カーニバルのチケット (tickets)

Ringo はシンガポールのカーニバルにいる. カーニバルにはチケットゲームの屋台があり, 彼のカバンにそのゲームで用いられるチケットが入っている. チケットには n 種類の色があり, Ringo のカバンの中には各色 m 枚のチケットがある. つまり, 合計で $n \cdot m$ 枚のチケットがある. 各チケットには非負整数が書かれており, 色 i の j 番目のチケットには整数 $x[i][j]$ が書かれている $(0 \leq i \leq n-1, 0 \leq j \leq m-1)$. 異なるチケットに書かれている整数が一致する場合もある. カーニバルの奇妙なルールにより, n は **偶数** であることが保証されている.

チケットゲームは k 回のラウンドがあり, 0 から $k-1$ まで番号が付けられている. それぞれのラウンドは以下のように行われる.

- Ringo は彼のカバンから, 各色 1 枚のチケットを選び, 合計 n 枚のチケットの **集合** を選ぶ. そのチケットの集合をゲームマスターに渡す.
- ゲームマスターはチケットの集合に書かれている整数 $a[0], a[1] \ \ldots \ a[n-1]$ をメモする. これら n 個の整数の順序は重要ではない.
- ゲームマスターは抽選箱からスペシャルカードを引き, そのカードに書かれている整数 b をメモする.
- ゲームマスターは 0 以上 $n-1$ 以下の各 i に対して $a[i]$ と b 間の差の絶対値を計算する. S をこれらの差の絶対値の総和とする.
- このラウンドでは, ゲームマスターは Ringo に価値が S であるような景品を与える.
- 選んだ集合の各チケットは処分され, 次回以降のラウンドに使用することはできない.

k 回のラウンドの後, Ringo のカバンに残っているチケットは処分される.

よく観察してみると, Ringo はチケットゲームが八百長であることが分かった! 抽選箱の中にプリンターが確実に存在する. 各ラウンドで, ゲームマスターは景品の価値が最小となるような整数 b を求めて, そのラウンドのスペシャルカードにはゲームマスターの選んだ値が書き込まれる.

これらの情報をもとに, Ringo はチケットを割り当てたいと思っている. つまり, 景品の価値の総和が最大となるように, 各ラウンドに用いるチケットの集合を選びたい.

実装の詳細

あなたは以下のプロシージャを実装しなさい

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

- k: ラウンドの回数
- x: $n \times m$ の二次元配列で, 各チケットに書かれている整数を表している. 各色のチケットは書かれている整数が広義単調増加になるように並べられている.

- このプロシージャはちょうど 1 回呼び出される.
- このプロシージャは, 各ラウンドで用いられるチケットの集合を報告する `allocate_tickets` (後述) をちょうど 1 回 を呼び出さなければならない. 割り当ては景品の価値の総和が最大にならなければならない.
- このプロシージャは景品の価値の総和の最大値を返さなければならない.

プロシージャ `allocate_tickets` は以下のように定義される.

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

- s: $n \times m$ の二次元配列. 色 i の j 番目のチケットがラウンド r で使われるならば $s[i][j]$ の値は r であり, 全く使用されないならば -1 でなければならない.
- $0 \leq i \leq n-1$ において, $s[i][0], s[i][1], \ldots, s[i][m-1]$ は $0, 1, 2, \ldots, k-1$ をそれぞれちょうど 1 個だけ含んでおり, それ以外は -1 でなければならない.
- 景品の価値の総和が最大となるような割り当てが複数存在する場合, どれを報告しても良い.

入出力例

入出力例 1

次の呼び出しを考える.

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

これは次のようなことを意味している.

- $k = 2$ 回のラウンドがある.
- 色 0 のチケットに書かれている整数はそれぞれ $0, 2, 5$ である.
- 色 1 のチケットに書かれている整数はそれぞれ $1, 1, 3$ である.

景品の価値の総和が最大となる可能な割り当ては以下のとおりである.

- ラウンド 0 では, Ringo は色 0 の 0 番目のチケット(整数 0 が書かれている) と色 1 の 2 番目のチケット(整数 3 が書かれている)を選ぶ. このラウンドでの可能な景品の価値の最小値は 3 である. 例えば, ゲームマスターが $b = 1$ と選ぶと, $|1-0| + |1-3| = 1 + 2 = 3$ となる.
- ラウンド 1 では, Ringo は色 0 の 2 番目のチケット(整数 5 が書かれている) と色 1 の 1 番目のチケット(整数 1 が書かれている)を選ぶ. このラウンドでの可能な景品の価値の最小値は 4 である. 例えば, ゲームマスターが $b = 3$ と選ぶと, $|3-1| + |3-5| = 2 + 2 = 4$ となる.
- したがって, 景品の価値の総和は $3 + 4 = 7$ となる.

この割り当てを報告するために, プロシージャ `find_maximum` は以下のように `allocate_tickets` を呼び出さなければならない.

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

最後に, プロシージャ `find_maximum` は 7 を返さなければならない.

入出力例 2

次の呼び出しを考える.

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

これは次のようなことを意味している.

- $k = 1$ 回のラウンドがある.
- 色 0 のチケットに書かれている整数はそれぞれ $5, 9$ である.
- 色 1 のチケットに書かれている整数はそれぞれ $1, 4$ である.
- 色 2 のチケットに書かれている整数はそれぞれ $3, 6$ である.
- 色 3 のチケットに書かれている整数はそれぞれ $2, 7$ である.

景品の価値の総和が最大となる可能な割り当ては以下のとおりである.

- ラウンド 0 では, Ringo は色 0 の 1 番目のチケット(整数 9 が書かれている) と色 1 の 0 番目のチケット(整数 1 が書かれている), 色 2 の 0 番目のチケット(整数 3 が書かれている), 色 3 の 1 番目のチケット(整数 7 が書かれている)を選ぶ. このラウンドでの可能な景品の価値の最小値は 12 である. 例えば, ゲームマスターが $b = 3$ と選ぶと,
  $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$ となる.

この割り当てを報告するために, プロシージャ `find_maximum` は以下のように `allocate_tickets` を呼び出さなければならない.

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

最後に, プロシージャ `find_maximum` は 12 を返さなければならない.

制約

- $2 \leq n \leq 1500$, n は偶数である.
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ $(0 \leq i \leq n - 1, 0 \leq j \leq m - 1)$
- $x[i][j - 1] \leq x[i][j]$ $(0 \leq i \leq n - 1, 1 \leq j \leq m - 1)$

小課題

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

6. (23 点) $n, m \leq 300$
7. (10 点) 追加の制約はない.

採点プログラムのサンプル

採点プログラムは以下の形式で入力を読み込む.

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

採点プログラムは以下の形式であなたの答えを出力する.

- 1 行目: `find_maximum` の戻り値
- $2 + i$ 行目 ($0 \leq i \leq n - 1$): $s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m - 1]$