ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 1: Recursion

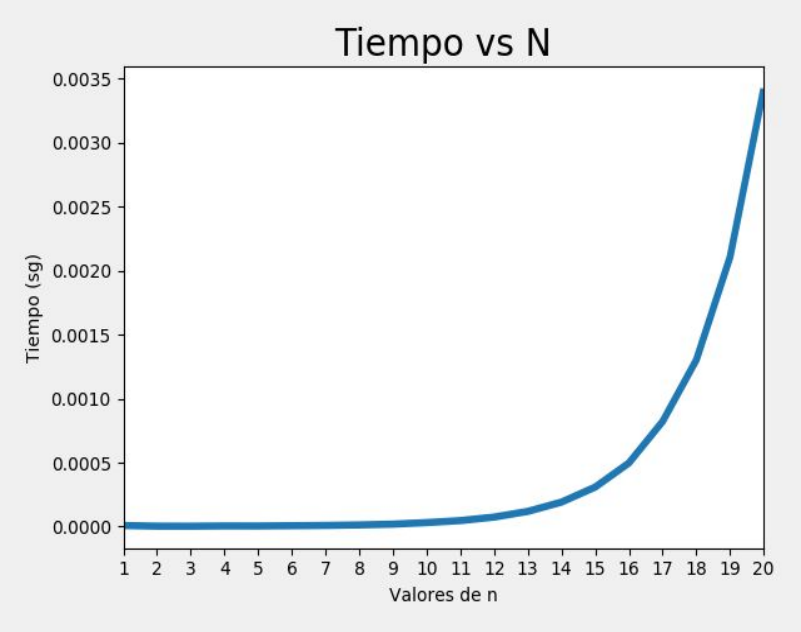
Vincent Alejandro Arcila Larrea
Universidad Eafit
Medellín, Colombia
vaarcilal@eafit.edu.co

Isabel Piedrahita Velez
Universidad Eafit
Medellín, Colombia
ipiedrahiv@eafit.edu.co

3) Practice for final project defense presentation

3.1

The asymptotic complexity is: F(n) = F(n-1) + F(n-2) + c
In big O notation: O(2^n)



3.2

According to big O notation there are 2^{50} instructions to do in order to get with a solution. Since one of my CPU cores performs 798 MHz (798000000 operations per second): $\frac{2^{50}}{798000000}$ = 1410902.14 seconds to find the solution.

3.3

In no way is this algorithm useful when the values of *n* are very large. It is not useful when solves the problem for a *2*n* square, and much less when is a *n*n*n* container. The algorithm efficiency is not so good when the problem sizes become larger.

3.4

When the value of *start* is greater than or equal to the size of *nums* will return the value of *target==0,* and this will be our base case. This will stop the algorithm when we have evaluated all the branches of our tree. In problems like this, the idea is to somehow iter over the array creating 2 cases: one on which we subtract the actual number to the target and another case on which we do not. If we get to our base case and the value of target equals 0 we return *True*. If target does not equals to 0 we return *False.* The way that we iter over the array is by calling *groupSum5* adding 1 to *start* and giving the same

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

array. In the first call we subtract *nums[start]* to target and on the second call we do not. But we have not taken into account the 2 constraints that have been given to us in this exercise. If the value of *nums[start]* equals a multiple of 5 we must subtract it to target, and if the consequent value to a multiple of 5 is one, it cannot be subtracted to *target*. So, when the value of *nums[start]* is not a multiple of 5 we should operate as indicated above. In other cases, if *nums[start]%5* equals 0 it means we have to look for to cases:

- **If it is the last value of the array:** in this case we will call *groupSum5* adding 1 to *start*, giving the same array and subtracting *nums[start]* to *target* in order to get to our base case.
- **In other cases:** we evaluate the value of *nums[start + 1]*. If it equals 1 we evaluate *groupSum5* with *start+2* and subtracting the actual number to *target*. If *nums[start + 1]* is not 1 we evaluate *groupSum5* adding 1 to start and subtracting *nums[start]* to target.

3.5

Recursion 1 : factorial

$T(n) = T(n-1) + c$ or
$T(n) = cn + c_1$ (c_1 *is an arbitrary parameter*)
Function in terms of n, taken from WolframAlpha.

