Taller en sala 4: Notación O Recursiva

Por:

Pablo Alberto Osorio Marulanda

Verónica Mendoza Iguarán

Datos y algoritmos I

Universidad Eafit

2018

RECORDAR QUE EL PROGRAMA SE HA DETENIDO UN SEGUNDO POR ITERACIÓN (PARA PODER TOMAR LOS DATOS)

1. Suma de los elementos de un arreglo

1.1 Código en Word

```
private static int suma(int[] a, int i){
    if (i == a.length)
      return 0;
    else
      return a[i] + suma(a,i+1);
}
```

1.2 Tamaño del problema ("n")

El tamaño de problema (n) son los elementos que faltan por procesar, es decir, las posiciones en el arreglo faltantes.

1.3 Etiquetar cuánto se demora cada línea

```
private static int suma(int[] a, int i){
    if (i == a.length)  // C1
      return 0;  // C2
    else
      return a[i] + suma(a,i+1); // C3 + T(n-1)
}
```

1.4 Ecuación de recurrencia

$$T(n) = \begin{cases} c_1 & \text{if } n = 1 \\ c_2 + T(n-1) & \text{if } n > 1 \end{cases}$$

1.5 Solución de la ecuación con Wolfram Alpha

T(n) = c_2 + T(n-1)

T(n) = c_2*n + c_1

1.6 Notación O

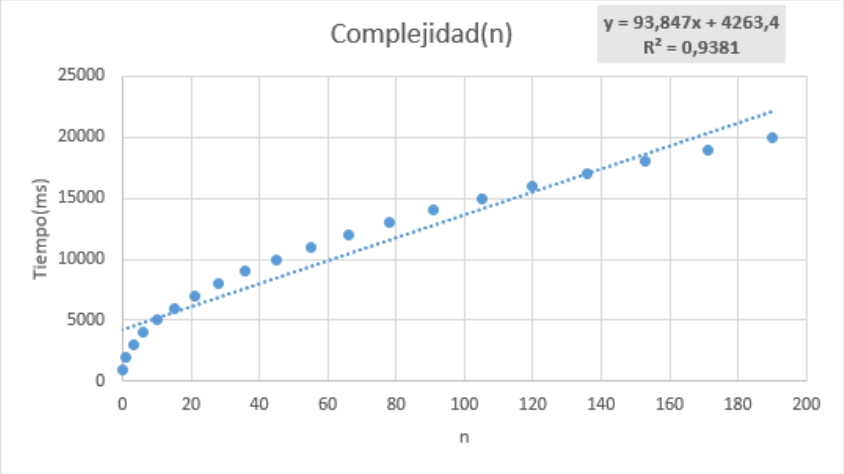
T(n) es O(c_2*n + c_1), por definición de O

T(n) es O(c_2*n), por Regla de la Suma
T(n) es O(n), por Regla del Producto

1.7 Gráfica

Suma(a)	Tiempo(ms)
0	1000
1	2000
3	3001
6	4002
10	5003
15	6001
21	7003
28	8004
36	9004
45	10003
55	11006
66	12004
78	13007
91	14005
105	15005
120	16007
136	17006
153	18010
171	19007
190	20007

Función T(n)
T(n)=c_2*n+c_1



1.8 Explicación en palabras

La complejidad asintótica (es decir, para valores grandes de n) para el peor de los casos (es decir, en el que el algoritmo hace más de una operación) para el algoritmo de sumar los elementos de un arreglo corresponde, aunque no exactamente (con un coeficiente de regresión de 0.94) a lo expresado en la notación O-grande, es decir, pertenece a la familia de curvas acotadas en O(n), tal como se encontró experimentalmente. De esta manera, se puede hacer una aproximación lo que sería el tiempo para n valores de suma.

Cabe resaltar que, aunque la función pareciese logarítmica, el coeficiente de regresión se acerca más a uno cuando se representa por una función lineal.

