Torneo Chileno de Programación 2025

04-10-2025

Sedes:

Universidad Austral de Chile, Valdivia

Universidad Católica del Maule, Talca

Universidad Católica del Norte, Coquimbo

Universidad de Concepción, Concepción

Universidad de Los Lagos, Puerto Montt

Universidad de Viña del Mar, Viña del Mar

Universidad Técnica Federica Santa María, Santiago

Agradecimientos

Queremos agradecer profundamente el enorme trabajo realizado por **Gabriel Carmona** en la planificación, coordinación y dirección del equipo encargado de elaborar este problemset. Sin su dedicación y esfuerzo, no hubiese sido posible llevarlo a cabo.

Asimismo, queremos reconocer a quienes propusieron ideas o colaboraron en la preparación de los problemas: **Benjamín Letelier**, **Benjamín Rubio**, **Blaz Korecic**, **Javier Oliva**, **Martín Andrighetti**, **Martín Muñoz**, **Mauricio Cari** y **Vicente Opazo**. Gracias a Gabriel y a este increíble equipo, hoy el *Torneo Chileno de Programación* cuenta con un problemset propio, creado con mucho cariño por un equipo chileno.

Agradecemos también a **ICPC Chile** por organizar esta competencia en las distintas sedes del país. Finalmente, nuestros mejores deseos para ustedes, las y los competidores. ¡Mucho éxito en la competencia!

Ignacio Muñoz

Sociedad Chilena de Programación Competitiva

A. Búsqueda chiquita

Autor: Gabriel Carmona

Tiempo límite: 2 segundos

Memoria límite: 4 megabytes

NOTA LA INUSUAL MEMORIA LÍMITE DEL PROBLEMA

¡Se abrieron las inscripciones para el Torneo Chileno de Programación (TCP)! Muchas personas se registraron para participar. Dentro de la organización está el comité de medición, en el cual trabaja Javier. Su tarea es anotar las alturas de los participantes para poder ordenarlos en la foto oficial del TCP.

Martín, encargado de acomodar a las personas en la foto, necesita consultar cuántos participantes tienen altura menor que cierta altura. Sin embargo, tiene memoria de pez dorado, así que a veces pregunta varias veces por la misma altura.

Javier, consciente de esto, quiere ayudarlo con un sistema de consultas eficiente. Pero como tiene problemas de visión, necesita que el sistema guarde todas las alturas usando poco espacio.

A último minuto, Javier se rompió las manos y no puede diseñar el sistema. Por eso te pide ayuda.

Entrada

La primera línea contiene dos enteros n y q ($1 \leq n \leq 3 \cdot 10^6$, $1 \leq q \leq 10^6$), correspondiente a la cantidad de personas inscritas al TCP y la cantidad de consultas realizadas por Martín.

Le sigue una segunda línea que contiene n enteros separados por un espacio, cada entero tiene valor entre 1 y $3 \cdot 10^6$. Cada entero corresponde a la altura de un participante. Se asegura que todas las alturas son distintas.

Finalmente, le siguen q líneas, donde cada línea contiene un entero correspondiente a la altura consultada por Martín. Cada altura consultada puede tener valor entre 1 y $3 \cdot 10^6$.

Nota: La entrada/salida de este problema es muy grande, por lo que se recomienda usar técnicas de entrada/salida rápida.

- *En C++: al inicio del programa main deberán escribir* `std::ios::sync_with_stdio(false);` `std::cin.tie(nullptr);`
- *En Java: usar* `BufferedReader` *con* `StringTokenizer` *para la entrada, y* `BufferedWriter` *o* `PrintWriter` *para la salida.*
- *En Python: usar* `sys.stdin.readline` *en lugar de* `input()` *y* `sys.stdout.write` *para la salida.*

Salida

Por cada consulta, se debe imprimir una línea con un entero, correspondiente a la cantidad de alturas menor que la altura consultada.

Ejemplos

Entrada 1	Salida 1
10 4 7 2 3 82 70 66 8 13 14 11 1 69 8 100	0 8 3 10

B. Todo al rojo

Autor: Martín Andrighetti

Tiempo límite: 1 segundo

Memoria límite: 256 megabytes

Lamentablemente, Nacho perdió los fondos del campamento de programación apostando en la ruleta. Como necesita los fondos urgentemente, Nacho buscó una manera rápida de hacer dinero, y solo conoce una: apostar en la ruleta.

La ruleta es un aparato con forma de anillo dividido en M casillas, numeradas del 1 al M en sentido horario. Notar que las casillas son cíclicas: después de la casilla i viene la casilla $i+1$, con la excepción que después de la casilla M viene la casilla 1. Cada casilla está pintada de rojo o de negro. En una de las casillas hay una bolita invisible.

El juego de la ruleta sucede en rondas. Al comenzar una ronda, Nacho realiza su apuesta e intenta adivinar si la bolita caerá en una casilla roja o negra. Luego, el operador gira la ruleta y la bolita avanza una cierta cantidad de casillas en sentido horario, cayendo en una nueva casilla. Al finalizar la ronda, Nacho gana su apuesta si la bolita cae en una celda del color que adivinó, y pierde en caso contrario.

Al principio Nacho está desorientado, sobre todo porque la bolita es invisible. Sin embargo, escucha con atención y hace una observación clave: la bolita invisible siempre avanza la misma cantidad de casillas por ronda. ¿Puedes ayudarlo a recuperar los fondos?

