Algorithmics	Student information	Date	Number of session
	UO: 276244	24/02/21	2
	Surname: Beltran Diaz		
	Name: Martin		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Time measurements for sorting algorithms

INSERTION ALGORITHM:

N	SORTED	INVERSE	RANDOM
10000	12	220	79
20000	15	625	362
40000	0	611	433
80000	0	1626	1099
160000	1	8333	3861
320000	3	40131	22875
640000	8	197128	86835
1280000	14	1669942	409953

Due to its big complexity in the worst and average case, only 8 cases with nTimes = 1 repetitions could be measured, because for values of N greater than 1280000 it could take, not minutes, but hours to execute the worst scenarios. According to what has been studied in theory lessons, the expected complexities are:

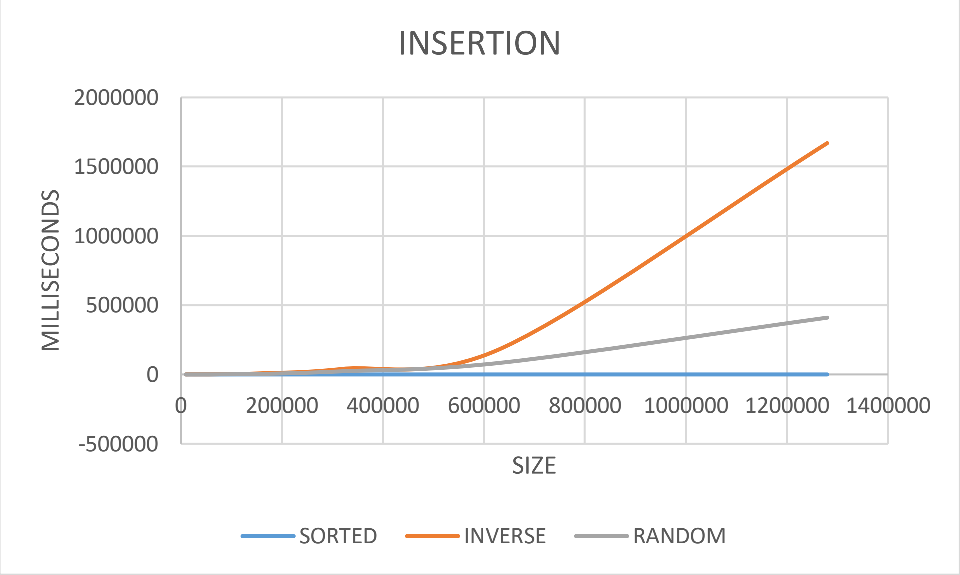
-Best Case: O(n)

-Worst Case: O(n^2)

-Average Case: O(n^2)

We can check that the obtained times follow the expected complexity trends:

Algorithmics	Student information	Date	Number of session
	UO: 276244	24/02/21	2
	Surname: Beltran Diaz		
	Name: Martin		

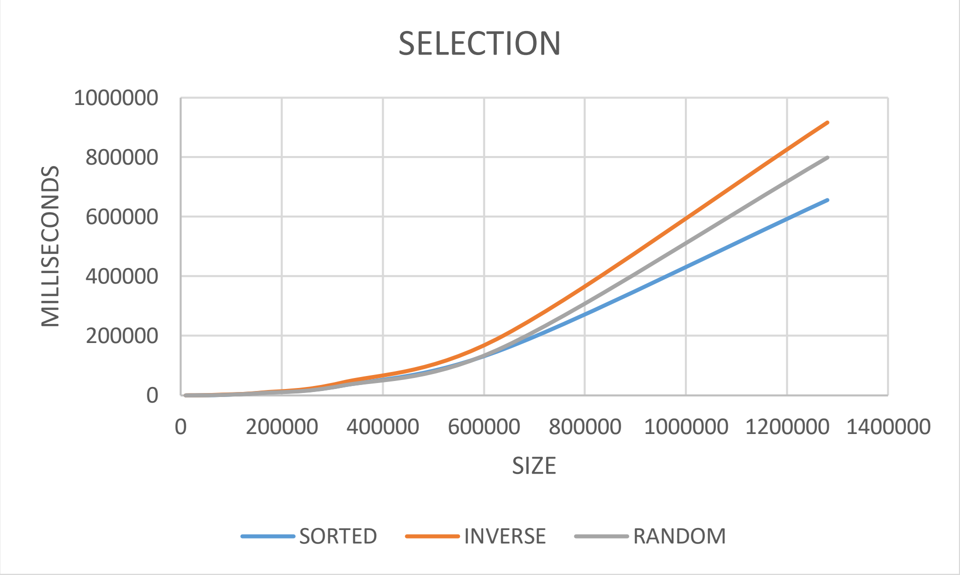


DELETION ALGORITHM:

N	SORTED	INVERSE	RANDOM
10000	48	64	34
20000	225	411	96
40000	576	527	366
80000	1520	2568	1554
160000	8733	9497	8009
320000	32949	43141	32914
640000	156118	201899	163064
1280000	656103	916625	798764

As in the previous algorithm, this measurement has been made under same conditions (n <= 1280000 and nTimes = 1). For the direct selection algorithm, the complexity is always quadratic, in the best, worst, and average scenario. We can prove this by looking at the measured time and its corresponding graph:

Algorithmics	Student information	Date	Number of session
	UO: 276244	24/02/21	2
	Surname: Beltran Diaz		
	Name: Martin		

