

Problema Graba

Fișier de intrare `graba.in`
Fișier de ieșire `graba.out`

Mutu studiază limbajul C++ și tocmai a învățat despre funcții și instrucțiunea “if”. Pentru a îi testa noile înțelegeri, Surdu, profesorul său, i-a dat temă să scrie o funcție f cu N argumente de tip întreg $x_1, \dots, x_N$ care returnează un întreg. Pentru a îi îngreuna misiunea, tema include M constrângeri suplimentare pe care funcția f trebuie să le satisfacă, specificate printr-o matrice de dimensiuni $M \times (N + 1)$, notată cu A , ale cărei linii și coloane sunt numerotate de la 1 la M și respectiv de la 1 la $N + 1$. Cea de-a i -a constrângere pentru $1 \leq i \leq M$ este că $f(A_{i,1}, \dots, A_{i,N}) = A_{i,N+1}$. Deoarece Mutu se grăbește să se califice în lotul național de informatică, el vă roagă pe voi să îi faceți tema: adică să scrieți funcția f pentru el.

Deoarece cunoștințele lui Mutu sunt încă limitate, funcția f trebuie să se încadreze în materia predată, altminteri Surdu își va da seama de tentativa de fraudă. Așadar, funcția f trebuie să respecte următorul format, din care Mutu nu poate alege decât numărul ℓ și tripletele $(i_1, j_1, k_1), (i_2, j_2, k_2), \dots, (i_\ell, j_\ell, k_\ell)$:

```
int f(int x1, ..., int xN) {  
    if (xi1 == j1) return k1;  
    if (xi2 == j2) return k2;  
    ...  
    if (xiℓ == jℓ) return kℓ;  
    return -1;  
}
```

Figura 1: Funcția pe care o va scrie Mutu.

Cerință

Date fiind N, M și matricea A determinați ℓ și ℓ triplete $(i_1, j_1, k_1), \dots, (i_\ell, j_\ell, k_\ell)$ astfel încât funcția f definită în Figura 1 să satisfacă cele M constrângeri. În cazul în care nu există nicio soluție, precizați acest lucru.

Date de intrare

Pe prima linie a fișierului de intrare `graba.in` se află N și M . Următoarele M linii conțin câte $N + 1$ numere întregi, al j -lea număr de pe a i -a dintre aceste linii reprezentând elementul $A_{i,j}$ al matricei.

Date de ieșire

Dacă nu există nicio soluție, fișierul de ieșire `graba.out` va conține o singură linie cu numărul “-1”. Altfel, fișierul de ieșire va conține ℓ linii, dintre care a x -a linie pentru $1 \leq x \leq \ell$ va conține numerele i_x, j_x, k_x , separate prin spații.

Restricții

- $1 \leq N \cdot M \leq 1\,000\,000$.
- $0 \leq A_{i,j} \leq 1\,000\,000$ pentru $1 \leq i \leq M$ și $1 \leq j \leq N + 1$.
- $1 \leq i_x \leq N$ pentru $1 \leq x \leq \ell$.
- $0 \leq j_x, k_x \leq 1\,000\,000$ pentru $1 \leq x \leq \ell$.
- $1 \leq \ell \leq 2\,000\,000$. Se garantează că în cazul în care există soluție, aceasta se poate obține cu cel

mult 2 000 000 de triplete.

#	Punctaj	Restricții
1	11	$N \cdot M \leq 50$.
2	23	$N \cdot M \leq 200\,000$.
3	18	$N \cdot M \leq 500\,000$.
4	48	Fără restricții suplimentare.

Notă: Subtaskul 4 conține două grupe de teste, în valoare de 29 și respectiv 19 puncte.

Exemple

graba.in	graba.out
4 3 3 2 3 4 4 8 2 2 5 4 3 3 3 6 2	4 9 0 2 2 4 1 3 2
2 4 0 0 0 0 1 1 1 0 1 1 1 0	-1

Explicații exemple

Pentru primul exemplu, Mutu alege $\ell = 3$ triplete $(4, 9, 0)$, $(2, 2, 4)$ și $(1, 3, 2)$, ducând la următoarea funcție f cu $N = 4$ argumente:

```
int f(int x1, int x2, int x3, int x4) {
    if (x4 == 9) return 0;
    if (x2 == 2) return 4;
    if (x1 == 3) return 2;
    return -1;
}
```

Această funcție respectă cele $M = 3$ constrângeri date:

$$f(3, 2, 3, 4) = 4$$

$$f(8, 2, 2, 5) = 4$$

$$f(3, 3, 3, 6) = 2$$

Pentru al doilea exemplu, nicio funcție f de forma dată nu respectă cele $M = 4$ constrângeri date.

Problema Perm

Fișier de intrare `perm.in`
Fișier de ieșire `perm.out`

Se dă o permutare $P[1], \dots, P[N]$ de N numere. Un *ciclu* al permutării este un șir $i_1, \dots, i_k \in \{1, \dots, N\}$ astfel încât $i_2 = P[i_1], i_3 = P[i_2], \dots, i_k = P[i_{k-1}], i_1 = P[i_k]$. Un interval $[x, y]$ este *bun* dacă există un ciclu $i_1, \dots, i_{y-x+1}$ astfel încât fiecare număr $x, x+1, \dots, y$ să apară exact o dată în $i_1, \dots, i_{y-x+1}$.