Entrada

En la primera línea recibirás dos enteros: M ($1 \leq M \leq 10^3$) y T ($10M \leq T \leq 10^4$), la cantidad de casillas en la ruleta y la cantidad de rondas que jugará Nacho.

La segunda línea consistirá de M caracteres, cada uno pudiendo ser `R` o `N`. Si el i-ésimo caracter es `R` entonces la i-ésima casilla es roja, y si el i-ésimo caracter es `N` entonces la i-ésima casilla es negra.

Luego, comenzará la sección *interactiva* de tu programa. ¡Presta especial atención al orden en que debes realizar las operaciones de leer e imprimir!

Interacción

Para cada una de las T rondas, debes imprimir una línea con la palabra "`ROJO`" o la palabra "`NEGRO`", representando respectivamente si apuesta al rojo o al negro.

Solo después de que imprimas esta línea, el operador girará la ruleta y debes leer una línea que contendrá la palabra "`GANASTE`" si Nacho ganó la apuesta (el color resultante coincide con la apuesta hecha), o "`PERDISTE`" si Nacho perdió la apuesta (el color resultante difiere de la apuesta hecha).

Después de T rondas tu programa debe terminar. Se considerará tu programa como correcto si Nacho **gana al menos el 90%** de las apuestas.

Ejemplos

Entrada 1	Salida 1
5 3 RNNRN PERDISTE GANASTE GANASTE	 ROJO NEGRO ROJO

Nota

El ejemplo anterior es sólo para demostrar la interacción. Se utilizan líneas en blanco para representar el orden de las de lecturas y escrituras. No debes leer ni escribir líneas en blanco en tu programa. Notar que en este ejemplo no se cumple la restricción $10M \leq T$, y además Nacho gana menos del 90% de las apuestas.

Este problema es *interactivo*, por lo que en algunos momentos es importante que imprimas antes de continuar leyendo. Algunos lenguajes no imprimen inmediatamente al realizar una operación de impresión, por lo que es necesario realizar una operación de *flush* manualmente:

- Python: Agregar el parámetro `flush` al imprimir: `print(..., flush=True)`
- C++: Utilizar `std::cout << std::flush;` tras imprimir.
- C: Utilizar `fflush(stdout);` tras imprimir.
- Otros lenguajes: Referir a la documentación.

C. Colocando al autor

Autores: Mauricio Cari, Gabriel Carmona, Benjamín Letelier

Tiempo límite: 2 segundos

Memoria límite: 256 megabytes

Como era de esperar, Cristina Ruiz seleccionó nuevamente a Coquimbo como sede del ICPC. En esta ocasión, el contest poseerá $K+1$ problemas, cada uno escrito por un autor diferente. Para asegurarse que los problemas no fueran filtrados, decidió instaurar una sencilla medida que aplicará a cada autor de un problema: encerrarlo en una casa con un computador sin internet. Felizmente, Cristina encontró un hermoso vecindario que cumple con estos requisitos.

El vecindario puede verse como un grafo ponderado simple y conexo con N casas y M calles, y para cada calle Cristina sabe cuántos minutos se demora en recorrerla. Ella también sabe que el computador ubicado en la i-ésima casa se demora C_i minutos en subir un archivo a su USB. Actualmente, K autores ya se encuentran posicionados en sus respectivas casas trabajando en sus problemas, pero a Cristina aún le falta decidir en cuál de las S posibles casas desocupadas colocar al autor encargado del I-ésimo problema de la competencia.

Lo que sí sabe es que su ruta consistirá en entrar a las casas en el mismo orden en el que los problemas serán presentados en la competencia. Luego, dentro de una casa, subirá el archivo del problema a su USB e irá a la siguiente casa. Su ruta comenzará en la casa del autor del primer problema, y terminará luego de almacenar el último problema en su USB. Obviamente, esto lo quiere hacer lo más rápido posible.

¿En cuál de las posibles S casas desocupadas se debe colocar al autor del I-ésimo problema para minimizar el tiempo de ruta de Cristina?

Entrada

La primera línea contiene los enteros N, M, K, I, S — el número de casas, el número de calles, el número de autores ya ubicados, el índice del problema del autor a ser colocado y el número de posibles casas para colocar al autor ($N \cdot (K+1) \leq 10^5$, $M \cdot (K+1) \leq 10^6$, $1 \leq I \leq (K+1)$, $2 \leq S \leq (N-K)$).

Cada una de las siguientes M líneas poseen una arista ponderada bidireccional de la forma (u, v, d), indicando que existe una calle entre las casas u y v que Cristina tarda d minutos en recorrer ($1 \leq u, v \leq N$, $1 \leq d \leq 10^9$).

La siguiente línea poseerá N enteros, donde la i-ésima posición corresponde a C_i — la cantidad de minutos que demora el computador de la casa i en subir un archivo al USB de Cristina ($1 \leq C_i \leq 10^9$).

La siguiente línea posee $K+1$ enteros diferentes, donde el i-ésimo entero k_i indica la casa donde se encuentra el autor del problema i ($1 \leq k_i \leq N$, $k_I = -1$).

La última línea posee S enteros diferentes, donde el i-ésimo entero s_i, indica una casa desocupada que puede ser elegida para colocar al autor del I-ésimo problema ($1 \leq s_i \leq N$).

