Septiembre 2016.pdf

CarlosGarSil98

Fundamentos de análisis de algoritmos

1º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería
Universidad de Huelva

EXAMEN SEPTIEMBRE 2016

$\boxed{\text{EJERCICIO 1}}$ Ordenar la complejidad de:

$n^2, n, n^3, n\log n, (n+3)^2, n\sqrt{n}, t(n)$:

$$t(n)\begin{cases} 1 & \text{si } n\le 1 \\ 4\,t(n/3)+n & \text{si } n>1\end{cases}$$

$n\sqrt{n} = n\cdot n^{0'5} = n^{1'5}$

$(n+3)^2 = n^2+6n+9 \in O(n^2)$

$t(n) - 4t(n/3) = n \longrightarrow$ cambio de base: $n=3^m$; $m=\log_3 n \longrightarrow$

$t_m - 4t_{m-1} = 3^m m^0$ NO HOMOGÉNEA

$(x-4)(x-3)=0 \begin{cases} r_1=4 \\ r_2=3\end{cases} \longrightarrow t(3^m) = C_1(4)^m m^0 + C_2(3)^m m^0$

$t(n) = C_1(4)^{\log_3 n} + C_2(3)^{\log_3 n} \longrightarrow t(n) = C_1(n)^{\log_3 4} + C_2(n)^{\log_3 3}$

$t(n) = C_1 n^{1'26184} + C_2 n \longrightarrow t(n) \in O(n^{1'26\ldots})$

Hacemos una ordenación estimada:

$$O(n) \in O(n^{1'26184}) \in O(n\log n) \in O(n^{1'5}) \in O(n^2) = O((n+3)^2) \in O(n^3)$$

Vamos a comprobar $n\log n$ y $n\sqrt{n}$

$$\lim_{n\to\infty} \frac{n\log n}{n\cdot n^{0'5}} = \frac{\log n}{n^{0'5}} \longrightarrow \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2}n^{-\frac{1}{2}}} = 0$$

$O(n\log n) \in O(n\sqrt{n})$

ordenación final:

$$O(n) \in O(n\log n) \in O(n^{1'26\ldots}) \in O(n\sqrt{n}) \in O(n^2) = O((n+3)^2) \in O(n^3)$$

$$\Omega(n^3) \in \Omega((n+3)^2) = \Omega(n^2) \in \Omega(n\sqrt{n}) \in \Omega(n^{1'26\ldots}) \in \Omega(n\log n) \in \Omega(n)$$

$$\Theta(n^2) = \Theta((n+3)^2)$$

EJERCICIO 2

a) Calcular complejidad para caso mejor ecuación característica

Caso mejor: ocurre cuando el vector queda dividido en dos partes iguales

$$\text{Quicksort}(n)\begin{cases} 1 & \text{si } n=1 \\ 1+1+1+\text{Partition}(n)+1+\text{Quicksort}(n/2)+\text{Quicksort}(n/2) & \text{si } n>1 \end{cases}$$

Partition siempre será lineal $\rightarrow cn$

mientras $j \geq i \rightarrow$ es lo mismo que $\sum_{i=1}^{n-1}(\ldots) = c\cdot(n-1+1) = cn$

$t(n) = 4 + cn + 2t(n/2) \rightarrow t(n) - 2t(n/2) = 4 + cn \rightarrow$ cambio base $n = 2^m$

$m = \log_2 n$

$t_m - 2t_{m-1} = 4 + c\cdot 2^m \rightarrow t_m - 2t_{m-1} = 4\cdot 1^m m^0 + c\cdot 2^m \cdot m^0$

NO HOMOGENEA

$(x-2)(x-1)(x-2) = 0 \begin{cases} r1 = 2 \text{ (doble)} \\ r2 = 1 \end{cases} \rightarrow t(2^m) = C_1(2)^m m^0 + C_2(2^m)m^1 + C_3(1)^m$

$t(n) = C_1(2)^{\log_2 n} + C_2(2)^{\log_2 n}\log_2 n + C_3 \rightarrow t(n) = C_1(n)^{\log_2 2} + C_2(n)^{\log_2 2}\log_2 n + C_3$

$t(n) = C_1 n + C_2 n\log_2 n + C_3 \rightarrow t(n) \in O(n\log n)$

b) Caso mejor a partir del teorema maestro

$t(n) = at(n/b) + O(n^k \log^{p+1} n)$

$t(n) = 2t(n/2) + O(n) + 4 \rightarrow a = b = 2 \rightarrow a = b^k \rightarrow k=1 \rightarrow 2 = 2^1;\ p = 0$

$t(n) \in O(n^k \log^{p+1} n) \rightarrow t(n) \in O(n\log n)$

c) Caso peor ecuación característica

$$t(n)\begin{cases} 1 & \text{si } n=1 \\ cn + t(n-1) & \end{cases}$$

$t(n) - t(n-1) = cn \longrightarrow t(n) - t(n-1) = c \cdot 1^n \cdot n$

$(x-1)(x-1)^2 = 0 \longrightarrow r1 = 1$ (triple)

$t(n) = C_1 (1)^n n^0 + C_2 (1)^n n + C_3 (1)^n n^2 \longrightarrow t(n) = C_1 + C_2 n + C_3 n^2$

