
TALLER 1

Cormen

Laura Valentina Castaño Morales- 1010229518

Ing. Germán Hernández

(Docente)

UNIVERSIDAD NACIONAL DE COLOMBIA

SEDE BOGOTÁ

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE SISTEMAS E INDUSTRIAL

ALGORITMOS (2016696 - 1)

BOGOTÁ D.C

2018

Desarrollo

1. Ejercicios del libro [Cormen09]

- a. 3.1-2 Show that for any real constants a and b , where $b > 0$, $(n + a)^b = \theta(n^b)$

Respuesta:

Podemos demostrar que existen constantes c_1 , c_2 y n_0 tal que se cumpla la siguiente desigualdad:

Para todo

$$n \geq n_0$$

$$0 \leq c_1 n^b \leq (n + a)^b \leq c_2 n^b$$

$$0 \leq c_1 n \leq (n + a) \leq c_2 n$$

Podemos observar que $(n + a) \leq 2n$ siempre y cuando $|a| \leq n$ y también $(n + 2) \geq \frac{n}{2}$ si $|a| \leq \frac{n}{2}$

Por lo tanto podemos verificar que $(n + a)^b = \theta(n^b)$ debido a que existen $c_1 = \frac{1}{2^b}$ y $c_2 = 2^b$ constantes tal que satisfacen la desigualdad si $n_0 \geq 2|a|$

- b. 3.1-7 Prove that $o(g(n)) \cap \omega(g(n))$ is the empty set.

Respuesta:

Podemos definir formalmente $o(g(n))$ y $\omega(g(n))$ de la siguiente manera:

1. Por definición $o(g(n)) = \{f(n): \forall c, n_0 \text{ tal que } 0 \leq f(n) \leq cg(n), \forall n > n_0\}$
2. Por definición $\omega(g(n)) = \{f(n): \forall c, n_0 \text{ tal que } 0 \leq cg(n) \leq f(n), \forall n > n_0\}$

Podemos entonces definir $o(g(n)) \cap \omega(g(n))$ como:

$$o(g(n)) \cap \omega(g(n)) = \{f(n): \forall c, n_0 \leq cg(n) \leq f(n) \leq cg(n), \forall n > n_0\}$$

Esta desigualdad no se cumple para ninguna función $f(n)$ por lo tanto la intersección de $o(g(n))$ y $\omega(g(n))$ debe ser el conjunto vacío.

- c. 3.3 a) Rank the following functions by order of growth; that is, find an arrangement g_1, g_2, \dots, g_{30} of the functions satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3), \dots, g_{29} =$

$\Omega(g)$. Partition your list into equivalence classes such that functions $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$.

$$\begin{array}{cccccc}
 \lg(\lg^* n) & 2^{\lg^* n} & (\sqrt{2})^{\lg n} & n^2 & n! & (\lg n)! \\
 \left(\frac{3}{2}\right)^n & n^3 & \lg^2 n & \lg(n!) & 2^{2^n} & n^{1/\lg n} \\
 \ln \ln n & \lg^* n & n \cdot 2^n & n^{\lg \lg n} & \ln n & 1 \\
 2^{\lg n} & (\lg n)^{\lg n} & e^n & 4^{\lg n} & (n+1)! & \sqrt{\lg n} \\
 \lg^*(\lg n) & 2^{\sqrt{2 \lg n}} & n & 2^n & n \lg n & 2^{2^{n+1}}
 \end{array}$$

Respuesta:

1	$2^{2^{n+1}}$		13	$n \lg(n)$	$\lg(n!)$
2	2^{2^n}		14	$2^{\lg(n)}$	n
3	$(n+1)!$		15	$(\sqrt{2})^{\lg(n)}$	
4	$n!$		16	$2^{\sqrt{2 \lg(n)}}$	
5	$n 2^n$		17	$\lg^2(n)$	
6	e^n		18	$\ln(n)$	
7	2^n		19	$\sqrt{\lg(n)}$	
8	$\left(\frac{3}{2}\right)^n$		20	$\ln(\ln(n))$	
9	$(\lg(n))!$		21	$2^{\lg^*(n)}$	
10	$n^{\lg(\lg(n))}$	$\lg(n)^{\lg(n)}$	22	$\lg^*(n)$	$\lg^*(\lg(n))$
11	n^3		23	$\lg(\lg^*(n))$	
12	n^2	$4^{\lg(n)}$	24	1	$n^{1/\lg(n)}$

b) Give an example of a single nonnegative function $f(n)$ such that for all functions $g_i(n)$ in part (a), $f(n)$ is neither $O(g_i(n))$ nor $\Omega(g_i(n))$.

