Olimpiada Chilena de Informática 2015-2023

Problemas de la OCI

Matías Fernández Taipe

Presentación

La Olimpiada Chilena de Informática (OCI) es una competencia nacional de programación para estudiantes de secundaria organizada por la Sociedad Chilena de Ciencia de la Computación.

La OCI es una competencia que busca fomentar el interés por la informática y la programación entre los estudiantes de secundaria en Chile. La competencia se lleva a cabo en varias etapas.

- Fase reginal: se lleva a cabo en varias ciudades de Chile y está abierta a todos los estudiantes. Los mejores de cada región se clasifican para la final nacional.
- Fase final nacional: se realiza en una región de Chile y se reúne a los mejores de cada región.
- Campamento IOI: los 10 mejores en la final nacional son seleccionados para participar en un campamento de entrenamiento de 2 semanas en Santiago, donde reciben entrenamiento especializado para prepararse para la IOI. Al final de este se seleccionan los 4 mejores para representar a Chile en la IOI.

La IOI es una competencia internacional de programación para estudiantes de secundaria. La competencia se lleva a cabo anualmente en diferentes países de todo el mundo. Son dos días de competencia, donde los estudiantes deben resolver cuatro problemas de programación en cada día. Los estudiantes son evaluados por un jurado internacional y reciben medallas de oro, plata o bronce según su desempeño.

Índice

OCI 2015

Regional

Problema A - ¿Cuánto pan es *una marraqueta*?

En la casa de Alejandra tendrán un asado familiar y su padre la ha enviado a comprar marraquetas para hacer choripanes. Alejandra no sabe exactamente cuánto pan es *una marraqueta* pues siempre ha comprado hallullas, así que antes de partir pregunta a su padre cuanto pan es *una marraqueta*. Su padre le responde que según recuerda, en las panaderías se encuentran como 4 rollos de pan pegados y que cada uno de estos rollos es *una marraqueta*. Alejandra queda extrañada, pues si los cuatro rollos vienen pegados deberían en su conjunto ser llamados *una marraqueta*. Para no discutir Alejandra simplemente acepta la definición de marraqueta de su padre y se dirige a la panadería para cumplir con la orden.

Por si no fuera ya el colmo al llegar a la panadería Alejandra queda aún más confundida, pues ve a un cliente comprar 2 marraquetas y salir con el conjunto de 4 rollos pegados. Al parecer en la panadería siguen la definición de la Asociación de Consumo de Marraquetas (ACM) y consideran que *una marraqueta* son dos rollos de pan pegados. En la panadería solo pueden vender una cantidad entera de marraquetas y Alejandra debe pedirlas según la definición de la ACM. Dada la cantidad de marraquetas que quiere el padre según su definición, ¿podrías ayudar a Alejandra a saber cuántas marraquetas pedirle al panadero? Notar que como solo se puede comprar una cantidad entera de marraquetas no siempre es posible llevar la cantidad exacta de marraquetas según la definición del padre. Si este es el caso Alejandra puede llevar más marraquetas, pero siempre debe comprar la menor cantidad posible.

Entrada

La entrada consiste en una línea con un único entero N que representa la cantidad de marraquetas que Alejandra debe comprar según la definición de su padre.

Salida

Debes imprimir una línea con un único entero correspondiente a la menor cantidad de marraquetas según la definición de la ACM que Alejandra debe comprar para llevar a su padre la cantidad de marraquetas que solicitó según su definición.

Subtareas y puntajes

Subtarea 1 (30 puntos)

Se probarán varios casos donde $0 < N \leq 100$ y siempre es posible comprar la cantidad exacta de marraquetas.

Subtarea 2 (70 puntos)

Se probarán varios casos donde $0 < N \leq 100$ y sin restricciones adicionales.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
78	39

Entrada de ejemplo	Salida de ejemplo
7	4

Solución

Primero que todo hagamos la conversión entre las dos defiinciones de *una marraqueta*. Según la definición del padre, *una marraqueta* son 4 rollos de pan pegados, mientras que según la definición de la ACM, *una marraqueta* son 2 rollos de pan pegados. Por lo tanto, para pasar de la definición del padre a la definición de la ACM debemos dividir por 2.

