ZIUA 1 CLASA a X-a

asediu Fişier sursă: asediu.pas, asediu.c, asediu.cpp

Se aproximează o zonă de război sub forma unui cerc pe circumferința căruia se stabilesc N puncte reprezentând comandamentele trupelor aliate cu proprietatea că nu există trei corzi cu capetele în aceste N puncte care să fie concurente într-un punct situat în interiorul cercului. Între oricare două puncte (comandamente) există un drum sigur de acces direct. Aceste drumuri împreună cu circumferința cercului delimitează un număr de regiuni distincte. Există informații că regiunile astfel delimitate reprezintă de fapt terenuri minate de combatanții inamici. Fiecare astfel de regiune va fi cercetată amănunțit de câte un soldat aliat echipat corespunzător cu detectoare de mine.
Cerință: Indicați numărul de soldați de care este nevoie pentru cercetarea tuturor regiunilor formate.

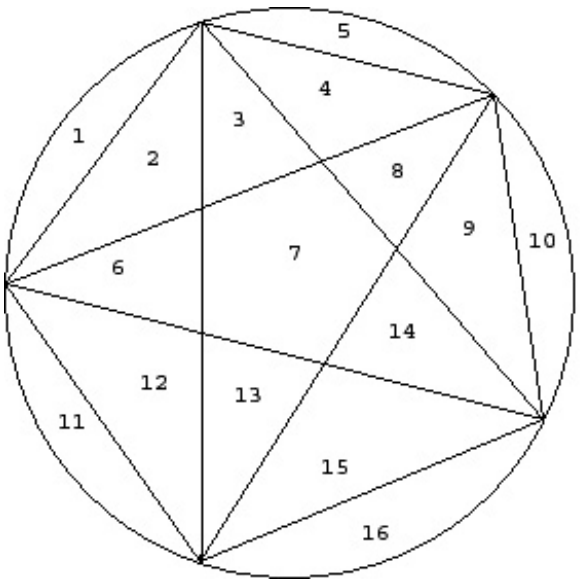
Date de intrare:
Fişierul de intrare **asediu.in**, conține:
Pe prima linie N numărul de comandamente

Date de ieşire:
Ieşirea se va face în fişierul **asediu.out** în formatul următor:
Pe prima linie se află numărul T de soldați necesari pentru cercetarea tuturor regiunilor formate.

Restricții:
$2 \leq N \leq 2.000.000$

Exemplu:

asediu.in	**asediu.out**
5	**16**



Timp maxim de executare: 1 secundă/test.

ZIUA 1 CLASA a X-a

muzeu Fişier sursă: muzeu.pas, muzeu.c, muzeu.cpp

Sunteţi un participant la Olimpiada Naţională de Informatică. În programul olimpiadei intră şi câteva activităţi de divertisment. Una dintre ele este vizitarea unui muzeu. Acesta are o structură de matrice dreptunghiulară cu M linii si N coloane; din orice cameră se poate ajunge în camerele vecine pe direcţiile nord, est, sud şi vest (dacă aceste camere există). Pentru poziţia (i,j) deplasarea spre nord presupune trecerea în poziţia (i-1,j), spre est în (i,j+1), spre sud în (i+1,j) şi spre vest în (i,j-1).

Acest muzeu are câteva reguli speciale. Fiecare cameră este marcată cu un număr între 0 şi 10 inclusiv. Mai multe camere pot fi marcate cu acelaşi număr. Camerele marcate cu numărul 0 pot fi vizitate gratuit. Într-o cameră marcată cu numărul i (i>0) se poate intra gratuit, dar nu se poate ieşi din ea decât dacă arătaţi supraveghetorului un bilet cu numărul i. Din fericire, orice cameră cu numărul i (i>0) oferă spre vânzare un bilet cu numărul i; o dată cumpărat acest bilet, el este valabil în toate camerele marcate cu numărul respectiv. Biletele pot avea preţuri diferite, dar un bilet cu numărul i va avea acelaşi preţ în toate camerele în care este oferit spre vânzare.

Dumneavoastră intraţi în muzeu prin colţul de Nord-Vest (poziţia (1,1) a matricei) şi doriţi să ajungeţi la ieşirea situată în colţul de Sud-Est (poziţia (M,N) a matricei). O dată ajuns acolo primiţi un bilet gratuit care vă permite să vizitaţi tot muzeul.

