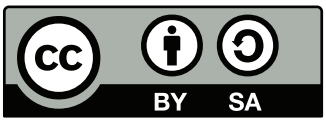
XXIII Olimpiada Murciana de Programación

+

AdaByron Murcia

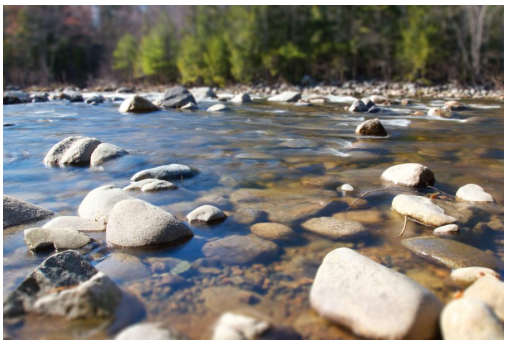
Problemas

A	Cruzando el Río
B	Asignación de Grupos
C	La Final de Boxa-Fighters
D	Calculadora de Beneficios
E	Proyecto Dolce
F	Presupuestos Cuestionables
G	Control de Producción
H	Lineanograma
I	Encriptación... ¿Oblonga?



A. Cruzando el Río

Tiempo de ejecución: 1s



Problema patrocinado por NTT Data.

Contexto

En una actividad de *coworking*, un grupo de empleados ha decidido pasar un día de excursión por el río. Sin embargo, siempre hay rocas u obstáculos que impiden el paso y la única manera de avanzar es saltándolos... pero es posible que algunos no puedan llegar al otro lado. Todo el grupo quiere permanecer unido, por lo que no irán por un camino que no puedan superar todos.

El problema

Cada persona i tiene una capacidad de salto a_i, que determina la distancia a la que puede saltar. Diremos que podrá salvar un obstáculo a distancia d si puede saltar estrictamente más allá del mismo.

Dado un obstáculo, y un grupo de personas con sus capacidades de salto, decidir si todo el grupo puede atravesarlo saltando o no. En caso de no poder todos, decir el número de personas que no pueden.

Entrada

La entrada consiste en múltiples casos de prueba. La primera línea contiene un entero T ($1 \leq T \leq 200$) con el número de casos de prueba. Cada caso de prueba está compuesto por múltiples líneas con el siguiente formato:

- La primera línea contiene dos enteros N ($1 \leq N \leq 10^4$) y D ($1 \leq D \leq 10^9$) con el número de personas en el grupo y la distancia al obstáculo, respectivamente.
- La segunda línea contiene N enteros, donde el i-ésimo entero (a_i) representa la distancia de salto de la persona i ($1 \leq a_i \leq 10^9$).

Salida

Para cada caso de prueba, imprimir una línea con "`TODOS`" en caso de que todas las personas del grupo puedan saltar el obstáculo, o bien "`X NO`", donde X es el número de personas que no pueden saltar el obstáculo.

Ejemplo de Entrada 1	**Ejemplo de Salida 1**
2 2 50 120 150 4 10 25 31 12 10	TODOS 1 NO

B. Asignación de Grupos

Tiempo de ejecución: 1s

Contexto

Como todos los años, hay que elegir grupos de prácticas y esta vez los estudiantes tienen que hacer una lista con sus preferencias. Se utilizará un método un tanto controvertido: asignar los grupos en función de las notas de los alumnos...

El problema

Cada grupo de prácticas tiene un cupo y unos horarios distintos. Cada estudiante tiene una lista ordenada de preferencias de grupos a los que querría ir. Los estudiantes eligen grupo en función de su nota n_i, y en caso de empate, prioriza el orden de aparición de la nota en la entrada del programa.

Tu tarea es calcular una asignación de grupos en la que cada estudiante i no esté asignado a un grupo que preferiría más porque en ese grupo haya plazas sin ocupar o porque otro estudiante con menor nota, o con la misma nota pero que aparezca más tarde según el orden de aparición de la nota de la entrada del programa, haya sido asignado a ese grupo.

