

Jonathan Asprilla Saavedra - 8926036 Tarea 1
2. Clasificar por orden asintótico las siguientes funciones

2.2.1

$$\rightarrow \log(\sqrt{n!}) \leq c \log(n!)$$

Aplicando euler

$$e \log(\sqrt{n!}) \leq e \cdot c \cdot \log(n!)$$

$$\sqrt{n!} \leq c \cdot n!$$

Para todo $n \geq n_0$

$$n \geq 2; \sqrt{2!} < 2!$$

$$c = 1$$

$$\rightarrow \log(n!) \leq cn^2 \log n$$

Aplicando euler

$$e \log(n!) \leq e \cdot cn^2 \log n$$

$$n! \leq cn^3$$

$$n=1, c=1$$

Para todo $n \geq n_0$ $n \geq 1; 1! \leq 1$

$$\rightarrow n^{2.5} \leq n^2 \log n \cdot c$$

$$c = 2$$

Es falso porque

$$n = 1$$

$$c = 1$$

$$1^{2.5} \leq 1^2 \log 1 \cdot 1$$

$$1 \leq 0$$

$$\therefore n^2 \log n \leq cn^{2.5}$$

Para todo $n \geq 1$

$$L \rightarrow n! \leq cn^{2,5}$$

Falso, porque

$$n=5, c=1$$

$$5! \leq 5^{2,5}$$

$$120 \leq 55,9$$

$$\therefore n^{2,5} \leq cn! \quad \forall n \geq 5$$

$$\rightarrow 2^n \leq cn!$$

Falso, porque

$$n=1, c=1$$

$$2^2 \leq 1!$$

$$4 \leq 1$$

$$\therefore n! \leq c2^{2^n} \quad \forall n \geq 1$$

Ordenado ascendentemente, sería:

$$1. \log(\sqrt{n!})$$

$$2. \log(n!)$$

$$3. n^2 \log(n)$$

$$4. n^{2,5}$$

$$5. n!$$

$$6. 2^{2^n}$$

2.2.12

a) Si $f \in O(n)$, entonces $f^2 \in O(n^2)$
 $f \in O(n)$

Se tienen valores $n_0 \in \mathbb{N}$ y $C \in \mathbb{R}_{>0}$ | $\forall n \geq n_0$

$\forall n \geq n_0$ se cumple que $f \leq C \cdot n$

$$f \leq Cn \quad n_0 = 1, C = 2$$

$$f \leq 2n; \quad \forall n \geq n_0, n \geq 1$$

\therefore Para $f^2 \in O(n^2)$

$$f^2 \leq Cn^2 \quad \text{con } n_0 = 1 \text{ y } C = 2$$

$$f^2 \leq 2n^2 \quad \forall n \mid n \geq 1$$

b) Si $f \in O(n)$, entonces $2^f \in O(2^n)$

$$f \in O(n)$$

Se tienen valores $n_0 \in \mathbb{N} \wedge C \in \mathbb{R}_{>0}$ | $\forall n \geq n_0$

Se cumple que $f \leq C \cdot n$

$$f \leq Cn \quad n_0 = 1, C = 2$$

$$f \leq 2n; \quad \forall n \geq n_0, n \geq 1$$

\therefore Para $2^f \in O(2^n)$

$$2^f \leq C \cdot 2^n, \quad n_0 = 1, C = 2$$

$$\log(2^f) \leq \log(2^n) \cdot C$$

$$f \log(2) \leq n \log(2) \cdot C$$

$$f \log(2) \leq 2n \log(2)$$

$$f \leq 2n$$

$$\therefore f \leq 2n \quad \forall n \geq n_0 \mid n \geq 1$$

Punto 3:

Ejercicio 4:

- g_1 está antes de g_5 , porque si se toman los logaritmos, se compararía $\sqrt{\log n}$ con $\log n + \log(\log n)$.
 $\log n + \log(\log n) \geq \log n$; cambiando de variable para verlo más sencillo sería $z = \log n$; $\sqrt{z} = z^{1/2}$ contra $z + \log z \geq z$.
- g_5 es menor que g_3 , ya que $(\log n)^3$ crece más rápido que $\log n$. Ambos son polinomios en $\log n$ pero $(\log n)^3$ tiene mayor grado.
- g_3 está antes que g_4 , porque si dividimos ambos entre n , compararíamos $(\log n)^3$ con $n^{1/3}$. Así pues, los logaritmos crecen más lento que los exponenciales.
- g_4 está primero que g_2 , gracias a que los polinomios crecen más lento que los exponenciales.
- g_2 está antes que g_7 , ya que, al tomar los logaritmos, se compararía n con n^2 y n^2 es el polinomio con mayor grado.
- g_7 está antes que g_6 ya que, si comparamos n^2 to 2^n , los polinomios crecen más lento que los exponenciales.

ordenados:

$$g_1) 2^{\sqrt{\log n}}$$

$$g_5) n^{\log n}$$

$$g_3) n(\log n)^3$$

$$g_4) n^{4/3}$$

$$g_2) 2^n$$

$$g_7) 2^{n^2}$$

$$g_6) 2^{2^n}$$

2.3.1 del Punto 2 de la Tarea

a) $\Omega(f) = \Omega(cf)$ - Tomado de las notas de clase

Si $g: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ es tal que $g \in \Omega(f)$, basta con demostrar $g \in \Omega(cf)$. Si $g \in \Omega(f)$, entonces hay $n_0 \in \mathbb{N}$ y $c_0 \in \mathbb{R}_{>0}$ | $g(n) \geq c_0 f$ para $n \geq n_0$. Tome $n_1 = n_0$ y $c_1 = \frac{c_0}{c}$, y note que para $n \geq n_1$ se tiene:

$$\begin{aligned} g(n) &\geq c_0 f(n) && (\text{por suposición}) \\ &= c_1 c f(n) && (\text{por definición de } c_1) \end{aligned}$$

Luego, $g \in \Omega(cf)$ con testigos n_1 y c_1 .

