



# Programación Competitiva

**B.S. Rodolfo Mercado Gonzales**  
**Universidad Nacional de Ingeniería**

# Grupo de Programación Competitiva UNI

- ❑ Grupo formado por estudiantes y egresados UNI.
- ❑ Capacitamos a los estudiantes en algoritmos y estructuras de datos a nivel avanzado.
- ❑ Nuestro objetivo es obtener los primeros puestos en los concursos de programación competitiva.



Facebook: ProgramacionCompetitivaUNI

# Programación Competitiva

Consiste en resolver problemas “lo antes posible”, a través de programas computacionales, teniendo en cuenta los siguientes recursos :

- ❑ Tiempo de ejecución del programa
- ❑ Memoria usada por el programa



# Programación Competitiva

“El objetivo es formar informáticos que estén preparados para producir mejor software y enfrentarse en el futuro a problemas de investigación.”

## Fundadores ACM-ICPC



# Tiempo de ejecución

- ❑ Es el tiempo que le toma a un programa procesar una determinada entrada.
- ❑ En las competencias hay un límite para el tiempo de ejecución, generalmente es **1 segundo**.

Time limit exceeded on test 12	2000 ms	16200 KB
--------------------------------	---------	----------

# Memoria

- ❑ Las variables usadas en un programa son las que principalmente ocupan espacio en memoria.
- ❑ En las competencias hay un límite para el uso de memoria, generalmente es **256 MB**.

Memory limit exceeded on test 1	93 ms	262100 KB
---------------------------------	-------	-----------

# ¿Cómo son los problemas?

- ❑ Nos enfrentamos a problemas algorítmicos, que previamente han sido resueltos al menos por el autor.
- ❑ No son imposibles de resolver.

# ¿Cómo son los problemas?

Enunciados en inglés, con la siguiente estructura :

- ☐ Restricciones del tiempo de ejecución y de la memoria.
- ☐ Descripción del problema (statement).
- ☐ Descripción de la entrada y salida (input, output)
- ☐ Restricciones de los datos de entrada (constraints).
- ☐ Ejemplo de entrada y salida (sample input, sample output).



# ¿Cómo son los problemas?

## A. Nearest Minimums

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given an array of  $n$  integer numbers  $a_0, a_1, \dots, a_{n-1}$ . Find the distance between two closest (nearest) minimums in it. It is guaranteed that in the array a minimum occurs at least two times.

### Input

The first line contains positive integer  $n$  ( $2 \leq n \leq 10^5$ ) — size of the given array. The second line contains  $n$  integers  $a_0, a_1, \dots, a_{n-1}$  ( $1 \leq a_i \leq 10^9$ ) — elements of the array. It is guaranteed that in the array a minimum occurs at least two times.

### Output

Print the only number — distance between two nearest minimums in the array.

### Examples

input
2 3 3
output
1

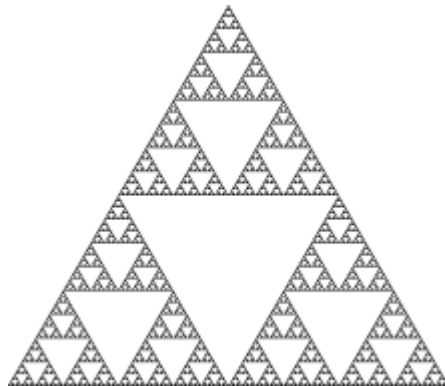
# ¿Qué conocimientos debo tener/adquirir?

- ❑ Razonamiento lógico-matemático.
- ❑ **Algoritmos y estructuras de datos.**
- ❑ Un lenguaje de programación a nivel básico.



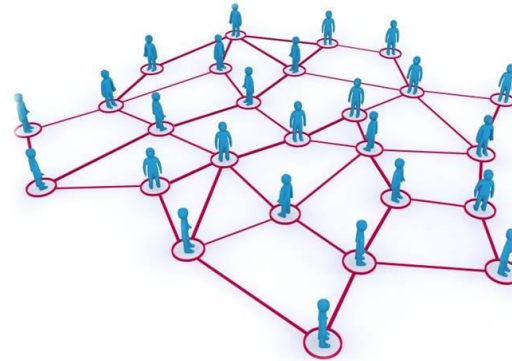
# Algoritmos y Estructuras de Datos

- ❑ Análisis de Algoritmos
- ❑ Standard Template Library
- ❑ Fuerza Bruta
- ❑ Recursividad
- ❑ Backtracking
- ❑ Divide y Vencerás
- ❑ Búsqueda Binaria
- ❑ Programación Dinámica



# Algoritmos y Estructuras de Datos

- ❑ Teoría de números
- ❑ Geometría Computacional
- ❑ Grafos
- ❑ Estructuras de Datos Avanzadas
- ❑ Procesamiento de Cadenas
- ❑ Teoría de Juegos



# Competencias de programación

- ❑ Las competencias consisten en resolver un conjunto de problemas algorítmicos en un determinado tiempo.
- ❑ Gana quien resuelva la mayor cantidad de problemas. En caso de empate, quien tuvo menor penalidad será el vencedor.



