uniandes

Ingeniería de Sistemas y Computación

Diseño y análisis de algoritmos

Profesor: Mateo Sanabria Ardila

Taller 1: Orden de complejidad y Ecuaciones de recurrencia

Fecha de entrega: 17-Feb | 2023-10 | Nota máxima: 100

1. (30 points) Para los siguientes problemas:

   I Proponga un algoritmo que solucione el problema. (`Python,Java`)

   II Proponga una tabla que enumere las operaciones que se puede suponer que se ejecutan en tiempo constante.

   III Derive la función de costo, $T(n)$, para el algoritmo que propuso, basado en las operaciones constantes del punto anterior.

   IV En caso de que la función sea una ecuación de recurrencia, resolver la ecuación.

   V Determinar el orden de complejidad del algoritmo.

   .....

   (a) (10 points) Dada una matriz cuadrada de números enteros sumar los elementos de la diagonal principal.

   (b) (10 points) Dada una matriz de números enteros y un numero entero, se debe devolver $True$ si el numero se encuentra en la matriz y $False$ en otro caso.

   (c) (10 points) Dada un arreglo de números enteros y un numero entero, se debe devolver $True$ si el numero se encuentra en el arreglo y $False$ en otro caso. **Solución recursiva**

2. (50 points) Soluciones las siguientes ecuaciones de recurrencia y determine el orden de complejidad, para las ecuaciones no lineales debe:

   I Proponer una cota usando arboles de recurrencia.

   II Demostrar la cota propuesta usando el método de sustitución.

   III **De ser posible** compruebe su respuesta usando el método maestro.

   .....

   (a) (10 points) $T(n+1) = 2T(n) - T(n-1)$, $T(0) = 1$, $T(1) = 5$.

   (b) (10 points) $T(n) = 3T(n) - n$, $T(0) = 1$.

   (c) (10 points) $T(n) = 7T(\frac{n}{2}) - 20n^2$.

   (d) (10 points) $T(n) = T(\frac{n}{2}) + 1$.

   (e) (10 points) $T(n) = 4T(\frac{n}{2} + 2) + n$.

3. (20 points) El profesor Marco desea desarrollar un algoritmo de multiplicación de matrices que sea asintóticamente más rápido que el algoritmo de Strassen. Su algoritmo utilizará el método de dividir y conquistar, dividiendo cada matriz en trozos de tamaño $n/4 \times n/4$, y los pasos de dividir y combinar juntos tardarán $\Theta(n^2)$. Necesita determinar cuántos subproblemas tiene que crear su algoritmo para vencer al algoritmo de Strassen. Si su algoritmo crea un subproblema, entonces la recurrencia para el tiempo de ejecución $T(n)$ se convierte en $T(n) = aT(n/4) + \Theta(n^2)$ ¿Cuál es el mayor valor entero de a para el que el algoritmo del profesor Marco sería asintóticamente más rápido que el algoritmo de Strassen?