
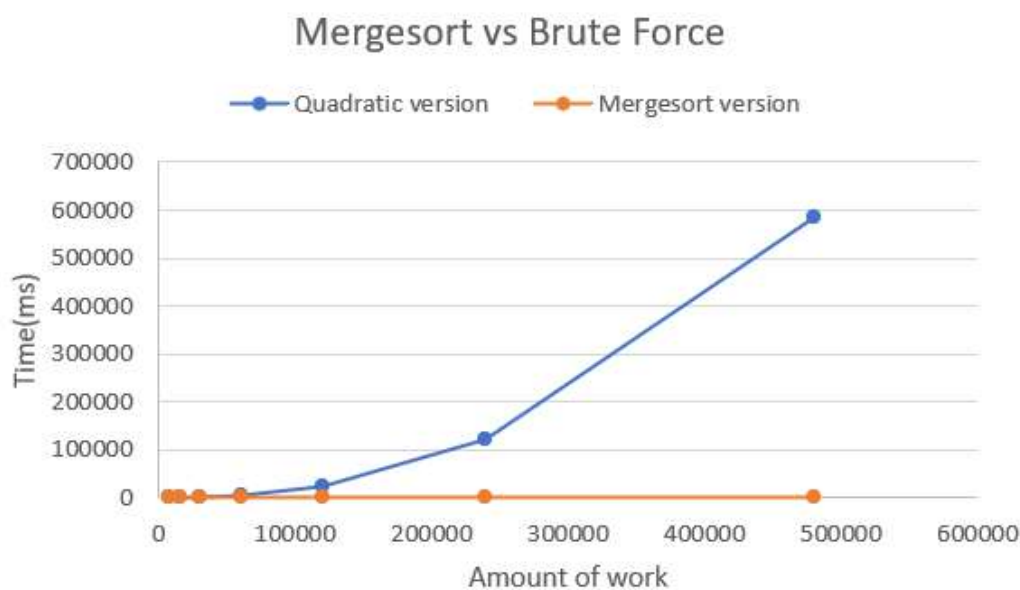


Algorithmics	Student information	Date	Number of session
	UO:276903	15/03/21	4
	Surname: Garriga Suárez	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Carlos		



Activity 1. Counting inversions

<i>file</i>	<i>t O(n²)</i>	<i>t O(n log n)</i>	<i>t O(n²)/t O(n log n)</i>	<i>n inversions</i>
Ranking1.txt	97	12	7,83	14.074.466
Ranking2.txt	217	7	31	56.256.142
Ranking3.txt	888	15	59,2	225.312.650
Ranking4.txt	3995	16	250	903.869.574
Ranking5.txt	22567	31	727,96	3.613.758.061
Ranking6.txt	123752	96	1289,03	14.444.260.441
Ranking7.txt	586564	162	3620,76	57.561.381.803



As we can observe in the sample of data, the mergesort algorithm applied to count the number of inversions is much better than doing this by means of brute force which will result in a quadratic complexity. We can also see that mergesort is much better compared with the other method when you have to calculate the time spent in counting the inversions with a big n .

Algorithmics	Student information	Date	Number of session
	UO:276903	15/03/21	4
	Surname: Garriga Suárez		
	Name: Carlos		

Let's do the theoretical times of Ranking4 with complexity $O(n^2)$. With ranking3 we have a n of 30000 and it lasts 888 ms. The expected time for Ranking4 with $n = 60000$ will be the following:

$$\begin{array}{lll} n_1 = 30000 & t_1 = 888\text{ms} & N2 = K * N1 \\ n_2 = 60000 & t_2 = ? & \end{array}$$

$$\text{Then } t_2 = (f(n_2)/f(n_1)) * t_1 = n_2^2 / n_1^2 * t_1 = k^2 * t_1 = 4 * t_1 = 4 * 888 = 3552$$

The expected time is less than the one obtained (3995). This is because the computer may be doing other tasks. Anyway, is a similar one.

Now let's do the same with mergesort complexity.

Let's do the theoretical times of Ranking7 with complexity $O(n \log n)$. With ranking6 we have a n of 240000 and it lasts 96 ms. The expected time for Ranking4 with $n = 60000$ will be the following:

$$\begin{array}{lll} n_1 = 240000 & t_1 = 96\text{ms} & N2 = K * N1 \\ n_2 = 480000 & t_2 = ? & \end{array}$$

$$\text{Then } t_2 = (f(n_2)/f(n_1)) * t_1 = n_2 \log n_2 / n_1 \log n_1 * t_1 = k * \log n_2 / \log n_1 * t_1 = 101,37$$

The expected time is less than the one obtained (162). This is because the computer may be doing other tasks. Anyway, is a similar one.

Regarding the brute force algorithm results in a quadratic complexity as I am comparing all the elements with the ones that are after it. We get two nested loops that result in a complexity $O(n^2)$.

Finally, we can also deduce that mergesort has a complexity $O(n \log n)$ as it is based on D&C strategy by division. The problem is split in two halves so we will have $a = 2$, $b = 2$ and $k = 1$

As $a = b^k$, then we can apply the formula and we get $O(n \log n)$. We get that $k = 1$ when we sort the elements of the two external arrays which is using a for loop.