Auxiliar 4 - Recurrencia y Programación Dinámica

Profesores: Nelson Baloian
Jeremy Barbay
Patricio Poblete
Auxiliares: Gabriel Flores, Sven Reisenegger
Juglar Díaz, Gabriel Norambuena
Cristóbal Muñoz

Teorema Maestro

El teorema maestro se utiliza harto para resolver este auxiliar y dice lo siguiente. Dada una función de recurrencia del tipo $T(n) = aT(\frac{n}{b}) + f(n)$, si es que $f(n) = kn^c = O(n^c)$ entonces la solución es:

$$T(n) = \Theta(n^{log_b a}) \quad \text{si } a > b^c \tag{1}$$
$$T(n) = \Theta(n^c) \quad \text{si } a < b^c \tag{2}$$
$$T(n) = \Theta(n^c log(n)) \quad \text{si } a = b^c \tag{3}$$

P1. Stooge Sort

Stooge sort es un algoritmo muy poco eficiente para ordenar una secuencia de números. El objetivo de este problema es analizar qué tan malo es el algoritmo. Este algoritmo funciona de la siguiente manera:

(a) Si el valor en el índice 0 es mayor que el valor en el último índice, entonces intercambia ambos valores.

(b) Recursivamente:

i. Ordena los primeros 2/3 del arreglo con stoogeSort
ii. Ordena los últimos 2/3 del arreglo con stoogeSort
iii. Ordena los primeros 2/3 del arreglo con stoogeSort

Escriba la ecuación de recurrencia para este algoritmo y luego resuélvala.

Solución:

En este caso la ecuación de recurrencia es $T(n) = 3T(\frac{n}{3/2}) + 1$ por lo que la solución es $T(n) = O(n^{log_{3/2}3}) = O(n^{2.709})$ (primer caso teorema maestro). Esto es mucho peor que los algoritmos que ya hemos visto, como bubble sort e insertion sort que tienen complejidad $O(n^2)$. Esto significa que el algoritmo es muy malo en términos de tiempo, además de que es complejo de entender su funcionamiento.

P2. Algoritmo de Strassen

Para obtener el producto P de dos matrices A y $B \in \mathcal{M}_{n \times n}$ se pensaba que la única forma era calcular $p_{ij} = \Sigma_k^n \, a_{ik} \, b_{kj}$, como seguro recordará de su curso de Álgebra Lineal.

(a) Formule la ecuación de recurrencia. *El orden del algoritmo es de la forma* $O(n^k)$.

Si se toma n como potencia de 2, y se dividen las matrices en cuadrantes, se puede demostrar (al menos, Strassen pudo) que hay un conjunto de 7 elementos ($\{M_1, \ldots, M_7\}$) formados por productos de combinaciones de los 8 cuadrantes de A y B. Estos elementos forman una base a partir de la cual se pueden obtener los cuadrantes del resultado P. Los términos son:

$$\begin{array}{ll} M_1: & (A_{11}+A_{22})(B_{11}+B_{22}) \\ M_2: & (A_{21}+A_{22})B_{11} \\ M_3: & A_{11}(B_{12}-B_{22}) \\ M_4: & A_{22}(B_{21}-B_{11}) \end{array} \qquad \begin{array}{ll} M_5: & (A_{11}+A_{12})B_{22} \\ M_6: & (A_{21}-A_{11})(B_{11}+B_{12}) \\ M_7: & (A_{12}-A_{22})(B_{21}+B_{22}) \end{array}$$

(b) Considerando que en cada término M_i, el producto puede descomponerse de la misma manera, y considerando que el costo de la suma de matrices cuadradas de lado n es $O(n^2)$, obtenga la ecuación de recurrencia para el Algoritmo de Strassen. Para calcular los cuadrantes de P, sólo basta hacer 8 sumas de los términos de $\{M_1, \ldots, M_7\}$. ¿Cuál es una cota superior para éste?

