嘉年华奖券 (tickets)

Ringo正在参加在新加坡举办的一个嘉年华活动。他的口袋里装有一些奖券，这些奖券可以在嘉年华的游戏展位使用。假设共有n种颜色的奖券，每张奖券涂上了其中的一种颜色并且印上了一个非负整数。不同奖券上的数字可能相同。依据嘉年华活动的规则要求，n保证是偶数。

Ringo每种颜色的奖券有m张，也就是说他共有$n \cdot m$张奖券。其中，第i种颜色对应的第j张奖券上印的数字为 $x[i][j]$ $(0 \leq i \leq n-1$ 且 $0 \leq j \leq m-1)$。

一次奖券游戏要进行 k 轮，轮次的序号从0到$k-1$。每一轮按照下面的方式进行：

- 首先，Ringo从每种颜色的奖券中各选出一张奖券，形成一个n 张奖券的**集合**。
- 随后，游戏负责人记录下这个集合中奖券上的数字$a[0]$，$a[1]$ $\ldots$ $a[n-1]$。不需要考虑这 n 个整数的排序。
- 接下来，游戏负责人从一个幸运抽奖箱中抽取一张特殊卡片，上面印有整数 b。
- 对于上述集合中每一个奖券上的数字 $a[i]$ $(0 \leq i \leq n-1)$，游戏负责人会计算 $a[i]$ 和 b的差的绝对值。让 S 代表这n个差的绝对值之和。
- 所得到的数字S 就是Ringo本轮能够获得的奖励数额。
- 一轮游戏结束后，本轮集合中的奖券全部被丢弃，不会在未来的轮次所使用。

当k轮游戏结束后， Ringo会丢弃口袋中的所有奖券。

通过仔细观察， Ringo发现这个奖券游戏被操控了！实际上，幸运抽奖箱里面内置了一台打印机。在每一轮，游戏负责人首先找到一个能够最小化当前轮次游戏奖励的整数b，然后将该数字打印在他所抽取的特殊卡片上。

知道了这些信息之后， Ringo想要设计每轮游戏中的奖券分配方案，使得 k 轮游戏中获得的总体奖励数额之和最大。

实现细节

你需要实现下面这个函数：

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

- k: 游戏的轮数。

- x: 一个 $n \times m$ 的数组，记录了奖券上的数字。每种颜色的奖券按照上面的数字非递减顺序排序。
- 这个函数只会被调用一次。
- 这个函数应该只调用一次函数 `allocate_tickets` (参见下面的内容)，它描述了 k 轮游戏中的奖券分配方案，每一轮对应一个奖券集合。奖券的分配方案应该使得所获奖励数额之和达到最大。
- 这个函数需要返回能够获得的最大的奖励数额之和。

函数 `allocate_tickets` 按照如下的方式进行定义:

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

- s: 一个 $n \times m$ 的数组。如果第 i 种颜色的第 j张奖券如果被分配到了第 r 轮游戏，那么 $s[i][j]$ 的值应该为 r；如果未被使用，应该为 -1。
- 对于 $0 \leq i \leq n-1$，在 $s[i][0], s[i][1], \ldots, s[i][m-1]$中，每个值 $0, 1, 2, \ldots, k-1$ 必须只出现一次，而其他元素应该为 -1。
- 如果存在多种奖券分配方案能够达到最优的奖励数值，可以给出其中任何一种最优方案。

数据样例

数据样例1

考虑下面的函数调用:

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

这意味着:

- 游戏共进行 $k = 2$ 轮;
- 第0种颜色奖券上的整数数字分别是0, 2 和 5;
- 第1种颜色奖券上的整数数字分别是1, 1 和 3;

一种能够获得最优奖励数值的分配方案是:

- 在第0轮， Ringo选择第0种颜色的第0张奖券 (印有整数0) 和第1种颜色的第2张奖券 (印有整数3)。本轮获得的最小奖励数额是3。例如，游戏负责人可以选择 $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$。
- 在第1轮，Ringo选择第0种颜色的第2张奖券 (印有整数 5) 和第1种颜色的第1张奖券 (印有整数1)。本轮能够获得的最小奖励是4。例如，游戏负责人可以选择 $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$。
- 因此，本次游戏两轮的奖励之和为 $3 + 4 = 7$。

为了给出这个分配方案，函数 `find_maximum` 应该按照如下方式调用 `allocate_tickets`：

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

最终，函数 `find_maximum`应该返回数字 7.

数据样例2

考虑下面的函数调用：

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

这意味着：

- 游戏只进行一轮；
- 第0种颜色奖券上的数字分别是5 和 9；
- 第1种颜色奖券上的数字分别是1 和 4；
- 第2种颜色奖券上的数字分别是3 和 6；
- 第3种颜色奖券上的数字分别是2 和 7；

一种能够获得最优奖励的分配方案是：

- 在第0轮， Ringo 选择第0种颜色的第1张奖券 (印有整数 9) ，第1种颜色的第0张奖券 (印有整数 1)，第2种颜色的第0张奖券 (印有整数3)，第3种颜色的第1张奖券 (印有整数7)。本轮能够获得的最小奖励是12。例如，游戏负责人可以选择$b = 3$:
  $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$。

为了给出这个分配方案，函数 `find_maximum` 应该按照如下方式调用 `allocate_tickets`：

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

最终，函数 `find_maximum`应该返回数字 12。

输入限制

- $2 \leq n \leq 1500$ 且 n 为偶数
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ (对于所有的 $0 \leq i \leq n-1$ 且 $0 \leq j \leq m-1$)
- $x[i][j-1] \leq x[i][j]$ (对于所有的 $0 \leq i \leq n-1$ 且 $1 \leq j \leq m-1$)

子任务

1. (11 分) $m = 1$
2. (16 分) $k = 1$
3. (14 分) $0 \leq x[i][j] \leq 1$ (对于所有的 $0 \leq i \leq n-1$ 且 $0 \leq j \leq m-1$)

4. (14 分) $k = m$
5. (12 分) $n, m \leq 80$
6. (23 分) $n, m \leq 300$
7. (10 分) 没有其他限制

评测程序示例

评测程序示例按照下面的格式读入数据：

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

评测程序示例按照下面的格式打印你的答案：

- 第1行：`find_maximum`的返回值
- 第$2 + i$行 ($0 \leq i \leq n - 1$)：$s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m - 1]$