


Algorithmics	Student information	Date	Number of session
	UO:276244	11/02/2021	1_1
	Surname: Beltran Diaz	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Martin		



## Activity 1. Measuring execution times

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

We can use it for another 292 million years (approximation)

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

It means that the execution of the program takes less time than minimum time needed to update the timer of the OS.

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

I start getting reliable times (over 50ms) when n is bigger than 80 million

## Activity 2. Grow of the problem size

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

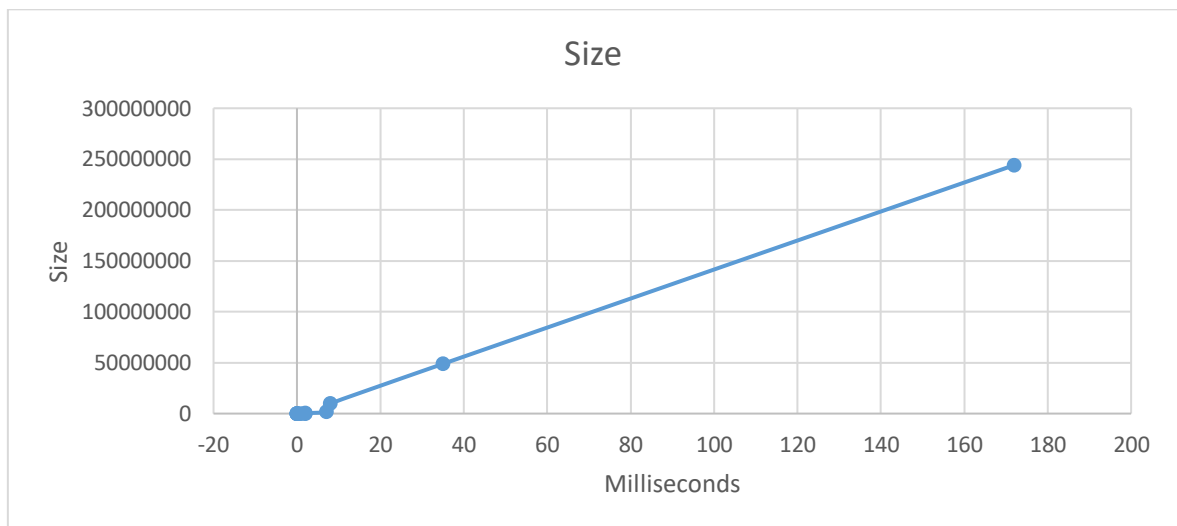
The times obtained are more or less also multiplied by 5

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

They are, because they increase in the same way as the size of the problem grow.

Algorithmics	Student information	Date	Number of session
	UO:276244	11/02/2021	1_1
	Surname: Beltran Diaz		
	Name: Martin		

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.



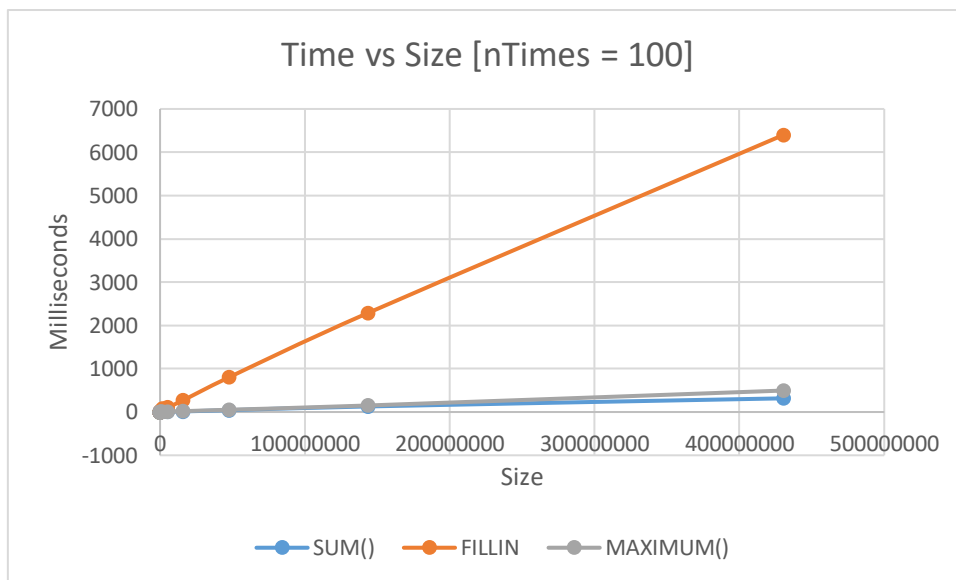
## Activity 3 Taking small execution times

Each calculation has been made with nTimes = 100

N	SUM()	FILLIN()	MAXIMUM()
10	0,04	0,03	0,01
30	0,1	0,05	0,03
90	0	0,03	0,02
270	0,01	0,06	0,05
810	0,31	0,06	0,05
2430	0,4	0,06	0,02
7290	0	0,2	0,06
21870	0	0,44	0,04
65610	4	1,4	0,07
196830	1,57	5,6	0,3
590490	0,86	22,5	1,06

Algorithmics	Student information	Date	Number of session
	UO:276244	11/02/2021	1_1
	Surname: Beltran Diaz		
	Name: Martin		

1771470	1,81	80,15	2,75
5314410	4,88	108,02	5,54
15943230	14,08	271,25	19,07
47829690	42,1	805,16	54,12
143489070	128,69	2286,72	150,89
430467210	316,41	6401,27	496,28



As you can see in the graph, the three methods have a linear complexity  $O(n)$ , since they move through the one-dimensional vector.

