Técnicas de diseño de algoritmos
Divide y Vencerás
Ejercicios (Bloque 2): con pautas de resolución

Luis Javier Rodríguez Fuentes
Amparo Varona Fernández

Departamento de Electricidad y Electrónica
Facultad de Ciencia y Tecnología, UPV/EHU
luisjavier.rodriguez@ehu.es
amparo.varona@ehu.es

OpenCourseWare 2015
Campus Virtual UPV/EHU

Divide y Vencerás – Ejercicios (Bloque 2)

(B2.1) Resolver exactamente la siguiente recurrencia mediante el método de la ecuación característica:

$$t(n) = \begin{cases} 1 & n = 0 \\ 2t(n-1) + (n+5)\ 3^n & n > 1 \end{cases}$$

Recurrencia lineal de coeficientes constantes no homogenea: la expresión general de la solución debe tener en cuenta las raíces de la parte homogenea y las de la parte no homogenea, con sus respectivas multiplicidades. A continuación, para resolver las constantes, se introduce la solución general en la recurrencia y se igualan coeficientes. Por último, se introducen las condiciones iniciales, resolviendo el sistema lineal de ecuaciones que resulta.

(B2.2) Resolver exactamente la siguiente recurrencia mediante el método de la ecuación característica:

$$t(n) = \begin{cases} 1 & n = 1 \\ 5t(n/2) + (n \log_2 n)^2 & n > 1 \end{cases}$$

Recurrencia lineal de coeficientes constantes no homogenea. En este caso, para obtener una ecuación en diferencias debe aplicarse el cambio de variable $i = \log_2 n$, o lo que es lo mismo, $n = 2^i$, de modo que $T(i) \equiv t(2^i) = t(n)$ y $t(n/2) \equiv T(i-1)$. Una vez hecho el cambio, se siguen los mismos pasos que en el ejercicio anterior, con cuidado de aplicar las condiciones iniciales para la nueva variable (en este caso, $T(0) = 1$). Por último, una vez resuelta la recurrencia, se deshace el cambio de variable.

Divide y Vencerás – Ejercicios (Bloque 2)

(B2.3) Escribir en lenguaje Python un algoritmo de tipo *Divide y Vencerás* que calcule la traspuesta de una matriz cuadrada M de tamaño $n = 2^k$ (con $k \geq 0$). Calcular su complejidad temporal.

La matriz M se divide en 4 submatrices cuadradas, cada una de tamaño $n/2$: M_{11}, M_{12}, M_{21} y M_{22}, tal que M_{11} y M_{22} ocupan la diagonal principal, y M_{21} y M_{12} la diagonal opuesta. La matriz traspuesta $T = M'$ se construye haciendo $T_{11} = M'_{11}$, $T_{22} = M'_{22}$, $T_{12} = M'_{21}$ y $T_{21} = M'_{12}$. Las traspuestas de las submatrices se calculan llamando recursivamente al mismo algoritmo. El caso base se alcanza cuando la matriz M tiene tamaño 1, en cuyo caso la traspuesta es la propia matriz.

(B2.4) Escribir en lenguaje Python un algoritmo de tipo *Divide y Vencerás* que calcule el producto de dos matrices cuadradas A y B, ambas de tamaño $n = 2^k$ (con $k \geq 0$). Calcular su complejidad temporal.

Como en el ejercicio anterior, cada matriz se divide en 4 submatrices cuadradas, cada una de tamaño $n/2$, tales que la matriz producto $P = AB$, formada a su vez por las 4 submatrices P_{11}, P_{12}, P_{21} y P_{22}, se obtiene como sigue: $P_{11} = A_{11}B_{11} + A_{12}B_{21}$, $P_{12} = A_{11}B_{12} + A_{12}B_{22}$, $P_{21} = A_{21}B_{11} + A_{22}B_{21}$ y $P_{22} = A_{21}B_{12} + A_{22}B_{22}$. Los productos de submatrices se calculan llamando recursivamente al mismo algoritmo. En el caso base, las matrices A y B tienen tamaño 1, es decir, son dos escalares a y b, y la matriz producto P será también un escalar $p = ab$.