Universidad Nacional de Ingeniería
Ciencias de la Computación
Master Method, Divide and Conquer

Yuri Nuñez Medrano *
ynunezm@gmail.com

Resumen

Se analizará el "Método Maestro" y el "Método Divide and Conquer" para el desarrollo de algorítmos recursivos.

1. Introducción

Se resolveran recurrencias con los tres casos del "Método Maestro" y utilizara el "Método Divide and Conquer" con su tres pasos Divide, Conquistar y Combinar.

2. Método Maestro

El Método Maestro se aplicará a las recurrencias que tengan la siguiente forma.
$T(n) = aT(\frac{n}{b}) + f(n)$
donde $a \geq 1$ $b > 1$.
$f(n) > 0$ para $n \geq n_0$
comparando con $n^{log_b a}$.

2.1. Casos del Master Method

- Caso 1: $f(n) = O(n^{log_b a - \varepsilon})$
  para algun $\varepsilon > 0$
  $\Rightarrow T(n) = \Theta(n^{log_b a})$
- Caso 2: $f(n) = \Theta(n^{log_b a} lg^k n)$
  para algun $k \geq 0$
  $\Rightarrow T(n) = \Theta(n^{log_b a} lg^{k+1} n)$
- Caso 3: $f(n) = \Omega(n^{log_b a + \varepsilon})$
  para algun $\varepsilon > 0$
  $\wedge af(n/b) \leq cf(n)$
  para algun $c < 1$ y todo n suficientemente grande
  $\Rightarrow T(n) = \Theta(f(n))$

Ejm: $T(n) = 4T(n/2) + \underbrace{n}_{f(n)}$
$a = 4, b = 2$
Reemplazamos a y b
$n^{log_b a} = n^{log_2 4} = n^2$
luego generamos constantes $\varepsilon = \{0.5,\ 1,\ 1.5,\ 2,\ 2.5\}$ que verifiquen, empezando por el caso1, y verificamos que se cumple para algun ε.
$f(n) = O(n^{log_b a - 0,5}) = O(n^{1,5})\checkmark$
$f(n) = O(n^{log_b a - 1}) = O(n^1)\checkmark$
$f(n) = O(n^{log_b a - 1,5}) = O(n^{0,5})\times$
$f(n) = O(n^{log_b a - 2}) = O(n^0)\times$
$f(n) = O(n^{log_b a - 2,5}) = O(n^{-0,5})\times$
$\Rightarrow T(n) = \Theta(n^{log_b a})$
$\Rightarrow T(n) = \Theta(n^2)$

Ejm: $T(n) = 4T(n/2) + \underbrace{n^2}_{f(n)}$
$a = 4, b = 2, f(n) = n^2$
Reemplazamos a y b
$n^{log_b a} = n^{log_2 4} = n^2$
luego generamos constantes $\varepsilon = \{0.5,\ 1,\ 1.5,\ 2,\ 2.5\}$ que verifiquen, empezando por el caso1, y verificamos del ejemplo anterior que no se cumple para algun ε.
Entonce evaluamos el caso2.
donde k=0
$f(n) = \Theta(n^{log_b a} lg^k n) = \Theta(n^{log_2 4}) = \Theta(n^2)$
si cumple
$\Rightarrow T(n) = \Theta(n^{log_b a} lg^{k+1} n)$
$\Rightarrow T(n) = \Theta(n^2 lgn)$

Ejm: $T(n) = 4T(n/2) + \underbrace{n^3}_{f(n)}$
$a = 4, b = 2, f(n) = n^3$
de los ejemplos anteriores verificamos que no cumple, el caso1 y el caso2, entonces evaluamos el caso3, y verificamos que se cumple para algun ε.
$f(n) = \Omega(n^{log_b a + 0,5}) = \Omega(n^{2,5})\checkmark$
$f(n) = \Omega(n^{log_b a + 1}) = \Omega(n^3)\checkmark$
$f(n) = \Omega(n^{log_b a + 1,5}) = \Omega(n^{3,5})\times$
$f(n) = \Omega(n^{log_b a + 2}) = \Omega(n^4)\times$
Y que cumpla un $c < 1$.

