Karnavalo kuponai (tickets)

Ringo yra karnavale Singapūre. Savo krepšyje jis turi keletą prizų kuponų, kuriuos nori panaudoti prizų žaidimui. Kiekvienas kuponas yra vienos iš n spalvų ir ant kiekvieno kupono yra atspausdinta po neneigiamą sveikąjį skaičių. Ant kelių kuponų gali būti atspausdinti vienodi skaičiai. Dėl karnavalo taisyklių ypatumų n yra **lyginis** skaičius.

Ringo savo krepšyje turi po m kiekvienos spalvos kuponų, t. y. iš viso $n \cdot m$ kuponų. Ant i-osios spalvos j-ojo kupono yra atspausdintas skaičius $x[i][j]$ ($0 \leq i \leq n-1$ ir $0 \leq j \leq m-1$).

Prizų žaidimą sudaryo k etapų, sunumeruotų nuo 0 iki $k-1$. Kiekvienas etapas vyksta tokiu būdu:

- Iš savo krepšio Ringo išsirenka n kuponų, kiekvienos spalvos po vieną. Pasirinktus kuponus jis paduoda žaidimo vedėjui.
- Vedėjas pasižymi ant kuponų atspausdintus skaičius $a[0],\ a[1]\ \ \ldots\ \ a[n-1]$. Šių n skaičių tvarka nėra svarbi.
- Iš laimingų traukimų dėžutės vedėjas ištraukia specialią kortelę ir pasižymi ant jos atspausdintą sveikąjį skaičių b.
- Vedėjas apskaičiuoja skirtumų tarp $a[i]$ ir b modulius kiekvienam i nuo 0 iki $n-1$. Šių skirtumų modulių sumą pavadinkime S.
- Šiame etape vedėjas duoda Ringo prizą, kurio vertė lygi S.
- Šiame etape panaudoti kuponai yra išmetami ir negali būti panaudoti tolesniuose etapuose.

Po k etapų Ringo krepšyje likę kuponai yra išmetami.

Įdėmiai stebėdamas, Ringo suprato, kad šis žaidimas yra nesąžiningas! Laimingųjų traukimų dėžutėje iš tiesų yra spausdintuvas. Kiekviename etape vedėjas pasirenka sveikąjį skaičių b, su kuriuo to etapo prizo vertė yra mažiausia. Vedėjo pasirinktas skaičius yra atspausdinamas ant to etapo specialiosios kortelės.

Žinodamas visa tai, Ringo nori parinkti kuponus etapams tokiu būdu, kad visų prizų verčių suma būtų didžiausia.

Realizacija

Jums reikia parašyti šią funkciją:

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

- k: etapų skaičius.
- x: $n \times m$ dydžio dvimatis masyvas, kuriame yra ant kuponų atspausdinti skaičiai. Kiekvienos

spalvos kuponai yra surikiuoti ant jų atspausdintų skaičių nemažėjimo tvarka.
- Ši funkcija bus iškviesta lygiai vieną kartą.
- Ši funkcija turėtų lygiai vieną kartą iškviesti funkciją `allocate_tickets` (žr. žemiau), pateikdama k kuponų rinkinių, po vieną kiekvienam etapui. Kuponų paskirstymas turėtų maksimizuoti prizų verčių sumą.
- Ši funkcija turėtų grąžinti didžiausią įmanomą prizų verčių sumą.

Funkcija `allocate_tickets` apibrėžta taip:

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

- s: $n \times m$ dydžio dvimatis masyvas. Reikšmė $s[i][j]$ turėtų būti r, jei i-osios spalvos j-asis kuponas yra naudojamas žaidimo r-ajame etape, arba -1, jei kuponas nėra naudojamas nė viename etape.
- Kiekvienam $0 \leq i \leq n-1$, tarp $s[i][0], s[i][1], \ldots, s[i][m-1]$ kiekvienas skaičius $0, 1, 2, \ldots, k-1$ turi būti lygiai po vieną kartą, o visi kiti skaičiai turi būti -1.
- Jei egzistuoja keletas paskirstymų, su kuriais gaunama didžiausia įmanoma prizų verčių suma, galima pateikti bet kurį iš jų.

