

## Maratón de programación de la SBC 2020

El set de problemas de este cuaderno es utilizado simultáneamente en las siguientes competencias:

Maratona de Programação da SBC 2020  
Segunda Fecha Gran Premio de México 2020  
Primera Fecha Gran Premio de Centroamérica 2020  
Torneo Argentino de Programación 2020

*14 de Noviembre de 2020*

### Cuaderno de Problemas

#### Información General

Este cuaderno contiene 15 problemas; Las páginas están numeradas de 1 a 25, sin contar esta página. Verifique que su cuaderno está completo.

#### A) Sobre los nombres de los programas

- 1) Para soluciones en C/C++ y Python, el nombre del archivo de código fuente no es significativo, puede ser cualquier nombre.
- 2) Si su solución es en Java, el archivo debe ser llamado: `codigo_de_problema.java` donde `codigo_de_problema` es la letra mayúscula que identifica al problema. Recuerde que en Java el nombre de la clase principal debe ser igual que el nombre del archivo.
- 3) Si su solución es en Kotlin, el archivo debe ser llamado: `codigo_de_problema.kt` donde `codigo_de_problema` es la letra mayúscula que identifica al problema. Recuerde que en Kotlin el nombre de la clase principal debe ser llamado igual que el nombre del archivo

#### B) Sobre la entrada

- 1) La entrada de su programa debe ser leída de *entrada standard*.
- 2) La entrada está compuesta de un único caso de prueba, descrito en un número de línea que depende del problema.
- 3) Cuando una línea de entrada contiene varios valores, estos están separados por un único espacio en blanco; la entrada no contiene ningún otro espacio en blanco.
- 4) Cada línea, incluyendo la última, contiene exactamente un caracter de final-de-línea.
- 5) El final de la entrada coincide con el final del archivo.

#### C) Sobre la salida

- 1) La salida de su programa debe ser escrita en *salida standard*.
- 2) Cuando una línea de salida contiene varios valores, estos deben ser separados por un único espacio en blanco; la salida no debe contener ningún otro espacio en blanco.
- 3) Cada línea, incluyendo la última, debe contener exactamente un caracter de final-de-línea.

Promocional:



Sociedade Brasileira de Computação

## Problema A

# Álbum de Cartas

¡El álbum de cartas del subregional del ICPC 2020 de Nlogonia ya está disponible! Para celebrar la competencia, los aficionados de la programación competitiva de todo el país están comprando los álbumes y coleccionando las cartas.

Este álbum de cartas es especial porque todas las cartas son iguales: una imagen del trofeo de este año. Para completar el álbum solo debes juntar suficientes cartas como para llenar todos los espacios en el.

Te debes estar preguntando: ¿Qué tiene de divertido coleccionar esas cartas? Para hacer las cosas interesantes, las cartas se venden en paquetes, cada uno de ellos con un número aleatorio de cartas. Los fans celebran cuando encuentran un paquete con muchas cartas, se burlan de los que tuvieron la mala suerte de encontrar un paquete con pocas cartas, y alardean cuando llenan su álbum usando pocos paquetes.

Acabas de adquirir tu álbum, y ¡quieres comenzar a llenarlo! Pero, antes de comprar tu primer paquete de cartas te has preguntando: ¿En promedio cuántos paquetes se deben comprar para llenar el álbum?

### Entrada

La única línea de entrada contiene tres enteros  $N$ ,  $A$ , y  $B$ , separados por un espacio, satisfaciendo  $1 \leq N \leq 10^6$ ,  $0 \leq A \leq B \leq 10^6$ , y  $B > 0$ , donde:

- $N$  es el número de cartas necesarios para llenar el álbum.
- $A$  es el mínimo número de cartas que puede haber en un paquete.
- $B$  es el máximo número de cartas que puede haber en un paquete.

El número de cartas en cada paquete está distribuido uniformemente en el rango cerrado  $[A, B]$ .

### Salida



















La salida consiste de una única línea, que debe contener el número esperado de paquetes que se requieren para completar el álbum. El número se considerará correcto si está dentro de un error absoluto o relativo de  $10^{-5}$  de la respuesta correcta.

|   |  |
|---|--|
| <b>Ejemplo de entrada 1</b><br>40 0 2   | <b>Ejemplo de salida 1</b><br>40.33333 |
| <b>Ejemplo de entrada 2</b><br>100 1 10 | <b>Ejemplo de salida 2</b><br>18.72727 |
| <b>Ejemplo de entrada 3</b><br>30 3 3   | <b>Ejemplo de salida 3</b><br>10.00000 |
| <b>Ejemplo de entrada 4</b><br>314 5 8  | <b>Ejemplo de salida 4</b><br>48.74556 |

## Problema B

# Batalla Naval

La Batalla Naval es un juego de estrategia clásico para dos jugadores. Cada jugador coloca su conjunto de barcos en una grilla de  $10 \times 10$  y luego el juego consiste en adivinar las posiciones de los barcos. Las reglas del juego no son importantes para este problema, y existen muchas variantes, pero aquí estamos interesados en un problema mucho más sencillo. Dada la lista de barcos y sus posiciones, su tarea es comprobar si el posicionamiento inicial es válido.

|    | 1   | 2   | 3   | 4  | 5  | 6  | 7  | 8 | 9   | 10 |
|----|---|---|---|--|--|--|--|---|---|----|
| 1  |   |   |   |  |  |  |  |   |   |    |
| 2  |   |    |  |   |  |  |  |   |  |    |
| 3  |   |   |   |  |  |   |  |   |  |    |
| 4  |   |    |   |  |  |   |  |   |  |    |
| 5  |   |   |   |  |  |  |  |   |  |    |
| 6  |   |   |   |  |  |   |  |   |  |    |
| 7  |   |   |   |  |  |  |  |   |   |    |
| 8  |   |   |   |  |  |  |  |   |   |    |
| 9  |   |   |   |  |  |  |  |   |   |    |
| 10 |  |  |   |  |  |  |  |   |   |    |

Las filas y columnas de la grilla están numeradas de 1 a 10, y los barcos se colocan horizontal o verticalmente, ocupando una secuencia contigua de cuadrados del tablero. Para este problema, un posicionamiento es válido si

- Ninguna posición está ocupada por más de un barco;
- Todos los barcos están totalmente contenidos dentro del tablero.

### Entrada

La primera línea de la entrada contiene un entero  $N$ ,  $1 \leq N \leq 100$ , el número de barcos. Cada una de las siguientes  $N$  líneas contiene cuatro enteros  $D$ ,  $L$ ,  $R$  y  $C$  con  $D \in \{0, 1\}$ ,  $1 \leq L \leq 5$  y  $1 \leq R, C \leq 10$  describiendo un barco. Si  $D = 0$  entonces el barco está alineado horizontalmente y ocupa las posiciones  $(R, C) \dots (R, C + L - 1)$ . De lo contrario, la nave está alineada verticalmente, ocupando las posiciones  $(R, C) \dots (R + L - 1, C)$ .

