Trilogía policíaca

Rosa es una escritora de novelas policiacas. Ahora está escribiendo una trilogía, y ha decidido que cada libro tiene que ser mucho más largo que el anterior. Cuando en las entrevistas le han preguntado a que se refiere con "mucho", Rosa ha dicho que como mínimo tiene que tener más que el número de páginas al cuadrado del anterior libro.

Rosa está dispuesta a hacer un libro de 1 página (es muy buena explicando historias), pero en la editorial le han dicho que pueden hacer libros de hasta n páginas.

Ayuda a Rosa a encontrar cuantas trilogías (a, b, c) puede hacer. Donde a, b y c son el número de páginas de cada libro.

Por ejemplo si $n = 200$, $(1, 10, 200)$ sería una trilogía aceptable, mientras que $(1, 10, 100)$ no.

Entrada

Un único valor n, con el número máximo de páginas que puede hacer la editorial.

Restricciones

- $1 \leq n \leq 200$

Salida

Un único entero con el número de trilogías que se pueden imprimir.

Ejemplo

Entrada	Salida
1	0
5	1
10	7
30	71

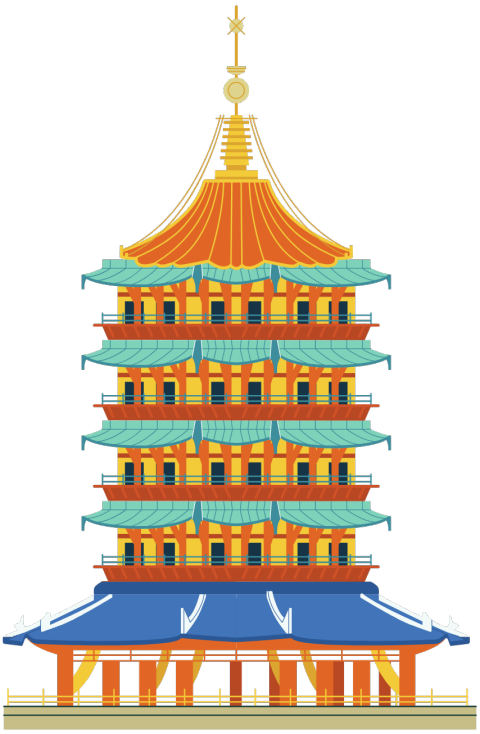
Olimpiada Informática Femenina III
Final día 2
`pagoda`

La pagoda más alta

La pagoda es un estilo de edificación común en varios paises asiaticos. Hecha de madera o piedra, consiste en varias plantas con un tejado cada una.
Cuando se construye una pagoda se tienen que tener en cuenta varios factores:

- Cada nivel consta de un piso y un tejado.
- En un mismo nivel el tejado tiene que ser estrictamente mayor que el piso.
- Para cada nivel, se da que todo piso superior al actual ha de tener un tamaño igual o inferior al del piso actual. Esto mismo ha de cumplirse para los tejados, donde un tejado en un nivel superior no podrá nunca tener un tamaño mayor que el de un nivel inferior.
- Tiene que haber como mínimo 3 niveles.
- El número total de niveles siempre es impar.

Dadas las medidas de los posibles pisos y de los posibles tejados, queremos saber cuánto mide la pagoda más alta (o sea, con más niveles) que puede ser construida siguiendo las normas establecidas.



Olimpiada Informática Femenina III
Final día 2
`pagoda`

Entrada

- Una línea con dos enteros: n y m.
- Una línea con n enteros $a_1, \ldots, a_n$ con las medidas de los posibles pisos en orden creciente.
- Una línea con m enteros $b_1, \ldots, b_m$ con las medidas de los posibles tejados en orden creciente.

Restricciones

- $1 \leq n, m \leq 1000$
- $1 \leq a_i, b_i \leq 10^7$
- 10 puntos: todos los pisos tienen la misma medida y todos los tejados tienen la misma medida.
- 20 puntos: todos los pisos tienen la misma medida.
- 20 puntos: todos los tejados tienen la misma medida.
- 50 puntos: restricciones originales

Salida

Un único entero con el número de niveles que tendrá la pagoda más alta, o 0 si no existe una pagoda que satisfaga las condiciones.

