카니발 티켓 (tickets)

링고는 싱가폴 카니발에 놀러 갔다. 링고의 가방에는 상품을 받을 수 있는 티켓들이 있어서 게임에 사용하려고 한다. 각 티켓은 n 가지 중 한 색이고 음이 아닌 정수가 하나 인쇄되어 있다. 서로 다른 티켓들에 인쇄된 수가 같은 경우도 있다. 게임 규칙의 특정한 사유 때문에 n은 항상 **짝수**이다.

링고의 가방에 n 가지 색 각각에 대해 티켓이 정확히 m개가 있다. 즉, 모두 $n \cdot m$개의 티켓이 있다. 색이 i번인 티켓들 중 j번인 티켓에는 정수 $x[i][j]$가 인쇄되어 있다. $(0 \leq i \leq n-1, 0 \leq j \leq m-1)$.

게임은 k번의 라운드로 진행된다. 라운드들은 0부터 $k-1$까지 번호가 붙어 있다. 각 라운드는 다음과 같은 과정으로 진행된다.

- 링고는 가방의 티켓들 중 각 색에서 하나씩 뽑아서 n개의 티켓을 모은다. 모은 티켓을 마스터에게 전달한다.
- 마스터는 각 티켓에 인쇄된 정수들을 $a[0],\ a[1]\ \ \ldots\ \ a[n-1]$로 기록한다. 순서는 상관이 없다.
- 마스터는 뽑기 상자에서 특별한 카드 한 장을 꺼낸다. 그 카드에 인쇄된 정수를 b라고 하자.
- 마스터는 각 $a[i]$와 b의 차이의 절대값을 모두 $(0 \leq i \leq n-1)$ 계산한다. 차이의 절대값을 모두 더한 값을 S라고 하자.
- 이번 라운드에서 링고가 받게 되는 상품의 가격은 S이다.
- 이번 라운드에서 사용된 티켓들은 모두 버려지고 더 이상 사용될 수 없다.

모든 (k번의) 라운드가 끝난 후 남은 티켓들도 모두 버려진다.

그런데, 자세히 보니 뽑기 상자에서 어떤 값이 인쇄된 카드가 랜덤한 것이 아니라는 것을 알게 되었다! 뽑기 상자 안에는 프린터가 있고 마스터가 정하는 값을 프린트할 수 있는 것이다. 각 라운드에서 마스터는 상품의 가격 S가 최소화되는 정수 b가 인쇄되도록 한다는 것을 알수 있었다.

이러한 정보를 알게 된 링고는 라운드들에서 선택하는 카드들을 최적화하고 싶어졌다. 즉, 링고는 모든 라운드에서 받는 상품 가치의 총합을 최대화하려고 한다.

Implementation details

다음 함수를 작성해야 한다.

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

- k: 라운드 개수.
- x: 크기 $n \times m$인 배열이다. 각 티켓에 인쇄된 정수를 저장하고 있다. 한가지 색의 티켓들에 인쇄된 수들은 감소하지 않는 순서로 배열에 저장되어 있다.
- 이 함수는 정확히 한번 호출된다.

- 이 함수는 아래에 정의된 `allocate_tickets` 함수를 정확히 한번 호출해야 한다. 이 호출의 인자로 k번의 라운드에서 선택되는 티켓들을 저장하여 호출해야 한다. 티켓의 선택은 상품 가치의 총합을 최대화하여야 한다.
- 이 함수는 상품 가치의 총합을 리턴하여야 한다.

함수 `allocate_tickets`는 다음과 같이 정의된다.

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

- s: 크기 $n \times m$인 배열이다. $s[i][j]$의 값은 그 티켓이 사용된 라운드 번호이다. 즉, 색 i인 티켓들 중 j번인 티켓이 라운드 r에 사용된 경우 r이다. 해당 티켓이 사용되지 않는다면 값이 -1이라야 한다.
- 각 $i (0 \le i \le n-1)$에 대해서 $s[i][0], s[i][1], \ldots, s[i][m-1]$들 중에 $0, 1, 2, \ldots, k-1$는 정확히 한번씩 등장해야 하고 나머지 값들은 -1이라야 한다.
- 최적이 되도록 티켓을 고르는 방법이 여러가지 있은 경우 그들 중 어떤 것을 기록해도 좋다.

