Karnevál sorsjegyek (tickets)

Ringó egy karneválon vesz részt Szingapúrban. A táskájában van néhány sorsjegy, melyekkel szeretne részt venni egy nyereményjátékban. Minden sorsjegynek van színe (összesen n-féle szín van), és egy-egy nemnegatív egész szám van rájuk nyomtatva. Ezek a számok nem feltétlenül különbözőek. A nyereményjáték furcsa szabályai miatt n biztosan **páros**.

Ringónak minden színből m sorsjegye van, tehát összesen $n \cdot m$ darab. A j-edik i színű sorsjegyen az $x[i][j]$ szám van ($0 \leq i \leq n-1$ és $0 \leq j \leq m-1$).

A nyereményjáték k körből áll, melyeket 0-tól $k-1$-ig számoznak. Az egyes körök a következőképp zajlanak:

- Ringó kiválaszt a sorsjegyeiből egy n elemű **halmazt**, minden színből egyet. Ezeket átadja a játékmesternek.
- A játékmester feljegyzi a kapott sorsjegyeken lévő $a[0], a[1] \ \ldots \ a[n-1]$ számokat. A számok sorrendje lényegtelen.
- A játékmester húz egy számkártyát a sorsoló dobozból és felírja a rajta lévő b számot.
- A játékmester kiszámolja $a[i]$ és b különbségének abszolútértékét minden i-re 0-tól $n-1$-ig. Legyen S a különbségek abszolútértékeinek összege.
- Ebben a körben Ringó egy S értékű nyereményt kap.
- Az ebben a körben átadott sorsjegyeket eltépik, a továbbiakban nem használhatók.

A k kör után Ringónál maradó sorsjegyeket szintén eltépik.

Ringó alaposan megfigyelte a játékot, és észrevette, hogy manipulált! A sorsoló dobozban egy nyomtatót rejtettek el. A játékmester minden körben kitalál egy olyan b számot, ami minimalizálja a nyeremény értékét, és ez a szám lesz a dobozból húzott számkártyára nyomtatva.

Mindezen információ birtokában Ringó szeretné beosztani a sorsjegyeit a játék köreire. Segíts neki kiválasztani a sorsjegy halmazt az egyes körökre úgy, hogy a nyereményeinek összértéke maximális legyen!

Megvalósítás

A következő függvényt kell elkészítened:

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

- k: a játék köreinek száma.
- x: egy $n \times m$ méretű tömb, ami a sorsegyeken lévő számokat írja le. Az egyes színekhez

tartozó sorsjegyek a számaik szerint nemcsökkenő sorrendben vannak.
- Ez a függvény pontosan egyszer lesz meghívva.
- Ennek az függvénynek pontosan egyszer kell meghívnia az `allocate_tickets` függvényt (lásd alább), megadva k sorsjegy halmazt, minden körhöz egyet. A megadott sorsjegy elosztásnak maximalizálnia kell a nyeremények összértékét.
- A függvény visszatérési értéke legyen a maximális nyeremény összértéke.

Az `allocate_tickets` függvényt a következőképpen definiáljuk:

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

- s: egy $n \times m$ méretű tömb. $s[i][j]$ értéke legyen r ha a j-edik i színű sorsjegyet az r-edik körben kell felhasználni, vagy -1 ha egyáltalán nem kell használni!
- Minden $0 \leq i \leq n-1$-re az $s[i][0], s[i][1], \ldots, s[i][m-1]$ értékek között a $0, 1, 2, \ldots, k-1$ számok mindegyikének pontosan egyszer kell szerepelnie, a többi elem pedig legyen -1!
- Ha több elosztás is maximalizálja a nyeremények összértékét, bármelyik megadható.

Példák

1. példa

Tekintsük a következő hívást:

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

Ez azt jelenti, hogy:

- $k = 2$ körből áll a játék;
- A 0-ás színű sorsjegyekre nyomtatott számok a 0, 2 és az 5;
- Az 1-es színű sorsjegyekre nyomtatott számok az 1, 1 és a 3.

Egy lehetséges elosztás ami maximális nyeremény összértéket eredményez:

- A 0. körben Ringó a 0-ás színből a 0. sorsjegyet választja (amin a 0 szám van), az 1-es színből pedig a 2. sorsjegyet (amin a 3 szám van). Ebben a körben a lehető legkisebb nyeremény érték a 3. Ehhez a játékmester választhatja például a $b = 1$-et: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- Az 1. körben Ringó a 0-ás színből a 2. sorsjegyet választja (amin az 5 szám van), az 1-es színből pedig a 1. sorsjegyet (amin az 1 szám van). Ebben a körben a lehető legkisebb nyeremény érték a 4. Ehhez a játékmester választhatja például a $b = 3$-at: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Így a nyeremények összértéke $3 + 4 = 7$.

Ennek az elosztásnak a megadásához a `find_maximum` függvénynek az alábbi módon kell

meghívnia az `allocate_tickets` függvényt:

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

Végül a `find_maximum` függvénynek a 7 értéket kell eredményül adnia.

2. példa

Tekintsük a következő hívást:

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

Ez azt jelenti, hogy:

- csak egy körből áll a játék,
- A 0-ás színű sorsjegyekre nyomtatott számok az 5 és a 9;
- Az 1-es színű sorsjegyekre nyomtatott számok az 1 és a 4;
- A 2-es színű sorsjegyekre nyomtatott számok a 3 és a 6;
- A 3-as színű sorsjegyekre nyomtatott számok a 2 és a 7.

Egy lehetséges elosztás ami maximális nyeremény összértéket eredményez:

- A 0. körben Ringó a 0-ás színből a 1. sorsjegyet választja (amin a 9 szám van), az 1-es színből a 0. sorsjegyet (amin a 1 szám van), a 2-es színből a 0. sorsjegyet (amin a 3 szám van), a 3-as színből pedig a 1. sorsjegyet (amin a 7 szám van). Ebben a körben a lehető legkisebb nyeremény érték a 12, melyhez a játékmester a $b = 3$-at választja: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

Ennek az elosztásnak a megadásához a `find_maximum` függvénynek az alábbi módon kell meghívnia az `allocate_tickets` függvényt:

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

Végül a `find_maximum` függvénynek a 12 értéket kell eredményül adnia.

Korlátok

- $2 \leq n \leq 1500$ és n páros.
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ ($0 \leq i \leq n - 1$ és $0 \leq j \leq m - 1$)
- $x[i][j - 1] \leq x[i][j]$ ($0 \leq i \leq n - 1$ és $1 \leq j \leq m - 1$)

Részfeladatok

1. (11 pont) $m = 1$
2. (16 pont) $k = 1$

3. (14 pont) $0 \leq x[i][j] \leq 1$ ($0 \leq i \leq n-1$ és $0 \leq j \leq m-1$)
4. (14 pont) $k = m$
5. (12 pont) $n, m \leq 80$
6. (23 pont) $n, m \leq 300$
7. (10 pont) Nincs további korlátozás.

Minta értékelő

A minta értékelő az alábbi formában olvassa a bemenetet:

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

A következő formában írja ki a választ:

- Az 1. sor: a `find_maximum` visszatérési értéke
- A $2+i$. sor ($0 \leq i \leq n-1$): $s[i][0] \;\; s[i][1] \;\; \ldots \;\; s[i][m-1]$