Además, las listas de preferencias no son exhaustivas, pero deben indicar al menos un grupo.

Entrada

La entrada consiste en múltiples casos de prueba. La primera línea contiene un entero T ($1 \leq T \leq 200$) con el número de casos de prueba. Cada caso de prueba está compuesto por múltiples líneas con el siguiente formato:

- La primera línea contiene dos enteros n ($1 \leq n \leq 10^4$) y m ($1 \leq m \leq 10^3$), que corresponden al número de estudiantes y número de grupos. Los estudiantes se identifican del 0 al $n-1$, y los grupos del 0 al $m-1$.
- La segunda línea del caso tiene n números reales con 2 decimales ($0.00 \leq n_i \leq 10.00$), que corresponden a las notas de los n alumnos. A mayor nota, mayor preferencia. Si hay 2 notas iguales, tendrá prioridad el orden de aparición de las notas.

- La tercera línea tiene m enteros con los cupos de los grupos de prácticas, $1 \leq q_j \leq n$.
- Siguen n líneas describiendo las listas de preferencias de los estudiantes:
  - Cada línea comienza un entero k, el número de grupos elegidos por el estudiante ($1 \leq k \leq m$), seguido de los k grupos en orden de preferencia, primero los más preferidos.

Se garantiza que en cada caso de prueba, la suma de los elementos de todas las listas de preferencias es menor que 10^6.

Salida

Para cada caso de prueba, imprimir n líneas con una asignación válida que respete las preferencias de los estudiantes y los cupos. Cada línea i de la salida (empezando por 0) debe contener 1 entero j, el grupo al que está asignado el estudiante i.

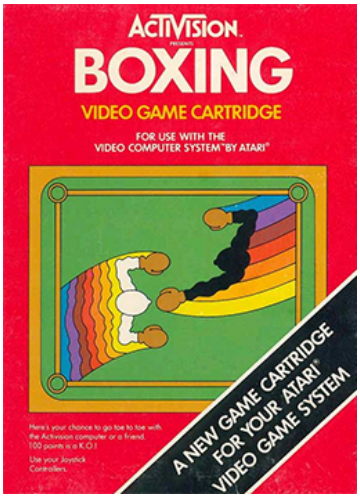
Si un estudiante no está asignado a un grupo, j debe ser -1.

En caso de haber varias asignaciones posibles, imprimir la primera en orden lexicográfico teniendo en cuenta que si hay 2 estudiantes con la misma nota, tiene preferencia el que tenga la primera de las notas según orden de aparición de las mismas.

Ejemplo de Entrada 1	**Ejemplo de Salida 1**
1 3 2 3.30 2.20 1.11 1 2 2 1 0 1 0 2 0 1	1 0 1

C. La Final de Boxa-Fighters

Tiempo de ejecución: 1s



Contexto

Horacio está siguiendo un torneo de su videojuego favorito, Boxa-Fighters. La final, disputada entre los dos mejores jugadores del mundo, Alfredo y Borja, tendrá lugar en EEUU, pero por cuestiones de horario en Murcia se celebrará a altas horas de la madrugada. Dado que no podrá verlo en directo, ha hecho una predicción del equipo que utilizará cada jugador en base a su historial. Sabiendo los equipos que usarán, ¿puedes ayudar a Horacio a predecir el resultado de la final?

El problema

Boxa-Fighters es un juego de estrategia por turnos competitivo en el que cada jugador controla un equipo de luchadores donde no interviene ningún elemento aleatorio. Cada equipo está compuesto por uno o más luchadores. Cada luchador tiene dos estadísticas: puntos de vida (HP) y puntos de daño (DP). Antes de comenzar la partida, cada jugador debe de establecer el orden de los luchadores de su equipo. Un jugador no puede modificar ni reordenar su equipo durante una partida.

Al comienzo de la partida, cada jugador pondrá a su primer luchador en el **puesto activo** y a su segundo luchador (si lo hay) en el **banquillo**, respetando el orden original del equipo.

