot

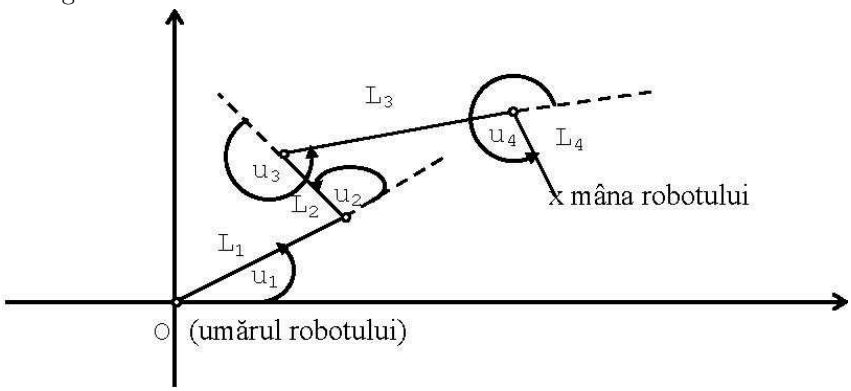
Lucraţi la o firmă care produce microprocesoare. Pentru asamblarea microprocesoarelor, firma utilizează un robot constituit dintr-un singur braţ. Braţul este fixat la unul dintre capete ("umărul") într-un punct plasat în centrul platformei de lucru, iar la celălalt capăt are un dispozitiv de lungime neglijabilă cu care poate "culege" componentele de pe platforma de lucru ("mâna"). Braţul se poate mişca numai în plan orizontal, deasupra platformei de lucru.

Braţul este constituit dintr-o succesiune de N segmente rigide de lungimi L_1, L_2, ..., L_N, conectate prin puncte de articulaţie. Mai exact, segmentul 1, este conectat printr-un punct de articulaţie în umărul robotului, segmentul 2 este conectat printr-un punct de articulaţie de segmentul 1, ..., segmentul N este conectat printr-un punct de articulaţie de segmentul $N-1$ şi are la celălalt capăt "mâna". Un punct de articulaţie permite rotaţia liberă (la orice unghi) a segmentului conectat în acel punct de articulaţie.

Pentru asamblarea unui microprocesor robotul trebuie să culeagă succesiv componentele acestuia de pe platforma de lucru. Fiecare componentă are o poziţie bine determinată pe platforma de lucru, prin coordonatele sale relativ la un sistem de coordonate cartezian, cu centrul în umărul robotului.

Rolul dvs. în firmă este de a programa mişcările robotului. În acest scop, pentru fiecare componentă pe care robotul o va culege trebuie să specificaţi "configuraţia" braţului robotului care să permită atingerea componentei respective (mâna robotului să fie plasată deasupra poziţiei în care se află componenta).

Configuraţia braţului robotului este definită de unghiurile dintre segmentele braţului rigid.



Cerinţă

Scrieţi un program care, pentru o poziţie dată, determină o configuraţie pentru braţul robotului care să-i permită acestuia să culeagă componenta din poziţia respectivă, dacă este posibil.

Date de intrare

Fişierul de intrare **robot.in** conţine pe prima linie un număr natural N, care reprezintă numărul de segmente din care este format braţul robotului.

Pe fiecare dintre următoarele N linii se află câte un număr natural. Numărul aflat pe linia $i+1$ este lungimea celui de-al i-lea segment al braţului robotului.

Pe ultima linie se află două numere întregi x şi y, separate prin câte un spaţiu, reprezentând coordonatele poziţiei la care trebuie să ajungă "mâna" robotului.

Date de ieşire

Fişierul de ieşire **robot.out** conţine o singură linie pe care se află valoarea 0

dacă nu este posibil ca mâna robotului să ajungă în poziţia x, y. Dacă problema are soluţie, fişierul de ieşire conţine N linii. Pe linia i se află valoarea reală u_i care reprezintă unghiul dintre segmentul i şi segmentul $i-1$ (pentru orice i de la 2 la N), iar valoarea u_1 reprezintă unghiul pe care segmentul 1 îl formează în umărul robotului cu axa OX.

Restricţii şi precizări

- $2 \leq N \leq 10000$
- $1 \leq L_i \leq 200$
- $0 \leq u_i < 360$
- $-100000 \leq x, y \leq 100000$
- Unghiurile se măsoară în sens trigonometric şi sunt exprimate în grade.
- Programul de evaluare va verifica dacă punctul în care este plasată mâna robotului pentru configuraţia dată de dvs. (x_p, y_p) coincide cu punctul de coordonate (x, y). Eroarea admisă este de 0.001. Mai exact, $\max\{|x-xp|, |y-yp|\} < 0.001$
- Orice linie se termină cu un marcaj de sfârşit de linie (Enter).

Exemple

robot.in	robot.out	robot.in	robot.out
3 10 5 25 15 20	125.6725 0 252.5424	3 10 5 25 2 4	0

Timp maxim de execuţie/test: 0.1 secunde pentru Linux şi 0.3 secunde pentru Windows.

4.2.1 Indicaţii de rezolvare - descriere soluţie

4.2.2 Rezolvare detaliată

4.2.3 Codul sursă

4.3 Telegraf

Până nu demult, comunicaţia la distanţă se făcea cu ajutorul telegrafului. Folosind telegraful, se pot transmite două tipuri de semnale: punct şi linie. În general, dorim să transmitem texte formate din litere ale alfabetului latin şi cifre (în total, 36 de

simboluri). Trebuie deci să folosim o codificare, adică să asociem fiecăruia din cele 36 de simboluri o succesiune distinctă de linii şi puncte. Pentru a putea decodifica o succesiune recepţionată de linii şi puncte, este necesar ca nici un simbol să nu aibă o codificare identică cu începutul codificării pentru un alt simbol. Să considerăm câteva exemple (presupunând că nu vrem să transmitem decât literele A, B, C):

Exemplul 1	Exemplul 2	Exemplul 3
A = ..	A = .–	A = .-..
B = .-	B = .-	B = -.
C = -	C = -	C = .-.

Exemplul 1 reprezintă o codificare corectă. Exemplul 2 reprezintă o codificare greşită, pentru că începutul codificării pentru A este identic cu codificarea pentru B (deci, o secvenţă de genul .– este ambiguă, putând însemna şi A şi BC). Exemplul 3 este de asemenea o codificare greşită pentru că începutul codificării pentru A este identic cu codificarea pentru C (o secvenţă precum .-..-. este ambiguă, putând însemna fie AB, fie CC).

Se ştie că într-o transmisie telegrafică, punctul durează o secundă, iar linia 2 secunde. Putem calcula astfel timpul necesar transmiterii unui text.

Folosind codificarea din exemplul 1, transmiterea textului CABCA = - .. .- - .. durează 11 secunde. Observaţi că lungimea transmisiei se poate calcula şi astfel: 2(A) + 1(B) + 2(C) = 2(..) + 1(.-) + 2(-) = 2*2 + 1*3 + 2*2 = 11.

Cerinţă

Se consideră un text, dat prin frecvenţa apariţiei fiecărui simbol (dintre cele 36 considerate). Să se găsească durata minimă necesară transmiterii acelui text, folosind o codificare aleasă corespunzător.

Date de intrare

Fişierul **telegraf.in** conţine o singură linie cu 36 de numere întregi nenegative, separate prin câte un spaţiu, reprezentând numărul de apariţii ale fiecărui simbol în textul ce trebuie transmis.

Date de ieşire

Fişierul **telegraf.out** va conţine un singur număr, c si anume lungimea minimă (în secunde) necesară pentru transmiterea textului.

Restricţii şi precizări

• nici unul din cele 36 de simboluri nu apare de mai mult de 1000000 de ori în textul considerat

• există cel puţin două simboluri cu număr de apariţii nenul

Exemplu

telegraf.in
2 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

telegraf.out
11

Se constată că este optim să se transmită acest text folosind codificarea din Exemplul 1, obţinnd o lungime minimă a transmisiei de 11 secunde.

Timp maxim de execuţie: 0.1 secunde pe test pentru Linux şi 0.3 secunde pentru Windows

Notă: 40% dintre teste vor conţine maxim 16 simboluri cu frecvenţă de apariţie nenulă.

4.3.1 Indicaţii de rezolvare - descriere soluţie

4.3.2 Rezolvare detaliată

4.3.3 Codul sursă

4.4 Ajutor

Dacă te-ai rănit, nu folosi Sprite să tratezi rana. Cel mai bine e să fugi la cel mai apropiat post de prim ajutor. Norocul tău este că ai o hartă din care poţi afla coordonatele carteziene ale posturilor de prim ajutor. Ghinionul este că, din cauza durerii, nu poţi să fugi direct spre un post, ci numai pe direcţiile nord-sud şi est-vest. Ca să ştii dacă mai are sens să fugi spre un post sau să te bucuri de ultimele clipe de viaţă, cel mai bine e să evaluezi distanţa până la cel mai apropiat post de prim ajutor.

Cel mai apropiat post este cel pentru care distanţa Manhattan este minimă.

De data aceasta ai scăpat pentru că timpul necesar ajungerii până la post a fost suficient; însă îţi pui întrebarea: dacă accidentul s-ar fi întâmplat în alt loc, ai fi scăpat?

Cerinţă

Se consideră N puncte în plan, reprezentând posturile de prim ajutor şi alte M puncte reprezentând posibile locaţii ale accidentului. Se cere pentru fiecare dintre cele M puncte distanţa Manhattan până la cel mai apropiat post.

Date de intrare

Fişierul **ajutor.in** conţine pe prima linie numerele întregi N şi M, în această ordine, separate printr-un spaţiu. Pe următoarele N linii se află câte două numere întregi, separate printr-un spaţiu, reprezentând coordonatele fiecărui post de prim ajutor. Pe următoarele M linii se află câte două numere întregi, separate printr-un

spaţiu, reprezentând coordonatele punctelor de accident. Coordonatele se dau în ordinea (abscisa, ordonata).

Date de ieşire

Fişierul **ajutor.out** va conţine M linii, cu câte un număr pe fiecare linie, reprezentând distanţa minimă până la cel mai apropiat post de prim ajutor.

Restricţii şi precizări

- $0 < N < 401$
- $0 < M < 500001$
- oricare coodonată este un număr întreg din intervalul $[0, 32000]$

Observaţii

- Dacă te afli deja la un post de prim ajutor (coordonatele sunt identice) distanţa e 0.
- Distanţa Manhattan este cel mai scurt drum între două puncte, mergând doar pe direcţii paralele cu axele de coordonate, adică $|x1 - x2| + |y1 - y2|$, unde $(x1, y1)$, $(x2, y2)$ sunt coordonatele celor 2 puncte.
- În fişierul de intrare pot exista puncte cu aceleaşi coordonate.
- Se acordă punctele pentru fiecare test, doar dacă toate valorile din fişierul de ieşire sunt corecte.
- Tot ce face Sprite e să-ţi potolească setea!

Exemplu

ajutor.in	ajutor.out
4 4 1 1 5 5 1 5 5 1 0 0 1 1 3 3 4 1	2 0 4 1

Timp maxim de execuţie/test: 1.3 secunde pentru Linux şi 2 secunde pentru Windows.

4.4.1 Indicaţii de rezolvare - descriere soluţie

4.4.2 Rezolvare detaliată

4.4.3 Codul sursă

4.5 Ghizi

Se caută ghizi pentru Olimpiada Naţională de Informatică. Deoarece la Olimpiadă participă K echipe, trebuie ca în fiecare moment de timp din intervalul $[0, 100)$ să existe exact K ghizi.

Pentru posturile de ghid s-au înscris N voluntari, care au fost numerotaţi distinct de la 1 la N. Fiecare dintre cei N voluntari a specificat un interval de timp în care poate asigura servicii de ghid. Voluntarul i poate fi ghid în intervalul $[T1_i, T2_i)$ (intervalul este închis în $T1_i$ şi deschis în $T2_i$).

Cerinţă

Dându-se intervalele de timp asociate celor N voluntari, determinaţi o variantă de angajare astfel înct în fiecare moment de timp să fie prezenţi exact K ghizi. Numărul total de voluntari angajaţi este irelevant.

Date de intrare

Prima linie a fişierului de intrare **ghizi.in** conţine două numere întregi N şi K, separate printr-un spaţiu, cu semnificaţiile de mai sus. Voluntarii sunt numerotaţi distinct, de la 1 la N. Fiecare dintre următoarele N linii conţine descrierea unui voluntar; mai exact linia $i+1$ conţine valorile întregi $T1_i$ şi $T2_i$ pentru voluntarul i.