Poziţiile (1,1) şi (M,N) sunt marcate cu numărul 0.

Cerinţă

Cunoscându-se structura muzeului, determinaţi o strategie de parcurgere a camerelor, astfel încât să ajungeţi în camera (M,N) plătind cât mai puţin. Dacă există mai multe variante, alegeţi una în care este parcurs un număr minim de camere (pentru a câştiga timp şi pentru a avea mai mult timp pentru vizitarea integrală a muzeului).

Date de intrare

Prima linie a fişierului de intrare **`muzeu.in`** conţine două numere întregi **`M şi N`**, separate printr-un spaţiu, numărul de linii, respectiv de coloane, al matricei care reprezintă muzeul. Următoarele M linii conţin structura muzeului; fiecare conţine N numere întregi între 0 şi 10 inclusiv, separate prin spaţii. Linia M+2 conţine 10 numere întregi între 0 şi 10000 inclusiv, reprezentând costurile biletelor 1, 2, 3, ... 10 în această ordine.

Date de ieşire

În fişierul **`muzeu.out`** veţi afişa:

- pe prima linie suma minimă necesară pentru a ajunge din (1,1) în (M,N);
- pe a doua linie numărul minim de mutări L efectuate dintr-o cameră într-o cameră vecină, pentru a ajunge din (1,1) în (M,N);
- pe a treia linie L caractere din multimea N, E, S, V reprezentând deplasări spre Nord, Est, Sud sau Vest .

Restricţii şi precizări

- **`2 <= N <= 50`**

Exemplu :

`muzeu.in`
```
5 6
0 0 0 0 0 2
0 1 1 1 4 3
0 1 0 0 0 0
0 1 5 1 0 0
0 0 0 1 0 0
1000 5 7 100 12 1000 1000 1000 1000 1000
```

`muzeu.out`
```
12
9
EEEEESSSS
```

Timp maxim de executare: 1 secundă/test.

ZIUA 1 CLASA a X-a

munte Fişier sursă: munte.pas, munte.c, munte.cpp

Într-o zonă montană se doreşte deschiderea unui lanţ de telecabine. Staţiile de telecabine pot fi înfiinţate pe oricare din cele N vârfuri ale zonei montane. Vârfurile sunt date în ordine de la stânga la dreapta şi numerotate de la 1 la N, fiecare vârf i fiind precizat prin coordonata X[i] pe axa OX şi prin înălţimea H[i].

Se vor înfiinţa exact K staţii de telecabine. Staţia de telecabine i ($2 \le i \le K$) va fi conectată cu staţiile i-1 şi i+1; staţia 1 va fi conectată doar cu staţia 2, iar staţia K, doar cu staţia K-1. Staţia 1 va fi obligatoriu amplasată în vârful 1, iar staţia K în vârful N.

Se doreşte ca lanţul de telecabine să asigure legătura între vârful 1 şi vârful N. Mai mult, se doreşte ca lungimea totală a cablurilor folosite pentru conectare să fie minimă. Lungimea cablului folosit pentru a conecta două staţii este egală cu distanţa dintre ele. În plus, un cablu care uneşte două staţii consecutive nu poate avea lungimea mai mare decât o lungime fixată L.

O restricţie suplimentară este introdusă de formele de relief. Astfel, vârfurile i şi j ($i < j$) nu pot fi conectate direct dacă există un vârf v ($i < v < j$) astfel încât segmentul de dreapta care ar uni vârfurile i şi j nu ar trece pe deasupra vârfului v. În cazul în care cele trei vârfuri sunt coliniare, se consideră toate trei ca fiind staţii, chiar dacă distanţa dintre vârfurile i şi j este mai mică decât L.

Cerinţă Dându-se amplasarea celor N vârfuri ale lanţului muntos, stabiliţi o modalitate de dispunere a celor K staţii de telecabine astfel încât lungimea totală a cablurilor folosite pentru conectare să fie minimă, cu restricţiile de mai sus.

Se garantează că, pe toate testele date la evaluare, conectarea va fi posibilă.

Date de intrare

Prima linie a fişierului de intrare **munte.in** conţine trei numere întregi N, K şi L, separate prin spaţii, cu semnificaţiile de mai sus. Următoarele N linii conţin coordonatele vârfurilor; linia i+1 conţine coordonatele vârfului i, X[i] şi H[i], separate printr-un spaţiu.

