Laboratorio 1

Estructuras de datos y algoritmos 1

Tomas Marin Aristizábal

Colombia

Eafit

tmarina@eafit.edu.co

Juan Andrés Vera

Colombia

Eafit

javeraa@eafit.edu.co

3.

3.1 Para el punto 1.1 tenemos el siguiente código:

```
if (m==0 || n==0){
    return 0;
}
if(string1.charAt(m-1) == string2.charAt(n-1)){
    return 1 + lcsDNAAux(string1, string2, m-1, n-1);
}else{
    return Math.max(lcsDNAAux(string1, string2, m, n-1), lcsDNAAux(string1, string2, m-1, n));
}
```

De donde podemos sacar 2 formulas las cuales serían:

l = n + m

T(l) = C_1 + 2T(l-2) que sería T(l) = 2^(l/2) (c_2 (-1)^l + c_1) - C_1 y

T(l) = C_2 + 2T(l-1) que sería T(l) = C^2 (2^l - 1) + c_1 2^ (l - 1)

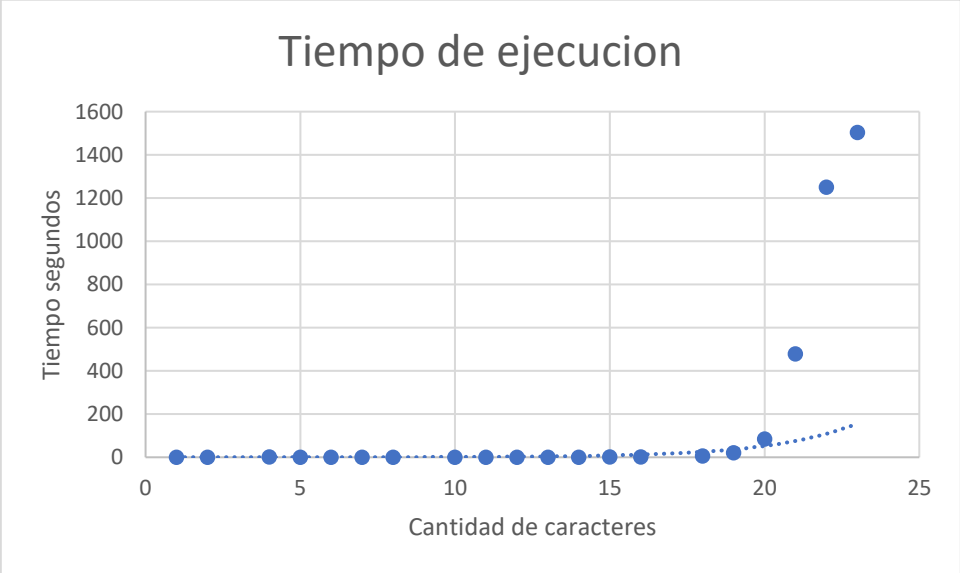
La ecuación más compleja seria T(l) seria:

T(l) = C^2 (2^l - 1) + c_1 2^ (l - 1)

Que sería una ecuación **O** (2^n)

3.2 Tomando 20 valores de cadenas de varios tamaños podemos ver que sus tiempos de ejecución (en segundos) fueron los siguientes:

El tiempo para n = 1 fue de 0.614s
El tiempo para n = 2 fue de 0,658 s
El tiempo para n = 4 fue de 0,98s
El tiempo para n = 5 fue de 0,663s
El tiempo para n = 6 fue de 0,886s
El tiempo para n = 7 fue de 0,631s
El tiempo para n = 8 fue de 0,94s
El tiempo para n = 10 fue de 0,773s
El tiempo para n = 11 fue de 0,687s
El tiempo para n = 12 fue de 0,915s
El tiempo para n = 13 fue de 0,918s
El tiempo para n = 14 fue de 0,846s
El tiempo para n = 15 fue de 0,993s
El tiempo para n = 16 fue de 1,369s
El tiempo para n = 18 fue de 5,992s
El tiempo para n = 19 fue de 21,562s
El tiempo para n = 20 fue de 84,608s
El tiempo para n = 21 fue de 478,228s
El tiempo para n = 22 fue de 1250,794s
El tiempo para n = 23 fue de 1504,392s



3.3 No es adecuado para realizar la secuencia de los ADN mitocondriales ya que estas presentan más de 300.000 caracteres cada cadena, por ende, la cantidad de operaciones que debe realizar es demasiado grande ya que es una ecuación O (2^n) con lo cual es una función que duplica su complejidad a medida que aumentan sus caracteres, con lo cual podría llegar a tardar días o más.

3.4 opcional

3.5

Recursión 1.

En wólfram alfa

1. T(n) = T(n) = c_2 n + c_1
2. T(n)= c_2 + n*2 +c_1
3. T(n) = -c_4 + c_1 F_n + c_2 L_n
4. T(n) = c_3 n + c_1
5. T(n) = c_2*n + c_1

Recursión 2.

1. T(n) = (c2 + c1/2)2^n - c2
2. T(n)= -c2+c1*Fn+c2Ln
3. T(n)= c4*n + c1
4. T(n)= c4*n + c1
5. T(n)= c2((2^n)-1)+c1*2^n-1

3.6

1.1 la m y n significan el tamaño del arreglo (. length) que sale de los Strings string1 y string2

Codingbat

Factorial: n representa el a que valor a aplicarle la factorial.
bunnyEars: n representa la cantidad de conejos.
Fibonaccci: n es el n-esimo numero de Fibonacci.
bunnyEasrs2: n representa la cantidad de conejos.
Triangle: n representa las filas del triángulo
groupSum: n representa el tamaño del arreglo
groupNoAdj: n representa el tamaño del arreglo.
groupSum5: n representa el tamaño del arreglo.
groupSumClump: n representa el tamaño del arreglo.
splitArray: n representa el tamaño del arreglo.

4. Preparación Parcial

4.1.1. A

4.1.2. C

4.1.3. A

4.2.1 B

4.2.2 a. Verdadera
b. falsa

c. falsa
d. verdadera

4.3.1 **B**

4.4.1 **línea 4: return Lucas(n-1) + Lucas(n-2);**

Respuesta es C

4.5.1 **A**

4.5.2 **B**

4.6.1 linea 10 : sumaAux(n,i+2);

4.6.2 linea 12 : sumaAux(n,i+1);