O interogare constă din două valori x și y , cu $1 \leq x \leq y \leq N$. Răspunsul este numărul minim de interschimbări ce trebuie aplicate asupra permutării inițiale astfel încât intervalul $[x, y]$ să devină bun. O interschimbare constă în a selecta două poziții i și j și a schimba valorile $P[i]$ și $P[j]$ între ele.

Atenție! Interogările sunt independente; adică, interschimbările aplicate pentru o interogare nu sunt păstrate la interogările ce urmează. Mai mult, pentru interogarea $[x, y]$ este voie să interschimbăm elemente din intervalul $[x, y]$ cu elemente din afara intervalului $[x, y]$.

Cerintă

Să se afișeze răspunsul pentru Q interogări.

Date de intrare

Fișierul de intrare `perm.in` conține pe prima linie două numere N și Q .

Pe a doua linie se vor afla N valori reprezentând, în ordine, valorile $P[1], P[2], \dots, P[N]$.

Pe următoarele Q linii se vor afla câte două numere x, y separate printr-un spațiu, reprezentând câte o întrebare.

Date de ieșire

Fișierul de ieșire `perm.out` trebuie să conțină Q linii, pe fiecare răspunsul pentru câte o interogare.

Restricții

- $1 \leq N, Q \leq 300\,000$.
- $1 \leq x \leq y \leq N$.

#	Punctaj	Restricții
1	2	$P[i] = i$.
2	11	$Q = 1$.
3	9	$N, Q \leq 7$.
4	18	$x = 1$.
5	40	$N, Q \leq 100\,000$.
6	20	Fără restricții suplimentare.

Exemplu

perm.in	perm.out	Explicații
5 3 4 1 2 3 5 2 4 1 4 1 5	1 0 1	Pentru a transforma intervalul $[2, 4]$ într-un interval bun, este suficient să facem o singură interschimbare între poziția 1 și poziția 2. Intervalul $[1, 4]$ este deja bun, deci nu avem nevoie de nicio interschimbare. Pentru ultimul interval, avem nevoie de o singură interschimbare între poziția 1 și poziția 5

Problema Redpanda

Fișier de intrare `redpanda.in`
Fișier de ieșire `redpanda.out`

Undeva într-o pădure din Himalaya, un panda roșu este nevoit să se cațere într-un arbore foarte înalt. Din păcate, acestui panda îi este frică de înălțimi și vă zice că nu are curaj să urce mai sus de nivelul K . Astfel, se întreabă dacă l-ați putea ajuta să-și imagineze cum poate face arborele să fie mai puțin înalt.

Se dă descrierea unui arbore cu rădăcină fixă în nodul 1. Considerăm nivelul rădăcinii ca fiind nivelul 1, nivelul copiilor săi ca nivelul 2 și așa mai departe. Arborele poate fi modificat prin următoarea operație în doi pași:

1. Se taie oricare muchie din graf.
2. Se adaugă o nouă muchie în graf.

Cerință

Se cer numărul minim de operații și descrierea acestora pentru a transforma arborele dat într-un arbore unde toate nodurile au nivelul cel mult K .

Date de intrare

Pe prima linie a fișierului de intrare `redpanda.in` se dau N , care reprezintă numărul de noduri din arbore, respectiv K , care este nivelul maxim dorit. Pe următoarele $N - 1$ linii se regăsesc câte două numere întregi X și Y , reprezentând o muchie din arbore între nodurile X și Y .

Date de ieșire

Pe prima linie a fișierului de ieșire `redpanda.out` se află numărul minim de operații S . Pe următoarele S linii se vor afla câte patru întregi A, B, C și D , ce descriu operația: se taie muchia dintre A și B și se adaugă muchie între C și D .

Operațiile descrise trebuie să respecte următoarele reguli:

- Să nu taie o muchie ce nu există.
- Să nu adauge o muchie deja existentă.
- Să nu adauge o muchie de la un nod la el însuși.
- La final graful rezultat să fie un arbore. După prima operație și până înainte de ultima operație, graful are voie să nu fie arbore.

Restricții

- $2 \leq N \leq 300\,000$.
- $1 \leq A, B \leq N$.
- $2 \leq K \leq N$.

#	Punctaj	Restricții
1	7	$K = 2$.
2	12	Muchiile arborelui sunt $1 - 2 - \dots - N$.
3	13	$K > \frac{N}{2}$.
4	17	$N \leq 1000$.
5	51	Fără restricții suplimentare.

Exemplu

redpanda.in	redpanda.out	Explicații
8 4 1 2 2 3 3 4 4 5 4 6 6 7 7 8	1 3 4 6 1	Tăiem muchia dintre nodurile 3 și 4 și adăugăm o nouă muchie între nodurile 6 și 1.

