ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 1: Recursion

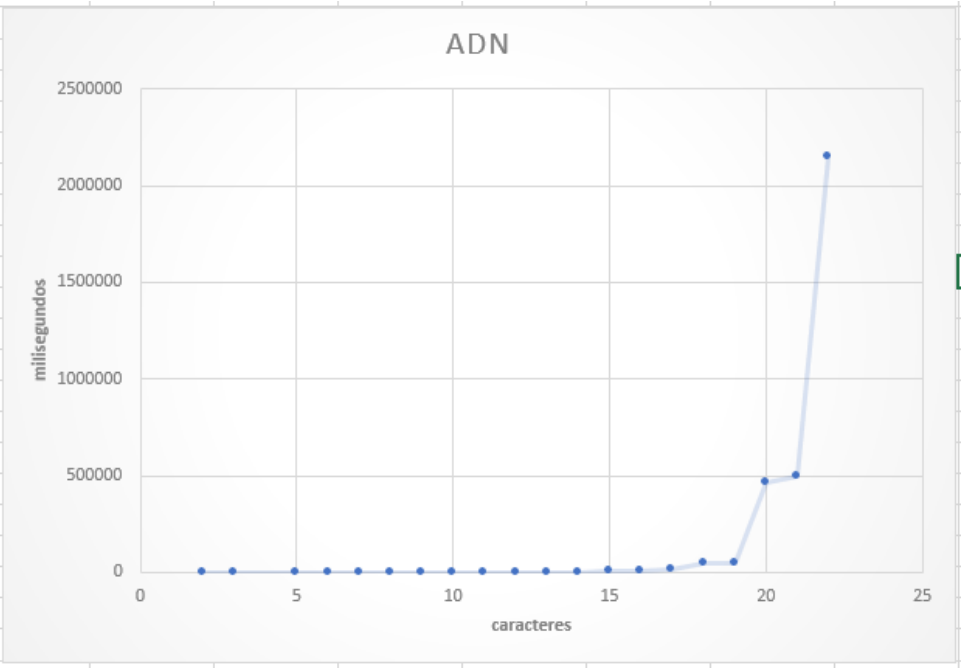
Brigith Lorena Giraldo Vargas
Universidad Eafit
Medellín, Colombia
blgiraldov@eafit.edu.co

Luisa Fernanda Ciro Restrepo
Universidad Eafit
Medellín, Colombia
lfciror@eafit.edu.co

3) Practice for final project defense presentation

3.1 O(4^n)
3.2

caracteres	ms
2	0
3	1
5	0
6	1
7	0
8	1
9	1
10	1
11	113
12	342
13	719
14	710
15	3593
16	10755
17	16456
18	49435
19	48325
20	463238
21	500485
22	2151158



The estimated time that the algorithm takes on the longest common subsequence between two mitochondrial DNAs which have approximately 300.000 chars each, is about 275869,5652 minutes.

3.3 The algorithm isn't optimum for the given datasets, since it takes a lot of time with the measure of medium size values, even though it works correctly, it's not efficient for big data as mitochondrial DNA.

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

3.4 The groupsum5 method tests the truthfulness or falsity of the possibility to do a sum from some or all the numbers from an integer array, so that the sum is equal to a given target. The method starts taking the values from the first position to start the sum, but taking notice of two restrictions: the first one it's that it must take all the multiples of 5, and the second one is to skip the number that follows the multiple if that number is 1.

3.5

- stringClean: O(n)
- bunnyEars: O(n)
- count11: O(n)
- array11: O(n)
- array6: O(n)
- groupSum6: O(2^n)
- groupSum5: O(2^n)
- split53: O(2^n)
- splitOdd10: O(2^n)
- groupNoAdj: O(2^n)

3.6

- stringClean n = intake chain
- bunnyEars n = number of rabbits
- count11 n = intake number
- array11 n = array size
- array6 n = array size
- groupSum6 n = array size
- groupSum5 n = array size
- split53 n = array size
- splitOdd10 n = array size
- groupNoAdj n = number of rows

4) Practice for midterms

4.1

4.1.1 a. s.substring(0, i)
4.1.2 c. true
4.1.3 a. solve(t, s.substring(i), n - i)

4.2 the complete lines are:

9. floodFillUtil(screen, x+1, y+1, prevC, newC, N, M);
10. floodFillUtil(screen, x+1, y-1, prevC, newC, N, M);
11. floodFillUtil(screen, x-1, y+1, prevC, newC, N, M);

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

12. floodFillUtil(screen, x-1, y-1, prevC, newC, N, M);
T(p) = p

4.3 b. T(n,m) = C x n x m2

4.4the complete and right line would be: return lucas(n-2) + lucas(n-1);

4.1.1 c. T(n) = T(n−1) + T(n−2) + c, which is O(2^n)

4.5

4.5.1 a. true

4.5.2 a. s.substring(0,s.length()-1).equals(s.substring(s.length()-1, 0))

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473