Descrierea generală a problemelor

Nr. crt.	Denumirea problemei	Restricția referitoare la volumul utilizat de memorie	Restricția referitoare la timpul de execuție, secunde	Punctajul alocat problemei
1.	Joc	≤ 256 Mb	≤ 0,5	100
2.	Zboruri	≤ 256 Mb	≤ 0,5	100
3.	Avioane	≤ 256 Mb	≤ 2,0	100

Notă. În caz de egalitate de punctaj, prioritate se va da concurentului care a obținut punctajul respectiv primul.

Joc

De ziua sa Arina a primit un joc video. Jocul constă dintr-o matrice cu *N* rânduri și *M* coloane cu valori *0* și *1*. Se definește prin *bloc*, o sub-matrice de dimensiunea *2*x*2* care conține aceeași valoare, *0* sau *1*. Unele valori pot face parte din mai multe blocuri. Arina are la dispoziție patru butoane: ***sus, jos, dreapta, stânga*** care corespunzător mută rândurile sau coloanele circular în direcția corespunzătoare butonului. De exemplu, operația ***sus*** va muta toate rândurile cu un o unitate în sus, iar primul rând devine ultimul rând. Respectiv, operația ***dreapta*** va muta toate coloanele în dreapta, iar ultima coloană va deveni prima coloană. Scopul jocului este să se obțină cât mai multe blocuri utilizând cele patru butoane.

Sarcină. Elaborați un program care utilizând cele patru butoane aranjează valorile din matrice astfel încât numărul de blocuri formate este maxim.

Date de intrare. Prima linie a intrării standard conține două numere întregi *N* și *M* separate prin spațiu, numărul de rânduri și coloane ale matricei. Fiecare dintre următoarele *N* linii conține *M* valori *0* sau *1* – valorile matricei (fără spații între valori).

Date de ieșire. Prima și singura linie a ieșirii standard va conține numărul maxim de blocuri pe care Arina le poate obține.

Restricții. $2 \leq N, M \leq 1\,000$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `joc.pas`, `joc.c` sau `joc.cpp`.

Exemplu 1.

Intrare	*Ieșire*
2 4 0110 0110	2

Explicație: O soluție ar fi mișcarea la dreapta, care va muta ultima valorile în dreapta cu o unitate, iar ultima coloană va deveni prima coloană. Ca rezultat vom obține 2 blocuri:

```
0011
0011
```

Exemplu 2.

Intrare	*Ieșire*
3 2 11 11 11	2

Explicație: În acest exemplu toate valorile din start sunt aranjate astfel încât formează 2 blocuri. Rândurile 1 și 2 formează un bloc, rândurile 2 și 3 formează al doilea bloc. Valorile din rândul 2 fac parte din ambele blocuri.

Exemplu 3.

Intrare	*Ieșire*

```
3 3
000
010
000
```

```
3
```

Explicație: O soluție ar fi să acționăm butoanele ***dreapta*** și ***jos***. Acesta va reconfigura matricea astfel:

```
000
000
001
```

Observăm că valorile cu 0 sunt aranjate optim și formează 3 blocuri.

Punctarea. Testele vor fi grupate în trei categorii, după cum urmează:

A. Pentru 35% din teste: $2 \leq N, M \leq 50$.
B. Pentru alte 30% din teste: $2 \leq N, M \leq 300$.
C. Pentru alte 35% din teste: $2 \leq N, M \leq 1000$.

Zboruri

Într-o anumită țară, există *N* orașe etichetate cu numere *1, 2, ..., N* și conectate între ele prin zboruri bidirecționale. În oraș există un dispecerat care gestionează programul de zboruri. În fiecare zi, dispeceratul face modificări la programul de zboruri în felul următor:

1. Se selectează un oraș *X*.
2. Se adaugă zboruri din orașul *X* către toate celelalte orașe pentru care nu există zboruri.
3. Toate zborurile existente anterior din orașul *X* sunt anulate.

De exemplu, dacă există zboruri de la Orașul 5 la Orașele 1 și 2, dar nu există zboruri la Orașele 3 și 4, după modificările efectuate, vor exista zboruri de la Orașul 5 la Orașele 3 și 4, iar zborurile către Orașele 1 și 2 vor fi anulate.

Cetățenii acestei țări se întreabă dacă ar putea dispeceratul realiza un program de zboruri complet, adică un program în care va exista un zbor direct între fiecare pereche de orașe diferite.

Sarcină. Elaborați un program care determină dacă dispeceratul poate realiza un program de zboruri complet, adică fiecare pereche de orașe să fie conectată direct prin zboruri, indiferent de secvența de adăugare și anularea zborurilor.