Respuesta:

Si definimos la función

$$f(n) = \begin{cases} g_1(n)! & n \bmod 2 = 0 \\ \frac{1}{n} & n \bmod 2 = 1 \end{cases} :$$

Teniendo en cuenta que $f(n)$ es asintóticamente positiva.

Para n par tenemos:

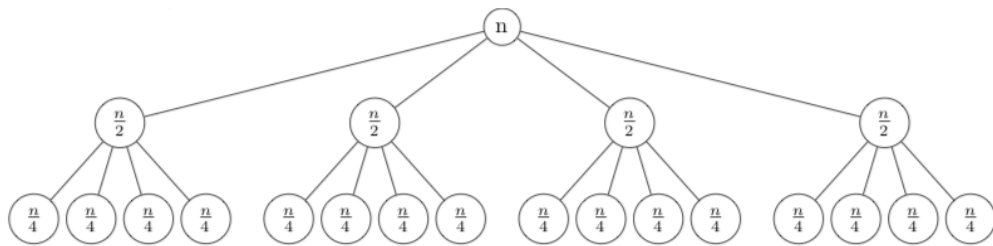
$$\begin{aligned}
\lim_{n \rightarrow \infty} \frac{f(2n)}{g_i(2n)} &\geq \lim_{n \rightarrow \infty} \frac{f(2n)}{g_1(2n)} \\
&= \lim_{n \rightarrow \infty} (g_1(2n) - 1)! \\
&= \infty
\end{aligned}$$

Para n impar tenemos:

$$\begin{aligned}
\lim_{n \rightarrow \infty} \frac{f(2n+1)}{g_i(2n+1)} &\leq \lim_{n \rightarrow \infty} \frac{f(2n+1)}{1} \\
&= \lim_{n \rightarrow \infty} \frac{1}{2n+1} \\
&= 0
\end{aligned}$$

- d. **4.4-7** Draw the recursion tree for $T(n) = 4T(\lfloor n/2 \rfloor) + cn$, where c is a constant, and provide a tight asymptotic bound on its solution. Verify your bound by the substitution method.

Respuesta:



Se incrementa el número de subproblemas por cada recursión en 4. Se disminuye el tamaño del subproblema en 2 más 2 adicionales. De esa manera, en cada nivel del árbol hay 4^i nodos cada uno de coste $c(n/2^i + 2)$ a una profundidad i , que sigue la secuencia $i = 0, 1, 2, \dots, \lg n$.

El costo total del árbol es:

$$\begin{aligned}
T(n) &= \sum_{i=0}^{\lg n} 4^i \cdot c \left(\frac{n}{2^i} + 2 \right) \\
&= \sum_{i=0}^{\lg n} 4^i \cdot c \frac{n}{2^i} + \sum_{i=0}^{\lg n} 4^i \cdot c \cdot 2 \\
&= cn \sum_{i=0}^{\lg n} \frac{4^i}{2^i} + 2c \sum_{i=0}^{\lg n} 4^i \\
&= cn \sum_{i=0}^{\lg n} 2^i + 2c \sum_{i=0}^{\lg n} 4^i \\
&= cn \frac{2^{\lg n + 1} - 1}{2 - 1} + 2c \frac{4^{\lg n + 1} - 1}{4 - 1} \\
&= cn(2^{\lg n + 1} - 1) + \frac{2c}{3}(4^{\lg n + 1} - 1) \\
&= cn(2 \cdot 2^{\lg n} - 1) + \frac{2c}{3}(4 \cdot 4^{\lg n} - 1) \\
&= cn(2 \cdot n - 1) + \frac{2c}{3}(4 \cdot n^2 - 1) \\
&= 2cn^2 - 2n + \frac{8cn^2}{3} - \frac{2c}{3} \\
&= O(n^2)
\end{aligned}$$

Se verifica el resultado del problema por el método de sustitución.

$$\begin{aligned}
T(n) &= 4T(n/2 + 2) + n \\
&\leq 4d((n/2 + 2)^2 - b(n/2 + 2)) + n \\
&= 4d(n^2/4 + 2n + 4 - bn/2 - 2b) + n \\
&= dn^2 + 8dn + 16d - 2dbn - 8db + n \\
&= dn^2 - dbn + 8dn + 16d - dbn - 8db + n \\
&= d(n^2 - bn) - (db - 8d - 1)n - (b - 2)8d \\
&\leq d(n^2 - bn)
\end{aligned}$$

