Olimpiada Chilena de Informática 2024

Final Nacional

14 de diciembre, 2024

Información General

Esta página muestra información general que se aplica a todos los problemas.

Envío de una solución

1. Los participantes deben enviar **un solo archivo** con el código fuente de su solución.
2. El nombre del archivo debe tener la extensión `.cpp` o `.java` dependiendo de si la solución está escrita en `C++` o `Java` respectivamente. Para enviar una solución en Java hay que seguir algunos pasos adicionales. Ver detalles más abajo.

Casos de prueba, subtareas y puntaje

1. La solución enviada por los participantes será ejecutada varias veces con distintos casos de prueba.
2. A menos que se indique lo contrario, cada problema define diferentes subtareas que lo restringen. Se asignará puntaje de acuerdo a la cantidad de subtareas que se logre solucionar de manera correcta.
3. A menos que se indique lo contrario, para obtener el puntaje en una subtarea se debe tener correctos todos los casos de prueba incluídos en ella.
4. Una solución puede resolver al mismo tiempo más de una subtarea.
5. La solución es ejecutada con cada caso de prueba de manera independiente y por tanto puede fallar en algunas subtareas sin influir en la ejecución de otras.

Entrada

1. Toda lectura debe ser hecha desde la **entrada estándar** usando, por ejemplo, las funciones `scanf` o `std::cin` en C++ o la clase `BufferedReader` en Java.
2. La entrada corresponde a un solo caso de prueba, el cual está descrito en varias líneas dependiendo del problema.
3. **Se garantiza que la entrada sigue el formato descrito** en el enunciado de cada problema.

Salida

1. Toda escritura debe ser hecha hacia la **salida estándar** usando, por ejemplo, las funciones `printf`, `std::cout` en C++ o `System.out.println` en Java.
2. El formato de salida es explicado en el enunciado de cada problema.
3. **La salida del programa debe cumplir estrictamente con el formato indicado**, considerando los espacios, las mayúsculas y minúsculas.
4. Toda línea, incluyendo la última, debe terminar con un salto de línea.

Envío de una solución en Java

1. Cada problema tiene un *nombre clave* que será especificado en el enunciado. Este nombre clave será también utilizado en el sistema de evaluación para identificar al problema.
2. Para enviar correctamente una solución en Java, el archivo debe contener una clase llamada igual que el nombre clave del problema. Esta clase debe contener también el método `main`. Por ejemplo, si el nombre clave es `marraqueta`, el archivo con la solución debe llamarse `marraqueta.java` y tener la siguiente estructura:

```
public class marraqueta {
  public static void main (String[] args) {
    // tu solución va aquí
  }
}
```

3. Si el archivo no contiene la clase con el nombre correcto, el sistema de evaluación reportará un error de compilación.
4. La clase no debe estar contenida dentro de un *package*. Hay que tener cuidado pues algunos entornos de desarrollo como Eclipse incluyen las clases en un *package* por defecto.
5. Si la clase está contenida dentro de un package, el sistema reportará un error de compilación.

Problema A

Aplausómetro

nombre clave: `aplausometro`

El equipo de producción del programa de televisión Domingo Colosal quiere mejorar el sistema de selección popular usado para determinar el ganador de sus concursos. Su sistema actual consiste en que los participantes son dispuestos en línea, y el animador los presenta de izquierda a derecha, pidiendo al público que aplauda por ellos. Después de presentar a todos los participantes, quién obtiene la mayor cantidad de aplausos es considerado el ganador.

El problema con este sistema es que un participante está en desventaja con respecto al siguiente, pues luego de que el público aplaude por alguien, la parte del público que quiere que gane el siguiente sabe cuanto tiene que aplaudir para que gane. Por lo tanto, si un participante recibe más aplausos que el anterior, eso no quiere decir que sea más preferido. Sin embargo, si un participante recibe menos aplausos que el anterior, entonces si sabemos que este claramente no debería ganar.

