Laboratory practice No. I: Recursion

Camila Barona Cabrera
Universidad Eafit
Medellín, Colombia
cbaronac@eafit.edu.co

Mariana Gómez Piedrahita
Universidad Eafit
Medellín, Colombia
mgomezp10@eafit.edu.co

3)

1. In the exercise they give us 3 variables, the first one in an integer that stores the position from where the array begins, the second is an integer array and the last is an integer which brings us the value of the sum that we must get. In general, In general, we seek to sum what variable 3 tells us with some of the elements within the arrangement; but for this, there are some conditions that must be met, firstly, the numbers multiples of 5 must be added to the sum and if the value that follows the multiple of 5 is 1, it does not count.

Explanation of the GROUPSUM5 exercise:

The decision to stop is when the variable that runs through the array (could say that as a counter) is equal or greater than the length of the array, the sum is zero, that is, we have already found the numbers of the total sum.

There is a first decision in which two things are done:

The recursive call, in which we increase the "counter" by 1, subtract what is in the position to the total sum (target) only if the call of the method (checkOne) is true, it does not give true if the value of the position in which we are standing is a multiple of 5 and the next position is different from 1, it is true if the opposite happens. Then, we make a second recursive call, first, we verify if the position in which we are standing is different from 5 and then we make the recursive call by passing the counter + 1, the arrangement and the sum.

The program continues to do all this, until we find the numbers within the arrangement that add what is requested by the "target", that is, when the stop decision is met.

2.

Complexity of the algorithms:

1.BunnyEars2:

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

Answer
T(n)= c_2*n+c_1

NotationO
O(c_2*n)
O(n)

- **The exercise bunnyEars2 is O (n) and n is the number of ears of rabbits.**

2. Triangle:

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

Answer
T(n)= c_2*n+c_1

Notation O

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

O(c_2*n)
O(n)

- **The Triangle exercise is O (n) and n is the row or blocks it has in a triangle**

3. SumDigits:

$$T(n) = \begin{cases} c_1 & \text{if } n < 10 \\ c_2 + T(n/10) & \text{if } n >= 10 \end{cases}$$

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

Notation O
O($\frac{c_2 \log(n)}{\log(10)}$)
O(log(n))

- **The exercise sumDigits is O (log (n)) and n is the number of which we want to add its figures.**

4. CountHi:

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

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

Notation O
O(c1)
O(1)

Answer
T(n)=c1

5. Count7:

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

Notation O
O($\frac{c_2 \log(n)}{\log(10)}$)
O(log(n))

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

- **The count7 exercise is O (log (n)) and n is the number to which we are going to tell the 7.**

RESOURCE 2:

1.GroupSum6:

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

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

Answer O
O ($c_1\, 2^{n-1}$)
O(2^{n-1})

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

2.GroupNoAdj:

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

Answer
$T(n) = c_1\, F_n + c_2\, L_n - c_2$ (c_1 , c_2 are arbitrary parameters)

Notation O
O (2^n)

Code took from: http://gregorulm.com/codingbat-java-recursion-2/

3.SplitArray

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

Notation O

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

O ($c_1 2^{n-1}$)
O(2^{n-1})

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

Code took from: http://gregorulm.com/codingbat-java-recursion-2/

4.SplitOdd10:

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

NotationO
O ($c_1 2^{n-1}$)
O(2^{n-1})

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

Code took from: http://gregorulm.com/codingbat-java-recursion-2/

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

3.3

In the complexity calculation, the n represents the main variable, that is, the variable that changes constantly during the whole algorithm, it is the same that accomplish the stop condition at some point.
In the same way we represent the recursive call with the letter T.
The letters C, symbolize constants that can be found throughout the algorithm.

3.4

It appears simply because the memory needed for the program is bigger than the one that I configure the execution environment in java, better said a stack overflow; This is mainly due to a bad exit condition in the recursive method.

3.5

One of the biggest problems of the Fibonacci succession, is that from a value the calculation takes longer to be made, since, if the n is increasing, the result also; That said, every time we enter a larger n the size of the variable in which we save the result must have the right size to be able to save the number, the largest number to calculate in Fibonacci is 92, having the result variable in Long type, whose maximum value is: 9.223.372.036.854.775.807.
Fibonacci of 92 is: 7,540,113,804,746,346,429
Because the 93 Fibonacci is already passed: 12,200,160,415,121,876,738
And the Fibonacci of 1 million can't be executed since the value of the result can't be saved in any variable of sufficiently long type.

Numerical data taken from: https://medium.com/@williamkhepri/fibonacci-y-optimizaci%C3%B3n-f0b9590e989a

3.6

One of the options is to use dynamic programming, for the Fibonacci problem, when we are going to calculate a number, we simply need the previous two and this we had already calculated before, the idea is to check if the solution already exists, that is, if we had done it, if so, we will use it for the new problem.

3.7

At the beginning, performing codingbat recursion 1 is difficult, but after performing several exercises it becomes simpler in general, one of the facts that most differentiate recursion 1 and recursion 2 are: in recursion 1 with a single recursive call and with only one method in most cases could be solved easily, in recursion 2 for the problem to work better, it is necessary to make several recursive calls, and even have help from other methods, likewise in recursion 2 must be taken into account without doubt, many more details that are of vital importance in stopping decisions and in the procedure, the number of parameters increases considerably; in parallel with recursion 1 all the above is taken into account, but to a lesser degree.

4) Practice for midterms

1. *Start+1, nus, target*
2. D
3.
4. The unknown algorithm adds the numbers entered in an array
   Complexity:
   O(n)
5. *Line 1: Return 0;*
   Line2:Formas(n-1)
   Line3:Formas(n-2)

5.2 *In the worst case the algorithm executes*
T(n)=T(n-1)+T(n-2)+c

6. *Line10: return Suma(n,i+2)*
   Line12: SumaAux(n,i+1)
7. *Line9:return comb(S,i+1,t-S[i])*
   Line10:comb(S,i+1,t)
8. *Return 0*
   Return ni+nj

Explanation online exercises:

1. BunnyEars2:
This code is based on counting how many ears a rabbit will have recursively, however if it is an even number, that is, bunnies% 2 == 0, 3 ears will be counted, the opposite of if it is an odd number, two will be added. Therefore, the condition of stop is made at the moment of not having any rabbit.

2. Triangle:
This code is based on counting how many blocks a triangle has, which gives us the number of rows, we know that the highest row has a single block and those that follow it increase it by 1.

3. SumDigits:
This code is based on counting the digits of a given number, in each recursive call we take care of: adding the previous number taken, and of obtaining the new number, this last one we are doing in pairs, it is achieved by dividing by 10 and to remove the last, using the% mod.

4.CountHi:
This code is based on counting the number of times "HI" in a chain, we go through the String and we do substring of 0.2 comparing if it matches "hi", if so, we add 1 and return the cut string to not return to Start the search from the same part.

5.Count7:
This code is based on counting the number of times that "7" appears in a chain, we use the same structure as SumDigits, but comparing if what we take from the String is a number 7.

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

6.GroupsSum6:

This code is based on choose a group that sum an int given, chosen all 6's that appears in the array. For that every time when 6 are in the array, we search another int that accomplish the sum.

7.SplitArray:

This code is based on divided the array in two groups and find if the sum of those groups are the same. The algorithm will stop when sum of group 1 will the same at group 2.

8.GroupNoAdj:

This code is based on choose a number or a group of numbers of the array, with the purpose of get a value (target) with their sums, but one can't be following the other in the array.

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