Esto se mantiene para $db - 1 - 8d \geq 0$.

e. Use el método maestro para dar cotas ajustadas para las recurrencias:

$$\begin{aligned}
T(n) &= 8T(n/2) + n \\
T(n) &= 8T(n/2) + n^3 \\
T(n) &= 8T(n/2) + n^5
\end{aligned}$$

En estos problemas se puede ver que $a=8, b=2$ y $f(n)=n, n^2, n^3$ respectivamente. Se compara $f(n)$ con $n^{\log_b a} = n^{\log_2 8}$. Las 3 recurrencias satisfacen los 3 diferentes casos del teorema Maestro.

Por lo tanto:

$$T(n) = 8T(n/2) + n$$

$n^{\log_2 8} = n^3 = \Theta(n^3)$, la función $f(n)$ tiene una cota $\Theta(n^{\log_b(a)-\epsilon})$ con $\epsilon = 2$ por lo tanto $T(n) = \Theta(n^3)$

$$T(n) = 8T(n/2) + n^3$$

$n^{\log_2 8} = n^3 = \Theta(n^3)$, la función $f(n)$ tiene una cota $\Theta(n^{\log_b a})$ por lo tanto $T(n) = \Theta((n^3)\log(n))$

$$T(n) = 8T(n/2) + n^5$$

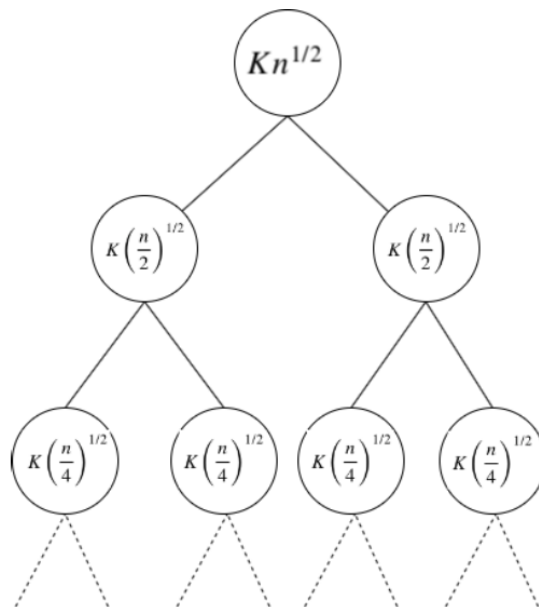
$n^{\log_2 8} = n^3 = \Theta(n^3)$, la función $f(n)$ tiene una cota $\Theta(n^{\log_b(a)+\epsilon})$ con $\epsilon = 2$ por lo tanto $T(n) = \Theta(n^5)$

2. Dado el pseudocódigo:

a. Plantee una ecuación de recurrencia para $T(n)$, el tiempo que toma la función misterio(n).

$$T(n) = 2T(n/2) + \Theta(n^{1/2})$$

b. Dibuje el árbol de recursión y calcule:



- i. La altura del mismo: $\log_2(n)$
- ii. El número de nodos por cada nivel: $2^{\log_2(n)}$
- iii. La suma de los nodos de cada nivel: $2^i k(\frac{n}{2^i})^{1/2}$
- iv. La suma total: $\Theta(n)$

c. Determine el comportamiento asintótico de $T(n)$ justificándolo de manera detallada.

Por el método maestro podemos estudiar el comportamiento asintótico de la función de recurrencia $T(n)$:

Donde es de la forma $O(n)$ con $\epsilon = 0,5$

por lo tanto $T(n) = \Theta(n)$

3. **22.3-1** (pág 610) Realizar un cuadro 3×3 con filas y columnas WHITE, GRAY, BLACK. En cada celda (i,j) , indique si en algún momento durante la búsqueda en la profundidad de un grafo dirigido, puede haber una arista o un vértice de color i a un vértice de color j . Por cada arista posible indique que tipo puede ser. Haga otro cuadro pero utilizando un grado no dirigido

Para el grafo dirigido.

i/j	White	Gray	Black
White	Tree, back, forward and cross.	Back and cross.	Cross.
Gray	Tree and forward.	Tree, back and forward.	Tree, forward and cross.
Black		Back.	Tree, back, forward and cross

Exercise 22.3-1

For directed graphs:

