ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. 1
Recursión

Mariajose Franco Orozco
Universidad Eafit
Medellín, Colombia
mfrancoo@eafit.edu.co

Susana Álvarez
Universidad Eafit
Medellín, Colombia
salvarezz1@eafit.edu.co

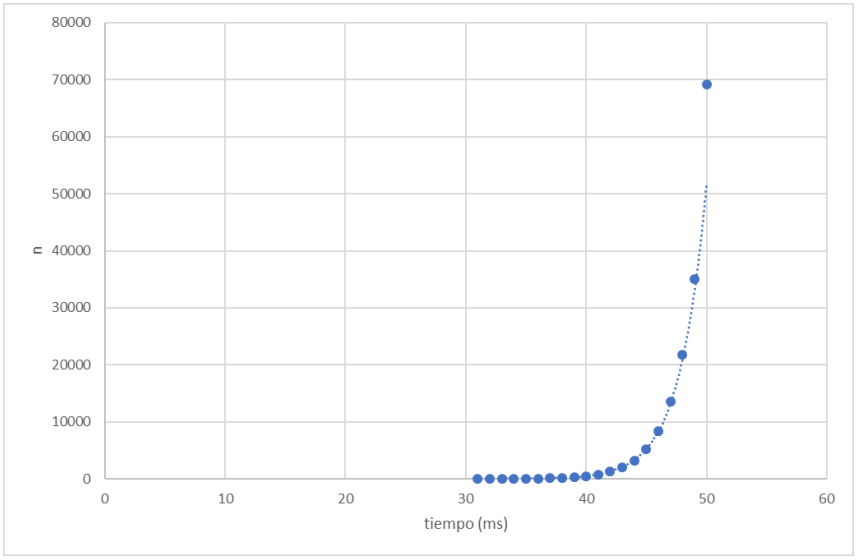
3) Simulacro de preguntas de sustentación de Proyectos

3.1 T(n) = T(n-1) + T(n-2) + C

```
public static int nRectangulos(int n)
 {
    if(n<=2){     //c1
      return n;     //c2
      }
    return nRectangulos(n-1)+nRectangulos(n-2);     //T(n-1)+T(n-2)
 }
```

3.2 Sol:

n	Tiempo(ms)
31	14
32	11
33	25
34	47
35	51
36	84
37	112
38	181
39	289
40	478
41	753
42	1284
43	1987
44	3197
45	5187



PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

46	8337
47	13520
48	21763
49	35096
50	69211

R/ Este algoritmo se demora 69211 milisegundos en calcular las formas que existen de llenar un rectángulo de 50x2 cm² con rectángulos de 1x2 cm²

3.3 No consideramos que sea viable este algoritmo para calcular contenedores que midan miles de centímetros ya que pusimos a correr el programa para 1000 cm y después de 3 horas de espera el algoritmo todavía seguía corriendo, por lo tanto se demoraría demasiado tiempo en calcularlo.

3.4 El ejercicio GroupSum5 recibe 3 parámetros: un entero que indica en qué posición del arreglo se debe comenzar a seleccionar los números, un arreglo de enteros que contiene los números que deseamos sumar, y otro entero que indica la suma que queremos obtener con los elementos del arreglo (el resultado deseado).
Primero se compara la posición para comenzar con la longitud del arreglo ya que si estas son iguales o la posición es mayor a la longitud, el programa va a retornar un falso ya que no se va a poder sumar ningún elemento del arreglo; si la posición es menor que la longitud del arreglo, el programa entra a verificar si el numero en la posición del arreglo es múltiplo de 5, de ser correcto se verifica que el número siguiente a este no sea el numero 1; si este si es 1, se vuelve a repetir el proceso tomando el numero múltiplo de 5 pero sin tomar el número 1. Si este no es 1, se toma el numero que es múltiplo de 5 y se repite el proceso. Si el numero no es múltiplo de 5, el programa verifica si la suma con los números del arreglo que si son números múltiplos de 5 y con los que no lo son se puede obtener el resultado deseado, de ser así, el programa retorna verdadero, si no se puede lograr este resultado, el programa retorna falso.

