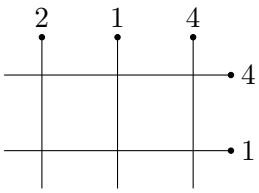
Political cost (cost)

En la ciudad donde vives hay N calles que van de Este a Oeste (de la Calle 0 a la Calle $(N-1)$) y M avenidas que van de Norte a Sur (de la Avenida 0 a la Avenida $(M-1)$). Cada calle y cada avenida tiene un *peso político*, que es la importancia del ciudadano más importante que vive en ella. Representamos los pesos políticos como dos arrays $A[0 \ldots N-1]$ y $B[0 \ldots M-1]$ de enteros de 1 a K. El siguiente gráfico representa una ciudad con 2 calles y 3 avenidas, con pesos políticos de $A = [1, 4]$ y $B = [2, 1, 4]$ respectivamente.



El alcalde quiere organizar un desfile por la ciudad. Si el desfile pasa por la intersección de la Calle x con la Avenida y, el tráfico de ambas vías se verá interrumpido, y el alcalde incurrirá en un *coste político* de $\max(A[x], B[y])$. Si el desfile atraviesa varias intersecciones, el coste político será el *máximo* del coste político de cada intersección. Obsérvese que los costes no se suman: lo que importa no es a cuánta gente molesta el desfile, sino lo importante que es el ciudadano más importante al que molesta el desfile.

La *distancia política* entre dos intersecciones es el menor coste político de un desfile que sale de la primera intersección y llega a la segunda intersección. Su tarea consiste en calcular la suma de las distancias políticas entre todos los pares de intersecciones de la ciudad.

Implementación

Deberá presentar un único fichero fuente `.cpp`.

> ☞ Entre los adjuntos de esta tarea encontrará una plantilla `cost.cpp` con un ejemplo de implementación.

Tienes que implementar la siguiente función:

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

- El número entero N representa el número de calles de Este a Oeste.
- El número entero M representa el número de avenidas de Norte a Sur.
- El array A, indexado de 0 a $N-1$, contiene los valores A_0, A_1, …, A_{N-1}, donde A_i es el peso político de la calle i-ésima de Este a Oeste.
- El array B, indexado de 0 a $M-1$, contiene los valores B_0, B_1, …, B_{N-1}, donde B_i es el peso político de la calle i-ésima de Norte a Sur.
- La función debe devolver la suma de las distancias políticas entre todos los pares posibles de intersecciones, **modulo** 1000003.

El grader llamará a la función `solve` e imprimirá su valor de retorno en el archivo de salida.

Sample Grader

El directorio de la tarea contiene una versión simplificada del grader jurado, que puede utilizar para

probar su solución localmente. El grader simplificado lee los datos de entrada de `stdin`, llama a las funciones que debe implementar, y finalmente escribe la salida a `stdout`.

La entrada se compone de 3 líneas, que contienen:

- Línea 1: los enteros N, M y K.
- Línea 2: los enteros A_i, separados por espacios.
- Línea 3: los enteros B_i, separados por espacios.

La salida se compone de una sola línea, que contiene el valor devuelto por el función `solve`.

Restricciones

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

Puntuación

Tu programa se probará en un conjunto de casos de prueba agrupados por subtareas. Para obtener la puntuación asociada a una subtarea, deberá resolver correctamente todos los casos de prueba que contiene.

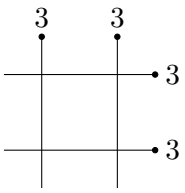
- **Subtask 1 [0 puntos]**: Sample test cases.
- **Subtask 2 [10 puntos]**: $N \leq 10^1, M \leq 10^1$.
- **Subtask 3 [10 puntos]**: $N \leq 10^2, M \leq 10^2$.
- **Subtask 4 [10 puntos]**: $N = 1, M \leq 10^4$.
- **Subtask 5 [10 puntos]**: $N = 1, M \leq 10^5$.
- **Subtask 6 [10 puntos]**: $N \leq 10^3, M \leq 10^3$.
- **Subtask 7 [10 puntos]**: $N \leq 10^4, M \leq 10^4$.
- **Subtask 8 [10 puntos]**: $N \leq 10^5, M \leq 10^5$ y los arrays A y B son no decrecientes, es decir, si $i < j$, entonces $A_i \leq A_j$ y $B_i \leq B_j$.
- **Subtask 9 [10 puntos]**: $N \leq 10^5, M \leq 10^5, K \leq 10^1$.
- **Subtask 10[10 puntos]**: $N \leq 10^5, M \leq 10^5$.
- **Subtask 11[10 puntos]**: No hay restricciones adicionales.

Ejemplos de entrada/salida

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

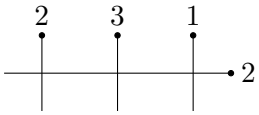
Explicación

En el **primer caso de muestra**, tenemos una ciudad con 2 calles y 2 avenidas, todas ellas de peso político 3:



Hay 16 pares de intersecciones. Dado que la distancia política entre cada par de intersecciones es 3, la solución es $3 \cdot 16 = 48$.

En el **segundo caso de muestra** hay 1 calle y 3 avenidas, con pesos políticos $A = [2]$ y $B = [2, 3, 1]$ respectivamente:



Hay 9 pares de intersecciones. Tres de estos pares empiezan y terminan en la misma intersección, y tienen distancias políticas de 2, 3 y 2 respectivamente (la avenida de más a la derecha tiene un peso político de 1, pero el peso político de la única calle es de 2, por lo que la distancia política de cualquier desfile es de al menos 2). Para cada par de intersecciones restante, el desfile que las une debe cruzar la avenida central y, por tanto, debe tener una distancia política de 3. Así que la suma total es de $2 + 3 + 2 + 6 \cdot 3 = 25$.

El **tercer caso de ejemplo** corresponde al ejemplo dado en el enunciado del problema. Aquí hay 2 calles y 3 avenidas. Usted puede comprobar, con un poco de paciencia, que la suma de las distancias políticas de los 36 pares de intersecciones es 135.