### Salida

Imprima una línea en la salida que contenga un solo carácter. Si el posicionamiento inicial de los barcos es válido, escriba la letra mayúscula ‘Y’; de lo contrario escriba la letra mayúscula ‘N’.

|  |                                 |
|--|---------------------------------|
| <b>Ejemplo de entrada 1</b><br>3<br>0 5 1 1<br>1 5 2 2<br>0 1 3 3  | <b>Ejemplo de salida 1</b><br>Y |
| <b>Ejemplo de entrada 2</b><br>2<br>0 2 1 1<br>1 1 1 2   | <b>Ejemplo de salida 2</b><br>N |
| <b>Ejemplo de entrada 3</b><br>1<br>0 2 10 10  | <b>Ejemplo de salida 3</b><br>N |
| <b>Ejemplo de entrada 4</b><br>7<br>0 3 2 2<br>1 5 2 9<br>1 2 3 6<br>1 1 4 2<br>0 1 6 6<br>0 4 8 4<br>0 2 10 1 | <b>Ejemplo de salida 4</b><br>Y |

## Problema C

# Concatenando Equipos

Pepito is an ICPC coach who very often gets to “concat” the names of two existing teams, such as “AJI” and “Oxidados”, in order to produce names for new teams, such as “AJIOxidados”.

Since Pepito coaches teams from two different universities where he teaches, he had an idea: he will consider all possible such concatenations of a team name from university  $A$ , with a team name from university  $B$  (always in that order: first the one from university  $A$ , then the one from university  $B$ ).

So, if team names from university  $A$  are “Buen” and “Kilo”, and team names from university  $B$  are “Pan” and “Flauta”, all the possible concatenations that he considers are the strings “BuenPan”, “BuenFlauta”, “KiloPan” and “KiloFlauta”.

Furthermore, he calls a certain team *peculiar* if removal of that team makes the set of concatenations lose all the concatenations that used the name of that team.

In effect, in the previous example all the teams are peculiar. If however we consider names from  $A$  to be “xx” and “xxy” and names from  $B$  to be “z”, “yz” and “xx”, then “xx” from university  $A$  is not peculiar, because the name “xx” + “yz” = “xxyz” = “xxy” + “z” and can thus be formed without the need to use “xx” from  $A$ . For the same reason, “yz”, “xxy” and “z” are not peculiar names. The only peculiar name in this case is “xx” from university  $B$ , because it is involved in creating the names “xxxx” and “xxyxx”, and it is completely impossible to create any of these without using “xx” from university  $B$ .

Given the team names from both universities, your task is to calculate how many peculiar teams are there in each university.

### Entrada

The first line contains two integers,  $M$  and  $N$ , separated by a space. The number of teams from university  $A$  is  $M$  and the number of teams from university  $B$  is  $N$ .

The second line contains the names of the teams from university  $A$ , separated by a space. The third line contains the names of the teams from university  $B$ , separated by a space.

All team names consist solely of lower case letters of the English alphabet.

No two teams from the same university have the same name.

$1 \leq M, N \leq 10^5$  and the total length of all team names is at most  $10^6$ .

### Salida

Output a line containing two integers: the number of peculiar teams from university  $A$ , and the number of peculiar teams from university  $B$ , separated by one space.

|   |                                   |
|---|-----------------------------------|
| <b>Ejemplo de entrada 1</b><br>2 2<br>buen kilo<br>pan flauta | <b>Ejemplo de salida 1</b><br>2 2 |
| <b>Ejemplo de entrada 2</b><br>2 3<br>xx xxy<br>z yz xx       | <b>Ejemplo de salida 2</b><br>0 1 |

## Problema D

# Danza de Divisibilidad

Los habitantes del país de Nlogonia realizan una danza especial para rendir homenaje al dios de la divisibilidad. La danza es realizada por  $N$  hombres y  $N$  mujeres dispuestos en dos círculos. Los hombres están en el círculo interno y las mujeres en el círculo externo. Cada mujer inicia de frente a un hombre.

La danza se compone de  $K$  movimientos; hombres y mujeres alternan movimientos, comenzando con los hombres. En el  $i$ -ésimo movimiento, las personas del círculo que corresponde rotarán  $P_i$  pasos en sentido horario mientras que las personas del otro círculo permanecerán quietas. Así, cada persona cambiará de pareja a una que está a  $P_i$  posiciones de distancia. Un movimiento es válido si las parejas de cada persona son diferentes al inicio y al fin del movimiento y, además de eso, ninguna pareja de personas se encuentre frente a frente en dos momentos de tiempo distintos.

Como forma de rendir homenaje, las danzas siempre deben terminar con parejas cuyas sumas de edades tienen el mismo residuo cuando se dividen por el número sagrado  $M$ . Es decir, si la suma de las edades de una pareja deja un residuo  $R$  cuando se divide entre  $M$ , entonces la suma de las edades de cualquier pareja debe dejar el mismo residuo  $R$ .

Dados  $N$ ,  $M$ ,  $K$ , y las edades de todos los danzantes, determine de cuántas formas se puede realizar la danza. Como la edad de los danzantes se mide en segundos, las edades pueden ser muy grandes.

### Entrada

La primera línea de entrada contiene tres enteros  $N$  ( $3 \leq N \leq 10^6$ ),  $M$  ( $1 \leq M \leq 10^9$ ), y  $K$  ( $1 \leq K \leq 10^9$ ), representando, respectivamente, la cantidad de personas en cada círculo, el número sagrado, y la cantidad de movimientos de la danza.

La segunda línea de entrada contiene  $N$  enteros  $A_i$  ( $1 \leq A_i \leq 10^9$ ) separados por un espacio en blanco, representando las edades de las mujeres.

La tercera línea de entrada contiene  $N$  enteros  $B_i$  ( $1 \leq B_i \leq 10^9$ ) separados por un espacio en blanco, representando las edades de los hombres.

Al inicio el  $i$ -ésimo hombre está alineado con la  $i$ -ésima mujer, y el primer elemento de cada vector se considera a la derecha de su último elemento.

