Laboratory practice No. 1: Recursion and software complexity

Santiago Puerta Florez
Universidad Eafit
Medellín, Colombia
spuertaf@eafit.edu.co

Jose Manuel Fonseca Palacio
Universidad Eafit
Medellín, Colombia
jmfonsecap@eafit.edu.co

3) Practice for final project defense presentation

3.1

```
private static int lcsDNAAux(String string1, String string2,int m,int n) {
    // p = n + m
      if (m<=0||n<=0) //T (n,m) = c1, c1 = 5
                        //T(p) = c1
      {
         return 0;
      }

      else if (string1.charAt(m-1) == string2.charAt(n-1)){ // T(n,m) = c2 + T(n-1, m-1)
                                                              // T(p) = c2 + T(p-2)
         return 1 + lcsDNAAux(string1.substring(0,m-1),string2.substring(0,n-1),string1.substring(0,m-1).length(),string2.substring(0,n-1).length());
      }
      return Math.max(lcsDNAAux(string1.substring(0,m-1),string2,string1.substring(0,m-1).length(),n), lcsDNAAux(string1,string2.substring(0,n-1),m,string2.substring(0,n-1).length()));
              // T(n,m) = c3 + T(n-1,m) + T(n,m-1)
              // T(p) = c3 (2^p - 1) + c1(2^(p-1))
              // T(p) = (c3 + c1/2)2^p -c3
   }
```

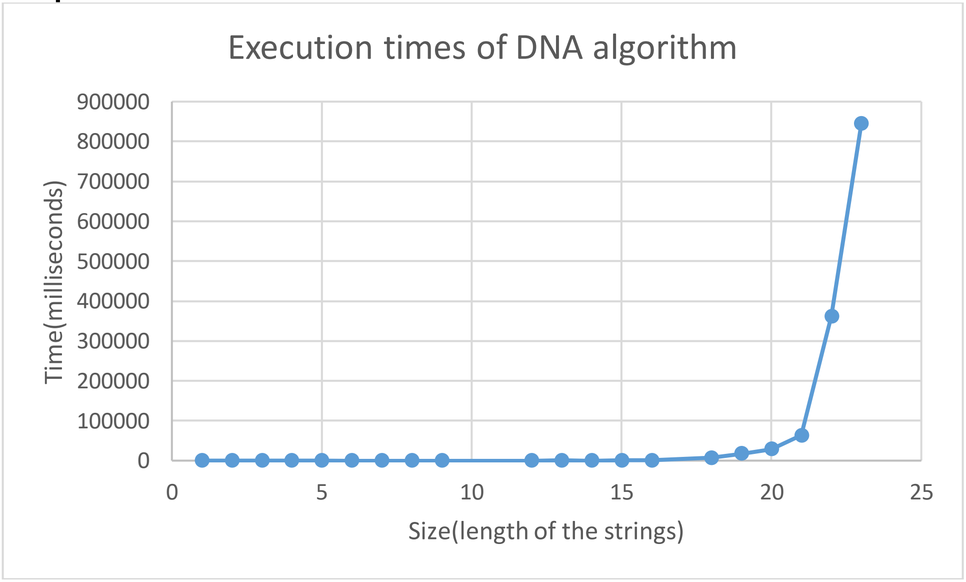
The complexity for the worst-case scenario is 2^p

3.2

The duration for two strings of lengths: 1 and 1 each, was: 0 milliseconds
The duration for two strings of lengths: 2 and 2 each, was: 0 milliseconds
The duration for two strings of lengths: 3 and 3 each, was: 0 milliseconds
The duration for two strings of lengths: 4 and 4 each, was: 0 milliseconds
The duration for two strings of lengths: 5 and 5 each, was: 0 milliseconds

