

Tarea 1

Análisis y Diseño de Algoritmos

Código de honor

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

Integridad académica: se refiere a ser honesto, dar crédito a quien se lo merece y a 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 filos o eliminar comentarios y reorganizar comandos no cambia el hecho de que se está copiando el programa de alguien más.

Punto 1)

$$f_1(n) = n^{2.5}, f_2(n) = \sqrt{2n}, f_3(n) = n+10, f_4(n) = 10^n, f_5(n) = 100^n, f_6(n) = n^2 \log n$$

Notación O grande:

Sea $f: \mathbb{N} \rightarrow \mathbb{R}_{>0}$

$$O(f_i) = \{g: \mathbb{N} \rightarrow \mathbb{R}_{>0} \mid \exists_{n_0} \in \mathbb{N} \cdot \exists_{c} \in \mathbb{R}_{>0} \cdot \forall_{n \in \mathbb{N}} \cdot n \geq n_0 \rightarrow g(n) \leq c \cdot f(n)\}$$

1) Teorema $f_4 \in O(f_5)$

Demostración: Se procede de manera directa partiendo de la definición de O,

De acuerdo con esto, es necesario encontrar ctes c y n_0 tales que

$$\forall_{n \geq n_0} \rightarrow f_4 \leq c \cdot f_5$$

$$10^n \leq c \cdot 100^n$$

$$10^n \leq c \cdot 10^{2n}$$

$$c \geq \frac{10^n}{10^{2n}}$$

$$c \geq \frac{1}{10^n}$$

Note que $1/10^n$ alcanza su valor más grande en $n=0$ ($1/1$), por lo tanto

$$\forall_{n \geq 0} \wedge c=1 \text{ se cumple que } f_4 \in O(f_5)$$

2) Teorema: $n^2 \log n \in O(n^{2.5})$, por lo tanto

Demostración: Se procede de manera directa partiendo de la definición de O,

De acuerdo con esto, es necesario encontrar ctes c y n_0 tales que

$$\forall_{n \geq n_0} \rightarrow f_6 \leq c \cdot f_2$$

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

$$\cancel{n^2 \log n} \leq c \cdot \cancel{n^2} \cdot n^{0.5}$$

$$\log n \leq c \sqrt{n}$$

$$c \geq \frac{\log n}{\sqrt{n}}$$

Note que no se puede simplificar más la expresión. No obstante si la expresión del denominador crece más rápido que la del numerador, para n 's grandes la expresión tenderá a 0, Por lo tanto

Si analizamos la expresión por medio de límites tenemos que

$$\lim_{n \rightarrow \infty} \frac{\log n}{\sqrt{n}} = \frac{\infty}{\infty}$$

Como tenemos un caso de $\frac{\infty}{\infty}$ podemos usar L'Hospital

$$\lim_{n \rightarrow \infty} \frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n \rightarrow \infty} \frac{2\sqrt{n}}{n} = \lim_{n \rightarrow \infty} 2n^{0.5} \cdot n^{-1} = \lim_{n \rightarrow \infty} \frac{2}{\sqrt{n}} = 0$$

Es posible notar que el valor más grande de $2/\sqrt{n}$ es con $n=1$, por lo tanto para $n=1 \rightarrow c=0$, es decir existe un n_0 y un c que cumpli dicha condición, además el límite indica que $\log n \in O(\sqrt{n})$, es decir que $\log n$ crece más lento que \sqrt{n} , en conclusión $f_3 \in O(f_6)$

3) Teorema: $f_3 \in O(f_6) = n+10 \in O(n^2 \log n)$,

Demarcación: Se procede de manera directa partiendo de la definición de O ,

De acuerdo con esto, es necesario encontrar cts c y n_0 tales que

$$\forall n \geq n_0 \rightarrow f_3 \leq c \cdot f_6$$

$$n+10 \leq c \cdot n^2 \log n$$

$$\frac{1}{n} + \frac{10}{n^2} \leq c \log n$$

$$g(n) = \frac{1}{n \log n} + \frac{10}{n^2 \log n} \leq c$$

Es posible notar que el valor más grande que puede tomar esta expresión es cuando $n=2$, ya que $\forall n \geq 2$ la función g decrece, por lo tanto

