Loterijas biļetes (tickets)

Ringo ir karnevālā Singapūrā. Viņam somā ir dažas loterijas biļetes, kuras viņš gribētu izspēlēt loteriju kioskā. Katra biļete ir vienā no n krāsām un uz katras ir uzdrukāts nenegatīvs vesels skaitlis. Uz dažādām biļetēm uzdrukātie skaitļi var būt arī vienādi. Karnevāla noteikumu dīvainības dēļ ir garantēts, ka n vērtība ir **pāra** skaitlis.

Ringo somā ir m katras krāsas biļetes, tas ir, kopā ir $n \cdot m$ biļetes. Uz j-tās biļetes krāsā i ir uzdrukāts skaitlis $x[i][j]$ ($0 \leq i \leq n-1$ un $0 \leq j \leq m-1$).

Loteriju izspēlē k kārtās, kas ir sanumurētas no 0 līdz $k-1$. Katru kārtu vada spēles vadītājs un tā notiek šādā secībā:

- Ringo no savas somas izvēlas n biļešu **kopu**, pa vienai biļetei katrā krāsā. Viņš tad iesniedz šo biļešu kopu spēles vadītājam.
- Spēles vadītājs pieraksta veselos skaitļus $a[0], a[1] \ \ldots \ a[n-1]$, kas ir uzdrukāti uz kopas biļetēm. Šo n veselo skaitļu secība nav svarīga.
- Spēles vadītājs no loterijas kastes izvelk īpašu kartīti un pieraksta uz šīs kartītes uzdrukāto veselo skaitli b.
- Spēles vadītājs aprēķina starpības starp $a[i]$ un b pēc moduļa visiem i no 0 līdz $n-1$. Ar S apzīmēsim šo starpību pēc moduļa summu.
- Spēles vadītājs iedod Ringo šīs kārtas balvu vērtībā S.
- Kopas biļetes tiek izmestas un nevar tikt izmantotas nākamajās kārtās.

Pēc k kārtām Ringo somā paliekošās biļetes arī tiek izmestas.

Cieši skatoties, Ringo saprata, ka loterijā notiek blēdīšanās! Īstenībā loterijas kastē iekšā ir printeris. Katrā kārtā spēles vadītājs atrod tādu veselu skaitli b, ka balvas vērtība ir mazākā iespējamā. Spēles vadītāja izvēlēto skaitli uzdrukā uz īpašās kartītes šai kārtai.

Ņemot vērā visu šo informāciju, Ringo vēlas sadalīt savas biļetes starp loterijas kārtām. Tas ir, viņš vēlas izvēlēties biļešu kopas katrai kārtai, tā, lai visās kārtās iegūto balvu kopējā vērtība būtu vislielākā.

Implementēšanas detaļas

Jums ir nepieciešams implementēt šādu funkciju:

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

- k: kārtu skaits.

- x: $n \times m$ veselu skaitļu masīvs, kas apraksta uz biļetēm uzdrukātos skaitļus. Katras krāsas biļetes ir sakārtotas pēc uz tām uzdrukātajiem skaitļiem nedilstošā secībā.
- Šī funkcija tiek izsaukta tieši vienreiz.
- Šai funkcijai jāveic tieši viens funkcijas `allocate_tickets` izsaukums(skat. zemāk), aprakstot k biļešu kopas, pa vienai katrai kārtai. Šim sadalījumam būtu jāmaksimizē balvu kopīgā vērtība.
- Šai funkcijai ir jāatgriež vislielākā iespējamā balvu kopējā vērtība.

Funkcija `allocate_tickets` tiek definēta šādi:

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

- s: $n \times m$ masīvs. Elementa $s[i][j]$ vērtībai ir jābūt r, ja j-to biļeti krāsā i izmanto r-tajā loterijas kārtā, vai arī -1, ja to neizmanto vispār.
- Visiem $0 \leq i \leq n-1$, starp $s[i][0], s[i][1], \ldots, s[i][m-1]$ katram skaitlim $0, 1, 2, \ldots, k-1$ ir jāparādās tieši vienreiz, un visiem pārējiem elementiem jābūt -1.
- Ja ir vairāki sadalījumi, kas atbilst vislielākai balvu kopīgai vērtībai, ir atļauts izvadīt informāciju par jebkuru no tiem.