Salida

Imprime un único número, indicando la posición de la casa desocupada a eligir que minimice el tiempo de ruta de Cristina. Si existen múltiples soluciones, imprime la casa con menor posición entre estas.

Ejemplos

Entrada 1	Salida 1
6 7 2 2 4 1 2 1 2 6 2 2 3 4 3 5 2 6 3 1 6 5 5 5 4 3 1 4 2 4 5 1 2 -1 5 1 3 4 6	6

Entrada 2	Salida 2
6 7 2 1 4 1 2 1 2 6 2 2 3 4 3 5 2 6 3 1 6 5 5 5 4 3 1 4 2 4 5 1 -1 2 5 1 3 4 6	1

Nota

- En el primer ejemplo, si colocamos al autor en la casa 6, el mejor tiempo de ruta de Cristina es de 15 minutos, el cuál es el mínimo posible a obtener entre las posibles opciones.

D. Roptal

Autor: Blaz Korecic

Tiempo límite: 1 segundo

Memoria límite: 256 megabytes

En la famosa serie de videojuegos *Roptal* desarrollada por la prestigiosa compañía *Palpe*, los jugadores controlan a los robots SALTA y Q-Body para resolver una serie de puzzles.

Como es de costumbre en los videojuegos de *Palpe*, se acerca la tercera entrega de la saga de *Roptal*. Esta presenta un novedoso puzzle en el que los robots deben rotar una serie de cajas de manera que un láser pueda llegar desde un inicio hasta el final y así abrir la puerta hacia el siguiente nivel.

Dado que en este puzzle todas las cajas son pequeñas y están en un mismo plano, las representaremos como *puntos* en un plano cartesiano. En particular, tenemos n puntos con coordenadas enteras, numerados del 1 al n.

Cuando el láser impacta en una caja desde cualquier dirección, este se redirige gracias a una cara especial de la caja, que siempre apunta en una dirección paralela a los ejes: **arriba, derecha, abajo o izquierda**. Por ejemplo, si el láser llega desde la izquierda a una caja cuya cara especial apunta hacia abajo, su trayectoria continuará hacia abajo a partir de esa caja.

Las cajas 1 y n son especiales: la caja 1 es el origen del láser (lo emite por su cara especial sin necesidad de recibirlo por alguna cara), mientras que la caja n es el objetivo al que el láser debe llegar (solo puede recibir el láser y no lo redirige, por lo que su cara especial es irrelevante).

Los jugadores pueden realizar la siguiente operación: escoger una caja y rotarla 90, 180 o 270 grados, cambiando la dirección hacia la que redirige el láser a otra de las direcciones posibles.

El objetivo es resolver el puzzle con la menor cantidad de operaciones. Para calcular el óptimo, los desarrolladores de *Roptal* escribieron un algoritmo, pero resultó ser muy lento, por lo que necesitan tu ayuda.

Entrada

La primera línea contiene un entero n ($2 \leq n \leq 2 \cdot 10^5$), la cantidad de cajas.

La i-ésima de las siguientes n líneas contiene dos enteros y un string x_i, y_i, d_i ($-10^9 \leq x_i, y_i \leq 10^9, d_i \in \{$`arriba`, `derecha`, `abajo`, `izquierda`$\}$), representando las coordenadas de la i-ésima caja y la dirección inicial a la que redirige el láser.

Salida

Imprime una sola línea con un número entero, indicando la mínima cantidad de operaciones para que el láser llegue hasta la caja n o `-1` si no es posible.

Ejemplos

Entrada 1	Salida 1
5 1 1 arriba 1 2 izquierda 1 3 derecha 2 2 derecha 2 3 izquierda	1

Nota

El ejemplo se puede resolver en una operación, rotando la caja que está en $(1, 2)$ de forma que redirija el láser hacia arriba.

E. Entrenando con gepeto

Autores: Benjamín Letelier, Vicente Opazo

Tiempo límite: 5 segundos

Memoria límite: 256 megabytes

Benjamín está cansado de no poder mejorar en programación competitiva, por lo que al fin hizo caso a sus amigos y le pidió ayuda a Gepeto. Como todos saben, Gepeto nunca se equivoca, asi que sus respuestas siempre son correctas aún por más tontas que suenen. En este caso, su sabia respuesta fue que, para mejorar, tenía que completar diariamente una tarea sobre una sopa de letras muy particular.

La sopa de letras consiste en una matriz de $N \times N$ letras mayúsculas del alfabeto inglés, y las palabras a encontrar se encuentran de manera horizontal ($\rightarrow$), vertical ($\downarrow$) y diagonal ($\searrow$) — únicamente como indican las flechas.

Por otro lado, la simple tarea consiste en indicarle a Gepeto si existe o no la sigla `TCP` (*Training Competitive Programming*) dentro de la sopa de letras.

Benjamín sabe que esto puede tomar tiempo si lo hace manualmente, y también es consciente de que es malo programando, por lo que te pidió ayuda para resolver la tarea encomendada por Gepeto.

Entrada

La primera línea de la entrada contiene el entero N ($1 \leq N \leq 1000$). Luego, para cada una de las siguientes N líneas, se entregará un *string* de tamaño N, con únicamente letras del alfabeto inglés, representando la sopa de letras que entregó Gepeto.

Salida

Imprime `SI`, si existe la sigla `TCP` dentro de la sopa de letras en alguna de las direcciones válidas. En caso contrario, imprime `NO`.