Date de ieşire

În fişierul **ghizi.out** veţi afişa pe prima linie numărul total de ghizi angajaţi (M). Pe a doua linie veţi scrie M numere distincte între 1 şi N, ordonate crescător, reprezentând numerele de ordine ale ghizilor angajaţi.

Restricţii şi precizări

- $1 \leq N \leq 5000$
- $0 \leq T1 < T2 \leq 100$ pentru fiecare dintre cei N voluntari
- $1 \leq K \leq N$
- Pot exista 2 voluntari cu acelaşi interval asociat.
- Toate testele date vor avea soluţie.
- Dacă există mai multe soluţii, afişaţi una oarecare.
- La preselecţie participă numai fete.

Exemplu

ghizi.in	ghizi.out
6 2 0 100 0 15 15 99 0 10 10 20 20 100	4 1 4 5 6

Timp maxim de execuţie: 0.1 secunde/test sub Linux şi 0.3 secunde sub Windows.

4.5.1 Indicaţii de rezolvare - descriere soluţie

4.5.2 Rezolvare detaliată

4.5.3 Codul sursă

4.6 Sala

Preşedintele unui Concurs de Informatică doreşte ca la festivitatea de încheiere să fie o atmosferă plăcută. Pentru acest lucru el vrea să aşeze participanţii pe n rânduri după anumite reguli. Aceaste reguli sunt:

– pe primul rând al sălii aşează una lângă alta n persoane.

– pe rândurile următoare aşeazarea pe poziţia i (numerotarea se face de la stânga la dreapta) este condiţionată de aşezarea persoanelor de pe rândul anterior, adică dacă pe rândul din faţă pe poziţiile i şi $i+1$ stau fie numai băieţi fie numai fete, atunci se va aşeza o fată, iar dacă pe aceste poziţii stau persoane de sex opus se va aşeza un băiat.

Conform acestei reguli pe rândul cu numărul de ordine i ($i \in \{1, 2, ..., n\}$) se vor aşeza $n - i + 1$ persoane.

Cerinţă

Pentru n dat se cere să se determine numărul maxim de băieţi ce pot lua loc în sală, astfel încât să se respecte regulile anterioare.

Date de intrare

În fişierul text **sala.in** pe prima linie se află numărul de cifre ale lui n, iar pe linia a doua se află cifrele numărului n separate între ele prin câte un spaţiu.

Date de ieşire
În fişierul text **sala.out** pe prima linie se va afişa numărul din cerinţă.

Restricţii şi precizări
- $1 \leq n \leq 10^{101}$
- Pentru 30% din teste $n < 100$
- Pentru 70% din teste, $n < 30000$

Exemple

sala.in	sala.out	sala.in	sala.out
1 5	10	2 1 3	61

Timp de execuţie/test: 0.1 secunde (atât sub Windows cât şi sub Linux).

4.6.1 Indicaţii de rezolvare - descriere soluţie

4.6.2 Rezolvare detaliată

4.6.3 Codul sursă

Capitolul 5

Baraj 2004

5.1 Invsort

Se dă un şir de N numere naturale, care trebuie ordonat crescător. Singura operaţie permisă este să consideraţi elementele de pe poziţiile $i, i+1, ..., j$ (pentru i şi j arbitrare, $i < j$), şi să inversaţi ordinea acestor elemente (adică elementul de pe poziţia i ajunge pe poziţia j, $i+1$ ajunge pe poziţia $j-1$, ..., j ajunge pe poziţia i). Costul unei astfel de operaţii este numărul de elemente din subşirul inversat, şi anume $j - i + 1$.

Cerinţă

Scrieţi un program care să determine o secvenţă de operaţii care ordonează crescător şirul dat şi are un cost total cât mai mic (dar nu obligatoriu minim).

Date de intrare

Fişierul de intrare **invsort.in** conţine pe prima linie numărul N, şi apoi N linii cu elementele şirului dat (nu neapărat distincte).

Date de ieşire

Fişierul de ieşire **invsort.out** va conţine pe fiecare linie descrierea unei operaţii. O operaţie este descrisă prin numerele i şi j, separate printr-un spaţiu.

Aceste numere satisfac $1 \leq i < j \leq N$.

Restricţii şi precizări

• $2 \leq N \leq 32000$

• valorile şirului care trebuie ordonat sunt între 0 şi 31999

Punctaj

• dacă şirul de operaţii (executate în ordinea din fişierul de ieşire) nu ordonează intrarea, primiţi 0 puncte

• în cazul în care costul total este cel mult 4.000.000 (patru milioane) primiţi punctaj maxim

• în cazul în care costul total este cel mult 40.000.000 (patruzeci de milioane) primiţi 40% din punctajul pe test

• în 50% din teste şirul de intrare conţine numai elemente de 0 şi 1

• pentru toate testele folosite la corectare, $N = 32000$

Exemplu

invsort.in	**invsort.out**	Explicaţie
5 1 0 1 1 0	3 5 1 3	• prima operaţie are efectul: 10[110] → 10011 • a doua operaţie are efectul: [100]11 → 00111 • costul total este 3 + 3 = 6

Timp maxim de execuţie/test: 0.5 secunde pentru Linux şi 2 secunde pentru Windows.

5.1.1 Indicaţii de rezolvare - descriere soluţie

5.1.2 Rezolvare detaliată

5.1.3 Codul sursă

5.2 Peri

Se consideră o matrice dreptunghiulară A cu m linii şi n coloane cu valori 0 sau 1, liniile şi coloanele fiind numerotate de la 1 la m, respectiv de la 1 la n. Numim dreptunghi de colţuri $(x1, y1)$ $(x2, y2)$ cu $x1 < x2$ şi $y1 < y2$ mulţimea elementelor

A_{ij} cu $x1 \leq i \leq x2$ şi $y1 \leq j \leq y2$. Numim perimetru al dreptunghiului de colţuri $(x1, y1)$ $(x2, y2)$ mulţimea elementelor A_{ij} pentru care $(i = x1$ şi $y1 \leq j \leq y2)$ sau $(i = x2$ şi $y1 \leq j \leq y2)$ sau $(j = y1$ şi $x1 \leq i \leq x2)$ sau $(j = y2$ şi $x1 \leq i \leq x2)$.

Cerinţă

Determinaţi diferenţa maximă dintre numărul de elemente egale cu 1 şi numărul de elemente egale cu 0 aflate pe perimetrul aceluiaşi dreptunghi, precum şi numărul de dreptunghiuri pentru care se obţine această diferenţă.

Date de intrare

Pe prima linie a fişierului de intrare **peri.in** sunt scrise numerele m şi n, separate printr-un singur spaţiu. Pe următoarele m linii este dată matricea A, numerele de pe aceeaşi linie fiind separate de câte un spaţiu.

Date de ieşire

Fişierul de ieşire **peri.out** va conţine o singură linie pe care se află două numere întregi separate printr-un spaţiu. Primul număr este diferenţa maximă dintre numărul de elemente 1 şi numărul de elemente 0 de pe perimetrul unui dreptunghi. Al doilea întreg este numărul de dreptunghiuri pentru care diferenţa dintre numărul de elemente 1 şi numărul de elemente 0 de pe perimetru este maximă.

Restricţii şi precizări

- $1 \leq m, n \leq 250$
- Prin diferenţă nu se înţelege diferenţă în valoare absolută!

Exemplu

peri.in	peri.out
4 5 1 0 0 1 0 0 1 1 0 0 0 1 0 1 0 1 1 1 0 1	4 2

Timp maxim de execuţie/test: 0.2 secunde pentru Linux şi 1 secundă pentru Windows.

5.2.1 Indicaţii de rezolvare - descriere soluţie

5.2.2 Rezolvare detaliată

5.2.3 Codul sursă

5.3 Trans

În depozitul unei companii de construcţii se află N blocuri de piatră, de culoare albă sau neagră. Ele sunt aşezate în ordinea $1, 2, .., N$, de la intrarea în depozit către interior.

Blocurile de piatră trebuie să fie transportate pe un şantier de construcţii, în ordinea în care ele sunt depozitate, iar pentru aceasta va trebui închiriat un camion de la o companie de transport. Aceasta deţine Q tipuri de camioane. Camionul de tipul i ($1 \leq i \leq Q$) poate transporta maxim K_i blocuri de piatră la un moment dat şi pentru un transport se percepe taxa T_i.

Compania de transport este de părere că, pentru a-şi păstra clientela, trebuie să impună anumite standarde, indiferent de cât de absurde ar fi, deci impune condiţia ca toate blocurile de piatră plasate în camion la un transport să aibă aceeaşi culoare. Aşadar, pentru a fi transportate toate blocurile pe şantier, compania de construcţii va alege un camion de un anumit tip, iar camionul va efectua unul sau mai multe transporturi.

Pentru a micşora suma totală plătită, compania de construcţii are posibilitatea de a schimba culoarea oricărui bloc de piatră (din alb în negru sau din negru în alb); pentru fiecare bloc i ($1 \leq i \leq N$) se cunoaşte suma S_i ce trebuie plătită pentru a-i schimba culoarea.

Cerinţă

Pentru fiecare dintre cele Q tipuri de camioane deţinute de compania de transport, determinaţi suma minimă pe care o va plăti compania de construcţii pentru a transporta toate cele N blocuri pe şantier.

Date de intrare

Fişierul de intrare **trans.in** conţine:

• pe prima linie numărul întreg N, reprezentând numărul de blocuri de piatră din depozit.

• Pe fiecare dintre următoarele N linii se află informaţii referitoare la câte un bloc de piatra. Pe a i-a dintre aceste N linii se găsesc două numere întregi separate printr-un spaţiu: C_i S_i, reprezentând culoarea celui de-al i-lea bloc (C_i este 0 pentru alb şi 1 pentru negru) şi respectiv suma ce trebuie plătită pentru a-i schimba culoarea (dacă este necesar).

• Pe următoarea linie se află numărul natural Q, reprezentând numărul de tipuri de camioane deţinute de compania de transport.

• Pe fiecare dintre următoarele Q linii se află informaţii referitoare la câte un camion. Pe cea de a i-a dintre aceste Q linii sunt scrise două numere naturale separate printr-un spaţiu K_i T_i, reprezentând numărul maxim de blocuri ce pot

fi transportate simultan de către un camion de tipul i şi respectiv taxa ce trebuie plătită pentru fiecare transport efectuat.

Date de ieşire

Fişierul de ieşire **trans.out** va conţine Q linii. Pe cea de a i-a dintre aceste linii va fi afişată suma totală minimă plătită de compania de construcţii pentru a transporta toate cele N blocuri de piatră, în cazul în care ar închiria un camion de tipul i.

Restricţii şi precizări

- $1 \leq N \leq 16000$
- $1 \leq S_i \leq 10000$
- $1 \leq Q \leq 100$
- $1 \leq K_i \leq N$
- $1 \leq T_i \leq 100000$
- Cel puţin 40% dintre teste vor avea $Q \leq= 10$ şi $K_i \leq \min(N, 100)$

Exemplu

trans.in	**trans.out**
4 0 2 1 3 0 10 1 2 3 4 1000 4 1 2 5	105 4 14

Timp maxim de execuţie/test: 0.3 secunde pentru Linux şi 1 secundă pentru Windows.

5.3.1 Indicaţii de rezolvare - descriere soluţie

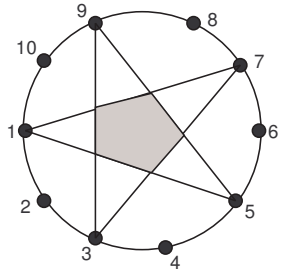
5.3.2 Rezolvare detaliată

5.3.3 Codul sursă

5.4 Poligon

Geo a învăţat o metodă de a fixa n puncte pe un cerc de rază r, astfel încât să împartă cercul în n coarde egale ca lungime. Apoi şi-a ales un număr k şi a început să unească punctele succesiv, din k în k, păstrând acelaşi sens, până ce a ajuns în punctul din care a pornit. Astfel, dacă a fixat $n = 10$ puncte pe cerc pe care le-a numerotat $1, 2, ..., 10$ (vezi figura) şi şi-a ales $k = 6$, atunci el uneşte punctul 1 cu 7, apoi pe 7 cu 3, apoi 3 cu 9, apoi 9 cu 5, şi în sfârşit 5 cu 1.

Apoi a colorat poligonul format în interior, pornind din centrul cercului şi



fără a depăşi vreuna dintre liniile desenate.

El se întreabă în final câte laturi are poligonul colorat şi care este aria acestuia.

Cerinţă

Pentru n, k şi r numere naturale date, se cere numărul de laturi L ale poligonului colorat şi aria S a acestuia (cu 2 zecimale exacte).