Se puede seguir el siguiente razonamiento:

$$1 \text{ marraqueta para el papá} \to 1 \text{ rollo} \to \frac{1}{2} \cdot (2 \text{ rollos}) \to \frac{1}{2} \text{ marraqueta para la ACM}$$

Es claro que si se nos pide una cantidad par de marraquetas entonces basta con pedir la mitad en la panadería; e.g. el papá pide 8 marraquetas (8 rollos) entonces Alejandra pide 4 marraquetas según la ACM ($2 \cdot 4$ rollos $= 8$ rollos).

Si se pide una cantidad impar entonces hay que pedir la función para que así no falte; e.g. el papá pide 7 marraquetas (7 rollos) entonces Alejandra pide 4 marraquetas según la ACM ($2 \cdot 4$ rollos $= 8$ rollos > 7 rollos).

Así el código nos queda:

```cpp
#include <iostream>
using namespace std;

int main() {
  int n;
  cin >> n;
  if (n % 2 == 0) {
    cout << n / 2 << endl;
  } else {
    cout << (n + 1) / 2 << endl;
  }
}
```

También se puede hacer el truco de sumar 1 a N antes de dividirlo por 2, pues así nos aseguramos de que la división siempre redondee hacia arriba. Esto es porque si N es par entonces $N + 1$ es impar y el resultado de la división no cambia.

```cpp
#include <iostream>
using namespace std;

int main() {
  int n;
  cin >> n;
  cout << (n + 1) / 2 << endl;
}
```

Final nacional

Problema A - Partido de ping-pong

Jota Pe y Nelman son fanáticos del ping-pong. Cada vez que se juntan aprovechan de jugar un partido. Sus partidos no son profesionales y siguen las siguientes reglas que ellos mismos han inventado:

- El partido se juega a N puntos y el primero que llega a los N puntos gana.
- La cantidad de puntos, es decir, el valor de N, puede ser distinto en cada partido.
- En cada partido uno de los dos comienza sacando.
- Cada jugador conserva su saque hasta perder un punto.

Lo que no saben Jota Pe y Nelman es que sus partidos son muy predecibles. Resulta que cada vez que a Jota Pe le toca sacar tiene una racha de A puntos, es decir, gana A puntos seguidos y luego pierde el saque. Lo mismo pasa con Nelman, cada vez que a él le toca sacar tiene una racha de B puntos y luego pierde su saque. El largo de las rachas, es decir, el valor de A y B, depende del día en que jueguen. Notar que cuando un jugador pierde el saque el oponente gana un punto.

Jota Pe y Nelman quedarían muy sorprendidos si alguien fuera capaz de adivinar quién de los dos va a ganar un partido antes de que lo jueguen. ¿Qué tal si haces un programa para sorprenderlos?

Entrada

La entrada está compuesta de dos líneas.

La primera línea contiene dos enteros N y P. N corresponde a la cantidad de puntos a los que se jugará el partido. P es un entero que indica quién comenzará sacando (Jota Pe = 1 y Nelman = 2). La siguiente línea contiene dos enteros A y B, que representan respectivamente el largo de las rachas de Jota Pe y de Nelman.

Salida

Tu programa debe imprimir un 1 si Jota Pe es quien ganará el partido y un 2 si Nelman es quien ganará el partido.

Subtareas y puntajes

Subtarea 1 (10 puntos)
Se probarán varios casos donde $1 \leq N \leq 100$, $P = 1$, y $A = B = 0$.

Subtarea 2 (10 puntos)
Se probarán varios casos donde $1 \leq N \leq 100$, $P = 1$, y $A = B = 1$.

Subtarea 3 (20 puntos)
Se probarán varios casos donde $1 \leq N \leq 100$, $A = B$, y $0 \leq A, B \leq N$.

