Cuarto examen parcial

AA 361075 Carlos Eduardo Sánchez Torres

5 de diciembre de 2021

V.1) Ejercicio 34.1-5. Demuestre que si un algoritmo realiza a lo más un número constante de llamadas a una subrutina de tiempo polinomial, y cualquier otra tarea adicional toma en total también tiempo polinomial, entonces el algoritmo completo se ejecuta en tiempo polinomial. Adicionalmente, demuestre de qué manera un número de llamados polinomiales a subrutinas de tiempo polinomial podrían resultar en un costo total exponencial.

$$T_2(n) = \sum_{i=0}^{l} c_i n^i \tag{1}$$

$$T_m(n) = (\overbrace{T_2 \circ T_2 \circ T_2 ...}^{\text{m veces}})(n) \tag{2}$$

Para $c \geq c_l$ y toda $n \geq 1$, siendo T_2 monótonamente creciente

$$T_2(n) = \sum_{i=0}^{l} c_i n^i \leq cn^l \tag{3}$$

A. Si un algoritmo realiza a lo más un número constante de llamadas a una subrutina de tiempo polinomial, y cualquier otra tarea adicional toma en total también tiempo polinomial, entonces el algoritmo completo se ejecuta en tiempo polinomial:

$$T_1(n) = T_m + \sum_{i=0}^{k} c_i n^i, m \in \mathbb{N} \tag{4}$$

Por 2 y 3

$$T_m(n) \leq (\overbrace{cn^l \circ cn^l \circ cn^l ...}^{\text{m veces}})(n) = c^{\sum_{i=0}^{m-1} l^i} n^{l^m} \tag{5}$$

Para $C_3 \geq 1$ y toda $l \geq 1$

$$T_m(n) \leq c^{C_3 l^{m-1}} n^{l^m} \tag{6}$$

Para $C_2 \geq c^k$ y toda $n \geq 1$

$$T_1(n) \leq c^{C_3 l^{m-1}} n^{l^m} + C_2 n^k \tag{7}$$

$$\leq Mn^{k+l^m}, M \geq c^k + c^{C_3 l^{m-1}} \tag{8}$$

Entonces,

$$T_1(n) = O(n^{k+l^m}) \tag{9}$$

A saber, $T_1(n)$ se ejecuta en tiempo polinomial.

B. Un número de llamados polinomiales a subrutinas de tiempo polinomial podrían resultar en un costo total exponencial:

$$T_1(n) = T_m, m = \sum_{i=0}^{k} c_i n^i \tag{10}$$

Para $M_2 \geq c^k$ y toda $n \geq 1$

$$m \leq M_2 n^k \tag{11}$$

Por 5

$$T_m(n) \le (\overbrace{cn^l \circ cn^l \circ cn^l ...}^{M_2 n^k \text{ veces}})(n) = c^{\sum_{i=0}^{M_2 n^k} l^i} n^{l^{M_2 n^k}} \tag{12}$$

$$\le c^{M_3 n^k} n^{l^{M_2 n^k}} \tag{13}$$

Como $c^{M_3 n^k} > n^{l^{M_2 n^k}}$ entonces,

$$T_1(n) = O(c^{n^k}) \tag{14}$$

A saber, $T_1(n)$ se ejecuta en tiempo exponencial.

IV.2) Ejercicio 34.2-6. Una Ruta Hamiltoniana (HAM-PATH) en un grafo es una ruta simple que visita todos y cada uno de los vértices exactamente una vez. Demuestre que HAM-PATH = $\{< G; u; v >$: existe una Ruta Hamiltoniana del vértice u al vértice v en el grafo G$\}$ pertenece a la clase de complejidad NP.

Para que un algoritmo pertenezca a la clase de complejidad NP, debe cumplir dos condiciones el algoritmo HAM-PATH debe ser de decisión (la cual ya cumple), y debe ser verificable en tiempo polinomial. Para lo segundo, con un certificado $p =< u_0, ..., v_{|V|} >$, puede crearse un algoritmo que verifique que para toda $i, j >= 0, v_i \neq v_j$ en p, $O(V^2)$, que correspondan los elementos con las aristas y vértices en el grafo $O(E)$ y $O(V)$ respectivamente, entonces HAM-PATH puede verificarse en tiempo polinomial. Por lo tanto, HAM-PATH pertenece a la clase de complejidad NP.

IV.3) Ejercicio 34.4-6. Suponga que le proporcionan un algoritmo de tiempo polinomial que decide la satisfabilidad de una fórmula (SAT). Describa cómo utilizaría este algoritmo para encontrar una asignación de valores a las variables de la fórmula, tal que la satisfagan, en tiempo polinomial.
Sea

$$SAT = \{< \phi >: \phi \text{ es una fórmula booleana satisfactible.}\} \tag{15}$$

Si definimos un algoritmo A tal que de una asignación de valores a las variables de la fórmula tal que la satisfagan. Primero llamamos a SAT para comprobar si es una fórmula booleana satisfactible. En caso de serlo, para una entrada ϕ con m variables $x_0, ..., x_m$, A llama a SAT con n variables y $x_i = 1$, , si no es factible $x_i = 0$, remplazando en ϕ con x_i llamamos de nuevo A con $m - 1$ variables, y así recursivamente desde $i = 0$ a $i = m$. Y sea T(n) el tiempo de ejecución de A para una codificación de tamaño n:

$$T(n) = O(n^k) + T'(n) = O(n^k) + O(1) = O(n^k) \tag{16}$$

El método realiza m llamadas recursivas y termina cuando $i = m$, entonces A se ejecuta en tiempo polinomial.

Referencias

[1] Cormen, T. H., & Leiserson, C. E. (2009). In Introduction to algorithms (3rd Edition).