Date de intrare

Din fişierul **poligon.in** se citesc trei numere naturale n, k şi r despărţite prin câte un spaţiu.

Date de ieşire

În fişierul **poligon.out** se scriu, pe linii diferite două valori: pe prima linie numărul L de laturi ale poligonului colorat, iar pe linia a doua numărul real reprezentând aria acestuia.

Restricţii şi precizări

- $3 < n < 10001$ număr natural
- $0 < k < n$ număr natural
- pentru n par, $2 * k \neq n$
- $10 < r < 501$
- Pentru fiecare test, dacă numărul de laturi determinat este corect, primiţi 20% din punctajul maxim de pe testul respectiv. În plus, dacă şi aria determinată este corectă, primiţi punctajul maxim.
- Pentru 70% din testele folosite la evaluare, $n < 501$

Exemple

poligon.in	poligon.out	poligon.in	poligon.out
10 6 100	5 3468.92	13 200 30	30 5452.04

Timp maxim de execuţie/test: 0.1 secunde pentru Linux şi 0.1 secunde pentru Windows.

5.4.1 Indicaţii de rezolvare - descriere soluţie

5.4.2 Rezolvare detaliată

5.4.3 Codul sursă

5.5 Poliţie

Gigel şi Costel sunt doi poliţişti cu experienăă. Ei lucrează împreună de mulţi ani şi de multe ori au fost nominalizaţi pentru premiul "Poliţiştii anului". Anul acesta sunt hotărâţi să-l câştige şi pentru aceasta trebuie să încaseze cât mai mulţi bani din amenzi.

În fiecare zi, Gigel şi Costel pot aplica trei tipuri de amenzi pentru următoarele evenimente:

Tip	Semnificaţie	Suma încasată	Durata de aplicare
1	Pentru pietoni care traversează neregulamentar	S1	T1
2	Pentru şoferi de autovehicule care încalcă regulile de circulaţie	S2	T2
3	Pentru şoferi de maşini grele, cu transport ilegal de mărfuri	S3	T3

Amenzile de tipul 1 şi 2 pot fi aplicate de un singur poliţist (Gigel sau Costel). Pentru o amendă de tipul 3, Gigel şi Costel trebuie să lucreze împreună (unul verifică actele de transport, iar celălalt verifică marfa).

Durata de aplicare a unei amenzi reprezintă timpul necesar poliţiştilor pentru a verifica acte, a scrie proces verbal, etc. Dacă un poliţist aplică o amendă la momentul x, iar durata aplicării amenzii este y, poliţistul care aplică amenda va deveni disponibil abia la momentul $x + y$. Poliţiştii nu fac minute suplimentare, ei sunt în activitate din minutul 1 până în minutul T exclusiv, deci trebuie să fie liberi să plece acasă în minutul T. Pentru că în noul cod rutier nu le mai este

permis poliţiştilor să tragă şoferii pe dreapta şi să-i lase să aştepte, pentru a aplica o amendă de tipul 1 sau 2 trebuie să fie liber măcar un poliţist, iar pentru a aplica o amendă de tipul 3 ambii poliţişti trebuie să fie liberi la momentul în care survine evenimentul.

Cei doi poliţişti stau la pândă şi observă evenimentele din trafic. Dacă nu pot aplica amenzi pentru toate evenimentele care intervin în trafic, ei sunt nevoiţi să le aleagă pe acelea care, în total, le aduc mai mulţi bani.

Cerinţă

Scrieţi un program care să determine suma maximă pe care o pot încasa din amenzi Gigel şi Costel într-o tură.

Date de intrare

Fişierul de intrare **politie.in** conţine:

– pe prima linie numărul natural T, reprezentând minutul la care cei doi poliţişti sunt liberi să plece acasă;

– pe linia a doua, două numere naturale $S1$ $T1$ (suma încasată şi durata aplicării unei amenzi de tipul 1);

– pe linia a treia, două numere naturale $S2$ $T2$ (suma încasată şi durata aplicării unei amenzi de tipul 2);

– pe linia a patra, două numere naturale $S3$ $T3$ (suma încasată şi durata aplicării unei amenzi de tipul 3);

– pe linia a cincea, un număr natural N (numărul de evenimente survenite în trafic);

– pe fiecare dintre următoarele N linii se află două numere naturale *tip timp* (*tip* poate fi 1, 2 sau 3 şi reprezentă tipul amenzii ce poate fi aplicată; *timp* reprezintă timpul la care a survenit evenimentul, exprimat în număr de minute faţă de începutul turei).

Numerele scrise pe aceeaşi linie sunt separate prin câte un spaţiu. Evenimentele sunt în ordine cronologică.

Date de ieşire

Fişierul de ieşire **politie.out** conţine o singură linie pe care este scrisă suma maximă ce poate fi încasată.

Restricţii şi precizări

- $0 < T, T1, T2, T3 < 201$
- $0 < S1, S2, S3 < 51$
- $0 < N < 501$
- Evident, există evenimente care intervin simultan în trafic.

Exemplu

politie.in	**politie.out**	Explicaţie
300	140	
10 20		Aplică amândoi amenda de tipul 3
30 30		din minutul 10, apoi
50 25		
8		Costel aplică singur amenda de tipul 1
1 5		din minutul 130, apoi
3 10		
3 20		Gigel aplică singur amenda de tipul 2
1 130		din minutul 142, apoi
2 142		
2 160		amândoi aplică amenda de tipul 3
3 180		din minutul 180.
2 280		

Timp maxim de execuţie/test: 0.2 secunde pentru Linux şi 1.2 secunde pentru Windows.

5.5.1 Indicaţii de rezolvare - descriere soluţie

5.5.2 Rezolvare detaliată

5.5.3 Codul sursă

5.6 Sea

Pe mare se află N vapoare. Malul este în mod curios perfect drept şi este reprezentat prin axa Ox a sistemului de coordonate. Cele N vapoare sunt reprezentate prin perechi de coordonate (Vx_i, Vy_i), unde Vy_i este strict pozitiv (marea este deasupra axei Ox). Pe mal se află M faruri, date prin coordonatele lor Fx_i (fiind exact la limita dintre mare şi uscat, y-ul lor este întotdeauna 0).

Cele M faruri sunt ciudate pentru că ele nu pot lumina decât în stânga. Astfel aria luminată de fiecare far i este delimitată de un sfert de cerc cu o rază Fr_i. Mai exact, un vapor este luminat de un anumit far dacă se află în stânga farului (are x-ul mai mic) şi distanţa de la far la vapor este mai mică sau egală cu valoarea Fr_i asociată farului respectiv.

Pentru fiecare far se mai dă şi un număr natural strict pozitiv Fn_i.

Din motive greu de înţeles, şeful portului doreşte ca fiecare far i să lumineze cel puţin Fn_i vapoare (un vapor poate fi luminat de mai multe faruri). El doreşte consum minim de energie şi vrea să afle pentru fiecare far raza minimă necesară pentru a lumina numărul cerut de vapoare.

Cerinţă

Determinaţi pentru fiecare far valoarea Fr_i care reprezintă raza minimă necesară pentru ca farul să lumineze cel puţin Fn_i vapoare.

Date de intrare

• Prima linie a fişierului **sea.in** conţine două numere întregi N şi M separate printr-un spaţiu, reprezentând numărul de vapoare, respectiv numărul de faruri.

• Fiecare dintre următoarele N linii conţine câte o pereche de numere reale separate printr-un spaţiu Vxi şi Vyi (coordonatele vapoarelor).

• Fiecare dintre următoarele M linii conţine câte o pereche de numere separate printr-un spaţiu, unul real Fx_i şi unul întreg Fni (coordonatele orizontale şi numerele asociate farurilor).

Date de ieşire

Fişierul **sea.out** va conţine M linii, fiecare linie conţinând un număr real, dat cu 4 zecimale: pe linia i se află raza minimă necesară pentru ca farul i să lumineze Fn_i vapoare.

Restricţii şi precizări

• $1 \leq N \leq 400$, $1 \leq M \leq 100000$

• $0 < y, r < 100000$, $-100000 < x < 100000$, $1 \leq Fn_i \leq N$

• în fişierul de intrare farurile sunt sortate crescător după coordonatele x.

• Nu vor exista două vapoare, sau un far şi un vapor cu acelaşi x. În schimb pot exista două sau mai multe faruri cu acelaşi x, caz în care ele vor fi unul lângă altul în fişierul de intrare (evident din moment ce sunt sortate după x). Ordinea în care apar în fişierul de intrare farurile cu acelaşi x nu este definită. Pot exista chiar două faruri identice.

• Numerele reale din fişierul de intrare vor avea maxim 4 zecimale

• Rezultatul va fi verificat cu o precizie de 0.001 (rezultatul va fi considerat corect dacă modulul diferenţei dintre rezultatul corect şi cel furnizat de concurent nu depăşeşte 0.001)

• Există întotdeauna soluţie (pentru fiecare far i vor exista întotdeauna cel puţin Fn_i vapoare în stânga lui).

Exemplu

sea.in	sea.out
3 5	5.0990
-0.5 0.5	0.7071
-2 5	5.3852
3 4	0.7071
-1 1	4.4721
0 1	
0 2	
0 1	
5 1	

Timp maxim de execuţie/test: 0.8 secunde pentru Linux şi 1.6 secunde pentru Windows.

5.6.1 Indicaţii de rezolvare - descriere soluţie

5.6.2 Rezolvare detaliată

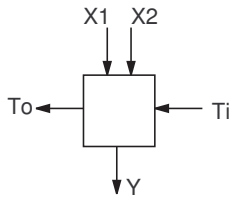
5.6.3 Codul sursă

Capitolul 6

Baraj 2005

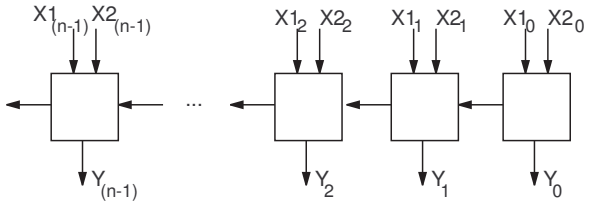
6.1 Anticip

Un sumator pe un bit este un mic dispozitiv cu 3 intrări şi 2 ieşiri. El primeste la intrare $X1$, $X2$ şi Ti. $X1$ şi $X2$ sunt biţii ce trebuie adunaţi, iar Ti este transportul anterior (ca intrare). La iesire furnizeaza Y şi To. Y este suma, iar To este transportul urmator (ca iesire).



Pentru a formaliza aceste lucruri putem scrie
$Y = (X1 + X2 + Ti) mod 2$ (mod este restul împărţirii întregi)
$To = (X1 + X2 + Ti) div 2$ (div este câtul împărţirii întregi)

Pentru a aduna numere pe N biţi se folosesc N astfel de sumatoare. Ele sunt legate ca în figura de mai jos, adică transportul de ieşire al unui sumator este transportul de intrare pentru urmatorul.



Problema cu aceste sumatoare pe mai mulţi biţi este că un sumator trebuie să astepte transportul de la unitatea anterioară (exceptând primul sumator).

Dacă un sumator pe un bit face calculul într-o secunda, atunci pentru un sumator pe N biţi (format din N sumatoare pe un bit) vor fi necesare N secunde.

Pentru a îmbunătăţi performanţa acestor sumatoare pe N biţi s-au introdus nişte unităti capabile să anticipeze transportul, adică intrarea T_i. Aceste unităţi verifică datele de intrare precedente $X1_{(i-1)}$ si $X2_{(i-1)}$. Dacă amândouă sunt 0 atunci T_i va fi 0, indiferent de ce primeşte acea unitate ca transport de intrare. De asemenea dacă amândoua sunt 1 atunci T_i va fi 1. Toate sumatoarele care folosind anticipaţia pot calcula transportul de la sumatorul precedent încep calculul odată cu primul sumator. Comunicarea transportului de la un sumator la urmatorul se realizează instantaneu.

Cercetătorii care au inventat aceste unităţi de transport vor să ştie cu cât îmbunatăţesc performanţa sistemului şi au hotarât să se facă toate adunările posibile, pentru a compara cu vechiul sistem. Prin toate adunările posibile se înţelege că se va aduna orice numar pe N biţi cu orice număr pe N biţi fix o dată (în total 4^N adunari). Ei vor să ştie cât au durat aceste operaţii, adică suma tuturor timpilor pentru fiecare adunare.

Cerinţă

Scrieţi un program care să determine numărul total de secunde necesare pentru toate adunările, folosind dispozitivele de anticipare.