Date de intrare. Prima linie a intrării standard conține un număr întreg *N* – numărul de orașe. A doua linie conține un număr întreg *M*, ce reprezintă numărul de zboruri curente. Fiecare dintre următoarele *M* linii conține două numere distincte, separate prin spațiu – etichetele orașelor conectate în prezent.

Date de ieșire. Pe o singură linie se va afișa răspunsul „DA” sau „NU”, respectiv dacă poate dispeceratul face un program de zboruri complet ori nu.

Restricții. $2 \leq N \leq 1\,000,\ 0 \leq M < N * (N-1)/2$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `zboruri.pas`, `zboruri.c` sau `zboruri.cpp`.

Exemplu 1.

Intrare

```
2
0
```

Ieșire

```
DA
```

Explicații: Avem 2 orașe și nici un zbor curent. Următoarea zi vom avea zbor direct între cele 2 orașe.

Exemplu 2.

Intrare

```
3
2
1 2
2 3
```

Ieșire

```
NU
```

Explicații: Avem *3* orașe și *2* zboruri curente. Observăm că în zilele pare vom avea zborurile *1 2* și *2 3*, iar în zilele impare vom avea zborul *1 3*, deci niciodată nu vom avea toate zborurile în aceeași zi.

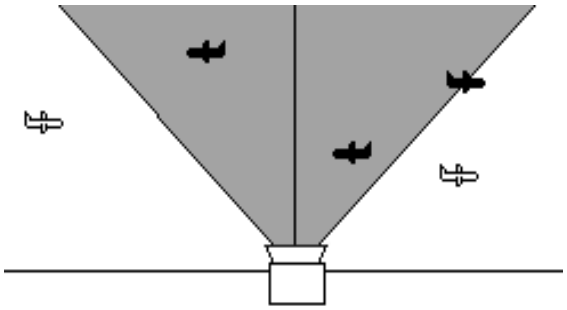
Punctarea. Testele vor fi grupate în următoarele categorii, după cum urmează:

A. Pentru 15% din teste: $N \leq 15$.
B. Pentru alte 23% din teste: $N \leq 25$.
C. Pentru alte 53% din teste: $N \leq 500$.
D. Pentru restul testelor (9%) nu există restricții adiționale.

Avioane

Alex iubește avioanele. Într-o zi el a decis să facă o poză cu cât mai multe avioane posibile.

Toate avioanele zboară într-un plan, cu aceeași viteză de o unitate pe secundă. În total sunt *N* avioane caracterizate prin *3* numere întregi, h_i - înălțimea la care zboară avionul *i* față de sol, x_i - poziția unde se află avionul *i* în momentul de timp $t = 0$ și d_i care este *0* dacă avionul *i* zboară în stânga și *1* dacă zboară în dreapta. E garantat că fiecare avion o să treacă prin punctul $(0, h_i)$ într-un moment de timp $t > 0$.



Alex va alege unde să pună camera (x_0, 0), îndreptată direct în sus. Unghiul de vedere al camerei este de *45* de grade față de axa verticală. Toate avioanele care se află în aria descrisă de unghi, vor nimeri în poză.

Odată ce a ales poziția, Alex poate să facă poza la orice moment de timp.

Sarcină. Elaborați un program care determină numărul maxim de avioane care pot apărea într-o poză.

Date de intrare. Prima linie a intrării standard conține un număr întreg *N* - numărul de avioane. Următoarele *N* linii conțin câte *3* numere întregi separate prin spațiu h_i, x_i și d_i – înălțimea, poziția și direcția avionului *i*.

Date de ieșire. Prima și singura linie a ieșirii standard va conține numărul maxim de avioane care pot apărea în poza lui Alex.

Restricții. $1 \le N \le 3*10^5$; $1 \le h_i, x_i \le 10^9$; $d_i = 0$ sau 1. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `avioane.pas`, `avioane.c` sau `avioane.cpp`.

Exemplu 1.

Intrare

```
3
3 -4 1
4 -12 1
2 3 0
```

Ieșire

```
2
```

Explicație: Alex poate plasa camera la coordonatele (0, 0) și să facă o poză în momentul de timp t = 1. Atunci avioanele 1 și 3 vor apărea în cadru. Nu este nici o metodă de a prinde toate trei avioane în cadru.

Exemplu 2.

Intrare

```
5
2 -2 1
1 1 0
3 8 0
1 4 0
4 -8 1
```

Ieșire

```
4
```

Punctare: Testele vor fi grupate în următoarele categorii:

- Pentru 5%, $N \leq 10$
- Pentru 21%, $N \leq 200$
- Pentru 33%, $N \leq 2000$
- Pentru 41%, fără restricții adiționale