progcompCL

CIPC 2024 - Día 1

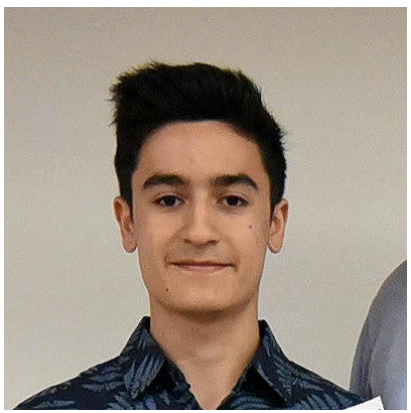
Introducción y C++

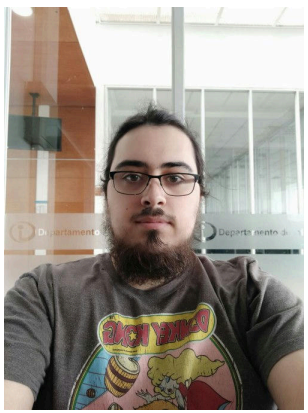
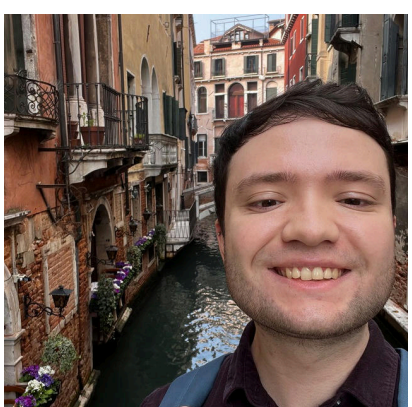
Gabriel Carmona T. Blaz Korecic

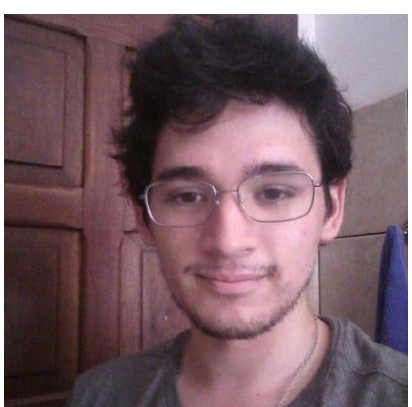
22 de Julio, 2024

Introducción

¿Quiénes somos?

Funcionamiento del campamento

Funcionamiento del campamento

- En la mañana:
  - Clase teórica [09:00–10:45]
  - Cafecito+galletitas [10:45–11:00]
  - Clase teórica [11:00–12:45]

Funcionamiento del campamento

- En la mañana:
  - Clase teórica [09:00–10:45]
  - Cafecito+galletitas [10:45–11:00]
  - Clase teórica [11:00–12:45]
- Almuerzo [12:45–13:30]

Funcionamiento del campamento

- En la mañana:
  - Clase teórica [09:00–10:45]
  - Cafecito+galletitas [10:45–11:00]
  - Clase teórica [11:00–12:45]
- Almuerzo [12:45–13:30]
- Simulación de competencia [13:30–18:30]

Funcionamiento del campamento

- En la mañana:
  - Clase teórica [09:00–10:45]
  - Cafecito+galletitas [10:45–11:00]
  - Clase teórica [11:00–12:45]
- Almuerzo [12:45–13:30]
- Simulación de competencia [13:30–18:30]

Cronograma completo en cipc.progcomp.cl

Programación Competitiva

International Collegiate Programming Contest (ICPC)

International Collegiate Programming Contest (ICPC)

ICPC en Chile

ICPC en Chile

Resolución de problemas

Resolución de problemas

Un problema consiste en un enunciado que nos dice:

- El problema a resolver
- Restricciones de tiempo y memoria
- Input con su formato y restricciones
- Formato del output
- Ejemplos

Watermelon

Veredictos

Time Limit Exceeded (TLE)

```python
w = int(input())
flag = false
while True:
  for i in range(1, w+1):
    for j in range(1, w+1):
      if i+j == w:
        flag = true

if flag: print("YES")
else: print("NO")
```

Veredictos

Memory Limit Exceeded (TLE)

```python
w = int(input())
flag = false
ans = [0] * 10000000000
for i in range(1, w+1):
  for j in range(1, w+1):
    if i+j == w:
      flag = true

if flag: print("YES")
else: print("NO")
```

Veredictos

Runtime Error

```python
# Puede ser acceder a un índice no existente
numeros = [1, 2, 3]
ans = numeros[5]

# O dividir por cero
a = 5
print(a/0)
```

Veredictos

Wrong Answer (WA)

```python
w = int(input())

if w%2 == 0:
  print("YES")
else:
  print("NO")
```

Veredictos

Presentation Error

```python
w = int(input())

if w%2 == 0 and w > 2:
    print("YES     ")
else:
    print("  NO")
```

Veredictos

Accepted (AC)

```python
w = int(input())

if w%2 == 0 and w > 2:
  print("YES")
else:
  print("NO")
```