Date de ieşire

În fişierul **munte.out** veţi afişa:

- pe prima linie lungimea totală minimă a cablurilor, rotunjită la cel mai apropiat numar întreg (pentru orice întreg Q, Q.5 se rotunjeste la Q+1);
- pe a doua linie K numere distincte între 1 şi N, ordonate crescător, numerele vârfurilor în care se vor înfiinţa staţii de telecabine. Dacă există mai multe variante, afişaţi una oarecare.

Restricţii şi precizări

- **2 <= N <= 100**
- **2 <= K <= 30 şi K <= N**
- **0 <= L, X[i], H[i] <= 100.000 şi X[i] < X[i+1]**

Exemplu

munte.in	munte.out
7 5 11 0 16 4 3 6 8 7 4 12 16 13 16 14 16	22 1 3 5 6 7

- trasarea unui cablu direct între vârfurile 1 şi 5 ar fi contravenit restricţiei referitoare la lungimea maximă a unui cablu; în plus, s-ar fi obţinut o soluţie cu 2 staţii de telecabine în loc de 3 (deci soluţia ar fi invalidă şi pentru valori mari ale lui L);
- pentru a ilustra restricţia introdusă de formele de relief, precizăm că vârfurile 1 şi 4 nu au putut fi conectate direct datorită înălţimii vârfului 3. De asemenea, vârfurile 5 şi 7 nu au putut fi conectate direct datorită înălţimii vârfului 6.

Timp maxim de executare: 1 secundă/test.

Observaţii

ZIUA 2 CLASA a X-a

Partitie Fişier sursă: partitie.pas, partitie.c, partitie .cpp

Se defineşte o partiție a unui număr natural n ca fiind o scriere a lui n sub forma:

$n = n_1 + n_2 + \dots n_k$, ($k \geq 1$)

unde $n_1, n_2, \dots n_k$ sunt numere naturale care verifică următoarea relație :

$n_1 \geq n_2 \geq \dots \geq n_i \geq \dots\dots \geq n_k \geq 1$

Cerință: Fiind dat un număr natural n, să se determine câte partiții ale lui se pot scrie, conform cerințelor de mai sus, ştiind că oricare număr n_i dintr-o partiție trebuie să fie un număr impar.

Date de intrare
Fişierul **partitie.in** conține pe prima linie numărul n

Date de ieşire
Fişierul **partitie.out** va conține pe prima linie numărul de partiții ale lui n conform cerințelor problemei.

Exemplu :
partitie.in
7
partitie.out
5

Explicații:
Cele cinci partiții sunt:
1+1+1+1+1+1+1
1+1+1+1+3
1+1+5
1+3+3
7

Restricții :
$1 \leq n \leq 160$

Timp maxim de executare : 3 secunde/test.

ZIUA 2 **CLASA a X-a**

rubine Fişier sursă: rubine.pas, rubine.c, rubine.cpp

Ali-Baba şi cei N hoţi ai săi au descoperit harta unei comori pe o insulă izolată. Se asimilează harta cu un caroiaj de L*C regiuni (L linii şi C coloane). În fiecare regiune se află un sipet fermecat, în care se găsesc rubine (0 sau mai multe), numărul de rubine din fiecare sipet fiind păstrat în elementul corespunzător al caroiajului.

Se stabileşte următorul plan de acţiune:

-deplasarea hoţului se poate face dintr-o regiune într-o altă regiune vecină (o regiune are maxim 8 regiuni vecine);

-fiecare hoţ pleacă dintr-o anumită regiune, de coordonate date, spre regiunea în care se află Ali-Baba (regiunea din colţul dreapta jos, de coordonate L şi C), pe drumul cel mai scurt (lungimea unui drum fiind egală cu numărul regiunilor parcurse);

-la trecerea printr-o regiune hoţul pune într-un sac rubinele existente în sipetul existent acolo;

-dacă există mai multe drumuri de lungime minimă, un hoţ îl va alege pe acela pe care parcurgându-l, reuşeşte să strângă cât mai multe rubine;

-primul hoţ care se deplasează este cel cu numărul de ordine 1, urmează apoi cel cu numărul de ordine 2, 3, ş.a.m.d., iar în timpul deplasării unui hoţ, ceilalţi stau pe loc până ce acesta ajunge la Ali-Baba;

-sipetul fiind fermecat, în locul rubinelor luate de hoţ, apar altele identice, apariţia petrecându-se înaintea plecării fiecărui hoţ din punctul lui de pornire spre Ali-Baba.