Solución:

(a) Considerando el algoritmo recursivo que divide la matriz en 4 y luego obtiene el resultado de cada cuadrante de C como la multiplicación de 4 cuadrantes de A y B (i.e. $C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1}$) entonces, considerando que la suma de estas matrices toma tiempo $O(n^2)$, la ecuación de recurrencia y su solución (usando el cuarto caso del teorema maestro) son:

$$T(n) = 8T(\frac{n}{2}) + f(n) = O(n^{log_2 8}) = O(n^3) \text{ donde } f(n) = O(n^2) \text{ y } 2 < log_2 8 = 3$$

*f(n) es la suma de las matrices en cada iteración

(b) como se realizan 7 multiplicaciones y todas las otras operaciones en una recurrencia son sumas $O(n^2)$ la ecuación de recurrencia y su solución (por el cuarto caso del teorema maestro) son:

$$T(n) = 7T(\frac{n}{2}) + f(n) = O(n^{log_2 7}) = O(n^{2.807}) \text{ donde } f(n) = O(n^2) \text{ y } 2 < log_2 7 = 2.807$$

*$f(n)$ es la suma de las matrices en cada iteración

P3. Distancia de Edición

Dados dos strings S y T queremos encontrar la mínima cantidad de pasos para transformar S en T. Para esto contamos con 4 posibles acciones para cada letra:

- Insertar
- Eliminar
- Cambiar
- Mantener

Todas las opciones, excepto mantener una letra, se consideran 1 paso. Por ejemplo, si queremos transformar el String *casa* a *calle*, tenemos que hacer 3 operaciones: Primero cambiamos la letra s en l, nos queda *cala*, luego cambiamos la última letra a en e, quedando *cale* y finalmente agregamos una l y llegamos a *calle*.
Programe un método `public static int distanciaEdicion(String S, String T)` que basado en una estrategia de Programación Dinámica, resuelva la distancia de edición entre Strings.

P4. BONUS Obtenga las cotas superiores para los siguientes problemas utilizando el teorema maestro.

(a) $T(n) = 2\ T(\frac{n}{2}) + n^3$

$$a = 2,\ b = 2,\ c = 3$$
$$b^c = 8$$
$$b^c = 8 > a = 2$$
$$\implies \Theta(T(n)) = \Theta(n^3)$$

(b) $T(n) = 16\ T(\frac{n}{4}) + n^2$

$$a = 16,\ b = 4,\ c = 2$$
$$b^c = 16$$
$$b^c = 16 = a$$
$$\implies \Theta(T(n)) = \Theta(n^2 log(n))$$

(c) $T(n) = 7\ T(\frac{n}{3}) + n^2$

$$a = 7,\ b = 3,\ c = 2$$
$$b^c = 9$$
$$b^c = 9 > 7 = a$$
$$\implies \Theta(T(n)) = \Theta(n^2)$$

(d) $T(n) = 7\ T(\frac{n}{2}) + n^2$

$$a = 7,\ b = 2,\ c = 2$$
$$b^c = 4$$
$$b^c = 4 < 7 = a$$
$$log_b(a) = log_2(7) \approx 2.81$$
$$\implies \Theta(T(n)) = \Theta(n^{log_b(a)}) = \Theta(n^{2.81})$$

(e) $T(n) = 2\ T(\frac{n}{4}) + \sqrt{n}$

$$a = 2,\ b = 4,\ c = 0.5$$
$$b^c = 2$$
$$b^c = 2 = a$$
$$\implies \Theta(T(n)) = \Theta(\sqrt{n}\ log(n))$$

(f) $T(n) = 3\ T(\frac{n}{2}) + n\ log(n)$

$$\text{Aqui consideramos que } n\ log(n) \approx n^{1+\epsilon}$$
$$a = 3,\ b = 2,\ c = 1 + \epsilon$$
$$b^c \approx 2$$
$$b^c \approx 2 < 3 = a$$
$$log_b(a) = log_2(3) \approx 1.58$$
$$\implies \Theta(T(n)) = \Theta(n^{log_b(a)}) = \Theta(n^{1.58})$$

(g) $T(n) = 4\ T(\frac{n}{2}) + n^2\ \sqrt{n}$

$$a = 4,\ b = 2,\ c = 2.5$$
$$b^c \approx 5.66$$
$$b^c \approx 5.66 > 4 = a$$
$$\implies \Theta(T(n)) = \Theta(n^{2.5})$$

(h) $T(n) = T(\frac{9n}{10}) + n$

$$a = 1,\ b = 10/9,\ c = 1$$
$$b^c \approx 1.11$$
$$b^c \approx 1.11 > 1 = a$$
$$\implies \Theta(T(n)) = \Theta(n)$$

(i) $T(n) = 2\ T(\frac{n}{2}) + n\ log(n)$

$$a = 2,\ b = 2,\ c = 1 + \epsilon$$
$$b^c \approx 2$$
$$b^c \approx 2 = a$$
$$\implies \Theta(T(n)) = \Theta(n\ log(n))$$