Ulaznice za karneval (tickets)

Emira se se nalazi na karnevalu u Singapuru. U torbi ima neke nagradne karte koje bi želio koristiti na štandu nagradnih igara. Svaka karta dolazi u jednoj od n boja i sadrži jedan cijeli nenegativni broj. Ovi cijeli brojevi isprintani na kartama mogu imati iste vrijednosti. Zbog neobičnih karnevalskih pravila, garantovano je da će n biti **paran broj**.

Emira posjeduje m karti od svake boje u svojoj torbi, dakle ukupno $n \cdot m$ karti. Karta j boje i ima cjelobrojnu vrijednost $x[i][j]$ isprintanu na njoj ($0 \leq i \leq n-1$ i $0 \leq j \leq m-1$).

Nagradna igra se sastoji od k rundi, numerisanim od 0 do $k-1$. Svaka runda se igra po sljedećem rasporedu:

- Iz svoje torbe Emira odabire **skup** od n ulaznica, po jednu kartu od svake boje. Zatim karte predaje organizatoru igre.
- Organzatoru igre zapisuje brojeve $a[0], a[1] \ \ldots \ a[n-1]$ isprintane na kartama iz dobivenog skupa. Poredak ovih n cijelih brojeva je nebitan.
- Organizator igre onda izvlači specijalnu kartu iz sretne kutije i zapisuje cijeli broj b isprintan na toj karti.
- Organizator igre računa apsolutnu razliku između $a[i]$ i b za svako i od 0 do $n-1$. Neka je S suma ovih apsolutnih razlika.
- Za ovu rundu, organizator igre daje Emiri nagradu u vrijednosti od S
- Sve ulaznice u skupu se odbacuju i ne mogu se koristiti u narednim rundama.

Preostale karte u Emirinoj torbi se odbacuju nakon k rundi igre.

Pomnim promatranjem, Emira je shvatila da je nagradna igra namještena! Unutar sretne kutije za izvlačenje zapravo se nalazi printer. U svakoj rundi, organizator igre pronalazi cijeli broj b koji minimizira vrijednost nagrade u toj rundi. Vrijednost koju je odabrao organizator igre ispisuje se na posebnoj karti za tu rundu.

Imajući sve ove informacije, Emira bi sada željela odrediti karte za svaku rundu igre. Odnosno, želi odabrati skup karata koji će se koristiti u svakoj rundi kako bi se maksimalizirala ukupna vrijednost nagrada.

Detalji implementacije

Potrebno je implementirati sljedeću proceduru:

```
int64 find_maximum(int k, int[][] x)
```

- k: broj rundi.
- x: niz veličine $n \times m$ opisuje cijele brojeve isprintane na svakoj karti. Karte svake boje sortirane su po opadajućem redoslijedu njihovih cijelih brojeva.
- Ova procedura se poziva tačno jednom.
- Ova procedura treba pozvati `allocate_tickets` tačno jednom (pogledaj dole), opisujući k setova karata, jedan za svaku rundu. Raspodjela bi trebala maksimizirati ukupnu vrijednost nagrada.
- Ova procedura bi trebala vratiti maksimalnu ukupnu vrijednost nagrada.

Procedura `allocate_tickets` je definisana na sljedeći način:

```
void allocate_tickets(int[][] s)
```

- s: niz veličine $n \times m$. Vrijednost $s[i][j]$ treba biti r ako je karta j boje i uzeta u set za rundu r, ili -1 ako nije iskorištena uopšte.
- Za svako $0 \leq i \leq n-1$, unutar reda $s[i][0], s[i][1], \ldots, s[i][m-1]$ svaka vrijednost $0, 1, 2, \ldots, k-1$ se mora pojaviti tačno jednom, i sve ostale vrijednosti moraju biti -1.
- Ako postoji više raspodjela koja rezultiraju maksimalnom ukupnom vrijednošću nagrade, dopušteno je prijaviti bilo koju od njih.