Fiecare hoţ a strâns în sacul său un număr de rubine, cunoscut doar de el şi ajunge cu sacul în regiunea lui Ali-Baba. La împărţirea rubinelor apare şi Sindbad Marinarul care vrea o parte din saci. Ali-Baba cunoscând anumite relaţii care există între perechi de hoţi din regiune va trebui să aleagă acele relaţii care să implice repartizarea unui număr minim de saci lui Sindbad.

Se ştie că alegerea unei perechi de hoţi (i,j) implică repartizarea sacului hoţului j din „cauza” lui i în visteria lui Ali-Baba, însă din acest moment nu se mai poate folosi nici o pereche de forma (j,k), hoţul j fiind eliminat.

Exemplu:

rubine.in	**rubine.out**
4 5	4
1 0 0 0 0	2 9 5 6
0 1 0 0 0	1 12 2 9
0 0 1 4 0	1 12 4 11
0 0 0 1 5	3 10 1 12
5	3
1 1	10
1 5	
4 1	
2 2	
4 4	
1 2	
1 4	
2 5	
3 1	
4 5	

Cerinţă: Ajutaţi-l pe Ali-Baba să aleagă ordinea perechilor astfel încât sacii care rămân să fie într-un număr cât mai mic posibil, ştiind că aceştia-i revin lui Sindbad.

Date de intrare:

Fişierul de intrare **rubine.in**, conţine:

Pe prima linie dimensiunile caroiajului: valorile lui L şi C separate printr-un spaţiu;

Pe următoarele L linii se află câte C valori separate prin spaţiu, reprezentând numărul de rubine din sipetul aflat în regiunea respectivă;

Pe următoarea linie urmează N, numărul de hoţi;

Pe următoarele N linii se află coordonatele regiunilor (valori separate printr-un spaţiu) în care se află iniţial hoţii, pe prima linie dintre acestea pentru hoţul unu, pe linia a doua pentru hoţul doi, etc;

Pe următoarele linii, până la sfârşitul fişierului se află perechile care reprezintă relaţiile din problemă.

Date de ieşire:

Ieşirea se va face în fişierul **rubine.out** în formatul următor:

Pe prima linie se află numărul T de perechi folosite de Ali-Baba, respectându-se cerinţele problemei.

Pe următoarele T linii se află câte patru valori separate două câte două printr-un spaţiu: **i nr1 j nr2**, unde i reprezintă prima valoare din pereche (hoţul cu numărul de ordine i), nr1 reprezintă numărul de rubine adunate de hoţul i, j reprezintă al doilea hoţ din pereche (hoţul cu numărul de ordine j), nr2 reprezentând numărul de rubine strânse de hoţul j.

Pe următoarea linie se află numerele de ordine ale hoţilor ai căror saci au rămas lui Sindbad, valori separate două câte două printr-un spaţiu.

Pe ultima linie se află suma rubinelor din sacii rămaşi lui Sindbad.

Restricţii:

$1 \le \mathbf{L,C,N} \le 50$

$0 \le \mathbf{A[i,j]} \le 10$

Observaţii:

Întotdeauna există soluţii, dacă există mai multe, se va alege una.

În fiecare regiune, indiferent dacă este punctul de pornire al unui hoţ sau regiunea în care se află Ali-Baba există câte un sipet cu un anumit număr de rubine. Ali-Baba nu cunoaşte numărul de rubine din fiecare sac.

Nu există doi hoţi care se află iniţial în aceeaşi regiune.

Timp maxim de executare : 1 secundă/test

ZIUA 2 **CLASA a X-a**

scufița **Fişier sursă: scufita.pas, scufita.c, scufita.cpp**

Majoritatea participanților la ONI2003 au auzit, în copilărie, povestea Scufiței Roşii. Pentru cei care o ştiu, urmează partea a doua; pentru cei care nu o ştiu, nu vă faceți griji, cunoaşterea ei nu este necesară pentru a rezolva această problemă. Povestea nu spune ce s-a întâmplat pe drumul pe care Scufița Roşie s-a întors de la bunicuță. Veți afla amănunte în continuare.

