ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. 1
Recursión

Juan Andres Henao Diaz
Universidad Eafit
Medellín, Colombia
jahenaod@eafit.edu.co

Carlos Andres Mosquera
Universidad Eafit
Medellín, Colombia
camosquerp@eafit.edu.co

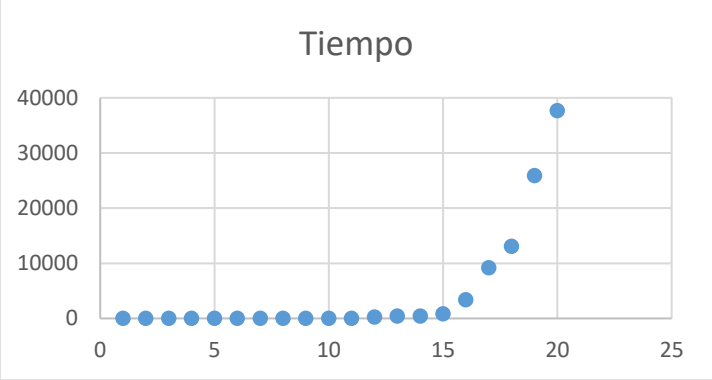
3) Simulacro de preguntas de sustentación de Proyectos

3.1 Complejidad asintótica para el ejercicio 1.1

T (n, m) = C3 + T(n-1, m) + T(n, m-1), no tiene ecuación de recurrencia, no es posible solucionarla

La complejidad de este algoritmo es de O(2n), lo que hace que crezca exponencialmente

3.2



Como podemos observar, la grafica crece exponencialmente a medida que se van presentando los datos, este tipo de graficas muestras una de las complejidades más extensas de desarrollar ya que si tenemos una cantidad de 300.000 datos para este laboratorio, a la hora de ser ejecutado tomara mucho tiempo ya que a medida de que van pasando los datos su tiempo de respuesta crece exponencialmente. En lo que podríamos afirmar que es poco práctico debido a su complejidad y su tiempo de ejecución.

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

3.3 ¿La complejidad del algoritmo del ejercicio 1?1 es apropiada para encontrar la subsecuencia común más larga entre ADNs mitocondriales como los de los datasets?

Como se pudo ver en los puntos anteriores, este algoritmo tiene una complejidad de O(2n) por lo cual lo hace una complejidad exponencial, al cargar tantos datos lo que haría seria un caos ya que demoraría demasiado tiempo en ejecutarse y no seria para nada eficiente implementarlo con los datasets.

3.4 GroupSum5

GroupSum5 tiene tres condiciones. la primera verifica si el número real es múltiplo de cinco y luego si el siguiente número es uno. suma el múltiplo de cinco y omite el uno
La otra condición comprueba si el número real es múltiplo de cinco y si es cierto, suma el múltiplo de cinco y pasa al siguiente número. Y finalmente tiene una condición que verifica si sumando el número real a la suma logrará el objetivo o no sumando y en ambos casos continuar con el siguiente número.

3.5
Recursion- 1 CodingBat

1. BunnyEars

```
public int bunnyEars(int bunnies) {
  if(bunnies  == 0) {                    C1
    return 0;
  }
  return 2 + bunnyEars(bunnies -1 );   T(n) = C1 + T(n-1)
 }
```

$$T(n) = \begin{cases} 0 \text{ if}(n) == 0 \\ 2 + t(n-1) \quad \text{if}(n>=1) \end{cases}$$

Ecuación de recurrencia: T(n) = c1 x n+c1
Calculo de la complejidad: O (c1 x n + c1)
O (c1 x n) por regla de la suma
O (n) por regla de la multiplicación

2. Fibonacci

```
public int fibonacci(int n) {
  if(n <= 1)                              c1
  return n;
```

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

```
  return  fibonacci(n -1)+ fibonacci(n - 2);        T(n) =  T(n-1) + T(n-2)
}
```

$$T(n) = \begin{cases} 1 & \text{if}(n) <= 1 \\ T(n-1) + T(n-2) & \end{cases}$$

Ecuación de recurrencia: T(n) = c1 Fn + c2 Ln
Cálculo de complejidad: O(2^n)

3. count11

```
public int count11(String str) {
  if(str.length() <= 1)                    c1
  return 0;

  if(str.substring(0,2).equals("11"))      c2
  return 1 + count11(str.substring(2));    c2 + T(n + 1)

  return count11(str.substring(1));        T(n – 1)
  }
```

$$T(n) \begin{cases} 1 \text{ if}(str.length() <=1) \\ T(n+1) \\ T(n-1) \end{cases}$$