Complejidad algorítmica

Análisis de complejidad

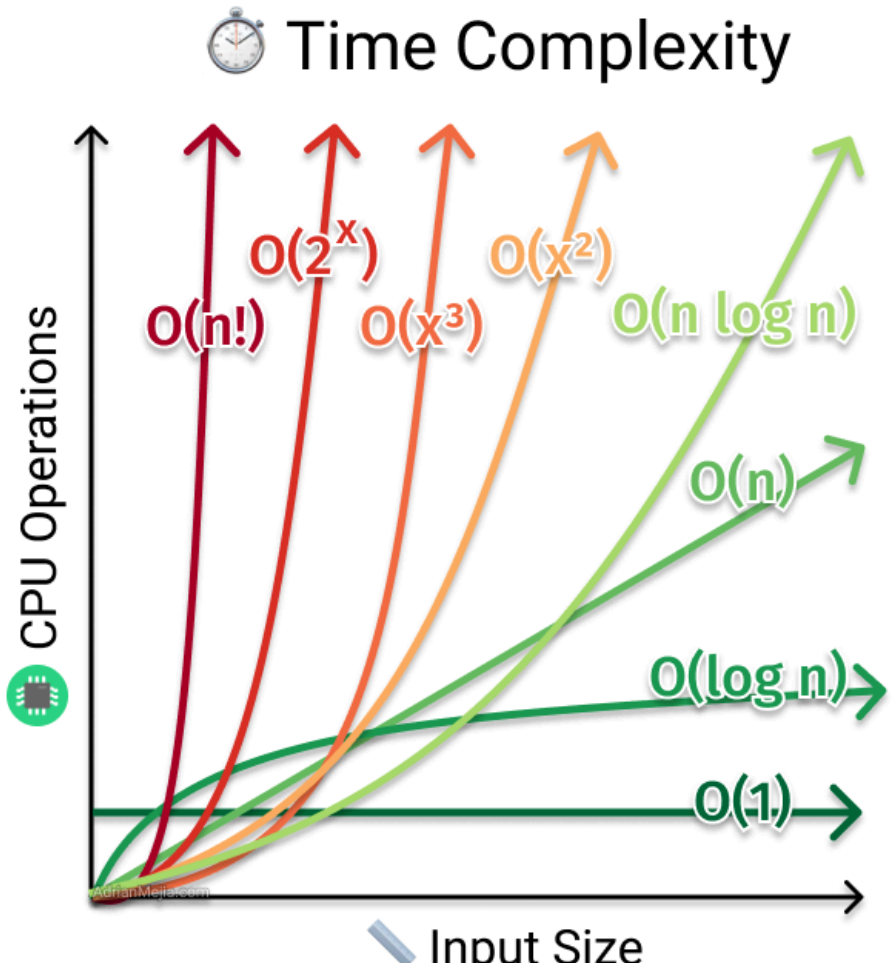
Analizar la complejidad de un algoritmo nos permite estimar qué tan **eficiente** es.

Análisis de complejidad

Analizar la complejidad de un algoritmo nos permite estimar qué tan **eficiente** es.

Decimos que un algoritmo es $O(f(n))$ si en el **peor caso** su tiempo de ejecución se comporta **asintóticamente** como $f(n)$.

Complejidades comunes



Cómo analizar complejidad

```python
suma = 0
for i in range(n):
  suma += 1
```

```python
suma = 0
for i in range(n):
  for j in range(m)
    suma += i*j
```

```python
if w%2 == 0:
  print("Es par")
else:
  for i in range(w):
    print(i)
```

```python
for i in range(n):
  j = 1
  while j < n:
    j *= 2
```

Cómo analizar complejidad

```python
suma = 0
for i in range(n):
  suma += 1
```

$$O(n)$$

```python
suma = 0
for i in range(n):
  for j in range(m)
    suma += i*j
```

```python
if w%2 == 0:
  print("Es par")
else:
  for i in range(w):
    print(i)
```

```python
for i in range(n):
  j = 1
  while j < n:
    j *= 2
```

Cómo analizar complejidad

```python
suma = 0
for i in range(n):
  suma += 1
```

$$O(n)$$

```python
suma = 0
for i in range(n):
  for j in range(m)
    suma += i*j
```

$$O(n \cdot m)$$

```python
if w%2 == 0:
  print("Es par")
else:
  for i in range(w):
    print(i)
```

```python
for i in range(n):
  j = 1
  while j < n:
    j *= 2
```

Cómo analizar complejidad

```python
suma = 0
for i in range(n):
  suma += 1
```

$$O(n)$$

```python
suma = 0
for i in range(n):
  for j in range(m)
    suma += i*j
```

$$O(n \cdot m)$$

```python
if w%2 == 0:
  print("Es par")
else:
  for i in range(w):
    print(i)
```

$$O(w)$$

```python
for i in range(n):
  j = 1
  while j < n:
    j *= 2
```

Cómo analizar complejidad

```python
suma = 0
for i in range(n):
  suma += 1
```

$$O(n)$$

```python
suma = 0
for i in range(n):
  for j in range(m)
    suma += i*j
```

$$O(n \cdot m)$$

```python
if w%2 == 0:
  print("Es par")
else:
  for i in range(w):
    print(i)
```

$$O(w)$$

```python
for i in range(n):
  j = 1
  while j < n:
    j *= 2
```

$$O(n \log_2 n)$$

Complejidad admitida según input

Tamaño del input (n)	Peor complejidad aceptada
<= [10..11]	$O(n!), O(n^6)$
<= [15..18]	$O(n^2 \cdot 2^n)$
<= [18..22]	$O(n \cdot 2^n)$
<= 100	$O(n^4)$
<= 400	$O(n^3)$
<= 2000	$O(n^2 \log n)$
<= 10000	$O(n^2)$
<= 1000000	$O(n \log n)$
<= 100000000	$O(n), O(\log n), O(1)$

Introducción a C++ y Librería Estándar