Mikel Fernández Esparta

UO275688

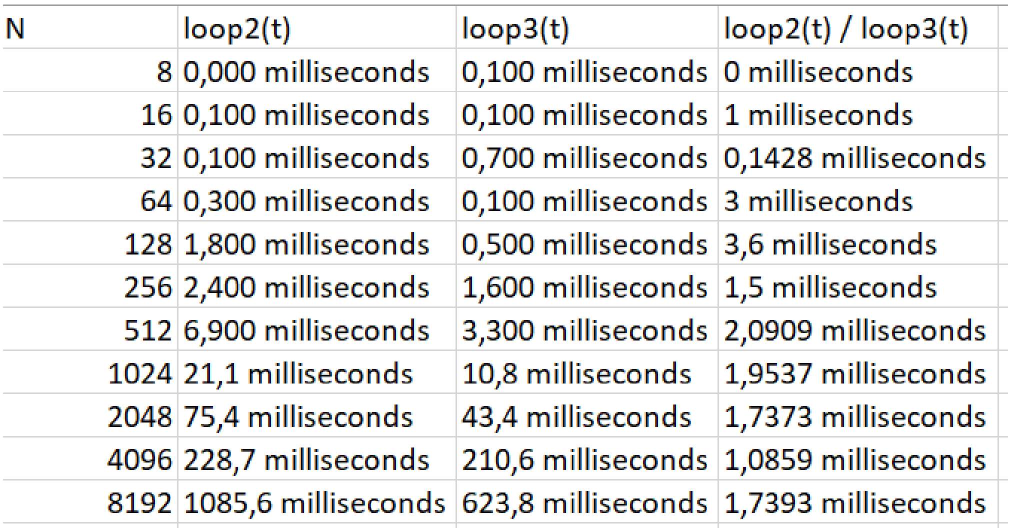
Algorithmics

LAB I-3

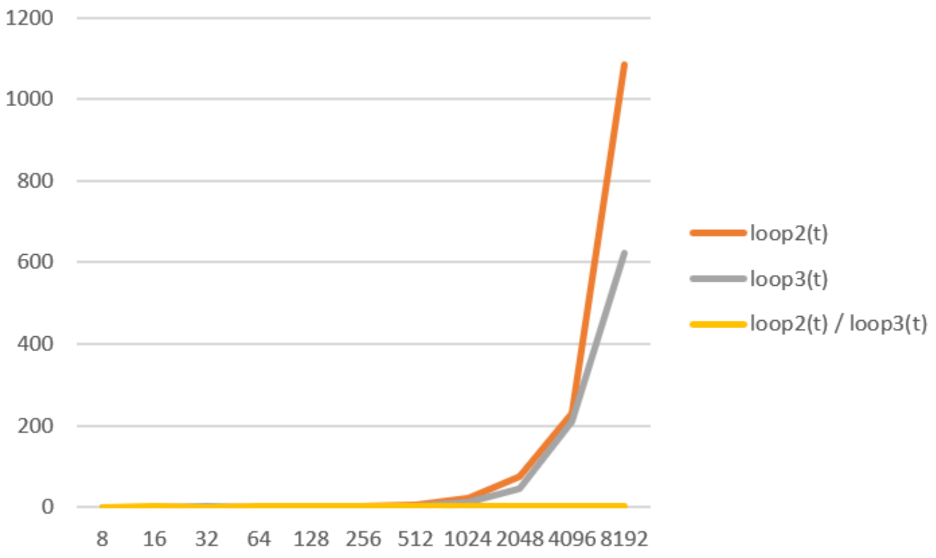
Activity 1. Two algorithms with the same complexity

We are going to compare loop2 and loop3 algorithms and obtain the value for the division loop2/loop3. To that, you should fill in the following table (remember to include always in all columns the units of time and the CPU and RAM of the machine where the measurement was made). In addition, briefly explain if the results make sense from the point of view of the complexities of the algorithms.

The results make sense according to the complexity. In this case both are quadratic. Loop2 is better.