3.5 Sol:

2.1) Recursión 1

- **Factorial: T(n) = n*T(n-1) + C**

```
public int factorial(int n) {
      if(n==1){      //c1
         return 1;      //c2
      }
      return n*factorial(n-1);      //n*T(n-1)
   }
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

- **SumDigits: T(n) = T(n/10) + C**

```
public int sumDigits(int n) {
    if(n==0){      //c1
        return 0;      //c2
    }
    else{
        int u=n%10;     //c3
        int nu=n/10;      //c4
        return u+sumDigits(nu);       //T(n)=c3+T(n/10)
    }
}
```

- **PowerN: T(n) = T(n-1)*C + C**

```
public int powerN(int base, int n) {
    if(n==0){           //c1
        return 1;         //c2
    }
    return base*powerN(base, n-1);          //T(n-1)*c
}
```

- **Fibonacci: T(n) = T(n-1) + T(n-2) + C**

```
public int fibonacci(int n) {
    if(n<=1){      //c1
        return n;      //c2
    }
    return fibonacci(n-1)+fibonacci(n-2);     //T(n-1) + T(n-2)
}
```

- **Array6: T(n) = T(n-1) + C**

```
public boolean array6(int[] nums, int index) {
    if(index==nums.length){     //c1
        return false;     //c2
    }
    else if(nums[index]==6){     //c3
        return true;      //c4
    }
    else
        return array6(nums,index+1);      //T(n-1)
}
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

2.2) Recursión 2

- **GroupSum6: T(n) = 3T(n-1) + C**

```
public boolean groupSum6(int start, int[] nums, int target) {
    if(start==nums.length){    //c1
        return target==0;     //c2
    }
    if(nums[start]==6){    //c3
        return groupSum6(start+1,nums,target-6);        //T(n-1)
    }
    else{
        return      groupSum6(start+1,nums,target)      ||      groupSum6(start+1,nums,target-
nums[start]);      //2T(n-1)
    }
  }
```

- **GroupSum5: T(n) = 3T(n-1) + T(n-2) + C**

```
public boolean groupSum5(int start, int[] nums, int target) {
    if(start==nums.length){         //c1
        return target==0;          //c2
    }
    if(start+1<nums.length&&nums[start]%5==0){           //c3
        if( nums[start+1]!=1) //c4
            return groupSum5(start+1,nums,target-nums[start]);          //T(n-1)
        else{
            return groupSum5(start+2,nums,target-nums[start]);          //T(n-2)
        }
    }
    else{
        return         groupSum5(start+1,nums,target)||         groupSum5(start+1,nums,target-
nums[start]);          //2T(n-1)
    }
  }
```

- **GroupSumClump: T(n) = 2T(n-1) + 2T(n-2) + C**

```
public boolean groupSumClump(int start, int[] nums, int target) {
    if(start==nums.length){             //c1
        return target==0;              //c2
    }
    if(start+1<nums.length&&nums[start]==nums[start+1]){           //c3
        return                                              groupSumClump(start+2,nums,target-
2*nums[start])||groupSumClump(start+2,nums,target);              //2T(n-2)
    }
    else{
        return  groupSumClump(start+1,nums,target)||  groupSumClump(start+1,nums,target-
nums[start]);          //2T(n-1)
    }
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
    }
```

- GroupNoAdj: T(n) = T(n-1) + T(n-2) + C

```
public boolean groupNoAdj(int start, int[] nums, int target) {
        if(start>=nums.length){          //c1
            return target==0;            //c2
        }

        else{
            return      groupNoAdj(start+1,nums,target)      ||      groupNoAdj(start+2,nums,target-
nums[start]);              //T(n-1)+T(n-2)
        }
    }
```

- SplitArray: T(n) = T(n) + 2T(n-1) + C

```
public boolean splitArray(int[] nums) {
        int i=0;          //c1
        int s1=0;         //c2
        int s2=0;         //c3
        return helper(nums,i,s1,s2);         //T(n)
    }

public boolean helper(int[] nums, int i, int s1,int s2){
        if(i==nums.length){           //c4
            if(s1==s2){              //c5
                return true;          //c6
            }
            else{
                return false;            //c7
            }
        }
        else{
            return helper(nums,i+1,s1+nums[i],s2)|| helper(nums,i+1,s1,s2+nums[i]);      //2T(n-1)
        }
    }
```

3.6 "n" representa la cantidad de valores a la que se le está haciendo la recursión. Depende del método al que nos refiramos la n puede significar la longitud de un arreglo o de un string, etc.

4) Simulacro de Parcial

4.1 *(start+1, nums, target)*
4.2 *T(n) = T(n/2) + C*
4.3 *Sol:*

1) (n-a, a, b, c)

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

2) (res, solucionar(n-b, a, b, c))
3) (res, solucionar(n-c, a, b, c))

4.4 *La suma de los elementos del arreglo a y es O(n)*

4.5 *Sol:*

4.5.1

1) return n
2) n-1
3) n-2

4.5.2

1) T(n) = T(n-1) + T(n-2) + C

4.6 *Sol:*

1) sumaAux(n, i+2)
2) sumaAux(n, i+1)

4.7 *Sol:*

1) *(S, i+1, t-S[i])*
2) *(S, i+1, t)*

4.8 *Sol:*

1) return 0
2) ni + nj

4.9 *22*

4.10 6

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473