ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 1: LABORATORY

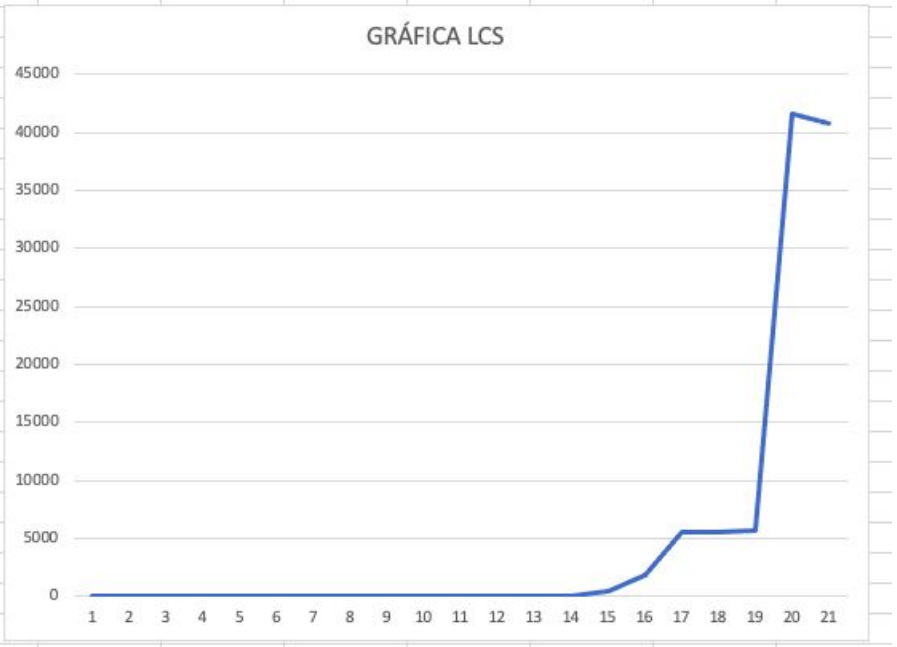
Edison Alejandro Torres Muñoz
Universidad Eafit
Medellín, Colombia
eatorresm@eafit.edu.co

Mateo Muñoz Cadavid
Universidad Eafit
Medellín, Colombia
mmunozc4@eafit.edu.co

3) Practice for final project defense presentation

3.1 O(2^n)
3.2

T (ms)	caracteres
23	3
18	6
20	8
18	9
19	7
21	10
23	11
24	12
29	13
30	14
50	15
78	16
73	17
385	18
1818	19
5544	20
5556	21
5682	22
41566	23
40812	24



3.3 The complexity of the algorithm that measures the longest common subsequence is not suitable to be able to measure chains of such magnitude as datasets are, since its recursive implementation is not fully optimized for this process. Apart from this, a machine with a larger memory is needed to achieve this effectively.

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

3.4 [OPTIONAL] The algorithm called groupSum5 contains the following variables: start that corresponds to the beginning of the index, an array of int numbers called nums, and a target variable that corresponds to the sum to be reached.

The algorithm always starts by comparing that the start variable is greater than the length of the array, if this is true, the target variable will be saved with a value of "0".

Subsequently, it is compared if the array in the position of the start value is a multiple of 5, if this is true, it will be seen if the length of the array is different from the increase of start by one unit and if the value that is in the start position plus one is equal to one, we will proceed to add two units to the start variable and subtract from the target variable the value found in the start position of the array, otherwise, only the amount of one will be added

At the end of the program, the algorithm always has the possibility of making two recursive calls, one is by adding a unit to the start variable and subtracting from the target variable the value found in the start position of the array, and the other call, corresponds simple and only to add a unit to start.

3.5, 3.6

RECURSION 1:

array220:

```
public boolean array220(int[] nums, int index) {
  if(nums.length-1 <=index){ //C1= 1
    return false; //C2 = 1
  }if(nums[index+1] == nums[index] *10){ //C3 = 1
    return true; //C4 = 1
  }
  return array220(nums, index+1); //T(n) = T(n-1)
  //T(n) =T(n-1)+C1+C2+C3+C4
  //T(n) = T(n-1)
  //O(C1n+C1)
  //O(n)
}
```

n: fix length

bunnyEars2:

```
public int bunnyEars2(int bunnies) {
  if(bunnies == 0){ //C1 = 1
    return 0; //C2 = 1
  }if(bunnies %2 == 0){ //C3= 1
    return bunnyEars2(bunnies-1)+3; // T(n) = T(n-1)+C4
  }else{ //C5
    return bunnyEars2(bunnies-1)+2; //T(n) = T(N-1)+C6
    //T(n) = T(n-1)
    //T(n) = T(n-1)+C1+C2+C3+C4+C5+C6
    //T(n) = T(n-1)
    //O(C1n+C1)
    //O(n)
}
}
```

n: number of numbers to evaluate (number of rabbits)

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

bunnyEars:

```java
public int bunnyEars(int bunnies) {
  if(bunnies == 0){ //C1 = 1
    return 0; //C2 = 1
  }else if(bunnies ==1){ //C3 = 1
    return 2; //C4 = 1
  }else{  //C5 = 1
    return bunnyEars(bunnies-1)+2; //T(n) = T(n-1)+C6
    //T(n) = T(n-1)+C1+C2+C3+C4+C5+C6
    //T(n) = T(n-1)
    //O(C1n+C1)
    //O(n)
  }
}
```

n: number of numbers to evaluate (number of rabbits).

