Laboratory practice No. 1: Recursion

Cristian Alexis Giraldo Agudelo
Universidad Eafit
Medellín, Colombia
cagiraldoa@eafit.edu.co

Daniel Alejandro Cifuentes Londoño
Universidad Eafit
Medellín, Colombia
dacifuentl@eafit.edu.co

3) Practice for final project defense presentation

3.1 Asymptotic complexity for the worst case of exercise 1

```
public static int nRectangulos(int n){
        if(n<=3)
            return n;
        else
            return nRectangulos(n-1)+nRectangulos(n-2);
    }
```

For the previous algorithm we will initially calculate its complexity using the following equation:

T(n) = T(n-2) + T(n-1) + C

Later we enter **WolframAlpha** but with a modification (***T(n) = T(n-1) + T(n-1) + C).***

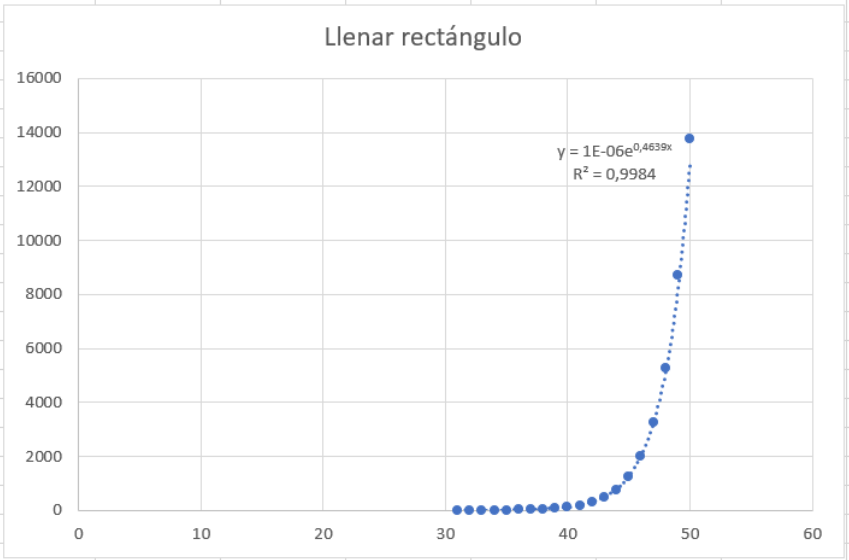
The result was as follows: $T(n) = c_1 2^{n-1} + C (2^n - 1)$

And in notation **O** through the calculations = O(2^n)

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

3.2 Graphic for 20 different sizes

n*2	Tiempo
31	2
32	4
33	5
34	6
35	11
36	22
37	35
38	45
39	69
40	128
41	189
42	303
43	483
44	772
45	1248
46	2016
47	3269
48	5281
49	8724
50	13753



We can appreciate the right decision in calculating the complexity previously because in the graph we appreciate an exponential trend line, which corresponds to that made in notation **O**.

For the calculation of time for the size of **50x2** can be seen through the code and stipulated in the graph of **13753** milliseconds.

3.3 Feasibility of using the algorithm in Puerto Antioquia

It would not be prudent. An exponential complexity is very incorrect to use in situations where we face exorbitant amounts. For example, for the size of **1000x2**, a complexity of incomparable quantities would be presented.

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 GroupSum5

```
public boolean groupSum5(int start, int[] nums, int target) {

if (start >= nums.length)
return target == 0;

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]) ;

}
return groupSum5(start + 1, nums, target - nums[start]) || groupSum5(start + 1, nums, target);
}
```

To explain the procedure of the previous code, I will initially explain what you are looking to do. A subset of elements must be found in the array that added to the **target**. But in addition to this, there are restrictions, all elements that are multiples of **5** must be used and those elements equal to (**1**) and previously this one (**5**) cannot be used.

The code does the following:

- As a stop method, a conditional is created that identifies when the entire array has been traversed. At that time, it is asked if the **target** was made **0**, that is, if there were the exact elements to be subtracted from the original **target,** they would give **0**. After, returns **true** or **false** depending on its compliance.
- To reach the previous conclusion, we present an algorithm where we determine that elements that are multiple of **5** are evaluated separately. There, we will begin to perform combinations of restoration with all these numbers to reach the comet.
- Additionally, as **if** nested from the above we have a restriction for the **5** followed by 1. To achieve not including the 1 what we do is advance from 2 positions in the array.
- Finally, we create a section for those that are not like any of the last things. There it is sought to complete the subset if necessary.

