Билеты на карнавале (tickets)

Ринго приехал в Сингапур на карнавал. У него есть несколько призовых билетов в сумке, которые он хочет использовать для участия в розыгрыше призов в киоске. Каждый билет покрашен в один из n цветов. На каждом билете написано целое неотрицательное число. Некоторые написанные на разных билетах числа могут совпадать. Из-за особенностей правил карнавала гарантируется, что n **четное**.

У Ринго в его сумке есть m билетов каждого цвета, таким образом, всего у него есть $n \cdot m$ билетов. На j-м билете цвета i написано число $x[i][j]$ ($0 \leq i \leq n-1$ и $0 \leq j \leq m-1$).

Розыгрыш призов проходит в k раундов, пронумерованных от 0 до $k-1$. Каждый раунд проходит следующим способом:

- Ринго выбирает n билетов из сумки, по одному билету каждого цвета, и отдает их ведущему игры.
- Ведущий записывает числа $a[0]$, $a[1]$ $\ldots$ $a[n-1]$, написанные на этих билетах. Порядок этих n чисел не имеет значения.
- Ведущий получает специальную карточку из лототрона и записывает число b, написанное на этой карточке.
- Ведущий вычисляет модуль разности между $a[i]$ и b для всех i от 0 до $n-1$. Пусть S — сумма этих модулей.
- В этом раунде ведущий выдает Ринго приз в размере S.
- Использованные билеты выбрасываются и не могут быть использованы в следующих раундах.

Билеты, оставшиеся в сумке Ринго после k раундов, выбрасываются.

Присмотревшись внимательно, Ринго понял, что игра — нечестная! Внутри лототрона на самом деле находится принтер. В каждом раунде ведущий находит такое число b, чтобы приз в этом раунде оказался минимально возможным. Это значение выбирается ведущим и печатается на специальной карте для этого раунда.

Зная эту информацию, Ринго хочет распределить билеты по раундам. А именно, он хочет выбрать множество билетов для каждого раунда так, чтобы максимизировать суммарный размер полученных призов.

Детали реализации

Вы должны реализовать следующую функцию:

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

- k: количество раундов.
- x: массив размера $n \times m$, описывающий числа на билетах. Билеты каждого цвета отсортированы по неубыванию.
- Функция будет вызвана ровно один раз.
- Функция должна ровно один раз вызвать функцию `allocate_tickets` (смотрите ниже), описывающую k множеств билетов, по одному множеству на каждый раунд. Этот способ выбрать билеты должен максимизировать суммарный размер призов.
- Функция должна вернуть максимальный суммарный размер полученных призов.

Функция `allocate_tickets` определена следующим образом:

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

- s: массив размера $n \times m$. Значение $s[i][j]$ должно быть равно r, если билет j цвета i используется в раунде r, или же -1, если билет не используется.
- Для всех $0 \leq i \leq n-1$ среди чисел $s[i][0], s[i][1], \ldots, s[i][m-1]$ каждое значение $0, 1, 2, \ldots, k-1$ должно встречаться ровно один раз, все остальные элементы должны быть равны -1.
- Если существует несколько способов выбрать билеты, максимизирующих значение суммарного размера призов, можно выбрать любой из них.

Примеры

Пример 1

Рассмотрим следующий вызов функции:

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

Данный вызов означает, что:

- в розыгрыше будет $k = 2$ раунда;
- на карточках цвета 0 написаны числа 0, 2 и 5;
- на карточках цвета 1 написаны числа 1, 1 и 3.

Один из возможных вариантов выбора билетов, который максимизирует суммарный размер призов, такой:

- В раунде 0 Ринго выбирает билет 0 цвета 0 (с числом 0) и билет 2 цвета 1 (с числом 3). Минимальный возможный размер приза за этот раунд равен 3. Например, ведущий может выбрать $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- В раунде 1 Ринго выбирает билет 2 цвета 0 (с числом 5) и билет 1 цвета 1 (с числом 1).

Минимальный возможный размер приза за этот раунд равен 4. Например, ведущий может выбрать $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Таким образом, суммарный размер призов равен $3 + 4 = 7$.

Чтобы выбрать такие билеты, функция `find_maximum` должна вызвать `allocate_tickets` следующим образом:

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

Функция `find_maximum` должна вернуть 7.

Пример 2

Рассмотрим следующий вызов функции:

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

Данный вызов означает, что:

- в розыгрыше только один раунд,
- на карточках цвета 0 написаны числа 5 и 9;
- на карточках цвета 1 написаны числа 1 и 4;
- на карточках цвета 2 написаны числа 3 и 6;
- на карточках цвета 3 написаны числа 2 и 7.

Один из возможных вариантов выбора билетов, который максимизирует суммарный размер призов, такой:

- В раунде 0 Ринго выбирает билет 1 цвета 0 (с числом 9), билет 0 цвета 1 (с числом 1), билет 0 цвета 2 (с числом 3), и билет 1 цвета 3 (с числом 7). Минимальный возможный размер приза за этот раунд равен 12, если ведущий выберет число $b = 3$: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

Чтобы выбрать такие билеты, функция `find_maximum` должна вызвать `allocate_tickets` следующим образом:

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

Функция `find_maximum` должна вернуть 12.

Ограничения

- $2 \leq n \leq 1500$ и n четное.
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ (для всех $0 \leq i \leq n - 1$ и $0 \leq j \leq m - 1$)
- $x[i][j - 1] \leq x[i][j]$ (для всех $0 \leq i \leq n - 1$ и $1 \leq j \leq m - 1$)

Подзадачи

1. (11 баллов) $m = 1$
2. (16 баллов) $k = 1$
3. (14 баллов) $0 \le x[i][j] \le 1$ (для всех $0 \le i \le n-1$ и $0 \le j \le m-1$)
4. (14 баллов) $k = m$
5. (12 баллов) $n, m \le 80$
6. (23 балла) $n, m \le 300$
7. (10 баллов) Нет дополнительных ограничений.

Пример проверяющего модуля

Пример проверяющего модуля считывает ввод в следующем формате:

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

Пример проверяющего модуля выводит ответ в следующем формате:

- строка 1: значение, возвращенное функцией `find_maximum`
- строки $2 + i$ ($0 \le i \le n-1$): $s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m-1]$