Durante la partida, los turnos de los jugadores se van alternando. En cada turno, el jugador activo puede ejecutar una de las siguientes acciones:

- **Atacar**: el luchador en su puesto activo inflinge DP puntos de daño al luchador en el puesto activo del rival, reduciendo sus HP en dicha cantidad.
- **Intercambiar y atacar**: el luchador del puesto activo y del banquillo se intercambian de posición, y el luchador que pasa a estar ahora en el puesto activo realiza inmediatamente la acción de **Atacar**.

Cuando los HP de un luchador se reducen a 0 o menos, ese luchador queda fuera de combate y pasa a ser automáticamente sustituido por el luchador en el banquillo de su mismo jugador. Si quedan luchadores en el equipo, automáticamente se preparará el próximo luchador disponible (en orden) y se rellenará el hueco del banquillo. Ganará el primer jugador que consiga vencer a todos los luchadores del equipo rival.

En la final, se sabe que Alfredo empezará jugando el primer turno. Dados los equipos de ambos jugadores y considerando que ambos jugadores jugarán la partida de forma óptima, determina si será Alfredo o Borja el próximo campeón mundial de Boxa-Fighters.

Entrada

La entrada consiste en múltiples casos de prueba. La primera línea contiene un entero T ($1 \leq T \leq 10$) con el número de casos de prueba. Cada caso de prueba está compuesto por múltiples líneas con el siguiente formato:

- La primera línea contiene un entero N ($1 \leq N \leq 10$) con el número de luchadores en el equipo de Alfredo.
- La segunda línea contiene N pares de enteros, a_i y b_i ($1 \leq a_i, b_i \leq 10$) siendo a_i los HP del i-ésimo luchador de Alfredo y b_i los DP del i-ésimo luchador de Alfredo.
- La tercera línea contiene un entero M ($1 \leq M \leq 10$) con el número de luchadores en el equipo de Borja.
- La cuarta línea contiene M pares de enteros, c_i y d_i ($1 \leq c_i, d_i \leq 10$) siendo c_i los HP del i-ésimo luchador de Borja y d_i los DP del i-ésimo luchador de Borja.

Salida

Para cada caso, muestra en una línea el nombre del ganador del combate ("`Alfredo`" o "`Borja`").

Ejemplo de Entrada 1	**Ejemplo de Salida 1**
2 1 1 2 2 1 1 1 1 2 1 1 2 2 2 2 1 1 2	Borja Alfredo

Explicación de los ejemplos

En el primer caso, Alfredo sólo puede atacar con su único luchador (1 HP, 2 DP), derrotando al primer luchador de Borja (1 HP, 1 DP). En el turno de Borja, sólo puede atacar con su

luchador restante (con también 1 HP y 1 DP), derrotando al único luchador de Alfredo y ganando la partida.

En el segundo caso, si Alfredo intercambia su primer luchador (1 HP, 1 DP) con el segundo (2 HP, 2 DP) y ataca, puede derrotar al primer luchador de Borja (2 HP, 1 DP). En el siguiente turno, Borja ataca con su segundo luchador (1 HP, 2 DP) derrotando al segundo luchador de Alfredo (que ahora está primero). En el tercer turno, Alfredo gana la partida atacando con su luchador restante (el que inicialmente era el primero).

Esta página se ha dejado en blanco intencionadamente.

D. Calculadora de Beneficios

Tiempo de ejecución: 1s

Problema patrocinado por ERI Bancaire.

Contexto

Juanito está aprendiendo a invertir en bolsa. Siguiendo las recomendaciones de los expertos, ha invertido todo su dinero en un fondo de inversión a medio año. Al cabo de ese tiempo, el fondo le ha dado una rentabilidad del 80 %, aunque le han quitado un 25 % de los beneficios en impuestos y comisiones. Después ha invertido todo su dinero y las ganancias en otro segundo fondo, que tras otro medio año le ha dado una rentabilidad del 30 %. En este caso los impuestos y comisiones han sido de $\frac{1}{6}$ de los beneficios.