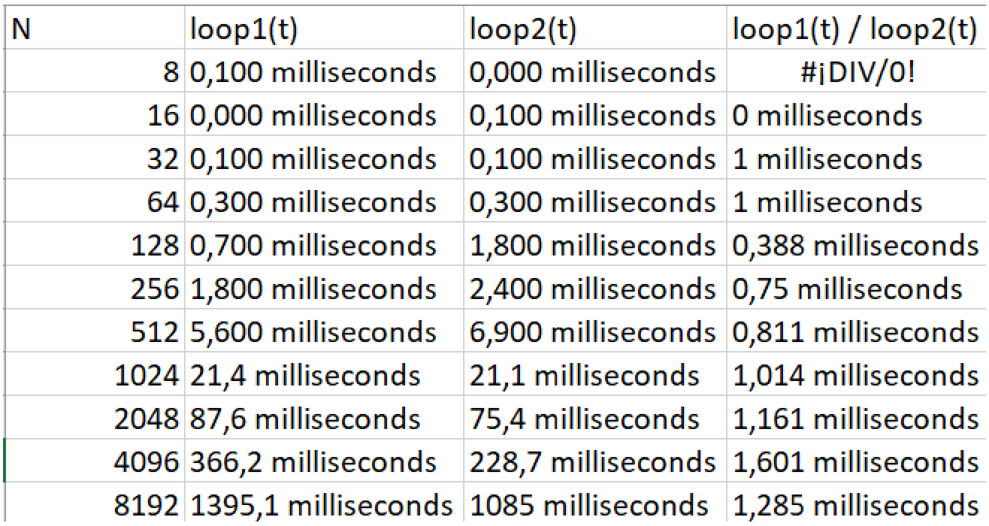
N	loop2(t)	loop3(t)	loop2(t) / loop3(t)
8	0,000 milliseconds	0,100 milliseconds	0 milliseconds
16	0,100 milliseconds	0,100 milliseconds	1 milliseconds
32	0,100 milliseconds	0,700 milliseconds	0,1428 milliseconds
64	0,300 milliseconds	0,100 milliseconds	3 milliseconds
128	1,800 milliseconds	0,500 milliseconds	3,6 milliseconds
256	2,400 milliseconds	1,600 milliseconds	1,5 milliseconds
512	6,900 milliseconds	3,300 milliseconds	2,0909 milliseconds
1024	21,1 milliseconds	10,8 milliseconds	1,9537 milliseconds
2048	75,4 milliseconds	43,4 milliseconds	1,7373 milliseconds
4096	228,7 milliseconds	210,6 milliseconds	1,0859 milliseconds
8192	1085,6 milliseconds	623,8 milliseconds	1,7393 milliseconds



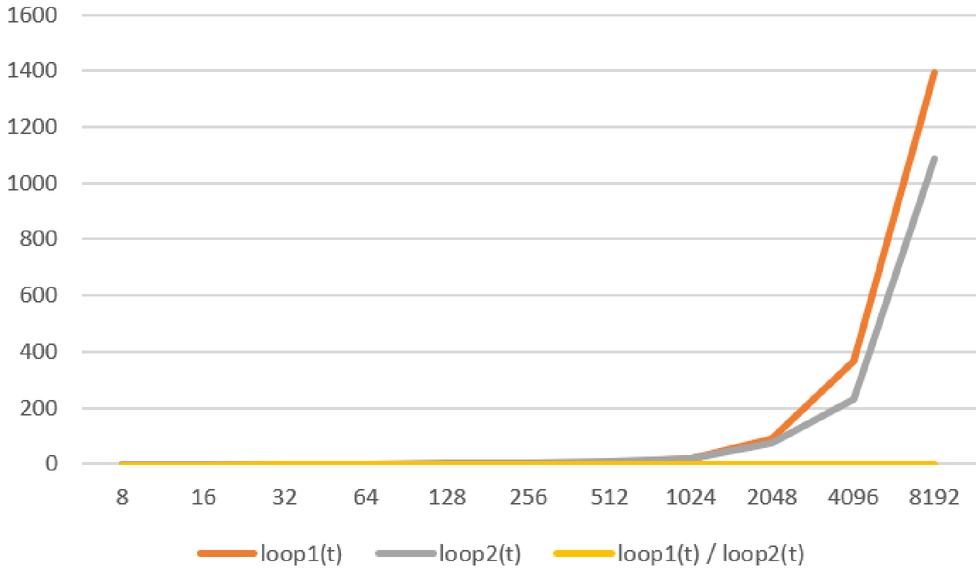
Activity 2. Two algorithms with different complexity

We are going to compare loop1 and loop2 algorithms and obtain the value for the division loop1/loop2. To that, you should fill in the following table (remember to include always in all columns the units of time and the CPU and RAM of the machine where the measurement was made). In addition, briefly explain if the results make sense from the point of view of the complexities of the algorithms.