Ejemplos

Entrada 1	Salida 1
8 TCPTCPTT GOODLUCK PABCFPCT CHAVEFUN TDOXNOTX GIVEUPLS TRYAGAIN XXXXXTCP	SI

Nota

- En el ejemplo, la sigla `TCP` aparece de la siguiente manera:

$\rightarrow$: Se encuentra 3 veces presente (dos veces en la primera fila y una vez en la última).

$\downarrow$: No se encuentra presente.

$\searrow$: No se encuentra presente.

Como la sigla aparece al menos una vez en las direcciones válidas, la respuesta es `SI`.

F. Choques circulares

Autor: Vicente Opazo

Tiempo límite: 3 segundos

Memoria límite: 256 megabytes

En un laboratorio se está simulando el crecimiento de n círculos en el plano. Inicialmente, cada círculo está representado solo por un punto en las coordenadas (x_i, y_i) y tiene radio 0. En el tiempo t, todos los círculos que siguen presentes crecieron hasta tener un radio t.

En el instante exacto en que ocurre una colisión — es decir, cuando al menos dos círculos se tocan o se superponen — la simulación se detiene, y se eliminan inmediatamente todos los círculos que participan en alguna colisión en ese momento. Esto incluye cualquier grupo de colisiones simultáneas, ya sea que ocurran en pares separados, en grupos de tres o más, o de cualquier otra forma. Una vez eliminados, esos círculos ya no vuelven a aparecer.

Después de resolver las colisiones, la simulación continúa con los círculos restantes, que siguen creciendo desde el mismo instante como si nada hubiese pasado. Este proceso se repite hasta que no puedan producirse más colisiones. Si al final queda un solo círculo creciendo indefinidamente, no se cuenta como una nueva detención.

Dadas las coordenadas de los n puntos, determina cuántas veces se detendrá la simulación antes de que termine.

Entrada

La primera línea contiene un entero n ($1 \leq n \leq 1000$), el número de círculos.

Cada una de las siguientes n líneas contiene dos enteros x_i y y_i ($1 \leq x_i, y_i \leq 10^9$), que representan las coordenadas del centro del i-ésimo círculo.

No hay dos puntos con coordenadas idénticas.

Salida

Imprime un entero: el número de veces que la simulación se detendrá.

Ejemplos

Entrada 1	Salida 1
4 2 5 3 5 9 10 10 10	1

Entrada 2	Salida 2
3 1 1 7 1 4 1	1

Entrada 3	Salida 3
5 1 1 5 1 11 1 20 1 100 1	2

G. Salta, salta, salta, pequeño Andrés

Autores: Martín Andrighetti, Vicente Opazo, Gabriel Carmona

Tiempo límite: 2 segundos

Memoria límite: 256 megabytes

Andrés siempre fue fanático de trepar cosas: árboles, torres de agua, cualquier estructura que pareciera un desafío. Con el tiempo descubrió su verdadera pasión: el parkour. Saltar entre edificios lo hacía sentir invencible… hasta que un mal salto lo venció, dejándolo con la rodilla destrozada.

El doctor le ofreció un tratamiento experimental: una rodilla ajustable. Con ella, Andrés puede elegir un número entero positivo k, y a partir de ese momento sólo puede realizar saltos con una altura que sea múltiplo de k.

Para probar su rodilla, Andrés viaja a Salamanca, donde hay n edificios en fila, cada uno con altura A_i. La normativa local exige que todos tengan alturas distintas. Andrés planea entrenar durante q días. En el día d, él quiere saber si puede moverse desde un edificio l_d hasta un edificio r_d (avanzando siempre a edificios adyacentes), usando su rodilla ajustada al valor k_d. Como a Andrés le gustan los desafíos extremos, cada día elige un valor de k_d estrictamente mayor que el del día anterior.

Formalmente, Andrés se encuentra en el edificio l_d y quiere llegar al edificio r_d. Puede moverse de un edificio i al edificio adyacente j ($|i-j|=1$) si y sólo si $|A_i - A_j|$ es divisible por k_d.

Tu tarea es responder, para cada consulta, si Andrés puede lograr su objetivo.

Entrada

La primera línea contiene dos enteros n y q ($1 \le n, q \le 10^5$).

La segunda línea contiene n enteros distintos $A_1, A_2, ..., A_n$ ($1 \le A_i \le 10^5$), las alturas de los edificios. Se garantiza que $A_i \neq A_j$ si $i \neq j$.

Cada una de las siguientes q líneas contiene tres enteros l_d, r_d, k_d ($1 \le l_d, r_d \le n$, $1 \le k_d \le 10^5$), representando la consulta del día d consulta. Se garantiza que $k_1 < k_2 < ... < k_q$.

Salida

Para cada consulta, imprimir "SI" si Andrés puede moverse desde l hasta r siguiendo las reglas, o "NO" en caso contrario.

Ejemplos

Entrada 1	Salida 1
5 3 2 7 4 10 3 1 5 1 1 5 2 2 4 3 5 1 5	SI NO SI NO

Nota

Para el caso de ejemplo:

- Día 1 ($d=1$): siempre es posible, porque cualquier diferencia es divisible por 1.
- Día 2 ($d=2$): no es posible, porque $|2-7|=5$ no es divisible por 2.
- Día 3 ($d=3$): sí es posible, porque $|7-4|=3$ y $|4-10|=6$ son divisibles por 3.
- Día 4 ($d=4$): no es posible, porque $|3-10|=7$ no es divisible por 5.