PoweN:

```java
public int powerN(int base, int n) {
  if (n==0){ //C1 =1
    return 1; //C2 = 1
  }
  if(n==1){ //C3 = 1
    return base; //C4 = 1
  }
  return powerN(base, n-1)*base; //T(n) = T(n-1)+C5
  //T(n) = T(n-1)+C1+C2+C3+C4+C5
  //T(n) = T(n-1)
  //O(C1n+C1)
  //O(n)

}
```

n: power of the number to evaluate (base).

Fibonacci:

```java
public int fibonacci(int n) {
  if (n==0){ //C1= 1
    return 0; //C2 = 1
  }else if(n==1){ //C3 = 1
    return 1; //C4 = 1
  }
  return fibonacci(n-1)+fibonacci(n-2); //T(n) = T(n-1) + T(n-2)
  //T(n) = T(n-1) + T(n-2)+C1+C2+C3+C4
  //T(n) = T(n-1) + T(n-2)
  //O(-C1+C1Fn+C2Ln)
  //O(Fn+Ln)
}
```

n: represents a succession of evaluated numbers.

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

RECURSION 2:

groupNoAdj:

```
public boolean groupNoAdj(int start, int[] nums, int target) {
  if (start >= nums.length){//  c1=1
    return (target == 0);// c2=1
  }
  if (groupNoAdj(start + 2, nums, target - nums[start])){ // t(n)= t(n+2) + c3
    return true;// c4=1
  }
  if (groupNoAdj(start + 1, nums, target)) return true; t(n)= t(n+1) + c5
  return false; // c6=1
}
//T(n) = T(n-1) + T(n-2) + c1 + c2 + c3 + c4 + c5 + c6
//T(n) = T(n-1) + T(n-2) + c1
//T(n) = -c1 + cFn + cLn
//O(-c1 + cFn + cLn)
//O(Fn+Ln)
```

n: size of the array.

groupSum6:

```
public boolean groupSum6(int start, int[] nums, int target) {

  if(start >= nums.length){ //C1 = 1
    return target==0; //c2= 1
  }
  if(nums[start] == 6){ //C3 = 1
    return groupSum6 (start + 1 , nums, target-nums[start]); //T(n) = T(n+1) + C4
  }
  return groupSum6(start + 1, nums , target) || //T(n) = T(n+1) + C5
  groupSum6(start + 1, nums, target- nums[start]);
  //T(n) = T(n-1)+T(n-1)+C1+C2+C3+C4+C5
  //T(n) = T(n-1)+T(n-1)
  //O(C1(2^n-1)+c12^n-1)
  //O(2^n-1+2^n-1)
  //O(2^n)
}
```

n: number of elements in the array.

groupSum5:

```
public boolean groupSum5(int start, int[] nums, int target) {
  if (start >=nums.length){
    return target ==0;
  }
  if (nums[start]%5 == 0){
    if(start +1!= nums.length && nums [start +1] == 1){
      return groupSum5(start+2,nums,target-nums[start]);
    }else{
      return groupSum5(start+1,nums,target-nums[start]);
    }
  }
  return (groupSum5(start+1,nums,target-nums[start]) || groupSum5(start +1, nums, t
}
        // T(n) = T(n-1)+C5
        //T(n) = T(n-1)+T(n-1)+C1+C2+C3+C4+C5
        //T(n) = T(n-1)+T(n-1)
        //O(C1(2^n-1)+C12^n-1)
        //O(2^n-1+2^n-1)
        //O(2^n)
```

n: number of elements in the array.

