


Algorithmics	Student information	Date	Number of session
	UO: UO277653	11/02/2021	1.1
	Surname: Stanci	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Stelian Adrian		



Activity 1. Measuring execution times

1- How many more years can we continue using this way of counting?

Considering the maximum value for a number in Java, using the long data type, the method `System.currentTimeMillis()` will overflow at around 292278994 years. So, we could continue using this way of counting for roughly 292 million more years.

2. What does it mean that the time measured is 0?

That the program has taken less than a millisecond to complete. Also, it could happen when we execute repetitive instructions, as the compiler optimizes the code.

3. From what size of problem (n) do we start to get reliable times?

After around $n=150000000$, which is when the time starts to reach 50 milliseconds and beyond (this amount could vary depending on the components of the computer)

Activity 2. Grow of the problem size

1. What happens with time if the size of the problem is multiplied by 5?

In the first iterations the time is 0 as the size of the problem is very small. Then, there comes a point where we start to get different values, and a linear function can be seen in the final iterations as the times for values 5 times higher are also 5 times longer.

Algorithmics	Student information	Date	Number of session
	UO: UO277653	11/02/2021	1.1
	Surname: Stanci		
	Name: Stelian Adrian		

```

SIZE-1250 TIME-0 milliseconds SUM-1951
SIZE-6250 TIME-0 milliseconds SUM--3332
SIZE-31250 TIME-0 milliseconds SUM-11726
SIZE-156250 TIME-1 milliseconds SUM-29943
SIZE-781250 TIME-0 milliseconds SUM--55216
SIZE-3906250 TIME-2 milliseconds SUM--81085
SIZE-19531250 TIME-6 milliseconds SUM--13899
SIZE-97656250 TIME-29 milliseconds SUM-1163028
SIZE-488281250 TIME-149 milliseconds SUM-Except

```

2. Are the times obtained those that were expected from linear complexity $O(n)$?

Yes, as the size of the problem is multiplied by 5, the execution time is too.

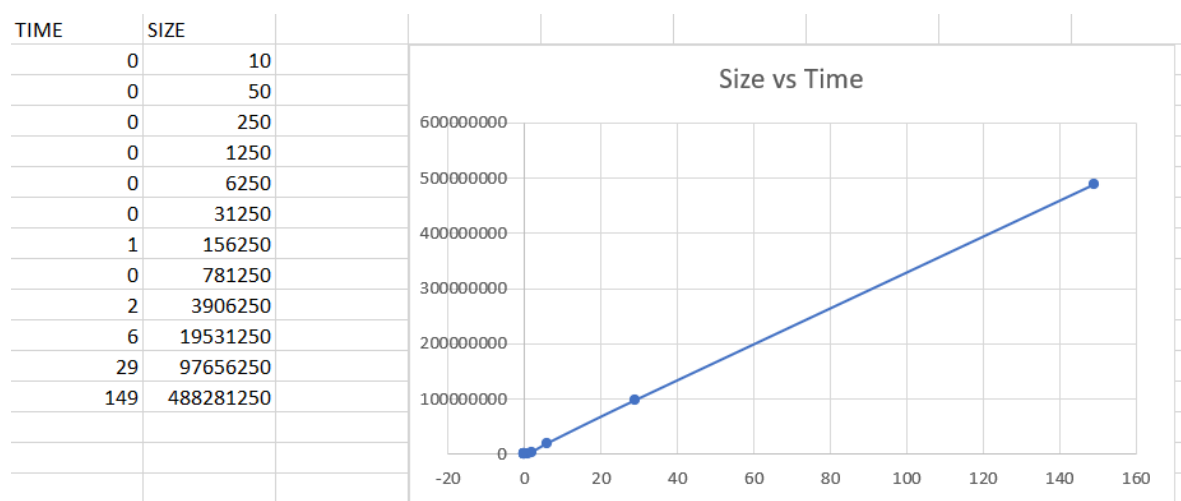
For example taking the last two iterations:

$N_1=97656250$ $T_1=29$

$N_2=488281250$ $T_2=?$

Applying the formula with k being 5 (n_1/n_2) we get that T_2 should be around 150 milliseconds, which it is very close to.

