Algorithmics	Student information	Date	Number of session
	UO: 258220	04-03-21	3.1
	Surname: Cuesta Martínez		
	Name: Miguel		

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Basic recursive models

1.1. Complexity of the provided classes

Substraction1:

Applying Divide and Conquer by substraction, we identify a=1, b=1 and k=0.

For the case a = 1, the complexity is $O(n^{k+1})$; that is, O(n).

Substraction2:

Applying Divide and Conquer by substraction, we identify a=1, b=1 and k=1.

For the case a = 1, the complexity is $O(n^{k+1})$; that is, $O(n^2)$.

Substraction3:

Applying Divide and Conquer by substraction, we identify a=2, b=1 and k=0.

For the case a > 1, the complexity is $O(a^{n \text{ div } b})$; that is, $O(2^n)$.

Division1:

Applying Divide and Conquer by division, we identify a=1, b=3 and k=1.

For the case $a < b^k$, the complexity is $O(n^k)$; that is, O(n).

Division2:

Applying Divide and Conquer by division, we identify a=2, b=2 and k=1.

For the case $a = b^k$, the complexity is $O(n^k \log n)$; that is, O(n log n).

Division3:

Applying Divide and Conquer by division, we identify a=2, b=2 and k=0.

For the case $a > b^k$, the complexity is $O(n^{\log_b a})$; that is, O(n).

Algorithmics	Student information	Date	Number of session
	UO: 258220	04-03-21	3.1
	Surname: Cuesta Martínez		
	Name: Miguel		

1.1. Complexity of the provided classes

Substraction4:

The conditions are the following:

- Recursion by substraction must be used.
- The final complexity must be $O(3^{n/2})$.

Out of all the cases in Divide and Conquer by substraction, the one that fits this result the most is complexity $O(a^{n \text{ div } b})$ for situations where $a > 1$. As such, to fit the formula some of the values are forced:

- The number of subproblems, a, must be 3 and higher than 1 (compatible).
- The reduction in the problem size, b, must be 2.
- k can be any value, as it doesn't affect neither expression. We choose a simple implementation with k = 0.

Once this values are deduced, we just have to write a program whose code outside of the recursive part has O(1) complexity, and performs 3 calls that reduce the size by 2 employing substraction.

Substraction2:

The conditions are the following:

- Recursion by division must be used.
- The final complexity must be $O(n^2)$
- The number of subproblems must be 4

Out of all cases in Divide and Conquer by division, the one that fits this result the most is $O(n^k)$ for situations where $a < b^k$. As such, to fit the formula some of the values are forced:

- The number of subproblems, a, is stated by the problem to be 4
- The reduction in the problem size, b, must be high enough to satisfy the expression $a < b^k$.
- The complexity outside the recursion, k, must be 2 to obtain that final complexity.

The only value not fixed is b, which only must be higher than 2. We choose b=4 as it makes sense that the reduction in size of the problem would fit the number of subproblems.

Once this values are deduced, we just have to write a program whose code outside of the recursive part has $O(n^2)$ complexity, and performs 4 calls that reduce the size by 4 employing division.