Laboratory practice Nro. 1
Recursion

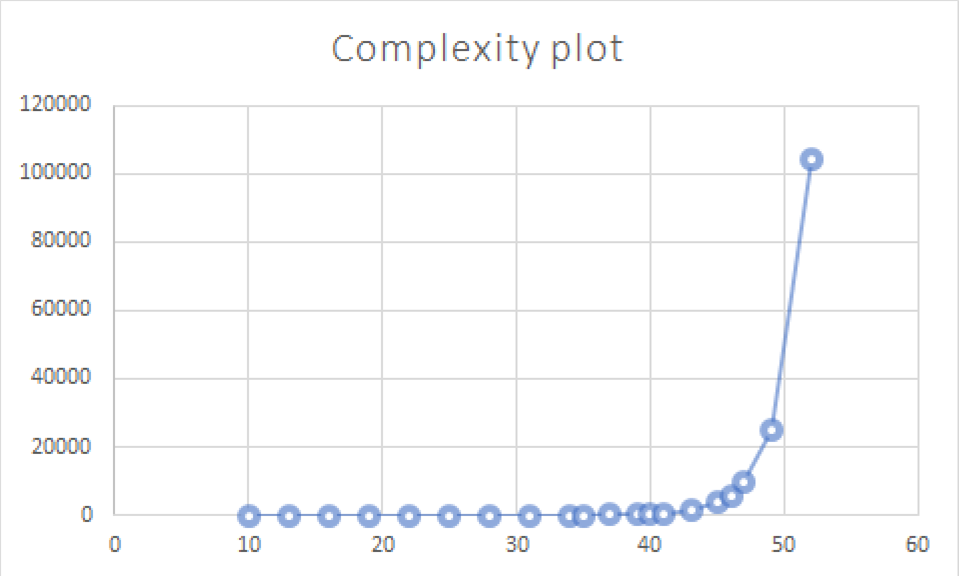
Juan Sebastián Díaz Osorio
Universidad Eafit
Medellín, Colombia
jsdiazo@eafit.edu.co

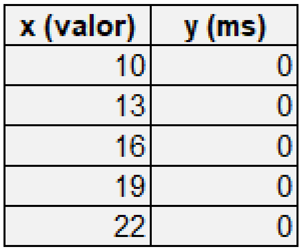
Liz Oriana Rodrigues Cruz
Universidad Eafit
Medellín, Colombia
lorodriguc@eafit.edu.co

3) Practice for final project defense presentation

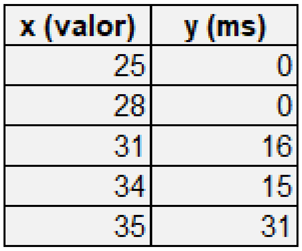
3.1 Let n be second dimension to area problem (It means 1 x n is total area), so: T(n) = T(n-1) + T(n-2) + C, is O(2^n)

3.2

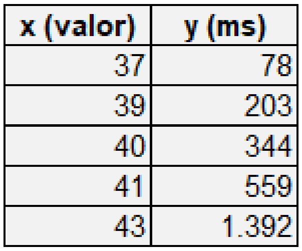




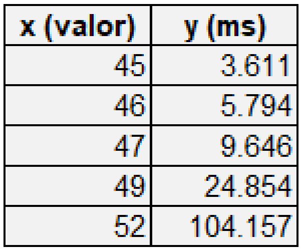
x (valor)	y (ms)
10	0
13	0
16	0
19	0
22	0



x (valor)	y (ms)
25	0
28	0
31	16
34	15
35	31



x (valor)	y (ms)
37	78
39	203
40	344
41	559
43	1.392



x (valor)	y (ms)
45	3.611
46	5.794
47	9.646
49	24.854
52	104.157

3.3 Its complexity is O(2^n). Therefore, the algorithm is so bad with big n numbers and Puerto Antioquia should not depend on this (For clarity, with n = 1000 we talk about 10^{301} processes before the answer).

3.4 *GroupSum5* algorithm, given an initial integer value (commonly 0), an array and a target value, sum each number in each sub-array from the initial array. For instance, [3, 4, 5], [3, 5], [3, 4], [4, 5], [3], [4], [5] are sub-arrays of [3, 4, 5]. Its sums are 14, 8, 7, 9, 3, 4, 5 respectively. There is a condition, though. Whether the sub-array has a 5 or a multiple of 5, sub-array with this value are valid only if next number, in the array, is not 1. For instance, [3, 15, 1, 4], [8, 15, 4], [3, 15] are sub-arrays from [3, 8, 15, 1, 4], but they are not valid because a multiple of 5 has forward an 1. GroupSum5 returns true if exists, at least, a sub-array with this condition that its sum is target value. Otherwise, returns false.

