 Activity 1. [D&C BY SUBSTRACTION]

Substraction1:

a = 1, b = 1, k = 0

Complexity:

O(nk+1) => O(n)

I cannot prove if the results fit or not with the theoretical complexity because the times are not accurate (below 50ms) and then the stack overflow exception rises.

Substraction2:

a = 1, b = 1, k = 1

Complexity:

O(nk+1) => O(n2)

The results obtained fits with the theoretical complexity. Although for values of n < 2048 are not accurate, for higher values until the stack overflow exception we can appreciate that the time grows by a factor of two, which is the square of the increment of the size (22).

Substraction3:

a = 2, b = 1, k = 0

Complexity:

O(an/b) => O(2n)

The results obtained are the ones expected according to the theoretical complexity, because as n increases one by one, the time increases exponentially (the new time is the previously obtained times 2, as 2n+1 = 2n \* 2).

* The operation is aborted not because the time but the memory occupied. As for each iteration a variable is created, the stack has less and less space until it is full occupied.
* N1=20 🡪 t1 = 41

N2=80 🡪 t2 = ?

t2 = 2N2/2N1 \* t1 = 2N2-N1 \* t1 = 260 \* t1 = 47.269.781.688.880.726.016ms \* 1s/1000ms \* 1h/3600s \* 1day/24h \* 1year/365.25days = 1.497.888.993,1 years

|  |  |
| --- | --- |
| Substraction4 | |
| n | time(ms) |
| 100 | 1,425 |
| 200 | 10,65 |
| 400 | 82,15 |
| 800 | 650 |
| 1600 | 5013 |
| 3200 | 40532 |

|  |  |
| --- | --- |
| Substraction5 | |
| n | time(ms) |
| 32 | 1098 |
| 34 | 3291 |
| 36 | 9873 |
| 38 | 29285 |

* N1=32 🡪 t1 = 1098

N2=80 🡪 t2 = ?

t2 = 3N2/2/3N1/2 \* t1 = 3N2/2 - N1/2 \* t1 = 324 \* t1 = 310.107.631.056.138ms\* 1s/1000ms \* 1h/3600s \* 1day/24h \* 1year/365.25days = 9.826,72years

Activity 2. [D&C BY DIVISION]

Division1:

a = 1, b = 3, k = 1

Complexity:

O(nk) => O(n)

The results obtained fits with the theoretical complexity because as the size increases by a factor of 2, same happens with the time, so the time increases linearly.

Division2:

a = 2, b = 2, k = 1

Complexity:

O(nk\*log(n)) => O(n\*log(n))

According to the theoretical complexity, the results obtained are the ones expected. The size increments by a factor of 2 but the time is increased by a bit more than 2 which reflects that multiplication by log(n) of the theoretical calculus.

Division3:

a = 2, b = 2, k = 0

Complexity:

O(nlogba) => O(n)

We are in the same case that with Division1 and the results obtained also fits the constraints required (has a linear time increment according to the size increment).

Division4:

|  |  |
| --- | --- |
| Division4 | |
| n | time(ms) |
| 1000 | 7,9 |
| 2000 | 29,2 |
| 4000 | 116 |
| 8000 | 465,2 |
| 16000 | 1820,5 |
| 32000 | 7293 |
| 64000 | 28918 |
| 128000 | Oot |

Division5:

|  |  |
| --- | --- |
| Division5 | |
| n | time(ms) |
| 1000 | 23,4 |
| 2000 | 92 |
| 4000 | 369,5 |
| 8000 | 1502,5 |
| 16000 | 5997 |
| 32000 | 23637 |
| 64000 | Oot |

Activity 3. [TWO BASIC PRINCIPLES]

Fibonacci1:

a = 1, b = 3, k = 1

Complexity:

O(nk) => O(n)

Fibonacci2:

a = 2, b = 2, k = 1

Complexity:

O(nk\*log(n)) => O(n\*log(n))

Fibonacci3:

a = 2, b = 2, k = 0

Complexity:

O(nlogba) => O(n)

Fibonacci4:

a = 2, b = 2, k = 0

Complexity:

O(nlogba) => O(n)