


Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Rita		



SESSION 1.1

Activity 1. Measuring execution times

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

The maximum value of a Java long is $2^{63} - 1$, and if you convert that many milliseconds into practical units of time, you find that the counter will overflow in approximately 290 million years. Therefore, we can continue using this way for 290 million years.

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

We get that the time measured is 0 when we are using small values for the argument (size of the vector). This means that the time to execute the program is less than 1 millisecon.

Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz		
	Name: Rita		

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

We start to get reliable times from the number 95000000, since it is the first number which makes the execution time greater than 50 milliseconds (times below 50 milliseconds are not reliable at all). However, it does not exist a fixed number which guaranteed reliable times because each computer runs in a different way and hence, the times can vary from one computer to another (for example, your computer can be faster and thus, the number 95000000 can be executed in less than 50 milliseconds).

(Output example)

SIZE (n) = 95000000 - TIME = 51 milliseconds

Activity 2. Grow of the problem size

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

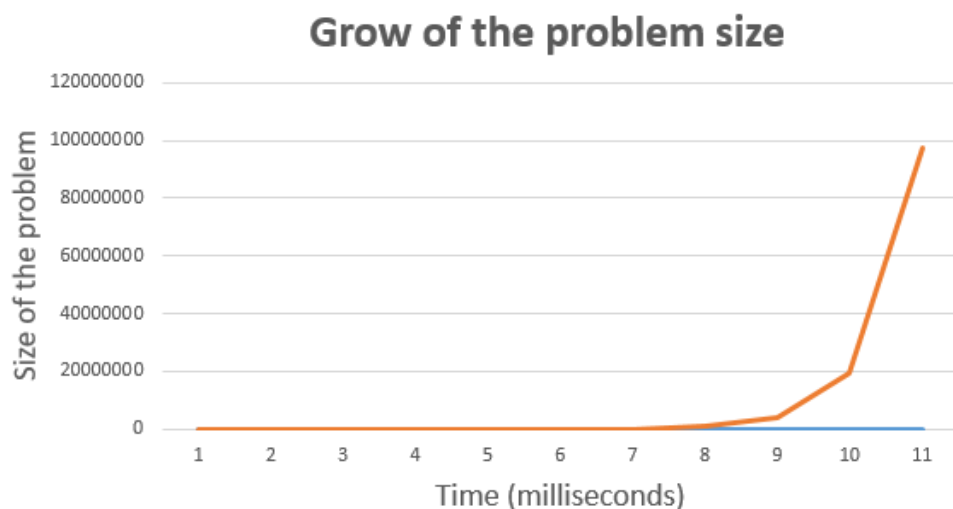
As the operation has complexity $O(n)$, if we increase the size by 5, the time is also increased by 5.

Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz		
	Name: Rita		

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

As the computer I used for measuring the time didn't have enough memory space, I didn't get reliable times (all of them are under 50 milliseconds). Then, I cannot derive a conclusion from those results.

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



These are the results I got before the program crashed because of the memory system.

Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz		
	Name: Rita		

Activity 3. Taking small execution times

n	$fillIn(n)$	$sum(t)$	$maximum(t)$
1771470	21	0	1
5314410	61	2	2
15943230	173	6	6
47829690	564	34	25
143489070	1513	63	46
430467210	4493	166	148

Measured in milliseconds with 1000 repetitions. These are the greatest values before the program crashed because of the memory system.

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

Procesador

Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz 3.19 GHz

Processor Intel(R) Core(TM) i5-6500

Memory 3.20GHz

Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz		
	Name: Rita		

- 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. Briefly explain the results.

According to the following result, we can conclude that the values obtained meet the expectations:

We know that $t_2 = (f(n_2) / f(n_1)) \times t_1$

Taking this values:

$n_1 = 15943230$

$n_2 = 47829690$

For filln() function with complexity $O(n)$:

$t_1 = 173$

$t_2 = 564 \qquad t_2 = (47829690/15943230) \times 173 = 519 = 564$

For sum() and maximum() functions, the values of t_1 and t_2 are less than 50 milliseconds. Then, those measurements are not reliable so I don't apply the algorithm.

Taking this values:

$n_1 = 143489070$

$n_2 = 430467210$

Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz		
	Name: Rita		

For fillIn() function with complexity $O(n)$:

t1 = 1513

t2 = 4493 $t2 = (430467210/143489070) \times 1513 = 4539 = 4493$

For sum() function with complexity $O(n)$:

t1 = 63

t2 = 166 $t2 = (430467210/143489070) \times 63 = 189 = 166$

For maximum() function with complexity $O(n)$:

t1 = 46

t2 = 148 $t2 = (430467210/143489070) \times 46 = 138 = 148$

Activity 4. Operations on matrices

<i>n</i>	<i>sumDiagonal1(t)</i>	<i>sumDiagonal2(t)</i>
10	3	5
30	7	15
90	8	9
270	52	158
810	394	1185
2430	2979	2846
7290	24667	28823
21870	219249	234982

Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz		
	Name: Rita		

Measured in milliseconds with 1000 repetitions. These are the greatest values before the program crashed because of the memory system.