La producción cuenta con un nuevo aplausómetro digital que mide la intensidad exacta de los aplausos en aplos (la unidad internacional de intensidad de aplausos). Aprovechando esta tecnología, la producción ha diseñado un nuevo proceso de selección que se ejecuta en varias rondas. En este nuevo proceso, el animador también presenta a cada uno de los participantes de izquierda a derecha y pide al público que aplaudan por ellos. Si el número de aplos para un participante i es estrictamente menor que los aplos para el participante directamente a su izquierda el participante i queda eliminado. Al final de la ronda, los participantes eliminados son sacados de la fila y comienza una nueva ronda. Notar que los participantes son sacados de la fila al finalizar la ronda. Notar además que el participante 1 nunca puede ser eliminado de esta forma pues no tiene nadie a su izquierda.

El proceso termina cuando al finalizar una ronda no hay ningún participante eliminado. Al finalizar el proceso, se declara como ganador a quién tenga más aplausos de los participantes restantes o empate en caso de haber más de uno con la misma cantidad.

Queda poco tiempo para el inicio de la nueva temporada y la producción está preocupada de que ejecutar el nuevo proceso tardará demasiadas rondas. Dada la intensidad con que el público aplaude por cada participante, y asumiendo que la intensidad del aplauso es siempre la misma por cada participante, tu tarea es determinar la cantidad de rondas necesarias para que el proceso termine.

Entrada

La primera línea de la entrada contiene un entero N ($1 \leq N \leq 10^6$) correspondiente a la cantidad inicial de participantes. Cada participante es numerado con un entero entre 1 y N. La segunda línea contiene N enteros $a_1 a_2 \cdots a_N$ ($0 \leq a_i \leq 10^9$). El entero a_i corresponde a la intensidad, medida en aplos, con que el público aplaude para el participante i. Sin importar la ronda, el público siempre aplaude a_i aplos por el participante i.

Salida

La salida debe contender un entero correspondiente a la cantidad de rondas necesarias para terminar el proceso.

Subtareas y puntaje

Subtarea 1 (40 puntos)

Se probarán varios casos de prueba donde $N \leq 10^3$.

Subtarea 2 (60 puntos)

Se probarán varios casos de prueba sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
6 4 5 2 1 3 4	4

Explicación ejemplo: Como se muestra a continuación, en este ejemplo se requieren cuatro rondas para terminar el proceso. En la primera ronda se eliminan los participantes 3 y 4. En la segunda se elimina al participante 5. En la tercera se elimina al participante 6. Finalmente, en la cuarta ronda no hay ningún eliminado y por lo tanto el proceso termina al finalizar esta ronda.

Primera ronda	Participante	1	2	~~3~~	~~4~~	5	6
	Aplos	4	5	~~2~~	~~1~~	3	4
Segunda ronda	Participante	1	2	~~5~~	6		
	Aplos	4	5	~~3~~	4		
Tercera ronda	Participante	1	2	~~6~~			
	Aplos	4	5	~~4~~			
Cuarta ronda	Participante	1	2				
	Aplos	4	5				

Entrada de ejemplo	Salida de ejemplo
5 5 2 1 3 2	3

Problema B

Burrito de letras

nombre clave: `burrito`

En el Instituto de Colección de Palabras en Cuadrículas (ICPC) son fanáticos de los juegos que involucran palabras tales como la famosa sopa de letras en donde un jugador debe buscar palabras ocultas en una cuadrícula de $n \times n$ casillas. Cada año, el ICPC organiza una convención donde se presentan innovadoras variantes de juegos de palabras. Este año, Norma Gibat, la famosa inventora de juegos de palabra, presentó una nueva versión de la sopa de letras basándose en su comida favorita: los burritos.

Esta nueva versión, llamada "burrito de letras", funciona igual que una sopa de letras normal, solo que además la cuadrícula es colocada alrededor de un cilindro como si estuviese envolviendo un burrito. Específicamente, las reglas del burrito de letras son las siguientes:

- El juego consiste en una cuadrícula de $n \times n$ donde cada casilla tiene una letra escrita.
- Al jugador se le presenta además una lista de m palabras, las cuales pueden estar o no escritas en la cuadrícula. Para cada una de ellas, el jugador debe responder si está o no en la cuadrícula.
- Una palabra puede estar escrita en cualquier lugar de forma vertical de arriba hacia abajo, o de forma horizontal de izquierda a derecha. Las palabras no pueden aparecer de forma diagonal.
- La cuadrícula está colocada alrededor de un cilindro y una palabra puede "dar la vuelta" en sentido horizontal. Es decir, la primera y última letra de cada fila se consideran adyacentes.