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

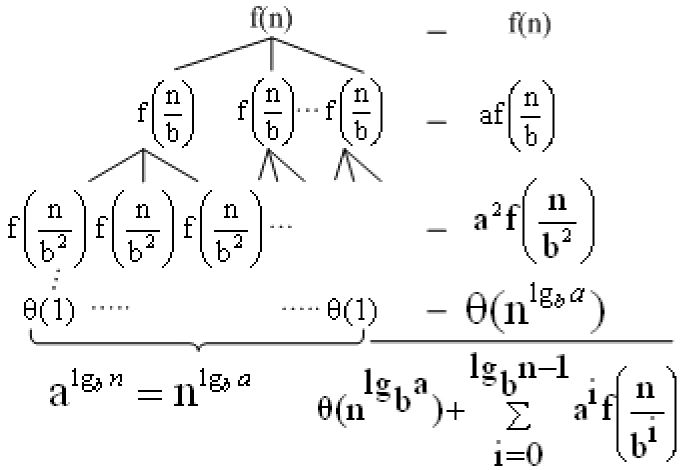
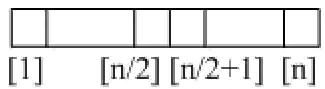


Figura 1: Método Maestro



$af(n/b) \leq cf(n)$
$4(n/2)^3 \leq cn^3$
$\Rightarrow T(n) = \Theta(f(n))$
$\Rightarrow T(n) = \Theta(n^3)$

2.2. Prueba del Método Maestro

Mediante una prueba intuitiva en base a la figura 1 se veran los tres casos.

- Caso 3: costo decreciente geometricamente. $\Rightarrow$ dominado por $f(n)$.
- Caso 1: costo incrementa geometricamente. $\Rightarrow$ dominado por $\Theta(n^{log_b a})$.
- Caso 2: cada nivel es $\approx$ a igual. $\Rightarrow$ costo $f(n) \underbrace{h}_{niveles}$.

3. Divide y Conquer

- 1. Divide: el problema se instancia en uno o mas problemas.
- 2. Conquistar: en cada subproblema recursivamente.
- 3. Combinar: las soluciones.

Ejm: Merge_Sort.
1. División: trivial
2. Conquistar: Ordenamiento recursivo en cada subarreglo.
3. Combinar: tiempo de ejecución.

$T(n) = 2T(n/2) + \Theta(n)$.
2T(...) numero de subproblemas.
(n/2) tamaño de subproblemas.
$\Theta(n)$ divide y conquistaras.
caso2 del MM $\Theta(nlgn)$

Ejm: Busqueda Binaria
Encontrar un X en un Array Ordenado
1. División: Comparar X con la mitad.
2. Conquistar: recursivamente en un subarreglo.
3. Combinar: Trivial. $T(n) = 1T(n/2) + \Theta(1)$.
$\Theta(lgn)$

Ejm: Potencia de un numero
Encontrar un número R, y un entero, n $\geq$ que de el resultado X^n. Algoritmo ingenuo $\underbrace{X.X...X}_{n}$. con $\Theta(n)$.
1. División: evaluar la mitad.
2. Conquistar: multiplicar grupos menores recursivamente.
3. Combinar: Trivial.

$$X^n = \left\{ \begin{array}{ccc} X^{n/2}.X^{n/2} & es & par \\ X^{n/2}.X^{n/2} & es & impar \end{array} \right\} \quad (1)$$

$T(n) = 1T(n/2) + \Theta(1)$.
$\Theta(lgn)$

Ejm: Multiplicaciones de Matrices
El algoritmo 1 es el clasico.

Algorithm 1: MUTIPLICACION(A,B)

Input: $A[a_{i,j}]$ y $B[b_{i,j}]$, $i, j = 1, 2, ..., n$
Output: $C[c_{i,j}] = A.B$

```
for i = 1 to n do
    for j = i to n do
        c_{i,j} = 0
        for j = 2 to n do
            c_{i,j} = c_{i,j} + a_{i,j} b_{i,j}
```

Tambien se puede resolver Recursivamente.
1. División: evaluar la mitad.
Una matriz n.n = 2 por 2, en bloques de matrices de = $\frac{n}{2}.\frac{n}{2}$ submatrices.
2. Conquistar: tenemos 8 multiplicaciones recursivas de = $\frac{n}{2}.\frac{n}{2}$ submatrices .

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

donde:
r=ae+bg
s=af+bh
t=ce+dg

u=cf+dh

3. Combinar: Trivial. $T(n) = 8T(n/2) + \Theta(n^2)$.
$\Theta(n^3)$

Ejm: Algoritmo de Strassen
Se puede reducir el número de multiplicaciones
P_1=a(f-h)
P_2=(a+b)h
P_3=(c+d)e
P_4=d(g-e)
P_5=(a+d)(e+h)
P_6=(b-d)(g+h)
P_7=(a-c)(e+f)
donde:
r=$P_5 + P_4 - P_2 + P_6$
s=$P_1 + P_2$
t=$P_3 + P_4$
u=$P_5 + P_1 - P_3 - P_7$
1. División: evaluar la mitad.
Una matriz n.n=2por2 en bloques de matrices de $= \frac{n}{2} \cdot \frac{n}{2}$ submatrices.
2. Conquistar: evaluar recursivamente $P_1, P_2, ...P_7$.
3. Combinar: r, s, t, u, $\Theta(n^2)$. $T(n) = 7T(n/2) + \Theta(n^2)$.
$\Theta(n^{2,8})$

Referencias

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