Pavyzdžiai

Pavyzdys 1

Panagrinėkime tokį funkcijos iškvietimą:

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

Tai reiškia:

- vyksta $k = 2$ etapai;
- ant 0-inės spalvos kuponų atspausdinti skaičiai yra atitinkamai 0, 2 ir 5;
- ant 1-osios spalvos kuponų atspausdinti skaičiai yra atitinkamai 1, 1 ir 3.

Galimas paskirstymas, su kuriuo gaunama didžiausia prizų verčių suma, yra:

- Nuliniame etape Ringo pasirenka 0-inės spalvos kuponą nr. 0 (su skaičiumi 0) ir 1-osios spalvos kuponą nr. 2 (su skaičiumi 3). Mažiausia įmanoma prizo vertė šiame etape yra 3. T. y., vedėjas gali pasirinkti $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- Pirmajame etape Ringo pasirenka 0-inės spalvos kuponą nr. 2 (su skaičiumi 5) ir 1-osios spalvos kuponą nr. 1 (su skaičiumi 1). Mažiausia įmanoma prizo vertė šiame etape yra 4. T. y., vedėjas gali pasirinkti $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Taigi bendra prizų vertė būtų $3 + 4 = 7$.

Šio paskirstymo pateikimui funkcija `find_maximum` turėtų atlikti tokį `allocate_tickets` iškvietimą:

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

Galiausiai, funkcija `find_maximum` turėtų grąžinti 7.

Pavyzdys 2

Panagrinėkime tokį funkcijos iškvietimą:

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

Tai reiškia:

- vyksta tik vienas etapas,
- ant 0-inės spalvos kuponų atspausdinti skaičiai yra atitinkamai 5 ir 9;
- ant 1-osios spalvos kuponų atspausdinti skaičiai yra atitinkamai 1 ir 4;
- ant 2-osios spalvos kuponų atspausdinti skaičiai yra atitinkamai 3 ir 6;
- ant 3-osios spalvos kuponų atspausdinti skaičiai yra atitinkamai 2 ir 7.

Galimas paskirstymas, su kuriuo gaunama didžiausia prizų verčių suma, yra:

- Nuliniame etape Ringo pasirenka 0-inės spalvos kuponą nr. 1 (su skaičiumi 9), 1-osios spalvos kuponą nr. 0 (su skaičiumi 1), 2-osios spalvos kuponą nr. 0 (su skaičiumi 3) ir 3-osios spalvos kuponą nr. 1 (su skaičiumi 7). Mažiausia įmanoma prizo vertė šiame etape yra 12, jei vedėjas pasirenka $b = 3$: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

Šio paskirstymo pateikimui funkcija `find_maximum` turėtų atlikti tokį `allocate_tickets` iškvietimą:

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

Galiausiai, funkcija `find_maximum` turėtų grąžinti 12.

Ribojimai

- $2 \le n \le 1500$ ir n yra lyginis.
- $1 \le k \le m \le 1500$
- $0 \le x[i][j] \le 10^9$ (visiems $0 \le i \le n - 1$ ir $0 \le j \le m - 1$)
- $x[i][j - 1] \le x[i][j]$ (visiems $0 \le i \le n - 1$ ir $1 \le j \le m - 1$)

Dalinės užduotys

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

6. (23 taškai) $n, m \leq 300$
7. (10 taškų) Jokių papildomų ribojimų.

Pavyzdinė vertinimo programa

Pavyzdinė vertinimo programa nuskaito įvestį tokiu formatu:

- 1 – oji eilutė: $n \;\; m \;\; k$
- $2 + i$ – oji eilutė ($0 \leq i \leq n - 1$): $x[i][0] \;\; x[i][1] \;\; \ldots \;\; x[i][m - 1]$

Pavyzdinė vertinimo programa išveda jūsų atsakymą tokiu formatu:

- 1 – oji eilutė: funkcijos `find_maximum` grąžinta vertė
- $2 + i$ – oji eilutė ($0 \leq i \leq n - 1$): $s[i][0] \;\; s[i][1] \;\; \ldots \;\; s[i][m - 1]$