20-08-2020

Solución Quiz Nº 2

Análisis de algoritmos

1. a) $\frac{(N^2 + 2N)}{N} = T(N)$

$N + \frac{2}{N} = T(N)$

$N \sim T(N)$

All= la función tilde no es 2N ya que según el procedimiento la función tilde si es de orden lineal ya que es el término que predomina, pero solo sería N

↳ Además $\lim_{N \to \infty} \frac{N+2}{2N} = \boxed{\frac{1}{2}}$ ↓ No da la unidad

b) $T(N) = \log(N^3 \cdot \sqrt[3]{N^2})$

$= \log(N^3 \cdot N^{2/3})$

$= \log(N^{11/3})$

$= 11/3 \log(N) \sim 11/3 \log(N)$

2) Modelo de Costo: Comparaciones entre los elementos del arreglo
Costo Constante

a) $T(N) = \sum_{i=0}^{N-1} \left(\sum_{j=0}^{N+2} 1 \right)$ ← Siempre va a hacer la comparación

$T(N) = \sum_{i=0}^{N-1} (N+1)$

$= (N-1) \cdot N$

$= N^2 - N$ ⇒ Total de Frecuencia

$\sim N^2$ ⇒ Función tilde

Orden de crecimiento N^2

b)

Z	X	W	U	S	M
X	W	U	S	M	Z

N-1

N-2

1

N-1
N-2
N-N

El condicional tiene 2 accesos al arreglo (La condición)
Dentro del condicional se tienen 4 accesos al arreglo

b)

z	x	w	u	s	m

x w u s m z = N-2 ... 1

N-2
N-N

El condicional tiene 2 accesos al arreglo (la condición)
Dentro del condicional se tienen 4 accesos al arreglo

En un peor caso no siempre se va a meter dentro del arreglo (A medida que el ciclo externo se va haciendo ya algunas estarán ordenadas

$$T(N) \leq \sum_{i=1}^{N} \left(\sum_{j=1}^{N-1} (2) + \sum_{j=1}^{N-i} (4) \right)$$

$$\leq \sum_{i=1}^{N} \left(2(N-1) + 4(N-i) \right)$$

$$\leq \sum_{i=1}^{N} (2N - 2 + 4N - 4i)$$

$$\leq \sum_{i=1}^{N} (6N - 4i - 2)$$

$$\leq 6N^2 - 4\,\frac{N(N+1)}{2} - 2N$$

$$\leq 6N^2 - 2N^2 - 2N - 2N$$

$\leq 4N^2 - 4N$ ⇒ Total de Frecuencia

$\sim 4N^2$ ⇒ Función tilde

Orden de crecimiento N^2

Estructura Union-Find (Asumimos que implementamos una estructura Union-Find)

3) a.

```
public int findMat ( int p , int [][] conexiones)
    int n = p;
    for (i=0 ; i < conexiones.length; i++) {
        if ( conexiones[i][0] == n && n != conexiones[i][1]){
            n = conexiones[i][1];
            i = -1;
        }
        else if (conexiones[i][1]== n && n== conexiones[i][2])
            break;
    }
    return n;
}

public boolean redConectada (int [][] conexiones)
```

```
        else if (conexiones[i][1]==n && n== conexiones[i][2])
            break;
        }
        return n;
    }

public boolean redConectada (int [][] conexiones)
{
    int raiz= findMat(0, conexiones)
    boolean conect=true;
    for (i=1; i< id.length; i++){
        if (findMat(i, conexiones)!= raiz)
        {
            conect= false;
            break;
        }
    }
    return boolean;
}

b.
public int numSegmentos (int [][] conexiones)
    boolean a= redConectada(conexiones);
    if ( a== True)
        return 1;
```

```
        else
            return countConexiones(conexiones)
    }

public int countConexiones (int [][] conexiones)
{
    for (i=0; i< conexiones.length; i++){
        union (conexiones[i][0], conexiones[i][1];
    }
    return count();
}
```