Subtarea 4 (60 puntos)
Se probarán varios casos donde $1 \leq N \leq 100$, y $1 \leq A, B \leq N$.

Ejemplos de entrada y salida

Entrada de ejemplo	Salida de ejemplo
11 1 0 0	2

Entrada de ejemplo	Salida de ejemplo
11 1 1 1	1

Entrada de ejemplo	Salida de ejemplo
7 2 2 2	2

Entrada de ejemplo	Salida de ejemplo
6 1 2 3	2

Solución

Como N es un valor chico (a lo más 100) entonces podemos simular el juego hasta que alguno de los dos obtenga los N puntos.

Una manera fácil de simularlo es teniendo variables que guarden los puntos de cada jugador y hagamos un bucle donde asignemos un punto según cuantos turnos quedan de racha al jugador que saca. Si ya no quedan turnos de racha entonces asignamos un punto al otro y cambiamos quien saca.

```
#include <iostream>
using namespace std;

int main() {
  int n, p, a, b;
  cin >> n >> p >> a >> b;
  int jota = 0, nelman = 0;
  int contador_racha;
  // Asignamos el contador de racha según quien comienza sacando
  if (p == 1) {
    contador_racha = a;
  } else {
    contador_racha = b;
  }
  while (jota < n && nelman < n) {  // Mientras ninguno haya ganado
    if(contador_racha != 0){        // Si quedan turnos de racha
      if (p == 1) {
        jota++;
      } else {
        nelman++;
      }
      contador_racha--;
    }
    else{                           // Si no quedan turnos de racha
      if(p == 1){                   // cambiamos quien saca y
```

```
        p = 2;                          // reasignamos el contador de racha
        nelman++;
        contador_racha = b;
      }
      else{
        p = 1;
        jota++;
        contador_racha = a;
      }
    }
  }

  // Imprimimos el ganador
  if (jota == n) {
    cout << 1 << endl;
  } else {
    cout << 2 << endl;
  }
}
```

OCI 2021

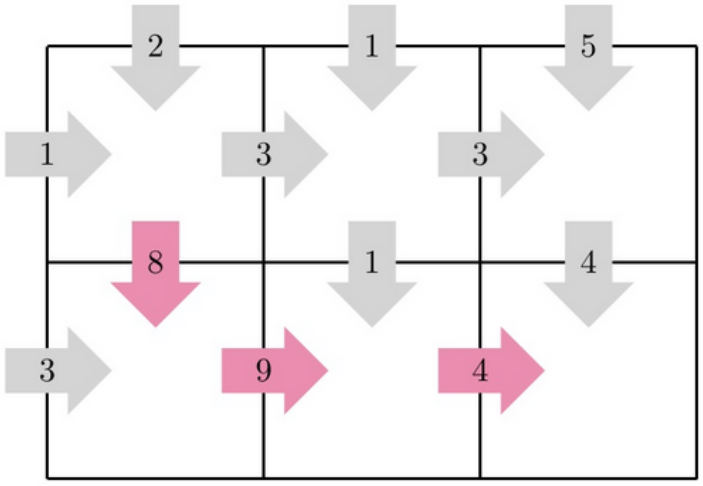
Final

Problema B - El mejor camino

Aburrida en su casa, Maiki acaba de inventar un nuevo juego. El juego se juega en una matriz de M filas y N columnas. Las filas de la matriz se enumeran de arriba a abajo entre 1 y M. Las columnas se enumeran de izquierda a derecha entre 1 y N. Identificamos con un par (i, j) a la casilla en la fila i y columna j.

Partiendo de la casilla $(1, 1)$, en cada turno uno puede moverse desde la casilla actual a la casilla inmediatamente abajo o inmediatamente a la derecha. Cada vez que uno se mueve a una nueva casilla se asigna un puntaje. El puntaje es distinto para cada casilla y depende de si uno se mueve a la casilla viniendo desde la izquierda o desde arriba. El objetivo del juego es llegar a la casilla (M, N) sumando la mayor cantidad de puntaje.