Ecuación de recurrencia: T(n) = c1 – c2n
Complejidad: O(c1- c2n)
O(n) por regla de diferencia

4. array6

```
public boolean array6(int[] nums, int index) {
  if(index >= nums.length)                 c1
  return false;

  if(nums[index] == 6)                     c2
    return true;

  return array6(nums, index +1);           c1 +  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

T(n) = { False index>= nums.length()
true nums[index] == 6
C1 + T(n+1)

Ecuacion: T(n) = c_1 - c_1 n
Complejidad: O(c1 – c1n)
O(n) ley de la diferencia

5. BunnyEars2

```
public int bunnyEars2(int bunnies) {
if(bunnies == 0)                          c1
return 0;
if(bunnies % 2 == 0)                      C2 * T(n-1) + c3
return bunnyEars2(bunnies - 1) + 3;
else
return bunnyEars2(bunnies - 1) + 2;       C2 * T(n-1) + c4
}
```

T(n) = | { 0 if(bunnies ==0)
C2 * T(n-1) + c3
C2 * T(n-1) + c4
Ecuacion: T(n) = C1 + T(n-1)
Complejidad: T(n) = c1*n * c1
O(n)

Recursion- 2

1. GroupSum6

```
public boolean groupSum6(int start, int[] nums, int target) {
if (start >= nums.length)                          C1
return target == 0;
if (groupSum6(start + 1, nums, target - nums[start]))   C2*T(n-2)
return true;
if (nums[start] == 6)
target -= 6;
return groupSum6(start + 1, nums, target);         C2*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

Ecuación: T(n) = C1(2^n-1) + C1 2^(n-1)
Complejidad: O(2^n) por reglas de suma y multiplicación

2. GroupNoAdj

```
public boolean groupNoAdj(int start, int[] nums, int target) {
if(start >= nums.length)
return target == 0;                                          C1
else
return (groupNoAdj(start + 2, nums, target - nums[start]) ||   C2*T(n - 2)
groupNoAdj(start + 1, nums, target));                        C3*T(n - 1)
}
```

Ecuación: T(n) = C1 + T(n-1) + T(n-2)
Complejidad: O(2^n)

3. SplitOdd10

```
public boolean splitOdd10(int[] nums)
{       return sidesAreOdd10(nums, 0, 0, 0);    }
public boolean sidesAreOdd10(int[] nums, int i, int group1, int group2)
{
if(i == nums.length)
        return (group1 % 2 == 1 && group2 % 10 == 0 || group2 % 2 == 1 && group1 %
10 == 0);
if(sidesAreOdd10(nums, i + 1, group1 + nums[i], group2))
                return true;

return sidesAreOdd10(nums, i + 1, group1, group2 + nums[i]);
}
```

Ecuación: T(n) = C_2 + 2*T(n-1)
Complejidad: O(2^n)

4. SplitArray

```
public boolean splitArray(int[] nums)
{       return sidesAreEqual(nums, 0, 0);       }

public boolean sidesAreEqual(int[] nums, int i, int balance)
{
        if(i == nums.length)
                return (balance == 0);
        if(sidesAreEqual(nums, i + 1, balance + nums[i]))
                return true;
```

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

```
        return sidesAreEqual(nums, i + 1, balance - nums[i]);
}
```

Ecuación: T(n) =c2(2^n – 1) + c1 2^n-1
Complejidad: O(2^n)

5. GroupSumClump

```
public boolean groupSumClump(int start, int[] nums, int target)
{
        if(start >= nums.length)
        {
                if(target == 0)
                return true;
        return false;
        }
        int i = start + 1;
        for(;  i < nums.length && nums[start] == nums[i]; i++);
        if(groupSumClump(i, nums, target - ((i - start) * nums[start])))
                return true;
        return groupSumClump(i, nums, target);
}
```

Ecuación: C1(2^n-1) + C1 2^(n-1)
Complejidad: **O(2^n)**

3.6
N = significa el tamaño del arreglo que recibe cada de los algoritmos

4) Simulacro de Parcial

4.2.1 A.
4.2.2 C.
4.3 A.
4.2.1 A.
4.2.2 A, C.
4.3 B.
4.4.1 C.
4.5.1 A.
4.5.2 B.
4.6 A.
4.7 E.
4.8 B.
4.6.1 sumaAux(n, i + 2)
4.6.2 sumaAux(n, i + 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

4.7.1 comb(S, i + 1, t – S[i])
4.7.2 comb(S, i + 1, t)
4.8 C.
4.9 A.
4.10 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