Laboratory practice No. 1: Recursion

Juliana Lalinde Velásquez
Universidad Eafit
Medellín, Colombia
jlalindev@eafit.edu.co

Isabel Urrego Gomez
Universidad Eafit
Medellín, Colombia
iurregog@eafit.edu.co

3) Practice for final project defense presentation

1. Exercise *GroupSum5*, from CodingBat, works by analyzing a series of conditions to see if there is a group of elements in a given array that add up to a given target number. It uses a start number to represent the position of the array that is being checked and then proceeds to check all the possible combinations of sums to see if at least one fits. To do this we subtract numbers (from the array) from the target so we can check in the end (when the start number equals or is bigger than the number of elements in the array) if the target number equals to zero, in which case there is a group of elements that sum the target. To be able to do this we need to know when to consider a number and when not to.
   The exercise has an additional condition: if there is a multiple of 5 it must be included in the group, and if the value following it is a 1 it cannot be included. To make sure these conditions are fulfilled we first check to see if the number at the position "start" is a multiple of 5, if it is we check if the following number is a 1. If both conditions are true, then we recursively call the method changing two things: we move the position we are taking up by two numbers (because we cannot include the 1 in the sum) and we subtract the multiple of 5 from the target number. If the following number is not a 1 the we move the position up only by one number and subtract the multiple from the target number. If the number is not a multiple of 5 then we have two possibilities, it can either belong to the group that sums the target or not. Since we need to check all the possible combinations we must execute two recursive calls: the first one moves the position we are taking up by one number and subtract the number we are checking from the target, or only move up the position by one number, without subtracting it. This way we make sure that every number different from a multiple of five is both considered part of a sum or not, which helps us check all the combinations.

2. Complexity

2.1. Recursion 1

_triangle:

$$T(n) = \begin{cases} c_1 & \text{n} = 0 \\ c_2 + T(n-1) & \text{n} \geq 0 \end{cases}$$

Solution:

$T(n) = c_2\, n + c_1$ (c_1 is an arbitrary parameter)

O(n)

_sumDigits:

$$T(n) = \begin{cases} c_1 & \text{n} \leq 10 \\ c_2 + T(n/10) & \text{n} \geq 10 \end{cases}$$

Solution:

$T(n) = \frac{c_2 \log(n)}{\log(10)} + c_1$ (c_1 is an arbitrary parameter)

O(log(n))

_powerN:

$$T(n) = \begin{cases} c_1 & n = 0 \\ c_2 & n = 1 \\ c_3 + T(n-1) & n > 1 \end{cases}$$

Solution:

$T(n) = c_3\, n + c_1$ (c_1 is an arbitrary parameter)

O(n)

PROFESSOR MAURICIO TORO BERMÚDEZ
Phone: (+57) (4) 261 95 00 Ext. 9473. Office: 19 - 627
E-mail: mtorobe@eafit.edu.co

_changePi

$$T(n) = \begin{cases} c_1 & \text{n} < 2 \\ c_2 + T(n-2) & \text{n} \geq 2 \\ c_3 + T(n-1) & \text{n} \geq 2 \end{cases}$$

Solution:

$T(n) = c_2\,(-1)^n - \frac{1}{4}\,c_2\,\left((-1)^{2\,n} - 2\,n\right) + c_1$ (c_1, c_2 are arbitrary parameters)

$T(n) = c_2\,n + c_1$ (c_1 is an arbitrary parameter)

O(n)

_noX

$$T(n) = \begin{cases} c_1 & \text{n} = 0 \\ T(n-1) & \text{n} > 0 \\ c_2 + T(n-1) & \text{n} > 0 \end{cases}$$

Solution:

$T(n) = c_1$ (c_1 is an arbitrary parameter)

$T(n) = c_2\,n + c_1$ (c_1 is an arbitrary parameter)

O(n)

2.2. Recursion 2

groupSum6

$$T(n) = \begin{cases} c_1 & \text{if } n = 0 \\ T(n-1) & \text{if } n > 0 \\ 2T(n-1) + c_2 & \text{if } n > 0 \end{cases}$$