Primjeri

Primjer 1

Pogledajmo sljedeći poziv procedure:

```
find_maximum(2, [[0, 2, 5],[1, 1, 3]])
```

Ovo znači da:

- postoje $k = 2$ runde;
- cijeli brojevi isprintani na kartama u boji 0 su 0, 2 i 5;
- cijeli brojevi isprintani na kartama u boji 1 su 1, 1 i 3, respektivno.

Mogući raspored karata koji daje maksimalnu ukupnu vrijednost nagrade je:

- U rundi 0, Ringo bira kartu 0 boje 0 (sa cijelim brojem 0) i kartu 2 boje 1 (sa cijelim brojem 3). Najniža moguća vrijednost nagrade u ovom krugu je 3. Npr. organizator igre može odabrati $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- U rundi 1, Ringo bira kartu 2 boje 0 (sa cijelim brojem 5) i kartu 1 boje 1 (sa cijelim brojem 1). Najniža moguća vrijednost nagrade u ovoj rundi je 4. Npr.organizator igre može odabrati $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Stoga, ukupna vrijednost nagrade će biti $3 + 4 = 7$.

Da biste prijavili ovu rapodjelu, procedura `find_maximum` mora pozvati proceduru

`allocate_tickets`:

- `allocate_tickets([[0, -1, 1], [-1, 1, 0]])`

Konačno, procedura `find_maximum` treba vratiti 7.

Primjer 2

Pogledajmo sljedeći poziv procedure:

```
find_maximum(1, [[5, 9], [1, 4], [3, 6], [2, 7]])
```

Ovo znači da:

- Poatoji samo jedna runda,
- cijeli brojevi isprintani na kartama boje 0 su 5 i 9, respektivno;
- cijeli brojevi isprintani na kartama boje 1 su 1 i 4, respektivno;
- cijeli brojevi isprintani na kartama boje 2 su 3 i 6, respektivno;
- cijeli brojevi isprintani na kartama boje 3 su 2 i 7, respektivno.

Mogući raspored karata koji daje maksimalnu ukupnu vrijednost nagrade je:

- U rundi 0, Ringo bira kartu 1 boje 0 (sa cijelim brojem 9), kartu 0 boje 1 (sa cijelim brojem 1), kartu 0 boje 2 (sa cijelim brojem 3), i kartu 1 boje 3 (sa cijelim brojem 7). Najniža moguća vrijednost nagrade u ovoj rundi 12, kada organizator igre odabere $b = 3$: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

Da biste prijavili ovu rapodjelu, procedura `find_maximum` mora pozvati proceduru `allocate_tickets`:

- `allocate_tickets([[-1, 0], [0, -1], [0, -1], [-1, 0]])`

Konačno, procedura `find_maximum` treba vratiti 12.

Ograničenja

- $2 \leq n \leq 1500$ i n je paran.
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ (za sve $0 \leq i \leq n - 1$ i $0 \leq j \leq m - 1$)
- $x[i][j - 1] \leq x[i][j]$ (za sve $0 \leq i \leq n - 1$ i $1 \leq j \leq m - 1$)

Podzadaci

1. (11 bodova) $m = 1$
2. (16 bodova) $k = 1$
3. (14 bodova) $0 \leq x[i][j] \leq 1$ (za sve $0 \leq i \leq n - 1$ i $0 \leq j \leq m - 1$)

4. (14 bodova) $k = m$
5. (12 bodova) $n, m \leq 80$
6. (23 boda) $n, m \leq 300$
7. (10 bodova) Nema dodatnih ograničenja.

Sample grader

The sample grader čita ulaz u sljedećem formatu:

- linija 1: n m k
- linija $2 + i$ ($0 \leq i \leq n - 1$): $x[i][0]$ $x[i][1]$ $\ldots$ $x[i][m - 1]$

The sample grader ispisuje vaš rezultat u sljedećem formatu:

- linija 1: rezulat funkcije `find_maximum`
- linija $2 + i$ ($0 \leq i \leq n - 1$): $s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m - 1]$