3. Use a spreadsheet to draw a graph with Excel. On the X axis we can put the time and on the Y axis the size of the problem.



Algorithmics	Student information	Date	Number of session
	UO: UO277653	11/02/2021	1.1
	Surname: Stanci		
	Name: Stelian Adrian		

Activity 3. Taking small execution times

n	fillIn(t) 10 ⁻⁴ s	sum(t) 10 ⁻⁴ s	maximum(t) 10 ⁻⁴ s
10	0	1	0
30	0	0	0
90	0	0	0
270	0	0	0
810	0	0	0
2430	1	0	1
7290	2	1	2
21870	3	1	3
65610	6	1	2
196830	18	1	1
590490	55	2	3
1771470	159	7	12
5314410	452	24	42
15943230	1362	64	107
47829690	4075	170	353
143489070	12263	515	941
430467210	37598	1513	2668

1. What are the main components of the computer in which you did the work (process, memory)?

Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz

RAM: 16GB

Algorithmics	Student information	Date	Number of session
	UO: UO277653	11/02/2021	1.1
	Surname: Stanci		
	Name: Stelian Adrian		

2. Do the values obtained meet the expectations? For that, you should calculate and indicate the theoretical values (a couple of examples per column) of the time complexity. Explain briefly the results.

Having t_1 , n_1 and n_2 , we calculate the theoretical t_2 using the formula

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1$$

Taking into account that all three functions are linear to obtain t_2 we have to multiply the previous time by k , which in this case is 3:

N	fillIn(t)	fillIn(t) theoretical	sum(t)	sum(t) theoretical	maximum(t)	maximum(t) theoretical
15943230	1362	1356	64	72	107	126
47829690	4075	4086	170	192	353	321
143489070	12263	12225	515	510	941	1059
430467210	37598	36789	1513	1545	2668	2823

So, as we can see, the values are close to the one that should be obtained, so we can conclude that they make sense

Activity 4. Operations on matrices

n	sumDiagonal1(t) $10^{-3}s$	sumDiagonal2(t) $10^{-3}s$
10	0	0
30	0	0
90	0	0
270	0	0

Algorithmics	Student information	Date	Number of session
	UO: UO277653	11/02/2021	1.1
	Surname: Stanci		
	Name: Stelian Adrian		

810	0	1
2430	2	0
7290	2	0
21870	1	1
65610	3	2
196830	10	5
590490	28	6
1771470	88	11
5314410	239	34
15943230	741	98
47829690	2112	284
143489070	6196	855
430467210	18817	2559

1. What are the main components of the computer in which you did the work (process, memory)?

Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz

RAM: 16GB

3. Do the values obtained meet the expectations? For that, you should calculate and indicate the theoretical values (a couple of examples per column) of the time complexity. Explain briefly the results.

We use the same formula as before, taking into account the complexity.

n	sumDiagonal1(t) $10^{-3}s$	Theoretical	sumDiagonal2(t) $10^{-3}s$	Theoretical
47829690	2112	2223	284	294

Algorithmics	Student information	Date	Number of session
	UO: UO277653	11/02/2021	1.1
	Surname: Stanci		
	Name: Stelian Adrian		

143489070	6196	6336	855	852
430467210	18817	18588	2559	2565

The difference in the complexities can be seen in the result. As the first method has a worse complexity, the times obtained are much higher

Activity 5. Benchmarking

4. Why you get differences in execution time between the two programs?

Because of the differences in the language used (Python is an interpreted language, which makes it slower, while Java is a compiled language). The VM also plays a role in the time obtained. We should also note that they use different compilers, and that may also lead to differences.

5. Regardless of the specific times, is there any analogy in the behavior of the two implementations??

They implement a linear and a quadratic function. Then, in the main program, they both call the functions several times in a period of 5 seconds, and print the time the function takes to execute for increasing values of n.