Solution:

$T(n) = c_1$ (c_1 is an arbitrary parameter)

$T(n) = c_1\,2^{n-1} + c_2\,(2^n - 1)$ (c_1 is an arbitrary parameter)

O(2^n)

PROFESSOR MAURICIO TORO BERMÚDEZ
Phone: (+57) (4) 261 95 00 Ext. 9473. Office: 19 - 627
E-mail: mtorobe@eafit.edu.co

groupNoAdj

$$T(n) = \begin{cases} c_1 & \text{if } n = 0 \\ T(n-2) + T(n-1) + c_2 & \text{if } n > 0 \end{cases}$$

Solution:

$T(n) = c_1\, 2^{n-1} + c_1\left(2^n - 1\right)$ (c_1 is an arbitrary parameter)

O(2^n)

groupSum5

Complexity of the internal conditional:

$$\begin{cases} T(n) = T(n-2) \\ T(n) = T(n-1) \end{cases}$$

$$T(n) = \begin{cases} c_1 & \text{if } n = 0 \\ T(n-1) + c_2 & \text{if } n > 0 \\ 2T(n-1) + c_3 & \text{if } n > 0 \end{cases}$$

Solution:

$T(n) = c_2\, n + c_1$ (c_1 is an arbitrary parameter)

$T(n) = c_1\, 2^{n-1} + c_3\left(2^n - 1\right)$ (c_1 is an arbitrary parameter)

O(2^n)

splitArray

$$T(n) = \begin{cases} c_1 & \text{if } n = 0 \\ 2T(n-1) + c_2 & \text{if } n > 0 \end{cases}$$

PROFESSOR MAURICIO TORO BERMÚDEZ
Phone: (+57) (4) 261 95 00 Ext. 9473. Office: 19 - 627
E-mail: mtorobe@eafit.edu.co

Solution:

$T(n) = c_1\,2^{n-1} + c_1\,(2^n - 1)$ (c_1 is an arbitrary parameter)

O(2^n)

splitOdd10

$$T(n) = \begin{cases} c_1 & \text{if } n = 0 \\ 2T(n-1) + c_2 & \text{if } n > 0 \end{cases}$$

Solution:

$T(n) = c_1\,2^{n-1} + c_1\,(2^n - 1)$ (c_1 is an arbitrary parameter)

O(2^n)

3. Explanation of the variables

3.1. Recursion 1

- triangle is O(n), where n is the number of rows
- sumDigits is O(log(n)), where n is the number whose digits are going to be summed
- powerN is O(n), where n is the exponent
- changePi is O(n), where n is the length of the String
- noX is O(n), where n is the length of the String

3.2. Recursion 2

- groupSum6 is O(2^n), where n is the difference between the number of elements in the array and the index
- groupNoAdj is O(2^n), where n is the difference between the number of elements in the array and the index
- groupSum5 is O(2^n), where n is the difference between the number of elements in the array and the index
- splitArray is O(2^n), where n is the difference between the number of elements in the array and the index
- splitOdd10 is O(2^n), where n is the difference between the number of elements in the array and the index

4. About Stack Overflow

The main difference between an iteration and recursion lies in the allocation of new memory. Recursive functions never lead to infinite loops since they collect information on the stack pile with each recursive call. The amount of recursive calls is

limited by the capacity of the stack pile and that is the reason why the program throws an error when the stack is full, called Stack Overflow.

5. The biggest value that we calculated in Fibbonaci using integers in Java is when n=46. Any value of n bigger than 46 is a number bigger than (2^32)-1 and cannot be represented by integers.

7. The complexity of the Recursion 1 problems is part of a smaller order than the complexity of the Recursion 2 problems. This fact is given by the number of recursive calls that you must implement in order to properly solve the problem, in Recursion 1 you only have to implement one recursive call, while in Recursion 2 you must implement two.