$n_0 = 2$ y $c \approx 9,96$. En conclusión $f_3 \in O(f_6)$

Nota: Se asume $n \geq 2$ ya que con $n < 2$ se genera una indeterminación

4. Teorema: $f_2 \in O(f_3)$

Demonstración: Se procede de manera directa partiendo de la definición de O ,

De acuerdo con esto, es necesario encontrar ctes c y n_0 tales que

$$\forall n \geq n_0 \rightarrow f_2 \leq c \cdot f_3$$

$$\sqrt{2n} \leq c(n+10)$$

$$c \geq \frac{\sqrt{2n}}{n+10}$$

Note que no se puede simplificar más la expresión. No obstante si la expresión del denominador crece más rápido que la del numerador, para n 's grandes la expresión tenderá a 0, Por lo tanto

Para valores grandes de n entonces

$$\lim_{n \rightarrow \infty} \frac{\sqrt{2n}}{n+10} = \frac{\infty}{\infty} \rightarrow \text{Aplicando L'Hopital} \rightarrow \lim_{n \rightarrow \infty} \frac{\frac{1}{\sqrt{2n}}}{1} = 0$$

Note que como el límite tiende a 0 entonces se cumple la condición,

Por lo que el valor más grande que puede tomar $\sqrt{2n} / n+10$ es con $n=0$

($\sqrt{2 \cdot 0} / 0+10 = 0 = c$) por lo tanto $\forall n \geq 0 \quad n \leq 0$ se cumple que $f_2 \in O(f_3)$

5. Teorema s $f_1 \in O(f_4)$

Demonstración: Se procede de manera directa partiendo de la definición de O ,

De acuerdo con esto, es necesario encontrar ctes c y n_0 tales que

$$\forall n \geq n_0 \rightarrow f_1 \leq c \cdot f_4$$

$$n^{2.5} \leq c \cdot 10^n$$

$$c \geq \frac{n^{2.5}}{10^n}$$

Note que no se puede simplificar más la expresión. No obstante si la expresión del denominador crece más rápido que la del numerador, para n 's grandes la expresión tenderá a 0, Por lo tanto

Analizando n cuando tiende a infinito entonces

$$\lim_{n \rightarrow \infty} \frac{n^{2.5}}{10^n} = \frac{\infty}{\infty}$$

Aplicamos L'Hopital

$$\lim_{n \rightarrow \infty} \frac{2.5 n^{1.5}}{10^n \ln 10} = \frac{\infty}{\infty}$$

Derivaremos de nuevo

$$\lim_{n \rightarrow \infty} \frac{3.75 n^{0.5}}{10^n (\ln 10)^2} = \frac{\infty}{\infty}$$

Una vez más

$$\lim_{n \rightarrow \infty} \frac{1.875}{10^n (\ln 10)^3 \cdot n^{0.5}} = 0$$

Nota que para valores grandes de n se cumple la condición,

Por lo que el valor mayor de $n^{2.5}/10^n$ es con $n=0$,

es decir que $C = 0^{2.5}/10^0 = 1$, por lo que $\forall_{n \geq 0} \quad n < 1$

Se cumple que $f_1 \in O(f_4)$

Para finalizar tenemos que

1. $f_4 \in O(f_5)$

2. $f_6 \in O(f_2)$

3. $f_3 \in O(f_6)$

4. $f_2 \in O(f_3)$

5. $f_1 \in O(f_4)$

La propiedad transitiva indica que

$$h \in O(g) \wedge g \in O(f) \rightarrow h \in O(f)$$

Por lo tanto se cumple el siguiente orden

$$f_2 \in O(f_3) \wedge f_3 \in O(f_6) \wedge f_6 \in O(f_1) \wedge f_1 \in O(f_4) \wedge f_4 \in O(f_5)$$

entonces al orden queda así

$$f_2, f_3, f_6, f_1, f_4, f_5$$

Punto 2)

-
- (a) Given an array $A[1..n]$ of characters, compute the number of partitions of A into words. For example, given the string **ARTISTOIL**, your algorithm should return 2, for the partitions **ARTIST·OIL** and **ART·IS·TOIL**.

