Algorithmics	Student information	Date	Number of session
	UO: 276244	13/03/2021	3.1
	Surname: Beltran Diaz		
	Name: Martin		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. BASIC RECURSIVE MODELS

SUBTRACTION CLASSES:

- **Subtraction 1:** The number of recursive calls is 1, so **a=1**, **b=1** because we subtract 1 in each call, and **k=0**, because the complexity of the rest of the algorithm is O(1) and 1=n^0. If we follow what is stated in the theory, when a=1, the time complexity is O(n^(k+1)). So the total time complexity of Subtraction 1 is **O(n)**.

```
public static long rec1(int n) {
    long cont = 0;
    if (n<=0)
        cont++;
    else {
        cont++;  // O(1)=O(n^0)
     rec1(n-1);
    }
    return cont;
}
```

- **Subtraction 2:** The number of recursive calls is 1, so **a=1**, **b=1** because we subtract 1 in each call, and **k=1**, because the complexity of the rest of the algorithm is O(n), due to the for-loop from 0 to n. According to the theory, when a=1, the time complexity is O(n^(k+1)). So the total time complexity of Subtraction 2 is **O(n^2)**.

```
public static long rec2(int n) {
    long cont = 0;
    if (n<=0)
        cont++;
    else {
        for (int i=0;i<n;i++)
            cont++; // O(n)
        rec2(n-1);
    }
    return cont;
}
```

- **Subtraction 3**: The number of recursive calls is 2, so **a=2**, **b=1** because we subtract 1 in each call, and **k=0**, because the complexity of the rest of the algorithm is O(1). According to the theory, when a>1, the time complexity is O(a^(n/b)). So the total time complexity of Subtraction 3 is **O(2^n)**.

```
public static long rec3(int n) {
    long cont = 0;
    if (n<=0)
        cont++;
    else {
        cont++;  //O(1)
        rec3(n-1);
        rec3(n-1);
    }
    return cont;
}
```

DIVISION CLASSES:

- **Division 1**: The number of recursive calls is 1, so **a=1**, **b=3** because we divide by 3 in each call, and **k=1**, because the complexity of the rest of the algorithm is O(n) due to the for-loop from 1 to n. According to the theory, when a<b^k, the time complexity is O(n^k) So the total time complexity of Division 1 is **O(n)**.

```
public static long rec1 (int n) {
    long cont = 0;
    if (n<=0) cont++;
    else {
        for (int i=1;i<n;i++)
            cont++ ;  //O(n)
        rec1(n/3);
    }
    return cont;
}
```

- **Division 2:** The number of recursive calls is 2, so **a=2**, **b=2** because we divide by 2 in each call, and **k=1**, because the complexity of the rest of the algorithm is O(n). According to the theory, when a = b^k, the time complexity is O(n^k * log(n)) So the total time complexity of Division 2 is **O(n * logn)**.

```java
public static long rec2 (int n) {
    long cont = 0;
    if (n<=0) cont++;
    else {
        for (int i=1;i<n;i++)
            cont++ ;  //O(n)
        rec2(n/2);
        rec2(n/2);
    }
    return cont;
}
```

- **Division 3:** The number of recursive calls is 2, so **a=2**, **b=2** because we divide by 2 in each call, and **k=0**, because the complexity of the rest of the algorithm is O(1). According to the theory, when a >b^k, the time complexity is O(n^(logb(a))) So the total time complexity of Division 3 is O(n^1) = **O(n)**.

```java
public static long rec3 (int n) {
    long cont = 0;
    if (n<=0)
        cont++;
    else {
        cont++ ; // O(1)
        rec3(n/2);
        rec3(n/2);
    }
    return cont;
}
```

Now, we will discuss the complexity of the remaining algorithms:

- **Subtraction 4**: The number of recursive calls is 3, so **a=3**, **b=2** because we subtract 2 in each call, and **k=0**, because the complexity of the rest of the algorithm is O(1). According to the theory, when a > 1, the time complexity is O(a^(n/b)) So the total time complexity of Subtraction 4 is **O(3^(n/2))**.

Algorithmics	Student information	Date	Number of session
	UO: 276244	13/03/2021	3.1
	Surname: Beltran Diaz		
	Name: Martin		

```java
public static long rec4(int n) {
    long cont = 0;
    if (n<=0)
        cont++;
    else {
        cont++; //O(1)
     rec4(n-2);
     rec4(n-2);
     rec4(n-2);
    }
    return cont;
}
```

- **Division 4**: The number of recursive calls is 4, so **a=4**, **b=2** because we divide by 2 in each call, and **k=0**, because the complexity of the rest of the algorithm is O(1). According to the theory, when a>b^k, the time complexity is O(n^(lob(a))) So the total time complexity of Division 4 is **O(n^2)**.

```java
public static long rec4 (int n) {
    long cont = 0;
    if (n<=0)
        cont++;
    else {
        cont++ ; // O(1)     == O(1)
        rec4(n/2);
        rec4(n/2);
        rec4(n/2);
        rec4(n/2);

    }
    return cont;
}
```