ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. X
Escribir el tema del laboratorio

Simón Correa Henao
Universidad Eafit
Medellín, Colombia
scorreah@eafit.edu.co

David Gómez Correa
Universidad Eafit
Medellín, Colombia
dgomezc10@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 Complejidad asintótica

Considerando la ecuación de recurrencia:

$$T(n,m) = \begin{cases} c1, \text{ donde } c1 = 5 \\ c2 + T(n-1,m-1) \\ c3 + T(n-1,m) + T(n,m-1) \end{cases}$$

Dado que no se puede calcular (con Wolfram). Podemos considerar una nueva variable, además para facilitar el cálculo:

$$p = n + m$$

Con lo cual la ecuación de recurrencia quedaría:

$$T(p) = \begin{cases} c\_1, \text{ donde } c1 = 5 \\ c\_2 + T(p-2) \\ c\_3 + T(p-1) + T(p-1) \end{cases}$$

Solucionando la ecuación

$$T(p) = c\_3\,(2^p - 1) + c\_1\,2^{(p - 1)}$$

Aplicando las propiedades de O

$$O(p) = c\_3\,(2^p - 1) + c\_1\,2^{(p - 1)}$$

$$O(p) = c\_3 * 2^p - c\_3 + c\_1 ½ * 2^p$$

$$O(p) = c\_3 * 2^p + c\_1 ½ * 2^p \quad \text{(por regla de la suma)}$$

$$O(p) = 2^p(c\_3 + c\_1 * ½)$$

$$O(p) = 2^p \quad \text{(por regla del producto)}$$

3.2

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

Podemos observar que los tiempos arrojados de la ejecución del algoritmo, con 20 diferentes casos con tamaños de n y m entre 1 y 20. Claramente la ecuación más cercana que agrupa los datos arrojados es una exponencial, dejando muy en claro que el algoritmo presentado para este ejercicio consumen mucho tiempo.

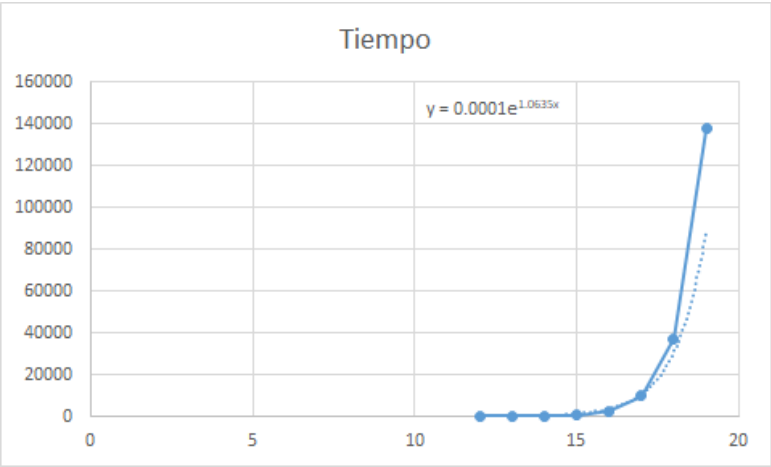
Además, el utilizar el algoritmo para evaluar dos cadenas mitocondriales de ADN de alrededor de 300.000 caracteres tomaría un tiempo muy prolongado, por lo cual sería ineficiente. Calculando el tiempo estimado con la fórmula proporcionada por Excel, quedaría:

$$T(seg) = 0.0001e^{1,0635*300000}$$

Es importante mencionar que ninguna calculadora fue capaz de arrojar algún resultado

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?

La complejidad hallada a partir del algoritmo del ejercicio 1.1 es como se observa en los datos experimentales, particularmente ineficiente a la hora de trabajar con la cantidad de datos contenidos en los datasets. Esto mismo ya se puede deducir del cálculo de complejidad ya que, en este caso en concreto, la complejidad en el tiempo crece exponencialmente al ritmo de los datos, por lo cual resultaría inapropiado para resolver el problema de calcular la cadena común más larga para 2 cadenas de ADN de 300mil caracteres.



