ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 1
Recursion

María Paulina Ocampo Duque
Universidad Eafit
Medellín, Colombia
mpocampod@eafit.edu.co

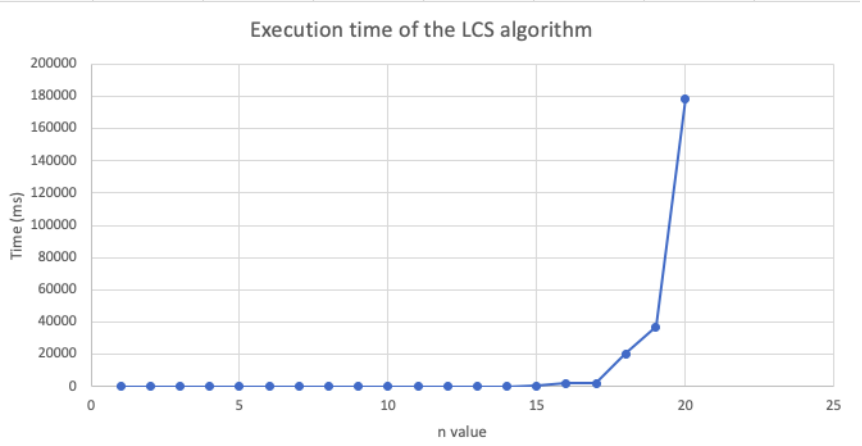
Maria Jose Gutierrez Estrada
Universidad Eafit
Medellín, Colombia
mjgutierre@eafit.edu.co

3) Practice for final project defense presentation

3.1 T(n+m)= c_3 T((m+n)-1)+ T((m+n)-1)
T(n+m)= c_3 T((2^m+n)-1)+ c_1(2^(m+n-1))
T(n+m)= 2^(m+n)
T(n+m)= T(p)
T(p)= O (2^p) complexity

3.2 With these results we can see how data grow exponencially, showing us the worst case of their performance.

n value	Time
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	3
10	1
11	2
12	14
13	38
14	159
15	397
16	2056
17	2110
18	20385
19	36847
20	178575



3.3 It wouldn´t be an optimal and recommended option because being an exponencial function means that it complexity is going to double in every single value. For that reason, find the DNA subsecuence algorithm would take to many time because each DNA sequence has approximately 300.000 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

3.4 How groupSum5 works?

It receives an array and execute the condition to verify there is a number multiple of 5. If the number is a multiple jump into the next condition that verify there is a numer 1, then add the multiple of 5 and omit the 1. If the number is not a multiple of 5, rejects the number. At the end we use a two recursive call to continue with the next number if the conditions are correct

3.5
CODINGBAT

RECURSION-1

- **Bunny Ears**

```
public int bunnyEars(int bunnies) {

 if(bunnies==0){                    //c1
   return bunnies;                  //c2
 }else{
   return 2+bunnyEars(bunnies-1);   // t(n)=c2 + t(n-1)
 }
}
```

Model:

$$T(n)= \begin{cases} 1 & \text{if } n==0 \\ 2+ t(n-1) & \text{if } n > 0 \end{cases}$$

Recurrence equation solution :

t(n) = c_2 * n + c_1

Calculate complexity:

O(C _2*n + c1)
O(C_2 * n) // By addition rule
O(n) //By multiplication rule

- **Fibonacci**

```
public int fibonacci(int n) {                 //
  if(n<=1)                                    //c1
    return n;                                 //c2
    return fibonacci(n-1) + fibonacci(n-2);   //t(n)= t(n-1)+t(n-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

Model :

$$T(n)=\begin{cases} c1+c2 & \text{if } n<=1 \\ T(n-1)+t(n-2) & \text{if } n>=2 \end{cases}$$

Recurrence equation solution :
t(n) = c_1 * 2^n + c_2

Calculate complexity:
O (C1 * 2^n + c2)
O (C1 * 2^n) // By “Laboratory practice No. 1: Recursion”point 4.4.1
O (2^n)

- **BunnyEars2**

```
public int bunnyEars2(int bunnies) {
  if(bunnies==0)
    return bunnies;                          // c1
  if (bunnies%2==1)
  return 2+bunnyEars2(bunnies-1);            // c2=2+t(n-1)
    return 3+bunnyEars2(bunnies-1) ;         // t(n) = c2 + t(n - 1)
}
```

Model :

$$T(n)=\begin{cases} 0 & \text{if } n==0 \\ 2+t(n-1) & \text{if } n==1 \\ 3+t(n-1) & \text{if } n>=2 \end{cases}$$

Recurrence equation solution :

t(n) = c_1 + 3 n

Calculate complexity:

O(C_1 + 3 * n)
O(3*n) // By addition rule
O(n) //By multiplication rule

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

- **AllStar:**

```
public String allStar(String str) {
  if(str.length() <=1)                                        //c1
  return str;                                                 //c2
   return str.charAt(0) + "*" + allStar(str.substring(1));  //t(n)=c2+t(n-1)
}
```

Model :

$$T(n)= \begin{cases} c1+c2 & \text{if n.lenght<=1} \\ c2+t(n-1) & \text{if n.lenght>=2} \end{cases}$$

Recurrence equation solution :
t(n) = c_2 * n + c_1

Calculate complexity:
O (c_2 *n + c_1)
O (c_2 * n) // By addition rule
O (n) //By multiplication rule

- **Triangle**

```
public int triangle(int rows){
   if(rows <=1)                          //c1
   return rows;                          //c2
   return rows+triangle(rows-1);         //t(n)=c2+t(n-1)
}
```

Model :

$$T(n)= \begin{cases} 1 & \text{if n<=1} \\ C2+t(n-1) & \text{if n>=2} \end{cases}$$

Recurrence equation solution :
t(n) = c2 n + c_1

Calculate complexity:
O (c2* n + c1)
O (c2 * n) // By addition rule
O (n) //By multiplication rule

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

- **groupSum5**