Big O: *O(n)*

```
public int factorial(int n) {
  if(n==0 || n==1){
    return 1;        // c
  }else{
    return n*factorial(n-1);          // T(n − 1) + c
  }
}
```

Recursion 1 : fibonacci

$T(n) = T(n-2) + T(n-1) + c$ or
$T(n) = c_1 F_n + c_2 L_n - c$ (c_1, c_2 *are arbitrary parameters*)
Function in terms of n, taken from WolframAlpha.

Big O: $O(2^n)$

```
public int fibonacci(int n) {
  if(n<=1){
    return n;        // c
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
 }else{
   return fibonacci(n-2)+fibonacci(n-1);  // T(n − 2) + T(n − 1) + c
 }
}
```

Recursion 1 : powerN

$T(n) = T(n-1) + c$ or
$T(n) = c\,n + c_1$ (c_1 *is an arbitrary parameter*)
Function in terms of n, taken from WolframAlpha.

Big O: *O(n)*

```
public int powerN(int base, int n) {
   if(n <= 1){
      return base;           //  c
   }
   return base * powerN(base, n - 1);                //  T(n − 1) + c
}
```

Recursion 1 : sumDigits

$T(n) = T(\frac{n}{10}) + c$ or
$T(n) = \frac{c\,log(n)}{log(10)} + c_1$ (c_1 *is an arbitrary parameter*)
Function in terms of n, taken from WolframAlpha.

Big O: *O(n)*

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

Recursion 1 : triangle

$T(n) = T(n-1) + c$ or

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

$T(n) = c\,n + c_1$ (c_1 *is an arbitrary parameter*)
Function in terms of n, taken from WolframAlpha.

Big O: *O(n)*

```
public int triangle(int n) {
  if(rows==0){
    return 0;        //  c
  }else{
    return n + triangle(n-1);          //  T(n − 1) + c
  }
}
```

Recursion 2 : groupNoAdj

$T(n) = T(n-2) + T(n-1) + c$ or
$T(n) = c_1 F\,n + c_2 L_n - c(c_1, c_2$ *are arbitrary parameters*)
Function in terms of n, taken from WolframAlpha.

Big O: $O(2^n)$

```
public boolean groupNoAdj(int start, int[] nums, int target) {
  if(start >= nums.length) return (target==0); //  c

  if(groupNoAdj(start+2,nums,target-nums[start])){ //  T(n − 2)
    return true;
  }
  if(groupNoAdj(start+1,nums,target)){ //  T(n − 1)
   return true;
  }
  return false;
}
```

Recursion 2 : groupSum5

$T(n) = 2T(n-1) + c$ or
$T(n) = c_1 2^{n-1}$ (c_1 *is an arbitrary parameter*)
Function in terms of n, taken from WolframAlpha.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Big O: $O(2^n)$

```
public boolean groupSum5(int start, int[] nums, int target) {
  if(start >= nums.length) return (target==0); //  c

  if(nums[start]%5==0){  c
    if(start+1<nums.length){
      if(nums[start+1]==1){
        if(groupSum5(start+2,nums,target-nums[start])) return true;
      }
      else{
        if(groupSum5(start+1,nums,target-nums[start])) return true;
      }
    }else{
      if(groupSum5(start+1,nums,target-nums[start])) return true;
    }
  }else{
    if(groupSum5(start+1,nums,target-nums[start])) return true;      //  T(n − 1)
    if(groupSum5(start+1,nums,target)) return true;          //  T(n − 1)
  }
  return false;
}
```

Recursion 2 : groupSum6

$T(n) = 2T(n-1) + c$ or

$T(n) = c_1 2^{n-1}$ (c_1 *is an arbitrary parameter*)

Function in terms of n, taken from WolframAlpha.