4) Practice for midterms

1. (start+1, nums, target)
2. a
3.
   3.1. Line 4: n-a,a,b,c
   3.2. Line 5: res, solucionar(n-b,a,b,c)+1
   3.3. Line 6: res, solucionar(n-c,a,b,c)+1
4. e
5.
   5.1. Line 2: return n
   Line 3: n-1
   Line 4: n-2
   5.2. b
6.
   6.1. Line 10: sumaAux(n, i+2)
   6.2. Line 12: sumaAux(n, i+1)
7.
   7.1. Line 9: S, i+1, t-S[i]
   7.2. Line 10: S, i+1, t
8.
   8.1. Line 9: return 0
   8.2. Line 13: ni+nj

5) Recommended reading (optional)

a) Recursion and Backtracking

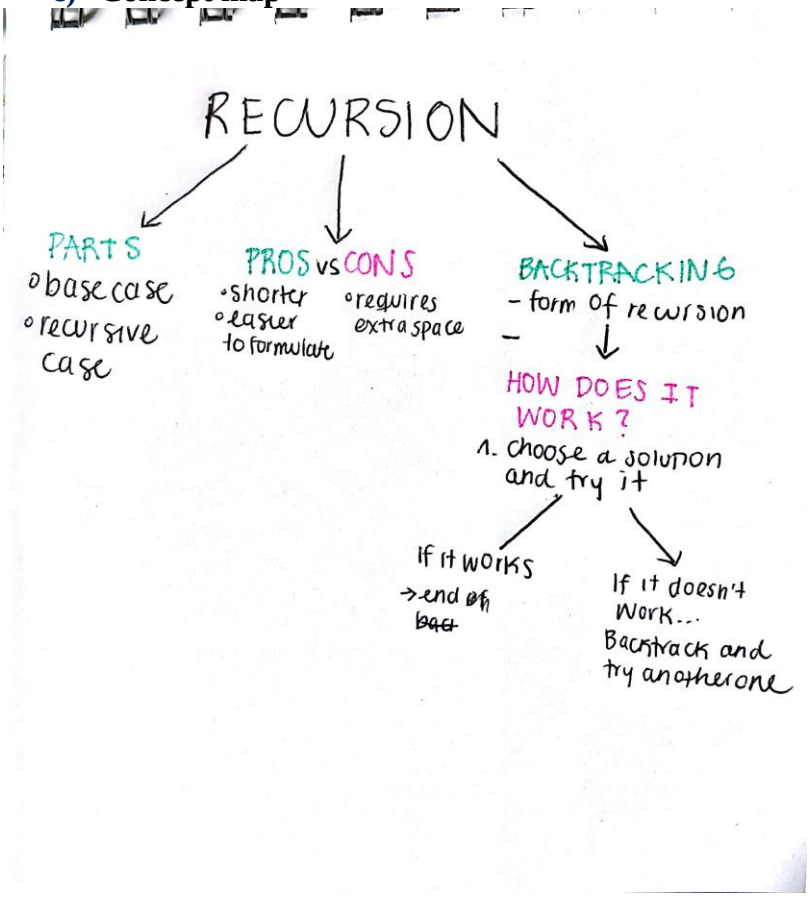
PROFESSOR MAURICIO TORO BERMÚDEZ
Phone: (+57) (4) 261 95 00 Ext. 9473. Office: 19 - 627
E-mail: mtorobe@eafit.edu.co

b) The term recursion is mainly used referring to a technique which solves a problem by calling itself and comes to a solution by reducing the problem into smaller parts. Additionally, a recursive function is made of two important things, the base case and the recursive case.

It's also important to notice that recursive calls need extra space on the memory and not reaching a base case leads to an error called Stack Overflow.

One example of a recursion form is Backtracking, an improvement of the brute force approach. It consists in looking for a solution by trying every single possibility till it finds one that solves the problem. If the option tried doesn't work, the function will backtrack and choose another possibility. That's where the name backtracking comes from.

c) Concept map



PROFESSOR MAURICIO TORO BERMÚDEZ
Phone: (+57) (4) 261 95 00 Ext. 9473. Office: 19 - 627
E-mail: mtorobe@eafit.edu.co