Laboratorio Nro. 1
Recursión y complejidad

Julián Gómez Benítez
Universidad Eafit
Medellín, Colombia
jgomezb11@eafit.edu.co

Juan Pablo Rincon Usma
Universidad Eafit
Medellín, Colombia
jprinconu@eafit.edu.co

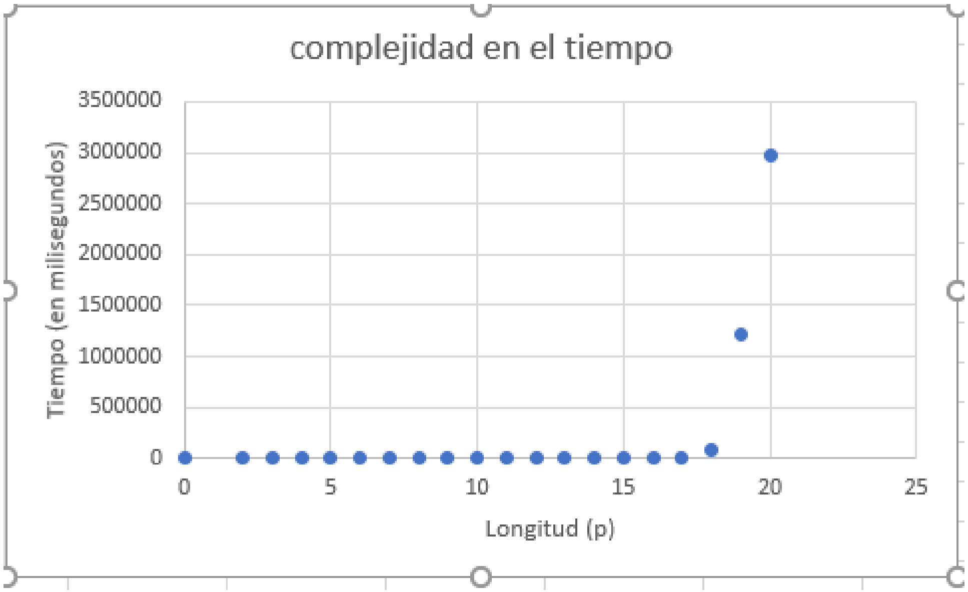
3) Simulacro de preguntas de sustentación de Proyectos

3.1 Siendo n la longitud de la primera cadena, m la longitud de la segunda cadena y p la suma de las dos anteriores (p = n + m), tenemos que T(p) = c_3 + T(p-1) + T(p-1).
Ingresando esto a Wolfram Alpha nos da: T(p) = c_3 (2^p - 1) + c_1 2^(p - 1). Sabemos que eso es igual a O(c_3 (2^p - 1) + c_1 2^(p - 1)) y después de aplicar la regla de la suma y la regla del producto nos queda en O(2^p)

3.2 Los valores de p que tomamos y su respectivo tiempo fueron:

Longitud (p)	Tiempo (ms)
0	0
2	0
3	0
4	0
5	0
6	1
7	2
8	3
9	9
10	101
11	149
12	130
13	151
14	428
15	2642
16	4323
17	13509
18	89590
19	1223498
20	2962477

La gráfica que generamos fue:



Si estimamos el tiempo que nos debería tomar el implementar el algoritmo en cadenas de 300.000 caracteres cada una sería una estimación de 2^{600000} milisegundos.

3.3 No, debido a que según complejidad (exponencial = O(2^p)) tomaría el doble de procesos por cada elemento que le sumemos a p, es decir, tomaría una gran cantidad de procesos ejecutar el método en cadenas de caracteres largas como las de los datasets (que contienen 300.000 caracteres cada una). Como planteamos en **3.2**, 2^{600000} ms no es un tiempo razonable para el tipo de tarea que buscamos. Además de que no es un número de procesos viable para un procesador promedio.

3.5 Recursión 1

- Factorial = T(n) = c_2 + T(n-1)
- bunnyEars = T(n) = c_3 + 2 + T(n-1)
- fibonacci = T(n) = T(n-1) + T(n-2)
- bunnyEars2 = T(n) = T(n-1) + c_1
- triangle = T(n) = c_2 + T(n-1)

Recursión 2

- GroupSum6 = T(n) = 2 T(n-1) + c_2
- GroupNoAdj = T(n) = 2 T(n-1) + c_2
- Split53 = T(n) = 2 T(n-1) + c_2
- GroupSumClump = T(n) = 2 T(n-1) + c_2
- SplitArray = T(n) = 2 T(n-1) + c_2

3.6

Recursión 1

- Para factorial, la n representa el valor que se va a encontrar de la factorial.
- Para bunnyEars, la n representa la cantidad de conejos que se van a calcular.
- Para fibonacci, la n representa el n-ésimo numero de la sucesión fibonacci.
- Para buunyEars2, la n representa la cantidad de conejos que se van a calcular.
- Para triangle, la n representa la cantidad de filas donde se colocan los bloques.

Recursión 2

- Para groupSum6, la n representa el tamaño del arreglo de enteros.
- Para GroupNoAdj, la n representa el tamaño del arreglo de enteros.
- Para split53: n representa el tamaño del arreglo de enteros.
- Para GroupSumClump: n representa el tamaño del arreglo de enteros.
- Para SplitArray: n representa el tamaño del arreglo de enteros.

4) Simulacro de Parcial

4.1

4.1.1 a
4.1.2 c
4.1.3 a

4.2

4.2.1 Falso
4.2.2 a Verdadera **b** Verdadera **c** Falso **d** Verdadera

4.3 b

4.4 lucas(n-1) + lucas (n-2)

4.4.1 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

4.5
- **4.5.1** a
- **4.5.2** b

4.6 (OPC) b
4.7 (OPC) e

4.8 (OPC) if(T == 0) return 1;
If (T < 0) return 0;
.
.
.
return f1+f2+f3

4.5.1 a
4.6
- **4.6.1** return 0;
- **4.6.2** return (n.charAt(i) - ‘0’) + sumaAux(n, i + 1) ;

4.7
- **4.7.1** comb(S, i + 1, t - S[i]) ||
- **4.7.2** comb(S, i , t);

4.8 c
4.9 (OPC) b
4.10 lucas(n-1) + lucas (n-2)
- **4.11.1** 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