RECORDAR QUE EL PROGRAMA SE HA DETENIDO UN SEGUNDO POR ITERACIÓN (PARA PODER TOMAR LOS DATOS)

2. SumaGrupo:

2.1 Código en Word

```
public static boolean sumaGrupo(int start, int[] nums, int target) {
   if (start >= nums.length){
      return target == 0;
   }
   if (sumaGrupo(start+1, nums, target - nums[start])){
      return true;
   }
   else   if ( sumaGrupo(start+1, nums, target)){
      return true;
   }  else {
      return false;
   }
}
```

2.2 Tamaño del problema (“n”)

El tamaño n del problema está dado por las posiciones n que faltan por ser sumadas, es decir, aquellas que aún no han sido procesadas.

2.3 Etiquetar cuánto se demora cada línea

```
public static boolean sumaGrupo(int start, int[] nums, int target) {

   if (start >= nums.length){ //C1

      return target == 0; //C2

   }

   if (sumaGrupo(start+1, nums, target - nums[start])){ // T(n-1)

      return true; //C4

   }

   else   if ( sumaGrupo(start+1, nums, target)){  // T(n-1)
```

```
        return true;  //C5
    } else {
Return sumaGrupo(start+1, nums, target - nums[start]) || sumaGrupo(start+1, nums, target)
        return false; //C6
    }
  }
```

2.4 Ecuación de recurrencia

$$T(n) = \begin{cases} c_1 & \text{if } n = 0 \\ T(n-1) + T(n-1) + c_2 & \text{if } n > 0 \end{cases}$$

2.5 Solución la ecuación con Wolfram Alpha

T(n) = T(n-1) + T(n-1) + c_2

T(n) = 2*T(n-1) + c_2

T(n) = c_1*2^(n-1) + c_2(2^n-1)

2.6 Notación O

T(n) es O(c_1*2^(n-1) + c_2*(2^n-1)), por definición de O

T(n) es O(2^(n-1) + 2^n-1), por Regla del Producto

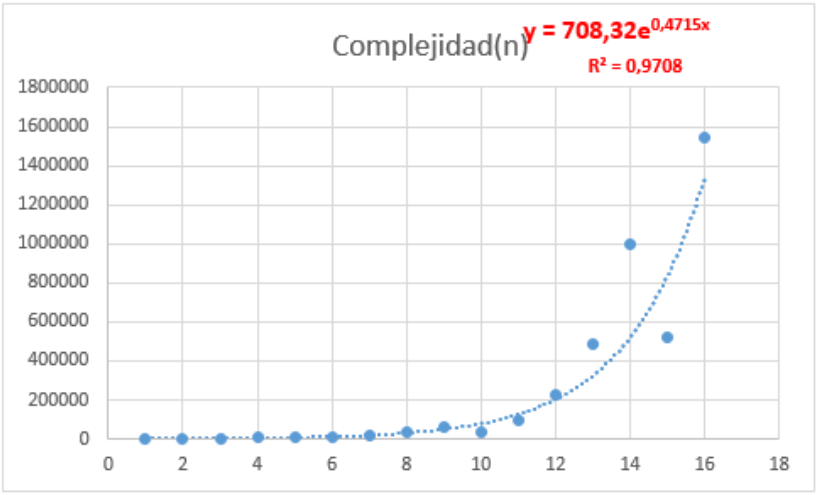
T(n) es O(2^n-1), por Regla de la suma

T(n) es O(2^n), por Regla del Producto

2.7 Gráfica

n	Tiempo(ms)
1	1017
2	3128
3	3002
4	5002
5	9009
6	7002
7	15041
8	31015
9	63086
10	36013
11	100149
12	228993
13	486684
14	997449
15	517925
16	1542325
17	3590704
18	7710816

Función T(n)
T(n) = c_1*2^(n-1) + c_2(2^n-1)



2.8 Explicación en palabras

La teoría asintótica nos arroja que la complejidad de tal problema está definida por una función exponencial, es decir, tal función pertenece al orden de curvas contenidas en O=(2^n). Tal como se puede ver en la gráfica, los datos obtenidos experimentalmente se adaptan a tal predicción. De esta manera, con la función allí representada se puede predecir el tiempo aproximado que se demorará un arreglo de n tamaños en organizarse respecto al algoritmo sumGroup.