H. La espera infinita

Autores: Javier Oliva, Vicente Opazo

Tiempo límite: 2 segundos

Memoria límite: 256 megabytes

Nacho y Martín están en la fila para almorzar en el CIPC, pero esta fila está tomando una eternidad en avanzar. Tanto así que decidieron sentarse en el piso y jugar un juego.

Hay n fichas y cada ficha tiene dos valores x_i y a_i. Nacho escogerá un subconjunto de estas fichas, denotado S, con puntaje

$$S_p = \sum_{(x_i,a_i)\in S} a_i$$

la suma de los a_i escogidos.

Después, Martín tiene dos opciones, dejar el puntaje final en $f = S_p$ o escoger un subconjunto M de las fichas escogidas por Nacho S y obtener un puntaje final igual a

$$f = S_p \times \oplus_{(x_i,a_i)\in M} x_i$$

donde $\oplus^*$ se entiende como el operador binario XOR.

Nacho quiere maximizar el valor de f y Martín quiere minimizarlo. Si ambos juegan de manera óptima, ¿cuál será el puntaje final?

Nota

El operador binario $\oplus$ trabaja sobre la representación binaria de sus operandos. Calcularemos $10 \oplus 12$ como ejemplo.

La representación binaria de 10 es $(1010)_2$ y la representación binaria de 12 es $(1100)_2$.

$$\begin{array}{r} 1010 \\ \oplus\, 1100 \end{array}$$

Vamos posición por posición (columna por columna) operando los bits de 10 y 12.

- $0 \oplus 0 = 0$.
- $0 \oplus 1 = 1$.
- $1 \oplus 0 = 1$.
- $1 \oplus 1 = 0$.

El último bit (o bit menos signitifativo) de $(1010)_2$ es 0 y el de $(1100)_2$ es 0. Entonces el bit menos significativo del resultado será $0 \oplus 0 = 0$.

El siguiente bit (el penúltimo) de $(1010)_2$ es 1 y el de $(1100)_2$ es 0. Entonces el bit menos significativo del resultado será $1 \oplus 0 = 1$.

Y seguimos así bit por bit y obtenemos

$$\begin{array}{r} 1010 \\ \oplus\, 1100 \\ = 0110 \end{array}$$

Y $(0110)_2$ es la representación binaria de 6. Entonces $10 \oplus 12 = 6$.

Entrada

La primera línea contiene el entero n $(1 \leq n \leq 10^6)$ — el largo de la lista.

La segunda línea contiene n enteros $x_1, x_2, ..., x_n$ $(0 \leq x_i \leq 10^{18})$ — el primer valor de cada ficha.

La tercera línea contiene n enteros $a_1, a_2, ..., a_n$ $(0 \leq a_i \leq 10^{12})$ — el segundo valor de cada ficha.

Nota: La entrada/salida de este problema es muy grande, por lo que se recomienda usar técnicas de entrada/salida rápida.

- *En C++: al inicio del programa main deberán escribir* `std::ios::sync_with_stdio(false);` `std::cin.tie(nullptr);`
- *En Java: usar* `BufferedReader` *con* `StringTokenizer` *para la entrada, y* `BufferedWriter` *o* `PrintWriter` *para la salida.*
- *En Python: usar* `sys.stdin.readline` *en lugar de* `input()` *y* `sys.stdout.write` *para la salida.*

Salida

Imprime el puntaje final.

Ejemplos

Entrada 1	Salida 1
4 10 11 12 13 9 1 3 4	16

Entrada 2	Salida 2
3 8 4 2 10 9 11	30

I. Idea intermedia

Autores: Vicente Opazo, Martín Andrighetti

Tiempo límite: 2 segundos

Memoria límite: 256 megabytes

En el campo de la inteligencia artificial es común representar ideas y conceptos como vectores en un espacio de alta dimensionalidad. Formalmente, a cada concepto a le asignamos un vector E_a, llamado el *embedding* de a. Por ejemplo, si usaramos un espacio de dimensionalidad $n = 4$, el vector asociado al concepto de "rojo" sería un vector de 4 elementos, por ejemplo $E_{\text{rojo}} = (0.5, 1, -3, 0)$. Otros conceptos tendrían sus propios vectores asociados, por ejemplo el concepto de "felicidad" podría representarse por el vector $E_{\text{felicidad}} = (0, -1, -2, 5)$.

Representar conceptos de esta manera nos permite hacer varias cosas interesantes. Por ejemplo, podemos *mezclar* varios vectores. Dado k vectores $E_1, E_2, ..., E_k$, podemos elegir k números $0 \leq x_1, x_2, ..., x_k \leq 1$ tal que $x_1 + x_2 + ... + x_k = 1$, y la *mezcla* resultante será $x_1E_1 + x_2E_2 + ... + x_kE_k$. Intuitivamente, al mezclar vectores el resultado será un vector intermedio, donde cada vector E_i aporta al resultado final en proporción a x_i. Por esta razón, llamamos a los números $x_1, ..., x_k$ como los *pesos* de la mezcla.

Dado $n + 2$ vectores $E_1, E_2, ..., E_{n+2}$, cada uno con n elementos, tu misión es:

1. Separar los $n + 2$ vectores en dos grupos, A y B. Ni A ni B pueden estar vacíos.
2. Mezclar los vectores en A usando pesos a tu elección para obtener un vector z.
3. Mezclar los vectores en B usando pesos a tu elección para obtener el mismo vector z.

