Laboratory practice No. 1: RECURSION

David Calle Gonzales
Universidad Eafit
Medellín, Colombia
dcalleg@eafit.edu.co

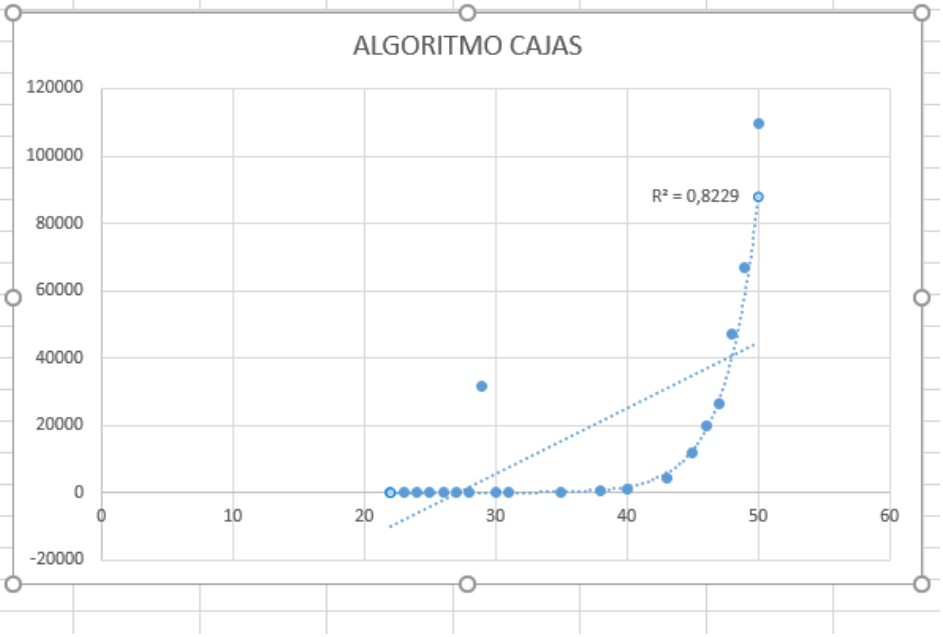
Julian Ramirez Giraldo
Universidad Eafit
Medellín, Colombia
jramirezg@eafit.edu.co

3) Practice for final project defense presentation

3.1. The asymptotic complexity given for exercise 1.2 can be described with this formula: T(n) = T(n-1) + T(n-2) + C. Processed by Wolfram app to obtain the solution of this formula we got: t(n) = c_1 2^(n - 1) + C (2^n - 1). Therefore we can conclude that working and analyzing this equation the asymptotic complexity of this algorithm is O(2^n).

3.2 Graph and data table for the algorithm:

22	1
23	2
24	2
25	2
26	8
27	4
28	6
29	31630
30	91
31	10
35	105
38	394
40	1006
43	4572
45	12028
46	19904
47	26397
48	47261
49	66724
50	109584



Estimated time for the algorithm to calculate the shapes for a size 50x2:

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

For a size of 50x2 cm squared, the time it takes to calculate the ways in which rectangles can be placed is 109584 milliseconds, equivalent to 1.8264 minutes.

3.3 It would not work; the complexity makes the algorithm take time to process the information while the area of the rectangle goes up. It would be necessary to find a way to develop another algorithm with a lower complexity that could allow a faster data processing.

3.4 Algorithm:

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

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

“Given an array of ints, is it possible to choose a group of some of the ints, such that the group sums to the given target with these additional constraints: all multiples of 5 in the array must be included in the group. If the value immediately following a multiple of 5 is 1, it must not be chosen. (No loops needed.)" -CodingBat

the parameters are an initial position, an array of integers and a target that must be equal to the sum of certain integers of the int array.

first, the stop condition is whether the initial position or position to evaluate is equal to the length of the ints array, in that case, if we reached our target, then the algorithm will return true, otherwise, it will return false.

now, if the integer in the position is not divisible by 5, then the algorithm will evaluate two cases: the first, substracting the int in the initial position from the target and increasing the initial position in 1, and the second, only increasing the initial position in 1.

if the integer in the position is divisable by 5, there are 2 cases, if the next integer in the array is the number 1 (taking into account that we are not in the last position of the array), the 1 will not be used and the algorithm will run again, substracting the integer in the position from the target and increasing by 2 the initial position. the other case is just if the integer is just divisable by 5, hence, it will substract the integer from the target and will run again with the initial position increased in 1.