Big O: $O(2^n)$

```
public boolean groupSum6(int start, int[] nums, int target) {
  if(start >= nums.length) return (target == 0); //  c
  if(nums[start]!=6){      //  c
    if(groupSum6(start+1,nums,target-nums[start])) return true; //  T(n − 1)
    if(groupSum6(start+1,nums,target)) return true;          //  T(n − 1)
  }else{
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

UNIVERSIDAD EAFIT® Acreditación Institucional Renovación 2018 - 2026 Resolución MEN 2158 de 2018

Inspira Crea Transforma | Vigilada Mineducación | www.eafit.edu.co

```
   if(groupSum6(start+1,nums,target-6)) return true;
  }
  return false;
}
```

Recursion 2 : splitArray

$T(n) = 2T(n-1) + c$ or
$T(n) = c_1 2^{n-1}$ (c_1 *is an arbitrary parameter*)
Function in terms of n, taken from WolframAlpha.

Big O: $O(2^n)$

```
public boolean splitArray(int[] nums) {
  return   fill(0,nums,0,0); //c1
}
public boolean fill(int index,int[] nums,int uno,int dos){
  if(index >= nums.length) return uno==dos; //  c
  return fill(index+1,nums,uno+nums[index],dos) ||   //  T(n − 1)
fill(index+1,nums,uno,dos+nums[index]);          //  T(n − 1)
}
```

Recursion 2 : groupSumClump

$T(n) = 2T(n-1) + c$ or
$T(n) = c_1 2^{n-1}$ (c_1 *is an arbitrary parameter*)
Function in terms of n, taken from WolframAlpha.

Big O: $O(2^n)$

```
public boolean groupSumClump(int start, int[] nums, int target) {
  int cont=1;
  if(start >= nums.length) return target==0;         // c
  if(start+1 < nums.length){        //  c
  if(nums[start] == nums[start+1]){         //  c
    for(int i =start+1;i<nums.length; i++){
      if(nums[i] != nums[start]) break;
      Cont++;       //c6
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
    }
  }
  }

  if(groupSumClump(start+cont,nums,target-nums[start]*cont)) return true;        // T(n − 1)
  if(groupSumClump(start+cont,nums,target)) return true;          // T(n − 1)

  return false;
}
```

3.6

Recursion 1 : factorial

n is the number whose factorial will be calculated by the algorithm.

Recursion 1 : fibonacci

n represents the nth number in the fibonacci sequence.

Recursion 1 : powerN

base represents the number that is going to be elevated.
n represents the power to which the base will be elevated.

Recursion 1 : sumDigits

n represents the input number of which the digits will be added together.

Recursion 1 : triangle

n represents the n^{th} row of a triangle made of wooden blocks.

Recursion 2 : groupNoAdj

start represents the position of the array from which each individual traversion will begin.
target represents the sum that is being looked for in the array.
nums represents the collection of numbers the target is being searched for on.
n in the complexity analysis represents the length of the section of the array that will be traversed.

Recursion 2 : groupSum5

start represents the position of the array from which each individual traversion will begin.
target represents the sum that is being looked for in the array.
nums represents the collection of numbers the target is being searched for on.
n in the complexity analysis represents the length of the section of the array that will be traversed.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

UNIVERSIDAD EAFIT® Acreditación Institucional Renovación 2018-2026 Resolución MEN 2158 de 2018

Inspira Crea Transforma | Vigilada Mineducación | www.eafit.edu.co

Recursion 2 : groupSum6

start represents the position of the array from which each individual traversion will begin.
target represents the sum that is being looked for in the array.
nums represents the collection of numbers the target is being searched for on.
n in the complexity analysis represents the length of the section of the array that will be traversed.

Recursion 2 : groupSumClump

start represents the position of the array from which each individual traversion will begin.
target represents the sum that is being looked for in the array.
nums represents the collection of numbers the target is being searched for on.
n in the complexity analysis represents the length of the section of the array that will be traversed.

Recursion 2 : splitArray

nums represents the collection of numbers on which the algorithm will be used.
n in the complexity analysis represents the length of the subsection of the array that will be traversed.

4) Practice for midterms

4.2 a) T(n) = T(n/2) + c
4.3.1 int res = solucionar(n - a, a, b, c) + 1;
4.3.2 res = Math.max(res, solucionar(n - b, b, a, c) + 1);
4.3.3 res = Math.max(res, solucionar(n - c, c, a, b) + 1);
4.5.1

2. if (n <= 2) return n;
3. return formas(n-1) +
4. return formas(n-2);

4.5.2 b) T(n-1) + T(n+2) + c
4.6.1 10. return sumAux(n, i+2);
4.6.2 12. return (n.charAt(i) - '0' + sumAux(n, i+1));
4.8.1 9. return 0;
4.8.2 int suma = ni + nj;
4.10 6

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473