Pablo Messina, la eminencia en *machine learning*, ya demostró que esto siempre se puede hacer, pero dejó la implementación como ejercicio para el lector. ¿Puedes implementarlo?

Entrada

La primera línea contiene el entero n ($1 \leq n \leq 500$), representando la cantidad de elementos por vector.

Luego hay $n + 2$ líneas adicionales. La i-ésima de estas líneas contiene n números reales $a_{i,1}, a_{i,2}, ..., a_{i,n}$ ($-1000 \leq a_{i,j} \leq 1000$). Estos números representan el vector de n elementos $E_i = \left(a_{i,1}, a_{i,2}, ..., a_{i,n}\right)$.

Salida

En la primera línea imprime una palabra de $n + 2$ caracteres, donde el i-ésimo caracter es "`A`" si el i-ésimo vector pertenece al grupo A, o "`B`" si el i-ésimo vector pertenece al grupo B. Siempre debe haber al menos un caracter "`A`" y al menos un caracter "`B`", porque los grupos no pueden estar vacíos.

Luego imprime $n + 2$ líneas. En la i-ésima de estas líneas, imprime un único número real: el peso x_i asociado al vector E_i. Este peso x_i corresponde al peso de E_i en la mezcla de A si el vector pertenece a A. En caso contrario, el vector pertenece a B y el peso x_i corresponde al peso de E_i en la mezcla de B.

Debes asegurar lo siguiente:

- $0 \leq x_i \leq 1$ para $i = 1, 2, ..., n + 2$.
- Si $A_1, ..., A_k$ son los índices de los vectores pertenecientes a A, entonces $x_{A_1} + ... + x_{A_k} \approx 1$.
- Si $B_1, ..., B_m$ son los índices de los vectores pertenecientes a B, entonces $x_{B_1} + ... + x_{B_m} \approx 1$.
- Los resultados de ambas mezclas deben ser aproximadamente iguales. Es decir, se debe cumplir que $x_{A_1}E_{A_1} + ... + x_{A_k}E_{A_k} \approx x_{B_1}E_{B_1} + ... + x_{B_m}E_{B_m}$.

Para este problema, dos números son aproximadamente iguales si su diferencia absoluta o relativa es a lo más 10^{-4}. Es decir, $a \approx b$ si y sólo si $|a - b| \leq 10^{-4}$ ó $\frac{|a-b|}{\min(|a|,|b|)} \leq 10^{-4}$. Dos vectores son aproximadamente iguales si cada par de componentes son aproximadamente iguales.

Ejemplos

Entrada 1	Salida 1
1 -1.5 2.5 0.5	AAB 0.500000000000 0.500000000000 1.000000000000

Entrada 2	Salida 2
4 0.50 1.00 -3.00 0.00 0.00 -1.00 -2.00 5.00 -5.00 0.50 4.50 0.00 4.50 2.00 5.00 -3.50 2.00 2.00 2.50 5.00 3.00 1.00 3.00 -1.50	BBBBAA 0.000426075841 0.289305496378 0.029825308905 0.680443118875 0.086919471666 0.913080528334

J. Búsqueda ciega

Autor: Vicente Opazo

Tiempo límite: 2 segundos

Memoria límite: 256 megabytes

Davo ha perdido la vista y para recuperarla necesita encontrar sus k pastillas especiales contra la ceguera. Tiene un pastillero con n compartimentos numerados del 1 al n. En exactamente k de estos compartimentos hay una pastilla contra la ceguera y en los demás hay pastillas comunes que no le hacen efecto.

Las pastillas contra la ceguera tienen un color distinto, por lo que una persona que puede ver sabría diferenciarlas fácilmente. Sin embargo, al tacto todas las pastillas son idénticas, por lo que Davo, que está ciego, no puede saber dónde están. Su amigo Lautaro sí puede verlas, pero como es juguetón, decide ayudarlo de una forma peculiar.

Lautaro le permite hacer consultas del siguiente tipo:

- Davo elige un compartimento x ($1 \leq x \leq n$).
- Lautaro le responde con un número: cuántas pastillas contra la ceguera hay en los compartimentos con índice menor que x (es decir, en los compartimentos $1, 2, ..., x-1$).

Con suficiente información, Davo siempre puede deducir exactamente qué compartimentos contienen las k pastillas especiales. Sin embargo, como no quiere seguirle el juego a Lautaro por mucho tiempo, quiere encontrar una estrategia óptima para hacer las consultas necesarias.

Dado n (el número de compartimentos) y k (el número de pastillas especiales), determina el número mínimo de consultas que Davo necesita para garantizar encontrar las k pastillas en el peor caso, asumiendo que sigue la mejor estrategia posible.

Entrada

La entrada contiene dos enteros n y k ($1 \leq k \leq n \leq 50$).

Salida

Imprime un entero: el número mínimo de consultas necesarias en el peor caso.

Ejemplos

Entrada 1	Salida 1
8 1	3

Entrada 2	Salida 2
5 2	4

Entrada 3	Salida 3
50 50	0

K. El neutrino

Autores: Alejandra Schild, Martín Muñoz

Tiempo límite: 1 segundo

Memoria límite: 256 megabytes

Hace unos años el CERN lanzó el laboratorio de física más grande del universo.

