Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	3
	Surname: Fernández Esparta		
	Name: Mikel		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. [Basic recursive models]

```
 9 public class Subtraction4 {
10     public static long rec4(int n) {
11         long cont = 0;
12         if (n<=0)
13             cont++;
14         else {
15             cont++;  //O(1)
16             rec4(n-2);
17             rec4(n-2);
18             rec4(n-2);
19         }
20         return cont;
21     }
22
23     public static void main(String arg []) {
24         long t1,t2,cont=0;
25         for (int n=1;n<=100;n++) {
26             t1 = System.currentTimeMillis();
27             cont=rec4(n);
28             t2 = System.currentTimeMillis();
29
30             System.out.println ("n="+n+ "**TIME="+(t2-t1)+"**cont="+cont);
31         }  // for
32     } // main
33 } //class
```

We created the Subtraction4 class in which we had to implement a recursive method by subtraction with a complexity O(3^(n/2)).

In order to achieve this, a should equal 3, then b= 2 and k=0 according to the subtraction scheme analysis.

To do this, we need to call the recursive method three times (a), the value of b corresponds to the number of elements that we subtract, in this case 2 from the rec4(n-2) and then k represents the complexity without the recursive calls, which is zero.

Subtraction scheme analysis

- Execution time
  - $T(n) = a * T(n-b) + c * n^K$ if n > basic case
  - $T(n) = c * n^k$ if n = basic case

- Complexity
  - **$O(n^k)$** **if a < 1 (never happens)**
  - **$O(n^{k+1})$** **if a = 1**
  - **$O(a^{n \text{ div } b})$** **if a > 1**

Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	3
	Surname: Fernández Esparta		
	Name: Mikel		

```java
public class Division4 {
    public static long rec4 (int n) {
        long cont = 0;
        if (n<=0)
            cont++;
        else {
            for (int i=0;i<n;i++)
                for (int j=0;j<n;j++)
                    cont++; // O(n)
            rec4(n/3);
            rec4(n/3);
            rec4(n/3);
            rec4(n/3);
        }
        return cont;
    }

    public static void main (String arg []) {
         long t1,t2,cont = 0;
         for (int n=1;n<=10000000;n*=2) {
              t1 = System.currentTimeMillis ();
              cont = rec4(n);
              t2 = System.currentTimeMillis ();

              System.out.println ("n="+n+ "**TIME="+(t2-t1)+"**cont="+cont);
         }  // for
    } // main
```

You should create a new class and implement a recursive method by division with a complexity O(n^2) and a number of subproblems = 4. Therefore a=4, which is also the number of recursive calls, then b=3 and k =2, in order to get a < b^k.

Division scheme analysis

- Execution time
  - $T(n) = a * T(n/b) + c * n^K$ if n > basic case
  - $T(n) = c * n^k$ if n = basic case

- Complexity
  - **$O(n^k)$** **if $a < b^k$**
  - **$O(n^k * \log n)$** **if $a = b^k$**
  - **$O(n^{\log_b a})$** **if $a > b^k$**