### Salida

La salida consiste en una línea con un único número entero, representando el residuo del número de danzas distintas posibles al dividirse por  $10^9 + 7$ .

|   |                                 |
|---|---------------------------------|
| <b>Ejemplo de entrada 1</b><br>4 10 1<br>3 4 1 7<br>13 27 36 9  | <b>Ejemplo de salida 1</b><br>1 |
| <b>Ejemplo de entrada 2</b><br>5 10 2<br>3 4 1 7 6<br>4 7 1 2 5 | <b>Ejemplo de salida 2</b><br>3 |

|   |                                  |
|---|----------------------------------|
| <b>Ejemplo de entrada 3</b><br>5 10 2<br>3 4 1 7 6<br>5 4 7 1 2               | <b>Ejemplo de salida 3</b><br>4  |
| <b>Ejemplo de entrada 4</b><br>6 21 3<br>10 58 23 31 37 2<br>45 17 9 24 38 30 | <b>Ejemplo de salida 4</b><br>42 |

## Problema E

# Empresa de Fiestas

Yankovich trabaja como Ingeniero de Software en una empresa llamada POI (Party Online Infinitely), como el nombre sugiere, esta empresa promueve las fiestas en línea. Para probar los sistemas, algunos empleados realizaron fiestas iniciales e invitaron solo a empleados de la empresa siguiendo algunas restricciones.

Los empleados de la empresa forman una estructura jerárquica, sin relaciones cíclicas, donde cada empleado tiene un gerente directo (excepto el dueño de la empresa). Debido a los procesos de promoción de la empresa, la edad de un empleado no puede ser mayor a la edad de su gerente directo.

Las fiestas iniciales son de la siguiente manera:

- Cada fiesta  $j$ , tiene un organizador  $O_j$ , y un rango de edad que va de  $L_j$  a  $R_j$ , inclusivos.
- Para ser invitado a una fiesta, un empleado debe tener edad en el rango de edad  $[L_j, R_j]$  de la fiesta (se garantiza que la edad del organizador de cada fiesta está en el rango de edad de su fiesta).
- Además de la restricción de edad, para ser invitado a una fiesta, un empleado debe trabajar directamente con otro empleado que también será invitado (como gerente directo, o subordinado directo), excepto para el organizador de la fiesta.

Yankovich es responsable de la aplicación que entrega información acerca de a qué fiestas un usuario ha participado, como tarea inicial, el debe de calcular a cuántas fiestas un empleado fue invitado. Como está algo retrasado en la entrega de su primer tarea, ayúdale a calcular esta información para las fiestas iniciales.

### Entrada

La primer línea de entrada contiene dos enteros  $N$ , y  $M$  ( $1 \leq N, M \leq 10^5$ ), que representan, respectivamente, el número de empleados, y el número de fiestas iniciales.

Las siguientes  $N$  líneas contienen la información acerca de los empleados, donde la  $i$ -ésima línea tendrá dos enteros  $A_i$ ,  $B_i$  ( $1 \leq A_i \leq 10^5$ ,  $1 \leq B_i \leq N$ ), representando, respectivamente, la edad del  $i$ -ésimo empleado, y su gerente directo (los empleados están enumerados con los números del 1 al  $N$ , el dueño de la empresa se identifica con el número 1, y es el único empleado donde  $B_i = i$ ). Se garantiza que  $A_i \leq A_{B_i}$ .

Las siguientes  $M$  líneas contienen la información sobre las fiestas iniciales, en la  $j$ -ésima de estas líneas habrá tres números enteros  $O_j$ ,  $L_j$ , y  $R_j$  ( $1 \leq L_j \leq A_{O_j} \leq R_j \leq 10^5$ ), representando el dueño de la  $j$ -ésima fiesta, y su rango de edad.

### Salida

Imprima una línea conteniendo  $N$  enteros separados por un espacio, donde el  $i$ -ésimo de estos números representa el número de fiestas a las que el empleado identificado con el número  $i$  fue invitado.



| Ejemplo de entrada 1  | Ejemplo de salida 1 |
|---|---------------------|
| 10 3<br>8 1<br>3 5<br>5 1<br>2 3<br>4 1<br>3 3<br>1 2<br>7 1<br>2 2<br>3 2<br>3 5 9<br>5 3 8<br>3 2 6 | 2 1 3 1 1 2 0 2 0 1 |

## Problema F

# Fastminton

La *Comisión para el desarrollo Regional del Fastminton* (CRLF) organiza torneos anuales del novedoso e inusual juego Fastminton, un derivado del Bádminton. CRLF se enorgullece de organizar el tercer gran torneo, que se celebrará este año.

El ex programador de la comisión ha desarrollado un sistema computarizado para capturar y almacenar los resultados de los partidos, punto por punto, en archivos individuales. Se fue antes de poder completar un analizador sintáctico para los archivos, por esto la CRLF requiere su ayuda para garantizar que todos los registros se puedan leer de los archivos escritos, para evitar perder años de emocionantes resultados de partidos.

Un resumen de las reglas de Fastminton se les ha entregado para ayudarlos en su tarea. Es, en esencia, una versión más corta (o más rápida) del Bádminton:

- Los partidos de Fastminton siempre se juegan con dos jugadores adversarios en una cancha que está dividida por una red;
- Los jugadores se identifican por sus posiciones relativas en el marcador (jugador de la izquierda y de la derecha);
- Un partido se compone de tres juegos. El jugador que llega a 2 juegos es el ganador;
- Se gana un juego cuando un jugador alcanza al menos 5 puntos con una diferencia de al menos 2 puntos con respecto a su oponente, o al llegar a 10 puntos (lo que ocurra primero);
- El jugador de la izquierda comienza a sacar en el primer juego. Para todos los demás juegos, el jugador que comienza sacando es el que ganó el último juego;
- Cada ronda da como resultado un punto para el jugador que saca o que recibe. El jugador que anota saca en la siguiente ronda.

### Entrada

La entrada consiste en una sola línea que contiene una secuencia de caracteres. Esta línea representa la secuencia completa de los eventos de un partido y puede contener los caracteres S (punto para el saque), R (punto para el receptor) o Q (anuncio de puntuación). La entrada no contiene anuncios de puntuación consecutivos.

### Salida

El programa debe imprimir una línea que contenga la puntuación actual para cada anuncio de puntuación (Q) que se encuentre en la entrada.

Si el juego está en marcha, el anuncio estará en el formato “GL (PL) – GR (PR)”, donde GL y GR son el número de *juegos* ganados por los jugadores de la izquierda y derecha, y PL y PR son los puntos actuales de los jugadores de la izquierda y derecha. Un asterisco (\*) se debe añadir al puntaje del jugador que sacará en la siguiente ronda.