Como Juanito no es muy bueno en matemáticas y, además, su calculadora no tiene suficiente precisión, se ha hecho un lío haciendo las cuentas. Ayuda a Juanito a calcular la cantidad total de dinero que tiene al final del año.

El problema

Dada la cantidad inicial de dinero, debes calcular el dinero que tendrá al final. Lleva mucho cuidado con los cálculos, porque Juanito se ha vuelto inmensamente rico.

Entrada

La entrada consiste en múltiples casos de prueba. La primera línea contiene un entero T ($1 \leq T \leq 1000$) con el número de casos de prueba. Cada caso de prueba está compuesto por una única línea con un número entero X ($0 \leq X \leq 10^{1000}$) que representa la cantidad inicial de dinero que tiene Juanito.

Salida

Para cada caso de prueba debes escribir un único número entero (redondeando al entero más cercano) con la cantidad total de dinero que tiene Juanito al final de sus inversiones.

Ejemplo de Entrada 1	Ejemplo de Salida 1
2 287 5823372036854775803	574 11646744073709551606

Esta página se ha dejado en blanco intencionadamente.

E. Proyecto Dolce

Tiempo de ejecución: 1s

Problema patrocinado por STEL Order.

Contexto

Como experto en Planificación Estratégica, te han pedido ayuda para gestionar el proyecto *Dolce*: la producción a larga escala de pasteles en la mayor fábrica de la región. Para estimar plazos realistas, la empresa modela cada pastel mediante un diagrama con tareas que dependen unas de otras, de forma que una tarea no puede empezar hasta que no terminan todas las tareas de las que depende. Por ejemplo, antes de decorar una tarta de la abuela, hay que preparar las galletas y las natillas.

En particular, a la empresa le conviene estimar la mínima cantidad de tiempo entre que comienza la primera tarea hasta que termina la última en el mejor caso de planificación posible, donde no hay un límite de tareas que se puedan realizar paralelamente.

A esta secuencia de tareas que determina la duración más corta posible entre el principio y el final de la tarta se la denomina ruta crítica.

El problema

Se darán n tareas, cada una con su duración estimada. La tarea inicial es la 0, y la final es la $n-1$. Para cada tarea i, se listan sus prerrequisitos: las tareas que deben terminar antes de que i pueda comenzar. Al ser fabricación masiva de pasteles, se asume que si una tarea está preparada para comenzar, simplemente empieza.

La longitud de la ruta crítica se define como la máxima suma de duraciones de las tareas a lo largo de cualquier sucesión de tareas dependientes cada una de la anterior que empiece en la tarea 0 y termine en la tarea $n-1$ (incluyendo ambas tareas). Tu objetivo es calcular la longitud de la ruta crítica para distintas tartas.

Se garantiza que la tarea 0 es la única tarea que no tiene dependencias, que la tarea $n-1$ tiene de prerrequisitos todas las tareas, y que no hay ciclos ni bucles en el diagrama.

Entrada

La entrada consiste en múltiplos planificaciones de tartas. La primera línea contiene un entero T ($1 \leq T \leq 200$) con el número de tartas. Cada proyecto de tarta está compuesto por múltiples líneas con el siguiente formato:

- La primera línea contiene un entero n ($2 \leq n \leq 10^4$) con el número de tareas.
- La segunda línea contiene n enteros d_i ($1 \leq d_i \leq 10^3$), donde el i-ésimo entero representa la duración de la tarea i.
- A continuación, hay n líneas, cada una describiendo las dependencias de cada tarea i. Cada una de estas líneas comienza con un entero m_i ($0 \leq m_i \leq n-1$), seguido de m_i enteros distintos entre 0 y $n-1$ (excluyendo i), que representan las tareas de las que depende la tarea i.

Se garantiza que la suma de los m_i no supera 10^5.