Para poder explicar mejor el juego, Norma necesita mostrar la solución a varios burritos de letras de ejemplo, pero está muy ocupada interactuando con todos sus fans para hacerlo ella misma. ¿Podrías ayudarla?

Figure 1: Imagen del prototipo presentado durante la convención.

Entrada

La primera línea de la entrada contiene un entero n ($2 \leq n \leq 50$) correspondiente al tamaño de la cuadrícula.

Cada una de las siguientes n líneas contiene n caracteres, describiendo las filas de la cuadrícula de arriba hacia abajo.

Luego sigue una línea conteniendo un entero m ($1 \leq m \leq 50$), correspondiente a la cantidad de palabras a buscar.

Cada una de las siguientes m líneas contienen una palabra a buscar. El largo de cada una de las palabras a buscar será mayor o igual a 2 y menor o igual que n.

Todos los caracteres de la cuadrícula y de las m palabras corresponden a letras mayúsculas del alfabeto inglés.

Salida

Para cada una de las m palabras, imprime una línea que diga `PRESENTE` si la palabra está en el burrito de letras o `AUSENTE` de lo contrario.

Subtareas y puntaje

Subtarea 1 (50 puntos)

Se probarán varios casos en que se garantiza que no aparecen palabras en sentido horizontal, es decir, si una palabra está presente, esta aparecerá de forma vertical.

Subtarea 2 (50 puntos)

Se probarán varios casos de prueba sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
5 ACCOZ TOBGA HIELF RCAKM UPOCI 3 OCI GATO ICPC	PRESENTE PRESENTE AUSENTE

Entrada de ejemplo	Salida de ejemplo
4 AABB BBCC CCDD DDEE 4 ABCD BA BBAA CDAB	PRESENTE PRESENTE PRESENTE AUSENTE

Problema C

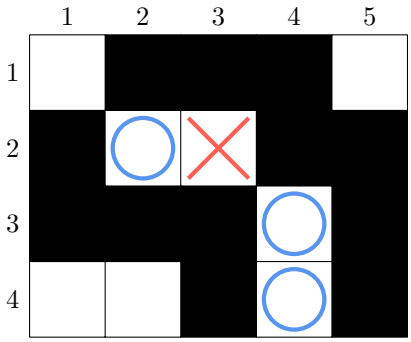
Cine

nombre clave: `cine`

Pedro y sus K amigos llegaron tarde al cine y para su desagradable sorpresa se dieron cuenta de que la mayoría de los asientos ya estaban ocupados. Aunque puede que ya no sea posible encontrar asientos contiguos, a Pedro le gustaría sentarse lo más cerca posible de sus amigos para disfrutar juntos de la película.

Los asientos en el cine están distribuidos en una grilla de N filas y M columnas. Las filas son numeradas de arriba hacia abajo de 1 a N y las columnas de izquierda a derecha de 1 a M. La posición de un asiento se representa con un par (x, y) donde x representa la columna e y la fila. Pedro quiere encontrar una configuración de asientos de modo que la distancia entre él y su amigo más lejano sea la mínima posible. La distancia entre Pedro y un amigo se define como el máximo entre la distancia horizontal y vertical. Específicamente, si Pedro está sentado en la posición (x_p, y_p) y su amigo k-ésimo en la posición (x_k, y_k), la distancia entre ellos se define como $\max(|x_p - x_k|, |y_p - y_k|)$. Llamaremos a la distancia entre pedro y su amigo más lejano la *distancia grupal*.

La siguiente imagen muestra un ejemplo para un cine con 4 filas y 5 columnas. La cantidad de amigos es $K = 3$ y se marca con negro los asientos que están ocupados. La imagen muestra una configuración con Pedro sentado en la posición $(3, 2)$ (marcado con una cruz) y con sus amigos sentados en las posiciones $(2, 2)$, $(4, 3)$ y $(4, 4)$ (marcados con un círculo). En este escenario, la distancia entre Pedro y su amigo más lejano (el amigo en la posición $(4, 4)$) es 2, es decir, la distancia grupal es 2. Esta distancia es además mínima, pues no hay ninguna otra configuración donde la distancia grupal sea menor que 2. Notar que esta no es la única configuración con distancia grupal mínima pues hay otras que también tienen distancia grupal 2.