3. Fibonacci:

3.1 Código en Word

```
public static int fibonacci(int n){
    if(n==0||n==1){
    return n;
 }else{
```

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

3.2 Tamaño del problema (“n”)

El tamaño del problema (“n”), para este caso se refiere a los elementos (números) que faltan por procesar.

3.3 Etiquetar cuánto se demora cada línea

```
  public static int fibonacci(int n){
        if(n==0||n==1){  //C1
        return n;  //C2
    }else{
        return fibonacci(n-1)+fibonacci(n-2);  //C3 + T(n-1) + T(n-2)
    }
  }
```

3.4 Ecuación de recurrencia

$$T(n) = \begin{cases} c_1 & \text{if } 0 <= n <= 1 \\ c_2 + T(n-1) + T(n-2) & \text{if } n > 1 \end{cases}$$

3.5 Solución de la ecuación con Wolfram Alpha

T(n) = c_2 + T(n-1) + T(n-2)

T(n) = (2^n-1) + (2^n-2) + c_2

3.6 Notación O

T(n) es O((2^n-1) + (2^n-2) + c_2), por definición de O

T(n) es O(2^(n-1)) + O(2^(n-2)), por Regla de la suma

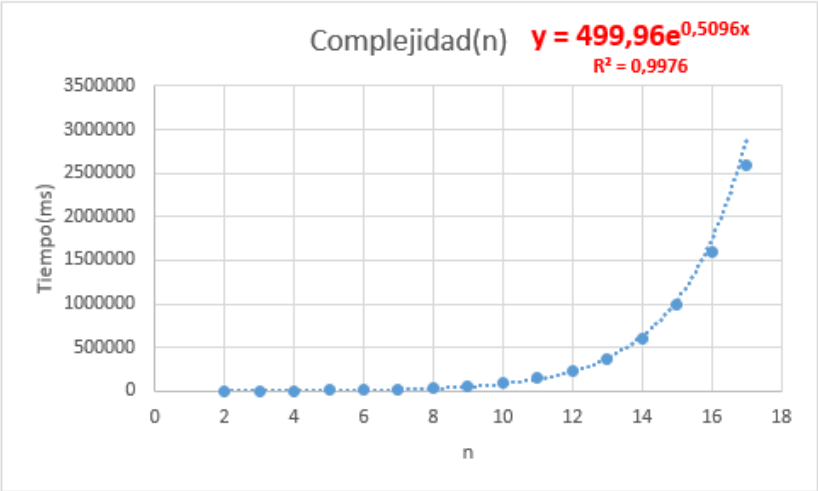
T(n) es O(2^(n-1)), por Regla de la suma

T(n) es O(2^n), por Regla del Producto

3.7 Gráfica

n	Tiempo
0	0
1	0
2	1016
3	2001
4	4001
5	7003
6	12005
7	20002
8	33006
9	54011
10	88015
11	143015
12	232039
13	376065
14	609080
15	987260
16	1596926
17	2584010

Función T(n)
T(n) = (2^n-1) + (2^n-2) + c_2



3.8 Explicación en palabras

La teoría asintótica nos arroja que la complejidad de Fibonacci está definida por una función exponencial. Esto último lo podemos ver en los datos experimentales tomados en el gráfico. De esta manera comprobamos que, a medida que crece **n**, la cantidad de tiempo que se demora es el doble de lo que era en **n-1**. La definimos de esta manera gracias a el dato resultante en la notación Big-O, por medio de la cual determinamos la familia de curvas a la que pertenece la función.