Date de intrare

Fişierul de intrare **anticip.in** conţine o singura linie pe care se află numărul natural N reprezentând numarul de biţi.

Date de ieşire

Fişierul de ieşire **anticip.out** va conţine o singură linie pe care va fi scris un număr natural reprezentând numărul de secunde necesare tuturor adunărilor posibile.

Restricţii şi precizări

- $0 < N < 51$
- Problema aceasta nu are nici o legatură cu vreun balaur.

Exemplu

anticip.in	anticip.out
3	128

Explicaţii

16 adunări se realizează într-o secundă; 32 adunări se realizează în două secunde; 16 adunări se realizează în trei secunde. De exemplu, adunarea 010+001 necesită 3 secunde. Adunarea 010+000 necesită 2 secunde. Adunarea 010+110 necesită o secundă (primul sumator adună biţii din dreapta).

Timp maxim de executie pe test: 0.5 secunde sub sistemul de operare Linux.

6.1.1 Indicaţii de rezolvare - descriere soluţie

6.1.2 Rezolvare detaliată

6.1.3 Codul sursă

6.2 Galaxii

Într-un viitor mai mult sau mai puţin apropiat, oamenii vor popula n galaxii, numerotate de la 1 la n. Datorită dezvoltării tehnologiei de transport spaţial este posibil ca din oricare galaxie să se ajungă printr-un zbor în oricare altă galaxie. Fiecare pereche de galaxii defineşte astfel un zbor care poate fi parcurs în orice sens.

La un moment dat se ia hotărârea ca $(n + 1)div2$ companii de transport să asigure libera circulaţie a oamenilor între galaxiile existente. Companiile sunt

numerotate de la 1 la $(n+1)div2$. Fiecarei companii x i se va atribui o mulţime de zboruri Z_x. Notam cu G_x mulţimea galaxiilor deservite de compania x prin zborurile din multimea Z_x.

Zborurile vor fi distribuite conform următoarelor reguli:

1) prin zborurile din Z_x, compania x poate transporta oameni între oricare două galaxii din G_x, (fie direct, fie trecând prin alte galaxii din G_x);

2) pentru oricare două galaxii din G_x, modul în care compania x transportă călătorii de la o galaxie la cealaltă este unic;

3) fiecare zbor trebuie să fie atribuit exact unei singure companii.

Cerinţă

Scrieţi un program care să determine o modalitate de atribuire a zborurilor celor $(n+1)div2$ companii în conformitate cu regulile enunţate mai sus.

Date de intrare

Fişierul de intrare **galax.in** conţine o singură linie pe care va fi scris un numar natural n, reprezentând numărul de galaxii.

Date de ieşire

Fişierul de iesire **galax.out** va conţine câte o linie pentru fiecare pereche de galaxii. Pe aceasta linie vor fi scrise 3 numere naturale a b c, cu semnificaţia "zborul dintre galaxiile a şi b este atribuit companiei c".

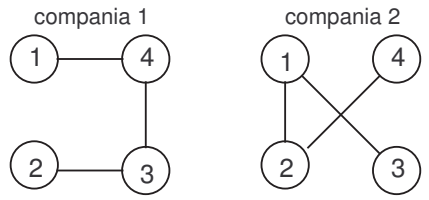
Restricţii şi precizări

- $3 < n < 1001$
- $adivb$ este câtul împărţirii întregi a lui a la b.

Exemple

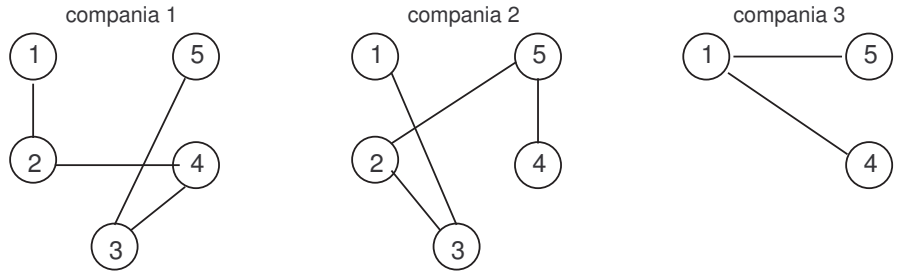
galax.in	galax.out
4	1 4 1 1 2 2 4 3 1 3 1 2 3 2 1 2 4 2

Explicaţie



galax.in	galax.out
5	1 2 1 1 5 3 2 4 1 4 3 1 5 3 1 1 3 2 2 3 2 1 4 3 5 2 2 5 4 2

Explicaţie



Timp maxim de executie pe test: 0.5 secunde sub sistemul de operare Linux.

6.2.1 Indicaţii de rezolvare - descriere soluţie

6.2.2 Rezolvare detaliată

6.2.3 Codul sursă

6.3 Texan

Un texan are o păşune cu frontiera sub formă de poligon convex. Fiindcă a ajuns la o vârsta care nu îi mai permite să meargă cu vitele la păscut pe păsunea sa, hotărăşte ca o parte din păsune să o doneze celui mai vrednic dintre nepoţii săi. Astfel el le pune la dispoziţie nepoţilor săi coordonatele carteziene ale colţurilor

păşunii şi le cere sa găsească pe frontiera păşunii 3 poziţii în care să plaseze trei tăruşi, astfel încât unind cei 3 ţăruşi prin sârma ghimpată să obţină un triunghi echilateral.

Cerinţă

Scrieţi un program care să ajute nepoţii să determine poziţiile celor 3 ţăruşi.

Date de intrare

Fişierul de intrare **texan.in** conţine:

• Pe prima linie numărul natural n, care reprezintă numărul de colţuri ale păşunii

• Pe următoarele n linii se află câte o pereche de numere reale, care reprezintă coordonatele colţurilor păşunii separate printr-un spaţiu (în ordinea: abscisa ordonata). Colţurile păşunii sunt specificate în ordinea inversă a acelor de ceasornic.

Date de ieşire

Fişierul de ieşire **texan.out** va conţine trei linii. Fiecare linie conţine coordonatele unuia dintre cei trei ţăruşi (în ordinea: abscisă ordonată) cu un spatiu între ele. Aceste coordonate vor fi specificate cu 6 zecimale (cu rotunjire).

Restricţii şi precizări

• $4 < n < 501$

• Coordonatele colţurilor păşunii sunt numere raţionale din intervalul $[-7000, 7000]$.

• La evaluare o soluţie este considerată corectă cu o marjă de eroare de 0.01.

• Dacă soluţia nu este unică, va fi afişată una oarecare.

• În fişierele de test, distanţa dintre oricare două colţuri ale păşunii este ≥ 1.

• Latura triunghiului echilateral determinat trebuie să fie > 0.1.

• Pentru datele de test există întotdeauna o soluţie care respectă cerinţele problemei.

Exemplu

texan.in	**texan.out**
5 10 0 15 15 0 20.5 -10 15 0 0	12.500000 7.500000 -3.150637 4.725956 2.272289 19.666827

Timp maxim de execuţie/test: 0.2 secunde sub sistemul de operare Linux

6.3.1 Indicaţii de rezolvare - descriere soluţie

6.3.2 Rezolvare detaliată

6.3.3 Codul sursă

6.4 Bşir

Se dă un număr natural N.

Definim un **bşir** de lungime N ca fiind şirul $x_0, x_1, ..., x_{N-1}$, unde

• $x_i = 1$ dacă i are un număr impar de biţi egali cu 1 în reprezentare binară şi

• $x_i = 0$ dacă i are un număr par de biţi egali cu 1 în reprezentare binară, pentru orice $0 \leq i < N$.

De exemplu, pentru $N = 7$ obţinem următorul **bşir** de lungime 7: 0110100

Explicaţii privind obţinerea **bşir**-ului:

i (în baza 10)	i (în baza 2)	valoarea de pe poziţia i din **bşir**
0	0	0
1	1	1
2	10	1
3	11	0
4	100	1
5	101	0
6	110	1

Cerinţă

Determinaţi numărul M de secvenţe palindromice de lungime cel puţin 2 dintr-un **bşir** de lungime N.

Date de intrare

Fişierul de intrare **bsir.in** conţine pe prima linie numărul natural N.

Date de ieşire

Fişierul de ieşire **bsir.out** va conţine M modulo 30103 (restul împărţirii lui M la 30103).

Restricţii şi precizări

• $2 \leq N \leq 10^{18}$

• Secvenţa x_i, x_{i+1}, ..., x_{j-1}, x_j se numeşte palindromică dacă $x_i = x_j$, $x_{i+1} = x_{j-1}$, ...

Exemple

bsir.in	**bsir.out**	**bsir.in**	**bsir.out**
10	8	21	30

Timp maxim de execuţie/test: 0.1 secunde sub sistemul de operare Linux.

6.4.1 Indicaţii de rezolvare - descriere soluţie

6.4.2 Rezolvare detaliată

6.4.3 Codul sursă

6.5 Evantai

Lui Algorel îi plac mult şirurile de numere naturale cu proprietăţi cât mai ciudate. Căutând astfel de ciudăţenii ale informaticii, a găsit printr-o carte prăfuită de vreme un nou tip de şir denumit evantai. Un evantai este un şir cu un număr par de termeni, $E_1E_2...E_{2K}$, cu următoarea proprietate:

$E_1 + E_{2K} > E_2 + E_{2K-1} > ... > E_K + E_{K+1}$

Cerinţă

Fiind dat un şir de numere naturale distincte $A_1A_2...A_N$, Algorel vrea să afle câte subşiruri ale acestuia sunt evantaie.

Date de intrare

Prima linie a fişierului **evantai.in** conţine numărul întreg N, reprezentând numărul de elemente ale şirului. Următoarele N linii conţin, în ordine, elementele şirului A.

Date de ieşire

Pe prima linie a fişierului **evantai.out** se va afla un singur număr întreg C, reprezentând numărul de subşiruri evantai. Rezultatul va fi afişat modulo 30103.

Restricţii şi precizări

• $2 \leq N \leq 700$

• elementele şirului sunt numere întregi distincte cuprinse între 1 şi 1000

• prin subşir se înţelege orice înşiruire de termeni A_{i1} A_{i2} ... A_{ik} astfel încât $i1 < i2 < ... < ik$.

Exemplu

evantai.in	evantai.out
4 1 2 3 6	7

Timp maxim de execuţie/test: 1 secundă sub sistemul de operare Linux

6.5.1 Indicaţii de rezolvare - descriere soluţie

6.5.2 Rezolvare detaliată

6.5.3 Codul sursă

6.6 Spioni

Serviciile secrete din Ţara Crivăţului au o reţea foarte bine pusă la punct. Reţeaua este formată din N centre, numerotate de la 1 la N. Între centre există drumuri ce pot fi parcurse în ambele sensuri, de lungimi cunoscute. Un drum uneşte două centre. Utilizând drumurile existente, între oricare două centre există legătură (directă sau trecând prin alte centre). Distanţa dintre două centre este lungimea totală minimă a drumurilor parcurse pentru a ajunge de la un centru la celălalt.

Şeful Teo a hotărât împărţirea tuturor centrelor în două departamente, de spionaj şi de contraspionaj. O împărţire este considerată optimală dacă maximul distanţelor dintre oricare două centre din cadrul aceluiaşi departament este minim.

Dacă există mai multe soluţii cu acelaşi maxim, se alege soluţia pentru care diferenţa (în valoare absolută) dintre numărul de centre din departamentul spionaj şi numărul de centre din departamentul contraspionaj este minimă. Dacă şi în acest caz există mai multe soluţii, este preferată prima în ordine lexicografică.

Cerinţă

Dându-se descrierea reţelei, să se scrie un program care să găsească o împărţire optimală a centrelor în două departamente.

Date de intrare

Fişierul de intrare **spioni.in** conţine pe prima linie numerele naturale N şi M reprezentând numărul de centre şi respectiv numărul de drumuri dintre ele. Pe fiecare dintre următoarele M linii vor fi scrise câte trei numere naturale; mai exact,

pe linia $i+1$ sunt scrise numerele ai b_i c_i cu semnificaţia "există un drum între centrul a_i şi centrul b_i de lungime c_i". Numerele scrise pe aceeaşi linie în fişierul de intrare sunt separate prin câte un spaţiu.

Date de ieşire

Fişierul de ieşire **spioni.out** va conţine două linii. Pe prima linie va fi scris un număr natural care reprezintă maximul distanţelor dintre două centre din acelaşi departament. Pe cea de a doua linie vor fi scrise N caractere. Caracterul i va fi litera C dacă centrul i va fi în departamentul contraspionaj sau litera S dacă centrul i va fi în departamentul spionaj în împărţirea optimală determinată.

