
# Olimpiada Chilena de Informática 2024

*11 de Enero, 2024*

*Las siguientes personas participaron en la elaboración de este conjunto de problemas:*

## 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 popular programa de televisión Domingo Colosal quiere mejorar el sistema de selección popular que usan 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 aplaudan 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 participante 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 sistema de selección que se ejecuta en varias rondas. En este nuevo sistema, el animador también presenta a cada uno de los participantes de izquierda a derecha y pide al público que aplaudan por ellos. Si la intensidad de los aplausos para un participante $a$ es estrictamente menor que la intensidad de aplausos para el participante directamente a su izquierda el participante $a$ 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 al finalizar la ronda.

El proceso termina cuando al finalizar una ronda no hay ningún participante eliminado. Notar que esto implica que el proceso termina cuando una ronda parte con un solo participante, pues este no puede ser eliminado si no hay más participantes. 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 sistema 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 $a_1, a_2, \cdots a_N$. El entero $a_i$ corresponde a la intensidad en aplos en que el público aplaude para el participante $i$. Sin importar la ronda, el público siempre aplaude $a_i$ 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<br>4 5 2 1 3 4 | 4 |

**Explicación ejemplo:** Como se muestra a continuación, en este ejemplo se ejecutan cuatro rondas. En la primera ronda se eliminan el participante 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.

**Primera ronda:**

- Participante 1: 4 aplos
- Participante 2: 5 aplos
- Participante 3: 2 aplos (eliminado)
- Participante 4: 1 aplos (eliminado)
- Participante 5: 3 aplos
- Participante 6: 4 aplos

**Segunda ronda:**

- Participante 1: 4 aplos
- Participante 2: 5 aplos
- Participante 5: 3 aplos (eliminado)
- Participante 6: 4 aplos

**Tercera ronda:**

- Participante 1: 4 aplos
- Participante 2: 5 aplos
- Participante 6: 4 aplos (eliminado)

**Cuarta ronda:**

- Participante 1: 4 aplos
- Participante 2: 5 aplos

| Entrada de ejemplo | Salida de ejemplo |
| --- | --- |
| 5<br>5 2 1 3 2 | 3 |

# Problema B

## Cine

*nombre clave:* `cine`

Pedro y sus $K$ amigos llegaron tarde a la reproducción de la película en el cine y descubrieron que varios asientos ya están ocupados. Sin embargo, a Pedro le gustaría que él y sus amigos pudieran sentarse juntos para disfrutar de la película.

La sala de cine se representa como una matriz de números de tamaño $N \cdot M$, donde los asientos disponibles se marcan con un 0 y los ocupados con un 1. Pedro quiere encontrar una configuración de asientos para él y sus $K$ amigos de modo la distancia entre Pedro y su amigo más lejano sea la mínima posible.

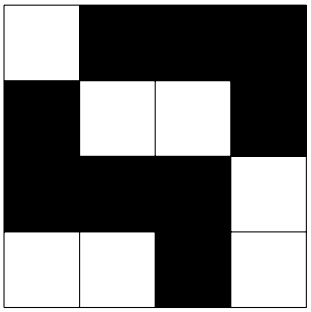

Figure 1: Ejemplo de un Cine de dimensiones $4 \times 4$.

La distancia entre Pedro y el $i$-ésimo amigo se define como $\max(|x_p - x_i|, |y_p - y_i|)$, donde $(x_p, y_p)$ representa la posición de Pedro en la matriz y $(x_i, y_i)$ la posición del $i$-ésimo amigo.

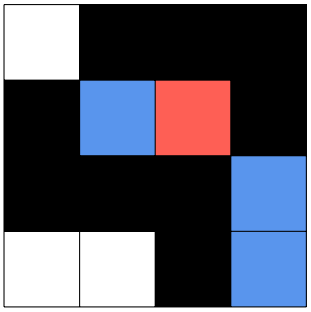

Figure 2: Una posible asignación optima para el Cine anterior si Pedro tiene $K = 3$ amigos.

Tu tarea es encontrar la distancia entre Pedro y su amigo más lejano, teniendo en cuenta que el puede elegir su propio asiento y la de sus $K$ amigos entre los asientos disponibles.

## 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 las dimensiones de la sala de cine y la cantidad de amigos que tiene Pedro, respectivamente.

Las siguientes $N$ líneas contienen $M$ enteros $A_{i,j}$ ($0 \leq A_{i,j} \leq 1$), donde 0 indica que el asiento está disponible y 1 que está ocupado.

Se asegura que la cantidad de ceros en la matriz es mayor o igual a $K + 1$.

## Salida

Un único número entero: la distancia mínima posible entre Pedro y su amigo más lejano en la mejor disposición encontrada.

## Subtareas y puntaje

### Subtarea 1 (20 puntos)

Se probarán varios casos de prueba donde $N \cdot M \leq 5000$.

### Subtarea 2 (30 puntos)

Se probarán varios casos de prueba donde $N = 1$.

### Subtarea 3 (50 puntos)

Se probarán varios casos de prueba sin restricciones adicionales.