Si el juego ha terminado, el anuncio estará en el formato “GL – GR” con la cadena “(winner)” agregada al final del puntaje de juego del jugador ganador.

| Ejemplo de entrada 1 | Ejemplo de salida 1              |
|----------------------|----------------------------------|
| SRSSQSSSSQRRSS       | 0 (1) – 0 (3*)<br>0 (0) – 1 (2*) |

|  |  |
|--|--|
| <b>Ejemplo de entrada 2</b><br>SRSSQSSSSQRRSSQ       | <b>Ejemplo de salida 2</b><br>0 (1) - 0 (3*)<br>0 (0) - 1 (2*)<br>0 - 2 (winner) |
| <b>Ejemplo de entrada 3</b><br>RSRSSRRRRRRRRSSSSRRSQ | <b>Ejemplo de salida 3</b><br>2 (winner) - 0                                     |

## Problema G

# Game Show!

La Sociedad de Brillantes Competidores (SBC) organiza programas televisivos para sus miembros (¡y actualmente también transmite en línea!). SBC usa un sistema de créditos llamado *sbecs*, que pueden ser usados por los jugadores para participar en concursos o canjearse por premios al final de cada temporada. SBC inició un nuevo tipo de juego, ¡y necesita hacer algunas simulaciones para evitar grandes pérdidas en el pozo de premios!

Ricardo va a probar el nuevo juego. Debe apostar 100 sbecs, que se transfieren a su saldo en el juego. Luego, una secuencia de cajas es colocada. El juego se desarrolla en rondas y el número máximo de rondas es igual al número de cajas. En cada ronda, Ricardo decide si abre la siguiente caja o abandona el juego. Si Ricardo abandona, recupera el saldo actual de sbecs. Si Ricardo abre la siguiente caja, su contenido, que es un número secreto, se suma a su saldo y el juego continúa. Como el número secreto en la caja puede ser negativo, ¡Ricardo puede terminar con pérdidas! El juego termina cuando Ricardo decide abandonar o cuando se abre la última caja.

SBC los contrató para probar el juego. A partir del contenido de las cajas, deben decidir cuál es el mayor saldo posible que Ricardo puede obtener.

### Entrada

La primera línea de la entrada contiene un entero  $C$ ,  $1 \leq C \leq 100$ , que es el número de cajas en el juego. Después de la primera línea de entrada, hay  $C$  líneas más. Cada una de estas  $C$  líneas contiene el número secreto de una caja. Las líneas están en el mismo orden que las cajas. Los números secretos son enteros,  $V$ ,  $-1000 \leq V \leq 1000$ .

### Salida

Imprima en la salida una línea que contenga un número entero que sea el mayor saldo posible que Ricardo puede obtener, dada esa secuencia de cajas.

|   |                                   |
|---|-----------------------------------|
| <b>Ejemplo de entrada 1</b><br>4<br>-1<br>-2<br>-3<br>-4          | <b>Ejemplo de salida 1</b><br>100 |
| <b>Ejemplo de entrada 2</b><br>5<br>-10<br>20<br>-30<br>40<br>-50 | <b>Ejemplo de salida 2</b><br>120 |

## Problema H

# Hangar del SBC

Un pequeño avión de carga del Sistema Binario de Cargas (SBC) fue protegido para transportar productos especiales y secretos. Esos productos son agrupados en cajas con diferentes pesos.

El avión tiene un rango de peso de seguridad, dentro del cual el avión es estable. Más específicamente, hay dos valores  $A$  y  $B$  tales que si el peso total de las cajas transportadas es menor que  $A$  o mayor que  $B$ , no será posible garantizar la seguridad del vuelo.

Se sabe que todas las cajas tienen pesos diferentes, y que, al comparar los pesos de cualquier par de cajas, la caja más pesada pesa al menos el doble que la caja de menor peso.

Su tarea es determinar de cuántas maneras puede elegir exactamente  $K$  cajas para ser transportados en el avión sin que sea desestabilizado.

### Entrada

La primera línea de entrada contiene dos enteros  $N$  ( $1 \leq N \leq 50$ ) y  $K$  ( $1 \leq K \leq 50$ ), que representan respectivamente el número total de cajas y la cantidad que se debe cargar en el avión.

La segunda línea de entrada contiene  $N$  enteros  $P_i$  ( $1 \leq P_i \leq 10^{18}$ ) separados por un espacio en blanco, y representando los pesos de cada caja.

La tercera línea de entrada contiene dos enteros,  $A$  y  $B$  ( $1 \leq A \leq B \leq 2 \times 10^{18}$ ), que representan, respectivamente, el límite mínimo y el límite máximo de peso para que el avión se mantenga estable.

Considere que todos los pesos están reportados en la misma unidad.

### Salida

La salida debe contener una única línea con un entero representando la cantidad de formas de elegir la cantidad especificada de cajas sin poner en riesgo el vuelo.

|   |                                  |
|---|----------------------------------|
| <b>Ejemplo de entrada 1</b><br>3 2<br>10 1 3<br>4 13          | <b>Ejemplo de salida 1</b><br>3  |
| <b>Ejemplo de entrada 2</b><br>4 3<br>20 10 50 1<br>21 81     | <b>Ejemplo de salida 2</b><br>4  |
| <b>Ejemplo de entrada 3</b><br>6 3<br>14 70 3 1 6 31<br>10 74 | <b>Ejemplo de salida 3</b><br>11 |

## Problema I

# Interactividad

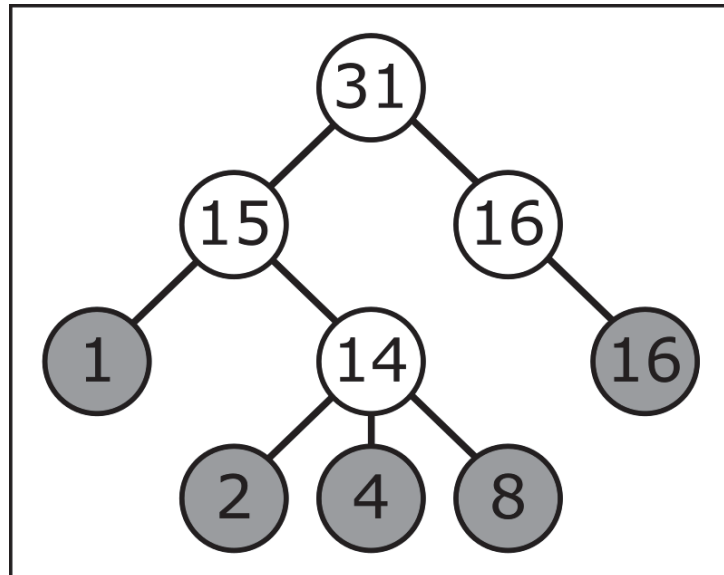
Un día, Alice retó a Bob con el problema interactivo de programación descrito a continuación:

—————}}—————

Tienes un árbol (un grafo conectado acíclico). Cada nodo de este árbol tiene exactamente un *padre*, excepto por el nodo *raíz*, el cual no tiene padre. Los nodos que no son padres de ningún otro nodo se llaman *hojas*. Tu conoces la estructura del árbol, porque conoces el padre de cada nodo que no es la raíz.