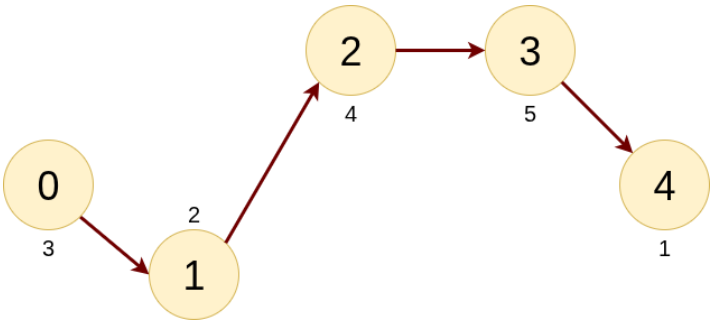
Salida

La salida consta de una línea para cada tarta, indicando la longitud de la ruta crítica desde el paso 0 hasta el $n-1$, incluyendo las duraciones de la primera y última tarea.

Ejemplo de Entrada 1	**Ejemplo de Salida 1**
3 5 3 2 4 5 1 0 1 0 2 0 1 1 2 4 0 1 2 3 5 1 1 1 1 1 0 1 0 1 1 1 2 4 0 1 2 3 6 10 5 1 1 1 3 0 1 0 1 0 1 0 1 0 5 0 1 2 3 4	15 5 18

Explicación de los ejemplos

La entrada consta de 2 casos de prueba. Para el primer caso, se muestra el diagrama con la ruta crítica. Primero se resuelve la tarea 0, después la 1, y secuencialmente 2, 3, y 4, por lo que la duración de la ruta crítica es: $3 + 2 + 4 + 5 + 1 = 15$.

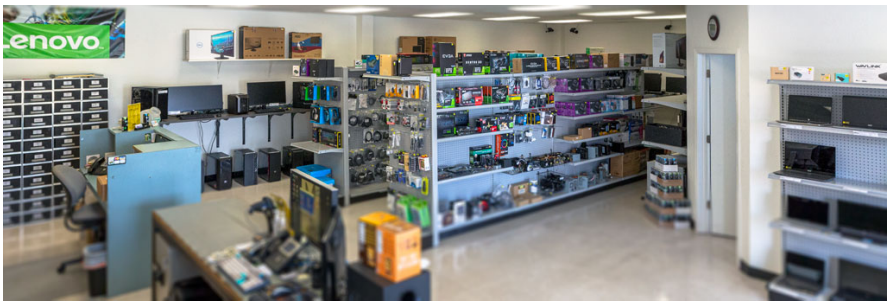


Ruta crítica del primer caso.

Esta página se ha dejado en blanco intencionadamente.

F. Presupuestos Cuestionables

Tiempo de ejecución: 1s



Contexto

Tras una dura selectividad, Carlos ha conseguido entrar en Ingeniería Informática. Sus padres, orgullosos, han decidido comprarle un buen ordenador para hacer la carrera, diciéndole el presupuesto que tiene disponible. Carlos no se ha visto nunca antes en una así y, como es un poco pillo, quiere colar en el presupuesto junto a la CPU también una buena GPU, convenciendo a sus padres de que realmente le hace falta (porque todos sabemos que la última ENBIDIA 89700XTR es imprescindible para programar un "Hola Mundo" en el bloc de notas). Asumiendo que el rendimiento de un equipo completo es equivalente a su precio total de venta, ¿puedes ayudar a Carlos a comprar el mejor PC que le permita su presupuesto?

El problema

Dado el catálogo de precios de todas las CPUs y GPUs de una tienda, determinar cuál es la combinación CPU + GPU más cara que puedes comprar con un presupuesto D.

Entrada

La entrada consiste en múltiplos casos de prueba. La primera línea contiene un entero T ($1 \leq T \leq 100$) con el número de casos de prueba. Cada caso de prueba está compuesto por múltiples líneas con el siguiente formato:

- La primera línea contiene un entero D ($1 \leq D \leq 10^9$) con el presupuesto total disponible.
- La segunda línea contiene dos enteros N y M ($1 \leq N, M \leq 10^5$) con el número de CPUs y GPUs disponibles respectivamente en el catálogo.
- La tercera línea contiene N enteros, donde el i-ésimo entero (n_i) representa el precio de la CPU i ($1 \leq n_i \leq 10^9$).
- La cuarta línea contiene M enteros, donde el j-ésimo entero (m_j) representa el precio de la GPU j ($1 \leq m_j \leq 10^9$).

