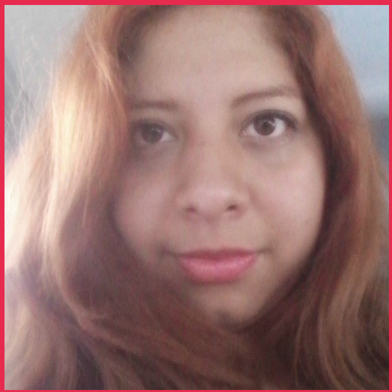
ESCOM-IPN

Ejercicio 4
Algoritmos Recursivos

Análisis de Algoritmos

Laura Andrea Morales López

Mayo 2018



Índice

1. Algoritmo 1

```
int FuncionRecursiva(int num){                    //Función a analizar
    if (num==0)
    {
        return 1;
    }
    else if (num<3)
    {
        resultado=0;
        for (int i = 0; i < num*num; i++)
        {
            resultado*=num;
            return resultado;
        }
    }
    else{
        return FuncionRecursiva(num-1)*FuncionRecursiva(num-2)/FuncionRecursiva(num-3);
    }
}
```

Consideramos los casos base de el algoritmo:

$$T(0) = 1; T(1) = 6; T(2) = 6$$

Dado que en el caso de que $num < 3$ entra al ciclo y cuando hace la multiplicación se sale, por lo tanto no entra más que una vez.

Además consideramos la multiplicación de las recursividades como operación básica.

$$T(n) = 2 + T(n-1) + T(n-2) + T(n-3)$$

Reacomodando los terminos:

$$T(n) - T(n-1) - T(n-2) - T(n-3) = 0^n(2x^0)$$

$$(x^3 - x^2 - x - 1)(x)^3 = 0$$

Con raíces

$$x = 0 \quad x = 1.83 \quad x = -0.41 - 0.6i \quad x = -0.41 + 0.6i$$

Tenemos la siguiente ecuación

$$T(n) = C_1(0)^n + C_2(1.83)^n + C_3(-0.41 - 0.6i)^n + C_4(-0.41 + 0.6i)^n$$

Podemos ver que la complejidad es: $\theta(2^n)$

2. Algoritmo 2

```
int Tribonacci(int num){
    if (num==0)
        return 0;
    else if(num==1||num==2)
        return 1;
    else
        return Tribonacci(num-1)+Tribonacci(num-2)+Tribonacci(num-3);

}
```

Consideramos los casos base de Tribonacci.

$$T(0) = 0; T(1) = 1; T(2) = 1$$

Además consideramos la suma como operaciones básicas

$$T(n) = 2 + T(n-1) + T(n-2) + T(n-3)$$

Reacomodando los terminos:

$$T(n) - T(n-1) - T(n-2) - T(n-3) = 0^n(2x^0)$$

$$(x^3 - x^2 - x - 1)(x)^3 = 0$$

Con raíces

$$x = 0 \quad x = 1.83 \quad x = -0.41 - 0.6i \quad x = -0.41 + 0.6i$$

Tenemos la siguiente ecuación

$$T(n) = C_1(0)^n + C_2(1.83)^n + C_3(-0.41 - 0.6i)^n + C_4(-0.41 + 0.6i)^n$$

Podemos ver que la complejidad es: $\theta(2^n)$

3. Ejercicio 3

Resolver las siguientes ecuaciones y dar su orden de complejidad:

3.1. A

Sea

$$T(n) = 3T(n-1) + 4T(n-2) \rightarrow n > 1; T(0) = 0; T(1) = 1$$

$$T(n) - 3T(n-1) - 4T(n-2) = 0$$

Se transforma a la ecuación característica

$$x^2 - 3x - 4 = 0$$

Con raices $x = -1, x = 4$

$$T(n) = C_1(-1)^n + C_2 4^n$$

De las condiciones iniciales tenemos:

$$C_1 + C_2 = 0 \quad - C_1 + 4C_2 = 1$$

Obteniendo así:
$C_1 = \frac{-1}{5} \quad C_2 = \frac{1}{5}$
Entonces tenemos la complejidad

$$T(n) = \frac{-1}{5}(-1)^n + \frac{1}{5}(4)^n$$

Por lo tanto el orden es $O(4^n)$

3.2. B

Sea:

$$T(n) = 3T(n-1) + 4T(n-2) + (n+5)2^n \rightarrow n > 1; T(0) = 5; T(1) = 27$$

Con las condiciones podemos deducir las siguientes 2 para obtener suficientes ecuaciones para resolver las constantes:

$$T(2) = 3T(1) + 4T(0) + 7(2^2) \rightarrow T(2) = 3(27) + 4(5) + 7(4) \rightarrow T(2) = 129$$

$$T(3) = 3(129) + 4(27) + 8(8) = 559$$

$$T(n) - 3T(n-1) - 4T(n-2) = 2^n(n+5)$$
$$(x^2 - 3x - 4)(x-2)^2 = 0$$

Con raíces

$$x = -1 \quad x = 4 \quad x = 2 \quad x = 2$$

Con esto tenemos la siguiente ecuación:

$$T(n) = C_1(-1)^n + C_2(4)^n + C_3(2)^n + C_4 n(2)^n$$

Se puede notar que la complejidad es de

$n2^n$

3.3. C

Sea:

$$T(n) = 2T(n-1) + (1)3^n \to n > 1; T(0) = 5; T(1) = 1$$

Reacomodamos

$$T(n) - 2T(n-1) = (1)3^n$$
$$(x-2)(x-3) = 0$$

Con raíces

$$x = 2 \qquad x = 3$$

Con esto tenemos la siguiente ecuación:

$$T(n) = C_1(2)^n + C_2(3)^n$$

Tenemos las siguientes ecuaciones:

$$0 = C_1 + C_2$$
$$1 = 2C_1 + 3C_2$$

Y las siguientes constantes:

$$C_1 = -1$$
$$C_2 = 1$$

Vemos facilmente que la complejidad es de $O(3^n)$

4. Algoritmo 4

Calcular la cota de complejidad del algoritmo de búsqueda binaria recursiva

```
int BusquedaBinaria(int num_buscado, int numeros[], int inicio, int centro, int final){
    if(inicio>final)
        return -1;
    else if(num_buscado=numeros[centro])
        return centro;
    else if(num_buscado < numeros[centro])
        return BusquedaBinaria(num_buscado,numeros,inicio,(int)((inicio+centro-1)/2),centro-1);
    else
        return BusquedaBinaria(num_buscado,numeros,centro+1,(int)((final+centro+1)/2),final)
}
```

Podemos ver que su caso base es

$$T(0) = 0$$

$$T(n) = 3 + T(n/2)$$

Entonces usamos el teorema maestro para resolver la cota.

Con este teorems decimos que $a = 1$ $b = 2$ $f(n) = 3$

Entonces:

$n^{log_2 1} = (n^0) = 1$

Con complejidad:

$$T(n) = \theta(logn)$$

5. Algoritmo 5

Calcular la cota de complejidad del algoritmo de Merge Sort recursiva

```
MergeSort(a,p,r){
    if(p<r)
    {
        q=parteEntera((p+r)/2);
        MergeSort(a,p,q);
        MergeSort(a,q+1,r);
        Merge(a,p,q,r);
    }
}
```

El caso base es de:

$$T(1) = 1$$

Con la ecuación;

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

Entonces usamos el teorema maestro para resolver la cota.

Con este teorema decimos que $a = 2\ \ b = 2\ \ f(n) = n$

Entonces:

$n^{log_2 2}{=}(n^1) = n$

Y tenemos la cota:

$$T(n) = \theta(nlogn)$$

6. Ejercicio 6

Calcular la cota de complejidad que tendrían los algoritmos con los siguientes modelos recurrentes.

6.1. A

$$T(n) = 3T(\frac{n}{3}) + 4T(\frac{n}{2}) + 2n^2 + n$$

Para resolver la ecuación la dividiremos y resoveremos usando el teorema maestro.

Para:

$$3T(\frac{n}{3}) + n$$

Con este teorema decimos que $a = 3\ \ b = 3\ \ f(n) = n$

Entonces:

$n^{log_3 3}{=}(n^1) = n$

Y tenemos la cota:

$$T(n) = \theta(nlogn)$$

Para:

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

Con este teorema decimos que $a = 4\ \ b = 2\ \ f(n) = 2n^2$

Entonces:

$n^{log_2 4}{=}(n^2)$

Y tenemos la cota:

$$T(n) = \theta(n^2logn)$$

6.2. B

$$T(n) = T(n-1) + T(n-2) + T(\frac{n}{2}) \rightarrow n > 1 \;\; T(0) = 5 \;\; T(1) = 1$$

Podemos partirlo en 2 partes:

Para

$$T(\frac{n}{2})$$

Con este teorema decimos que $a = 1 \;\; b = 2 \;\; f(n) = 0$

Entonces:

$n^{log_2 1} {=} (n^0) = 1$

Y tenemos la cota:

$$T(n) = \theta(logn)$$

Para:

$$T(n) = T(n-1) + T(n-2) \rightarrow n > 1 \;\; T(0) = 5 \;\; T(1) = 1$$

Reacomodamos

$$T(n) - T(n-1) - T(n-2) = 0$$

$$x^2 - x - 1 = 0$$

Con raíces

$$x = \frac{1}{2}(1 - \sqrt{5}) \qquad x = \frac{1}{2}(1 + \sqrt{5})$$

$$x = -0.61 \qquad x = 1.61$$

Con esto tenemos la siguiente ecuación:

$$T(n) = C_1(-0.61)^n + C_2(1.61)^n$$

Tenemos las siguientes ecuaciones:

$$5 = C_1 + C_2$$

$$1 = C_1(-0.61) + (1.61)C_2$$

Y las siguientes constantes:

$$C_1 = 3.17$$

$$C_2 = 1.82$$

Vemos facilmente que la complejidad es de

$$T(n) = 3.17(-0.61)^n + 1.82(1.61)^n + logn$$

Vemos facilmente que la complejidad es de orden:

$\theta(1.61)^n$ aproximadamente $O(2^n)$

6.3. C

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

Para resolver la ecuación la dividiremos y resoveremos usando el teorema maestro.

Para:

$$T(\frac{n}{2})$$

Con este teorema decimos que $a = 1$ $b = 2$ $f(n) = 2$

Entonces:

$n^{log_2 1} = (n^0) = 1$

Y tenemos la cota:

$$T(n) = \theta(logn)$$

Para:

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

Con este teorema decimos que $a = 2$ $b = 4$ $f(n) = 2$

Entonces:

$n^{log_4 2}$=$(n^{\frac{1}{2}})$

Y tenemos la cota:

$$T(n) = \theta(n^{\frac{1}{2}} logn)$$

Entonces el orden será: $O(n^{\frac{1}{2}} logn)$