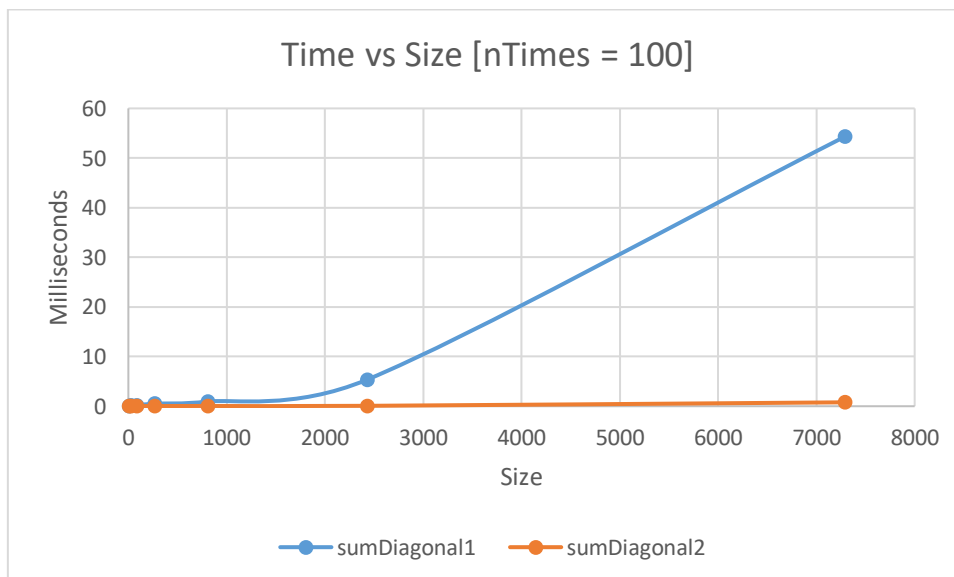
Algorithmics	Student information	Date	Number of session
	UO:276244	11/02/2021	1_1
	Surname: Beltran Diaz		
	Name: Martin		

## Activity 4 Operation on matrices

Each calculation has been made with nTimes = 100

N	sumDiagonal1	sumDiagonal2
10	0	0,04
30	0,18	0,01
90	0,2	0,025
270	0,52	0,03
810	0,95	0,05
2430	5,36	0,08
7290	54,31	0,78

As the matrix is of order N (NxN elements), the maximum value N can reach in my PC before it crashes is N = 7290.



The process took place mainly in the processor and in memory (as a lot of space must be reserved to save all the values in the matrix of order 7290).

Algorithmics	Student information	Date	Number of session
	UO:276244	11/02/2021	1_1
	Surname: Beltran Diaz		
	Name: Martin		

The chart meet the expectations, as the sumDiagonal1 () method was implemented with a complexity of  $O(n^2)$  and sumDiagonal2 () was implemented with linear complexity  $O(n)$ .

## Activity 5 Benchmarking

### TIMES FOR JAVA PROGRAM:

Linear times in Java (milliseconds)

counter=1000000 n=1000000 Time=39

counter=2000000 n=2000000 Time=31

counter=4000000 n=4000000 Time=48

counter=8000000 n=8000000 Time=79

counter=16000000 n=16000000 Time=167

counter=32000000 n=32000000 Time=168

counter=64000000 n=64000000 Time=39

counter=128000000 n=128000000 Time=107

counter=256000000 n=256000000 Time=217

counter=512000000 n=512000000 Time=426

Counter=1024000000 n=1024000000 Time=845

counter=2048000000 n=2048000000 Time=1956

counter=4096000000 n=4096000000 Time=4227

counter=8192000000 n=8192000000 Time=8072

Quadratic times in Java (milliseconds)

counter=10000 n=100 Time=1

counter=40000 n=200 Time=2

counter=160000 n=400 Time=5

counter=640000 n=800 Time=4

counter=2560000 n=1600 Time=3

counter=10240000 n=3200 Time=8

Algorithmics	Student information	Date	Number of session
	UO:276244	11/02/2021	1_1
	Surname: Beltran Diaz		
	Name: Martin		

counter=40960000 n=6400 Time=45

counter=163840000 n=12800 Time=172

counter=655360000 n=25600 Time=683

counter=2621440000 n=51200 Time=2707

counter=10485760000 n=102400 Time=10285

### **TIMES FOR PYTHON PROGRAM:**

#### LINEAR TIMES (MILLISEC.)

COUNTER 1000000 n= 1000000 \*\*\* time 95

COUNTER 2000000 n= 2000000 \*\*\* time 153

COUNTER 4000000 n= 4000000 \*\*\* time 314

COUNTER 8000000 n= 8000000 \*\*\* time 629

COUNTER 16000000 n= 16000000 \*\*\* time 1357

COUNTER 32000000 n= 32000000 \*\*\* time 2984

COUNTER 64000000 n= 64000000 \*\*\* time 5936

#### QUADRATIC TIMES (MILLISEC.)

COUNTER 10000 n= 100 \*\*\* time 1

COUNTER 40000 n= 200 \*\*\* time 4

COUNTER 160000 n= 400 \*\*\* time 15

COUNTER 640000 n= 800 \*\*\* time 60

COUNTER 2560000 n= 1600 \*\*\* time 242

COUNTER 10240000 n= 3200 \*\*\* time 1047

COUNTER 40960000 n= 6400 \*\*\* time 3722

COUNTER 163840000 n= 12800 \*\*\* time 16087

### 1- Why you get differences in execution time between the two programs?

I think it is because java uses a compiler while python uses an interpreter.

Algorithmics	Student information	Date	Number of session
	UO:276244	11/02/2021	1_1
	Surname: Beltran Diaz		
	Name: Martin		

2- Regardless of the specific times, is there any analogy in the behaviour of the two implementations?

For each counter, they have the same “n”.