ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No:1

Nayibe Tatiana Vélez David
Universidad Eafit
Medellín, Colombia
ntvelezd@eafit.edu.co

Alejandro Fernández Restrepo
Universidad Eafit
Medellín, Colombia
afernander@eaft.edu.co

3) Practice for final project defense presentation

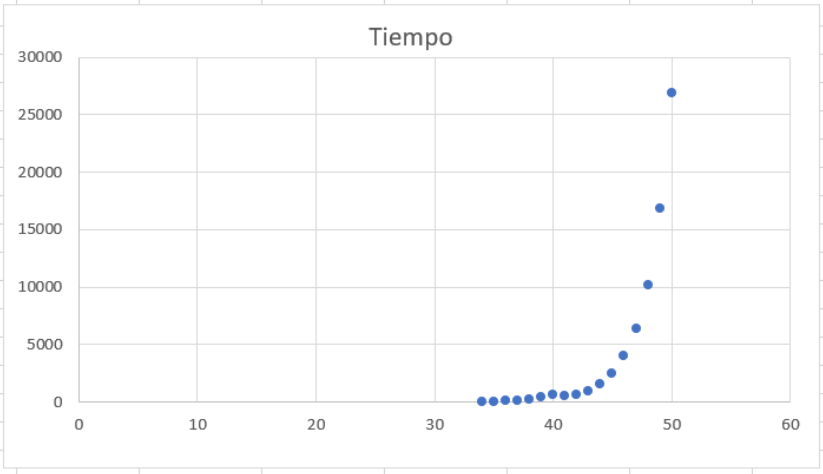
3.1 The asymptotic complexity of exercise 1.2:

First we calculate the complexity in T (n): C1+T(n-2)+ T(n-1).
Then we calculate the same equation for n-1 for a better calculation : T (n): C1+T(n-1)+ T(n-1).
The solution in Wolfram is: T(n): T(n) = c1 2n-1 + C (2n – 1).
The asymptotic complexity: O(2^n).

3.2

We can see how the time in an algorithm is determined and the time it takes to make that case. In this graph you can determine for 20 different sizes the time it takes for the agorithm; also, for a rectangle of 50x2 cm, a time of 26873 seconds will be delayed.

31	5
32	6
33	10
34	13
35	21
36	74
37	156
38	270
39	411
40	588
41	523
42	596
43	966
44	1542
45	2459
46	3988
47	6335
48	10198
49	16826
50	26873



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

3.3 Flexibility of use:

This use would not be of good functionality because this leads to exponential complexity, which incurs a malfunction

.

3.4 Explication of the algorith GroupSum5

Groupsum5 is an algorithm that, given an array of integers and a target number, the algorithm will add all possible combinations of the array numbers to see if there is a combination that adds the target number, with the proviso that if there is a number that be a multiple of 5 will have to be included if or if in the sum to find the combination given by the target number and also if after the number multiple of 5 there is a 1 this will not be taken into account for the sum. If the condition is met and there is a subset of numbers that give a sum of the target number, the algorithm will return true otherwise it will return false.

Functioning
The algorithm has as inputs an integer "start" that will serve as a counter of the position of the arrangement, the arrangement with the numbers and an integer "target" which is the target number to be evaluated.
First, the algorithm verifies that it has not reached the end, comparing start with the length of the arrangement, if this condition is met, it will be verified that if there is a subgroup that added to the target number being if target is equal to 0, the add and return true, otherwise false returns, this will be the stop condition of the algorithm so that it has a point to reach.
The algorithm to know that if the sum has been found in the array that of the first objective number it does is find the numbers multiples of 5, for this the numbers in all the positions of the array are evaluated and its module is extracted with 5 and if this is equal to 0 the number in that position of the array will be subtracted from the target and a recursive call will be made to the function with the new target value with the position to be evaluated of the array increased by 1 so that it does not return to Count the values already evaluated. At the same time, if the number is a multiple of 5, the aforementioned will be evaluated and also if the number that follows is equal to 1, if this happens, the multiple number of 5 will be subtracted from the target to find the sum and start will not increase in 1 if not two in the next recursive call so that both the number already added and 1 are not taken into account in the next call.

At the end of the algorithm, the possible combinations for the sum of the arrangement will be looked for, taking into account all the numbers by making a recursive call by subtracting the current number from the position, given by the start, and adding 1 to start, if this is true, it returns true resulting in a possible combination where the above conditions are met,

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

otherwise a recursive call is returned that will be false since a combination that meets the conditions and the sum of the target number was not found.

3.5
Recursion1

- public int factorial(int n)
  if(n <= 1)
  return 1;
  else
  return n * factorial(n-1);
  }

The complexity of this algorithm is:
T(n): C1+T(n-1)
T(n) = C1 + C n
O(n)

- public int bunnyEars(int bunnies) {
  if(bunnies == 0)
  return 0; // C1
  else
  return 2 + bunnyEars(bunnies-1);
  }

The complexity of this algorithm is:
T(n) = C + T(n-1)
T(n) = C1 + C n
O(n)

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

The complexity of this algorithm is:
T(n) = C1 + T(n-2) + T(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

T(h) = c1 2 n-1 + C (2 n – 1)
O(2 n)

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

The complexity of this algorithm is:
T(n) = C1 + T(n - 1)
T(i) = C1 + C n
O(n)

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

The complexity of this algorithm is: T(n)=
C1 + T(n /10)
T(n) = c1 + $\boldsymbol{C\ log}$(n) $\boldsymbol{log}(\mathbf{10})$
O (log (n))

Recursion2

- 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);

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

```
}
```

The complexity of this algorithm is:
T(n) = C + 2T(n-1)
T(n) = c1 2 ^ n-1 + C (2 ^ n – 1)
O(2 ^ n)

- 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));
  }

The complexity of this algorithm is:
T(n) = C + 2T(n-1)
T(n) = c1 2 ^n-1 + C (2 ^n – 1)
O(2 ^ n)

- 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);
  }

The complexity of this algorithm is
T(n) = C + 2T(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

T(n) = c1 2 ^n-1 + C (2 ^n – 1)
O(2 ^ n)

- 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);
  }

The complexity of this algorithm is:
T(n) = C + 2T(n-1)
T(n) = c1 2 ^n-1 + C (2 ^n – 1)
O(2 ^ n)

- 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]);
  }
  }

The complexity of this algorithm is:
T(n) = C + 2T(n-1)
T(n) = c1 2 ^n-1 + C (2 ^n – 1)
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

3.6 Explanation of the variables:

Start: it is an integer that will increase to reach each position of the arrangement
Objective: it is an integer that contains the value that you want to find in the sum of the subgroups.

Complexity variables
N: it is the length of the array which will determine how long the algorithm will take to execute.

Constants c: are the number of actions that have to be done in that part of the algorithm in which it depends on N.

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;

4.9 c. 22

4.10 b. 6

4.11

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

4.11.1 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