3.5

- **Recursion 1:**
  - Algorithm Count7:

```
public int count7(int n) {
int aux = 0;
if(n<7){
return 0;
}else{

  if(n%10 == 7){
    aux = 1;
  }else{
    aux = 0;
  }

  return aux + count7(n/10);

}
}
```

The asymptotic complexity of this algorithm is given by:
T(n) = T(n-1) + C
T(n) = c1 + C

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

complexity = O(n)
“n” is the number from which the 7´s will be counted

- Algorithm CountX:

countX

```
public int countX(String str) {
if (str.equals("")){
return 0;
}else if(str.charAt(0)=='x'){
return 1+countX(str.substring(1));
}
return countX(str.substring(1));
}
```

T(n) = T(n-1) + C
T(n) = c1 + C
complexity =O(n)
“n” is the string length

- Algorithm PowerN:

```
public int powerN(int base, int n) {
return n == 0 ? 1: base * powerN(base,n-1);
}
```

The asymptotic complexity of this algorithm is given by:
T(n) = T(n-1) + C
T(n) = c1 + C
complexity = O(n)
“n” is the exponent

- Algorithm noX:

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

```
public String noX(String str) {
if (str.length() == 0)
{
return "";
}
else if (str.charAt(0) == 'x')
{
return noX(str.substring(1, str.length()));
}
else
{
return str.substring(0,1) + noX(str.substring(1, str.length()));
}
}
```

The asymptotic complexity of this algorithm is given by:
T(n) = T(n-1) + C
T(n) = c1 + Cn
complexity = O(n)
“n” is the string length

- Algorithm bunnyEars2:

```
public int bunnyEars2(int n) {
if(n == 0)
{
return 0;
}
if(n % 2 == 0)
{
return 3 + bunnyEars2(n-1);
}
else
{
return 2 + bunnyEars2(n-1);
}

}
```

The asymptotic complexity of this algorithm is given by:
T(n) = T(n-1) + C
T(n)) c1 + Cn
complexity = O(n)
“n” is the quantity of bunnies

- **Recursion 2:**

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

- Algorithm splitArray:

```
public boolean splitArray(int[] nums) {
return help(nums, 0, 0, 0);
}

public boolean help(int[] nums, int g1, int g2, int ind)
{
if(nums.length == ind)
{
return (g1 == g2);
}
return help(nums, g1 +nums[ind], g2, ind+1) || help(nums, g1, g2+nums[ind],
ind+1);
}
```

The asymptotic complexity of this algorithm is given by:
T(n) = T(n-1) + T(n-1) + C
T(n) = c1*2^(n-1) + C(2^n - 1)
complexity = O(2^n)
"n" is the quantity of elements in the array that have not been used yet

- Algorithm split53:

```
public boolean split53(int[] nums) {
return help(nums, 0, 0, 0);
}

public boolean help(int[] nums, int g1, int g2, int ind)
{
if(nums.length == ind)
{
return (g1 == g2);
}
if(nums[ind] % 5 == 0)
{
return help(nums, g1+nums[ind], g2, ind+1);
}
else if(nums[ind] % 3 == 0)
{
return help(nums, g1, g2 + nums[ind], ind+1);
}
return help(nums, g1 +nums[ind], g2, ind+1) || help(nums, g1, g2+nums[ind],
ind+1);
}
```

The asymptotic complexity of this algorithm is given by:
T(n) = T(n-1) + T(n-1) + C
T(n) = c1*2^(n-1) + C(2^n - 1)

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

complexity = O(2^n)
"n" is the quantity of elements in the array that have not been used yet

- Algorithm splitOdd10:

```
public boolean splitOdd10(int[] nums) {
return help(nums, 0, 0, 0);
}

public boolean help(int[] nums, int m10, int odd, int ind)
{
if(nums.length == ind)
{
return (m10 %10 == 0 && odd%2 != 0);
}
return help(nums, m10+nums[ind], odd, ind+1) || help(nums, m10,
odd+nums[ind], ind+1);
}
```

The asymptotic complexity of this algorithm is given by:
T(n) = T(n-1) + T(n-1) + C
T(n) = c1*2^(n-1) + C(2^n - 1)
complexity = O(2^n)
"n" is the quantity of elements in the array that have not been used yet

- Algorithm groupNoAdj:

```
public boolean groupNoAdj(int start, int[] nums, int target) {
if (nums.length <= start)
{
if(target == 0)
{
return true;
}
return false;
}

return groupNoAdj(start+2, nums, target-nums[start]) || groupNoAdj(start+1,
nums, target);
}
```

The asymptotic complexity of this algorithm is given by:
T(n) = T(n-2) + T(n-1) + C
T(n) = c1Fn + c2Ln + C
complexity = =(2^n)
"n" is the quantity of elements in the array that have not been used yet