# Evaluación de una solución

1. Se envía el código fuente a un juez online.



# Evaluación de una solución

2. Este juez se encargará de testear tu “solución” con un input secreto.

Who	Problem	Lang	Verdict
system_1	<a href="#">911D - Inversion Counting</a>	GNU C++	Running on test 1
army_of_one	<a href="#">459A - Pashmak and Garden</a>	GNU C++11	Running on test 6
manojkannekanti	<a href="#">911A - Nearest Minimums</a>	GNU C++14	Running on test 4

# Evaluación de una solución

3. En unos segundos el juez dará un veredicto para tu solución.

AC (accepted ) – Tu solución es correcta.

WA (wrong answer) – Tu programa da una respuesta incorrecta.

TLE (time limit exceeded ) – Tu programa tarda mucho tiempo.

MLE (memory limit exceeded ) – Tu programa usa demasiada memoria.

RE (runtime error) – Tu programa se cae durante su ejecución.

CE (compilation error) – Tu programa no compila.



# Concursos en equipo



**IEEE**XTREME  
**PROGRAMMING**  
COMPETITION

# ACM-ICPC



- ❑ Competencia organizada por la ACM.
- ❑ Compiten equipos de 3 estudiantes.
- ❑ Consta de 2 fases de manera presencial, cada una de 5 horas :
  - **Regional Sudamérica/Sur:** Argentina, Perú, Chile, Bolivia, Uruguay y Paraguay (3 cupos para el mundial).
  - **Final Mundial :** todos los equipos clasificados de las distintas regionales.

# ACM-ICPC



# ACM-ICPC



# IEEEExtreme



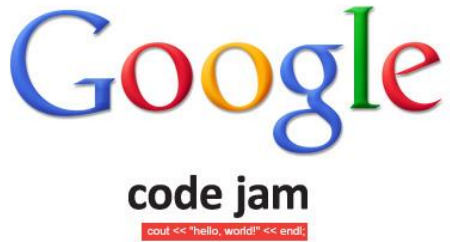
- ❑ Competencia organizada por la IEEE.
- ❑ Compiten equipos de 3 estudiantes.
- ❑ Consta de una sola fase online.
- ❑ El concurso dura 24 horas.



# IEEExtreme



# Concursos individuales



# Beneficios de competir

1. Dominarás algoritmos y estructuras de datos.
2. Mejorarás tus habilidades para la resolución de problemas.
3. Trabajarás en equipo y bajo presión.
4. Aprenderás C++ y Python.

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

struct Tuple{
    int gcd, x, y;
    Tuple( int a, int b, int c ){
        gcd = a, x = b, y = c;
    }
};

Tuple extGcd( int a, int b ){
    if ( b == 0 ) return Tuple( a, 1, 0 );
    Tuple ret = extGcd( b, a % b );
    return Tuple( ret.gcd, ret.y, ret.x - a/b * ret.y );
}

