Universidad Nacional de Colombia

Algoritmos

Taller 01

Autor:
Edgar David Garay Forero

Profesor:
German Hernandez

Miercoles, 06 de Noviembre de 2018

1. Desarrolle los siguientes ejercicios del libro:

   a) Demuestre que para cualquier constante real a y b, donde $b > 0$, se cumple que: $(n+a)^b = \Theta(n^b)$

   Para demostrar que $(n+a)^b = \Theta(n^b)$ necesitamos encontrar las constantes c_1, c_2 y $n_0 > 0$ tal que $0 < c_1 n^b \leq (a+n)^b \leq c_2 n^b$ para todo $n > n_0$

$$\begin{aligned} n + a &\leq n + |a| \\ &\leq 2n \qquad \text{Cuando } |a| \leq n \\ n + a &\geq n - |a| \\ &\geq \tfrac{1}{2}n \qquad \text{Cuando } |a| \leq \tfrac{1}{2}n \end{aligned}$$

   Entonces, cuando $n \geq 2a$

$$0 \leq \frac{1}{2}n \leq n + a \leq 2n$$

   Ya que $b > 0$, la desigualdad se mantiene cuando todas sus partes se elevan al cuadrado, quedando:

$$0 \leq \left(\frac{1}{2}n\right)^b \leq (n+a)^b \leq (2n)^b$$

$$0 \leq \left(\frac{1}{2}\right)^b n^b \leq (n+a)^b \leq 2^b n^b$$

   Lo que nos permite concluir que $c_1 = \left(\frac{1}{2}\right)^b$, $c_2 = 2^b$ y $n_0 = 2|a|$, y finalmente satisface la definición de $\Theta(f(n))$.

   b) Demuestre que $o(g(n)) \cap \omega(g(n))$ es el conjunto vacío

   Suponiendo que

$$f(n) \in \theta(g(n)) \cap \omega(g(n))$$

   Entonces

$$f(n) = \omega(g(n))$$

   Si y solo si

$$g(n) = o(f(n)) \text{ y } f(n) = o(g(n))$$

   Y por propiedad de transitividad,

$$f(n) = o(f(n))$$

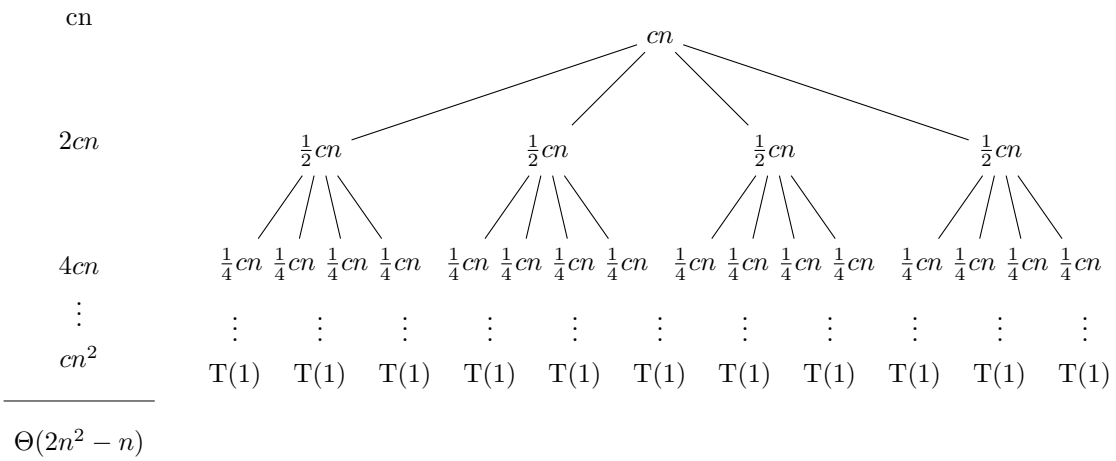
   Para todas las constantes $c > 0$ tenemos que $f(n) < cf(n)$ De esta manera si tomamos un $c < 1$ obtenemos una contradicción de la no negatividad asintotica de f(n)

   c) ordene las siguientes funciones por orden de crecimiento

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(\frac{3}{2})^n$	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	$n \lg n$	$2^{2^{n+1}}$

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

d) Dibuje el árbol de recursion para T(n)=4T [(n/2)] + cn donde cn es una constante



