Algorithmics	Student information	Date	Number of session
	UO: 283428		2
	Surname: Triana Fernández		
	Name: Carlos		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Time measurements for sorting algorithms.

Insertion:

n	sorted(t)	inverse(t)	random(t)
10000	0,4	40,8	8,8
20000	0,3	86,1	34,3
40000	0,1	377	122,6
80000	0,1	1329,4	704,8
160000	0,2	5445,3	2495,5
320000	0,2	19716,7	9775,4

The complexity of this algorithm changes significantly depending on the case under which it performs. The temporal complexity under the best scenario, where the sample is already sorted, is linear. Under any other case, the complexity is quadratic.

Although the results are not high enough to be reliable, we can clearly see how the linear complexity of the sorted case outcomes the fastest results. The differences between the inverse and the random cases lie on the fewer iterations that must be executed once the pivot to be inserted is on its position.

Algorithmics	Student information	Date	Number of session
	UO: 283428		2
	Surname: Triana Fernández		
	Name: Carlos		

Selection:

n	sorted(t)	inverse(t)	random(t)
10000	40,5	43,5	81,8
20000	56,4	175,7	317,2
40000	201,4	774,8	1297,8
80000	662,6	3254,4	5957,4
160000	3315,3	12053,7	203789
320000	12358,9	46016,8	77149,6

Although the complexity of the algorithm is quadratic under all cases, this algorithm is also affected by its scenario. It relies on a high amount of comparations to select the lowest element. Therefore, the difference between cases is finding the lowest element. In the sorted scenario, the minimum value is the first, and in the inverse, it is necessary to adjust the minimum value n times. The random column has unexpected high values, as they shouldn't be higher than the inversed ones.

Bubbles:

n	sorted(t)	inverse(t)	random(t)
10000	23,6	77,3	101,7
20000	83,5	289	491,1
40000	355,2	1165,9	2137,2
80000	1667,8	4702,4	8926,6
160000	5574,6	18936,7	35924,3
320000	21289,2	75410,9	144896,8

Algorithmics	Student information	Date	Number of session
	UO: 283428		2
	Surname: Triana Fernández		
	Name: Carlos		

The bubbles algorithm is the least complex one. Through each iteration, all the elements in the array that are not sorted yet are compared with the adjacent ones. If any left's element is greater, they will swap.

This behavior is the same no matter the case, though, the number of interchanges depends on the scenario.

Because of this, the complexity of the algorithm is always O(n^2).

Quicksort:

n	sorted(t)	inverse(t)	random(t)
10000	0,163	0,235	0,865
20000	0,169	0,328	2,108
40000	0,345	0,666	4,28
80000	0,8	1,213	9,357
160000	1,484	2,639	19,86
320000	3,2	5,425	37,867
640000	6,387	11,355	67,556
1280000	16,494	23,531	139,839

The quicksort algorithm is based on the consecutive partitioning of the array into subarrays until the union of all of them is the final solution. To perform the process of dividing the problem into smaller ones, it is necessary to choose the pivot that will divide the array into two others. Choosing the pivot is related with the complexity of the algorithm. When picking a good pivot, the result is always a binary tree that will lead to a complexity of O(log n). In this version of the algorithm, we always choose as a pivot the element in the middle of the array.

Algorithmics	Student information	Date	Number of session
	UO: 283428		2
	Surname: Triana Fernández		
	Name: Carlos		

We can see the logarithmic complexity with the inversed and already sorted scenarios. On these cases, the median (the real or imaginary number that divides the sample in two equal halves above and below that value) will match the middle point of the array. This way, the algorithm divides the problem in equal parts that will lead to a binary tree like structure.

Activity 2. Quicksort Fateful

n	random(t) Middle	random(t) First
10000	0,865	0,743
20000	2,108	1,506
40000	4,28	3,126
80000	9,357	6,731
160000	19,86	14,197
320000	37,867	30,174
640000	67,556	62,903
1280000	139,839	131,334

The Quicksort Fateful, is a version of the Quicksort algorithm where the pivot is always the first element in the subarray. Selecting a wrong pivot in the Quicksort algorithm causes a worse complexity of the algorithm.

Generally, this method of choosing the pivot doesn't provide the worst performance, despite being a poor option. However, in the first two scenarios, where the elements are sorted and inversed, each iteration will make an array of size n -1 and another of 1 because of the ascending or descending order of the elements. Due to this, the size of the problem only reduces itself in one unit with each recursive call.

This only happens in the scenarios where the elements have an order, in any other context, the algorithm has the expected performance.

Algorithmics	Student information	Date	Number of session
	UO: 283428		2
	Surname: Triana Fernández		
	Name: Carlos		

Measurements taken on: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz

16,0 GB (15,6 GB usable)