Kirmes-Lose (tickets)

Ringo ist auf einer Kirmes in Singapur. Er hat einige Lose in seiner Tasche, die er an der Losbude einlösen möchte. Jedes Los hat eine von n Farben und eine nichtnegative Ganzzahl aufgedruckt. Die Zahlen auf verschiedenen Losen können gleich sein. Dank einer Eigenart der Regeln auf der Kirmes ist n auf jeden Fall **gerade**.

Ringo hat m Lose einer jeden Farbe in seiner Tasche, insgesamt also $n \cdot m$ Lose. Das Los j mit der Farbe i hat die Ganzzahl $x[i][j]$ aufgedruckt ($0 \leq i \leq n-1$ und $0 \leq j \leq m-1$).

Das Losspiel wird über k Runden gespielt, die durchnummeriert sind von 0 bis $k-1$. Jede Runde wird nach dem folgenden Ablauf gespielt:

- Ringo wählt aus seiner Tasche eine **Menge** von n Losen, ein Los einer jeden Farbe. Die Menge gibt er dann dem Spielleiter.
- Der Spielleiter notiert sich die Zahlen $a[0],\ a[1]\ \ \ldots\ \ a[n-1]$, die auf den Losen der Menge aufgedruckt sind. Die Reihenfolge dieser n Zahlen ist nicht wichtig.
- Der Spielleiter zieht eine Extrakarte aus einer Losbox und notiert sich die Ganzzahl b, die auf dieser Karte aufgedruckt ist.
- Der Spielleiter berechnet die absolute Differenz zwischen $a[i]$ und b für jedes i von 0 bis $n-1$. Sei S die Summe dieser absoluten Differenzen.
- In dieser Runde gibt der Spielleiter Ringo einen Preis mit Wert S.
- Die Lose in der Menge werden weggeworfen und können in weiteren Runden nicht mehr verwendet werden.

Die nach k Runden des Spiels in Ringos Tasche verbleibenden Lose werden verworfen.

Bei genauerem Hinsehen hat Ringo herausgefunden, dass das Spiel gezinkt ist! In der Losbox ist ein Drucker versteckt. In jeder Runde überlegt sich der Spielleiter eine Ganzzahl b, die den Wert des Preises in dieser Runde minimiert. Der Wert, den sich der Spielleiter überlegt hat, wird auf die Extrakarte dieser Runde gedruckt.

Mit diesem Wissen möchte Ringo die Lose auf die Runden des Spiels aufteilen. Das heißt, dass er die Menge an Losen, die er in jeder Runde verwendet, so auswählen möchte, dass der Gesamtwert der Preise maximal ist.

Implementierungsdetails

Du sollst die folgende Funktion implementieren:

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

- k: die Anzahl an Runden.
- x: ein Array der Größe $n \times m$, das die Zahl auf jedem der Lose beschreibt. Die Lose jeder Farbe sind in nicht-absteigender Reihenfolge ihrer Zahlen sortiert.
- Diese Funktion wird genau ein Mal aufgerufen.
- Diese Funktion soll genau ein Mal `allocate_tickets` aufrufen (siehe unten) und k Losmengen beschreiben, eine für jede Runde. Die Aufteilung soll den Gesamtwert der Preise maximieren.
- Diese Funktion soll den maximalen Gesamtwert der Preise zurückgeben.

Die Funktion `allocate_tickets` ist wie folgt definiert:

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

- s: ein Array der Größe $n \times m$. Der Wert von $s[i][j]$ soll r sein, wenn das Los j der Farbe i in der Menge der Runde r des Spiels verwendet wird, oder -1, wenn es gar nicht verwendet wird.
- Für jedes $0 \leq i \leq n-1$ muss unter $s[i][0], s[i][1], \ldots, s[i][m-1]$ jeder Wert $0, 1, 2, \ldots, k-1$ genau ein Mal vorkommen und alle anderen Einträge müssen -1 sein.
- Falls es mehrere Zuweisungen gibt, die zum maximalen Gesamtwert der Preise führen, darf eine beliebige davon ausgegeben werden.