Restricţii şi precizări

- $2 < N < 101$
- $0 < a_i, b_i \leq N$
- $0 < M, c_i < 16001$
- Spunem că împărţirea $(x_1, x_2, ..., x_N)$ precedă din punct de vedere lexicografic împărţirea $(y_1, y_2, ..., y_N)$ dacă există k astfel încât $x_i = y_i$, pentru orice $i < k$ şi $x_k < y_k$; litera $C <$ litera S.
- Se vor acorda punctaje parţiale după cum urmează:

– pentru distanţa maximă determinată corect: 20% din punctajul testului respectiv

– dacă soluţia este corectă (din punctul de vedere al distanţei maxime şi al diferenţei absolute minime), dar nu este prima în ordine lexicografică: 60%

– pentru obţinerea primei soluţii corecte în ordine lexicografică: 100%.

Exemple

spioni.in	**spioni.out**
5 4 1 2 1 2 3 1 3 4 1 2 5 7	3 CCCCS

Explicaţie:

Maximul distanţelor dintre oricare două centre din cadrul aceluiaşi departament este 3.

Diferenţa în valoare absolută dintre numărul de centre din departamentul spionaj şi cel de contraspionaj este 3.

Soluţia este prima în ordine lexicografică.

O altă soluţie ar fi SSSC, dar aceasta este mai mare lexicografic.

spioni.in	spioni.out
5 5 1 3 1 3 2 1 2 5 7 3 4 7 2 4 1	3 CCCCS

Explicaţie

Maximul distanţelor dintre oricare două centre din cadrul aceluiaşi departament este 3.

Diferenţa în valoare absolută dintre numărul de centre din departamentul spionaj şi cel de contraspionaj este 3.

Timp maxim de execuţie/test: 0.1 secunde sub sistemul de operare Linux

6.6.1 Indicaţii de rezolvare - descriere soluţie

6.6.2 Rezolvare detaliată

6.6.3 Codul sursă

Capitolul 7

Baraj 2006

7.1 Acolor

Omida-agent Smith s-a săturat să tot distrugă arborii şi acum îşi dezvoltă simţul artistic - îi place mult mai mult să-i coloreze.

De fiecare dată când vrea să creeze o nouă arbo-pictură îşii ia cu el cele K creioane colorate, îşi alege un arbore din grădină şi porneşte la lucru.

Arborele ales de Smith este alcătuit din N noduri, are ca rădacină nodul R şi o formă potrivită pentru pictură:

- fiecare nod are cel mult două crengi care duc spre două noduri: unul la stânga şi/sau unul la dreapta;

- între oricare două noduri există un drum unic format din crengi distincte, pe care omida se poate plimba pentru a ajunge de la un nod la celălalt;

- nodurile din subarborele stâng al unui nod sunt toate plasate mai la stânga decât acesta, iar cele din subarborele drept sunt toate mai la dreapta, de aceea nodurile au fost etichetate de la 1 la N de la cel mai din stânga până la cel mai din dreapta.

Omida a observat că picturile sale sunt frumoase doar dacă respectă unele reguli de bază pe care le-a citit într-o carte:

- orice nod trebuie să fie colorat cu exact una dintre cele K colori;

- un nod trebuie să fie colorat diferit faţă de părintele dinspre rădăcină (adică faţă de nodul care precedă nodul respectiv atunci când omida se plimbă pe drumul de la rădăcină la nod);

- privit din exterior arborele trebuie să fie colorat diferit de la stânga la dreapta: orice nod are o culoare diferită de cel mai apropiat nod la stânga de el şi faţă de cel mai apropiat nod la dreapta (cu alte cuvinte culoarea nodului etichetat cu i trebuie să fie diferită de culoarea nodurilor etichetate cu $i-1$, $i+1$).

Cerinţă

Scrieţi un program care să determine pentru un arbore dat în câte picturi frumoase (picturi care să respecte criteriile din enunţ) poate fi transformat acesta. Deoarece numărul cerut poate fi foarte mare, este suficient să aflaţi restul împărţirii la 10007.

Date de intrare

Fişierul de intrare **acolor.in** va conţine pe prima linie numerele întregi N, R, K separate prin câte un spaţiu. Pe următoarele N linii este descrisă structura arborelui. Mai exact, pe linia $i+1$ vor exista două numere st_i, dr_i separate printr-un spaţiu, reprezentând nodul fiu spre stânga şi respectiv nodul fiu spre dreapta al nodului i. Dacă un nod nu are fiu spre stânga şi/sau fiu spre dreapta atunci numărul corespunzător va fi 0.

Date de ieşire

Fişierul de ieşire **acolor.out** va conţine o singură linie pe care va fi scris numărul de picturi frumoase care se pot obţine pentru arborele dat.

Restricţii şi precizări

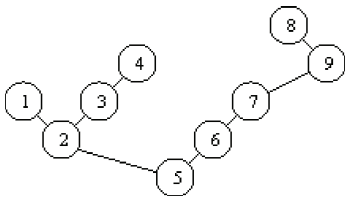
$0 < N \leq 100000$, $1 \leq R \leq N$, $1 \leq K \leq 100$

Se acordă 40 de puncte pentru teste cu $N \leq 100$ şi $K \leq 10$. Se acordă 60 de puncte pentru teste cu $N \leq 400$ şi $K \leq 15$.

Memorie disponibilă: 64 MB, din care 5 MB pentru stivă.

Exemplu

acolor.in	acolor.out	acolor.in	acolor.out
9 5 4 0 0 1 3 0 4 0 0 2 6 0 7 0 9 0 0 8 0	3601	3 1 2 0 3 0 0 2 0	0



Timp maxim de execuţie: 0.6 secunde/test

7.1.1 Indicaţii de rezolvare - descriere soluţie

7.1.2 Rezolvare detaliată

7.1.3 Codul sursă

7.2 Cifru

Copiii solarieni se joacă adesea trimiţându-şi mesaje codificate. Pentru codificare ei folosesc un cifru bazat pe o permutare p a literelor alfabetului solarian şi un număr natural d.

Alfabetul solarian conţine m litere foarte complicate, aşa că noi le vom reprezenta prin numere de la 1 la m.

Dat fiind un mesaj în limbaj solarian, reprezentat de noi ca o succesiune de n numere cuprinse între 1 şi m, $c_1c_2...c_n$, codificarea mesajului se realizează astfel: se înlocuieşte fiecare literă c_i cu $p(c_i)$, apoi şirul obţinut $p(c_1)p(c_2)...p(c_n)$

se roteşte spre dreapta, făcând o permutare circulară cu d poziţii rezultând şirul $p(c_{n-d+1})...p(c_{n-1})p(c_n)p(c_1)p(c_2)...p(c_{n-d})$.

De exemplu, pentru mesajul 2 1 3 3 2 1, permutarea $p = (312)$(adică $p(1) = 3$, $p(2) = 1$, $p(3) = 2$) şi $d = 2$. Aplicând permutarea p vom obţine şirul 1 3 2 2 1 3, apoi rotind spre dreapta şirul cu două poziţii obţinem codificarea 1 3 1 3 2 2.

Cerinţă

Date fiind un mesaj necodificat şi codificarea sa, determinaţi cifrul folosit (permutarea p ţ număul d).

Date de intrare

Fişierul de intrare **cifru.in** conţine pe prima linie numele naturale n şi m, separate prin spaţiu, reprezentând lungimea mesajului şi respectiv numărul de litere din alfabetul solarian. Pe cea de a doua linie este scris mesajul necodificat ca o succesiune de n numere cuprinse între 1 şi m separate prin câte un spaţiu. Pe cea de a treia linie este scris mesajul codificat ca o succesiune de n numere cuprinse între 1 şi m separate prin câte un spaţiu.

Date de ieşire

Fişierul de ieşire **cifru.out** va conţine pe prima linie numărul natural d, reprezentînd numărul de poziţii cu care s-a realizat permutarea circulară spre dreapta. Dacă pentru d există mai multe posibilităţi se va alege valoarea minimă. Pe următoarea linie este descrisă permutarea p. Mai exact se vor scrie valorile $p(1), p(2), ..., p(m)$ separate prin câte un spaţiu.

Restricţii şi precizări

$n \leq 100000$

$m \leq 9999$

Mesajul conţine fiecare număr natural din intervalul $[1, m]$ cel puţin o dată.

Pentru teste cu $m \leq 5$ se acordă 40 de puncte din care 20 pentru teste şi cu $n \leq 2000$.

Exemplu

cifru.in	cifru.out
6 3 2 1 3 3 2 1 1 3 1 3 2 2	2 3 1 2

Timp maxim de execuţie/test: 0.2 secunde

7.2.1 Indicaţii de rezolvare - descriere soluţie *

Fiecare apariţie a unui simbol din alfabet într-un şir se înlocuieşte cu distanţa faţă de precedenta apariţie a aceluiaşi simbol (considerând şirul circular, deci pentru prima apariţie a simbolului se ia distanţa faţă de ultima apariţie a aceluiaşi simbol).

Făcând această recodificare pentru cele două şiruri reducem problema la determinarea permutării circulare care duce primul şir în al doilea, care poate fi

rezolvată cu un algoritm de pattern matching, dacă concatenăm primul şir cu el însuşi rezultând o complexitate $O(n)$.

Pentru m mic se pot genera toate permutările mulţimii $\{1, 2, ..., m\}$ făcând pentru fiecare permutare o căutare (cu KMP de exemplu), iar pentru n mic se poate căuta permutarea pentru fiecare $d = 0, 1, ..., n$.

7.2.2 Rezolvare detaliată

7.2.3 Codul sursă *

```
import java.io.*;
class kmp
{
  static int[] t0; // text mesaj necodificat --> spatiu ... de eliberat !
  static int[] t1; // text mesaj codificat --> spatiu ... de eliberat !

  static int[] d0; // distante ... mesaj necodificat
  static int[] d1; // distante ... mesaj codificat

  static int[] t;  // text in KMP ... (d0,d0) ... d0 dublat ... spatiu !!!
  static int[] s;  // sablon in KMP ... (d1)
  static int[] p; // prefix in KMP ... 1,2,...n

  static int[] ua; // pozitia ultimei aparitii ... 1,2,...,m ... ==> d[] mai rapid ...
  static int[] perm;// permutarea

  static int n,m; // ... n=100.000, m=9.999 ... maxim !!! ==> 200K

  public static void main(String[] args) throws IOException
  {
    StreamTokenizer st=new StreamTokenizer(
      new BufferedReader(new FileReader("9-cifru.in")));
    PrintWriter out=new PrintWriter(
      new BufferedWriter(new FileWriter("cifru.out")));

    int i,j,j0,j1,k,deplasarea=-1;

    st.nextToken(); n=(int)st.nval;
    st.nextToken(); m=(int)st.nval;
```

```
ua=new int[m+1];

t0=new int[n+1];
t1=new int[n+1];
d0=new int[n+1];
d1=new int[n+1];
p=new int[n+1];

for(i=1;i<=n;i++) { st.nextToken(); t0[i]=(int)st.nval; }
for(i=1;i<=n;i++) { st.nextToken(); t1[i]=(int)st.nval; }
distanta(t0,d0);
distanta(t1,d1);
//afisv(t0,1,n); afisv(d0,1,n); System.out.println();
//afisv(t1,1,n); afisv(d1,1,n); System.out.println();

s=d0;
prefix(s,p,n);
//afisv(s,1,n); afisv(p,1,n); System.out.println();

t=new int[2*n+1]; // ocupa spatiu prea mult; aici standard dar ...
for(i=1;i<=n;i++) t[i]=t[n+i]=d1[i];
//afisv(t,1,2*n);

deplasarea=kmp(t,2*n,s,n)-1; // d1 dublat si caut d0 ...
out.println(deplasarea);
System.out.println(deplasarea);

// permutarea ...
perm=ua;            // economie de spatiu ...
for(i=1;i<=m;i++) perm[i]=0;
k=0;     // nr elemente plasate deja in permutare ...
j1=0;
for(i=1;i<=n;i++)
{
  j1++;
  j0=n-deplasarea+i;
  if(j0>n) j0=j0-n;
  //System.out.println(i+" : "+j0+" "+j1);
  if(perm[t0[j0]]==0)
  {
    perm[t0[j0]]=t1[j1];
    k++;
  }
  if(k==m) break;
```

```
    }
    //afisv(perm,1,m);

    for(i=1;i<=m;i++) out.print(perm[i]+" ");
    out.close();
  }// main

  static int kmp(int[] t, int n, int[] s, int m)// t1,...,tn si s1,...,sm
  {
    int k,i,pozi=-1;
    k=0;
    for (i=1;i<=n;i++)
    {
      while(k>0&&s[k+1]!=t[i]) k=p[k];
      if (s[k+1]==t[i]) k++;
      if(k==m)
      {
        pozi=i-m+1;
        //System.out.println("incepe pe pozitia "+pozi);
        break; // numai prima aparitie ... !!!
      }
    }// for
    return pozi;
  }// kmp(...)

  static void distanta(int[] t,int[] d) // t=text, d=distante ...
  {
    int i,j,k;
    for(i=1;i<=m;i++) ua[i]=0;

    for(i=1;i<=n;i++)
    {
      if(ua[t[i]]!=0) // stiu pozitia spre stanga a lui t[i] ...
      {
        if(ua[t[i]]<i)
          d[i]=i-ua[t[i]];  // e mai la stanga ...
        else
          d[i]=i-ua[t[i]]+n; // e mai la dreapta ...

        ua[t[i]]=i; // noua pozitie a lui t[i] ...
        continue;
      }

      // nu a aparut inca in 1..i-1 ==> de la n spre stanga
```

```
      k=i; // distanta spre stanga ... pana la n inclusiv ...
      j=n; // caut in zona n,n-1,n-2,...
      while(t[i]!=t[j])
      {
        k++;
        j--;
      }
      d[i]=k;
      ua[t[i]]=i;
    }// for i
  }// distanta(...)

  static void prefix(int[] s,int[] p,int m) // s=sablon, p=prefix, m=dimensiune
  {
    int i,k;
    p[1]=0;
    for(i=2;i<=m;i++)
    {
      k=p[i-1];
      while(k>0&&s[k+1]!=s[i]) k=p[k];
      if(s[k+1]==s[i]) p[i]=k+1; else p[i]=0;
    }
  }// prefix()

  static void afisv(int[] x, int i1, int i2)
  {
    int i;
    for(i=i1;i<=i2;i++) System.out.print(x[i]+" ");
    System.out.println();
  }// afisv(...)
}// class
```