Una tarea muy importante que realizan acá se trata de lanzar neutrinos por un tubo que forma una rueda grande. Normalmente, el neutrino entra al tubo, da una vuelta y llega donde mismo.

Sin embargo, hace poco encontraron un neutrino que es más impredecible y lo bautizaron "Pablo". Por cuestiones cuánticas que aún están estudiando, este neutrino parece que a veces salta entre secciones del tubo que los demás neutrinos no.

El comportamiento del neutrino en el tubo se puede modelar de la siguiente forma: el tubo está formado por las secciones $1, 2, ..., n$, donde la i está conectada bidireccionalmente a la $i + 1$, cuando $1 \leq i < n$; y la n está conectada a la 1.

El neutrino comienza en la sección 1 y luego puede seguir por la 2 o la n. La regla general del movimiento del neutrino es que si ya ha estado en las secciones $a, a + 1, ..., n - 1, n, 1, 2, ..., b - 1, b$, la siguiente sección puede ser $a - 1$ o $b + 1$.

Aparte de estos movimientos, el neutrino, en vez de moverse, puede entrar a un estado muy extraño que los físicos han modelado como "estar en la sección ∞". Básicamente el neutrino desaparece por un momento, y con eso desbloquea'' todas las secciones, y puede comenzar a moverse de sección a sección sin importar si está conectada o no.

Por ejemplo, si $n = 10$, y comienza en las secciones $1, 2, 10, 9, 3$ luego puede entrar a la sección ∞ y seguir con $6, 7, 5, 8$. Otra secuencia posible es $1, 2, 3, \infty, 4, 5, 7, 9, 8, 10, 6$.

El neutrino continúa hasta que haya visto todas las secciones, incluyendo la sección ∞.

Los físicos quieren estudiar todas los movimientos posibles que puede hacer Pablo. El primer paso es contar cuántas secuencias hay. ¿Los ayudas?

Entrada

Un único entero n ($3 \leq n \leq 10^6$).

Salida

Imprime la cantidad de secuencias. El número puede ser muy grande, así que imprímelo módulo $10^9 + 7$.

Ejemplos

Entrada 1	Salida 1
3	6

Entrada 2	Salida 2
4	18

Entrada 3	Salida 3
10	473280

Nota

Para $n = 4$, las secuencias posibles son:

- $1, 2, 3, 4, \infty$
- $1, 2, 3, \infty, 4$

- $1, 2, 4, 3, \infty$
- $1, 2, 4, \infty, 3$
- $1, 2, \infty, 3, 4$
- $1, 2, \infty, 4, 3$
- $1, 4, 2, 3, \infty$
- $1, 4, 2, \infty, 3$
- $1, 4, 3, 2, \infty$
- $1, 4, 3, \infty, 2$
- $1, 4, \infty, 2, 3$
- $1, 4, \infty, 3, 2$
- $1, \infty, 2, 3, 4$
- $1, \infty, 2, 4, 3$
- $1, \infty, 3, 2, 4$
- $1, \infty, 3, 4, 2$
- $1, \infty, 4, 2, 3$
- $1, \infty, 4, 3, 2$

L. A la moda

Autor: Vicente Opazo

Tiempo límite: 3 segundos

Memoria límite: 256 megabytes

A Valentina le encanta estar a la moda. Para armar sus atuendos dispone de N tipos de prendas (por ejemplo: camisas, pantalones, zapatos, etc.). Para cada tipo de prenda i, existen A_i opciones distintas que ella puede elegir.

Un día Valentina ganó el primer premio de una rifa: la posibilidad de obtener K prendas adicionales, pudiendo escoger libremente de qué tipo serán. Con estas prendas extra, Valentina busca maximizar la cantidad total de atuendos distintos que podrá formar.

Un atuendo se compone de exactamente una prenda de cada tipo, y por lo tanto el número de atuendos posibles es el producto de las cantidades disponibles de cada tipo de prenda. Valentina quiere elegir de qué tipos serán sus K prendas adicionales para que este producto sea lo más grande posible.

Como el número puede ser muy grande, se pide calcular el resultado módulo $10^9 + 7$.

Entrada

La primera línea contiene dos enteros $1 \leq N \leq 10^6$ y $0 \leq K \leq 10^9$, el número de tipos de prendas y la cantidad de prendas adicionales que Valentina puede obtener.

La segunda línea contiene N enteros $1 \leq A_1, A_2, ..., A_N \leq 10^9$, donde A_i representa cuántas opciones de la i-ésima prenda existen inicialmente.

Salida

Imprimir un único entero: la cantidad máxima de atuendos distintos que Valentina puede formar, módulo $10^9 + 7$.