Examples:

➢ **GroupSum5(init = 0, array = [2, 5, 10, 4], target = 19)**

[2, 5, 10, 4] = 21 X
[2, 5, 10] = 17 X
[2, 5, 4] = 11 X
[5, 10, 4] = 19 V
returns **true**

➢ **GroupSum5(init = 0, array = [3, 5, 1], target = 9)**

[3, 5, 1] NO VALID
[3, 5] NO VALID
[5, 1] NO VALID
[3, 1] X
return **false**

3.5 CodingBat - Recursion 1

strCount: Let n be the str length and m the sub length:

$T(n, m) = T(n - m, m) + C_2$

$T(n) = T(n- km) + kC_2$ *(Through induction)*

$T(n) = T(n - (n/m)\ m) + (n/m)\ C_2$ *(Let k = n/m)*

$T(n) = T(0) + (n/m)\ C_2$

$(T(n) = C_1 + (n/m)\ C_2)$

It is **O(n/m).**

strCopies: Let n be the times that a substring is inside a string:

$T(n) = T(n - 1) + C_2$ *(Worst case is to search by whole string)*

$(T(n) = C_2n + C_1)$

It is **O(n).**

strDist: Let n be the string length:

$T(n) = T(n - 1) + C_2$ *(Worst case is to cut each character from string)*

$(T(n) = C_2n + C_1)$

It is **O(n).**

parenBit: Let n be the string length:

$T(n) = T(n - 1) + C_2$ *(Worst case is to cut each character from string)*

$(T(n) = C_2n + C_1)$

It is **O(n).**

nestParen: Let n be the string length:

$T(n) = T(n - 2) + C_3$ *(Worst case is to cut each character from string)*

$(T(n) = \frac{1}{4}C_3(2n - 1) + C_2(-1)^n + C_1)$

It is **O(n).**

CodingBat - Recursion 2

groupSum6: Let n be the array length:

$T(n) = 2T(n - 1) + C_2$ *(Worst case is to use two options in the return)*

$T(n) = C_2(2^n - 1) + C_1(2^{n - 1})$

It is **$O(2^n)$.**

groupNoAdj: Let n be the array length:

$T(n) = T(n - 1) + T(n - 2) + C$

It is **$O(2^n)$.** *(via analysis)*

groupSumClump: Let n be the array length:

$T(n) = n + 2T(n - 1) + C$ *(Worst case is to use two options in the return)*

$T(n) = 2^{n-1}(C_1 + 2C + 4) - C - n - 2$

It is **$O(2^n)$.**

splitOdd10: Let n be the array length:

$T(n) = 2T(n - 1) + C$

$T(n) = C_2(2^n - 1) + C_1(2^{n - 1})$

It is **$O(2^n)$.**

splitArray: Let n be the array length:

$T(n) = 2T(n - 1) + C$

$T(n) = C_2(2^n - 1) + C_1(2^{n - 1})$

It is **$O(2^n)$.**

3.6 CodingBat - Recursion 1

strCount

- ***n*** is a string of characters
- ***m*** is a substring of characters

strCopies

- ***n*** is the number of times a substring of characters is repeated *strDist*

- ○ ***n*** is the length of a character string

parenBit

- ○ ***n*** is the length of a character string

nestParen

- ○ ***n*** is the length of a character string

CodingBat - Recursion 2

groupSum6

- ○ ***n*** is the length of an array of numbers

groupNoAdj

- ○ ***n*** is the length of an array of numbers

groupSumClump

- ○ ***n*** is the length of an array of numbers

aplitOdd10

- ○ ***n*** is the length of an array of numbers

splitArray

- ○ ***n*** is the length of an array of numbers

4) Practice for midterms

4.1 SumaGrupo(start + 1, nums, target);

4.2 a)

4.3 Answers:

4.3.1 int res = solucionar (n – a, a, b, c);

4.3.2 res = Math.max(res, solucionar(n, b, c, n + 1));

4.3.3 res = Math.max(res, solucionar(n, c, n + 1, n + 1));

4.4 a)

4.5 Answers:

4.5.1 line 2 : return n; }

line 3 and 4: return (desconocido (n - 1) + desconocido (n - 2));

4.5.2 d)

4.6 line 10: return sumaAux(n, i + 2);

line 12: return (n.charAt (i) - ‘0’) + sumaAux(n, i + 1);

4.7 line 9 and 10: return comb (S, I + 1, t - S[i]) || comb (S, I + 1, t);

4.8 Answers:

4.8.1 line 9: return 0;

4.8.2 line 13: int suma = ni + nj;

4.9 b)

4.10 a)

4.11 Answers:

4.11.1 return fun(n - 2) + fun(n - 1);

4.11.2 c)

5) Recommended Reading (optional)

We made the conceptual map via Prezi:

https://prezi.com/go7xtgmnkm9h/?utm_campaign=share&utm_medium=copy

6) Teamwork and gradual progress (optional)

We meet once only. This is the record: https://bit.ly/31Gb4qo

It was because we use in-class time, Trello and chats to distributed and analyze the work.

This is the progress report with GitHub commits and Kanban board screenshots

https://bit.ly/2ZbNYX6