Laboratory practice No. 1: Recursion

Isabella Quintero Villegas
Universidad Eafit
Medellín, Colombia
iquinterov@eafit.edu.co

Sofia Vega Lopez
Universidad Eafit
Medellín, Colombia
svegal@eafit.edu.co

1) Practice for Project(References)

1.1

For this point, we decided to use an auxiliary private method for calculating the length of the longest common sequence between two characters string received as parameters in the public method, the auxiliary method receives, besides the two strings, the length of these two, then it evaluates if either one is an empty string, if it is the case it will return cero, meaning there is no common sequence between them, if it is not the case, it will assess if there are characters in common in position length-1 of each string, in which case the final return value will increase in one, finally, the method evaluates each and every character of string #1 with string #2 and vice versa, and it will return the maximum value between these two actions.

*Based on explanation and source code of:

Kumar.S(No date). Java Program for Longest Common Subsequence. https://bit.ly/2ZcT49e

1.2

In this point, the method calculates how many ways there are for organizing rectangles of 1x2 in a rectangle of 2xn, it works like this: first, if n equals to 0,1 or 2 it will return n, if n is different of these three values, it will return the sum between the number 1 position before n and the number two positions before n, this way, it will always come to rectangles of1x2 and 2x2, in which case we already know the result.

*Toro.M(2019) Recursion (Version 1.0). Explained in class.

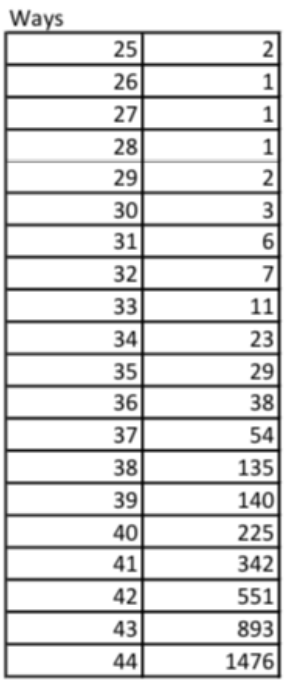
3) Practice for final project defense presentation

3.1
```
public static int ways (int n) {
      if(n<=2) {            //C1
            return n;       //C2
      }
      return ways(n-1) + ways(n-2);    // C3+ T(n-1) + T(n-2)
   }
```

T(n) = C1+ C2
T(n) = C3+ T(n-1) + T(n-2)

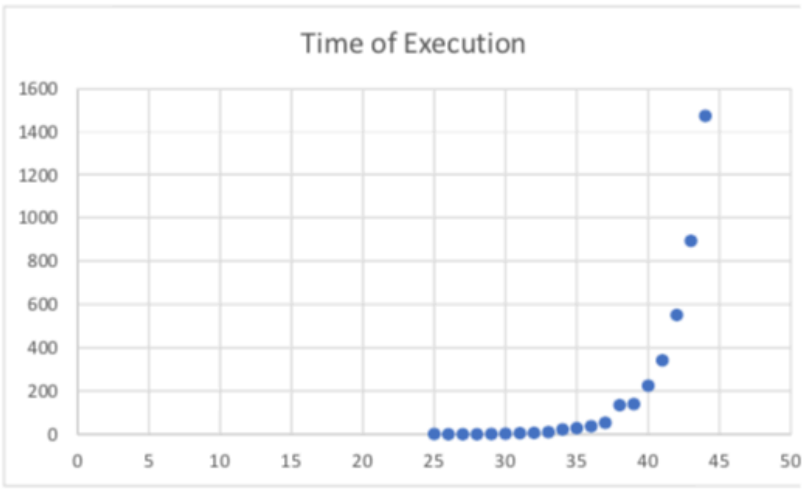
Recurrence Equation (Calculated with Wolfram Alpha)
T(n) = C*2^n + C'
In Big O notation
O(2^n) (After applicate addition and product rules)

3.2



Ways

25	2
26	1
27	1
28	1
29	2
30	3
31	6
32	7
33	11
34	23
35	29
36	38
37	54
38	135
39	140
40	225
41	342
42	551
43	893
44	1476



We estimate it will take around 26000 milliseconds to execute, when n = 50;

3.3
Whit this particular method, it wouldn't de the best choice to use it in Puerto Antioquia, first, because of the parameter, we would need larger data type than int, beside, because of the size that the function occupies (O(2^n)), it will take much more memory and time than we would like to optimize the job.

