Algorithmics	Student information	Date	Number of session
	UO: 276244	13/03/2021	3.2
	Surname: Beltran Diaz		
	Name: Martin		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. COUNTING INVERSIONS

We have developed two ways of counting inversions when sorting a vector. The first one is an algorithm with complexity O(n^2) and the second one with complexity O(n * log(n)), by means of implementing a modified version of MergeSort. The main difference of this modified version is that it counts how many inversions have been performed by means of some additional counters in merge() method whose functionality will be explained now:

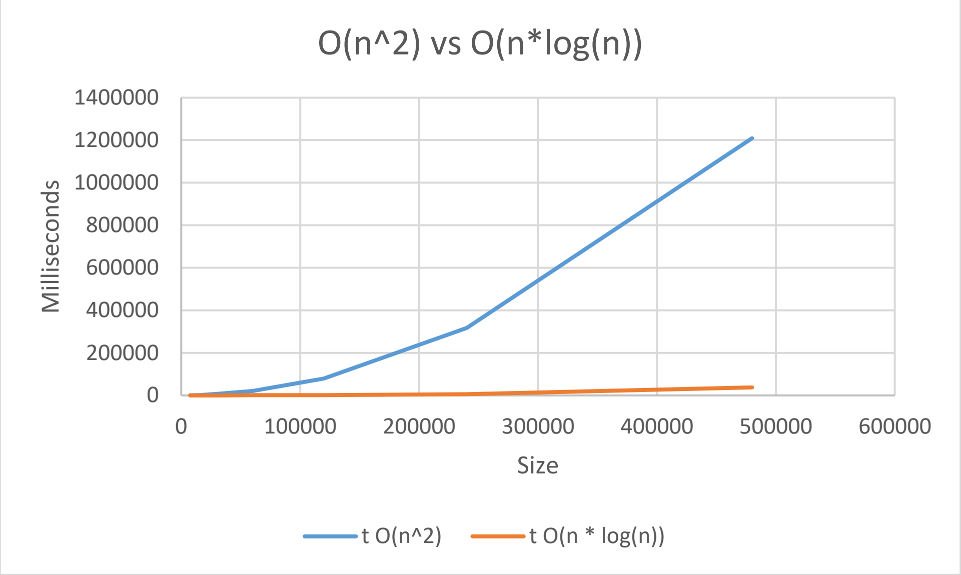
- **auxPointer**: counter developed to allow the iterations in the original elements list starting in the beginning of the left partition of this one.
- **elemsMovedFromAuxLists**: counter that is incremented every time that the first element in the left sub list is grater that the first element in the right sub list.
- **inversionsCounter**: count the number of inversions performed, it is calculated adding in ever iteration of the loop the correspondent number of elements moved due to the last setting performed.

Now let us discuss the obtained measurements:

FILE	t O(n^2)	t O(n * log(n))	t O(n^2)/O(n * log(n))	INVERSIONS	SIZE
Ranking1.txt	330	90	3,666666667	14074466	7500
Ranking2.txt	989	256	3,86328125	56256142	15000
Ranking3.txt	7210	362	19,91712707	225312650	30000
Ranking4.txt	20622	1336	15,43562874	903869574	60000
Ranking5.txt	79079	1655	47,78187311	3613758061	120000
Ranking6.txt	317825	6370	49,89403454	14444260441	240000
Ranking7.txt	1209368	42590	28,39558582	57561381803	480000

If we plot the obtained results we can see that the obtained times follow the expected trend:

Algorithmics	Student information	Date	Number of session
	UO: 276244	13/03/2021	3.2
	Surname: Beltran Diaz		
	Name: Martin		



As we can see, the trends of growth are the expected ones. Now let us check their division: if we divide n^2 by n * log(n), the result will be n/log(n).

I will not show the plot because as the measurements are only performed once per start() invocation, there is a lot of “noise” in the graph, and I wouldn’t portray its real representation.