La siguiente figura muestra un ejemplo para $M = 2$ y $N = 3$ Las flechas representan los puntajes asociados a cada casilla. Notar que las casillas del borde superior tienen un puntaje asociado a llegar desde arriba. Estos puntajes son irrelevantes pues uno nunca puede moverse a estas casillas desde arriba. Similarmente, las casillas del borde izquierdo tienen un puntaje asociado a llegar desde la izquierda a pesar de ser irrelevante.



Las flechas marcadas de color **rojo** representan un posible camino. Partiendo desde la casilla $(1, 1)$ primero se mueve hacia la casilla $(2, 1)$ obteniendo 8 puntos. Posteriormente se mueve dos veces a la derecha obteniendo respectivamente en cada movimiento 9 y 4 puntos. El puntaje total es entonces $8 + 9 + 4 = 21$. Notar que cualquier otro camino obtiene un menor puntaje y por lo tanto 21 es el puntaje máximo que es posible obtener.

Dada una matriz con los puntajes asociados a cada casilla, tu tarea es encontrar el puntaje máximo que es posible obtener en el juego.

Entrada

La primera línea de la entrada contiene dos enteros M y N ($1 \leq N \leq 500$ y $1 \leq M \leq 500$) correspondientes respectivamente a la cantidad de filas y columnas en la matriz.

A continuación siguen M líneas cada una conteniendo N enteros mayores o iguales que cero y menores que 10^6. El j-ésimo entero de la línea i-ésima corresponde al puntaje asociado a moverse desde arriba a la casilla (i, j).

Salida

La salida consiste en un entero - el puntaje máximo que es posible obtener en el juego.

Subtareas y puntajes

Subtarea 1 (10 puntos)

Se probarán varios casos en que $M = 1$ (ver primer caso de ejemplo).

Subtarea 2 (10 puntos)

Se probarán varios casos en que el puntaje asociado a moverse desde arriba es igual al puntaje de moverse desde la izquierda y además este es igual para todas las celdas (ver segundo caso de ejemplo).

Subtarea 3 (30 puntos)

Se probarán varios casos en que para cada celda el puntaje de moverse desde arriba es igual al puntaje de moverse desde la izquierda. Este valor puede ser distinto para celdas distintas (ver tercer caso de ejemplo).

Subtarea 4 (50 puntos)

Se probarán varios casos sin restricciones adicionales (ver cuarto caso de ejemplo).

Ejemplos de entrada y salida

Entrada de ejemplo	**Salida de ejemplo**
1 6 0 0 0 0 0 0 0 2 3 1 4 1	11

Entrada de ejemplo	**Salida de ejemplo**
3 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	12

Entrada de ejemplo	**Salida de ejemplo**
4 2 1 2 4 2 3 4 1 3 1 2 4 2 3 4 1 3	14

Entrada de ejemplo	Salida de ejemplo
2 3 2 1 5 8 1 4 1 3 3 3 9 4	21

Solución

Pensemos este problema de forma recursiva. Si estamos en una casilla (i, j) entonces se tuvo que haber llegado o por la izquierda o por arriba. Suponiendo que se tienen ya las sumas máximas para llegar a $(i-1, j)$ y $(i, j-1)$ entonces la suma máxima hasta la casilla (i, j) está dada por la siguiente identidad:

$$\text{max\_sum}(i, j) = \max(\text{max\_sum}(i-1, j) + \text{arriba}(i, j), \text{max\_sum}(i, j-1) + \text{izquierda}(i, j))$$