- Especificación

Entrada: un arreglo $A[0..n]$ de letras

Salida: Número de posibles particiones de A , donde las particiones sean palabras

- Planteamiento función objetivo

Sea $0 \leq l \leq N$

$\phi(l)$: calcula cuantas posibles maneras se puede particionar A

- Reformulación Especificación

Salida: $\phi(0)$

- Planteamiento recursivo

$$\phi(l) = \begin{cases} 1 & \text{if } l > N \\ \sum_{i=l}^N \phi(i+1) & \text{if } \text{isWord}(A, l, i) \end{cases}$$

Nota:

isWord(A, l, r) es una función que recibe un arreglo A y retorna un booleano diciendo si A entre los indices l, r es una palabra, suponemos que su complejidad $\rightarrow O(n)$

Complejidad temporal:

$$1. \quad T(n) = T(n-1) + T(n-2) + \dots + T(0) + O(n)$$

$$2. \quad T(n-1) = T(n-2) + \dots + T(0) + O(n)$$

Restamos 1 y 2:

$$T(n) - T(n-1) = T(n-1)$$

$$T(n) = 2T(n-1)$$

Expandimos $T(n)$

$$T(n) = 2T(n-1)$$

$$T(n-1) = 2T(n-2)$$

$$T(n-2) = 2T(n-3)$$

Si hacemos esto i veces hasta que $i=n$, entonces

$$T(n) = 2^i T(n-i), \quad 1 \leq i \leq n$$

Es decir que en $i=n$

$$T(n) = 2^n T(0) = O(2^n)$$

- (b) Given two arrays $A[1..n]$ and $B[1..n]$ of characters, decide whether A and B can be partitioned into words at the same indices. For example, the strings **BOTHEARTHANDSATURNSPIN** and **PINSTARTRAPSANDRAGSLAP** can be partitioned into words at the same indices as follows:

BOT · HEART · HAND · SAT · URNS · PIN
 PIN · START · RAPS · AND · RAGS · LAP

- Especificación

Entrada: Dos arreglos, $A[0..n]$ y $B[0..n]$ compuestos de caracteres

Salida: Ver si es posible dividir A y B en palabras de igual longitud

- Planteamiento función objetivo

Sea $0 \leq l \leq N$

$\phi(l)$: retorna bool, en donde dice si es posible dividir A y B
 se pueden dividir en indices de igual longitud

- Reformulación Especificación

Salida: $\phi(0)$

- Planteamiento recursivo

$$\phi(l) \begin{cases} \text{True} & \text{if } l > N \\ \bigvee_{i=l}^N (\phi(i+1) \wedge \text{isWord}(A, l, i) \wedge \text{isWord}(B, l, i)) & \text{otherwise} \end{cases}$$

Complejidad temporal:

$$1. \quad T(n) = T(n-1) + T(n-2) + \dots + T(0) + O(n)$$

$$2. \quad T(n-1) = T(n-2) + \dots + T(0) + O(n)$$

Restamos 1 y 2:

$$T(n) - T(n-1) = T(n-1)$$

$$T(n) = 2T(n-1)$$

Expandimos $T(n)$

$$T(n) = 2T(n-1)$$

$$T(n-1) = 2T(n-2)$$

$$T(n-2) = 2T(n-3)$$

Si hacemos esto i veces hasta que $i=n$, entonces

$$T(n) = 2^i T(n-i), \quad 1 \leq i \leq n$$

Es decir que en $i=n$

$$T(n) = 2^n T(0) = O(2^n)$$

- (c) Given two arrays $A[1..n]$ and $B[1..n]$ of characters, compute the number of different ways that A and B can be partitioned into words at the same indices.

- Especificación

Entrada: Dos arreglos, $A[0..n]$ y $B[0..n]$ compuestos de caracteres

Salida: Número de particiones de A y B de mismo tamaño

- Planteamiento función objetivo

Sea $0 \leq l \leq N$

$\phi(l)$: Determina el número de particiones de A y B con el mismo tamaño

- Reformulación Especificación

Salida: $\phi(0)$

- Planteamiento recursivo

$$\phi(l) = \begin{cases} 1 & \text{if } l > N \\ \sum_{i=l}^N \phi(i+1) & \text{if } \text{isWord}(A, l, i) \wedge \text{isWord}(B, l, i) \end{cases}$$

Complejidad temporal:

$$1. \quad T(n) = T(n-1) + T(n-2) + \dots + T(0) + O(n)$$

$$2. \quad T(n-1) = T(n-2) + \dots + T(0) + O(n)$$

Rostamos 1 y 2:

$$T(n) - T(n-1) = T(n-1)$$

$$T(n) = 2T(n-1)$$

Expandimos $T(n)$

$$T(n) = 2T(n-1)$$

$$T(n-1) = 2T(n-2)$$

$$T(n-2) = 2T(n-3)$$

Si hacemos esto i veces hasta que $i=n$, entonces

$$T(n) = 2^i T(n-i), \quad 1 \leq i \leq n$$

Es decir que en $i=n$

$$T(n) = 2^n T(0) = O(2^n)$$