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

- Algorithm groupSum6:

```
public boolean groupSum6(int start, int[] nums, int target) {
if (nums.length == start)
{
if(target == 0)
{
return true;
}
return false;
}
if (nums[start] != 6)
{
return groupSum6(start+1, nums, target-nums[start]) || groupSum6(start+1,
nums, target);
}
else
{
return groupSum6(start+1, nums, target-nums[start]);
}

}
```

The asymptotic complexity of this algorithm is given by:
T(n) = T(n-1) + T(n-1) + C
T(n) = c1*2^(n-1) + C(2^n - 1)
complexity = O(2^n)
"n" is the quantity of elements in the array that have not been used yet

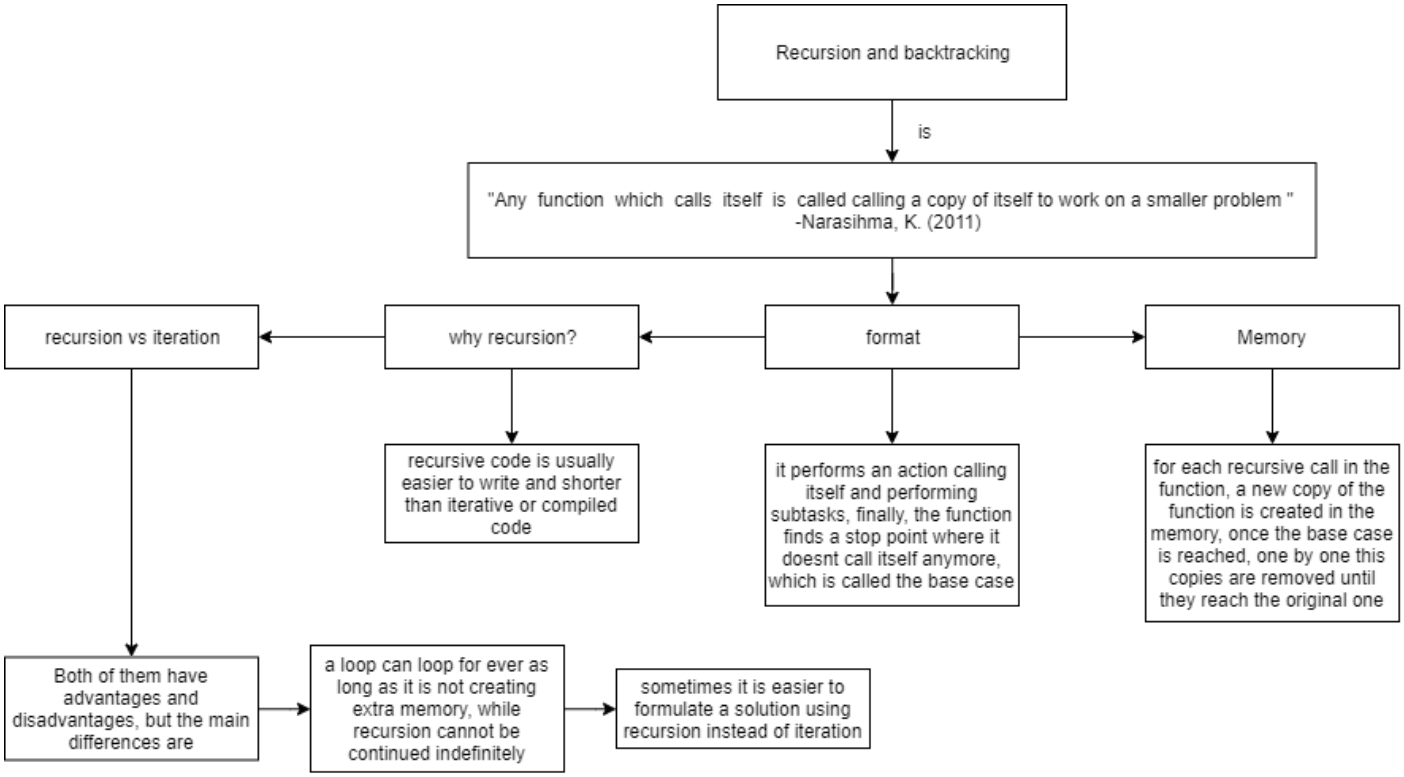
4) Practice for midterms

4.1 (start + 1, nums, target)
4.2 a
4.3.1 (n-a,a,b,c)
4.3.2 (res,solucionar(n,b,c,n+1))
4.3.3 (res,solucionar(n,c,n+1,n+1))
4.4 e
4.5.1 L2: return 1; // L3: n-2 // L4:n-4
4.5.2 b
4.6.1 sumaAux(n, i+2)
4.6.2 sumaAux(n, i+1)
4.7.1 S, i+1, t-S[i]
4.7.2 S, i+1, t