Ejemplos

Entrada 1	Salida 1
3 2 2 3 1	18

Entrada 2	Salida 2
4 0 5 2 3 2	60

Nota

- Para el primer caso de prueba el producto inicial es $2 \cdot 3 \cdot 1 = 6$. Valentina podría sumar sus dos prendas extra al segundo tipo (que tenía 3 opciones), de tal forma de conseguir $2 \cdot 5 \cdot 1 = 10$ combinaciones. Sin embargo, una mejor estrategia sería sumar una prenda extra al primer tipo y otra prenda extra al tercer tipo, de tal forma de conseguir $3 \cdot 3 \cdot 2 = 18$ combinaciones. Se puede demostrar que no se puede conseguir un resultado mejor.
- En el segundo caso de prueba no puede sumar ninguna prenda extra, así que el resultado es simplemente $5 \cdot 2 \cdot 3 \cdot 2 = 60$

M. Metamorans

Autores: Javier Oliva, Gabriel Carmona

Tiempo límite: 1 segundo

Memoria límite: 256 megabytes

El tráiler de la nueva película de terror, "Terror en Cerro Punta" (TCP), se volvió viral en todas las redes que te puedas imaginar. En particular, Instagram, TikTok y X muestran miles (cientos de miles) de comentarios que muestran la mejor (y peor) parte de Internet. La sección de comentarios deja en evidencia la verdadera naturaleza humana gracias al anonimato que protege a sus usuarios. Se puede ver de todo, desde comentarios positivos hasta blasfemias e hilos de cientos de comentarios más con peleas entre usuarios donde no se puede ver ninguna relación entre el primer comentario y el último.

Dejando de lado el contenido de los comentarios, y asumiendo que no hay bots valorando el tráiler de ninguna manera, ya sea positiva o negativa, los de marketing quieren averiguar la presencia de los distintos rangos demográficos en estas secciones. Cosas como la edad, el sexo, el género, el estado civil, etc, pueden dar pistas sobre cómo seguir con la estrategia para captar la atención del mayor número de personas posible.

Los encargados de estos estudios, mediante variadas técnicas de minería de datos, encontraron una fuerte correlación entre comentar una publicación del tráiler y tener una edad en el rango $[n_1, n_2]$ (escrita como el timestamp Unix actual menos el de la fecha de nacimiento). De hecho, es tan fuerte como para decir que si una persona comenta entonces tiene una edad en el rango $[n_1, n_2]$.

Esta información tiene implicancias importantes, por ejemplo el rango de edad de las personas que muestran un fuerte interés por la película podría suponer una gran preocupación al equipo de marketing por diversas razones, pero nos saltaremos todas ellas puesto que no van al punto de este problema.

Gabriel y Benjamín son mejores amigos desde que tienen memoria y ambos comentaron en la última publicación del tráiler por TikTok. La edad mínima para ver la película es de k, es decir que una persona con una edad de n puede ver la película si y solo si $n \geq k$. Ambos quieren ver la película pero no están seguros de que podrán. Si alguno de ellos tiene una edad menor a k, pueden realizar la danza de fusión de los Metamorans la cual consiste en una secuencia de m pasos para que dos personas se vuelvan una, combinando varias de sus características individuales.

Podríamos describir cada uno de los m pasos de la fusión pero no es lo suficientemente relevante como para hacerlo. Lo que sí diremos es que el resultado de la fusión tiene una edad igual a la suma de las edades de los fusionados. Es decir que si se fusionan dos personas de edades e_1 y e_2, entonces el resultado tendrá una edad de $e_1 + e_2$.

No conoces las edades de Gabriel y Benjamín, pero si conoces el rango de edad n_1 y n_2. Debes indicar si es posible determinar si podrán o no ver la película.

Entrada

La primera y única línea consiste de tres enteros n_1, n_2 y k ($8 \leq n_1 \leq n_2 \leq 30, 10 \leq k \leq 60$) — el rango de edad de los comentaristas y la edad mínima para ver la película.

Salida

Imprime "`NO`" si no se puede determinar si podrán o no ver la película y "`SI`" en el caso contrario.

Ejemplos

Entrada 1	Salida 1
`20 20 40`	`SI`

Entrada 1	Salida 1
`10 20 31`	`NO`

N. Noentiendo Co.

Autores: Martín Andrighetti, Gabriel Carmona, Benjamín Letelier

Tiempo límite: 2 segundos

Memoria límite: 256 megabytes

Chuntaro Furukagua, presidente de Noentiendo Co., observa con preocupación cómo su compañía atraviesa una crisis. Apenas les quedan 1000 pesos para desarrollar un último juego que podría salvarlos de la quiebra.

En medio de la desesperación, Chijeru Millamoto, creador del famoso fontanero italiano Nachino, propone una idea que puede realizarse exactamente con ese presupuesto.

El juego consiste en un mundo representado por una línea, la cual puede contener obstáculos imposibles de atravesar. El protagonista comienza en la posición a y su objetivo es encontrar el tesoro mágico. Si los obstáculos hacen que el tesoro sea inalcanzable, el jugador pierde. En caso contrario, gana.

Dado que ninguno de sus empleados está dispuesto a trabajar por apenas 1000 pesos, Millamoto pide tu ayuda para programar este juego y así intentar salvar la compañía.

Entrada

La primera línea contiene un entero n ($2 \leq n \leq 1000$), que indica la cantidad de posiciones en el mundo.

La segunda línea contiene un string s de largo n, donde cada carácter representa una posición:

- `.`: posición libre, sin obstáculo
- `#`: posición con obstáculo
- `A`: posición inicial del protagonista
- `T`: posición del tesoro

Se asegura que solo hay una posición donde aparece el carácter `A` y otra donde aparece el carácter `T`.

Salida

Imprime "`ganaste`" si el tesoro es alcanzable por el jugador, en caso contrario imprime "`perdiste`".

Ejemplos

Entrada 1	Salida 1
6 ..A..T	ganaste

Entrada 2	Salida 2
6 ..A#.T	perdiste