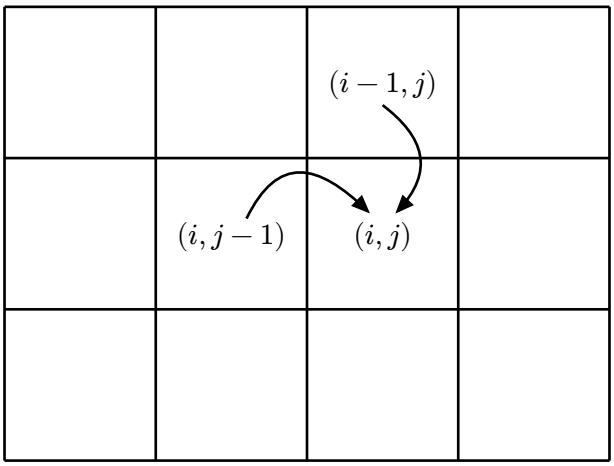


Figura 1: Posibles casillas de procedencia para llegar a (i, j)

Donde $\text{arriba}(i, j)$ es el puntaje que se suma llegando desde arriba, y lo análogo para $\text{izquierda}(i, j)$.

Esta fórmula es válida siempre y cuando $0 < i \leq M$ y $0 < j \leq N$.

Veamos que $\text{max\_sum}(0, 0) = 0$ ya que es por donde se parte y no se ha sumado nada aún.

Cuando $i = 0$ y $j \neq 0$, nos estaríamos fijando en la primera columna. Donde la única forma de llegar a (i, j) es ir hacia bajos desde $(0, 0)$. Por lo que está dado por:

$$\text{max\_sum}(i, j) = \text{max\_sum}(i-1, j) + \text{arriba}(i, j)$$

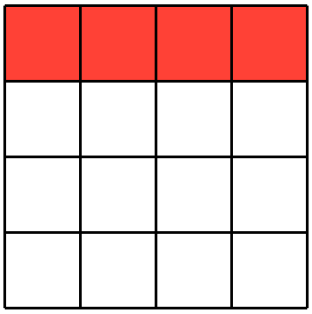


Figura 2: Primera fila donde $i = 0$

Por otro lado cuando $i \neq 0$ y $j = 0$, nos estaríamos fijando en la primera fila. Donde la única forma de llegar a (i, j) es ir hacia la derecha desde $(0, 0)$. Por lo que está dado por:

$$\text{max\_sum}(i, j) = \text{max\_sum}(i, j - 1) + \text{izquierda}(i, j)$$

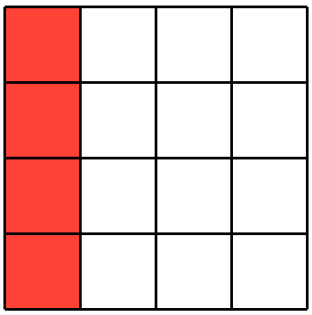


Figura 3: Primera columna donde $j = 0$

Así en el código podemos hacer un array de dos dimensiones donde guardaremos las sumas máximas hasta cada casilla. Y luego recorrer la matriz de izquierda a derecha y de arriba hacia abajo, calculando las sumas máximas para cada casilla. Finalmente la suma máxima hasta la casilla $(M - 1, N - 1)$ es la solución al problema.

```
#include <bits/stdc++.h>
using namespace std;

#define rep(i, n) for(int i = 0; i < n; i++)

int main() {
  int n, m;
  cin >> n >> m;

  int arriba[n][m], izquierda[n][m];

  rep(i, n) rep(j, m) cin >> arriba[i][j];
  rep(i, n) rep(j, m) cin >> izquierda[i][j];

  int max_sum[n][m];

  rep(i, n){
    rep(j, m){
      if(i == 0 && j == 0)
        max_sum[0][0] = 0;
      else if(i == 0)
        max_sum[i][j] = max_sum[i][j - 1] + izquierda[i][j];
      else if(j == 0)
        max_sum[i][j] = max_sum[i-1][j] + arriba[i][j];
      else
        max_sum[i][j] = max(max_sum[i][j - 1] + izquierda[i][j], max_sum[i-1][j] +
arriba[i][j]);
    }
  }

  cout << max_sum[n-1][m-1] << '\n';

  return 0;
}
```