<i>from\to</i>	<i>BLACK</i>	<i>GRAY</i>	<i>WHITE</i>
<i>BLACK</i>	<i>Allkinds</i>	<i>Back, Cross</i>	<i>Back, Cross</i>
<i>GRAY</i>	<i>Tree, Forward, Cross</i>	<i>Tree, Forward, Back</i>	<i>Back, Cross</i>
<i>WHITE</i>	<i>Cross, Tree, Forward</i>	<i>Cross, Back</i>	<i>allkinds</i>

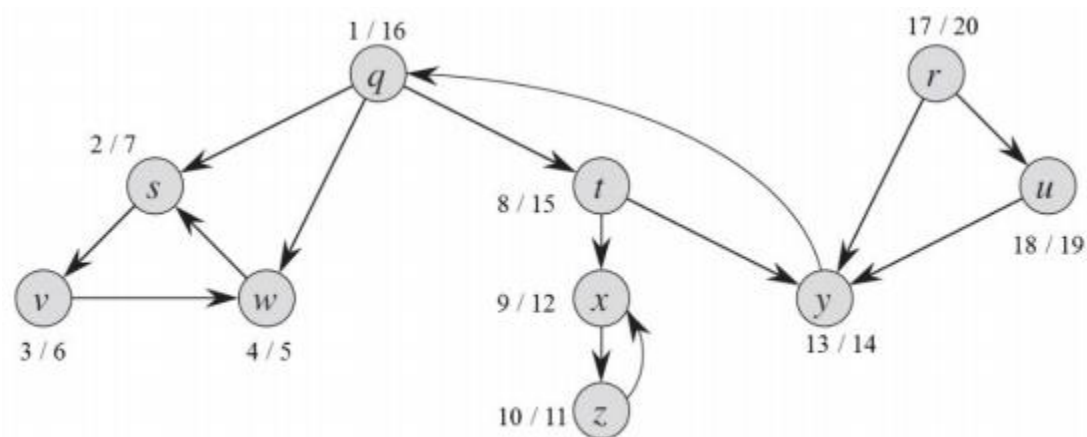
For undirected graphs, note that the lower diagonal is defined by the upper diagonal:

Para el grafo no dirigido.

i/j	White	Gray	Black
White	Tree and back.	Tree and back.	
Gray	Tree and back.	Tree and back.	Tree and back.
Black		Tree and back.	Tree and back.

<i>from\to</i>	<i>BLACK</i>	<i>GRAY</i>	<i>WHITE</i>
<i>BLACK</i>	<i>Allkinds</i>	<i>Allkinds</i>	<i>Allkinds</i>
<i>GRAY</i>	—	<i>Tree, Forward, Back</i>	<i>Allkinds</i>
<i>WHITE</i>	—	—	<i>Allkinds</i>

4. **22.3-2** (Pág 611) Muestre cómo DFS funciona en el grafo de la figura 22.6. Asuma que el ciclo for de la línea 5-7 del DFS considera los vértices en orden alfabético, y asuma que cada lista de adyacencia está ordenada alfabéticamente. Muestre el descubrimiento y los tiempos de cada vértice y muestre la clasificación de cada arista.



- Forward edges: (q,w).
 - Back edges: (z,x), (w,s), (y,q).
 - Tree edges: (q,s), (s,v), (v,w), (q,t), (t,x), (x,z), (t,y), (r,u).
 - Cross edges: (u,y), (r,y)
5. **22.4-2** Dé un algoritmo de tiempo lineal que tome como input un grafo dirigido acíclico $G = (V, E)$ y dos vértices s y t , y retorne el número de caminos simples de s a t en G .

Añada un campo a la representación de vértices para mantener un número entero. Inicialmente, establece la cuenta del vértice t en 1 y la de otros vértices en 0. Comienza ejecutando DFS con s como vértice de inicio. Cuando se descubre t , debe marcarse inmediatamente como terminado (NEGRO), sin ningún otro tratamiento a partir de él. Posteriormente, cada vez que DFS finaliza un vértice v , establece el conteo de v como la suma de los conteos de todos los vértices adyacentes a v . Cuando DFS finalice el vértice s , deténgase y devuelva el conteo calculado para s .

```

if u == v then
    Return 1
else if u.paths != NIL then
    Return u.paths

```



```
else
    for each  $w \in \text{Adj}[u]$  do
         $u.\text{paths} = u.\text{paths} + \text{SIMPLE-PATHS}(w, v)$ 
    end for
    Return  $u.\text{paths}$ 
end if
```