Solución Primer examen parcial
Fundamentos de análisis y diseño de algoritmos
Duración: 2 horas

Carlos Andres Delgado S, Ing *

25 de Abril 2017

Nombre:________________________________ **Código:**________________

1. Crecimiento de funciones [15 puntos]

Indique si existen funciones $f(n)$, $g(n)$ y $h(n)$, tales que $f(n)$ es $\Theta(g(n))$, $g(n)$ es $o(h(n))$ y $h(n)$ es $\omega(f(n))$. Realice una demostración y muestre con un ejemplo.

1.1. Respuesta

Si existen estas funciones debe cumplirse

1. $f(n)$ es $\Theta(g(n))$, se cumple que $c_1 g(n) \geq f(n)$ y $c_2 g(n) \leq f(n)$
2. $g(n)$ es $o(h(n))$, se cumple $c_3 h(n) > g(n)$
3. $h(n)$ es $\omega(f(n))$, se cumple que $c_4 f(n) < h(n)$

De aqui se observa que $h(n)$ debe ser una función de mayor grado polinomail que $f(n)$ y $g(n)$ las cuales deben ser del mismo orden polinomial, por ejemplo sirve

- $f(n) = n^2$
- $g(n) = n^2$
- $h(n) = n^4$

2. Ecuaciones de recurrencia [30 puntos]

1. (15 puntos) Utilizando el método de iteración, solucione la siguiente ecuación de recurrencia

$$T(n) = T(n-1) + 3n, T(0) = O(1)$$

Empezando a iterar

$$\begin{aligned} T(n-1) &= T(n-2) + 3(n-1) \\ T(n) &= T(n-2) + 3(n-1) + 3n \\ &\ldots \\ T(n) &= T(n-i) + 3(n-1) + 3(n-2) + \ldots + 3(n-i-1) \end{aligned}$$

Colocando en forma de sumatoria

$$T(n) = T(n-i) + \sum_{j=1}^{i-1} 3(n-j)$$

*carlos.andres.delgado@correounivalle.edu.co

Las iteraciones se detienen cuando $T(n-i) = T(0)$, entonces $i = n$ quedando:

$$T(n) = T(0) + \sum_{j=1}^{n-1} 3(n-j)$$
$$T(n) = T(0) + \sum_{j=1}^{n-1} 3n - \sum_{j=1}^{n-1} 3j$$

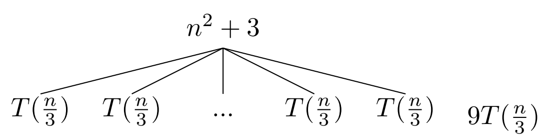
Solucionando

$$T(n) = C + 3n^2 - 3n + 3\frac{n(n+1)}{2} - 3n$$
$$O(n^2)$$

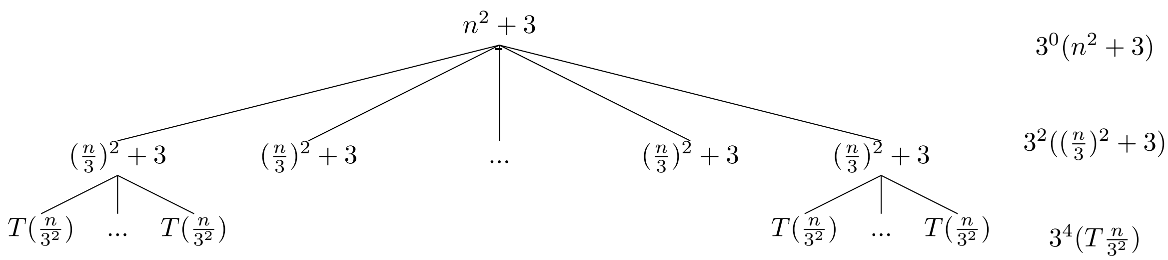
2. (15 puntos) Utilizando el método de árboles, solucione la siguiente ecuación de recurrencia

$$T(n) = 9T(\frac{n}{3}) + n^2 + 3, T(1) = O(n)$$

Representación:



Primera expansión:



En una expansión i se obtiene:

$$T(n) = 3^{2i}T(\frac{n}{3^i}) + \sum_{j=0}^{i-1}(3^{2j}\frac{n}{3^j} + 3)$$
$$T(n) = 3^{2i}T(\frac{n}{3^i}) + \sum_{j=0}^{i-1}(3^j n + 3)$$

Las expansiones se detienen cuando: $T(\frac{n}{3^i}) = T(1)$ entonces $i = log_3(n)$ entonces:

$$T(n) = 3^{2log_3(n)}T(1) + \sum_{j=0}^{log_3(n)-1}(3^j n + 3)$$

Podemos operar $2log_3(n) = log_3(n^2)$

$$T(n) = 3^{log_3(n^2)}T(1) + \sum_{j=0}^{log_3(n)-1} (3^j n + 3)$$
$$T(n) = n^2 T(1) + \sum_{j=0}^{log_3(n)-1} (3^j n + 3)$$
$$T(n) = cn^3 + \sum_{j=0}^{log_3(n)-1} 3^j n + \sum_{j=0}^{log_3(n)-1} 3$$

