Algorithmics	Student information	Date	Number of session
	UO: 276903	23/2/21	1
	Surname: Garriga Suárez		
	Name: Carlos		

Escuela de Ingeniería Informática
Universidad de Oviedo

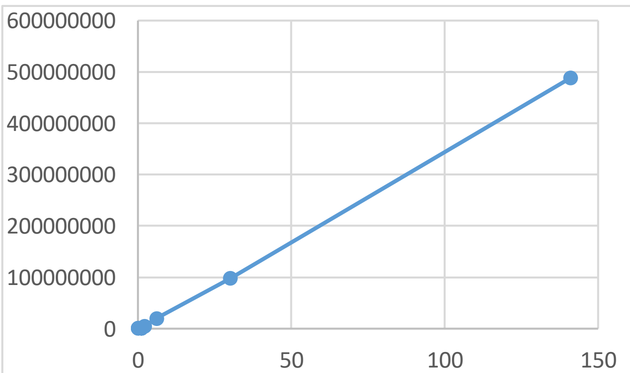
Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Measuring Execution Times

1. The maximum number that can be represented in a Long type is 9.223.372.036.854.775.807 The number of milliseconds that have passed since 1970 Jan 1 are: the number of milliseconds in a year (31.536.000.000 ms) times the years passed (51). The milliseconds that have passed are 1.608.336.000.000 so if we do the subtraction between the max Long value and the milliseconds calculated before we get the remaining milliseconds, we will be able to represent in the future. The estimated years are 292471157.677536.
2. The time measured equals 0 means that the n is so small that it cannot give a such small time.
3. We start to get reliable times at n = 180000000 approximately.

Activity 2. Grow of the problem size.

1. The time will grow exponentially.
2. The times obtained are the ones expected from a linear complexity O(n) because the algorithm that is measured has a O(n) complexity.
3.



Algorithmics	Student information	Date	Number of session
	UO: 276903	23/2/21	1
	Surname: Garriga Suárez		
	Name: Carlos		

Activity 3. Taking small execution times

n	*fillIn(t)*	*sum(t)*	*maximum(t)*
10	1	0	0
30	1	0	0
90	1	0	0
270	1	0	0
810	1	1	0
2430	3	1	0
7290	7	2	2
21870	20	2	2
65610	60	2	2
196830	180	5	4
590490	529	15	14
1771470	1605	51	46
5314410	4798	163	146
15943230	14262	463	446
47829690	42768	1392	1309
143489070	130323	4180	3922
430467210	385922	12775	11960
Until it crashes	…..	…..	…..

All these results are in hundreds of milliseconds.
The main components that are doing the work are the CPU as you are doing operations like for example in the sum(t) method and the memory as you are working with arrays.
Theoretical results:

n	*fillIn(t)*	*sum(t)*	*maximum(t)*
10			
30			
90			
270			
810			
2430	(3)		
7290	(9)		
21870	(27)		
65610	(81)		
196830	(243)	(5)	(4)

Escuela de Ingeniería Informática

590490	(729)	(15)	(12)
1771470	(2187)	(45)	(36)
5314410	(6561)	(135)	(108)
15943230	(19683)	(405)	(324)
47829690	(59049)	(1215)	(972)
143489070	(177147)	(3645)	(2916)
430467210		(10935)	(8748)
Until it crashes			

All these results are in hundreds of milliseconds. The results of the theoretical values are in some cases very similar but with higher values it tends to vary more. I did not calculate the theoretical values in small n because the calculus was not reliable.

Activity 4. Operations on matrices

n	*sumDiagonal1(t)*	*sumDiagonal2(t)*
10	0	0
30	1	0
90	3	0
270	3	0
810	16	1
2430	146	3
7290	1249	10
21870	11133	57
…		
Until it crashes		

The main components that are doing the work are the CPU and the memory as you are working with matrices. The first algorithm consumes more memory accesses as it is iterating all over the matrix.

Theoretical values:

n	*sumDiagonal1(t)*	*sumDiagonal2(t)*
10		
30		
90		
270	3	
810	27	1
2430	243	3
7290	2187	9
21870	19683	27
…		
Until it crashes		

Algorithmics	Student information	Date	Number of session
	UO: 276903	23/2/21	1
	Surname: Garriga Suárez		
	Name: Carlos		

I did not calculate the theoretical values in small n because the calculus was not reliable.

Activity 5. Benchmarking

It is normal to have different times of execution despite is the same program because Java and Phyton do not execute the code in the same way. Probably the times taken from a second time in any of the IDEs would be different from the first one. The code is just the same in both languages but each of them in its syntax.