嘉年華票券 (tickets)

Ringo　正在新加坡參加一場嘉年華會。他的提袋裡有一些嘉年華會提供的票券，打算到一個遊戲攤位使用。票券共有 n 種顏色，而且每張票券上印有一個非負整數。不同票券上的整數可能相同。基於嘉年華會的特別規定，n 必為 **偶數**。

Ringo 的提袋中每種顏色的票券皆有 m 張，即總共有 $n \cdot m$ 張票券。顏色 i 的第 j 張票券上的整數為 $x[i][j]$ ($0 \leq i \leq n-1$ 且 $0 \leq j \leq m-1$)。

遊戲攤位中的遊戲將進行 k 回合，編號由 0 至 $k-1$。每回合的進行順序如下：

- Ringo 由提袋中選出 一組 票券，共有 n 張，每種顏色各一。然後將這組票券交給遊戲主持人。
- 主持人記下這組票券上印的數字 $a[0],\ a[1]\ \ \ldots\ \ a[n-1]$。這 n 個數字的順序無關緊要。
- 主持人由摸彩箱中抽出一張卡片，並記下卡片上印的數字 b。
- 對 0 至 $n-1$ 的每個整數 i，主持人計算 $a[i]$ 與 b 之差的絕對值。令 S 為這些差之絕對值的和。
- 在此回合，主持人會給予 Ringo 價值為 S 的獎勵。
- 這組票券會被捨棄，在之後的回合中不得使用。

在進行 k 個回合後，Ringo 提袋中剩餘的票券會被捨棄不用。

經由仔細的觀察，Ringo　發現這個遊戲有被動手腳！其實在摸彩箱中有一個印表機。每回合主持人會找出最小化該回合獎勵的整數 b。主持人找出的這個數值將被印在該回合要被抽出的卡片上。

由上述的資訊，Ringo　打算好好配置各個回合的票券。意即，他想選擇好每回合的票券組，使得最後的總獎勵最大化。

實作細節

你應該實作下列程序：

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

- k: 回合數。
- x: 一 $n \times m$ 陣列描述每張票券上的整數。每種顏色的票券依其上的整數做非遞減排序。
- 此程序被呼叫恰好一次。
- 此程序應呼叫 `allocate_tickets` (如下) 恰一次，`allocate_tickets` 描述 k 個票券組，一回合一組。此配置應最大化總獎勵。
- 此程序應回傳總獎勵之最大值。

程序 `allocate_tickets` 定義如下：

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

- s: 一 $n \times m$ 陣列。若顏色 i 的第 j 個票券被用於回合 r 的票券組，則 $s[i][j]$ 的值應設為 r，若該票券完全沒被使用，則 $s[i][j]$ 應設為 -1。
- 對 $0 \leq i \leq n-1$，在 $s[i][0], s[i][1], \ldots, s[i][m-1]$ 中，$0, 1, 2, \ldots, k-1$ 須出現恰好一次，而其他值必須為 -1。
- 若有多種票券組的配置能使總獎勵最大化，則回報任意一種皆可。

範例

範例 1

考慮下列呼叫：

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

這表示：

- 有 $k = 2$ 回合；
- 顏色 0 的票券上印的數字分別為 0、2 與 5；
- 顏色 1 的票券上印的數字分別為 1、1 與 3。

一獲得最大獎勵的票券組配置為：

- 在回合 0，Ringo 選了顏色 0 票券的第 0 張 (其上整數為 0) 以及顏色 1 票券的第 2 張 (其上整數為 3)。此回合最低可能的獎勵為 3。例如，主持人可選 $b = 1$：$|1 - 0| + |1 - 3| = 1 + 2 = 3$。
- 在回合 1，Ringo 選了顏色 0 票券的第 2 張 (其上整數為 5) 以及顏色 1 票券的第 1 張 (其上整數為 1)。此回合最低可能的獎勵為 4。例如，主持人可選 $b = 3$：$|3 - 1| + |3 - 5| = 2 + 2 = 4$。
- 因此，總獎勵為 $3 + 4 = 7$。

要回報此票券組配置，程序 `find_maximum` 應呼叫以下的 `allocate_tickets`：

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

最後，程序 `find_maximum` 會回傳 7。

範例 2

考慮下列呼叫：

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

這表示：

- 僅有一回合，
- 顏色 0 的票券上印的數字分別為 5 與 9；
- 顏色 1 的票券上印的數字分別為 1 與 4；
- 顏色 2 的票券上印的數字分別為 3 與 6；
- 顏色 3 的票券上印的數字分別為 2 與 7。

一獲得最大獎勵的票券組配置為：

- 在回合 0，Ringo 選了顏色 0 票券的第 1 張 (其上整數為 9)、顏色 1 票券的第 0 張 (其上整數為 1)、顏色 2 票券的第 0 張 (其上整數為 3) 以及顏色 3 票券的第 1 張 (其上整數為 7)。此回合最低可能的獎勵為 12，即當主持人選 $b = 3$：$|3-9| + |3-1| + |3-3| + |3-7| = 6 + 2 + 0 + 4 = 12$。

要回報此票券組配置，程序 `find_maximum` 應呼叫以下的 `allocate_tickets`：

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

最後，程序 `find_maximum` 會回傳 12。

條件限制

- $2 \le n \le 1500$ 且 n 為偶數。
- $1 \le k \le m \le 1500$
- $0 \le x[i][j] \le 10^9$ (對所有 $0 \le i \le n-1$ 與 $0 \le j \le m-1$)
- $x[i][j-1] \le x[i][j]$ (對所有 $0 \le i \le n-1$ 與 $1 \le j \le m-1$)

子任務

1. (11 points) $m = 1$
2. (16 points) $k = 1$
3. (14 points) $0 \le x[i][j] \le 1$ (對所有 $0 \le i \le n-1$ 與 $0 \le j \le m-1$)
4. (14 points) $k = m$
5. (12 points) $n, m \le 80$
6. (23 points) $n, m \le 300$
7. (10 points) 無額外限制。

範例評分程式

此範例評分程式以下述格式讀取輸入：

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

此範例評分程式以下述格式輸出你的答案：

- line 1: `find_maximum` 的回傳值

- line $2 + i$ ($0 \le i \le n - 1$): $s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m - 1]$