int main( ){
    Tuple ans = extGcd( 3, 5 );
    cout << ans.x << " " << ans.y << endl;
}
```



# Beneficios de competir

5. Podrás participar en campamentos de programación.

Argentina



# Beneficios de competir

6. Incrementarás tus posibilidades de trabajar en las mejores empresas del ámbito tecnológico a nivel mundial.



Minimum qualifications:

- Currently enrolled in a full-time Bachelor's in Computer Science or related technical field, returning to BA/BS program or enrolled in another full-time degree program after completion of the internship.
- Completed projects (inside or outside of school) or classes focused on Data Structures and Algorithms; experience with algorithms, using data structures to solve problems, and interpreting algorithms and contributing ideas to their development.
- Experience writing code fixes and tools to solve problems in one or more of the following languages: C, C++, Java, JavaScript, or Python (e.g., remove duplicate elements from a list).
- Experience with linear coding and use language features when necessary (e.g., data structures, branching, function-calls, and conditionals).



# Beneficios de competir

6. Incrementarás tus posibilidades de trabajar en las mejores empresas del ámbito tecnológico a nivel mundial.



# Beneficios de competir

7. Incrementarás tus posibilidades de hacer estudios en el extranjero.



# Beneficios de competir

8. Podrás ganar muchos premios.

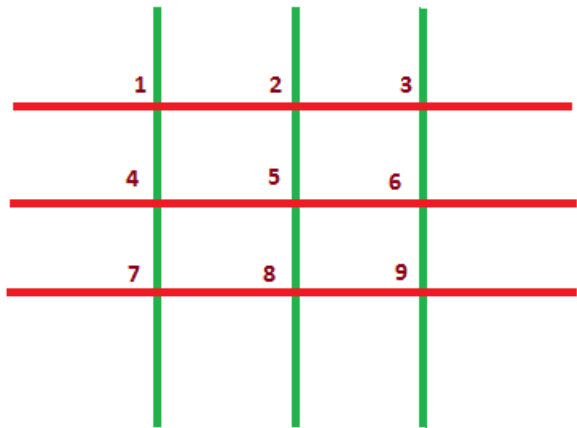


# Nuestro primer problema

Egor y Peter empiezan un juego sobre una grilla compuesta de  $n$  varillas horizontales y  $m$  verticales.

Ambos juegan en turnos, siendo Egor el que empieza. Durante su turno, el jugador debe escoger un punto de intersección de la grilla y retirar todas aquellas varillas que pasen por dicho punto.

Un jugador perderá si es que no puede hacer movimientos (ya no hay puntos de intersección).



# Nuestro primer problema

Asumir que ambos juegan óptimamente (buscan ganar). ¿Quién ganará el juego?

## Entrada

Se le darán los enteros  $n$  y  $m$ .

## Salida

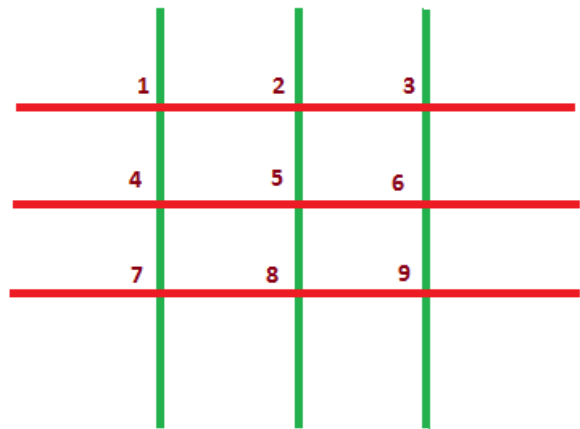
Indicar el nombre del ganador(Egor o Peter)

## Entrada Ejemplo

2 2

## Salida Ejemplo

Peter



[Fuente: codeforces – game with sticks](#)

# ¿Cómo afrontar un problema?

## Método Pólya

- ❑ **Entender el problema :** identificar los datos de entrada, salida y las restricciones.
- ❑ **Diseñar un plan:** recordar problemas similares, resolver el problema reduciendo restricciones (caso particulares), resolver casos generales, intentar con los algoritmos que conocemos, ir desde la solución “más ingenua” hasta la buscada.
- ❑ **Ejecutar el plan:** implementar nuestro algoritmo (si encontramos alguna dificultad es posible cambiar de estrategia).
- ❑ **Examinar la solución:** verificamos los pasos anteriores y testeamos nuestro programa.

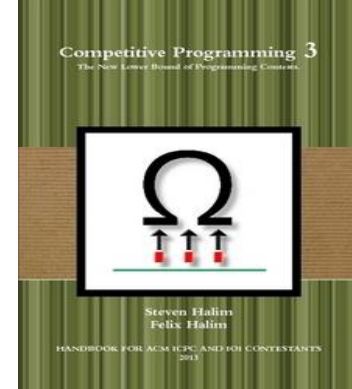
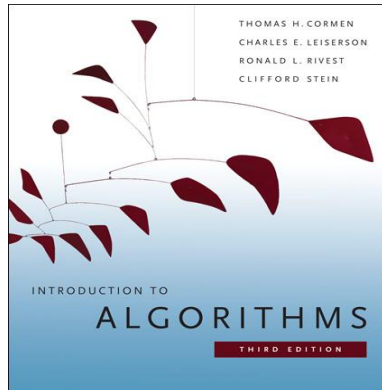


# Tips para ser competitivo

- ❑ **Tipear rápido** ([typingtest](#)).
- ❑ Identificar rápidamente el tipo de problema.
- ❑ Hacer análisis algorítmico.
- ❑ Dominar tu lenguaje de programación preferido.
- ❑ Testear tu solución.
- ❑ Practicar a full.
- ❑ Trabajo en equipo

# Referencias

- ❑ Cormen, Thomas - **Introduction to Algorithms**
- ❑ e-maxx - **E-Maxx Algorithms**
- ❑ Halim, Steven and Halim, Felix - **Competitive Programming 3**



¡ Good luck and have fun !