Ejemplo

Entrada	Salida
5 5 2 3 6 7 9 2 4 6 8 8	3
6 6 101 102 103 104 105 106 1 2 8 9 9 10	0

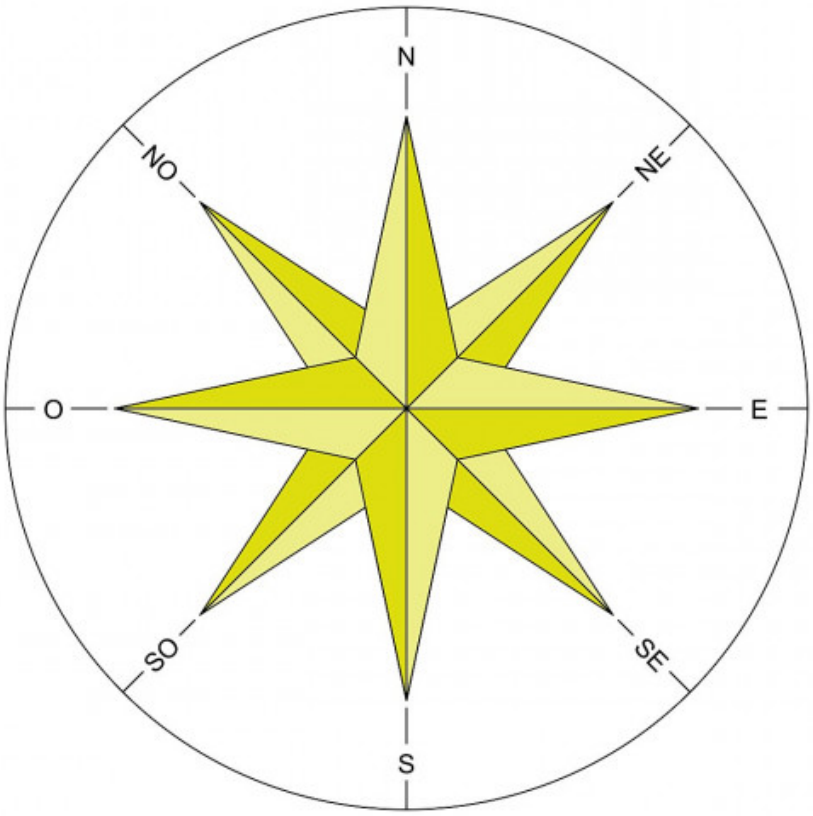
En el primer ejemplo, la pagoda podría ser:

1. Nivel 1: Piso 7 Tejado 8
2. Nivel 2: Piso 3 Tejado 6
3. Nivel 3: Piso 2 Tejado 4

En el segundo ejemplo, es imposible construir una pagoda que satisfaga la restricción de que en todo nivel, el piso sea menor que su tejado.

Tesoro

Hay un tesoro con un localizador. Tienes un dispositivo que te permite saber en qué dirección está (8 direcciones en un tablero de $N \times N$: `N`, `NE`, `E`, `SE`, `S`, `SO`, `O`, `NO`), pero no puedes usarlo muchas veces ya que cada vez que lo usas gasta batería. Puedes moverte a cualquier punto y tienes un máximo de Q consultas antes de quedarte sin batería. Podrás determinar las coordenadas del tesoro?