7.3 Evoluţie

Este binecunoscut faptul că informaţia genetică a unui organism poate fi codificată sub forma unui şir format din simboluri din mulţimea g, a, t, c. Pornind de la această codificare biologii au identificat 3 operaţii asupra şirurilor de simboluri, operaţii care pot modela evoluţia anumitor organisme.

1. **Complementaritate**. Simbolul a este complementarul lui t (şi reciproc), iar simbolul c este complementarul lui g (şi reciproc). Pentru un simbol x vom nota cu $c(x)$ complementarul său. Prin extensie, dacă w este un şir de simboluri din mulţimea a, c, g, t notăm cu $c(w)$ şirul obţinut prin complementarea simbolurilor lui w. De exemplu, pentru $w = aaactg$, avem $c(w) = tttgac$.

2. **Oglindire**. Vom nota prin w^R şirul obţinut prin oglindirea lui w. De exemplu pentru $w = aaagatat$, $w^R = tatagaaa$.

3. **Hairpin**. Pentru un şir de simboluri w, care poate fi descompus î patru subşiruri $w_1w_2w_3w_4$ (unele dintre cele patru şiruri pot fi vide) prin operaţia hairpin se obţine: $w_1w_2w_3w_4c(w_1)^R$, dacă $w_2 = c(w_4)^R$ şi lungimea lui w_2 este mai mare sau egală cu 1, sau $c(w_4)^Rw_1w_2w_3w_4$, dacă $w_1 = c(w_3)^R$ şi lungimea lui w_1 este mai mare sau egală cu 1.

Dacă ambele condiţii sunt verificate, oricare dintre cele două şiruri se poate obţine.

În grădina *Acolor* a fost descoperită o specie de omizi cu simţ artistic. Informaţia genetică a omizilor este codificată printr-o mulţime S formată din n şiruri de simboluri din mulţimea $\{a, c, g, t\}$. Mulţimea S este denumită mulţime iniţială. În evoluţia omizilor, informaţia genetică iniţială a suferit o serie de modificări. Pentru omizi, toate aceste modificări pot fi descrise prin aplicarea operaăiei hairpin de un număr arbitrar de ori asupra şirurilor din mulăimea iniţială S.

Cerinţă

Date fiind cele n şiruri din mulţimea iniţială S şi o succesiune de m şiruri de simboluri, să se decidă care dintre cele m şiruri poate reprezenta codul genetic al unei omizi, cod obţinut prin aplicarea unor operaţii hairpin.

Date de intrare

Fişierul de intrare **evo.in** conţine $n + m + 2$ linii.

Pe prima linie este scris numărul natural n reprezentând numărul de şiruri din mulţimea iniţială S. Urmează n linii, pe fiecare linie fiind scris un şir din mulţimea S.

Pe linia $n + 2$ este scris numărul natural m, reprezentând numărul de şiruri care trebuie să fie analizate. Pe următoarele m linii sunt scrise cele m şiruri care trebuie analizate, câte un şir pe o linie.

Date de ieşire

Fişierul de ieşire **evo.out** va conţine m linii, câte una pentru fiecare şir de analizat. Pe linia i se va scrie cuvântul **da**, dacă al i-lea şir dintre cele m şiruri de analizat poate fi codul genetic al unei omizi, respectiv cuvântul **nu**, în caz contrar.

Restricţii şi precizări

- $0 < n < 5, 0 < m < 1001$
- Lungimea unui şir din mulţimea iniţială S este mai mică decât 101.
- Lungimea totală a şirurilor de analizat este mai mică decât 16001. Lungimea fiecărui şir de analizat este mai mică decât 4001.
- În 55% din teste lungimea maximă a unui şir de analizat este 700.
- Memoria totală disponibilă este de 18MB din care 1MB pentru stivă.

Exemplu

evo.in	evo.out
2	da
acgtcg	nu
gaaaat	da
4	da
gaaaat	
gaaaatcc	
gaaaattc	
cgacgtcgtcg	

Explicaţie:

Primul şir de analizat este gaaaat. Acesta poate fi obţinut din gaaaat fără a aplica vreo operaţie hairpin.

Al doilea şir de analizat este gaaattc. Acesta nu poate fi obţinut prin aplicarea operaţiei hairpin asupra şirurilor acgtcg sau gaaaat.

Al treilea şir de analizat este gaaaattc. Acesta se obţine aplicând operaţia hairpin o singură dată asupra şirului gaaaat (considerând $w_1 = ga$, $w_2 = a$, $w_3 = aa$, $w_4 = t$).

Al patrulea şir de analizat este cgacgtcgtcg. Acesta se poate obţine din acgtcg aplicând de două ori operaţia hairpin (din acgtcg obţinem cgacgtcg pentru $w_1 = ac$, $w_2 =$şir vid, $w_3 = gt$, $w_4 = cg$, operaţia de hairpin adăugând $c(w_4)^R$ la începutul şirului, şi apoi din cgacgtcg obţinem cgacgtcgtcg pentru $w_1 = cga$, $w_2 = c$, $w_3 = gtc$ şi $w_4 = g$, operaţia de hairpin adăugând $c(w_1)^R$ la sfârşitul şirului.

Timp maxim de execuţie/test: 3 secunde

7.3.1 Indicaţii de rezolvare - descriere soluţie

7.3.2 Rezolvare detaliată

7.3.3 Codul sursă

7.4 Partiţie

Ionică a primit de ziua lui de la tatăl său un joc format din piese de formă de triunghiulară de dimensiuni diferite şi o suprafaţă magnetică pe care acestea pot fi aşezate. Pe suprafaţa magnetică este desenat un triunghi dreptunghic cu lungimile catetelor a, respectiv b şi un sistem de coordonate xOy cu originea în unghiul drept al triunghiului, semiaxa $[Ox$ pe cateta de lungime a, respectiv semiaxa $[Oy$ pe

cateta de lungime b. La un moment dat Ionică aşează pe tabla magnetică n piese, pentru care se cunosc coordonatele vârfurilor lor. Tatăl lui Ionică vrea să verifice dacă pe tablă piesele realizează o partiţie a triunghiului dreptunghic desenat, adică dacă sunt îndeplinite condiţiile:

- nu există piese suprapuse;
- piesele acoperă toată porţiunea desenată (în formă de triunghi dreptunghic);
- nu există porţiuni din piese în afara triunghiului desenat.

Cerinţă

Se cere să se verifice dacă piesele plasate pe tabla magnetică formează o partiţie a triunghiului desenat pe tabla magnetică.

Date de intrare

Fişierul de intrare **part.in** conţine pe prima linie un număr natural k, reprezentând numărul de seturi de date din fişier. Urmează k grupe de linii, câte o grupă pentru fiecare set de date. Grupa de linii corespunzătoare unui set este formată dintr-o linie cu numerele a, b, n separate între ele prin câte un spaţiu şi n linii cu câte şase numere întregi separate prin spaţii reprezentând coordonatele vârfurilor (abscisă ordonată) celor n piese, câte o piesă pe o linie.

Date de ieşire

În fişierul **part.out** se vor scrie k linii, câte o linie pentru fiecare set de date. Pe linia i ($i = 1, 2, ..,$) se va scrie 1 dacă triunghiurile din setul de date i formează o partiţie a triunghiului desenat pe tabla magnetică sau 0 în caz contrar.

Restricţii şi precizări

$1 \leq n \leq 150$

$1 \leq k \leq 10$

a, b sunt numere ntregi din intervalul $[0, 31000]$

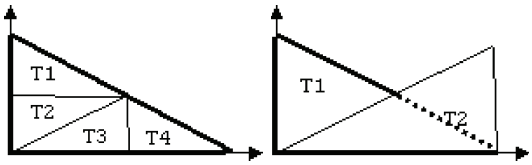
Coordonatele vârfurilor pieselor sunt numere întregi din intervalul $[0, 31000]$.

Exemplu

part.in	**part.out**
2	1
20 10 4	0
0 5 0 10 10 5	
0 0 10 5 0 5	
0 0 10 0 10 5	
10 0 20 0 10 5	
20 10 2	
0 0 0 10 10 5	
0 0 20 0 20 10	

Timp maxim de execuţie: 0.3 secunde/test

7.4.1 Indicaţii de rezolvare - descriere soluţie



7.4.2 Rezolvare detaliată

7.4.3 Codul sursă

7.5 Platou

Se consideră un şir de 1048576 (2^{20}) elemente care sunt numere întregi. Se definesc noţiunile:

- platou ca fiind o secvenţă de elemente egale aflate în vector în poziţii consecutive;

- lungimea unui platou ca fiind numărul de elemente care alcătuiesc un platoul.

Se ştie că în şirul considerat, lungimea maximă a unui platou este 8192 (2^{13}) şi că există cel puţin un platou care are această lungime. Amplasarea unui platou este determinată de poziţia cea mai mică şi de poziăia cea mai mare dintre poziţiile elementelor care alcătuiesc platoul.

Cerinţă

Să se scrie un program care determină amplasarea unui platou de lungime 8192 în şirul considerat. Dumneavoastră nu cunoaşteşi şirul. Determinarea amplasării platoului se face adresând întrebări comisiei. O întrebare constă în a preciza două poziţii din şir, fie acestea p_1 şi p_2. Comisia vă dă un răspuns care reprezintă lungimea cea mai mare a unui platou din şir aflat între poziţiile p_1 şi p_2.

Intercţiune

Programul vostru nu va efectua operaţii cu nici un fişier. El va interacţiona cu un program al comisiei care rulează în paralel. După lansarea în execuţie, programul vostru trebuie procedeze astfel:

- programul poate pune întrebări programului comisiei. Formatul unei întrebări este :

ask p_1 p_2

unde p_1 p_2 reprezintă poziţiile pentru care faceţi interogarea; după ce a afişat întrebarea la ieşirea standard, programul va citi de la intrarea standard o valoare care reprezintă lungimea cea mai mare a unui platou aflat între poziţiile p_1 ?i p_2.

- după ce programul a terminat de pus întrebări, el va afişa o linie de forma:

done p_1 p_2

unde p_1 şi p_2 reprezintă poziţiile ce determină amplasarea unui platou de lungime 8192 din şir.

Instrucţiuni de programare

După fiecare linie complet scrisă la ieşirea standard programatorii în C trebuie să apeleze funcţia fflush(stdout), cei în C++ să apeleze cout.flush(), iar cei din Pascal procedura flush(output).