Beispiele

Beispiel 1

Betrachte den folgenden Aufruf:

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

Das bedeutet das Folgende:

- Es gibt $k = 2$ Runden;
- die Zahlen, die auf den Losen mit der Farbe 0 aufgedruckt sind, sind 0, 2 beziehungsweise 5;
- die Zahlen, die auf den Losen mit der Farbe 1 aufgedruckt sind, sind 1, 1 beziehungsweise 3.

Eine mögliche Zuweisung, die zum maximalen Gesamtwert der Preise führt, ist:

- In Runde 0 wählt Ringo Los 0 mit der Farbe 0 (mit der Zahl 0) und Los 2 mit der Farbe 1 (mit der Zahl 3). Der minimale Preiswert in dieser Runde ist 3. Beispielsweise kann der Spielleiter $b = 1$ wählen: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- In Runde 1 wählt Ringo Los 2 mit der Farbe 0 (mit der Zahl 5) und Los 1 mit der Farbe 1 (mit der Zahl 1). Der minimale Preiswert in dieser Runde ist 4. Beispielsweise kann der Spielleiter

$b = 3$ wählen: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Daher wäre der Gesamtwert der Preise $3 + 4 = 7$.

Um diese Zuweisung auszugeben, soll die Funktion `find_maximum` den folgenden Aufruf an `allocate_tickets` tätigen:

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

Schlussendlich soll die Funktion `find_maximum` den Wert 7 zurückgeben.

Beispiel 2

Betrachte den folgenden Aufruf:

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

Das bedeutet das Folgende:

- Es gibt nur eine Runde,
- die Zahlen, die auf den Losen mit der Farbe 0 aufgedruckt sind, sind 5 beziehungsweise 9;
- die Zahlen, die auf den Losen mit der Farbe 1 aufgedruckt sind, sind 1 beziehungsweise 4;
- die Zahlen, die auf den Losen mit der Farbe 2 aufgedruckt sind, sind 3 beziehungsweise 6;
- die Zahlen, die auf den Losen mit der Farbe 3 aufgedruckt sind, sind 2 beziehungsweise 7.

Eine mögliche Zuweisung, die zum maximalen Gesamtwert der Preise führt, ist:

- In Runde 0 wählt Ringo Los 1 mit der Farbe 0 (mit der Zahl 9) und Los 0 mit der Farbe 1 (mit der Zahl 1), Los 0 mit der Farbe 2 (mit der Zahl 3), und Los 1 mit der Farbe 3 (mit der Zahl 7). Der minimale Preiswert in dieser Runde ist 12, indem der Spielleiter $b = 3$ wählt: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

Um diese Zuweisung auszugeben, soll die Funktion `find_maximum` den folgenden Aufruf an `allocate_tickets` tätigen:

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

Schlussendlich soll die Funktion `find_maximum` den Wert 12 zurückgeben.

Beschränkungen

- $2 \leq n \leq 1500$ und n ist gerade.
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ (für alle $0 \leq i \leq n - 1$ und $0 \leq j \leq m - 1$)
- $x[i][j - 1] \leq x[i][j]$ (für alle $0 \leq i \leq n - 1$ und $1 \leq j \leq m - 1$)

Teilaufgaben

1. (11 points) $m = 1$
2. (16 points) $k = 1$
3. (14 points) $0 \leq x[i][j] \leq 1$ (für alle $0 \leq i \leq n-1$ und $0 \leq j \leq m-1$)
4. (14 points) $k = m$
5. (12 points) $n, m \leq 80$
6. (23 points) $n, m \leq 300$
7. (10 points) Keine weiteren Beschränkungen.

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe im folgenden Format:

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

Der Grader gibt Deine Antwort im folgenden Format aus:

- Zeile 1: der Rückgabewert von `find_maximum`
- Zeile $2 + i$ ($0 \leq i \leq n-1$): $s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m-1]$