Si $g: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ es tal que $g \in \Omega(cf)$, basta con demostrar $g \in \Omega(f)$. Si $g \in \Omega(cf)$, entonces hay $n_0 \in \mathbb{N}$ y $c_0 \in \mathbb{R}_{>0}$ tales que $g(n) \geq c_0 cf$ para $n \geq n_0$. Tome $n_1 = n_0$ y $c_1 = c_0 c$, y note que para $n \geq n_1$ se tiene:

$$\begin{aligned} g(n) &\geq c_0 c f(n) \\ &= c_1 f(n) \end{aligned}$$

Luego, $g \in \Omega(f)$ con testigos n_1 y c_1 .

b) $\Theta(f) = \Theta(cf)$

Si $g: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ es tal que $g \in \Theta(f)$, basta con demostrar $g \in \Theta(cf)$. Si $g \in \Theta(f)$, entonces hay $n_0 \in \mathbb{N}$ y $c_0 \in \mathbb{R}_{>0}$ | $g(n) \leq c_0 f$ para $n \geq n_0$. Tome $n_1 = n_0$ y $c_1 = \frac{c_0}{c}$, y note que para $n \geq n_1$ se tiene:

$$g(n) \leq c_0 f(n) \\ = c_1 c_0 f(n)$$

Luego, $g \in \Theta(cf)$ con testigos n_1 y c_1

• Si: $g: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ es tal que $g \in \Theta(cf)$, basta con demostrar $g \in \Theta(f)$. Si $g \in \Theta(cf)$, entonces hay $n_0 \in \mathbb{N}$ y $c_0 \in \mathbb{R}_{>0}$ tales que $g(n) \leq c_0 cf$ para $n \geq n_0$. Tome $n_1 = n_0$ y $c_1 = c_0 c$, y note que para $n \geq n_1$ se tiene:

$$g(n) \leq c_0 c f(n) \\ = c_1 f(n)$$

Luego, $g \in \Theta(f)$ con testigos n_1 y c_1 .

Part 4.

Ejercicio 13: Inversions

Let $A[1 \dots n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an inversion of A .

We need to recursively divide the array into halves and count number of inversions in the sub-arrays. This will result in $\log n$ steps and $\Theta(n)$ operations in each step to count the inversions. All in all a $\Theta(n \log n)$ algorithm

```
def Inversions (A, p, r)
```

```
    if  $p \geq r$ 
```

```
        return 0
```

```
     $q = \lfloor (p+r) / 2 \rfloor$ 
```

```
    left = Inversions (A, p, q)
```

```
    right = Inversions (A, q+1, r)
```

```
    inversions = left + right + merge (A, p, q, r)
```

```
    return inversions
```

Modified merge-sort

Modified merge-sort

def Merge(A, p, q, r):

$n_1 = q - p + 1$

$n_2 = r - q$

let $L[1..n_1]$ and $R[1..n_2]$ be new arrays

for $i = 1$ to n_1

$L[i] = A[p+i-1]$

for $j = 1$ to n_2

$R[j] = A[q+1+j]$

$L[n_1+1] = \infty$

$R[n_2+1] = \infty$

$i = 1$

$j = 1$

inversions = 0

for $k = p$ to r

if $L[i] \leq R[j]$

$A[k] = L[i]$

$i = i + 1$

else:

inversions = inversions + $(n_1 - i + 1)$

$A[k] = R[j]$

$j = j + 1$

return inversions

Punto 5

Ejercicio 31: Fixed Point

Suppose we are given an array $A[1..n]$ of n distinct integers, which could be positive, negative, or zero, sorted in increasing order so that $A[1] < A[2] < \dots < A[n]$.

- a) Suppose we define a second array $B[1..n]$ by setting $B[i] = A[i] - i$ for all i . For every index i we have
- $$B[i] = A[i] - i \leq (A[i+1] - 1) - i = A[i+1] - (i+1) = B[i+1]$$

$A[i] = i$ if and only if $B[i] = 0$

def FindMatch(l, r):

if $l > r$:

return None

mid = $(l+r)/2$

If $A[mid] = mid$

$B[mid] = 0$

return mid

else if $A[mid] < mid$

$B[mid] < 0$

return FindMatch($mid+1, r$)

else

$B[mid] > 0$

return FindMatch($l, mid-1$)

b.

def FindMatchPos($A[1..n]$):

if $A[1] = 1$

return 1

else

return None

Again, the array $B[1..n]$ defined by setting $B[i] = A[i] - i$ is sorted in increasing order. It follows that if $A[1] > 1$ (that is, $B[1] > 0$), then $A[i] > i$ (that is, $B[i] > 0$) for every index i . $A[1]$ cannot be less than 1.