Spre exemplu,

C	C++	Pascal
`printf("ask 1 3\n");` `fflush(stdout);`	`cout<<"ask 1 3" <<endl;` `cout.flush();`	`writeln('ask 1 3');` `flush(output);`

Exemplu de interacţiune

Interogare	ask 1 8192	Interoghez lungimea maximă a unui platou întâlnită între poziţiile 1 şi 8192
Citesc răspunsul la prima interogare	8192	Această lungime este 8192
Scriu că am terminat	done 1 8192	Un platou cu lungime de 8192 se află între poziţile 1 şi 8192

Restricţii şi precizări

Poziţiile din şir sunt numerotate de la 1 la 1048576.

Punctaj

Programul vostru obţine 0 puncte la un test dacă:

- nu respectă modul de interacţiune cu programul comisiei;
- utilizează mai mult de 20 interogări;
- interoghează cu poziţii incorecte: poziţii mai mici decât 1 sau mai mari decât 1048576 sau prima poziţie e mai mare strict decât a doua poziţie.

Altfel, programul vostru obţine un punctaj care depinde de numărul de întrebări după cum urmează:

- pentru cel mult 11 întrebări obţine 100% din punctaj;
- pentru un număr de întrebări cuprins între 12 şi 15 obţine 50% din punctaj;
- pentru un număr de întrebări cuprins între 16 şi 20 obţine 25% din punctaj.

Timp maxim de execuţie/test: 0.7 secunde

Observaţie: se garantează că programul comisiei răspunde pentru 20 de întrebări în 0,6 secunde; programul vostru are la dispoziţie 0,1 secunde.

7.5.1 Indicaţii de rezolvare - descriere soluţie

7.5.2 Rezolvare detaliată

7.5.3 Codul sursă

7.6 Trasee

Fiindcă nu m-am calificat la ONI, am vacanţă de primăvară. Am luat harta rutieră a ţării şi am marcat n oraşe pe care le consider interesante şi merită vizitate. Am numerotat cele n oraşe de la 1 la n.

Oraşul în care locuiesc este numerotat cu x.

Vacanţa nu este lungă, aşa că am hotărât că aş putea vizita exact k oraşe situate pe un traseu care respectă simultan următoarele condiţii (numim un astfel de traseu valid):

1. Traseul porneşte din oraşul în care locuiesc (oraşul x).

2. Între oricare două oraşe consecutive de pe traseu trebuie să existe un drum direct (drum care nu trece prin alte oraşe).

3. Traseul trece prin exact k oraşe distincte.

4. Pentru orice oraş y de pe traseu, distanţa (numărul de drumuri directe parcurse) pe traseul respectiv de la oraşul x la oraşul y (exprimată în număr de drumuri directe parcurse pe traseu) este minimă (adică nu există un alt traseu pe hartă de la x la y având lungime strict mai mică).

Două trasee valide sunt considerate disjuncte dacă ele nu conţin un acelaşi drum direct.

Cerinţă

Cum eu nu m-am calificat la ONI, scrieţi un program care să mă ajute să determin numărul maxim de trasee valide disjuncte.

Date de intrare

Fişierul de intrare **trasee.in** conţine pe prima linie patru numere naturale n, m, x şi k separate prin câte un spaţiu, care reprezintă numărul de oraşe, numărul

de drumuri directe dintre oraşe, numărul oraşului în care locuiesc şi respectiv numărul de oraşe aflate pe un traseu. Următoarele m linii conţin cele m drumuri directe de pe hartă, câte un drum pe o linie. Fiecare drum direct este specificat prin două numere naturale distincte cuprinse între 1 şi n, separate printr-un spaţiu, reprezentând oraşele conectate de drumul respectiv.

Date de ieşire

Fişierul de ieşire **trasee.out** va conţine o singură linie pe care va fi scris numărul maxim de trasee valide disjuncte.

Restricţii şi precizări

$1 \leq n \leq 200$

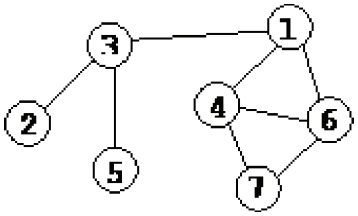
$1 \leq x \leq n$

$1 \leq k \leq n$

Între două oraşe poate exista cel mult un drum direct. Drumurile sunt bidirecţionale.

Exemplu

trasee.in	**trasee.out**
7 8 1 3 1 3 2 3 5 3 4 6 4 7 1 4 6 1 6 7	3



Explicaţie:

Oraşul de plecare este oraşul 1. Există patru trasee valide care trec prin exact 3 oraşe, plecând din 1 (1-3-2, 1-3-5, 1-4-7, 1-6-7). Numărul maxim de trasee valide disjuncte este 3. O soluţie ar putea fi 1-3-2, 1-4-7, 1-6-7.

Timp maxim de execuţie/test: 0.2 secunde

7.6.1 Indicaţii de rezolvare - descriere soluţie

7.6.2 Rezolvare detaliată

7.6.3 Codul sursă

Capitolul 8

Baraj 2007

8.1 Dist

Să considerăm două propoziţii formate din cuvinte scrise cu litere mari ale alfabetului englez, oricare două cuvinte consecutive fiind separate de unul sau mai multe spaţii.

Să considerăm $c = c_1c_2...c_n$ şi $d = d_1d_2...d_m$ două cuvinte.

Pentru a calcula distanţa dintre cuvintele c şi d, notată $dist(c, d)$, cuvântul mai scurt se completează la sfârşit cu caracterul '@' (care are codul ASCII 64), până se obţin două cuvinte de aceeaşi lungime, apoi se calculează suma diferenţelor absolute dintre codurile ASCII ale caracterelor situate în cuvintele c şi d pe poziţii corespondente:

$$dist(c, d) = |c_1 - d_1| + |c_2 - d_2| + ... + |c_{lg} - d_{lg}|, \text{ unde } lg = \max\{n, m\}.$$

Definim distanţa dintre două propoziţii ca fiind suma distanţelor dintre cuvintele situate în propoziţii pe poziţii corespondente. Dacă una dintre propoziţii are mai

puţine cuvinte decât cealaltă se consideră că la sfârşitul acestei propoziţii se află cuvinte vide (cuvinte de lungime 0), până la completarea numărului necesar de cuvinte.

De exemplu, să considerăm propoziţia P1="ANA ARE MERE" şi propoziţia P2="VASILE NU". Distanţa dintre propoziţia P1 şi propoziţia P2 este:

$dist(P1, P2) =$
$dist("ANA","VASILE") + dist("ARE","NU") + dist("MERE","").$

$dist("ANA","VASILE") = |'A' - 'V'| + |'N' - 'A'| + |'A' - 'S'| + |'@' - 'I'| + |'@' - 'L'| + |'@' - 'E'| = |65 - 86| + |78 - 65| + |65 - 83| + |64 - 73| + |64 - 76| + |64 - 69| = 21 + 13 + 18 + 9 + 5 = 66$

$dist("ARE","NU") = |'A' - 'N'| + |'R' - 'U'| + |'E' - '@'| = |65 - 78| + |82 - 85| + |69 - 64| = 13 + 3 + 5 = 21$

$dist("MERE","") = |'M' - '@'| + |'E' - '@'| + |'R' - '@'| + |'E' - '@'| = |77 - 64| + |69 - 64| + |82 - 64| + |69 - 64| = 13 + 5 + 18 + 5 = 41.$

Deci $dist(P1, P2) = 66 + 21 + 41 = 128$.

În scopul de a minimiza distanţa dintre cele două propoziţii, asupra celei de a doua propoziţii putem executa una sau mai multe operaţii. O operaţie constă în a muta prima literă dintr-un cuvânt la sfârşitul cuvântului precedent (dacă acesta există) sau ultima literă dintr-un cuvânt la începutul cuvântului următor. Cuvinte vide se pot afla doar la sfârşitul unei propoziţii, nu şi la începutul sau în interiorul ei (nici în propoziţiile date, nici în propoziţiile obţinute în urma aplicării operaţiilor). Cuvintele din propoziţie şi cuvintele obţinute în urma operaţiilor nu pot să depăşească 20 de litere.

Cerinţă

Să se determine distanţa minimă care se poate obţine între cele două propoziţii efectuând operaţii de tipul celor descrise în enunţ. Se cere de asemenea să se determine şi numărul minim de operaţii ce trebuie să fie executate asupra celei de a doua propoziţii pentru a obţine distanţa minimă.

Date de intrare

Fişierul de intrare **dist.in** conţine pe prima linie prima propoziţie, iar pe cea de a doua linie a doua propoziţie.

Date de ieşire

Fişierul de ieşire **dist.out** va conţine o singură linie pe care vor fi scrise două numere naturale separate prin spaţiu *dmin nrmin*, reprezentând în ordine distanţa minimă dintre cele două propoziţii, respectiv numărul minim de operaţii ce trebuie să fie executate asupra celei de a doua propoziţii pentru a obţine distanţa minimă.

Restricţii şi precizări

Lungimea totală a unei propoziţii nu depăşeşte 500 caractere.

Lungimea maximă a unui cuvânt nu depăşeşte nici în propoziţiile date, nici în propoziţia obţinută în urma aplicării operaţiilor din enunţ 20 de caractere.

Numărul maxim de cuvinte dintr-o propoziţie este 100.

Se acordă 60% din punctaj pentru determinarea distanţei minime şi 100% pentru rezolvarea ambelor cerinţe.

Exemplu

dist.in	**dist.out**	**Explicaţie**
ANA ARE MERE VASILE NU	62 9	Propoziţia a doua, după aplicarea celor 9 operaţii, este: V ASI LENU

Memorie totală disponibilă: 2 Mb din care 1 Mb pentru stivă.

Timp maxim de execuţie/test: 0.1 secunde

8.1.1 Indicaţii de rezolvare - descriere soluţie

8.1.2 Rezolvare detaliată

8.1.3 Codul sursă

8.2 Promo

Compania ONIx comercializează N produse. Pentru a creşte vânzările, compania a pus la dispoziţia clienţilor M oferte promoţionale. Fiecare ofertă constă din exact 2 produse diferite, care sunt vândute împreună la un preţ mai scăzut decât dacă ar fi vândute separat (de exemplu, suc şi apă minerală).

Produsele sunt identificate prin numere de la 1 la N, iar ofertele promoţionale prin numere de la 1 la M.

Deoarece şi-au schimbat de curând aplicaţia software ce gestionează baza de date a companiei, angajaţii nu s-au obişnuit cu noul sistem şi, din neatenţie, unul dintre aceştia a şters toate informaţiile despre produsele şi ofertele existente. Singurele informaţii rămase sunt cele ale departamentului de statistică, care foloseşte o bază de date proprie. Aceste informaţii sunt reprezentate de numărul M de oferte şi de toate cele K perechi de oferte ce au un produs în comun (în mod evident, oricare 2 oferte pot avea cel mult un produs în comun).

Cerinţă

Folosind informaţiile departamentului de statistică, determinaţi numărul de produse şi cele 2 produse din cadrul fiecărei oferte.

Date de intrare

Prima linie a fişierului de intrare **promo.in** conţine numerele întregi M şi K, separate printr-un spaţiu. Următoarele K linii conţin câte 2 numere întregi A şi B, separate printr-un spaţiu, având semnificaţia că oferta cu numărul A şi cea cu numărul B au un produs în comun.

Date de ieşire

Pe prima linie a fişierului de ieşire **promo.out** veţi afişa numărul întreg N, reprezentând numărul de produse. Următoarele M linii trebuie să conţină câte 2 numere întregi, separate printr-un spaţiu. A i-a linie dintre aceste M linii va conţine numerele produselor din care este formată a i-a ofertă.

Restricţii şi precizări

- $1 \leq M \leq 2007$
- $0 \leq K \leq 100000$
- Numărul de produse determinat trebuie să fie cel mult egal cu $2 * M$.
- Se garantează existenţa cel puţin a unei soluţii. Dacă există mai multe soluţii, puteţi afişa oricare dintre ele.

Exemplu

promo.in	promo.out
11 7	17
1 4	1 2
4 7	3 4
7 1	5 6
2 5	1 7
8 2	9 10
10 11	1 11
	3 12
	13 14
	15 16
	15 17

Memorie totală disponibilă: 16 Mb din care 1 Mb pentru stivă.

Timp maxim de execuţie/test: 0.4 secunde

8.2.1 Indicaţii de rezolvare - descriere soluţie

8.2.2 Rezolvare detaliată

8.2.3 Codul sursă

8.3 Puncte

Zăhărel a desenat pe o foaie de hârtie N puncte în plan. Curios din fire, şi-a ales încă M puncte pe axa Ox şi s-a întrebat pentru fiecare dintre cele M puncte de pe axa Ox care dintre cele N puncte este cel mai apropiat (situat la distanţă minimă). Se consideră că distanţa dintre două puncte (x_1, y_1) şi (x_2, y_2) este $(x_1 - x_2)^2 + (y_1 - y_2)^2$.