Piemēri

1. piemērs

Aplūkosim šādu izsaukumu:

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

Tas nozīmē, ka:

- ir $k = 2$ kārtas;
- veselie skaitļi, kas uzdrukāti uz krāsas 0 biļetēm ir, attiecīgi, 0, 2 un 5;
- veselie skaitļi, kas uzdrukāti uz krāsas 1 biļētēm ir, attiecīgi, 1, 1 un 3.

Iespējamais sadalījums, kas dod vislielāko kopīgo balvu vērtību, ir:

- 0. kārtā Ringo izvēlas 0. biļeti krāsā 0 (uzdrukāta 0) un 2. biļeti krāsā 1 (uzdrukāts 3). Šajā kārtā vismazākā iespējamā balvas vērtība ir 3. Piemēram, spēles vadītājs var izvēlēties $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- 1. kārtā Ringo izvēlas 2. biļeti krāsā 0 (uzdrukāts 5) un 1. biļeti krāsā 1 (uzdrukāts 1). Šajā kārtā vismazākā iespējamā balvas vērtība ir 4. Piemēram, spēles vadītājs var izvēlēties $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Līdz ar to kopīgā balvu vērtība būtu $3 + 4 = 7$.

Lai paziņotu šo sadalījumu, funkcijai `find_maximum` jāizdara šāds izsaukums funkcijai `allocate_tickets`:

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

Beigās funkcijai `find_maximum` ir jāatgriež 7.

2. piemērs

Aplūkosim šādu izsaukumu:

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

Tas nozīmē, ka:

- ir tikai viena kārta;
- veselie skaitļi, kas uzdrukāti uz krāsas 0 biļetēm ir, attiecīgi, 5 un 9;
- veselie skaitļi, kas uzdrukāti uz krāsas 1 biļetēm ir, attiecīgi, 1 un 4;
- veselie skaitļi, kas uzdrukāti uz krāsas 2 biļetēm ir, attiecīgi, 3 un 6;
- veselie skaitļi, kas uzdrukāti uz krāsas 3 biļetēm ir, attiecīgi, 2 un 7.

Iespējamais sadalījums, kas dod vislielāko kopīgo balvu vērtību, ir:

- $0.$ kārtā Ringo izvēlas $1.$ biļeti krāsā 0 (uzdrukāts 9), $0.$ biļeti krāsā 1 (uzdrukāts 1), $0.$ biļeti krāsā 2 (uzdrukāts 3) un $1.$ biļeti krāsā 3 (uzdrukāts 7). Šajā kārtā vismazākā iespējamā balvas vērtība ir 12 — tad, ja spēles vadītājs izvēlas $b = 3$: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

Lai paziņotu šo atrisinājumu, funkcijai `find_maximum` ir jāveic šāds funkcijas `allocate_tickets` izsaukums:

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

Beigās funkcijai `find_maximum` ir jāatgriež 12.

Ierobežojumi

- $2 \le n \le 1500$ un n ir pāra
- $1 \le k \le m \le 1500$
- $0 \le x[i][j] \le 10^9$ (visiem $0 \le i \le n-1$ un $0 \le j \le m-1$)
- $x[i][j-1] \le x[i][j]$ (visiem $0 \le i \le n-1$ un $1 \le j \le m-1$)

Apakšuzdevumi

1. (11 punkti) $m = 1$
2. (16 punkti) $k = 1$
3. (14 punkti) $0 \le x[i][j] \le 1$ (visiem $0 \le i \le n-1$ un $0 \le j \le m-1$)
4. (14 punkti) $k = m$
5. (12 punkti) $n, m \le 80$

6. (23 punkti) $n, m \leq 300$
7. (10 punkti) Bez papildu ierobežojumiem

Paraugvērtētājs

Paraugvērtētājs ielasa datus šādā formātā:

- $1.$ rinda: $n \;\; m \;\; k$
- $(2+i)$-tā rinda ($0 \leq i \leq n-1$): $x[i][0] \;\; x[i][1] \;\; \ldots \;\; x[i][m-1]$

Paraugvērtētājs izvada jūsu atbildes šādā formātā:

- $1.$ rinda: `find_maximum` atgrieztā vērtība
- $(2+i)$-tā rinda ($0 \leq i \leq n-1$): $s[i][0] \;\; s[i][1] \;\; \ldots \;\; s[i][m-1]$