$t(n) \in O(n^2)$

$$\lim_{n \to \infty} \frac{n \cdot \log n}{n \cdot n} = \frac{\log n}{n} = 0 \longrightarrow O(n \log n) \in O(n^2)$$

es caso peor es más ineficiente que el caso mejor

$\boxed{\text{EJERCICIO 3}}$ Resolver las siguientes ecuaciones:

1.
$$t(n) \begin{cases} 2 & \text{si } n = 1 \\ 2 + 2t(n-1) & \text{si } n > 1 \end{cases}$$

$t(n) - 2t(n-1) = 2$ NO HOMOGENEA

$t(n) - 2t(n-1) = 2 \cdot 1^n \cdot n^0$

$(x-2)(x-1) = 0 \longrightarrow t(n) = C_1 (2)^n n^0 + C_2 (1)^n n^0 \longrightarrow t(n) = C_1 (2)^n + C_2$

$t(n) \in O(2^n)$

$$h(n) \begin{cases} 2 & \text{si } n = 1 \\ 2 + h(n-1) + 2 \cdot 2^{n-1} & \text{si } n > 1 \end{cases}$$

$h(n) - h(n-1) = 2 + 2^n \longrightarrow h(n) - h(n-1) = 2 \cdot 1^n n^0 + 2^n n^0$

$(x-1)(x-1)(x-2) = 0 \begin{cases} r1 = 1 \text{ (doble)} \\ r2 = 2 \end{cases} \longrightarrow t(n) = C_1 (1)^n n^0 + C_2 (1)^n n^1 + C_3 (2)^n n^0$

$t(n) C_1 + C_2 n + C_3 2^n \longrightarrow t(n) \in O(n)$

2.

$$t(n)\begin{cases} 2 & \text{si } n=1 \\ 2+1+t(n-1) & \text{si } n>1 \end{cases}$$

$t(n) = 3 + t(n-1) \rightarrow t(n) - t(n-1) = 3 \cdot 1^n n^0 \rightarrow$

$(x-1)(x-1)=0$; $r_1 = 1$ (doble) $\rightarrow t(n) = C_1(1)^n n^0 + C_2(1)^n n^1$

$t(n) \in O(n)$

3.

$$t(n)\begin{cases} a & \text{si } n=1 \\ 2t(n/4) + Lg(n) & \text{si } n>1 \end{cases}$$

$t(n) - 2t(n/4) = Log\, n \rightarrow$ cambio de base: $n = 4^m \rightarrow m = Log_4 n$

$t_m - 2t_{m-1} = Log(4^m) \rightarrow t_m - 2t_{m-1} = Log(4)\, 1^m m^1$

$(x-2)(x-1)^2 = 0 \begin{cases} r_1 = 2 \\ r_2 = 1 \text{ (doble)} \end{cases} \rightarrow t(4^m) = C_1(2)^m m^0 + C_2(2)^m m^0 + C_3(2)^m m^1$

$t(n) = C_1(2)^{Log_4 n} + C_2(2)^{Log_4 n} + C_3(2)^{Log_4 n} Log_4 n \rightarrow$

$t(n) = C_1 n^{Log_4 2} + C_2 n^{Log_4 2} + C_3 n^{Log_4 2} Log_4 n \rightarrow n^{Log_4 2} = n^{1/2} = \sqrt{n}$

$t(n) \in O(\sqrt{n}\, Log_4 n)$

$\boxed{\text{EJERCICIO 4}}$

```
funcion Greedy(i[1...n], bi[1..n], di[1...n] : entero) : entero
Var:
    S ← Ø   // conjunto solución, almacena la i
inicio
    Quicksort (i, bi, di)  // ordena conjunto di y cambia el resto
    S[1] ← i[1]
    para j ← 2 hasta n hacer
        si di[j-1] = di[j] AND bi[j-1] < bi[j] entonces
            S[j-1] ← i[j]
        sino
            si di[j-1] < di[j]
                S[j-1] ← i[j]
            fsi
        fsi
    fpara
    devuelve S
```

Esquema general algoritmo voraz:

Candidatos:

- conjunto "i"
- conjunto "bi"
- conjunto "di"

Conjunto solución:

- conjunto "S"

Hemos dado por hecho que siempre habrá solución, no hay método "solucion()"

El método "seleccionar()" es sustituido por el bucle for

EL método "factible()" lo realizan los dos condicionales

Y por último el método "insertar()" es sustituido por una asignación "S[j-1] ← I[j]"

Traza para:

i	1	2	3	4
bi	50	10	15	30
di	2	1	2	1

Quicksort(i, bi, di)

	1	2	3	4
i	2	4	1	3
bi	10	30	50	15
di	1	1	2	2

S	2	-	-	-

j=2 → 1=1 AND 10 < 30 → S[1] = 4

i	2	4	1	3
bi	10	30	50	15
di	1	1	2	2

S	4	-	-	-

j=3 → 1 = 2 AND 30 < 50
→ 1 < 2 → S[2] = 1

i	2	4	1	3
bi	10	30	50	15
di	1	1	2	2

S	4	1	-	-

j=4 → 2=2 AND 50 < 15
→ 2 < 2 (No hace nada)

j=5 → j>n (Acaba bucle) ⟹

Devuelve

S	4	1