Se garantiza que la suma de todos los N y la suma de todos los M en todos los casos de prueba no excede 10^6.

Salida

Para cada caso, muestra el precio de la combinación CPU + GPU más cara que se puede comprar con el presupuesto D. Si no es posible comprar ninguna combinación, muestra -1.

Ejemplo de Entrada 1	**Ejemplo de Salida 1**
2 100 5 4 60 20 40 80 50 90 10 70 30 10 2 2 7 9 6 4	90 -1

G. Control de Producción

Tiempo de ejecución: 1s

Problema patrocinado por Neuromobile.

Contexto

En un intento de digitalizar la agricultura, un grupo de inversores ha apostado por el uso de drones para monitorizar los cultivos. Estos van tomando fotografías aéreas del terreno y procesándolas para determinar qué parcelas contienen cultivo y cuáles no. Sin embargo, el presupuesto es limitado y los drones no pueden cubrir todo el terreno. ¿Podrías ayudarles a estimar la cantidad de cultivo que hay en el terreno?

El problema

Tenemos un terreno de N parcelas de alto y M parcelas de ancho. Cada parcela es un cuadrado de 1×1 y la parcela en la posición superior izquierda se indexa como $(1, 1)$. Un dron ha tomado F fotografías aéreas del terreno en posiciones (x, y), siendo cada una de 3×3 parcelas. Algunas de las parcelas pueden aparecer en más de una fotografía. En cada fotografía se ha procesado qué contiene cada parcela: tierra o cultivo.

Dada la descripción del terreno y las F fotografías, determinar cuál es el número mínimo y máximo de parcelas que contienen cultivo en el terreno.

Entrada

La entrada consiste en múltiples casos de prueba. La primera línea contiene un entero T ($1 \leq T \leq 100$) con el número de casos de prueba. Cada caso de prueba está compuesto por múltiples líneas con el siguiente formato:

- La primera línea contiene tres enteros N, M y F ($3 \leq N, M \leq 10^6$, $1 \leq F \leq 10^5$) con las dimensiones del terreno y el número de fotografías respectivamente.
- A continuación, vienen F bloques con la descripción de cada fotografía:

- La primera línea de cada bloque contiene dos enteros x e y ($1 \leq x \leq N-2$, $1 \leq y \leq M-2$) con la posición de la esquina superior izquierda de la fotografía.
- Las siguientes tres líneas contienen cada una 3 caracteres, siendo cada carácter '`#`' si la parcela contiene cultivo y '`.`' si contiene tierra.

Se garantiza que la suma de F en todos los casos de prueba no supera 10^5.

Salida

Para cada caso de prueba, imprimir una línea con dos enteros separados por un espacio: el número mínimo y máximo de parcelas que contienen cultivo en el terreno.

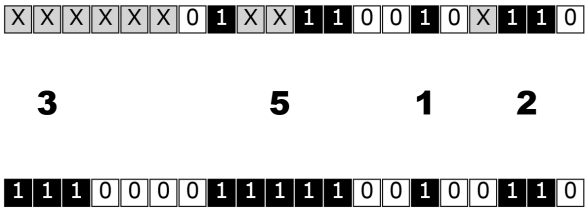
Ejemplo de Entrada 1	**Ejemplo de Salida 1**
1 8 8 3 1 1 ... .#. ... 4 5 .## ..# #.. 6 3 #.# #.. ##.	9 47

Explicación de los ejemplos

La imagen del problema se corresponde con el caso de ejemplo.

H. Lineanograma

Tiempo de ejecución: 1s



Contexto

Un nonograma es un rompecabezas lógico que consiste en una cuadrícula que debe ser rellenada en celdas negras o blancas de acuerdo con ciertas pistas numéricas, de tal manera que se forme un patrón válido. En su versión clásica bidimensional, cada fila y cada columna tiene una secuencia de números que indica el tamaño de los bloques consecutivos de celdas negras que deben aparecer en esa línea, separados siempre por al menos una celda blanca. Resolver un nonograma consiste en deducir, a partir de estas restricciones, qué celdas deben ser negras y cuáles blancas.

