


Algorithmics	Student information	Date	Number of session
	UO: 276903	27/02/21	3
	Surname: Garriga Suárez	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Carlos		



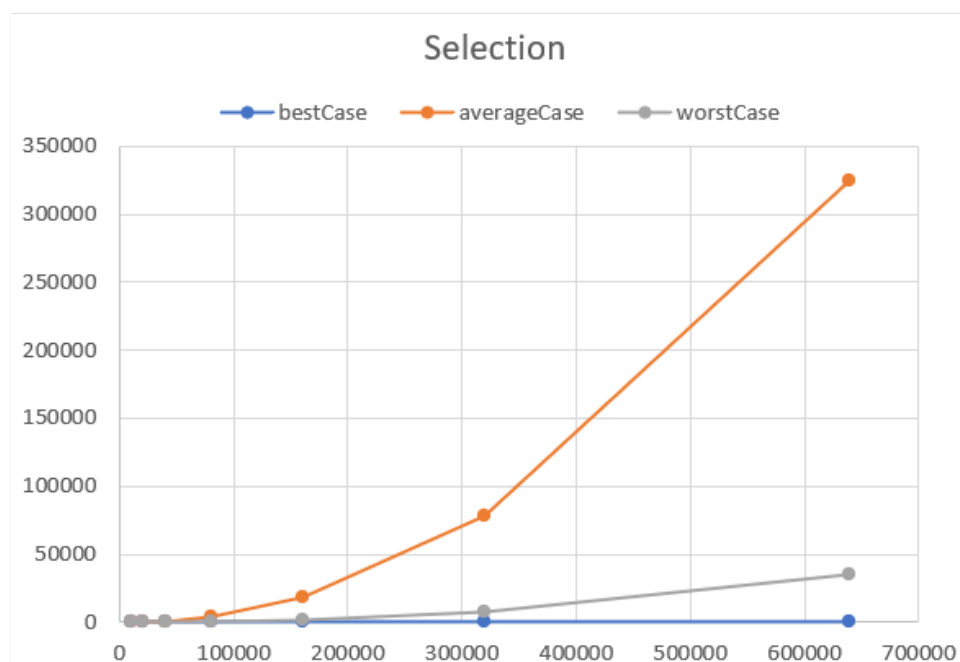
## Activity 1. Time measurements for sorting algorithms

Insertion algorithm:

$n$	$sorted(t)$	$inverse(t)$	$random(t)$
10000	0	88	38
20000	1	270	153
40000	1	905	198
80000	2	4109	470
160000	3	18871	1980
320000	4	78462	8064
640000	6	324463	35475
1280000			
...			
Until an exception is thrown			

The Insertion algorithm is an algorithm that in the best case will have a linear complexity but in the average and worst cases will have a quadratic complexity.

The obtained results make sense according to the complexities of the different cases in the sorted case (the best one) we have almost linear times, in the inverse case (worst case) we almost have a quadratic time and in random (average case) we have a quadratic complexity but lighter than the worst case.



Algorithmics	Student information	Date	Number of session
	UO: 276903	27/02/21	3
	Surname: Garriga Suárez		
	Name: Carlos		

### Selection algorithm:

$n$	$sorted(t)$	$inverse(t)$	$random(t)$
10000	18	31	12
20000	67	119	47
40000	236	431	191
80000	752	1873	657
160000	2575	8570	2576
320000	11267	28133	10544
640000	46600	133891	46916
1280000			
...			
Until an exception is thrown			

The Selection algorithm is an algorithm that in all cases will have a quadratic complexity.

Obviously, we are not having the same curve in the different cases. We could think that the worst case (inverse sorted vector) is going to be the one that lasts more and that the random sorted vector (average case) we will have a lighter curve in the graph and the last case (sorted vector) the time would be more reduced. But I obtained the following:



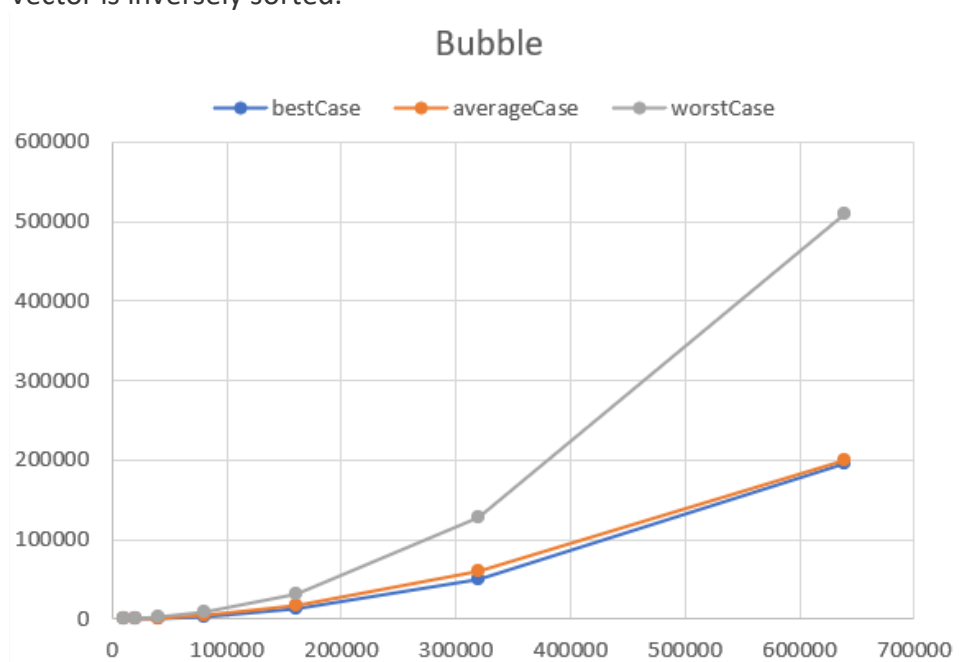
Algorithmics	Student information	Date	Number of session
	UO: 276903	27/02/21	3
	Surname: Garriga Suárez		
	Name: Carlos		

