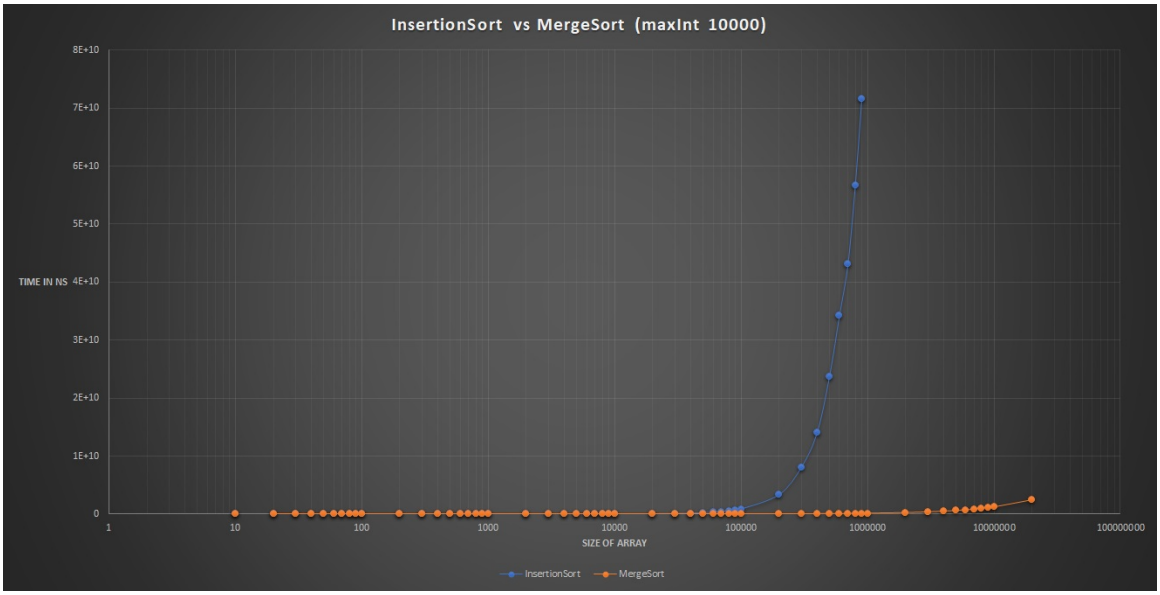
Algorithm and Data Structures - ID 1020
TCOMK2 - Lab 2