Pedro está muy estresado pues la película está por comenzar y no sabe qué asientos comprar así que te ha pedido ayuda. Dada la descripción del cine y cuáles asientos están ocupados, tu tarea es encontrar la mínima distancia grupal posible.

Entrada

La primera línea contiene tres enteros N, M y K ($1 \leq N \cdot M \leq 10^6$, $1 \leq K \leq N \cdot M - 1$) que representan respectivamente la cantidad de filas, la cantidad de columnas, y la cantidad de amigos que tiene Pedro.

Cada una de las siguientes N líneas contiene M enteros. El entero x-ésimo en la línea y-ésima describe el asiento en la posición (x, y). El entero será 0 si el asiento está disponible o 1 si está ocupado.

Se garantiza que la cantidad total de ceros es mayor o igual a $K+1$, es decir, siempre es posible sentar a Pedro y a todos sus amigos.

Salida

La salida debe contener un único entero correspondiente a la distancia grupal mínima posible.

Subtareas y puntaje

Subtarea 1 (10 puntos)
Se probarán varios casos de prueba donde $N \leq 20$ y $M \leq 20$.

Subtarea 2 (20 puntos)
Se probarán varios casos de prueba donde $N \cdot M \leq 5000$.

Subtarea 3 (20 puntos)
Se probarán varios casos de prueba donde $N = 1$.

Subtarea 4 (50 puntos)
Se probarán varios casos de prueba sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
4 5 3 0 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 1 0 1	2

Problema D

Guerrera Dragona

nombre clave: `dragona`

Como todos sabrán, Anita fue recientemente declarada la nueva guerrera dragona, encargada de proteger del mal al Valle de la Paz. En su primera misión, nuestra heroína debe ir a la Torre de la Llama Sagrada, donde el malvado Marcos nuevamente amenaza con destruir todo a su paso. Para su pesar, al llegar a la torre se dio cuenta de que debe primero subir hasta el último piso, enfrentándose a su viejo enemigo en el proceso: las escaleras.

La Torre de la Llama Sagrada tiene h pisos numerados de 1 a h. Para cada piso i ($1 \leq i < h$) existe una escalera que puede ser usada para subir al piso $i+1$. Adicionalmente, la torre posee n ascensores. Estos están en muy malas condiciones y por lo tanto el ascensor j solo sirve para subir desde el piso a_j hasta el b_j, sin parar entre medio. Además, cabe notar que estos solo son ascensores y no descensores, por lo que no es posible utilizarlos para bajar.

Anita quiere minimizar la cantidad de escaleras que necesita subir para alivianar su sufrimiento, sin embargo, evitará a toda costa bajarlas porque tiene problemas en las rodillas. Específicamente, usando los ascensores y las escaleras para subir, Anita quiere encontrar una ruta desde el piso 1 al h que minimice la cantidad de escaleras que debe subir. Normalmente lo calcularía mentalmente, pero está demasiado ocupada planificando cómo derrotar a Marcos, por lo que te pidió ayuda con esta tarea. ¿Podrás ayudar a Anita a subir la torre para que enfrente a Marcos y así pueda salvar al Valle de la Paz?

Entrada

La primera línea de la entrada contiene dos enteros n y h ($0 \leq n \leq 10^5$, $1 \leq h \leq 10^6$) correspondientes a la cantidad de ascensores y la cantidad de pisos en la torre respectivamente.

Luego, siguen n líneas describiendo cada uno de los ascensores.

La j-ésima línea contiene dos enteros a_j, b_j ($1 \leq a_j < b_j \leq h$), que corresponden respectivamente al piso de inicio y al de llegada del ascensor j.

Salida

La salida debe tener un único entero, correspondiente a la cantidad mínima de escaleras con la que es posible ir desde el piso 1 al h.

Subtareas y puntaje

Subtarea 1 (5 puntos)

Se probarán varios casos de prueba donde $n = 1$, es decir, hay exactamente un ascensor.

Subtarea 2 (20 puntos)

Se probarán varios casos de prueba donde $b_j < a_{j+1}$ para todo j. Es decir, los ascensores están ordenados y no se intersectan.

Subtarea 3 (20 puntos)

Se probarán varios casos de prueba donde $n \leq 20$.

Subtarea 4 (55 puntos)

Se probarán varios casos de prueba sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
1 5 1 3	2

Entrada de ejemplo	Salida de ejemplo
2 10 1 4 2 10	1