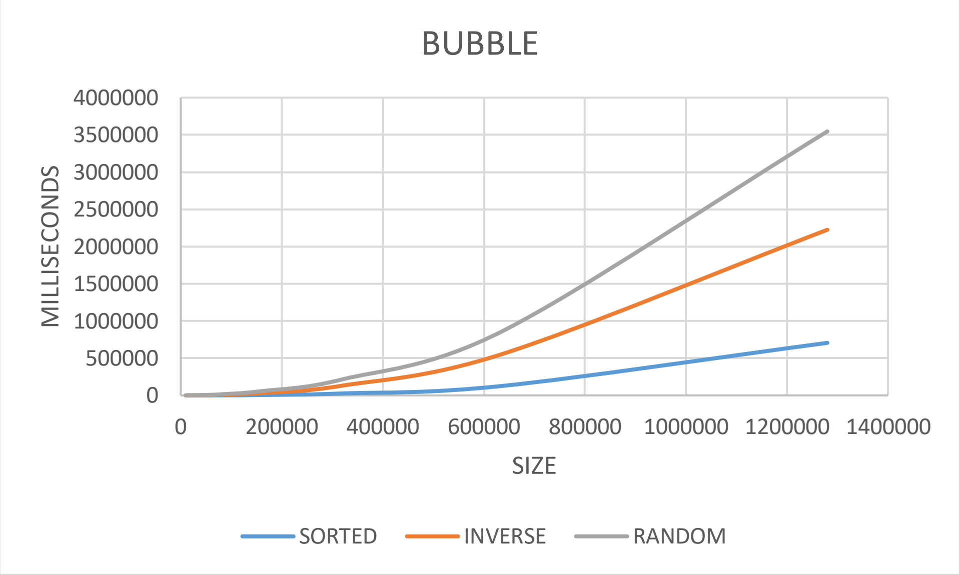


BUBBLE ALGORITHM:

N	SORTED	INVERSE	RANDOM
10000	37	317	376
20000	99	1240	1141
40000	383	2195	3756
80000	1455	8026	14776
160000	5273	31047	57141
320000	26407	131028	215448
640000	129584	563420	873954
1280000	706232	2225510	3547221

Beware of bubble algorithm!!! Even under the “easy”conditions as in previous experiments (N <= 1280000 and nTimes = 1), this algorithm takes a lot more time to be executed. Just compare, its time for biggest size in the average case is outrageously bigger than any of the measured times. This happens because, apart from having a complexity of O(N^2) in every scenario, bubble performs a high number of comparisons and exchanges.

Algorithmics	Student information	Date	Number of session
	UO: 276244	24/02/21	2
	Surname: Beltran Diaz		
	Name: Martin		

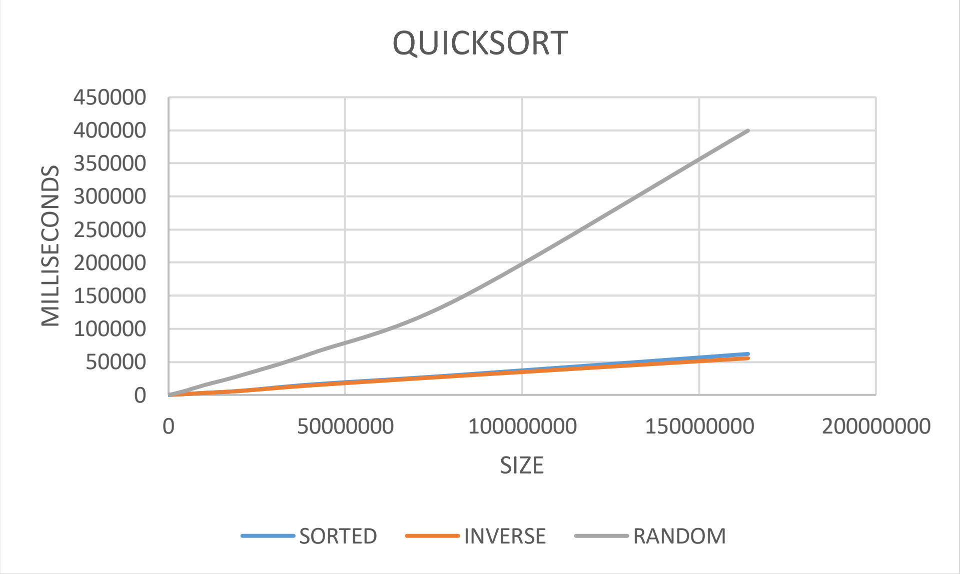


QUICKSORT WITH ENTRAL ELEMENT:

N	SORTED	INVERSE	RANDOM
10000	16	27	62
20000	11	23	23
40000	9	33	45
80000	28	25	166
160000	70	54	153
320000	109	107	302
640000	229	240	641
1280000	365	492	1358
2560000	709	673	3075
5120000	1428	1472	6739
10240000	3064	3095	14973
20480000	6213	6105	29617
40960000	15944	14603	63716
81920000	30135	28706	145015
163840000	61947	55447	399452

Algorithmics	Student information	Date	Number of session
	UO: 276244	24/02/21	2
	Surname: Beltran Diaz		
	Name: Martin		

In this case, thanks to its low complexity, a better experiment could be performed to measure the time. I could use the algorithm with values of N up to 163840000 and each measured was taken 10 times (nTimes = 10). Quicksort has a complexity of O(N*logN) in its best case, and O(N^2) in its worst case, so in an average scenario, its complexity should be O(N*logN). This is confirmed by the values measured, plotted in this graph:



Activity 2. QUICKSORTFATEFUL

In this version of the quicksort, we are using the left or the right element, that is, the first or the last one. Selecting any of those could be dangerous because they can be the biggest or lowest value, and that would enworse the complexity of the algorithm or even cause problems in it execution. While executing this algorithm in my computer, wheter it was sorted or inversed, only one measured was performed, when N = 10000, the next iterations threw StackOverfloewError exception, because it has to perform a lot of recursive calls.