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

groupSumClump:

```
public boolean groupSumClump(int start, int[] nums, int target) {
  int temp = 0; //C1 = 1
  int i = start; //C2 = 1
  if(start >= nums.length){ //C3 = 1
    return target == 0; //C4 = 1
  }
  while (i<nums.length && nums[start] == nums[i]){ //C5 = 1
    temp += nums[i]; //C6 = 1
    i++; //C7 = 1
  }
  if(groupSumClump(i, nums, target-temp) || groupSumClump(i, nums, target)){ //T(n)
    return true;
  }
  return false;
}
//T(n) = T(n-1)+T(n-1)+C1+C2+C3+C4+C5+C6+C7
//T(n) = T(n-1)+T(n-1)
//O(C1(2^n-1)+C12^n-1)
//O(2^n-1+2^n-1)
//O(2^n)
```

n: number of elements in the array.

splitArray:

```
public boolean splitArray(int[] nums) {
  int index = 0; //C1 = 1
  int sum1 = 0; //C2 = 1
  int sum2 = 0; //C3 = 1

  return listaArray (nums, index, sum1, sum2); //C4 = 1
}
private boolean listaArray(int []nums, int index, int sum1, int sum2){ //C5 = 1
  if(index >= nums.length){ //C6 = 1
    return sum1 == sum2; //C7 = 1
  }
  int value = nums[index]; //C8 = 1
  return (listaArray(nums, index +1, sum1 + value, sum2) ||
  listaArray(nums, index +1, sum1, sum2+value)); //T(n) = T(n-1)+T(n-1)

}
//T(n) = T(n-1)+T(n-1)+C1+C2+C3+C4+C5+C6+C7+C8
//T(n) = T(n-1)+T(n-1)
//O(C1(2^n-1)+C12^n-1)
//O(2^n-1+2^n-1)
//O(2^n)
```

n = number of elements in the array.

4) Practice for midterms

4.1

4.1.1 c

4.1.2 c

4.1.3 a

4.2

4.2.1 línea 9 -> floodfillUtil(Screen, (x+1, y-1), prevC, newC, N, M)
4.2.2 línea 10 -> floodfillUtil(Screen, (x-1, y+1), prevC, newC, N, M)
4.2.3 línea 11-> floodfillUtil(Screen, (x+1, y+1), prevC, newC, N, M)
4.2.4 línea 12 -> floodfillUtil(Screen, (x-1, y-1), prevC, newC, N, 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

T(n,m) = T(n-1,m) + T(n, m-1) + T(n-1,m-1)

4.3 b
4.4
4.4.1 return lucas(n-1) + lucas(n-2);
4.4.2 c
4.5
4.5.1 a
4.5.2 b

6. [OPTIONAL]

Member	Date	Done	Doing	To do
Alejandro Torres	21/02/21	I read the lab guide.		Inquire about the SCML (Longest Common Subsequence)
Mateo Muñoz	21/02/21	I read the lab guide.		Investigate the complexity of algorithms and establish the most viable one for the project.
Alejandro Torres	21/02/21	I inquired about the SCML in java..	Review Matthew's research.	Establish the development environment and implement the algorithm that calculates the SCML between two character strings.

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

Mateo Muñoz	21/02/21	I did research on the complexity of algorithms.	Watch a video about SCML.	Establish the programming language to use and implement the algorithm that calculates the SCML between two character strings.
Alejandro Torres	22/02/21	I set the development environment	Develop the algorithm.	Test the algorithm
Mateo Muñoz	22/02/21	I set the programming language.	Develop the algorithm.	Solve exercises online.
Alejandro Torres	23/02/21	I tested the algorithm.		Solve exercises online.
Mateo Muñoz	23/02/21	I solved exercises online (Recursion I, II).	Analyze the exercises that Mateo solved..	Analyze the complexity of the online exercises.
Alejandro Torres	23/02/21	I solved exercises online (Recursion I, II).	Analyze the exercises Alejandro solved.	Analyze the complexity of the online exercises.
Mateo Muñoz	24/02/21	I analyzed the complexity of the online exercises (Recursion I, II).	Detail the complexity of the exercises that Mateo did.	Analyze the complexity of the SCML algorithm.

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

Alejandro Torres	24/02/21	I analyzed the complexity of the online exercises (Recursion I, II).	Detail the complexity of the exercises carried out by Alejandro.	Analyze how appropriate is the complexity of the SCML algorithm between mitochondrial DNAs.
Mateo Muñoz	24/02/21	I found the complexity of the SCML algorithm.		Generate a graph on the SCML and analyze the results.
Alejandro Torres	24/02/21	I analyzed and answered about the feasibility of the algorithm.		
Mateo Muñoz	25/02/21	We generate a graph about the SCML.	Check Alejandro's answer.	Analyze the results of the graph.
Alejandro Torres	25/02/21		Check the complexity found by Mateo.	Explain how GroupSum5 works.
Mateo Muñoz	25/02/21	I analyzed the results of the graph.		Explain the variables of the complexity of the online exercises.
Alejandro Torres	25/02/21	I explained how GroupSum5 works.		
Mateo Muñoz	25/02/21	We explain the complexity of the exercises.	I analyzed Alejandro's answer.	Begin partial mock exercises.

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

Alejandro Torres	25/02/21		I analyzed the results obtained by Mateo.	
Mateo Muñoz	26/02/21	We finished the partial mock exercises.		Finalize delivery details.
Alejandro Torres	26/02/21			
Mateo Muñoz	27/02/21	We finalize delivery details.		DONE.
Alejandro Torres	27/02/21			

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