Bilhetes de Carnaval (tickets)

Ringo está em um carnaval em Singapura. Ele tem alguns bilhetes de prêmio em sua bolsa, os quais ele gostaria de usar no jogo da barraca de prêmios. Cada bilhete vem com uma de n cores e possui um inteiro não negativo impresso nele. Os inteiros impressos em bilhetes diferentes podem ser iguais. Devido a uma peculiaridade nas regras do carnaval, é garantido que n é **par**.

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

O jogo de prêmios é jogado em k rodadas, numeradas de 0 a $k-1$. Cada rodada é jogada na seguinte ordem:

- Da sua bolsa, Ringo seleciona um **conjunto** de n bilhetes, um bilhete de cada cor. Ele então entrega o conjunto para o 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 puxa uma carta especial de uma caixa de sorteio e anota o inteiro b impresso nesta carta.
- O mestre do jogo calcula as diferenças absolutas entre $a[i]$ e b para cada i de 0 a $n-1$. Seja S a soma dessas diferenças absolutas.
- Para esta rodada, o mestre do jogo dá a Ringo um prêmio com valor igual a S.
- Os bilhetes no conjunto são descartados e não podem ser usados em rodadas futuras.

Os bilhetes restantes na bolsa de Ringo após k rodadas do jogo são descartados.

Olhando de perto, Ringo percebeu que o jogo de prêmios é manipulado! Há na verdade uma impressora dentro da caixa de sorteio. Em cada rodada, o mestre do jogo encontra um inteiro b que minimiza o valor do prêmio daquela rodada. O valor escolhido pelo mestre do jogo é impresso na carta especial para aquela rodada.

Tendo todas estas informações, Ringo gostaria de alocar bilhetes às rodadas do jogo. Isto é, ele deseja selecionar o conjunto de bilhetes para usar em cada rodada de forma a maximizar o valor total dos prêmios.

Detalhes de implementação

Você deve implementar o seguinte procedimento:

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

- k: o número de rodadas.
- x: um $n \times m$ array descrevendo os inteiros em cada bilhete. Bilhetes de cada cor estão ordenados de forma não decrescente de acordo com seus inteiros.
- Esse procedimento é chamado exatamente uma vez.
- Esse procedimento deve realizar exatamente uma chamada a `allocate_tickets` (veja abaixo), descrevendo k conjuntos de bilhetes, um para cada rodada. A alocação deve maximizar o valor total dos prêmios.
- Esse procedimento deve retornar o valor total máximo dos prêmios.

O procedimento `allocate_tickets` é definido a seguir:

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

- s: um $n \times m$ array. O valor de $s[i][j]$ deve ser r se o bilhete j da cor i é usado no conjunto da rodada r do jogo, ou -1 se ele nunca for usado.
- Para cada $0 \leq i \leq n-1$, entre $s[i][0], s[i][1], \ldots, s[i][m-1]$ cada valor $0, 1, 2, \ldots, k-1$ deve aparecer exatamente uma vez, e todos os outros números devem ser -1.
- Se existir múltiplas alocações resultando no valor total máximo dos prêmios, é permitido reportar qualquer uma delas.

Exemplos

Exemplo 1

Considere a seguinte chamada:

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

Isso significa que:

- há $k = 2$ rodadas;
- os inteiros impressos nos bilhetes da cor 0 são 0, 2 e 5, respectivamente;
- os inteiros impressos nos bilhetes da cor 1 são 1, 1 e 3, respectivamente.

Uma possível alocação que resulta no valor total máximo dos prêmios é:

- Na rodada 0, Ringo seleciona o bilhete 0 da cor 0 (com o inteiro 0) e o bilhete 2 da cor 1 (com o inteiro 3). O menor valor possível do prêmio nessa rodada é 3. E.g., o mestre do jogo pode escolher $b = 1$: $|1 - 0| + |1 - 3| = 1 + 2 = 3$.
- Na rodada 1, Ringo seleciona o bilhete 2 da cor 0 (com o inteiro 5) e o bilhete 1 da cor 1 (com o inteiro 1). O menor valor possível para o prêmio nessa rodada é 4. E.g., o mestre do jogo pode escolher $b = 3$: $|3 - 1| + |3 - 5| = 2 + 2 = 4$.
- Então, o valor total dos prêmios seria $3 + 4 = 7$.

Para reportar esta alocação, o procedimento `find_maximum` deve realizar a seguinte chamada a `allocate_tickets`:

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

Finalmente, o procedimento `find_maximum` deve retornar 7.

Exemplo 2

Considere a seguinte chamada:

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

Isso significa que:

- há apenas uma rodada,
- os inteiros impressos nos bilhetes da cor 0 são 5 e 9, respectivamente;
- os inteiros impressos nos bilhetes da cor 1 são 1 e 4, respectivamente;
- os inteiros impressos nos bilhetes da cor 2 são 3 e 6, respectivamente;
- os inteiros impressos nos bilhetes da cor 3 são 2 e 7, respectivamente.

Uma possível alocação que resulta no valor total máximo dos prêmios é:

- Na rodada 0, 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 menor valor possível para o prêmio nesta rodada 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, o procedimento `find_maximum` deve realizar a seguinte chamada a `allocate_tickets`:

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

Finalmente, o procedimento `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) $0 \le x[i][j] \le 1$ (para todo $0 \le i \le n-1$ e $0 \le j \le m-1$)
4. (14 pontos) $k = m$
5. (12 pontos) $n, m \le 80$
6. (23 pontos) $n, m \le 300$
7. (10 pontos) Nenhuma restrição adicional.

Corretor Exemplo

O corretor exemplo lê a entrada no seguinte formato:

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

O corretor exemplo escreve sua resposta no seguinte formato:

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