3.4 How does groupSum5 algorithm work?

```
public boolean groupSum5(int start, int[] nums, int target) {
  if(start==nums.length)return target==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

```
  if(nums[start]%5==0){
    if(start<nums.length-1 && nums[start+1] == 1){
      return groupSum5(start+2,nums,target-nums[start]);
    }
    target -= nums[start];
  }
  return groupSum5(start+1,nums,target-nums[start]) ||
              groupSum5(start+1,nums,target);
}
```

This recursive algorithm calculates if it is possible to choose a group of some of the int from an array, such that the group sums to the given target, with the additional constraint that every five present in the array, must be add to the target, and if there is a 1, right after a five, this must not be chosen.
Well, first the method evaluates if the number in the position “start” is multiple of five, if it is, then it asses if the number next to the multiple it’s one, if it is, the method makes a recursive call, adding 2 to “start” so it will not take the 1, that we just found, but if the number next to a multiple of five is not a 1, then this multiple has to be int the group’s sum, therefore we subtract it from it; finally the method makes two different recursive call, one in which we take the number and put it in group sum and 1 in which we don´t, the only case the method will return false, is when both of this calls return false, and they will only if when we evaluate the base case, if target equals to cero, is false.

3.5

EXERCISE OF CODINGBAT	EQUATION OF RECURRENCE	T(N) COMPLEXITY	(BIG O) COMPLEXITY
RECURSION 1			
Factorial	$T(n)=C_1n+C_2$	$T(n)=C_1+T(n-1)$	$O(n)$
Bunny Ears	$T(n)=C_1n+C_2$	$T(n)= C_1+T(n-1)$	$O(n)$
Triangle	$T(n)=C_1n+C_2$	$T(n)=C_1+T(n-1)$	$O(n)$
Fibbonacci	$T(n)=C_1(2^n-1)+C_22^{n-1}$	$T(n)=C_1+T(n-1) +T(n-2)$	$O(2^n)$
PowerN	$T(n)=C_1n+C_2$	$T(n)=C_1+T(n-1)$	$O(n)$
RECURSION 2			
GroupSum5	$T(n)=C_1(2^n-1)+C_22^{n-1}$	$T(m)=C_1+2T(m-1)$	$O(2^m)$
GroupSum6	$T(n)=C_1(2^n-1)+C_22^{n-1}$	$T(m)=C_1+2T(m-1)$	$O(2^m)$
GroupSumClump	$T(n)=C_1(2^n-1)+C_22^{n-1}$	$T(m)=C_1+2T(m-1)$	$O(2^m)$
SplitArray	$T(n)=C_1(2^n-1)+C_22^{n-1}$	$T(m)=C_1+2T(m-1)$	$O(2^m)$
Split53	$T(n)=C_1(2^n-1)+C_22^{n-1}$	$T(m)=C_1+2T(m-1)$	$O(2^m)$

3.6

- n is the only parameter in every method of recursion 1, is the one that changes in every recursive call each method makes, the size of it determines the execution time and memory occupation.

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

- m, in every case, is the length of the array selected in the parameter (Also can be defined as the difference between the length and the index “start”), it determines the size occupied in memory and execution time of the method

4) Practice for midterms

4.1 Line 4: start+1, nums, target
4.2 a
4.3 Line 04: n-a,a,b,c
Line 05: res,solucionar(n-b,a,b,c)+1
Line 06: res,solucionar(n-c,a,b,c)+1
4.4 e
4.5 **4.5.1** Line 2: return n
Line 3: n-1
Line 4: n-2
4.5.2 b
4.6 **4.6.1** Line 10: sumaAux(n,i+2)
4.6.2 Line 12: sumaAux(n,i+1)
4.7 **4.7.1** Line 9: S,i+1,t-S[i]
4.7.2 Line 10: S,i+1,t
4.8 **4.8.1** Line 9: return 0
4.8.2 Line 13: ni+nj
4.9 c
4.10 b
4.11 **4.11.1** Line 4: n-1, lucas(n-2)
4.12 **4.12.1** Line 13: 0
4.12.2 Line 17: math.max(fi,fj)
4.12.3 Line 18: sat