En este problema se considera una variante reducida y lineal del nonograma tradicional, llamada *Lineanograma*. Aquí la cuadrícula se restringe a una sola línea horizontal de longitud n. En lugar de pistas bidimensionales, se dan únicamente las pistas de esa línea: una secuencia de enteros positivos que indica los tamaños de cada bloque de celdas negras que deben colocarse, separados por al menos un espacio blanco entre ellos. Además, se proporciona un estado parcial de la línea, en el que algunas posiciones ya están determinadas (como negras o blancas), mientras que otras permanecen inciertas.

El problema

Dada una línea de longitud n, un estado parcial de sus casillas (donde cada casilla puede estar marcada como negra, blanca o indeterminada) y una secuencia de pistas $(a_1, a_2, \ldots, a_M)$ que indican la longitud de los bloques de celdas negras que deben aparecer en el orden dado, se pide determinar si existe al menos una solución válida que complete el lineanograma.

Entrada

La entrada consiste en múltiplos casos de prueba. La primera línea contiene un entero T ($1 \leq T \leq 1000$) con el número de casos de prueba. Cada caso de prueba está compuesto por múltiples líneas con el siguiente formato:

- La primera línea contiene dos enteros N y M, ($1 \leq N \leq 3000$, $1 \leq M \leq \lceil \frac{N}{2} \rceil$) la longitud del lineanograma y el número de pistas:
- La segunda línea contiene M enteros $(a_1, a_2, \ldots, a_M)$, el tamaño de cada bloque de celdas negras.

- La tercera línea contiene una cadena representando el estado parcial, dónde cada carácter indica que la celda en su posición es blanca, negra o indeterminada con un "`0`", "`1`" o "`X`", respectivamente.

Se garantiza que la suma de N en todos los casos de prueba no supera $3 \cdot 10^4$.

Salida

Para cada caso de prueba, escribe en una línea "`SI`" si existe, al menos, una solución o "`NO`" si no existe ninguna.

Ejemplo de Entrada 1	**Ejemplo de Salida 1**
3 20 4 3 5 1 2 XXXXXX01XX110010X110 20 4 3 5 1 2 XX110XXXX11100XXXXXX 20 4 3 5 1 2 1X110XXXX11100XXXXXX	SI SI NO

Explicación de los ejemplos

La imagen del problema se corresponde con el primer caso de prueba.

I. Encriptación... ¿Oblonga?

Tiempo de ejecución: 1s

Problema patrocinado por Qualcomm.

Contexto

Un grupo de investigadores en criptografía está explorando propiedades de ciertas curvas elípticas. Concretamente, las de la forma:

$$y^2 = x^3 - N^2 \cdot x$$

Donde N es un número oblongo. Un número natural es considerado oblongo si se expresa como producto de dos naturales consecutivos.

Los investigadores están interesados en identificar rápidamente si un número dado es oblongo o no. ¿Puedes ayudarles a desarrollar un programa que determine si un número es oblongo?

El problema

Dado un número natural N, determinar si es un número oblongo o no.

Entrada

La entrada consiste en varios casos de prueba. Cada caso de prueba está compuesto por un único número natural N ($1 \leq N \leq 10^{18}$). La entrada termina con un caso de prueba con $N = 0$, que no debe ser procesado.

Se garantiza que el total de casos de prueba no excede 10^6.

***OJO**: N puede desbordar un entero, si es necesario, utiliza un tipo de datos que permita almacenarlos. En C++, `long long`. En Java, `Long`. En Python, no es necesario indicar el tipo.*

Salida

Para cada caso de prueba, se debe imprimir una línea con "`SI`" si N es un número oblongo, o "`NO`" en caso contrario.

Ejemplo de Entrada 1	Ejemplo de Salida 1
12 100 20880535878006 0	SI NO SI