

Laboratorio Nro. 1 Recursión

María Antonia Velásquez
Universidad Eafit
Medellín, Colombia
mavelasqur@eafit.edu.co

Diego Alexander Múnera Tobón
Universidad Eafit
Medellín, Colombia
damunerat@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 Se tiene que **m** es el número de caracteres del String1 y **n** es el número de caracteres del String2, luego **p** es la suma de estas dos longitudes, así $p = m + n$.

Complejidad del ejercicio:

$$T(p) = c_3 + T(p-1) + T(p-1)$$

$$T(p) = c_3(2^{p-1}) + c_1 * 2^{p-1}$$

$$T(p) = (c_3 + c_1/2) * 2^p - c_3$$

$$T(p) = O((c_3 + c_1/2) * 2^p - c_3)$$

$$O((c_3 + c_1/2) * 2^p)$$

$$O(2^p)$$

T(n) es $O(2^p)$ lo que en resumidas cuentas es una complejidad de tipo exponencial (en el peor de los casos).

3.2 Según la complejidad del algoritmo encontrada, la subsecuencia en común más larga entre dos cadenas de ADN mitocondrial con 300.000 caracteres cada una, es de 2^P donde **P** se reconoce como la suma de los caracteres de ambas cadenas. En otras palabras, $p = m + n$ donde $m = 300.000$ y $n = 300.000$, así pues, teniendo a $p = 600.000$, el tiempo tardado en encontrar dada subsecuencia es de $2^{600.000}$.

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



ESTRUCTURA DE DATOS 1

Código ST0245

$P(m+n)$	$O(2^p)$
6	0
8	0
10	0
12	0
14	0
16	1
18	1
20	1
22	4
24	20
26	38
28	139
30	588
32	1796
34	4974
36	18149
38	70040
40	275072
42	1401184
44	3180947



3.3

Según las recomendaciones y el análisis, este algoritmo no es recomendado para la ejecución con cadenas muy largas debido a su tiempo exponencial a partir de dado n , se tomaría mucho tiempo realizar tareas como el cálculo para cuando la magnitud de cada cadena de caracteres es de 20, se deben hacer muchas operaciones y tomaría un tiempo bastante grande. Y si tenemos en cuenta el ejemplo de arriba sobre la magnitud de 300.000 caracteres en cada cadena tomaría una increíble cantidad de tiempo (aproximadamente 3 meses).

3.5

Complejidad Algoritmos Recursivos Parte 1

BunnyEars2:

- $T(n) = c1 + t(n-1)$
- $T(n) = c3 * n + c1$
- $T(n) = c3 * n$
- $T(n) = n$
- Complejidad: $O(n)$

BunnyEars:

- $T(n) = c1$
- $T(n) = c2 * T(n-1)$
- $T(n) = c2 * n + c1$
- $T(n) = n$
- Complejidad: $O(n)$

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

ESTRUCTURA DE DATOS 1
Código ST0245

Triangle:

- $T(n) = c1 + t(n-1)$
- $T(n) = c3 \cdot n + c1$
- $T(n) = c3 \cdot n$
- $T(n) = n$
- Complejidad: $O(n)$

Factorial:

- $T(n) = c1$
- $T(n) = c2 + T(n-1)$
- $T(n) = c2 \cdot n + c1$
- $T(n) = n$
- Complejidad: $O(n)$

Fibonacci:

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

Complejidad Algoritmos Recursivos Parte 2

SplitOdd10

- $T(n) = c2 + T(n-1)$
- $T(n) = c2 \cdot (2^n - 1) + c1 \cdot 2^{n-1}$
- $T(n) = c2 \cdot (2^n - 1) + c1 \cdot 2^{(n-1)}$
- $T(n) = c2 \cdot (2^n - 1)$
- $T(n) = (2^n - 1)$
- $T(n) = (2^n)$
- Complejidad $O(2^n)$

groupSum6

- $T(n) = c1 + t(n-1) + t(n-1)$
- $T(n) = c1 \cdot (2^n - 1) + c1 \cdot 2^{(n-1)}$
- $T(n) = (2^n - 1) + 2^{(n-1)}$
- $T(n) = (2^n + 2^n)$
- $T(n) = 2(2^n)$
- $T(n) = (2^n)$
- Complejidad $O(2^n)$

split53

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

ESTRUCTURA DE DATOS 1
Código ST0245

- $T(n) = c1 + t(n-1) + t(n-1)$
- $T(n) = c1 * (2^n - 1) + c1 * 2^{(n-1)}$
- $T(n) = (2^n - 1) + 2^{(n-1)}$
- $T(n) = (2^n + 2^n)$
- $T(n) = 2(2^n)$
- $T(n) = (2^n)$
- Complejidad $O(2^n)$

groupNoAdj

- $T(n) = c1 + t(n-2) + t(n-1)$
- $T(n) = (2^n)$
- Complejidad $O(2^n)$

splitArray

- $T(n) = c2 + t(n-1) + t(n-1)$
- $T(n) = c2 * (2^n - 1) + c1 * 2^{(n-1)}$
- $T(n) = (c2 * (2^n - 1) + c1 * 2^{(n-1)})$
- $T(n) = c2 * (2^n - 1)$
- $T(n) = (2^n - 1)$
- $T(n) = (2^n)$
- Complejidad $O(2^n)$

3.6

Las variables n, m son los vectores o cadenas de caracteres encargadas de proporcionar datos suficientes para el reconocimiento de p como la suma de los caracteres de ambas cadenas. En otras palabras, $p = m + n$ dónde en el peor caso experimental $m = 300.000$ y $n = 300.000$ según los datasets.

4) Simulacro de Parcial

4.1

1. a. $s.substring(0, i)$
2. c. true
3. a. solve(t, s.substring(i), n - i)

4.3

- b. $T(n, m) = C \times n \times m^2$

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

ESTRUCTURA DE DATOS 1
Código ST0245

4.4

```
1 int lucas(int n){  
2 if(n == 0) return 2;  
3 if(n == 1) return 1;  
4 return lucas(n-1) + (n-2);  
5 }
```

1. Complejidad:
c. $T(n)=T(n-1)+T(n-2)+c$, que es $O(2^n)$

4.5

1. a. true
2. b. `s.charAt(0) == (s.charAt(s.length()-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