Cerinţă

Scrieţi un program pentru Zăhărel care să determine pentru fiecare dintre cele M puncte de pe axa Ox, care este distanţa la cel mai apropiat punct dintre cele N desenate pe hârtie.

Date de intrare

Fişierul de intrare **puncte.in** conţine pe prima linie numerele naturale N, M separate prin spaţii.

Fiecare dintre următoarele N linii conţine câte o pereche de numere naturale nenule x y, separate prin spaţii, reprezentând coordonatele celor N puncte (în ordinea abscisă, ordonată).

Fiecare dintre următoarele M linii conţine câte un număr natural x, reprezentând abscisele (coordonatele pe axa Ox) ale celor M puncte.

Date de ieşire

Fişierul de ieşire **puncte.out** va conţine M linii. Pe linia i va fi scris un număr natural reprezentând distanţa la cel mai apropiat punct dintre cele N de pe hârtie pentru al i-lea punct de pe axa Ox (considerând ordinea punctelor din fişierul de intrare).

Restricţii şi precizări

- $1 \leq N \leq 100000$
- $1 \leq M \leq 200000$
- Toate coordonatele din fişierul de intrare sunt numere naturale din intervalul $[1, 109]$
- Cele N puncte din fişierul de intrare sunt sortate după coordonata x crescător, iar în cazul în care două puncte au aceeaşi abscisă, ele sunt ordonate crescător după coordonata y.

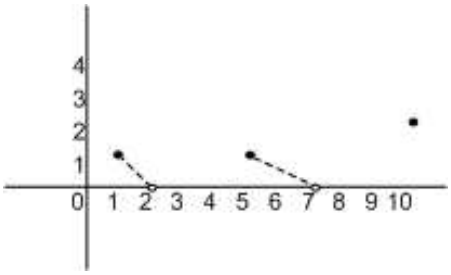
• Pentru 50% din teste $N \geq 90000$ şi $M \geq 150000$.

Exemplu

puncte.in	**puncte.out**
3 2 1 1 5 1 10 2 2 7	2 5

Explicaţie

Pe hârtie au fost desenate 3 puncte, având coordonatele $(1, 1)$, $(5, 1)$, respectiv $(10, 2)$. Pe axa Ox se află 2 puncte, având abscisa 2, respectiv 7.



Distanţa minimă dintre punctul de pe axa Ox de abscisă 2 este 2 (cel mai apropiat punct fiind cel de coordonate $(1, 1)$).

Distanţa minimă dintre punctul de pe axa Ox de abscisă 7 este 5 (cel mai apropiat punct fiind cel de coordonate $(5, 1)$).

Memorie totală disponibilă: 5 Mb din care 1 Mb pentru stivă.

Timp maxim de execuţie/test: 0.4 secunde

8.3.1 Indicaţii de rezolvare - descriere soluţie

8.3.2 Rezolvare detaliată

8.3.3 Codul sursă

8.4 Cover

Se consideră N intervale închise, având extremităţile numere naturale cuprinse între 1 şi L.

Fiecare număr natural i din intervalul $[1, L]$ are asociată o pondere c_i.

Numim acoperire o mulţime de numere naturale cuprinse între 1 şi L cu proprietatea că fiecare interval conţine cel puţin un element al mulţimii.

Costul unei acoperiri este egal cu suma ponderilor numerelor din acoperire.

Cerinţă

Pentru un set de intervale dat să se determine costul minim al unei acoperiri.

Date de intrare

Fişierul de intrare **cover.in** conţine pe prima linie cele două numere naturale N L separate printr-un spaţiu. Pe următoarea linie se află L numere naturale separate prin câte un spaţiu c_1 c_2 ... c_L reprezentând ponderile numerelor naturale din intervalul $[1, L]$. Următoarele N linii conţin fiecare câte două numere naturale a b ($1 \leq a \leq b \leq L$) reprezentând extremităţile intervalelor.

Date de ieşire

Fişierul de ieşire **cover.out** va conţine o singură linie pe care va fi scris un număr natural reprezentând costul minim al unei acoperiri.

Restricţii şi precizări

- $1 \leq N \leq 60000$
- $1 \leq L \leq 1000000$
- $0 \leq c_i \leq 1024$, pentru orice $1 \leq i \leq L$
- Pentru 40% din teste $N \leq 1000$ şi $L \leq 10000$.

Exemplu

cover.in	**cover.out**
2 5 100 5 9 6 90 1 3 3 5	9

Explicaţie

Se construieşte acoperirea $\{3\}$ care are costul 9. Elementul 3 aparţine ambelor intervale date în fişierul de intrare.

Există şi alte acoperiri posibile de exemplu $\{2, 4\}$ dar costul acesteia este 11 care nu este minim.

Exemplu

cover.in	**cover.out**
4 10 1 3 6 4 5 1 0 1 3 2 1 3 3 5 6 9 4 4	6

Explicaţie
Se construieşte acoperirea $\{1, 4, 7\}$ care are costul 5.

Memorie totală disponibilă: 32 Mb din care 1 Mb pentru stivă.

Timp maxim de execuţie/test: 0.5 secunde

8.4.1 Indicaţii de rezolvare - descriere soluţie

8.4.2 Rezolvare detaliată

8.4.3 Codul sursă

8.5 Munte

Gigel este un pasionat excursionist. Îi plac în special excursiile la munte. La sfârşitul acestei săptămâni el şi-a propus să traverseze un munte din apropierea oraşului Cluj. Atâta doar că echipa Salvamont locală i-a impus nişte condiţii:

- lungimea drumului trebuie să fie exact $2n - 2$ metri, valoarea n fiind dată de salvamontişti;
- trebuie să plece de la poalele muntelui şi trebuie să ajungă tot la poalele muntelui de partea cealaltă la aceeaşi altitudine;
- nu are voie să coboare sub altitudinea de plecare;
- poate traversa drumul doar folosind trei tipuri de paşi:
  - pas pe orizontală de lungime 2, deci de tipul $(2, 0)$
  - pas "în sus" de lungime 1, deci de tipul $(1, 1)$

– pas "în jos" de lungime 1, deci de tipul $(1, -1)$

- drumul lui nu are voie să aibă "vârf" la altitudinea 1, adică nu are voie ca fiind la un moment dat, pe parcursul drumului, la altitudinea de plecare, să facă un pas în sus urmat imediat de un pas în jos.

Cerinţă

Dată fiind valoarea n să se determine în câte moduri poate Gigel să traverseze muntele respectând condiţiile echipei Salvamont.

Date de intrare

Fişierul de intrare **munte.in** conţine o singură linie pe care se află num?rul natural n.

Date de ieşire

Fişierul de ieşire **munte.out** va conţine o singură linie pe care va fi scris numărul de modalităţi în care Gigel poate realiza traversarea muntelui.

Restricţii şi precizări

$1 \leq n \leq 100$

Pentru 60% din teste rezultatul este un întreg pe 64 de biţi.

Exemplu

munte.in	munte.out
1	1

Explicaţie

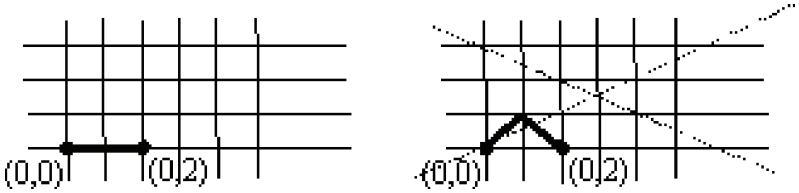
Lungimea drumului fiind $2 * 1 - 2 = 0$, există o singură modalitate de a traversa muntele (aceea de a sta pe loc)

Exemplu

munte.in	munte.out
2	1

Explicaţie

Lungimea drumului fiind $2 * 2 - 2 = 2$, există o singură modalitate de a traversa muntele printr-un pas de lungime 2.



Varianta din dreapta nu este corectă deoarece nu respectă ultima condiţie

Exemplu

munte.in	munte.out
3	3

Explicaţie

Cele 3 modalităţi corecte de a traversa muntele cu un drum de lungime 4 sunt:

Orice alt mod de a traversa muntele pe un drum de lungime 4 este incorect.

Memorie totală disponibilă: 16 Mb din care 1 Mb pentru stivă.

Timp maxim de execuţie/test: 0.1 secunde

8.5.1 Indicaţii de rezolvare - descriere soluţie

8.5.2 Rezolvare detaliată

8.5.3 Codul sursă

8.6 Role

Gigel este mare fan al muzicii anilor 70. El are o colecţie impresionantă de benzi magnetice cu melodiile compuse în acea perioadă. Tehnica folosită în acea perioadă poate părea rudimentară în zilele noastre, însă Gigel doreşte să reconstituie cât mai mult din atmosfera epocii.

Fiecare melodie este înregistrată pe o bandă magnetică; benzile sunt numerotate de la 1 la N. Fiecare bandă este înfăşurată pe câte o rolă din plastic, iar Gigel mai dispune de o rolă goală.

Rolele sunt numerotate de la 0 la N, fiecare bandă avându-şi locul pe rola cu acelaşi număr, iar rola 0 fiind rola goală.

Când Gigel ascultă o melodie, magnetofonul derulează banda de pe rola ei şi o înfăşoară pe rola goală; la final banda se găseşte pe rola ce iniţial era goală, bobinată invers, adică cu începutul benzii în centrul rolei, iar rola pe care se găsea iniţial banda devine goală.

Gigel are întotdeauna grijă ca, după ce ascultă o melodie, să rebobineze banda înapoi pe rola ei.

Fratele mai mic al lui Gigel este mai dezordonat şi lasă adesea banda pe rola pe care ajunge în urma ascultării. El foloseşte apoi rola proaspăt eliberată pe post de rolă goală pentru a asculta următoarea melodie. Astfel, după o vreme, multe dintre benzi se găsesc pe altă rolă decât cea proprie, şi unele benzi sunt bobinate invers, adică inceputul melodiei este în interiorul înfăşurării (trebuind derulate înainte de-a putea fi ascultate).

Gigel doreşte acum să restabilească ordinea, aducând fiecare bandă pe rola ei şi bobinată cu începutul în exterior. În acest scop el poate executa o secvenţă de operaţii. La o operaţie el poate doar să ia o rolă şi să deruleze banda de pe ea pe singura rolă ce este goală în acel moment, banda inversându-şi cu această ocazie sensul de bobinare.

Cerinţă

Se cere să se determine o secvenţă minimă de operaţii în urma cărora fiecare bandă ajunge pe rola cu numărul egal cu numărul benzii şi înfăşurată cu începutul în exterior.

Date de intrare

Fişierul de intrare **role.in** va conţine pe prima linie un număr natural N reprezentând numărul de benzi. Pe următoarele N linii se găsesc câte două numere naturale separate printr-un spaţiu, R şi I. A k-a pereche descrie poziţia benzii cu numărul k: R reprezintă numărul rolei pe care se află banda k, iar I are valoarea 0 dacă banda este înfăşurată normal (cu începutul în exterior) şi 1 dacă este înfăşurată invers.

Date de ieşire

Fişierul de ieşire **role.out** va conţine o secvenţă de numere naturale, câte un număr pe o linie. Numărul de pe linia i reprezentă numărul rolei de pe care se mută banda pe rola goală la cea de a i-a operaţie executată.

Restricţii şi precizări

$1 \leq N \leq 100000$

Pe o rolă poate exista cel mult o bandă la un moment dat.

Pentru toate datele de test va exista o soluţie care necesită cel puţin o operaţie.

Punctaj

Pentru soluţie optimă se acordă 100% din punctaj.

Dacă soluţia nu este optimă, dar este corectă, se vor acorda punctaje parţiale după cum urmează:

1. dacă diferenţa dintre numărul de operaţii executate de concurent şi numărul optim de operaţii este mai mică sau egală decât 10% din numărul optim (parte întreagă), se acordă 60% din punctaj;

2. dacă diferenţa dintre numărul de operaţii executate de concurent şi numărul optim de operaţii este mai mare decât 10% din numărul optim (parte întreagă), se acordă 20% din punctaj.

Exemple

role.in	role.out	role.in	role.out
2 1 0 0 1	0	3 2 0 0 1 3 1	3 2 1 3 2 0

Memorie totală disponibilă: 16 Mb din care 1 Mb pentru stivă.

Timp maxim de execuţie/test: 0.3 secunde

8.6.1 Indicaţii de rezolvare - descriere soluţie

8.6.2 Rezolvare detaliată

8.6.3 Codul sursă