In this case, loop1 is better.



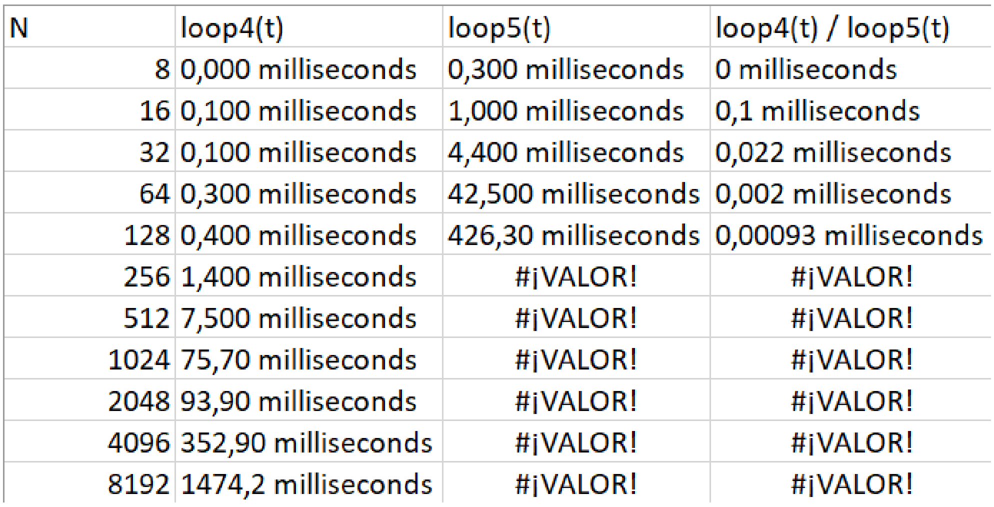
N	loop1(t)	loop2(t)	loop1(t) / loop2(t)
8	0,100 milliseconds	0,000 milliseconds	#¡DIV/0!
16	0,000 milliseconds	0,100 milliseconds	0 milliseconds
32	0,100 milliseconds	0,100 milliseconds	1 milliseconds
64	0,300 milliseconds	0,300 milliseconds	1 milliseconds
128	0,700 milliseconds	1,800 milliseconds	0,388 milliseconds
256	1,800 milliseconds	2,400 milliseconds	0,75 milliseconds
512	5,600 milliseconds	6,900 milliseconds	0,811 milliseconds
1024	21,4 milliseconds	21,1 milliseconds	1,014 milliseconds
2048	87,6 milliseconds	75,4 milliseconds	1,161 milliseconds
4096	366,2 milliseconds	228,7 milliseconds	1,601 milliseconds
8192	1395,1 milliseconds	1085 milliseconds	1,285 milliseconds



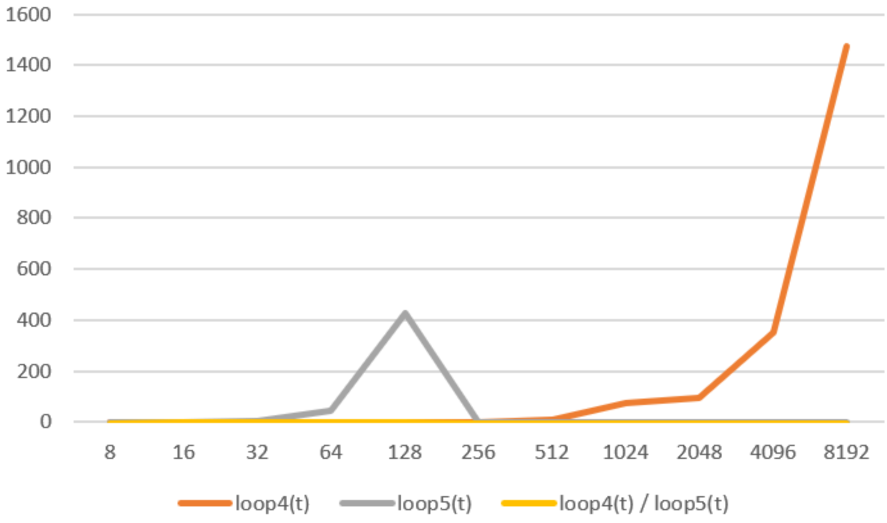
Activity 3. Complexity of other algorithms

