Bilhetes de Jogo (tickets)

O Ringo está num parque de diversões em Singapura. Ele tem na sua mochila alguns bilhetes, que pretende usar na tenda de jogos. Cada bilhete tem uma de n cores e nele está impresso um número inteiro não negativo. Os inteiros impressos em bilhetes diferentes podem ser os mesmos. Devido a uma peculiaridade das regras do parque de diversões, é garantido que n é **par**.

O Ringo tem na sua mochila m bilhetes de cada cor, isto é, um total de $n \cdot m$ bilhetes. O bilhete j da cor i tem nele impresso o inteiro $x[i][j]$ ($0 \leq i \leq n-1$ e $0 \leq j \leq m-1$).

O jogo decorre ao longo de k rondas, numeradas de 0 a $k-1$. Cada ronda é jogada com a seguinte ordem:

- Da sua mochila, o Ringo seleciona um **conjunto** de n bilhetes, um bilhete de cada cor. De seguida dá este conjunto ao mestre do jogo.
- O mestre do jogo anota os inteiros $a[0], a[1] \ \ldots \ a[n-1]$ impressos nos bilhetes do conjunto. A ordem destes n inteiros não é importante.
- O mestre do jogo tira um cartão especial de uma caixa de sorteio e anota o inteiro b impresso nesse cartão.
- O mestre do jogo calcula as diferenças absolutas entre $a[i]$ e b para cada i entre 0 e $n-1$. Seja S a soma destas diferenças absolutas.
- Na ronda atual, o mestre do jogo dá ao Ringo um prémio com um valor igual a S.
- Os bilhetes no conjunto são descartados e não podem ser usados em rondas futuras.

Os bilhetes que sobrarem na mochila do Ringo depois das k rondas do jogo são descartados.

Ao observar com atenção, o Ringo percebeu que o jogo é uma fraude! Na verdade existe uma impressora dentro da caixa de sorteio. Em cada ronda, o mestre do jogo encontra um inteiro b que minimiza o valor do prémio para essa ronda. O valor escolhido pelo mestre do jogo é impresso no cartão especial dessa ronda.

Com toda esta informação, o Ringo pretende alocar os bilhetes para as rondas do jogo. Isto é, ele quer selecionar o conjunto de bilhetes a usar em cada ronda de forma a maximizar o valor total dos prémios.

Detalhes de implementação

Deves implementar a seguinte função:

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

- k: o número de rondas.
- x: uma matriz $n \times m$ descrevendo os inteiros em cada bilhete. Os bilhetes de cada cor vêm ordenados por ordem não decrescente dos seus inteiros.
- Esta função é chamada exatamente uma vez.
- Esta função deve fazer exatamente uma chamada a `allocate_tickets` (ver abaixo), descrevendo k conjuntos de bilhetes, um para cada ronda. A alocação deve maximizar o valor total dos prémios.
- Esta função deve retornar o máximo valor total dos prémios.

A função `allocate_tickets` está definida como:

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

- s: uma matriz $n \times m$. O valor de $s[i][j]$ deve ser r se o bilhete j da cor i for usado no conjunto da ronda r do jogo, ou -1 caso nunca seja usado.
- Para cada $0 \leq i \leq n-1$, em $s[i][0], s[i][1], \ldots, s[i][m-1]$ cada valor $0, 1, 2, \ldots, k-1$ deve ocorrer exatamente uma vez e todas as restantes entradas devem ser -1.
- Se existirem várias alocações que resultem no mesmo máximo valor total dos prémios, podes reportar qualquer uma.

Exemplos

Exemplo 1

Considera a seguinte chamada:

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

Isto significa que:

- há $k = 2$ rondas;
- os inteiros impressos nos bilhetes de cor 0 são 0, 2 e 5, respetivamente;
- os inteiros impressos nos bilhetes de cor 1 são 1, 1 e 3, respetivamente.

Uma alocação possível que dá o máximo valor total nos prémios é:

- Na ronda 0, o Ringo escolhe o bilhete 0 da cor 0 (com o inteiro 0) e o bilhete 2 da cor 1 (com o inteiro 3). O valor mais baixo possível para o prémio desta ronda é 3. Por exemplo, o mestre do jogo pode escolher $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- Na ronda 1, o Ringo escolhe o bilhete 2 da cor 0 (com o inteiro 5) e o bilhete 1 da cor 1 (com o inteiro 1). O valor mais baixo possível para o prémio desta ronda é 4. Por exemplo, o mestre do jogo pode escolher $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Assim, o valor total dos prémios seria $3 + 4 = 7$.

Para reportar esta alocação, a função `find_maximum` deve fazer a seguinte chamada a `allocate_tickets`:

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

Finalmente, a função `find_maximum` deve retornar 7.

Exemplo 2

Considera a seguinte chamada:

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

Isto significa que:

- só há uma ronda;
- os inteiros impressos nos bilhetes de cor 0 são 5 e 9, respetivamente;
- os inteiros impressos nos bilhetes de cor 1 são 1 e 4, respetivamente;
- os inteiros impressos nos bilhetes de cor 2 são 3 e 6, respetivamente;
- os inteiros impressos nos bilhetes de cor 3 são 2 e 7, respetivamente.

Uma alocação possível que dá o máximo valor total nos prémios é:

- Na ronda 0, o Ringo escolhe o bilhete 1 da cor 0 (com o inteiro 9), o bilhete 0 da cor 1 (com o inteiro 1), o bilhete 0 da cor 2 (com o inteiro 3) e o bilhete 1 da cor 3 (com o inteiro 7). O valor mais baixo possível para o prémio desta ronda é 12, quando o mestre do jogo escolhe $b = 3$: $|3 - 9| + |3 - 1| + |3 - 3| + |3 - 7| = 6 + 2 + 0 + 4 = 12$.

Para reportar esta solução, a função `find_maximum` deve fazer a seguinte chamada a `allocate_tickets`:

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

Finalmente, a função `find_maximum` deve retornar 12.

Restrições

- $2 \leq n \leq 1500$ e n é par.
- $1 \leq k \leq m \leq 1500$
- $0 \leq x[i][j] \leq 10^9$ (para todo $0 \leq i \leq n - 1$ e $0 \leq j \leq m - 1$)
- $x[i][j - 1] \leq x[i][j]$ (para todo $0 \leq i \leq n - 1$ e $1 \leq j \leq m - 1$)

Subtarefas

1. (11 pontos) $m = 1$
2. (16 pontos) $k = 1$

3. (14 pontos) $0 \leq x[i][j] \leq 1$ (para todo $0 \leq i \leq n-1$ e $0 \leq j \leq m-1$)
4. (14 pontos) $k = m$
5. (12 pontos) $n, m \leq 80$
6. (23 pontos) $n, m \leq 300$
7. (10 pontos) Sem restrições adicionais.

Avaliador exemplo

O avaliador exemplo lê o input no seguinte formato:

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

O avaliador exemplo escreve o output com o seguinte formato:

- linha 1: o valor de retorno de `find_maximum`
- linha $2 + i$ ($0 \leq i \leq n-1$): $s[i][0]$ $s[i][1]$ $\ldots$ $s[i][m-1]$