

Introduccion a la Programacion Competitiva

Federico Nahuel Quijada

Universidad Tecnológica Nacional - Facultad Regional Santa Fe

Training Camp 2025



Gracias Sponsors!

Organizador



Diamond Plus



GTS

Gracias Sponsors!

Platino



FOLDER IT

INTERNATIONAL
SOFTWARE COMPANY

Gold

NeuralSoft

Oro

 JERÁRQUICOS

Aliado



Outline

- ① Presentacion
- ② ¿Qué es un problema?
- ③ Operaciones
- ④ Estructuras útiles
- ⑤ Entrada y salida
- ⑥ Consejos personales

Outline

- 1 Presentacion
- 2 ¿Qué es un problema?
- 3 Operaciones
- 4 Estructuras útiles
- 5 Entrada y salida
- 6 Consejos personales

Presentacion



Anarap (Egipto '24)



Fruta Fresca (México '24)



Fruta Fresca (Kazajistán '24)



Champán en Lata (Brasil '25)

Outline

- ① Presentacion
- ② ¿Qué es un problema?
- ③ Operaciones
- ④ Estructuras útiles
- ⑤ Entrada y salida
- ⑥ Consejos personales

¿Qué es un problema?

Problema de ejemplo:

Juan tiene $0 \leq X \leq 90$ manzanas, y quiere repartirlas entre $0 \leq Y \leq 10$ amigos.

Suponiendo que quiere maximizar la cantidad de manzanas que reparte, y que todos reciban la misma cantidad,
¿cuántas manzanas recibiría cada uno?

Input:

2

15 2

17 5

Output:

7

5

Outline

- ① Presentacion
- ② ¿Qué es un problema?
- ③ Operaciones**
- ④ Estructuras útiles
- ⑤ Entrada y salida
- ⑥ Consejos personales

¿Cuántas operaciones entran en tiempo?

Podemos dividir en tres secciones:

- $\leq 10^8$ **operaciones**: Siempre entran en 1 segundo.
- **Entre 10^8 y 10^9 operaciones**: Puede funcionar, dependerá principalmente de qué hace cada operación.
- $> 10^9$ **operaciones**: **PELIGRO**: casi siempre lleva a TIME LIMIT.

Los tipos de datos tienen sus límites también:

- `int` \Rightarrow desde -2^{31} hasta $2^{31} - 1$ ($\approx \pm 2 \cdot 10^9$).
- `long long` \Rightarrow desde -2^{63} hasta $2^{63} - 1$ ($\approx \pm 9 \cdot 10^{18}$).

Al superar esos límites, llegamos al famoso **OVERFLOW.**

¿Cómo evitar el overflow?

El **overflow** es un error **MUY COMÚN**.

El uso de **aritmética modular** (A.K.A. operaciones MOD) es fundamental:

- `%` (MOD): devuelve el resto de una división.
- `ADD mod`: $(a + b) \bmod m$
- `SUB mod`: $((a - b) \bmod m) + m \bmod m$
- `MUL mod`: $(a \cdot b) \bmod m$

Recomendaciones:

- Usar tipos de 64 bits: `long long` en C++.
- Es mejor usar MOD de más que de menos.

Outline

- ① Presentacion
- ② ¿Qué es un problema?
- ③ Operaciones
- ④ Estructuras útiles
- ⑤ Entrada y salida
- ⑥ Consejos personales

Operaciones comunes:

- `sort(v.begin(), v.end(), comparador);`

```
vector<int> v;  
sort(v.begin(), v.end());
```

- `lower_bound(v.begin(), v.end(), x);` (requiere orden)
- `set.lower_bound(x);` (asi se usa en sets)
- `upper_bound();`

Estructuras importantes

Hay algunas estructuras que son **muy importantes**:

- vector
- queue
- deque
- set
- map

- `vector<T>` en C++, con `push_back` y `pop_back`
- `ArrayList<T>` en Java, con `add` y `remove(list.size()-1)`
- `list` en Python, con `append` y `pop`
- Acceso con `v[pos]` o `v.get(pos)`
- Todas estas operaciones son **$O(1)$**

- `queue<T>` en C++, con `push`, `front`, `pop`
- `ArrayDeque<T>` en Java, con `add`, `getFirst`, `remove`
- `collections.deque` en Python, con `append`, `deque[0]`, `popleft`
- Se comporta como una cola (FIFO)
- Operaciones en **$O(1)$**

- `deque<T>` en C++, con `push_front`, `push_back`, `pop_front`, `pop_back`
- `ArrayDeque<T>` en Java, con `addFirst`, `addLast`, `removeFirst`, `removeLast`
- `collections.deque` en Python, con `appendleft`, `append`, `popleft`, `pop`
- Acceso en **$O(1)$**
- Todas las operaciones anteriores son también **$O(1)$**