Olimpiada Informática Femenina III
Final día 2
`tesoro`

El tesoro está en las coordenadas (x^*, y^*), desconocidas para nosotros. Si usas el localizador en las coordenadas (x, y), las siguientes respuestas son posibles, dependiendo de la dirección del tesoro respecto a estas coordenadas:

- `TESORO` si $x^* = x$ e $y^* = y$
- `N` si $x^* = x$ e $y^* > y$
- `S` si $x^* = x$ e $y^* < y$
- `E` si $x^* > x$ e $y^* = y$
- `O` si $x^* < x$ e $y^* = y$
- `NE` si $x^* > x$ e $y^* > y$
- `NO` si $x^* < x$ e $y^* > y$
- `SE` si $x^* > x$ e $y^* < y$
- `SO` si $x^* < x$ e $y^* < y$

Entrada y salida

En la primera línea tendrás dos enteros Q, N.
A continuación empieza la interacción. **Este es un problema interactivo.**
Por lo tanto, has de refrescar la salida cada vez que imprimas datos (`cout << endl` o `cout << flush` en C++, `System.out.flush()` en Java, `stdout.flush()` en Python).
El juez quedará a la espera de tu primer uso del localizador. Cada vez que quieras usarlo imprime las coordenadas enteras x, y desde donde quieres usarlo.
Tras cada uso, el juez escribirá una línea con la salida del localizador.
Si usas el localizador en la posición del tesoro ganarás automáticamente, recibiendo `TESORO` como respuesta del juez. Tu programa debe acabar tras esto (si no acaba y queda a la espera de leer algo más, excederá el límite de tiempo).

Restricciones

- $20 \leq Q \leq 200$
- $1 \leq N \leq 10^5$
- $1 \leq x^*, y^* \leq N$
- 21 puntos: $Q \geq 50$, $N \leq 10$
- 29 puntos: $Q \geq 200$, $N \leq 100$
- 8 puntos: $Q \geq 200$, $N \leq 500$
- 32 puntos: $Q \geq 40$
- 10 puntos: sin restricciones adicionales

Ejemplos de interacción

Entrada	Salida
200 2 NE TESORO	1 1 2 2

Borrando elementos

Izan les ha propuesto un juego a Joana y Lucía. Este consiste en, dada una lista de enteros $A = (a_1, \ldots, a_n)$, transformarla con el objetivo de que la lista transformada tenga un coste menor que el de su contrincante.

Dada una lista de enteros $A = (a_1, \ldots, a_n)$, decimos que su penalización es la suma de sus diferencias, es decir, $(a_2 - a_1) + (a_3 - a_2) + \cdots + (a_n - a_{n-1})$. Por ejemplo, si tienemos $A = (1, 2, -3, 4)$ sus diferencias son $(1, -5, 7)$, sumando una penalización de 3.

Hay una única transformación válida: eliminar elementos de la lista, que ambas podrán hacer tantas veces como quieran. Es decir, Lucía y Joana tendrán cada una una copia individual de la lista y podrán, sin ver lo que hace la otra, borrar tantos elementos como quieran de su propia lista, con la restricción de dejar mínimo dos elementos en la lista.

El coste de la lista final será igual a la suma de los elementos quitados más la penalización de la lista final. Y tú, ¿serías capaz de ganar en este juego? Redacta un programa que, dado la lista inicial, calcule el menor coste posible de la lista final.

Entrada

Una línea con un entero T, el número de casos de entrada. Para cada caso de entrada:

Una línea con un entero n, la longitud de la lista A, seguida de otra línea con n enteros separados por un espacio: $a_1, \ldots, a_n$.

Restricciones

- $1 \leq T \leq 100$
- $2 \leq n \leq 10^6$
- La suma de las n de todos los casos de entrada de un mismo archivo (llamémosla S) cumple que $S \leq 10^6$
- $-10^9 \leq a_i \leq 10^9$ para $i = 1, \ldots, n$
- 19 puntos: $2 \leq n \leq 10$, $S \leq 100$
- 23 puntos: $2 \leq n \leq 1000$, $S \leq 10000$
- 31 puntos: $a_i \geq 0$ para $i = 1, \ldots, n$
- 27 puntos: sin restricciones adicionales

Salida

Una línea con el mínimo coste posible.