## Ejemplos de entrada y salida

| Entrada de ejemplo | Salida de ejemplo |
|---|---|
| 4 4 3<br>0 1 1 1<br>1 0 0 1<br>1 1 1 0<br>0 0 1 0 | 2 |

# Problema C

## Guerrera Dragona

*nombre clave:* `dragona`

Como todos sabrán, Anita fue recientemente declarada la nueva guerrera dragona, encargada de proteger del mal al reino de Ocilandia. Hace unos momentos, se iluminó una Anita-señal frente a la torre de $h$ pisos.

¡Resulta que el malvado Marcos nuevamente amenaza con destruir la OCI!

Anita fue al rescate, pero se dio cuenta de que antes de poder derrotar a Marcos, debe subir hasta el último piso de la torre, enfrentándose contra su mayor enemigo en el proceso: las escaleras.

Afortunadamente, la torre contiene $n$ sets de ascensores, pero como están en malas condiciones, el ascensor $i$ solo sirve para subir desde el piso $a_i$ $b_i$, sin parar entre medio. Cabe notar que estos ascensores no son también descensores (como la mayoría de ascensores), por lo que no es posible utilizarlos para bajar.

Anita desea minimizar la cantidad de escaleras que necesita subir para alivianar su sufrimiento, sin embargo, evitará a toda costa bajar escaleras porque tiene problemas en las rodillas.

Para poder planificar bien su viaje, necesita saber la cantidad mínima de escaleras que necesita 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 llegar a Marcos?

### Entrada

La primera línea de la entrada contiene dos enteros $n$ ($0 \leq n \leq 10^5$) y $h$ ($1 \leq h \leq 10^5$) 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 i-ésima línea contiene dos enteros $a_i, b_i$ ($1 \leq a_i < b_i \leq h$), que corresponde respectivamente al piso de inicio y al de llegada del ascensor.

### Salida

La salida debe tener un único entero, correspondiente a la cantidad mínima de escaleras que Anita debe subir.

### Subtareas y puntaje

#### Subtarea 1 (?? puntos)

Se probarán varios casos de pureba donde $n \leq 1$.

**Subtarea 2 (?? puntos)**
Se probarán varios casos de prueba donde $b_i < a_{i+1}$ para todo $i$. Es decir, los ascensores están ordenados y no se intersectan.

**Subtarea 3 (?? puntos)**
Se probarán varios casos de prueba donde $n \leq 10^3$ y $h \leq 10^3$.

**Subtarea 4 (?? puntos)**
Se probarán varios casos de prueba sin restricciones adicionales

## Ejemplos de entrada y salida

| Entrada de ejemplo | Salida de ejemplo |
|---|---|
| 1 5<br>1 3 | 2 |

| Entrada de ejemplo | Salida de ejemplo |
|---|---|
| 2 10<br>1 4<br>2 10 | 1 |

# Problema D
## Sopa de letras

*nombre clave:* `sopa`

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 el jugador debe buscar palabras ocultas en una cuadrícula de $n \times n$ casillas con letras.

Cada año la ICPC organiza una convención donde se presentan innovadoras variantes de juegos de palabras, y este año no fue la excepción. Esta vez se anunció que traerían una nueva versión de la sopa de letras llamada "sopa de letras cilíndrica". La sopa de letras cilíndrica funciona igual que una sopa de letras normal, solo que además la cuadrícula es colocada alrededor de un cilindro.

En particular, las reglas de la sopa de letras cilíndrica 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.
- La cuadrícula está colocada alrededor de un cilindro. Es decir, después de la última letra de cada fila viene la primera letra de la misma fila, y antes de la primera letra de cada fila viene la última letra de la misma fila.

Como ya has demostrado tus habilidades de lógica y programación clasificando a la final nacional de la OCI, decides hacer un programa que resuelva una sopa de letras cilíndricas y así impresionar a los jueces en la convención de la ICPC.

### Entrada

La primera línea de la entrada contiene el entero $n$ ($1 \leq n \leq 50$).

Cada una de las siguientes $n$ líneas contienen $n$ caracteres, describiendo cada fila de la cuadrícula de arriba hacia abajo.

La siguiente línea contiene un entero $m$ ($1 \leq m \leq 50$).

Cada una de las siguientes $m$ líneas contienen una palabra que se debe buscar en la cuadrícula, las cuales son de un largo menor o igual a $n$.

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 la sopa de letras cilíndrica o `AUSENTE` de lo contrario.

## Subtareas y puntaje

### Subtarea 1 (50 puntos)

Se probarán varios casos de prueba donde solo hay palabras escritas de forma vertical de arriba hacia abajo.

### 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<br>AJCOZ<br>TOBGA<br>HUELF<br>RPAKM<br>UXOCI<br>3<br>OCI<br>GATO<br>ICPC | PRESENTE<br>PRESENTE<br>AUSENTE |

| Entrada de ejemplo | Salida de ejemplo |
|---|---|
| 4<br>AABB<br>BBCC<br>CCDD<br>DDEE<br>4<br>ABCD<br>BA<br>BBAA<br>CDAB | PRESENTE<br>PRESENTE<br>PRESENTE<br>AUSENTE |