The duration for two strings of lengths: 6 and 6 each, was: 1 millisecond
The duration for two strings of lengths: 7 and 7 each, was: 1 millisecond
The duration for two strings of lengths: 8 and 8 each, was: 1 millisecond
The duration for two strings of lengths: 9 and 9 each, was: 5 milliseconds
The duration for two strings of lengths: 12 and 12 each, was: 85 milliseconds
The duration for two strings of lengths: 13 and 13 each, was: 333 milliseconds
The duration for two strings of lengths: 14 and 14 each, was: 191 milliseconds
The duration for two strings of lengths: 15 and 15 each, was: 258 milliseconds
The duration for two strings of lengths: 16 and 16 each, was 725 milliseconds
The duration for two strings of lengths: 18 and 18 each, was: 6789 milliseconds
The duration for two strings of lengths: 19 and 19 each, was: 17189 milliseconds
The duration for two strings of lengths: 20 and 20 each, was 28374 milliseconds
The duration for two strings of lengths: 21 and 21 each, was: 63415 milliseconds
The duration for two strings of lengths: 22 and 22 each, was: 362068 milliseconds
The duration for two strings of lengths: 23 and 23 each, was: 845062 milliseconds

Graph:



While the size of the Strings rises the algorithm will take more time to compute the longest subsequence between them.

The algorithm would take approximately 275869,5652 minutes to give an answer, this is equivalent to 524.9 decades.

3.3

Is the complexity of the algorithm appropriate for finding the largest common subsequence of mitochondrial DNA like the ones given on the data sets?

No, it is not the algorithm would take 2^p instructions to calculate the largest common subsequence between two strings. Also, we must consider that p is equal to n + m, n and m

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

being the length of both strings. So, for two strings with lengths of 300.000 each the algorithm would take 2^900.000 instructions before giving an answer.

3.4
The method group sum 5 verifies truth or false if given an array of integers it can take a group of some of the integers for reaching a target, the method begins by the index or position of start, and also this method has a twist in it giving the instruction that all multiples of five must be in the group and if the value that immediately follows a multiple of 5 is a 1, this value must not be included in the group. The code for this exercise is the following:

```
public boolean groupSum5(int start, int[] nums, int target)
{
        if(start >= nums.length)
        {
                if(target == 0)
                return true;
        return false;
        }
        if(nums[start] % 5 == 0)
        {
                if(start < nums.length - 1 && nums[start+1] == 1)
                        return groupSum5(start + 2, nums, target - nums[start]);
                return groupSum5(start + 1, nums, target - nums[start]);
        }
  if(groupSum5(start + 1, nums, target - nums[start]))
        return true;
  return groupSum5(start + 1, nums, target);
}
```

3.5
Recursion 1:
Factorial:
T(n) = c_2 + T(n-1)
Fibonacci:
T(n) = T(n-2) + T(n-1)
Array11:
T(n) = T(n-1) + c
BunnyEars:
T(n) = T(n-1) + c_2
powerN:
T(n) = c_2 + T(n-1)
Recursion 2:

groupSum6:
T(n) = 2 T(n-1) + c_2
groupSumClump:
T(n) = T(n-m) + T(n-1) + Cn + c
groupNoAdj:
T(n) = T(n-1) + c_2
splitArray:
T(n) = 2 T(n-1) + c_2
splitOdd10:
T(n) = 2 T(n-1) + c_2

3.6
Factorial:
n represents the value to find the factorial for.
Fibonacci:
n represents the nth Fibonacci number to the sequence.
Array11:
n represents the number of elements in the array.
BunnyEars:
n represents quantity of bunnies.
powerN:
n represents the power to which a base rise.
groupSum6:
n represents the size of the integer array
groupSumClump:
n represents the size of the integer array.
groupNoAdj:
n represents the size of the integer array.
splitArray:
n represents the size of the integer array.
splitOdd10:
n represents the number of elements of the array.

4) Practice for midterms

4.1
- ***4.1.1*** A
- ***4.1.2*** C
- ***4.1.3*** A

4.3 B
4.4
- ***4.4.1*** C

4.5
- ***4.5.1*** A
- ***4.5.2*** B

4.6 D

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

4.7 E
4.9
  4.9.1 return 0
  4.9.2 +sumaAux (n,i+1)
4.11 *C (22)*
4.12 *B*

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