3.4

3.5 Condingbat: Recursion 1

Triangle: T(n) = T(n-1) + c_2
En notación O: O(n)

Fibonacci: T(n) = c_3 + T(n-1) + T(n-2)
En notación O: O(2^n)

CountHi: T(n) = c_4 n + c_1
En notación O: O(n)

powerN: T(n) = c_2 + T(n-1)
En notación O: O(n)

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

endX: T(n) = c_2 + T(n-1)
En notación O: O(n)

Condingbat: Recursion 2

GroupSum5: T(n) = c_4 + 2T(n-1)
En notación O: $O(2^n)$

GroupSum6: T(n) = c_3 + 2T(n-1)
En notación O: $O(2^n)$

splitArray: T(n) = c_2 + 2T(n-1)
En notación O: $O(2^n)$

Split53: T(n) = c_4 + 2T(n-1)
En notación O: $O(2^n)$

groupNoAdj: T(n) = T(n) = -c_2 + c_1 F_n + c_2 L_n
En notación O: $O(2^n)$

3.6

Triangle: La variable n en el cálculo de complejidad representa el número de filas que tiene dicho triangulo, c_2 nos habla de que existen dos constantes dentro del algoritmo y O(n) que es un problema con consumo de tiempo lineal

Fibonacci: La variable n en el cálculo representa el número de Fibonacci a calcular, c_3 nos habla de que existen 3 constantes dentro del algoritmo; además podemos observar que existen dos llamados de recursividad, por lo cual la complejidad se torna $O(2^n)$ es decir una complejidad exponencial.

CountHi: La variable n en el cálculo de complejidad representa el tamaño del String, c_4 nos habla de que existen cuatro constantes dentro del algoritmo y O(n) que es un problema con consumo de tiempo lineal

PowerN: La variable n en el cálculo de complejidad representa el número de veces que se va a multiplicar por si mismo el número base, c_2 nos habla de que existen dos constantes dentro del algoritmo y O(n) que es un problema con consumo de tiempo lineal

EndX: La variable n en el cálculo de complejidad representa el tamaño del String, c_2 nos habla de que existen dos constantes dentro del algoritmo y O(n) que es un problema con consumo de tiempo lineal

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

GroupSum5: La variable n en el cálculo, representa el tamaño del arreglo de int, c_4 nos habla de que existen 4 constantes dentro del algoritmo; además podemos observar que existen dos casos de recurrencia, por lo cual la complejidad se torna $O(2^n)$ es decir una complejidad exponencial.

GroupSum6: La variable n en el cálculo, representa el tamaño del arreglo de int, c_3 nos habla de que existen 3 constantes dentro del algoritmo; además podemos observar que existen dos casos de recurrencia, por lo cual la complejidad se torna $O(2^n)$ es decir una complejidad exponencial.

SplitArray: La variable n en el cálculo, representa el tamaño del arreglo de int, c_2 nos habla de que existen 2 constantes dentro del algoritmo; además podemos observar que existen dos casos de recurrencia, por lo cual la complejidad se torna $O(2^n)$ es decir una complejidad exponencial.

Slipt53: La variable n en el cálculo, representa el tamaño del arreglo de int, c_4 nos habla de que existen 4 constantes dentro del algoritmo; además podemos observar que existen dos casos de recurrencia, por lo cual la complejidad se torna $O(2^n)$ es decir una complejidad exponencial.

GroupNoAdj: La variable n en el cálculo, representa el tamaño del arreglo de int, c_2 nos habla de que existen 2 constantes dentro del algoritmo; además podemos observar que existen dos casos de recurrencia, uno con el número de Fibonacci y otro con el número de Lucas, por lo cual la complejidad se torna $O(2^n)$ es decir una complejidad exponencial.

4) Simulacro de Parcial

Respuestas:

4.1 1. A) , 2. C), 3. A)

4.2 *1. B), 2. A) y C)*

4.3 *B)*

4.4 C)

4.5 1. A), 2. B)

4.6 1. 0 , 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) (Tel: (+57) (4) 261 95 00 Ext. 9473

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