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

4.8.1 return 0
4.8.2 ni + nj
4.9 c.22
4.10 b. 6
4.11.1 (n-1) // lucas(n-2)
4.11.2 c. O(2^n)
4.12.1 sat
4.12.2 Math.max(fi,fj)
4.12.3 sat

5) Recommended reading (optional)



Narasimha, K. (2011) Data Structures And Algorithms. Recuperado de: https://www.docdroid.net/ZPfHmS5/data-structures-and-algorithms-narasimha-

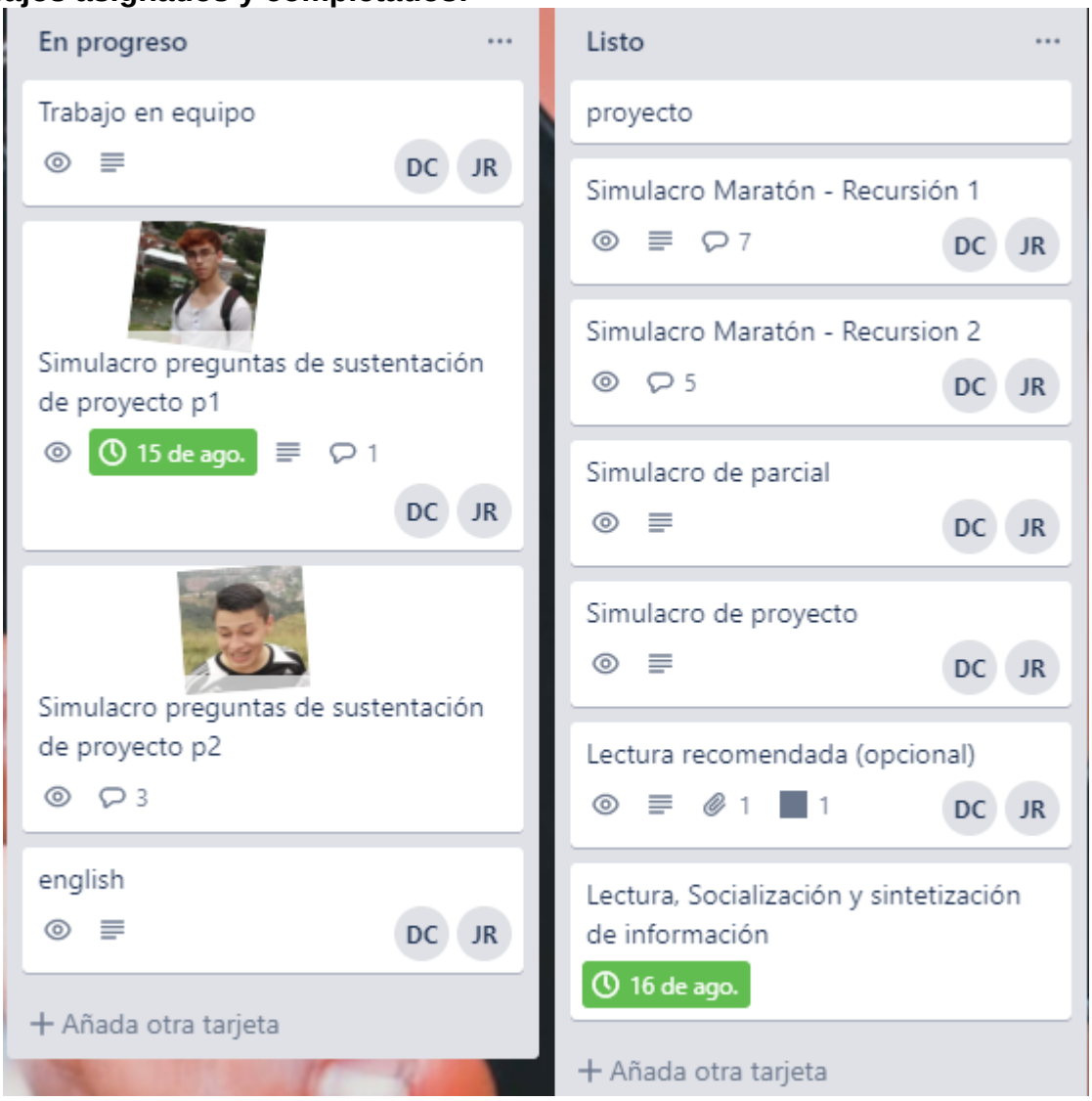
6) Teamwork and gradual progress (optional)

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

Reuniones:

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

Trabajos asignados y completados:



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