Laboratory practice No. 1: Recursion and Complexity

Juan David Echeverri Villada
Universidad Eafit
Medellín, Colombia
jdecheverv@eafit.edu.co

Octavio Vásquez Zapata
Universidad Eafit
Medellín, Colombia
ovasquezz@eafit.edu.co

3) Practice for final project defense presentation

3.1 Complexity of point 1.1 (ADN)

t(n) = t(n-1) + t(n-1)
t(n) = 2t(n-1)
t(n) = c1 * 2^n-1
O(c1 * 2^n-1) by O's definition
O(2^n-1) by sum's rule
O(2^n) by sum's rule

3.2

3.3 Is the complexity of the point 1.1 convenient to find the longest common subsequence between mitochondrial ADNs as the ones in the datasets?

No, definitively no, due the fact that the complexity of the algorithm 1.1(and) is O(n^2), one of the highest complexity kinds. For that reason, using that algorithm in datasets as big as the ones provides would consume a lot of memory and could take a quite long time to finish its execution. For that reason, development this algorithm using recursion and using it with big datasets is not actually very convenient

3.4 GroupSum5

GroupSum5 verify if is possible to reach a target sum with the numbers of one given array but with two conditions. Every multiple of 5 must be included and if the value next to a 5 is 1, it must be ignored.

3.5 Complexity of the codingBat exercises

Recursion 1

Factorial
t(n) = t(n-1) +c2
t(n) = c2*n + c1
O(c2*n + c1) by O's definition
O(c2*n) by sum's rule
O(n) by product's rule

BunnyEars
t(n) = (n-1) + c2
t(n) = c2*n + c1
O(c2*n + c1) by O's definition
O(c2*n) by sum's rule
O(n) by product's rule

Fibonacci
t(n) = t(n-1) + t(n-2)
t(n) = 2^n
O(2^n) O's definition

BunnyEars2
t t(n) = (n-1) + c2
t(n) = c2*n + c1
O(c2*n + c1) by O's definition
O(c2*n) by sum's rule
O(n) by product's rule

Triangle
t(n)= c2 + t(n-1)
t(n) = c2n + c1
O(c2*n + c1) by O's definition
O(c2*n) by sum's rule
O(n) by product's rule

Recursion 2

GroupSum6
t(n) = t(n-1) + t(n-1)
t(n) = 2t(n-1)
t(n) = c1 * 2^n-1
O(c1 * 2^n-1) by O's definition
O(2^n-1) by sum's rule
O(2^n) by sum's rule

GroupSumClump
t(n)= t(n-1) + t(n-1)
t(n)= 2t(n-1)
t(n)= c1 * 2^n-1

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

O(c1 * 2^n-1) by O’s definition
O(2^n-1) by sum’s rule
O(2^n) by sum’s rule

GroupSum5
t(n)= t(n-1) + t(n-1)
t(n) = 2t(n-1)
t(n) = c1 * 2^n-1
O(c1 * 2^n-1) by O’s definition
O(2^n-1) by sum’s rule
O(2^n) by sum’s rule

GroupNoAdj
t(n) = t(n-1) + t(n-2)
t(n) = 2^n
O(2^n) O’s definition

SplitArray
t(n)= t(n-1) + t(n-1)
t(n) = 2t(n-1)
t(n) = c1 * 2^n-1
O(c1 * 2^n-1) by O’s definition
O(2^n-1) by sum’s rule
O(2^n) by sum’s rule

3.6 Variables ‘n’ and ‘m’

When calculating an algorithm complexity, the variable ‘n’ means the size of the dataset that is going to be used. When the complexity of the algorithm depends on two different parts of the datasets, the variable ‘n’ is usually used to represent the first part and the ‘m’ the second one

4) Practice for midterms

4.1 *a*
4.2 *b*
4.3 *length-1*

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