Laboratorio Nro. 1
Complejidades
- Recursión

Julian Andres Ramirez Jimenez
Universidad Eafit
Medellín, Colombia
jaramirezj@eafit.edu.co

Samuel David Villegas Bedoya
Universidad Eafit
Medellín, Colombia
sdvillegab@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1

T(n,m) = {
C1, si i =0 o j==0
C2 + T (n-1, m-1), si S1 [i-1] = S2[j-1]
C3 + T (n, m-1)+T(n-1,m), sino
}

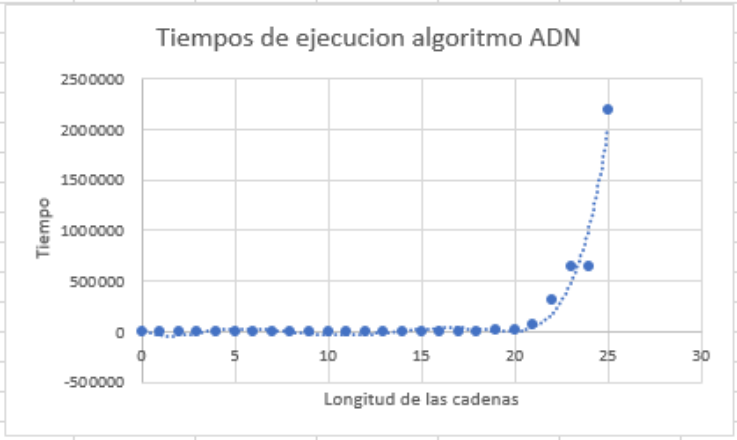
Al poner la ecuación T(n,m) = C3 + T (n, m-1)+T(n-1,m) en la herramienta Wolfram este nos arroja la siguiente ecuación algo extraña:

$$c_3 = -T(n - 1, m) - T(n, m - 1) + T(n, m)$$

3.2

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

1	0
2	0
3	0
4	16
5	0
6	3
7	0
8	0
9	0
10	0
11	0
12	104
13	76
14	68
15	64
16	447
17	1395
18	2359
19	7304
20	9821
21	73535
22	304345
23	638303
24	643196
25	2184442



3.3 No, consideramos que, al ver el comportamiento en tiempo del algoritmo presentado, este tendría un comportamiento Exponencial, por lo cual para cadenas muy grandes este algoritmo tomaría mucho tiempo en evaluar los inputs.
3.4 (No)
3.5

Recursión 1:

allStar: T(n) = {
Caso 1: c_1
Caso 2: c_2 + T(n-1)
Caso 3: c_3 + T(n-1)
}
Peor de los casos: T(n) = c_3 + T(n-1) => T(n) = c_3*n +c

countPairs: T(n) = {
Caso 1: c_1
Caso 2: c_2 + T(n-1)
Caso 3: c_3 + T(n-1)
}
Peor de los casos: T(n) = c_3 + T(n-1) => T(n) = c_3*n +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

stringClean: T(n) = {
Caso 1: c_1
Caso 2: c_2 + T(n-1)
Caso 3: c_3 + T(n-1)
}
Peor de los casos: T(n) = c_3 + T(n-1) => T(n) = c_3*n +c

nestParen: T(n) = {
Caso 1: c_1
Caso 2: c_2
Caso 3: c_3 + T(n-1)
Caso 4: c_4
}
Peor de los casos: T(n) = c_3 + T(n-1) => T(n) = c_3*n +c

changeXY: T(n) = {
Caso 1: c_1
Caso 2: c_2 + T(n-1)
Caso 3: c_3 + T(n-1)
}
Peor de los casos: T(n) = c_3 + T(n-1) => T(n) = c_3*n +c

Recursión 2

splitOdd10: T(n) = {
Caso 1: c_1
Caso 2: c_2 + 2T(n-1)
}
Peor de los casos: T(n) = c_2 + 2T(n-1) => T(n) = c_2 (2^n - 1) + c_1 2^(n - 1)

groupSum6: T(n) = {
Caso 1: c_1
Caso 2: c_2+T(n-1)
Caso 3: c_3 + 2T(n-1)
}
Peor de los casos: T(n) = c_3 + 2T(n-1) => T(n) = c_3 (2^n - 1) + c_1 2^(n - 1)

groupNoAdj: T(n) = {
Caso 1: c_1
Caso 2: c_2+2T(n-1) + 2T(n-2)
}
Peor de los casos: T(n) = c_2+2T(n-1) + 2T(n-2)=> T(n) =-c_2/3 + c_1 (1 - sqrt(3))^n + c_2 (1 + sqrt(3))^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

splitArray: T(n) = {
Caso1: c_1
Caso2: c_2 + 2T(n-1)
}
Peor de los casos: T(n) = c_2 + 2T(n-1) => c_2 (2^n - 1) + c_1 2^(n – 1)

split53: T(n) = {
Caso1: c_1
Caso2: c_2 + T(n-1)
Caso3: c_3 + T(n-1)
Caso4: c_4 + 2T(n-1)
}
Peor de los casos: T(n) = c_4 + 2T(n-1) => c_4 (2^n - 1) + c_1 2^(n – 1)

3.6 n se puede ver como la longitud de un arreglo y la longitud de una cadena, y m como la longitud de una segunda cadena y el número de columnas de una matriz.

4) Simulacro de Parcial

4.1.1 c
4.1.2 c
4.1.3 a
4.2.1 a
4.2.2 a, c.
4.3 b
4.4 return Lucas(n-1) + Lucas (n-2);
4.4.1 c
4.5.1 a
4.5.2 b
4.6.1 return sumAux(n,i+1);
4.6.2 sumAux(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) (4) 261 95 00 Ext. 9473