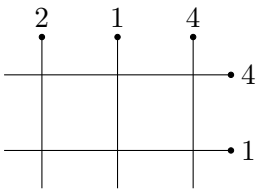
Political cost (cost)

Na cidade onde moras existem N ruas que vão de este para oeste (da rua 0 à rua $N-1$) e M avenidas que vão de norte para sul (da avenida 0 para a avenida $M-1$). TOdas as ruas e avenidas têm um dado *peso político*, que é a importância do mais importante cidadão que aí vive. Representamos os pesos políticos com dois arrays $A[0 \ldots N-1]$ e $B[0 \ldots M-1]$ com inteiros de 1 até K. A figura seguinte representa uma cidade com 2 ruas e 3 avenidas, com pesos políticos de $A = [1, 4]$ e $B = [2, 1, 4]$, respetivamente:



O presidente da câmara quer organizar um cortejo pela cidade. Se o cortejo atravessar a intersecção da rua x com a avenida y, o tráfego de ambas as ruas será perturbado, e o presidente irá sofrer um *peso político* de $\max(A[x], B[y])$. Se o cortejo atravessar múltiplas intersecções, o custo político será o *máximo* do peso político de cada intersecção. Nota que os custos não são somados: o que interessa não é quantas pessoas o cortejo incomoda, mas quão importante é o mais importante cidadão incomodado.

A *distância política* entre duas intersecções é o menor *peso político* de um cortejo que sai da primeira intersecção e chega à segunda intersecção. A tua tarefa é calcular a soma das distâncias políticas entre todos os pares de intersecções na cidade.

Implementação

Deves submeter um único ficheiro de código `.cpp`.

> ☞ Entre os ficheiros do problema encontrarás um template `cost.cpp` com um exemplo de implementação.

Tens de implementar a seguinte função:

C++
```cpp
int solve(int N, int M, int K, vector<int> A, vector<int> B);
```

- O inteiro N representa o número de ruas de este para oeste.
- O inteiro M representa o número de avenidas de norte para sul.
- O array A, indexado de 0 até $N-1$, contém os valores A_0, A_1, ..., A_{N-1}, onde A_i é o peso político da rua i.
- O array B, indexado de 0 até $M-1$, contém os valores B_0, B_1, ..., B_{N-1}, onde B_i é o peso político da avenida i.
- A função deve devolver a soma das distâncias políticas entre todos os possíveis pares de intersecções, **modulo** 1000003.

O avaliador irá chamar a função `solve` e irá escrever o valor devolvido no ficheiro de output.

Avaliador Padrão

O diretório do problema contém uma versão simplificada do avaliador oficial, que podes usar para testar o teu problema localmente. O avaliador exemplo lê os dados de input de `stdin`, chama as funções que deves implementar, e finalmente escreve o output para `stdout`.

O input é feito de 3 linhas, contendo:

- Linha 1: os inteiros N, M e K.
- Linha 2: os inteiros A_i, separados por espaços.
- Linha 3: os inteiros B_i, separados por espaços.

O output é feito de uma única linha, contendo o valor devolvido pela função `solve`.

Restrições

- $1 \leq N \leq 3 \times 10^5$.
- $1 \leq M \leq 3 \times 10^5$.
- $1 \leq K \leq N + M$.
- $1 \leq A_i \leq K$ para $i = 0 \ldots N - 1$.
- $1 \leq B_i \leq K$ para $i = 0 \ldots M - 1$.

Pontuação

O teu program será testado num conjunto de casos de teste agrupados por subtarefa. Para obteres a pontuação associada a uma subtarefa, tens de resolver corretamente todos os casos de teste dessa subtarefa.

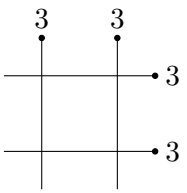
- **Subtarefa 1 [0 pontos]**: Casos de exemplo.
- **Subtarefa 2 [10 pontos]**: $N \leq 10^1, M \leq 10^1$.
- **Subtarefa 3 [10 pontos]**: $N \leq 10^2, M \leq 10^2$.
- **Subtarefa 4 [10 pontos]**: $N = 1, M \leq 10^4$.
- **Subtarefa 5 [10 pontos]**: $N = 1, M \leq 10^5$.
- **Subtarefa 6 [10 pontos]**: $N \leq 10^3, M \leq 10^3$.
- **Subtarefa 7 [10 pontos]**: $N \leq 10^4, M \leq 10^4$.
- **Subtarefa 8 [10 pontos]**: $N \leq 10^5, M \leq 10^5$ e os arrays A e B são não decrescentes, isto é, se $i < j$, então $A_i \leq A_j$ e $B_i \leq B_j$.
- **Subtarefa 9 [10 pontos]**: $N \leq 10^5, M \leq 10^5, K \leq 10^1$.
- **Subtarefa 10[10 pontos]**: $N \leq 10^5, M \leq 10^5$.
- **Subtarefa 11[10 pontos]**: Nenhuma restrição adicional.

Exemplos

stdin	stdout
2 2 4 3 3 3 3	48
1 3 4 2 2 3 1	25
2 3 5 1 4 2 1 4	135

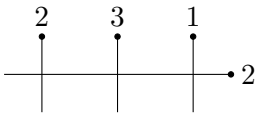
Explicação

No **primeiro caso de exemplo**, temos uma cidade com 2 ruas e 2 avenidas, todas com um peso político de 3:



Existem 16 difeerntes pares de intersecções. Como a distância política entre cada para de intersecções é 3, a solução é $3 \cdot 16 = 48$.

No **segundo caso de exemplo** existem 1 rua e 3 avenidas, com pesos políticos $A = [2]$ e $B = [2, 3, 1]$ respetivamente:



Existem 9 pares de intersecções. Três destes pares começam e terminam na mesma intersecção, e têm distâncias políticas de 2, 3 e 2 respetivamente (a avenida mais à direita tem peso político de 1, mas o peso político da única rua é 2, pelo que a distância política de qualquer cortejo é pelo menos de 2). Para cada um dos restantes pares de intersecções, o cortejo que as junta deve atravessar a avenida do meio, e por isso tem de ter distância política de 3. Portanto, a soma total é $2 + 3 + 2 + 6 \cdot 3 = 25$.

O **terceiro caso de exemplo** corresponde ao exemplo dado no enunciado do problema. Aqui existem 2 ruas e 3 avenidas. Com alguma paciência, podes verificar que a soma das distâncias políticas dos 36 pares de intersecções é de 135.