Algorithmics	Student information	Date	Number of session
	UO: 276903	15/03/21	4
	Surname: Garriga Suárez		
	Name: Carlos		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Basic recursive models

1. Complexity of the existent classes.

1.1. Division1.java

In this class we have the following parameters: a = 1, b = 3 and k = 1. Then we can conclude that a < b^k and the formula is O(n^k) so we can conclude that the time complexity is O(n).

1.2. Division2.java

In this class we have the following parameters: a = 2, b = 2 and k = 1. Then we can conclude that a = b^k and the formula is O (n^k logn) so we can conclude that the time complexity is O(nlogn).

1.3. Division3.java

In this class we have the following parameters: a = 2, b = 2 and k = 0. Then we can conclude that a > b^k and the formula is O ($n^{\log_b a}$) so we can conclude that the time complexity is O($n^{\log_2 2}$) = O(n).

1.4. Subtraction1.java

In this class we have the following parameters: a = 1, b = 1 and k = 0. Then we can conclude that a = 1 and the formula is O (n^{k+1}) so we can conclude that the time complexity is O(n).

1.5. Subtraction2.java

In this class we have the following parameters: a = 1, b = 1 and k = 1. Then we can conclude that a = 1 and the formula is O (n^{k+1}) so we can conclude that the time complexity is O(n^2).

1.6. Subtraction3.java

In this class we have the following parameters: a = 2, b = 1 and k = 0. Then we can conclude that a > 1 and the formula is O ($a^{n \text{ div } b}$) so we can conclude that the time complexity is O(2^n).

2. Write the code for the new classes.

2.1. Divide and conqueror by subtraction.

For the new class Substraction4.java, I was asked to develop a method whose complexity was O($3^{n/2}$). To do that we need to deal with divide and conqueror by subtraction.

Escuela de Ingeniería Informática

The first requirement is that we need a number of recursive calls to the method itself higher than one. This is because we need to get a complexity with the following shape O (a $^{n \text{ div } b}$).

If we know this then ‘a’ must be 3, and b must be 2, being ‘a’ the number of calls. ‘B’ is a constant where all the subproblems will have a size (n - b).

2.2. Divide and conqueror by division.

For the new class Division4.java, I was asked to develop a method whose complexity was O(n^2). To do that we need to deal with divide and conqueror by division.

We have to meet the following requirement: **O(n^k)** **if a < b^k**

As the requested number of subproblems was 4, we have that a = 4. Then b^k must be higher than a. I choose b to be 3 so the size of the subproblems would be (n/3). Then I wrote two nested loops outside the calls of the method in order to have k = 2. At the end we have that a = 4, b = 3 and k = 2 so we meet the requirement, and we have a quadratic complexity.