Laboratorio Nro. 1
Recursión y Complejidad Asintótica

Stiven Yepes
Universidad Eafit
Medellín, Colombia
esyepesv@eafit.edu.co

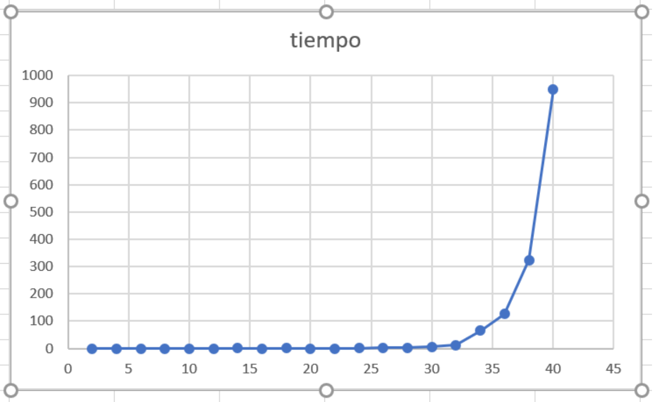
Sara Rodríguez
Universidad Eafit
Medellín, Colombia
srodriguev@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 T(n)= T(n-1) + T(n-2) + c

3.2

iteracion	tiempo
2	0
4	0
6	0
8	0
10	0
12	0
14	1
16	0
18	1
20	0
22	0
24	1
26	2
28	3
30	6
32	13
34	66
36	127
38	324
40	948



- Podemos ver que los datos siguen una tendencia exponencial, la cual aproximadamente a partir de n=38 tiene un cambio más brusco con respecto a los datos anteriores, lo que nos dice que a partir de ese punto el tiempo seguirá aumentando cada vez más con esta misma tendencia. Igualmente, podemos concluir por la naturaleza de la gráfica, que este es un algoritmo que puede ser de orden n^3 o n^2.

- Corriendo el programa, el resultado para n=50 fue: <u>137148 ms.</u>

3.3

No, la complejidad de este algoritmo hace casi imposible aplicarlo a números muy altos como los que se emplearían en Puerto Antioquia, ya que el tiempo de ejecución de estos seria exorbitante, poco eficiente e incluso imposible para maquinas normales.

3.5
Recursion-1:

- PowerN: T(n)= T(n-1) + c, donde n es la potencia a la que se va a elevar la base.
- CountX: T(n) = T(n-1) + T(n-1) + c, donde n es la longitud de la nueva cadena de caracteres(str.substring) creada a partir de la cadena original(str).
- ChangePi: T(n) = T(n-2) + T(n-1) + c, donde n es la longitud de la nueva cadena de caracteres(str.substring) creada a partir de la cadena original(str).
- AllStar: T(n)= T(n-1) + c, donde n es la longitud de la nueva cadena de caracteres(str.substring) creada a partir de la cadena original(str).
- EndX: T(n) = T(n-2) + T(n-1) + c, donde n es la longitud de la nueva cadena de caracteres(str.substring) creada a partir de la cadena original(str).

Recursion-2:

- GroupNoAdj: T(n) = T(n-2) + T(n-1) + c, donde n es lo que le falta a start para acabar el arreglo.
- SplitArray: T(n) = T(n-1) + T(n-1) + c, donde n es la cantidad de números en el arreglo(nums).
- GroupSum6: T(n) = T(n-1) + T(n-1) + T(n-1) + c, donde n es lo que le falta a start para acabar el arreglo.
- GroupSumClump: T(n) = n + T(n-1) + T(n-1) + c, donde n es lo que le falta a start para acabar el arreglo.
- IsSideOdd: T(n) = T(n-1) + T(n-1) + c, donde n es lo que le falta a i para acabar el arreglo.

4) Simulacro de Parcial

4.1
Línea 3: true;
Línea 4: s.charAt(0) == s.charAt(s.length()-1)

4.2
a) T(n)=T(n/2)+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

4.3
4.3.1 Linea 4: n-a,a,b,c
4.3.2 Linea 5: res,solucionar(n-b,a,b,c)+1
4.3.3 Linea 6: res,solucionar(n-c,a,b,c)+1

4.4 (Opc)
e) La suma de los elementos del arreglo a y es O(n)

4.5
4.5.1
Línea 3: if(T==0) return 1;
Línea 4: if (T<=1) return 0;
Línea 8: return f1+f2+f3;

4.5.2
a. T(n)=T(n−1)+C

4.6
4.6.1. Linea 10: sumaAux(n.substring(2+i),i++);
4.6.2. Linea 12: sumaAux(n.substring(i+1),i++);

4.7 (Opc)

4.8
4.8.1 Linea 9: return 0;
4.8.2 Linea 13: ni+nj

4.9 (Opc)
c. 22

4.10
b. 6

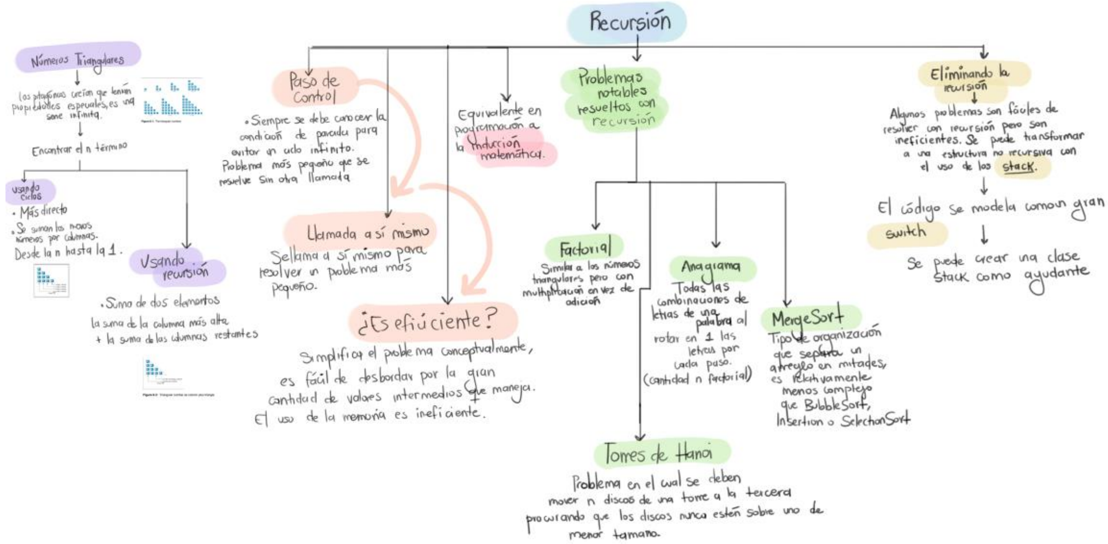
4.11
4.11.1. Línea 4: n – 1 , lucas(n - 2);
4.11.2 T(n)=T(n−1)+T(n−2)+c, que es O(2^n)

4.12
4.12.1 Línea 13: sat
4.12.2 Línea 17: Math.max(fi , fj)
4.12.3 Línea 18: sat

5) Lectura recomendada (opcional)

Mapa conceptual:

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



Disponible en tamaño real:
https://eafit-my.sharepoint.com/:i:/g/personal/srodriguev_eafit_edu_co/EeXyicE4ZllCm1SlKx6nSEEBBalA5UhUpArPEIdcZE5gzQ?e=N6Nx9K

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