- **What are the main components of the computer in which you did the work (process, memory)**

Procesador

Intel(R) Core(TM) i5-6500 CPU @
3.20GHz 3.19 GHz

Processor Intel(R) Core(TM) i5-6500

Memory 3.20GHz

- **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. Briefly explain the results**

According to the following result, we can conclude that the values obtained meet the expectations:

We know that $t_2 = (f(n_2) / f(n_1)) \times t_1$

Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz		
	Name: Rita		

Taking this values:

$n1 = 270$

$n2 = 810$

For sumDiagonal1() function with complexity $O(n^2)$:

$t1 = 52$

$t2 = 394 \quad t2 = (810^2/270^2) \times 52 = 468 = 394$

For sumDiagonal2() function with complexity $O(n)$:

$t1 = 158$

$t2 = 1185 \quad t2 = (810/270) \times 158 = 1422 = 1185$

Taking this values:

$n1 = 7290$

$n2 = 21970$

For sumDiagonal1() function with complexity $O(n^2)$:

$t1 = 24660$

$t2 = 219249 \quad t2 = (21970^2/7290^2) \times 24660 = 223.974 = 219249$

For sumDiagonal2() function with complexity $O(n)$:

$t1 = 28823$

$t2 = 234982 \quad t2 = (21970/7290) \times 28823 = 259407 = 234982$

Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz		
	Name: Rita		

Activity 5. BenchMarking

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

According to these results:

```
Linear times in Java (milliseconds)
counter=1000000 n=1000000 Time=10
counter=2000000 n=2000000 Time=10
counter=4000000 n=4000000 Time=5
counter=8000000 n=8000000 Time=10
counter=16000000 n=16000000 Time=21
counter=32000000 n=32000000 Time=31
counter=64000000 n=64000000 Time=51
counter=128000000 n=128000000 Time=111
counter=256000000 n=256000000 Time=176
counter=512000000 n=512000000 Time=325
counter=1024000000 n=1024000000 Time=617
counter=2048000000 n=2048000000 Time=1151
counter=4096000000 n=4096000000 Time=2319
counter=8192000000 n=8192000000 Time=5641
Quadratic times in Java (milliseconds)
counter=10000 n=100 Time=0
counter=40000 n=200 Time=1
counter=160000 n=400 Time=4
counter=640000 n=800 Time=5
counter=2560000 n=1600 Time=3
counter=10240000 n=3200 Time=6
counter=40960000 n=6400 Time=23
counter=163840000 n=12800 Time=89
counter=655360000 n=25600 Time=399
counter=2621440000 n=51200 Time=1455
counter=10485760000 n=102400 Time=5853
```

```
LINEAR TIMES (MILLISEC.)
COUNTER 1000000 n= 1000000 *** time 52
COUNTER 2000000 n= 2000000 *** time 84
COUNTER 4000000 n= 4000000 *** time 203
COUNTER 8000000 n= 8000000 *** time 390
COUNTER 16000000 n= 16000000 *** time 738
COUNTER 32000000 n= 32000000 *** time 1486
COUNTER 64000000 n= 64000000 *** time 3052
COUNTER 128000000 n= 128000000 *** time 6099
QUADRATIC TIMES (MILLISEC.)
COUNTER 10000 n= 100 *** time 0
COUNTER 40000 n= 200 *** time 0
COUNTER 160000 n= 400 *** time 0
COUNTER 640000 n= 800 *** time 31
COUNTER 2560000 n= 1600 *** time 124
COUNTER 10240000 n= 3200 *** time 484
COUNTER 40960000 n= 6400 *** time 1908
COUNTER 163840000 n= 12800 *** time 7719
```

Algorithmics	Student information	Date	Number of session
	UO: 284185	8/2/2022	1.1
	Surname: Fernández-Catuxo Ortiz		
	Name: Rita		

Comparing both results, we can conclude that java is faster than python: This is because python interprets the code as it executes it. However, Java compiles the code before its execution. As a result, the execution time is shorter.

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

In the linear, every time the number of repetitions is multiplied by 2, the time also does. However, in the quadratic, the times are increasing by 4 (in each execution is squared).