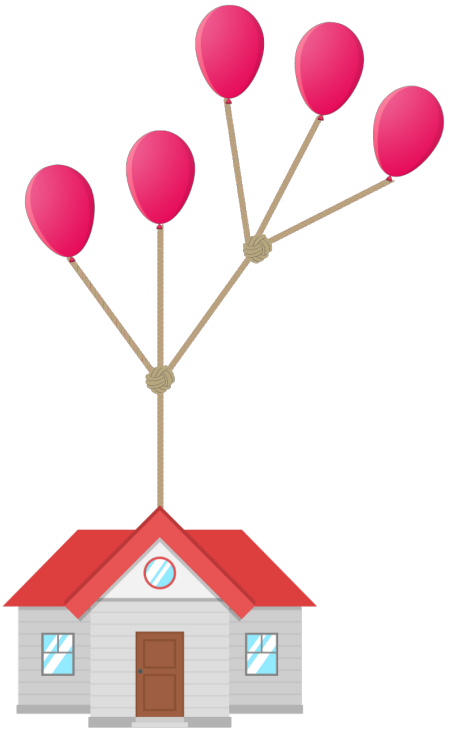
Ejemplos

Entrada	Salida
1 5 1 2 -3 4 5	1
1 6 2 -3 -1 4 5 3	-3

Globos de Up 2

La casa de Up tiene un montón (concretamente, n) de globos atados. Cada globo puede levantar a_i kg y en total la casa pesa b kg. Hay un sistema algo complejo de globos, cuerdas y nudos para levantar la casa.

De cada globo cuelga una cuerda y su otro extremo puede estar atado a un nudo o a la casa. Hay una única cuerda atada a la casa (la cuerda 0). Los nudos son puntos donde se juntan los extremos de varias cuerdas. Por otra parte, no todas las cuerdas han de tener en su extremo atado a un globo. Puede haber cuerdas conectando dos nudos o la casa a un nudo, por ejemplo, como se ve en el diagrama:



El problema que tenemos es que la suma de las a_i es mayor que b, por lo que la casa no para de subir. Acabamos de llegar a la altura deseada y queremos quedarnos sin subir ni bajar, y esto se consigue solo si la suma de las a_i es exactamente b. Podemos cortar algunas cuerdas de modo que todos los globos cuya conexión con la casa dependa de esas cuerdas se perderán y, por lo tanto, la suma de las a_i decrecerá. Cada cuerda tiene un tiempo de corte concreto r_j, que es el tiempo que tenemos que dedicar para cortar esa cuerda. Queremos saber si podemos conseguir dejar la casa a esta altura y, en tal caso, cuánto tiempo vamos a tardar en hacerlo, buscando tardar lo menos posible.

Olimpiada Informática Femenina III
Final día 2
`globos2`

Entrada

- Una línea con cuatro enteros: n (el número de globos), m (el número de cuerdas), t (el número de nudos) y b (el peso total de la casa).
- Una línea con m enteros separados por un espacio: $r_0, \ldots, r_{m-1}$ (el tiempo que tardamos en cortar cada cuerda).
- Una línea con n enteros separados por un espacio: $a_0, \ldots, a_{n-1}$ (cuánto peso levanta cada globo).
- Una línea con n enteros: $c_0, \ldots, c_{n-1}$, el índice de la cuerda atada a cada globo (empezamos a indexarlas en 0).
- t líneas, donde la i-ésima línea nos dirá qué cuerdas tienen un extremo atado al i-ésimo nudo. Esta línea empezará con un entero k_i (el número de cuerdas), y lo seguirán k_i enteros $p_1, \ldots, p_{k_i}$: los índices de las cuerdas con un extremo atado a este nudo.

Restricciones

- $1 \leq n, a_i, r_j \leq 10^4$
- $1 \leq m \leq 2 \cdot 10^4$
- $1 \leq \sum a_i - b \leq 10^4$
- Se garantiza que cortar cualquier cuerda soltará al menos un globo
- 18 puntos: $n \leq 10$
- 26 puntos: $a_i = 1, r_j = 1$
- 9 puntos: $a_i = 1$
- 19 puntos: $n, b \leq 100$
- 28 puntos: sin restricciones adicionales