Solucionado:

$$T(n) = cn^3 + n\frac{3^{log_3(n)} + 1}{3-1} + 3log_3(n) - 3$$
$$T(n) = cn^3 + n\frac{n+1}{2} + 3log_3(n) - 3$$
$$T(n) = O(n^3)$$

3. Estructuras de datos [15 puntos]

1. **(5 puntos)** ¿Cual es la complejidad de las operaciones en una pila? ¿Porque?
   Las operaciones son $O(1)$, ya que las operaciones top y pop, solo requieren modificaciones en la variable $s.top$ y una verificación con la variable $s.size$, para evitar underflow u overflow

2. **(10 puntos)** ¿Cual es la complejidad de las operaciones de búscar, insertar y eliminar en una tabla hash con hashing uniforme con n elementos y m llaves?
   Tomando en cuenta que en Hashing uniforme el factor de carga es $\alpha = \frac{n}{m}$, la complejidad de las operaciones es $O(1 + \alpha)$

4. Computación iterativa [40 puntos]

1. (25 puntos) Para el siguiente algoritmo iterativo:

```
algoritmo2(int a, int n){
    int b = n;
    int c = 3;
    int r = 0;

    while(b >= 0){
       while(c>=0){
           r+=a;
           c--;
       }
       r+=n;
       c = 3;
       b--;
    }
    print(r);

}
```

   a) Indique la forma de estado, estado inicial y estado final

      - Estado (b, r). n no importa ya que en cada iteración siempre toma el mismo valor $c = 3$.
      - Estado inicial $(n, 0)$
      - Estado final $(-1, \sum_{i=0}^{n+1}(4a+n)) = (-1, 4(n+1)+n(n+1))$. Observe que en cada iteración siempre $r = r + 4a + n$ y se cuenta desde -1 hasta n.

   b) Indique la transición de estados
      Para cada estado se tiene

$$(b, r) \to (b-1, r+4a+n)$$

   c) Indique la invariante de ciclo
      Tomando en cuenta la transición de estados, la invariante de ciclo es:

$$(n-i, \sum_{i=0}^{i}(4a+n)))$$

      El indice va hasta $n - i$ tomando en cuenta que el estado va desde n hasta 0

2. (15 puntos) Para el siguiente algoritmo recursivo indique la ecuación de recurrencia y calcule la complejidad el términos de O(n):

```
// n es un numero par, a >= 1
algoritmo3(int n, int a){
    if(n == 1){
        return a;
    }
    else{
        int c = algoritmo4(n,a)
        c + algoritmo3(n/2,a) +  algoritmo3(n/2,a)
    }

}

algoritmo4(int n, int a){
   int i = 0;
   int j = 0;
   int k = 0;
   while(i<n){
       while(j<n){
           while(k<=n){
                 a+=k;
                 k++;
           }
           j++;
       }
       i++;
   }

}
```

La complejidad de este algoritmo se puede analizar:

a) En algoritmo3, siempre toma $O(1)$ más lo que cuesta c = algoritmo4(n,a). Cada vez que se llama algoritmo3 la entrada se transforma $n = \frac{n}{2}$

b) La condición de parada es con $n = 1$ y la complejidad es $O(1)$

c) La complejidad del algoritmo4 es $O(n^3)$

Por lo tanto se plantea la siguiente ecuación de recurrencia:

$$T(n) = 2T(\frac{n}{2}) + O(n^3), T(1) = O(1);$$

Solucionando

$$T(n) = 2T(\frac{n}{2}) + cn^3$$
$$T(n) = 2^2T(\frac{n}{2^2}) + c(\frac{n}{2^1})^3 + c(\frac{n}{2^0})^3$$
$$T(n) = 2^iT(\frac{n}{2^i}) + \sum_{j=0}^{i-1} c(\frac{n}{2^i})^3$$

Se tiene que $T(1) = C$ entonces, $i = log_2(n)$

$$T(n) = 2^{log_2(n)}T(1) + \sum_{j=0}^{log_2(n)-1} c(\frac{n}{2^i})^3$$
$$T(n) = c_1n + \sum_{j=0}^{log_2(n)-1} c(\frac{n^3}{2^{3i}})$$
$$T(n) = c_1n + c_2n^3 \sum_{j=0}^{log_2(n)-1} (\frac{1^i}{2^{3i}})$$
$$T(n) = c_1n + c_2n^3 \sum_{j=0}^{log_2(n)-1} (\frac{1}{2^3})^i$$
$$T(n) = c_1n + c_2n^3 \frac{(\frac{1}{2^3})^{log_2(n)} - 1}{\frac{1}{2^3} - 1}$$
$$T(n) = c_1n + c_2n^3 \frac{(\frac{1}{2})^{log_2(n^3)} - 1}{\frac{1}{2^3} - 1}$$
$$T(n) = c_1n + c_2n^3 \frac{(\frac{1}{2})^{log_2(n^3)} - 1}{-c_3} T(n) = c_1n + c_2n^3 \frac{(\frac{1}{n^3} - 1}{-c_3}$$
$$T(n) = c_1n - c_2 + c_3n^3$$
$$T(n) = O(n^3)$$