

Laboratory practice 1: Recursion

Santiago Isaza Cadavid

Universidad EAFIT
Medellín, Colombia
sisazac@eafit.edu.co

Hamilton Smith Gómez Osorio

Universidad EAFIT
Medellín, Colombia
hsgomezo@eafit.edu.co

August 18, 2018

1) Project questions Simulation

1.a. Explain how does GroupSum5 works

In this exercise the objective is to check if is possible to get a target (num) adding the numbers that are in an array. But there are restrictions: all the multiples of 5 in the array must be added. Procedure: the program receives an index, an array of integers and the target/objective. We evaluate if the index is equal or greater than the length of the array, if it is, if the target is zero then it returns “true” if not, then it returns “false”. While it is not the case we keep evaluating every position in the array and checking if it is multiple of 5, if it is then we verify that the next number is or not, a number one. If it is, we make a recursive call with the index plus two units (to omit the number one) and subtracting the multiple already found from the objective; when it is not like that, the parameters that we give to the program are the same but the index is now index plus one. When the number in the position of the index is not multiple of five then we make a recursive call in which we intend to analyze the next position of the array without counting the previous visited number and other recursive call in which we do count the previous visited number in order to check in which option we could get the target.

1.b. Calculate the complexity of the online exercises

1.2.1 powerN

```
public int powerN(int base, int n) {  
    int prod=1;  
    if(base==1 —— n==0) return 1;  
    else prod= base * powerN(base,n-1);  
    return prod;  
}
```

1.2.2 SumDigits

```
public int SumDigits(int n) { int sum=0;
    if(n==0)return 0;
    else sum= n%10 + sumDigits(n/10);
    return sum; }
```

1.2.3 Triangle

```
public int triangle(int rows) {
    int block=0;
    if(rows==0)return 0;
    else block= rows + triangle(rows-1);
    return block;
}
```

1.2.4 count7

```
public int count7(int n) {
    int cont=0;
    if(n==0) return 0;
    else if(n%10==7) cont= 1 + count7(n/10);
    else cont= count7(n/10);
    return cont;
}
```

1.2.5 bunnyEars

```
public int bunnyEars(int bunnies) {
    int ears=0;
    if(bunnies==0) return 0;
    else ears= 2+bunnyEars(bunnies-1);
    return ears;
}
```

1.c. What did you learn about Stack Overflow? Why does this happen?

Thanks to this error we learned that a good base case (stopping condition) is necessary in order to avoid an infinite loop, and to get the expected answer in a recursive algorithm. The Stack Overflow error happens due to a excessive use of memory. It happens when the algorithm makes a bad recursive call.

1.d. What is the greatest number you could get with the Fibonacci algorithm? Why? Why can't it work with Fibonacci one million?

We made two tests, one with the page visualgo, in which the program only calculated twenty-six numbers, then when we implemented it in Java and it returned values until the forty-sixth number. The program didn't returned more values because it was a recursive algorithm, this type of algorithms create a pile of cases to analyze that do not conclude until it arrives to the case base and, we it tries to do it with very large numbers, the time and memory that the machine needs to get the job done is a lot, so this program doesn't have a very optimal behavior.

1.e. What can you do to calculate bigger Fibonacci 's values?

One of the solutions could be changing the type of data that stores this values. For example, in our code we used int type but if we used a long int type we would get bigger values. Other solution is using a big data structure, for example a dictionary in Python and add the values to it, but this a solution proposed by Functional Programming.

1.f. What do you conclude about the complexity of CodingBat's Recursion 1 with respect Recursion 2?

2) Midterm Simulation

2.a. Exercise 1

start+1, nums, target

2.b. Exercise 2

The recurrence equation that describes the algorithm's behavior is: a) $T(n/2)+C$

2.c. Exercise 3

2.3.1 Line 4

n - a , a , b , c

Line 5

res , solucionar(n-b,a,b,c) + 1

2.3.2 Line 6

res , solucionar(n-c,a,b,c) + 1

2.d. Exercise 4

The algorithm calculates: e) The sum of the elements that are in the array a and its complexity is $O(n)$

2.e. Exercise 5**2.5.1 Complete the lines**

Line 2: return n Line 3: n-1 Line 4: n-2

2.5.2 How many instructions does this algorithm executes in the worst case scenario?

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

2.f. Exercise 6**2.6.1 Line 10**

0;

2.g. Line 12

sumaAux(n.substring(i+i), i+1)

2.h. Exercise 7**2.8.1 Line 9**

s, i+1, t;

2.8.2 Line 10

s, i+1, t-s[i];

2.i. Exercise 8**2.9.1 Complete line 9**

return 0;

2.9.2 Complete line 13

ni + nj