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

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

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

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

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

a = 8

b = 2

c = 1

f(n) = n

log_ba = $log_2 8 = 3$

como 3 >1 por teorema maestro tenemos que T(n) = $O(n^3)$

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

a = 8

b = 2

c = 3

f(n) = n

log_ba = $log_2 8 = 3$

como log_ba = 3 = c por teorema maestro tenemos que T(n) = $\theta((n^3)logn)$

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

a = 8

b = 2

c = 3

f(n) = n

log_ba = $log_2 8 = 3$

como log_ba = 3

c = 5 >log_ba = 3

por teorema maestro tenemos que T(n) = $\theta(n^5)$

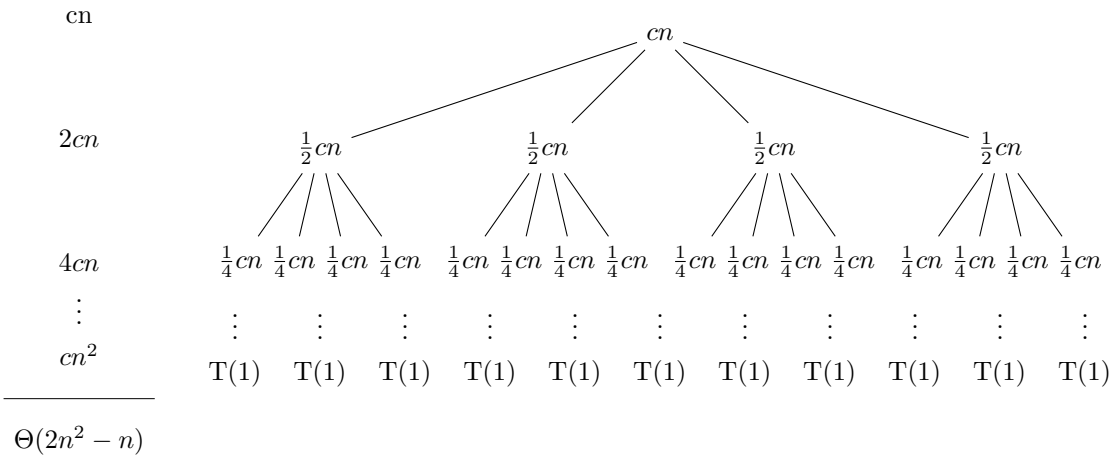
2. Dado el siguiente pseudocodigo

```
def misterio(n):
    if n <= 1:
        return 1
    else:
        r = misterio(n / 2)
        i = 1
        while n <= i*i:
            i = i + 1
            r = r + misterio(n / 2)
        return r
```

a) plantee una ecuación de recurrencia para T(n) el tiempo q toma la función misterio (n)

$T(n) = T(n/2) + \sqrt{n} \cdot T(n/2) + c$

b) Dibuje el árbol de recursión



c) Calcule:

1) La altura del árbol $log_2 n$

2) El número de nodos por cada nivel $\sqrt{1/2 * n}$

3) La suma de los nodos de cada nivel $\sqrt{1/2 * n}$

4) La suma total

3. Haga un cuadro de 3 por 3 con etiquetas de filas y columnas BLANCO, GRIS y NEGRO. En cada celda (i, j) e indique si, en cualquier punto durante una búsqueda en profundidad de un gráfico dirigido, puede haber un borde desde un vértice de color i hasta un vértice de color j. Para cada borde posible, indique qué tipos de borde puede ser. Haga una segunda tabla de este tipo para la búsqueda en profundidad de un gráfico no dirigido.
   Tree edge: T
   Back edge: B
   Forward edge: F
   Cross edge: C

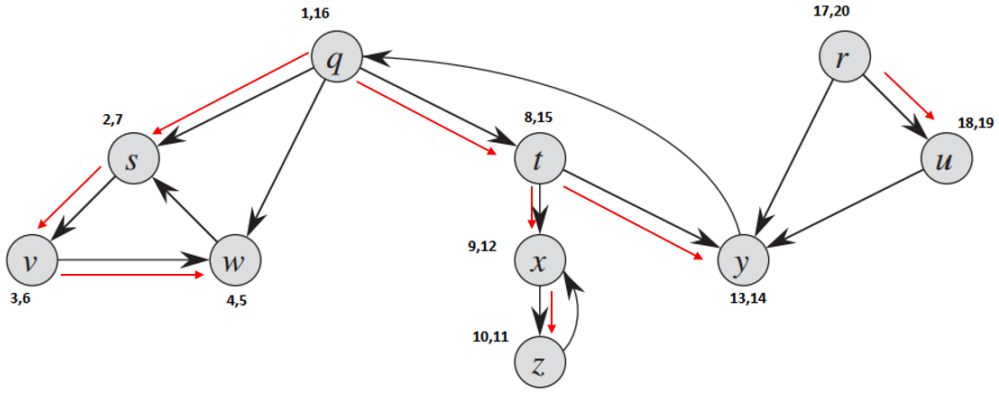
   Dirigido:

(i,j)	White	Gray	Black
White	TBFC	BC	C
Gray	TF	TFB	TFC
Black		B	TFBC

   No Dirigido:

(i,j)	White	Gray	Black
White	TB	TB	
Gray	TB	TB	TB
Black		TB	TB

4. Muestre cómo funciona la búsqueda de profundidad en el gráfico de la figura. Supongamos que el bucle for de las líneas 5-7 del procedimiento DFS considera los vértices en orden alfabético y supone que cada lista de adyacencia se ordena alfabéticamente. Muestre los tiempos de descubrimiento y finalización para cada vértice y muestre la clasificación de cada borde.



Tree edges: (q, s), (s, v), (v, w), (q, t), (t, x), (x, z), (t, y), (r, u)
Back edges: (w, s), (z, x), (y, q)
Forward edges: (q, w)
Cross edges: (r, y), (u, y)

5. Proporcione un algoritmo de tiempo lineal que tome como entrada un gráfico acíclico dirigido G = (V, E) y dos vértices s y t, y devuelve el número de caminos de s a t en G. Por ejemplo, en el gráfico acíclico dirigido de Figura, hay exactamente cuatro caminos desde el vértice p al vértice v: pov, por yv, posr yv, y psr yv. (Su algoritmo solo necesita contar las rutas, no enumerarlas).

   para Realizar el algoritmo Agregamos un campo a la representación de los vértices de esta manera se mantiene un conteo de enteros. Inicialmente se establece el conteo de vértices t en 1 y el conteo de los otros vértices en 0. Se inicia el DFS con s como el vértice de inicio. Cuando se descubre t, se marca como terminado black , sin iniciar el el siguiente proceso. Posteriormente, cada vez que DFS finalice un vértice v, se establece el recuento de v en la suma de los conteos de todos los vértices adyacentes a v. Cuando el DFS finalice los vértices, se detiene la ejecucion y se devuelve el conteo calculado para s.