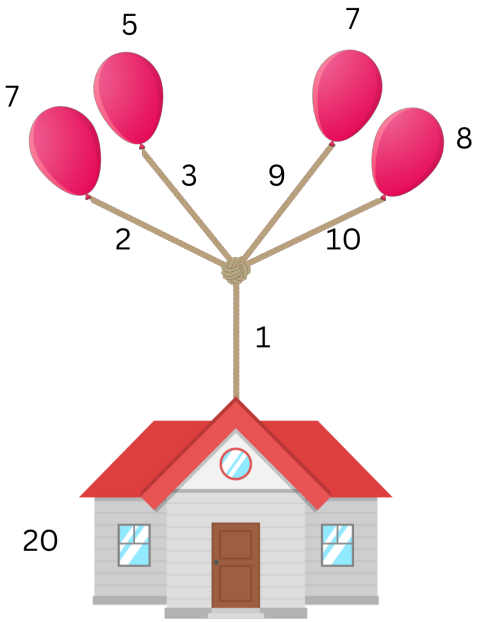
Salida

Una línea con el mínimo tiempo necesario para cortar las cuerdas o un -1 si no es posible hacerlo cumpliendo con los requisitos del enunciado.

Olimpiada Informática Femenina III
Final día 2
`globos2`

Ejemplo

Entrada	Salida
4 5 1 20 1 2 3 9 10 7 5 7 8 1 2 3 4 5 0 1 2 3 4	2



Buscamos que los globos levanten un peso de exactamente 20kg y ahora mismo levantan 27kg. Las únicas posibilidades son cortar el primer o el tercer globo y, como el primero tardaremos menos en cortarlo, será la opción que escojamos.

Olimpiada Informática Femenina III
Final día 2
`topos`

Topos

Manuel es un fotógrafo de naturaleza que quiere sacar la mejor foto de topos que se ha hecho nunca. Su cámera está delante de n madrigueras de topos que están en línea, preparada para sacar una foto. Manuel se ha dado cuenta que las madrigueras están conectadas por túneles de una sola dirección, de tal forma que la $i-$ésima madriguera está conectada con exactamente un túnel que va de la $i-$ésima madriguera a la p_i-ésima madriguera. Se cumple ademàs que a cada madriguera llega exactamente uno de estos túneles.

Hay k topos repartidos por las madrigueras, a cada madriguera puede haber como máximo un solo topo. Cada minuto, si hay un topo en la $i-$ésima madriguera se irá por el túnel hacia la p_i-ésima madriguera. Una vez han hecho el cambio todos los topos hacen un salto a la superficie todos a la vez durante un segundo y vuelven a entrar a las madrigueras.

La cámera de Manuel está programada para sacar una foto cada minuto, justo en el momento que los topos saltan. El problema es que no puede sacar una foto de todas la madrigueras, solo puede sacar una foto de un intervalo de madrigueras $[l, r]$, desde la madriguera $l-$ésima a la $r-$ésima. Se cumple además que $r-l+1 = k$, es decir, que el número de madrigueras que salen en la foto es el mismo que el número de topos. Manuel se pregunta si es posible que en algun momento en la foto salgan todos los topos saltando juntos.

Entrada

Una línea con dos enteros n, k.
La siguente línea contiene n enteros positivos indicando los p_i. Será una permutación de los enteros de 1 a n.
La tercera línea contine k enteros positivos t_i indicando la posición inicial de cada topo, todas son distintas.
La última línea contien los enteros l, r, que cumplen $r - l + 1 = k$.

Restricciones

- $1 \leq n, k \leq 4 \cdot 10^5$
- $1 \leq l \leq r \leq n$
- $1 \leq t_i, p_i \leq n$
- 22 puntos: $k = 1$
- 14 puntos: $p_i = i + 1$ para $1 \leq i < n$ y $p_n = 1$
- 6 puntos: $1 \leq n \leq 50$
- 11 puntos: $k \leq 10$
- 29 puntos: $1 \leq n \leq 500$
- 18 puntos: Sin restricciones

Salida

Una línea con la respuesta: ”Si” o ”No” en función de si puede sacar la foto o no.

Ejemplos

Entrada	Salida
3 2 2 3 1 1 3 2 3	Si
4 2 2 1 4 3 1 3 2 3	No