I observed that the times obtained from a vector that was already sorted were meaningless because it is not usual that the times obtained in the average case are better than the best case.

#### Bubble Algorithm:

$n$	$sorted(t)$	$inverse(t)$	$random(t)$
10000	56	89	109
20000	215	247	479
40000	824	902	1943
80000	3394	3926	8001
160000	13879	16399	32374
320000	49151	60406	127303
640000	196138	200383	508901
1280000			
...			
<i>Until an exception is thrown</i>			

With the bubble algorithm we are going to have always a quadratic complexity. As we can observe in the times measured, the worst case may be when the vector is randomly sorted, the best case of course when the vector is already sorted and the average case when the vector is inversely sorted.

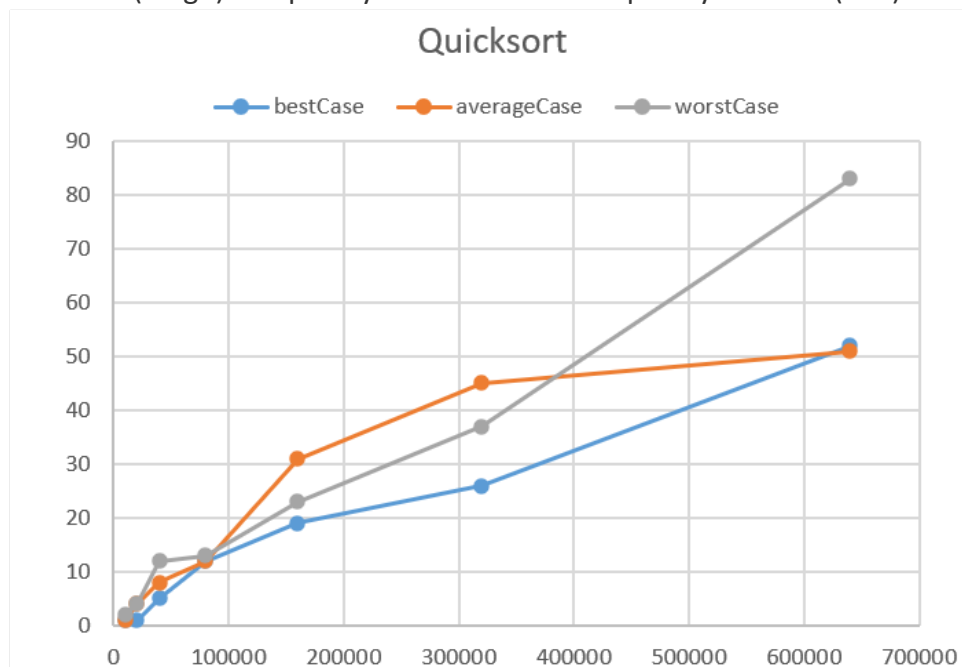


Algorithmics	Student information	Date	Number of session
	UO: 276903	27/02/21	3
	Surname: Garriga Suárez		
	Name: Carlos		

### QuickSort centralElement:

$n$	$sorted(t)$	$inverse(t)$	$random(t)$
10000	1	1	2
20000	1	4	4
40000	5	8	12
80000	23	12	13
160000	19	31	23
320000	26	45	29
640000	52	51	83
1280000	71	72	131
...			
<i>Until an exception is thrown</i>			

In the quicksort algorithm we have the following complexities: the best case and average cases will have a  $O(n \log n)$  complexity and the worst complexity will be  $O(n^2)$ .



As we can observe in the measured times, we can more less observe the complexities of the different cases.

Algorithmics	Student information	Date	Number of session
	UO: 276903	27/02/21	3
	Surname: Garriga Suárez		
	Name: Carlos		

## Activity 2. QuicksortFateful

In this case the algorithm is choosing as the pivot the leftmost element in the list. This can be very bad for measuring times with big vectors because the partitioning is more unbalanced. However, if your vector to be sorted is short you won't have very hard differences. The usual way of choosing a pivot is to choose the element that is more centered.