Tarea 1 ADA 2022-1

Jonathan Asprilla Saavedra – 8926036

Punto 1

Código de Honor

Como miembro de la comunidad académica de la Pontificia Universidad Javeriana Cali, los valores éticos y la integridad son tan importantes como la excelencia académica. En este curso se espera que los estudiantes se comporten ética y honestamente, con los más altos niveles de integridad escolar. En particular, se asume que cada estudiante adopta el siguiente *código de honor*:

Como miembro de la comunidad académica de la Pontificia Universidad Javeriana Cali me comprometo a seguir los más altos estándares de integridad académica.

Integridad académica se refiere a ser honesto, dar crédito a quien lo merece y respetar el trabajo de los demás. Por eso es importante evitar plagiar, engañar, 'hacer trampa', etc. En particular, el acto de entregar un programa de computador ajeno como propio constituye un acto de plagio; cambiar el nombre de las variables, agregar o eliminar comentarios y reorganizar comandos no cambia el hecho de que se está copiando el programa de alguien más. Para más detalles consultar el *Reglamento de Estudiantes*, Sección VI.

4 /

Jonathan Asprilla Saavedra - 8926036 Tarea 1

1. Clasificar por orden asintótico las siguientes funciones

2.2.1

→ $\log(\sqrt{n!}) \leq c\log(n!)$

Aplicando euler

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

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

Para todo $n \geq n_0$ $\quad n \geq 2$; $\sqrt{2!} < 2!$

$c = 1$

→ $\log(n!) \leq cn^2\log n$

Aplicando euler

$e^{\log(n!)} \leq \cancel{e^{cn^2\log n}}$

$n! \leq cn^3$

$n=1$, $c=1$

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

→ $n^{2,5} \leq n^2\log n \cdot c$

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 $n \geq 1$

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

Falso, porque

$n = 5\ ,\ c = 1$

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

$120 \leq 55,9$

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

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

Falso, porque

$n = 1,\ c = 1$

$2^2 \leq 1!$

$4 \leq 1$

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

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 $c \in \mathbb{R}_{>0} \mid \forall n \geq n_0$ se cumpla que $f \leq c \cdot n$

$f \leq cn \quad n_0 = 1, \ c = 2$

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

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

$f^2 \leq cn^2$ con $n_0 = 1$ y $c = 2$

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

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

$f \in O(n)$

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

Se cumpla que $f \leq c \cdot n$

$f \leq cn \quad n_0 = 1, \ c = 2$

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

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

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

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

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

$f \cancel{\log(2)} \leq 2n \cancel{\log(2)}$

$f \leq 2n$

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

Punto 3:

Ejercicio 4:

→ g_1 está antes de g_5, porque si se toman los logaritmos, se compararía $\sqrt{\log n}$ con $\log n + \log(\log n) \geq \log n$; 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 $g: \mathbb{N} \to \mathbb{R}_{\geq 0}$ 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:

$g(n) \geq C_0 f(n)$ (por suposición)
$= C_1 c f(n)$ (por definición de C_1)

Luego, $g \in \Omega(cf)$ con testigos n_1 y C_1

Si $g: \mathbb{N} \to \mathbb{R}_{\geq 0}$ 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 $g(n) \geq C_0 c f$ 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:

$g(n) \geq C_0 c f(n)$
$= C_1 f(n)$

Luego, $g \in \Omega(f)$ con testigos n_1 y C_1.

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

Si $g: \mathbb{N} \to \mathbb{R}_{\geq 0}$ 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:

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

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

- Si $g: \mathbb{N} \to \mathbb{R}_{\geq 0}$ 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 $g(n) \leq c_0 c f$ 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.

Punto 4.

Ejercicio 13: Inversions

Let $A[1 \cdots n]$ 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 ≥ r
        return 0
    q = ⌊(p+r)/2⌋
    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):
    n1 = q - p + 1
    n2 = r - q
    let L[1..n1] and R[1..n2] be new arrays
    for i = 1 to n1
        L[i] = A[p+i-1]
    for j = 1 to n2
        R[i] = A[q+j]
    L[n1+1] = ∞
    R[n2+1] = ∞
    i = 1
    j = 1
    inversions = 0
    for k = p to r
        if L[i] ≤ R[j]
            A[k] = L[i]
            i = i+1
        else:
            inversions = inversions + (n1 - 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, wich could be positive, negative, or zero, sorted in increasing order so that $A[1] < A[2] < \cdots < A[n]$.

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

$$B[i] = A[i] - i \leq (A[i+j]-1) - i = A[i+1] - (i+1) = B[i+1]$$

$A[i] = i$ 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.