Pe drum, ea s-a întâlnit cu Lupul (fratele lupului care a părăsit povestea în prima parte). Acesta dorea să o mănânce, dar a decis să-i acorde o şansă de scăpare, provocând-o la un concurs de cules ciupercuțe.

Scufița Roşie se află în poziția (1,1) a unei matrici cu N linii şi N coloane, în fiecare poziție a matricei fiind amplasate ciupercuțe. Lupul se află în poziția (1,1) a unei alte matrici similare. Pe parcursul unui minut, atât Scufița, cât şi Lupul se deplasează într-una din pozițiile vecine (pe linie sau pe coloană) şi culeg ciupercuțele din poziția respectivă. Dacă Scufița Roşie ajunge într-o poziție în care nu sunt ciupercuțe, va pierde jocul. Dacă la sfârşitul unui minut ea are mai puține ciupercuțe decât Lupul, ea va pierde jocul de asemenea. Jocul începe după ce amândoi participanții au cules ciupercuțele din pozițiile lor inițiale (nu contează cine are mai multe la începutul jocului, ci doar după un număr întreg strict pozitiv de minute de la început). Dacă Scufița Roşie pierde jocul, Lupul o va mânca.

Înainte de începerea jocului, Scufița Roşie l-a sunat pe Vânător, care i-a promis că va veni într-un sfert de ora (15 minute) pentru a o salva. Deci Scufița Roşie va fi liberă să plece dacă nu va pierde jocul după 15 minute.

Din acest moment, scopul ei este nu numai să rămână în viață, ci şi să culeagă cât mai multe ciupercuțe, pentru a le duce acasă (după ce va veni, vântorul nu o va mai lăsa să culeagă).

Lupul, cunoscut pentru lăcomia sa proverbială, va alege la fiecare minut mutarea în câmpul vecin cu cele mai multe ciupercuțe (matricea sa este dată astfel încât să nu existe mai multe posibilități de alegere la un moment dat).

Povestea spune că Scufița Roşie a plecat acasă cu coşulețul plin de ciupercuțe, folosind indicațiile date de un program scris de un concurent la ONI 2003 (nu vom da detalii suplimentare despre alte aspecte, cum ar fi călătoria în timp, pentru a nu complica inutil enunțul problemei). Să fi fost acest program scris de dumneavoastră? Vom vedea...

Cerință

Scrieți un program care să o ajute pe Scufița Roşie să rămână în joc şi să culeagă cât mai multe ciupercuțe la sfârşitul celor 15 minute!

Date de intrare

Fişierul **scufita.in** are următoarea structură:

N - dimensiunea celor două matrice
a_{11} a_{12} … a1n -matricea din care culege Scufița Roşie
a21 a22 … a2n
…
an1 an2 … ann
b11 b12 … b1n - matricea din care culege Lupul
b21 b22 … b2n
…
bn1 bn2 … bnn

Date de ieşire

Fişierul **scufita.out** are următoarea structură:

NR - numărul total de ciupercuțe culese
d1 d2 ... d15 - direcțiile pe care s-a deplasat Scufița Roşie, separate prin câte un spațiu

(direcțiile pot fi N, E, S, V indicând deplasări spre Nord, Est, Sud, Vest; poziția (1,1) este situată în colțul de Nord-Vest al matricei)

Restricții

- 4<=N<=10;
- valorile din ambele matrici sunt numere naturale mai mici decât 256;
-nici Scufița Roşie şi nici Lupul nu vor părăsi matricele corespunzătoare;
- după ce unul din jucători culege ciupercuțele dintr-o poziție, în poziția respectivă rămân 0 ciupercuțe;
- pe testele date, Scufița Roşie va avea întotdeauna posibilitatea de a rezista 15 minute.

Exemplu

scufita.in	scufita.out
4 **2 2 3 4** **5 6 7 8** **9 10 11 12** **13 14 15 16** **1 2 3 4** **5 6 7 8** **9 10 11 12** **13 14 15 16**	**137** **SSSEEENVVNEENVV**

Explicație:

Scufița Roşie a efectuat aceleaşi mutări cu cele efectuate de Lup şi a avut tot timpul o ciupercuță în plus. În final ea a cules toate ciupercuțele din matrice.

Timp maxim de executare : 1 secundă/test