- `set<T>` en C++
- `TreeSet<T>` en Java
- En Python no hay un equivalente ordenado nativo
- Mantiene elementos **únicos y ordenados**
- Permite insertar, borrar, consultar existencia, `lower_bound`, `upper_bound`
- Todas en **$O(\log N)$**
- Variantes:
 - `unordered_set` (C++) / `HashSet` (Java): $O(1)$ amortizado
 - `multiset`: permite elementos repetidos

Map

- `map<K, V>` en C++
- `TreeMap<K, V>` en Java
- En Python: `collections.OrderedDict` (aproximado)
- Mantiene pares **clave** \rightarrow **valor**
- Acceder, insertar, borrar en **$O(\log N)$**
- Variante: `unordered_map` (hash, $O(1)$ amortizado)

Outline

- ① Presentacion
- ② ¿Qué es un problema?
- ③ Operaciones
- ④ Estructuras útiles
- ⑤ Entrada y salida**
- ⑥ Consejos personales

Entrada y salida en C++

Forma elegante de entrada/salida:

```
using namespace std;
int main(){
    int a,b;
    cin >> a >> b;
    cout << a << " " << b << '\n'; // << endl; (es mas lento)
}
```

Leer y escribir consume tiempo, por lo tanto hay que tener cuidado.

Recomendaciones en C++:

```
int main() {  
    ios::sync_with_stdio(false); // no mezclar con scanf/printf  
    cin.tie(nullptr); // no espera cout antes de leer  
    int a, b;  
    cin >> a >> b;  
    cout << a << " " << b << '\n';  
}
```


Outline

- ① Presentacion
- ② ¿Qué es un problema?
- ③ Operaciones
- ④ Estructuras útiles
- ⑤ Entrada y salida
- ⑥ Consejos personales

Consejos personales

- Usar una tablita para anotar ideas por problema

A	B	C	D	E	F	G	H	I	J	K	L	M	N
			contar swaps en algoritmo de ordenamiento loco						la resp es siempre < 30 , y probablemente bastante menor inclusive, tiene pinta que sale con dp o bfs	grafo completo grande pintado en dos colores, te dan el coloreo de un color, contar una porqueria, no parece facil		feo, pero capaz no tan difícil. ESPERA PC	nano
		si pudiera preguntar el mas frecuente en $[0, N]$ - $[L, R]$ + $[L, R]$ para todo L, R ganaria		grafos		math		para mi es - si ves una monedita, aumentas la siguiente posicion con la que chocarias arriba, y con la que chocarias abajo, pero antes queda e viador de alguna de las dos. Si usa			tiene pinta de greedy, pero no es tan trivial como parece		fede
PC	Dado un grafo completo con pesos G , llevarlo a un grafo completo con pesos H , ejecutando una operacion la cantidad de veces que quieras. (no es necesario		imprimir la secuencia y ver algun patron???????						espera PC				juampi

- Aprovechar simulaciones para mejorar la dinámica
- Mantener una buena comunicación con el equipo

Consejos personales

- Tener un template útil y propio

```
#include <bits/stdc++.h>
#define forr(i,a,b) for(int i=(a);i<(b);i++)
#define forn(i,n) forr(i,0,n)
#define dforn(i,n) for(int i=n-1;i>=0;i--)
#define forall(it,v) for(auto it=v.begin();it!=v.end();it++)
```

- Acostumbrarse a usar un notebook
- Aprender a estimar tiempos
- Leer editoriales → entender el **cómo**, no solo el código
- Usar IA como herramienta, no como solución

- **UPSOLVEAR**, esa es la clave

FedeNQ	B - In Case of an Invasion, Please...	C++20 (GCC 13-64)	Accepted	733 ms
FedeNQ	B - In Case of an Invasion, Please...	C++20 (GCC 13-64)	Wrong answer on test 19	718 ms
Fruta Fresca: marianoferesin, FedeNQ , Aristides	B - In Case of an Invasion, Please...	C++20 (GCC 13-64)	Wrong answer on test 10	2999 ms
Fruta Fresca: marianoferesin, FedeNQ , Aristides	B - In Case of an Invasion, Please...	C++20 (GCC 13-64)	Compilation error	0 ms

- No quedarse con un solo problema, leer otros.
- Intentar resolver casos mas simples, ver como mejorar la solucion.
- Disfrutar, tanto las competencias como los entrenamientos

Fin!! Gracias por asistir!
Muchos éxitos en la compe de hoy