Cada nodo contiene un valor entero. Un nodo que no es una hoja contiene la suma de los valores de todos sus hijos directos. Entonces, todos los valores del árbol son determinados por los valores contenidos en sus hojas.

La siguiente imagen muestra un ejemplo. Las hojas están marcadas con color gris, mientras que todos los demás nodos están en blanco. Cada nodo muestra el valor que contiene.



Inicialmente, tu no conoces el valor de ninguno de los nodos, pero, puedes consultarlos de uno por uno. Tu tarea es encontrar cuál es el valor de cada nodo en el árbol, haciendo la menor cantidad de consultas posibles.

—————}}—————

Bob resolvió este problema de una manera muy simple. Entonces, para dificultar las cosas, Alice le preguntó: “Dada la estructura del árbol, ¿cuántas soluciones distintas a este problema existen?” Esto es, ¿cuántos conjuntos mínimos de consultas existen que permitan determinar los valores contenidos en cada nodo del árbol? ( Dos conjuntos de consultas se consideran diferentes si y solo si existe un nodo que es consultado en un conjunto pero no en el otro).

### Entrada

El árbol tiene  $N$  nodos en total. Cada nodo está identificado por un entero entre 1 y  $N$ , donde el nodo 1 es la raíz.

La entrada consiste de dos líneas. La primer línea contiene un entero  $N$ .

La segunda línea de entrada contiene  $N - 1$  enteros  $P_1, P_2, \dots, P_{N-1}$ , separados por un espacio, donde  $P_i$  es el padre del nodo  $i + 1$ , para  $i = 1, 2, \dots, N - 1$ .

$2 \leq N \leq 10^5$ .

$1 \leq P_i \leq N$ , para  $i = 1, 2, \dots, N - 1$ .

**Salida**

La salida consiste de una sola línea, la cual debe contener el número de soluciones mínimas diferentes para el problema al que Bob fue retado. Como este número puede ser grande, su respuesta debe ser calculada módulo 1000000007 ( $10^9 + 7$ ).

|   |                                 |
|---|---------------------------------|
| <b>Ejemplo de entrada 1</b><br>3<br>1 1     | <b>Ejemplo de salida 1</b><br>3 |
| <b>Ejemplo de entrada 2</b><br>4<br>1 2 3   | <b>Ejemplo de salida 2</b><br>4 |
| <b>Ejemplo de entrada 3</b><br>5<br>1 2 2 2 | <b>Ejemplo de salida 3</b><br>7 |

## Problema J

# Juntando Datos

Acre y Amanda son muy curiosos en la manera que siempre buscan patrones. Ellos rutinariamente recolectan y analizan información de varias fuentes (tráfico de la ciudad, lluvias, número de hojas que caen de los árboles), esperando encontrar algunos patrones interesantes.

Su última expedición produjo un conjunto de datos bastante prometedor: ¡seguía una línea recta perfecta! Formalmente, era una lista de  $N/2$  pares de enteros, posiblemente repetidos. Cuando estos pares se graficaron como puntos en un plano cartesiano, ¡todos ellos eran perfectamente colineales! Satisfechos, Acre y Amanda guardaron estos datos como una tabla conteniendo los pares de enteros.

Desafortunadamente, mientras Acre y Amanda estaban recolectando más información, su hijo menor entró a su oficina y modificó los datos de la tabla, revolviendo todos los valores que ésta tenía. Ahora todo lo que les quedó a Acre y Amanda son los  $N$  enteros revueltos. Ellos desean reconstruir la tabla con los datos que les quedaron.

Formalmente, Acre y Amanda, quieren reorganizar estos enteros en  $N/2$  pares, donde cada par representa un punto en el plano cartesiano, de tal modo que todos estos puntos sean colineales. La lista de enteros puede tener valores repetidos, y, cada valor debe ser usado tantas veces como aparece en la lista. El conjunto de datos resultante puede tener también puntos repetidos.

Como puede haber varios diferentes conjuntos de datos aceptables que se pueden formar con los enteros dados, Acre y Amanda quieren conocer la cantidad de dichos conjuntos de datos. Dos conjuntos de datos son considerados diferentes si y solo si hay un punto que aparece más veces en un conjunto que en el otro.

### Entrada

La primer línea de entrada contiene un entero  $N$ , la longitud de la lista de enteros.  $N$  siempre es un número par, porque es el doble de puntos que en el conjunto de datos original. La segunda línea contiene  $N$  enteros, representando la lista de valores revueltos de la tabla de datos.

Los enteros están separados por un espacio.

$$4 \leq N \leq 200.$$

Cada valor  $I$  de la lista está en el rango:  $-10000 \leq I \leq 10000$ .

### Salida

La salida debe contener una única línea con un número entero, la cantidad de formas diferentes en que la lista de enteros puede ser acomodada en puntos colineales. Como este número puede ser muy grande, imprima la respuesta módulo  $1000000007$  ( $10^9 + 7$ ).

La respuesta podría ser zero en algunos casos.

|   |                                  |
|---|----------------------------------|
| <b>Ejemplo de entrada 1</b><br>8<br>1 2 3 5 20 18 16 12 | <b>Ejemplo de salida 1</b><br>2  |
| <b>Ejemplo de entrada 2</b><br>6<br>1 2 3 4 5 20        | <b>Ejemplo de salida 2</b><br>0  |
| <b>Ejemplo de entrada 3</b><br>8<br>1 2 5 5 5 5 8 9     | <b>Ejemplo de salida 3</b><br>10 |



## Problema K

### K personas entre nosotros

Los empates suelen ser un problema en las elecciones y los juegos. Recientemente, un nuevo juego, llamado *Between Us*, fue creado. En el juego participan varios jugadores a través de una red social. En cada turno, hay múltiples votaciones, donde un jugador puede recibir solo votos de amigos. El jugador que obtiene la mayor cantidad de votos gana el juego.

El juego aún se encuentra en su fase de diseño, pero los desarrolladores del juego se enfrentaron a un problema muy común: dado que el número de amigos de un jugador suele ser pequeño, los empates son muy comunes, lo que hace que el juego sea menos divertido. Para evitar esto, decidieron agregar un nuevo módulo al sistema de emparejamiento, donde se intentará formar grupos de jugadores de manera que cada jugador tenga un número impar de amigos en el mismo grupo.

Estos problemas resultaron ser más difíciles de lo que esperaban y ahora se están enfocando en una versión más simple: dado un conjunto de jugadores, el módulo debe encontrar una *partición* de los jugadores en como máximo dos grupos, satisfaciendo la restricción de que cada jugador debe tener un número impar de amigos en su grupo. El problema es que esta partición no siempre es posible. Su tarea es determinar si la partición es posible.

