Universidad Nacional de Ingeniería
Ciencias de la Computación
Notación Asintótica y Recurrencias

Yuri Nuñez Medrano *
ynunezm@gmail.com

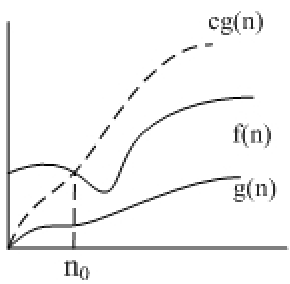

Figura 1: "gran O"

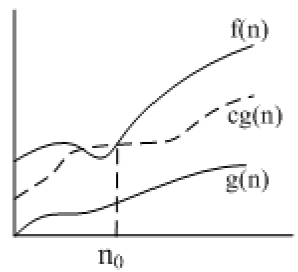

Figura 2: "gran Ω "

Resumen

Se analizará los diferentes casos de la Notación Asintótica y el Análisis de Algoritmos recursivos.

1. Introducción

Se analizará los diferentes casos de la Notación " Gran O", " Gran Ω", " Gran Θ", "pequeña o" y la "pequeña ω".
Se introducirá al análisis de recurrencias con sus métodos de resolución,

2. Notación Asintótica

Se analizara en los diferentes casos.

2.1. Notación " Gran O"

$f(n) = O(g(n))$
$\exists c > 0 \wedge n_0 > 0$
de tal manera que $0 \leq f(n) \leq cg(n)$
$\forall n \geq n_0$
se representa en la figura 1

Ejm: $2n^2 = O(n^3)$
$2n^2 \in O(n^3)$

*Escuela de Ciencias de la Computación, 27-08-15

En una formula se representa una función anónima Ejm:
$f(n) = n^3 + O(n^2)$
significa que hay una función anónima.
$h(n) \in O(n^2)$ de tal manera
$f(n) = n^3 + h(n)$

Ejm: $n^2 + O(n) = O(n^2)$
significa que para cualquier
$f(n) \in O(n)$
hay un $h(n) \in O(n^2)$ de tal manera que
$n^2 + f(n) = h(n)$

2.2. Notación "Gran Ω"

$f(n) = \Omega(g(n))$
$\exists c > 0 \wedge n_0 > 0$
de tal manera que $0 \leq cg(n) \leq f(n)$
$\forall n \geq n_0$
se representa en la figura 2

ejm: $\sqrt{n} = \Omega(lgn)$

2.3. Notación "Gran Θ"

$f(n) = \Theta(g(n))$
$\exists c_1 > 0 \wedge c_2 > 0 \wedge n_0 > 0$
de tal manera que $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$
$\forall n \geq n_0$

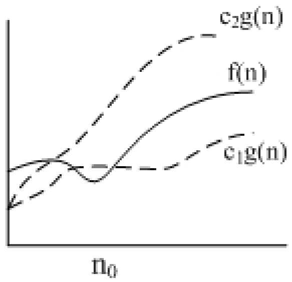

Figura 3: ”gran Θ ”

se representa en la figura 3

$O(n)$ es a menudo mal utilizada en lugar de $\Theta(g(n))$.

Algorithm 1: MERGE(A,p,q,r)

Input: Array A de n elementos de números
Output: Array A ordenado

$n_1 = q - p + 1$
$n_2 = r - q$
Let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
for $i = 1$ **to** n_1 **do**
    $L[i] = A[p + i - 1]$
for $j = 1$ **to** n_2 **do**
    $R[j] = A[q + j]$
$L[n_1 + 1] = \infty$
$R[n_2 + 1] = \infty$
$i = 1$
$j = 1$
for $k = p$ **to** r **do**
    if $L[i] \leq R[j]$ **then**
        $A[k] = L[i]$
        $i = i + 1$
    else
        $A[k] = R[j]$
        $j = j + 1$

2.4. Notación ” Pequeña o”

$f(n) = o(g(n))$
$\forall c > 0 \wedge n_0 > 0$
de tal manera que $0 \leq f(n) < cg(n)$
$\forall n \geq n_0$
Ejm: $2n = o(n^2)$
$2n^2 \neq o(n^2)$

2.5. Notación ”Pequeña ω”

$f(n) = \omega(g(n))$
$\forall c > 0 \wedge n_0 > 0$
de tal manera que $0 \leq cg(n)$ <f(n)
$\forall n \geq n_0$
Ejm: $\frac{n^2}{2} = \omega(n)$
$\frac{n^2}{2} \neq \omega(n^2)$

3. Recurrencias

Se analizarán el tiempo de ejecución (complejidad) de algoritmos recursivos.
Evaluando el tiempo de ejecución del algoritmo MERGE del algorítmo 1 es de $O(n)$.

Algorithm 2: MERGE_SORT(A,p,r)

Input: Array A de n elementos de número, donde el primer indice es p y el final es r
Output: Array A ordenado

1 **if** $p < r$ **then**
2     $q = \lfloor (p + r)/2 \rfloor$
3   MERGE_SORT(A,p,q)
4   MERGE_SORT(A,q+1,r)
5   MERGE(A,p,q,r)

Evaluando el tiempo de ejecución del algoritmo 2.
1: $O(1)$
2: $O(1)$
3: $t(n/2)$
4: $t(n/2)$
5: $O(n)$

El tiempo de ejecución del principal algoritmo 2 es $T(n)$, en las lineas 3 y 4 se denotan $T(n/2)$, significa que la recursividad se aplica a la misma funcion inicial pero con dimencion $n/2$, como se tienen 2 funciones recursivas al sumarlas tenemos $2T(n/2)$. Entonces deducimos la función del tiempo de ejecución del algoritmo MERGE_SORT.

$$t(n) = \left\{ \begin{array}{ll} \Theta(1) & si \quad n = 1 \\ 2T(n/2) + \Theta(n) & \end{array} \right\} \tag{1}$$

la que tiene diferentes metodos de resolución.