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

3.5

Recursion 1

```
public int bunnyEars(int bunnies) {
  if(bunnies == 0)
    return 0;
  else
    return 2 + bunnyEars(bunnies-1);
}
```

$T(n) = C + T(n-1)$
$T(n) = C_1 + C\ n$

$O(n)$

```
public int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 2) + fibonacci(n - 1);
}
```

$T(h) = C + T(h-1) + T(h-2)$
$T(h) = c_1\ 2^{h-1} + C\ (2^h - 1)$

$O(2^h)$

```
public int triangle(int rows) {
  if (rows == 0) {
    return 0;
  }
    return rows + triangle(rows - 1);
}
```

$T(i) = C + T(i-1)$
$T(i) = C_1 + C\ i$

$O(i)$

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 int sumDigits(int n) {
    if (n == 0) {
      return 0;
    }
    return n % 10 + sumDigits(n / 10);
}
```

$T(w) = C + T(w/10)$

$T(w) = c_1 + \frac{C\ log(w)}{log(10)}$

$O\ (log\ (w))$

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

$T(m) = C + T(m-1)$

$T(m) = C_1 + C\ m$

$O(m)$

Recursion 2

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

$T(o) = C + 2T(o-1)$

$T(o) = c_1\ 2^{o-1} + C\ (2^o - 1)$

$O(2^o)$

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 boolean groupNoAdj(int start, int[] nums, int target) {
  if(start >= nums.length){
    return (target==0);
  }
   return (groupNoAdj(start+2, nums, target - nums[start])) || (groupNoAdj(start+1, nums, target));
}
```

T(o) = C + 2T(o-1)
$T(o) = c_1 2^{o-1} + C (2^o - 1)$

$O(2^o)$

```
public boolean groupSum5(int start, int[] nums, int target) {

if (start >= nums.length)
return target == 0;

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]) ;

}
return groupSum5(start + 1, nums, target - nums[start]) || groupSum5(start + 1, nums, target);
}
```

T(o) = C + 2T(o-1)
$T(o) = c_1 2^{o-1} + C (2^o - 1)$

$O(2^o)$

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 boolean groupSumClump(int start, int[] nums, int target) {
  if(start >= nums.length){
    return (target==0);
  }
 if(start < nums.length-1 && nums[start]==nums[start+1]){
    return groupSumClump(start+2, nums, target - nums[start]-nums[start+1]) || groupSumClump(start+2, nums, target);
 }
  return groupSumClump(start+1, nums, target - nums[start]) || groupSumClump(start+1, nums, target);
}
```

T(o) = C + 2T(o-1)
$T(o) = c_1 2^{o-1} + C (2^o - 1)$

$O(2^o)$

```
public boolean splitArray(int[] nums) {
    return  split(0, nums, 0, 0);
 }
 public boolean split(int n, int [] nums, int n1, int n2){
  if(n==nums.length)
  return n1 == n2;
  return split(n+1, nums, n1-nums[n], n2) || split(n+1, nums, n1, n2-nums[n]);
 }
```

T(o) = C + 2T(o-1)
$T(o) = c_1 2^{o-1} + C (2^o - 1)$

$O(2^o)$

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

3.6 Variables "n", "m" … in complexity

The previous variables that were used in the complexity equations or any others are those that directly influence the type of equation depending on the complexity of the algorithm presented.

n **=** Number of bunnies
h **=** Succession index
i **=** Rows in the triangle
w **=** Number length
m **=** Power exponent
o **=** Array length

4) Practice for midterms

4.1 Start +1, nums, target
4.2 **b.** T(n)=T(n/2)+C

4.3
4.3.1 *int res = solucionar(n-a,a,b,c) + 1;*
4.3.2 *res = Math.max(res, solucionar(n-c,a,b,c)+1));*
4.3.3 *res = Math.max(res, solucionar(n-b,a,b,c)+1));*
4.4 ***e.*** *La suma de los elementos del arreglo a y es O(n)*
4.5
4.5.1
Línea 2: return n;
Línea 3: formas(n-1) +
Línea 4: (n-2)
4.5.2 ***b.*** *T(n)=T(n−1)+T(n−2)+C*

4.6

4.6.1 return 0;
4.6.2 return (charAt(i) - ‘o’) + (charAt(i + 1) - ‘o’;

4.8
4.8.1 *return 0;*
4.8.2 *int suma = ni + nj;*

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

4.9 **c. 22**
4.10 ***b. 6***

4.11

4.11.1 return lucas(n – 1) + lucas(n – 2)
4.11.2 **c.** T(n)=T(n−1)+T(n−2)+c, que es O(2 elevado a la n)

4.12

4.12.1 return sat;
4.12.2 return sat += Math.max(Fi, Fj);
4.12.3 return sat;

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