Laboratory practice No. X: Complete the title of the laboratory practice

Manuel Garcia Jimenez
Universidad Eafit
Medellín, Colombia
correoinegrante1@eafit.edu.co

Isaias Labrador Sanchez
Universidad Eafit
Medellín, Colombia
Correointegrante2@eafit.edu.co

2)

3. Explanation of some codes from "CodingBat Java"

GroupSum5: For the forming of the different subsets of numbers, the logic is the same as the GroupSum exercise; we return the universe were the next number of the array is included in the subset and the universe in which it isn´t. The differences between this exercise and Groupsum is that if the next number of the array is a 5 we only return the universe in which it is put into the subset, and as for the number after it, if it´s a 1 we only return the universe in which it isn´t put into the subset.

SplitArray: <Gregor Ulm> <March 28, 2013> <CodingBat: Java. Recursion-2> http://gregorulm.com/codingbat-java-recursion-2/

Same logic as GroupSum, only that two variables are created to be the sums of the different subsets. When we have a number we either add it to the first variable or don´t, case in which it gets put into the second one. Both universes are returned. Since we repeated this process for every number on the array one by one, every number gets put into one variable or the other, but it is never skipped. When there are no numbers left on the array, we ask if the two variables are equal.

4. Complexity calculations:

Factorial: o(n!) factorial
bunnyEars: o(n) linear
Fibonacci: o(c ^(n)) Exponential
BunnyEars2: o(n) Linear
powerN: o(c ^(n)) Exponential
groupSum, groupSum6, groupNoAdj, groupSum5, and splitArray: o(c ^(n)) Exponential

5. Explanation of point 2.4:

Factorial: n*T(n-1), N is the number for which we want to know the factorial
bunnyEars: c+T(n-1), c is the amount of steps before the recursive call, n is the number of bunnies.
Fibonacci: c+T(n-1)+T(n-2), c is the amount of steps before the recursive calls, n is the number on the series we're looking for
BunnyEars2: c+T(n-1), same as bunnyEars.

powerN: n*T(n-1), n is the base.

groupSum and other recurssion2 methods: c+T(n)+T(n-1), c is steps before the recursive calls, n is the index number we look at on the array.

3) Practice for final project defense presentation

1. The space in RAM memory for the computer to execute recursive methods is like a basket were iterations of the method are put into a stack. If the stack overflows the basket, the iterations “drip off” and the execution of the method falls apart.
2. 46. No more can be done because it would not only take to long, but the answer wound be a negative number, which is noticeably wrong.
3. To calculate the nth number of the Fibonacci sequence use Benet’s formula. Since it is a closed formula we don’t need the n-1th and n-2th numbers.
4. Since on recursion one all of the method only invoke one recursive call, they tend to be less complex than those of recurssion2 which always have at least 2 recursive calls.

4) Practice for midterms

1. *Start+1, nums, target*
2. *a*
3. **line 4:** int res= solucionar(n-a, a,b,c)+1;
   line 5: res=Math.max(res,solucionar(n-b,a,b,c));
   line 6: res=Math.max(res,solucionar(n-c,a,b,c));
4. *e*
5.
   5.1 return n;
   formas(n-1)+
   formas(n-2)
   5.2 b
6. line 10: return sumaAux(n, i+2);
   line 12: return (n.charAt(i)-‘0’)+sumaAux(n, i+1);
7. ***line 9:*** *return comb(s, i+1, t-S[i]) ||*
   line 10: *comb(s, i+1, t);*