```
public boolean groupSum5(int start, int[] nums, int target) {
  if(start==nums.length)
     return target==0;                                                    //c1
  if(nums[start] % 5 == 0){
     if(start<nums.length-1 && nums[start+1]==1)
       return groupSum5(start+2,nums,target-nums[start]);                 //c2
  else
    return groupSum5(start+1,nums,target-nums[start]);                    //c3+t(n-2)
  }
     ReturngroupSum5(start+1,nums,target-nums[start])||groupSum5(start+1,nums,target);
//c4+2(tn-1)
}
```

Recurrence equation solution:
$T(n)= c_4 (2^n - 1) + c_1 2^(n - 1)$

Calculate complexity:

```
O(c_4 (2^n - 1) + c_1 2^(n – 1))
O(c_4(2^n )+ c_1 2^n)              //By addition rule
O((2^n )+ 2^n)                     //By multiplication rule
O(2*  2^n)                         //By addition rule
O(2^n)                             //By multiplication rule
```

- **groupSum6**

```
  public boolean groupSum6(int start, int[] nums, int target) {
  if (start >= nums.length)
        return target == 0;                                              //C1
      if (groupSum6(start + 1, nums, target - nums[start]))
        return true;                                                     //C2 + T(n-2)
      if (nums[start] == 6)
        target -= 6;
      return groupSum6(start + 1, nums, target);                         //C3+2t(n-1)
}
```

Recurrence equation solution:
$T(n)= c3(2^n -1)+ c_1 2^n-1$

Calculate complexity:

```
O(c3(2^n -1)+ c_1 2^n-1)
O(c3(2^n )+ c_1 2^n)       //By addition rule
O(2^n )+ 2^n)              //By multiplication rule
O(2*  2^n)                 //By addition rule
O(2^n)                     //By multiplication rule
```

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

- **groupNoAdj**

```
public boolean groupNoAdj(int start, int[] nums, int target) {
  if (start >= nums.length)
    return (target == 0);                                    //c
  if (groupNoAdj(start+1, nums, target))
    return true;                                             //c+t(n-1)
  if (groupNoAdj(start+2, nums, target-nums[start]))
  return true;                                               //c2+t(n-1)+t(n-2
  return false;
}
```

Recurrence equation solution:
T(n)= c_1 ***** 2^n + c_2

Calculate complexity:
O (C1 * 2^n + c2)
O (C1 * 2^n) // By “Laboratory practice No. 1: Recursion”point 4.4.1
O (2^n)

- **SplitOdd10**

```
public boolean splitOdd10Aux(int start, int[] nums, int mult, int odd) {
    if(start >= nums.length)
        return mult % 10 == 0 && odd % 2 == 1;                  //c
    if(splitOdd10Aux(start+1, nums, mult + nums[start], odd))   //c+(tn-1)
        return true;
    if(splitOdd10Aux(start+1, nums, mult, odd + nums[start]))   //c2+2t(n-1)
        return true;
    return false;
}
```

Recurrence equation solution:
t(n) = c_2 (2^n - 1) + c_1 2^(n - 1)

Calculate complexity:
O(c2(2^n -1)+ c_1 2^n-1)
O(c2(2^n)+ c_1 2^n) //By addition rule
O(2^n)+ 2^n) //By multiplication rule
O(2* 2^n) //By addition rule
O(2^n) //By multiplication rule

- **Split53**

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

    public boolean splitAux(int[] nums, int prefix, int sum1, int sum2){
```

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

```
    if(prefix==nums.length)return sum1==sum2;
    if(nums[prefix]%5==0){                                          //c1
       return splitAux(nums, prefix+1, sum1, sum2+nums[prefix]);
    }else if(nums[prefix]%3==0){                                    //c2+t(n-1)
       return splitAux(nums, prefix+1, sum1+nums[prefix], sum2);
    }else{
       return splitAux(nums, prefix+1, sum1, sum2+nums[prefix])||
       splitAux(nums, prefix+1, sum1+nums[prefix], sum2);           //c3+2t(n-1)
    }
  }
```

Recurrence equation solution:
T(n)= c3(2^n -1)+ c_1 2^n-1

Calculate complexity:
O(c3(2^n -1)+ c_1 2^n-1)
O(c3(2^n)+ c_1 2^n) //By addition rule
O(2^n)+ 2^n) //By multiplication rule
O(2* 2^n) //By addition rule
O(2^n) //By multiplication rule

3.6

- **BunnyEars:** The variable **Bunnies** sets the length and complex of the problema**, n** represents the number of bunnies
- **Fibonacci:** The variable **n** is the Fibonacci number to calculate
- **BunnyEars2:** The variable **Bunnies** represents the number of bunnnies
- **AllStar:** The **str** is a string the algorithm needs to separate by * all the letters
- **Triangle:** The variable **rows** sets the amount of blocks we need to complete the triangle
- **Groupsum5:** The variable n is the size of the array
- **Groupsum6:** The variable n is the size of the array
- **Groupnoadj:** The variable n is the size of the array
- **SplitOdd10:** The variable n is the size of the array
- **Split53:** the variable n is the size of the array

4) Practice for midterms

4.1 *1)a*
2)c
3)a
4.2 *1)b*
2)a-c
4.3 *b*
4.4 *c*
4.5 *1) a*

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

2) b

4.6 *1) Complete la línea 10=*
return sumaAux(n, i+1);
2) Complete la línea 12=
return (n.charAt(i)- '0' + sumaAux(n, i+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