#### Entrada

La primera línea de entrada contiene dos enteros,  $P$  y  $F$ , el número de jugadores y el número de relaciones de amistad, respectivamente, que satisfacen  $2 \leq P \leq 100$  y  $1 \leq F \leq P \times (P - 1)/2$ . Cada una de las siguientes  $F$  líneas contiene dos enteros  $A$  y  $B$ , con  $1 \leq A, B \leq P$  y  $A \neq B$ , indicando que los jugadores  $A$  y  $B$  son amigos. Cada relación de amistad aparece a lo sumo una vez, es decir, si una línea contiene  $A$  y  $B$ , no hay otra línea con ambos números.

#### Salida

Imprima en la salida una sola línea que contenga un carácter. Si la partición es posible, escriba la letra mayúscula 'Y'; de lo contrario, escriba la letra mayúscula 'N'.

|  |                                 |
|--|---------------------------------|
| <b>Ejemplo de entrada 1</b><br>4 4<br>4 2<br>1 3<br>2 3<br>1 4 | <b>Ejemplo de salida 1</b><br>Y |
| <b>Ejemplo de entrada 2</b><br>4 3<br>4 2<br>2 3<br>1 2        | <b>Ejemplo de salida 2</b><br>Y |

| Ejemplo de entrada 3                   | Ejemplo de salida 3 |
|--|---------------------|
| 5 5<br>3 5<br>3 1<br>1 4<br>2 5<br>2 4 | N                   |

## Problema L

### Labaspar

Sopa de Letras es un pasatiempo bien conocido, sin embargo, este ha perdido su prestigio en los últimos años. El objetivo de este juego es el de encontrar palabras de una lista, en una matriz, donde cada celda de la matriz contiene una letra.

Bibika y su hermano estaban jugando Sopa de Letras, pero perdieron el interés en el juego muy pronto debido a que encontrar todas las palabras se estaba volviendo algo relativamente fácil. Como Bibika quiere alejar a su hermano de la computadora, se puso a buscar juegos similares en internet y encontró el *Cacería de Labaspar*.

Cacería de Labaspar es un juego que sigue la misma idea que la famosa Sopa de Letras. Sin embargo, en lugar de tener que encontrar simplemente una palabra en una matriz, el objetivo es encontrar cualquier anagrama de la palabra, haciendo que el juego sea más difícil e interesante. El anagrama se puede encontrar en cualquier fila, columna, o diagonal de la matriz.

Un anagrama es una palabra formada por el reacomodo de las letras de otra. Algunas veces, un anagrama no existe como una palabra en el lenguaje, pero esto no importa. BALO, LOBA, y AOLB, son ejemplos de anagramas de la palabra BOLA.

Bibika se dio cuenta que era posible que la misma celda de la matriz forme parte de anagramas de distintas palabras, a las cuales llamó *celdas especiales*.

A ella le gustaría saber, dada una matriz de configuración y una lista de palabras, ¿Cuántas celdas especiales tiene la matriz?

|   |   |   |   |   |
|---|---|---|---|---|
| X | B | O | I | C |
| D | K | I | R | A |
| A | L | B | O | A |
| B | H | G | E | S |

La imagen de arriba ilustra el primer ejemplo, donde la lista de palabras consiste de tres palabras: BOLA, CASA, y BOI. Los rectángulos de cada color representan anagramas de diferentes palabras de la lista. Las 3 celdas especiales están coloreadas de amarillo.

#### Entrada

La primer línea de entrada contiene dos enteros,  $L$  y  $C$  ( $2 \leq L, C \leq 40$ ), que representan, respectivamente, el número de líneas y columnas en la matriz.

Cada una de las siguientes  $L$  líneas contienen  $C$  letras mayúsculas.

La siguiente línea de entrada contiene un entero,  $N$ , el cual representa la cantidad de palabras en la lista de palabras.

Finalmente, hay  $N$  líneas, cada una de ellas contiene una palabra  $P_i$  ( $2 \leq P_i \leq \min(15, \max(L, C))$ ) de la lista.

Todas las letras en la matriz y en cada palabra de la lista de palabras serán letras mayúsculas del alfabeto inglés.

Se garantiza que las palabras de la lista no son anagramas entre si.

### Salida

La salida consiste de una línea con un número entero, representando el numero de celdas especiales en la matriz.

|  |                                 |
|--|---------------------------------|
| <b>Ejemplo de entrada 1</b><br>4 5<br>XBOIC<br>DKIRA<br>ALBOA<br>BHGES<br>3<br>BOLA<br>CASA<br>BOI | <b>Ejemplo de salida 1</b><br>3 |
| <b>Ejemplo de entrada 2</b><br>3 3<br>AAB<br>ABA<br>BAA<br>2<br>ABA<br>BBB                         | <b>Ejemplo de salida 2</b><br>3 |
| <b>Ejemplo de entrada 3</b><br>2 4<br>AAAA<br>AAAA<br>2<br>AAA<br>BBB                              | <b>Ejemplo de salida 3</b><br>0 |

## Problema M

### Metralleta

Fulanito juega un juego de arcade de la vieja escuela. En el juego, él puede poner una metralleta en cualquier lugar dentro de su base, la cual contiene todos los puntos  $(x, y)$ , donde  $x < 0$ . Hay  $N$  enemigos en el campo de batalla. El  $i$ -ésimo enemigo ( $1 \leq i \leq N$ ) está en la posición  $(x_i, y_i)$  con  $x_i > 0$ . Todas estas posiciones se entregan por adelantado.

Una metralleta en el punto  $(x_m, y_m)$  cubre un ángulo de visión hacia la derecha, centrado en la línea  $y = y_m$ , y sus bordes están dados por las líneas  $y = y_m \pm \frac{x - x_m}{2}$ . Al ser posicionada, elimina a todos los enemigos en ese ángulo, incluyendo sus bordes.