We are going to create and compare two new algorithms, loop4 (it should have a O(n4) complexity) and loop5 (it should have a O(n3 logn) complexity) algorithms and obtain the value for the division loop4/loop5. To that, you should fill in the following table (remember to include always in all columns the units of time and the CPU and RAM of the machine where the measurement was made). In addition, briefly explain if the results make sense from the point of view of the complexities of the algorithms.

So far, loop5 seemed to be better however the program crashes at value 256.



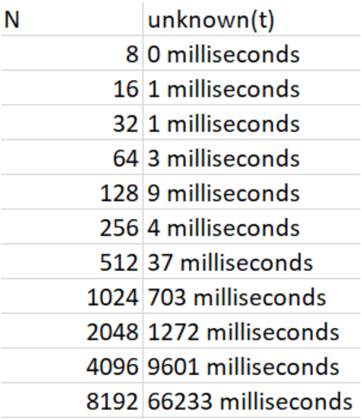
N	loop4(t)	loop5(t)	loop4(t) / loop5(t)
8	0,000 milliseconds	0,300 milliseconds	0 milliseconds
16	0,100 milliseconds	1,000 milliseconds	0,1 milliseconds
32	0,100 milliseconds	4,400 milliseconds	0,022 milliseconds
64	0,300 milliseconds	42,500 milliseconds	0,002 milliseconds
128	0,400 milliseconds	426,30 milliseconds	0,00093 milliseconds
256	1,400 milliseconds	#¡VALOR!	#¡VALOR!
512	7,500 milliseconds	#¡VALOR!	#¡VALOR!
1024	75,70 milliseconds	#¡VALOR!	#¡VALOR!
2048	93,90 milliseconds	#¡VALOR!	#¡VALOR!
4096	352,90 milliseconds	#¡VALOR!	#¡VALOR!
8192	1474,2 milliseconds	#¡VALOR!	#¡VALOR!



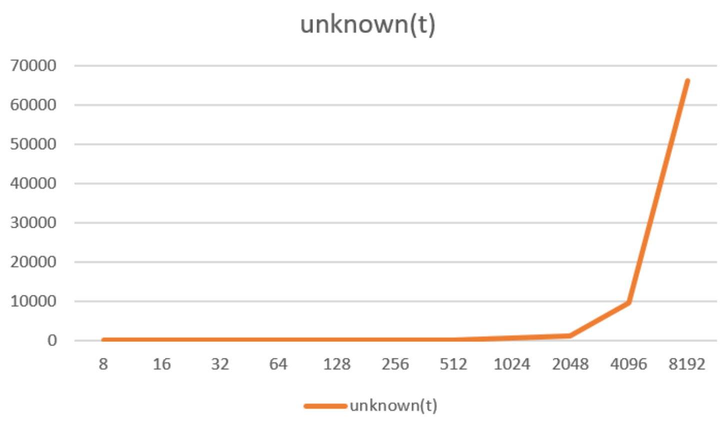
Activity 4. Study of Unknown.java

You should create another table with execution times for different sizes of the problem for the method contained in Unknown.java together with a brief explanation of its complexity. Does it make sense according to the theoretical complexity? Use the formula explained in class to calculate the expected execution time when the size of the problem changes and compare it with the time you took empirically. Do it twice.

The complexity is O(n4), as we can see of the graph plotted it starts growing a lot when the size is bigger.



N	unknown(t)
8	0 milliseconds
16	1 milliseconds
32	1 milliseconds
64	3 milliseconds
128	9 milliseconds
256	4 milliseconds
512	37 milliseconds
1024	703 milliseconds
2048	1272 milliseconds
4096	9601 milliseconds
8192	66233 milliseconds


unknown(t)
unknown(t)