Examples

Example 1

다음 호출을 보자.

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

호출의 의미는 다음과 같다.

- 라운드의 수는 $k = 2$이다.
- 0번 색에 인쇄된 수들은 0, 2, 5이다.
- 1번 색에 인쇄된 수들은 1, 1, 3이다.

상품 가치의 총합을 최대화하는 선택 방법 중 한가지는 아래와 같다.

- 라운드 0에서, 링고는 0번 색 카드 중 0번을 고른다. 인쇄된 수는 0이다. 1번 색 카드 중 2번을 고른다. 인쇄된 수는 3이다. 이 라운드에서 최소 상품 가치는 3이다. 예를 들어, 마스터가 $b = 1$을 고르면 상품 가치는 $|1 - 0| + |1 - 3| = 1 + 2 = 3$이 된다.
- 라운드 1에서, 링고는 0번 색 카드 중 2번 티켓을 고른다. 인쇄된 수는 5이다. 1번 색 카드 중 1번을 고른다. 인쇄된 수는 1이다. 이 라운드에서 최소 상품 가치는 4이다. 예를 들어 마스터가 $b = 3$을 고르면 상품 가치는 $|3 - 1| + |3 - 5| = 2 + 2 = 4$이다.
- 따라서, 전체 상품 가치는 $3 + 4 = 7$이다.

위와 같이 선택한 경우 아래와 같이 인자를 설정하여 `allocate_tickets`를 호출해야 한다.

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

마지막으로 `find_maximum`은 7을 리턴해야 한다.

Example 2

다음 호출을 보자.

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

이 호출의 의미는 아래와 같다.

- 단 한번의 라운드가 있음.
- 0번 색의 티켓에는 5, 9가 인쇄되어 있다.
- 1번 색의 티켓에는 1, 4가 인쇄되어 있다.
- 2번 색의 티켓에는 3, 6가 인쇄되어 있다.
- 3번 색의 티켓에는 2, 7가 인쇄되어 있다.

상품 가치의 총합을 최대화하는 선택 방법 중 한가지는 아래와 같다.

- 라운드 0에서 링고는 0번색의 카드 중 1번 (인쇄된 값은 9), 1번색의 카드 중 0번 (인쇄된 값은 1), 2번색의 카드 중 0번 (인쇄된 값은 3), 3번색의 카드 중 1번 (인쇄된 값은 7)을 선택한다. 라운드에서 가능한 최소의 상품 가치는 12이다. 예들 들어 마스터가 $b = 3$을 선택할 경우 가치는 $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$이다.

위와 같이 선택한 경우 아래와 같이 인자를 설정하여 `allocate_tickets`를 호출해야 한다.

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

마지막으로 `find_maximum`은 12를 리턴해야 한다.

Constraints

- $2 \le n \le 1500$, n은 짝수이다.
- $1 \le k \le m \le 1500$
- $0 \le x[i][j] \le 10^9$ $(0 \le i \le n - 1, 0 \le j \le m - 1)$
- $x[i][j - 1] \le x[i][j]$ $(0 \le i \le n - 1, 1 \le j \le m - 1)$

Subtasks

1. (11 points) $m = 1$
2. (16 points) $k = 1$
3. (14 points) $0 \le x[i][j] \le 1$ $(0 \le i \le n - 1, 0 \le j \le m - 1)$
4. (14 points) $k = m$
5. (12 points) $n, m \le 80$
6. (23 points) $n, m \le 300$
7. (10 points) 추가적 제한이 없음.

Sample grader

샘플 그레이더는 다음의 형식으로 입력을 받는다.

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

샘플 그레이더의 출력은 다음의 형식이다.

- line 1: `find_maximum`의 리턴 값
- line $2 + i$ ($0 \le i \le n - 1$): $s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m - 1]$