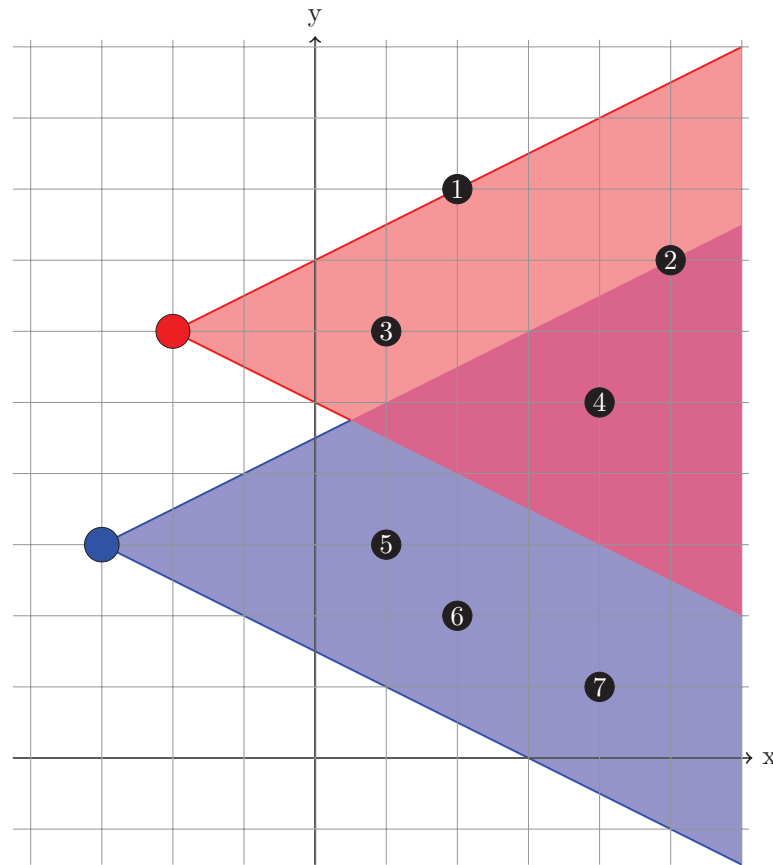


Figura 1: Pictorial representation of the sample input

El sistema de puntuación usado en este video juego es extremadamente extraño: muchos creen que de hecho era un horrible bug de los desarrolladores del juego, quienes responden gritando “¡no es un bug, es un feature!” a cualquiera que les pregunte sobre el tema. Específicamente, el puntaje que se obtiene por cierto posicionamiento de la metralleta se calcula de la siguiente manera::

- Se toman los ids ( $i$  entre 1 y  $N$ ) de **todos** los enemigos que elimina.
- Ordenar esos ids en orden ascendente.
- Sean los valores ordenados  $i_0 < i_1 < \dots < i_{k-1}$
- Calcular el puntaje como  $\left( \sum_{j=0}^{k-1} i_j \cdot 5782344^j \right) \bmod (10^9 + 7)$

- Nota: Una metralleta que no elimina a ningún enemigo obtiene un puntaje de exáctamente 0 puntos.

Para mejorar en este juego, Fulanito te hará  $q$  consultas: cada consulta pregunta por el puntaje que obtendría al posicionar una metralleta en algún  $(x_m, y_m)$ .

Para hacer el problema más retador, cada  $(x_m, y_m)$  no es entregado por adelantado. En su lugar, se entregan los valores  $a$ , y  $b$ , de modo que  $x_m = -1 - ((p + a) \bmod (10^9 + 7))$  y  $y_m = (p + b) \bmod (10^9 + 7)$ , donde  $p$  es la respuesta a la consulta anterior ( $p = 0$  cuando se procesa la primer consulta).

**NOTA:** Se garantiza que en total, tomando todas las consultas juntas, el total de enemigos eliminados es a lo más  $10^6$ .

$$1 \leq x_i, y_i \leq 10^9$$

$$1 \leq N, q \leq 10^5$$

$$0 \leq a, b < 10^9 + 7$$

## Entrada

La primer línea de entrada contiene dos enteros  $N$ , y  $q$ .

Cada una de las siguientes  $N$  líneas contiene dos enteros:  $x_i$ , y  $y_i$ .

Cada una de las siguientes  $q$  líneas contienen dos enteros:  $a$ , y  $b$  que especifican cada una de las consultas como se explica en el enunciado.

## Salida

Para cada consulta, imprima una línea con un número entero, representando la respuesta a esa consulta.

| Ejemplo de entrada 1 | Ejemplo de salida 1 |
|----------------------|---------------------|
| 7 2                  | 626214369           |
| 2 8                  | 981053491           |
| 5 7                  |                     |
| 1 6                  |                     |
| 4 5                  |                     |
| 1 3                  |                     |
| 2 2                  |                     |
| 4 1                  |                     |
| 2 3                  |                     |
| 373785639 373785644  |                     |

## Problema N

# Números Multiplicados

Eugenius es un matemático brillante que disfruta de multiplicar números.

Una vez, encontró  $M$  nodos escritos en un pedazo de papel, enumerados del 1 a  $M$ , llamamos a estos nodos *nodos- $M$* . Cada nodo  $i$  estaba etiquetado con un número primo único  $p_i$ , tal que los primos se encontraban ordenados (i.e. si  $i < j$ , entonces  $p_i < p_j$ ).

Más tarde, decidió dibujar  $N$  nodos nuevos, llamados *nodos- $N$* , y agregó aristas entre *nodos- $M$*  y *nodos- $N$* . Al hacerlo, fue muy cuidadoso en no conectar un *nodo- $M$*  con otro *nodo- $M$* , ni un *nodo- $N$*  con otro *nodo- $N$* . No fue tan cuidadoso con la cantidad de aristas que dibujó entre cada par de nodos.

De esta forma, Eugenius consiguió un multigrafo bipartito.

Cómo su pasión siempre fue multiplicar números, decidió etiquetar cada *nodo- $N$*  con el producto de de todos los *nodos- $M$*  directamente conectados a él, multiplicando cada *nodo- $M$*  tantas veces como aristas los conectasen.

Cada *nodo- $N$*  terminó, entonces, siendo etiquetada por un número  $c_i$ . Estos números están caracterizados por la siguiente fórmula:

$$c_i = \prod_{(j,i) \in E} p_j,$$

Donde  $E$  es el multiconjunto de aristas, y cada arista  $e$  satisface que  $e \in M \times N$ . Agotado, Eugenius decidió comer algo. Mientras disfrutaba su torus, accidentalmente tiro café sobre las etiquetas  $p_i$ , que se borraron instantaneamente.

¿Lo podrías ayudar a recuperar los números primos originales?

### Entrada

La primera línea contiene tres enteros  $M$ ,  $N$  y  $K$ , la cantidad de *nodos- $M$* , la cantidad de *nodos- $N$*  y la cantidad de aristas *distintas*. Estos valores satisfacen que  $1 \leq M, N < 10^3$  and  $1 \leq K < 10^4$ .

La siguiente línea tiene  $N$  números  $c_i$ , las etiquetas de los *nodos- $N$* , tal que  $1 < c_i < 10^{15}$ .

Finalmente, hay  $K$  líneas, cada una con tres números  $m$   $n$   $d$ , que representan que hay  $d$  aristas entre el *nodo- $M$*   $m$  y el *nodo- $N$*   $n$ , satisfaciendo que  $1 \leq m \leq M$ ,  $1 \leq n \leq N$  and  $1 \leq d \leq 50$ .

Se garantiza que todos los nodos (*nodos- $M$* , y *nodos- $N$* ) tienen grado por lo menos 1.