3.1. Metodo de Substitución

Es un método para resolver la recursión y tiene de tres pasos generales:

- Adivinar, la solución.
- Verificar, por inducción.
- Resolver, por constantes.

Ejm: T(n)=4T(n/2)+n
1. Adivinar $T(n) = O(n^3)$
2. Verificar $T(k) \leq ck^3$ para $k < n$
3. Resolver $T(n) = 4T(n/2) + n$
$\leq 4c(n/2)^3 + n$
$= \frac{1}{2}cn^3 + n$
$= \underbrace{cn^3}_{requerido} - \underbrace{(\frac{1}{2}cn^3 - n)}_{residuo\,positivo}$
$\leq cn^3$, si cumple $\frac{1}{2}cn^3 - n \geqslant 0$

Si cumple con la condición con su respectiva restricción.

Ejm: T(n)=4T(n/2)+n
1. Adivinar $T(n) = O(n^2)$
2. Verificar $T(k) \leq ck^2$ para $k < n$
3. Resolver $T(n) = 4T(n/2) + n$
$\leq 4c(n/2)^2 + n$
$= cn^2 + n$
$= \underbrace{cn^2}_{requerido} - \underbrace{(-n)}_{residuo\,positivo}$
$\not\leq cn^2$, no cumple $-n \geq 0$
no cumple con la inducción, $O(n^2)$ no es usada.
Pero si se usa una funcion cuadratica como lo siguiente.

Ejm: T(n)=4T(n/2)+n
2. Verificar $T(k) \leq c_1k^2 - c_2k$ para $k < n$
3. Resolver $T(n) = 4(c_1(\frac{n}{2})^2 - c_2\frac{n}{2}) + n$
$\leq c_1n^2 - c_2n - c_2n + n$
$= \underbrace{c_1n^2 - c_2n}_{requerido} - \underbrace{(c_2n - n)}_{residuo\,positivo}$

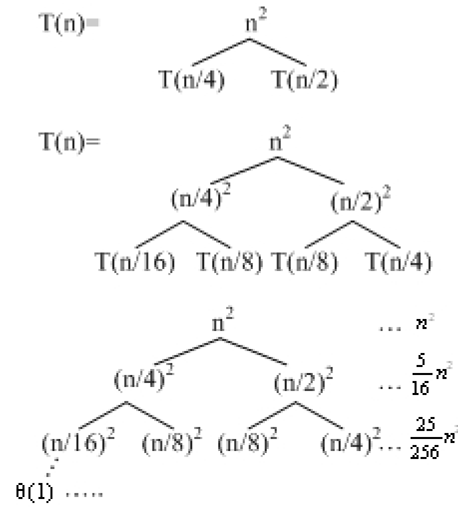

Figura 4: Construcción del arbol recursivo ”

$\leq c_1n^2 + c_2n$, si $c_2 \geqslant 1$

3.2. Metodo del arbol recursivo

Es la manera gráfica de resolver una recursión.
Ejm: $T(n) = T(n/4) + T(n/2) + n^2$ se resuelve en la figura 4. En el que se remplaza en la funcion n^2, las recursividades $T(n/4)$ y $T(n/2)$, en un nivel, luego en dos niveles y por ultimo en h niveles. Encontramos diferentes sumatorias por cada nivel y luego sumamos todos los niveles en.
$\leqslant \underbrace{(1 + \frac{5}{16} + \frac{25}{256} + ... + \frac{5^k}{16^k})}_{\approx 2} n^2$
$< 2n^2$
$= O(n^2)$

Ejm: Evaluar el tiempo de ejecución del algoritmo MERGE_SORT,de la ecuación 1, que seria.
$T(n) = 2T(n/2) + cn$.
se resuelve en la figura 5. En el que se remplaza en la funcion cn, las recursividades $T(n/2)$ y $T(n/2)$, en un nivel, luego en dos niveles y por ultimo en h niveles, que seria lgn. La sumatoria de cada nivel tiene el mismo resultado cn, multiplicandolo con la altura h, seria en total $cnlgn$, adicionando la suma de las constantes que es $\Theta(n)$. La suma total esta constituidad por $cnlgn + \Theta(n)$, por lo tanto el tiempo de ejecución estaria dado por $\Theta(nlgn)$.

Referencias

[H.Cormen et al., 2009] H.Cormen, T., Leiserson, C., and Riverson, R. L. (2009). *Algorithms*. The MIT Press.

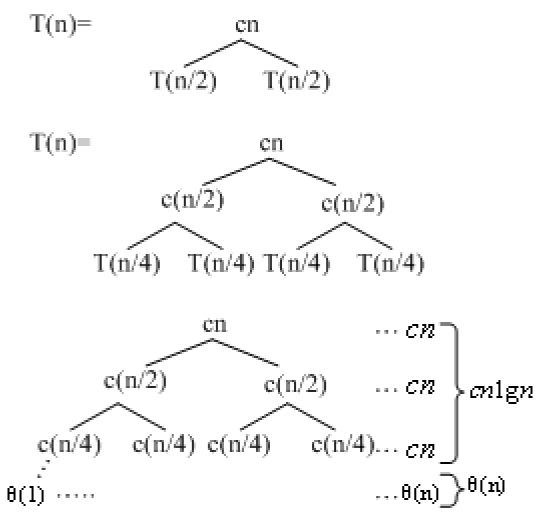

Figura 5: Construcción del arbol recursivo de MERGE_SORT"