### Salida

Imprima una sola línea con  $M$  números ordenados, representando los números primos que etiquetan a los *nodos- $M$*  con índices  $1, \dots, M$  que le quitan el sueño a Eugenius.

| Ejemplo de entrada 1 | Ejemplo de salida 1 |
|----------------------|---------------------|
| 4 3 4                | 2 3 5 13            |
| 15 16 13             |                     |
| 2 1 1                |                     |
| 3 1 1                |                     |
| 1 2 4                |                     |
| 4 3 1                |                     |

| Ejemplo de entrada 2  | Ejemplo de salida 2 |
|---|---------------------|
| 4 5 7<br>3 9 7 143 143<br>1 1 1<br>1 2 2<br>2 3 1<br>3 4 1<br>3 5 1<br>4 5 1<br>4 4 1 | 3 7 11 13           |



## Problema O

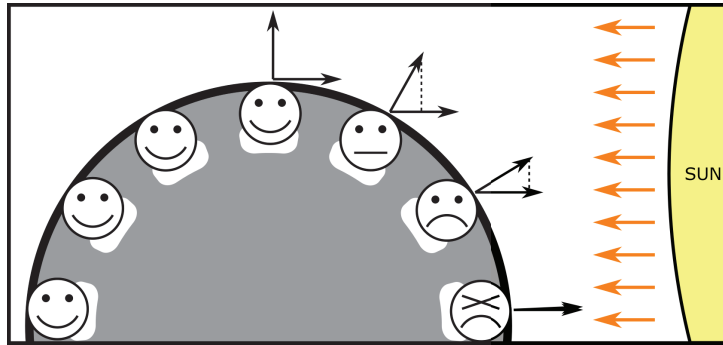
### Ómnibus Venusiano

La Colonia Humana en Venus está prosperando. Ahí, el transporte público principal es el Ómnibus Venusiano: este es un disco volador redondo, con ventanas y asientos a lo largo de su borde. En este ómnibus todos los asientos tienen ventana. Y las personas no tienen permitido cambiar de asiento, así que cuando alguien elige un asiento, permanecerá en este hasta bajar del ómnibus.

A pesar de ser un vehículo autónomo, cada ómnibus opera con un ingeniero a bordo que se encarga de manejar cualquier problema inesperado. Tu eres el ingeniero del ómnibus 1C9C, y te pasas la mayor parte de tu turno de trabajo leyendo libros. El problema, es que detestas estar expuesto a la luz solar. Entonces, quieres elegir el asiento que minimice la cantidad total de luz solar que obtienes durante todo tu turno de trabajo.

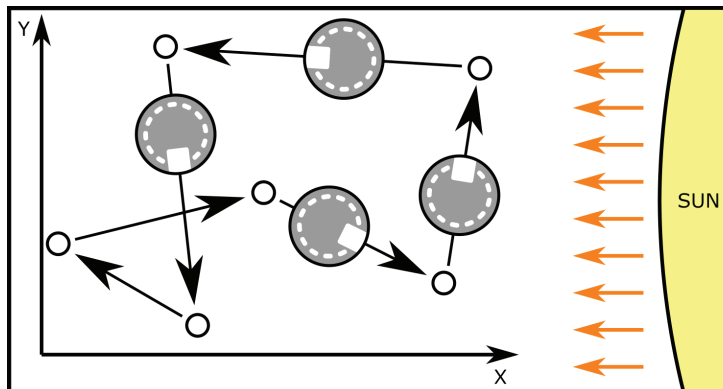
La colonia está representada por un plano cartesiano, donde el eje  $X$  apunta hacia el Este, y el eje  $Y$  apunta hacia el Norte. Los días en Venus son muy largos (de hecho, ¡más largos que un año!), así que puedes suponer que el sol siempre brilla en la misma dirección desde el Este. Esto es, la luz solar siempre viaja hacia el Oeste, en el sentido negativo de  $X$ .

Vea la figura de abajo, mientras tu ventana apunte más al Este, estarás más expuesto a la luz solar. Pero si la ventana apunta al oeste no te expondrás a la luz solar en absoluto.



Formalmente, suponga que el vector  $(D_x, D_y)$  representa la dirección a la que apunta su ventana. Note que solo se expondrá a la luz solar si  $D_x > 0$ . Sea  $\theta$  el ángulo entre los vectores  $(D_x, D_y)$  y  $(1, 0)$  (un vector que apunta directamente hacia el sol). Si  $\cos(\theta) \leq 0$  entonces no te expones a la luz solar. De otro modo, te expondrás a  $\cos(\theta)$  unidades de luz solar por segundo.

La ruta del ómnibus consiste de una secuencia de estaciones en la colonia. El ómnibus inicia el turno de trabajo en la primera estación, visita todas las estaciones en orden, y después regresa a la primera.



El ómnibus siempre sigue una línea recta cuando va de una estación a la siguiente. A pesar de que el ómnibus es redondo, tiene un “lado de frente”: este lado siempre está mirando hacia la dirección en la que se está moviendo, y el ómnibus rota de acuerdo a esto cada que cambia de dirección en las estaciones.

Puedes ignorar el tiempo que se toma el ómnibus en girar, y en subir o bajar pasajeros.

### Entrada

La primer línea de entrada contiene un único entero  $N$ , el número de estaciones visitadas durante la ruta del ómnibus.

Después siguen  $N$  líneas, cada una contiene las coordenadas  $X$  y  $Y$  de una estación, separadas por un espacio.

Las estaciones son dadas en el orden que son visitadas.

Cada estación puede ser visitada más de una vez en la ruta.

Estaciones consecutivas siempre son distintas, así como la última y la primer estación.

Todas las coordenadas son dadas en metros.

El ómnibus viaja a una velocidad constante de un metro por segundo.

$2 \leq N \leq 100000$ .

Las coordenadas de cada estación son enteros en el rango  $-10000 \leq X, Y \leq 10000$ .

### Salida

La salida consiste de una única línea que contiene un único número real, la cantidad mínima total de luz solar que puedes recibir en una vuelta de la ruta del ómnibus.

Su respuesta debe tener exactamente dos dígitos después del punto decimal.

|   |                                    |
|---|------------------------------------|
| <b>Ejemplo de entrada 1</b><br>3<br>2 5<br>17 5<br>11 11        | <b>Ejemplo de salida 1</b><br>6.00 |
| <b>Ejemplo de entrada 2</b><br>4<br>3 0<br>3 6<br>6 3<br>0 3    | <b>Ejemplo de salida 2</b><br>4.24 |
| <b>Ejemplo de entrada 3</b><br>4<br>3 2<br>1 1<br>-3 -1<br>-1 0 | <b>Ejemplo de salida 3</b><br>0.00 |