Leonardo Filippeschi

20 september 2020

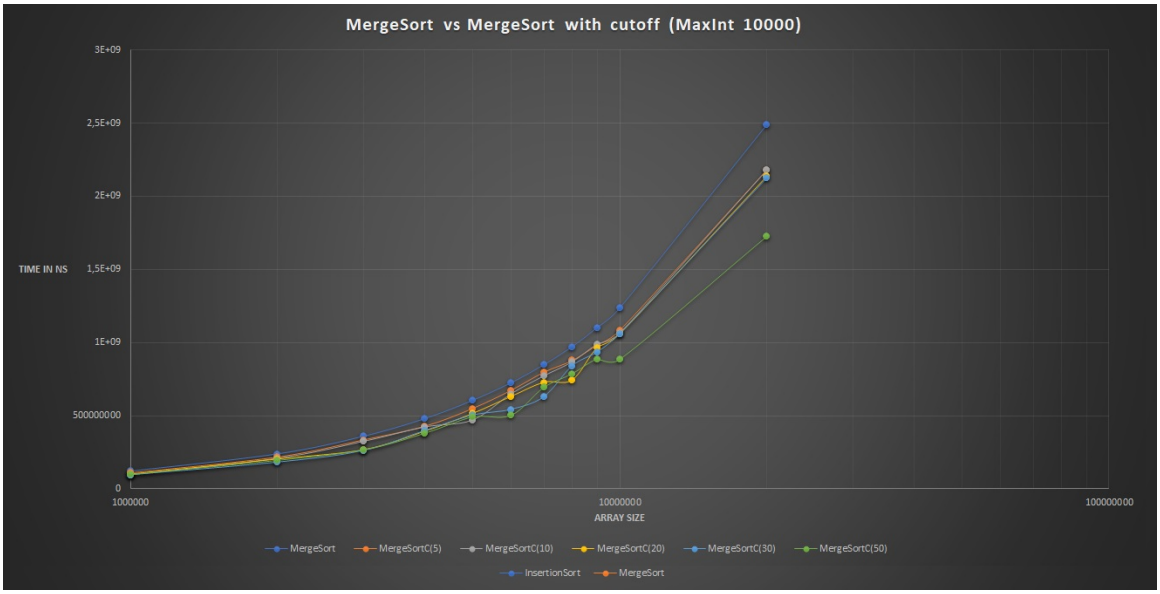
lfil@kth.se - 19950609-0233

Insertion Sort vs Merge Sort:



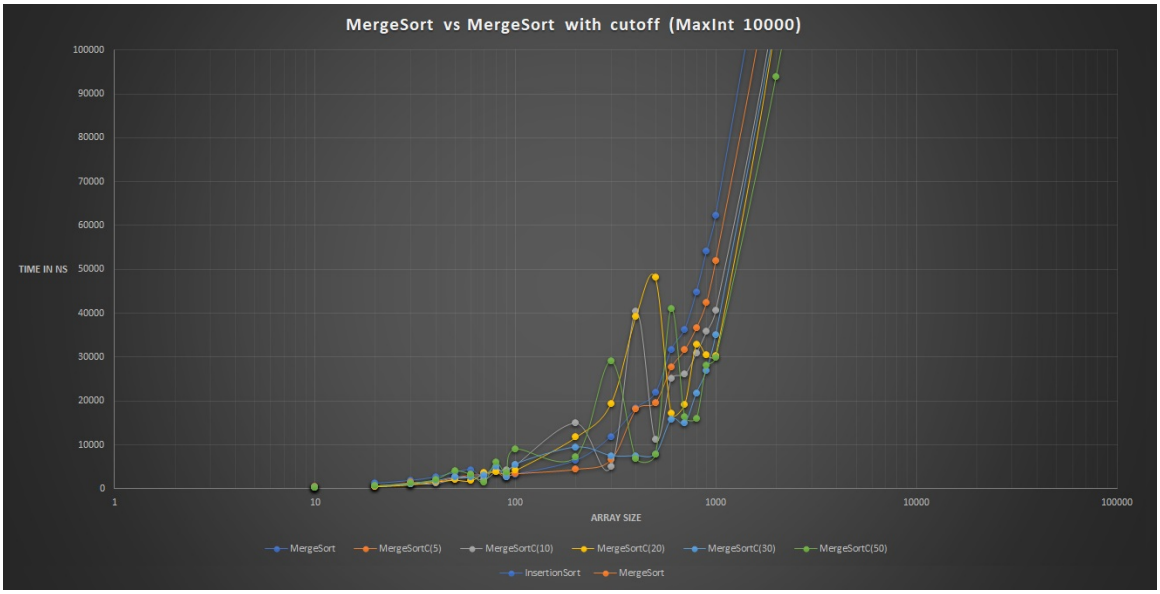
Looking at the graph we clearly see the difference between Insertion Sort and Merge Sort. The first one is in the time complexity of $O(N^2)$ while the second one belongs to $O(Nlog(N))$ in all the cases, best, worst and average.
The main advantage of insertion sort is when the size of the array to be sorted is very small: less than 70 approximately and in these cases insertion sort is actually faster.
Another advantage of insertion sort is that the algorithm is in place, which means that the memory needed is $O(1)$. The memory complexity of merge sort is $O(N)$ and in cases where memory is an issue, we might prefer to use insertion sort.

Merge Sort with cutoff:



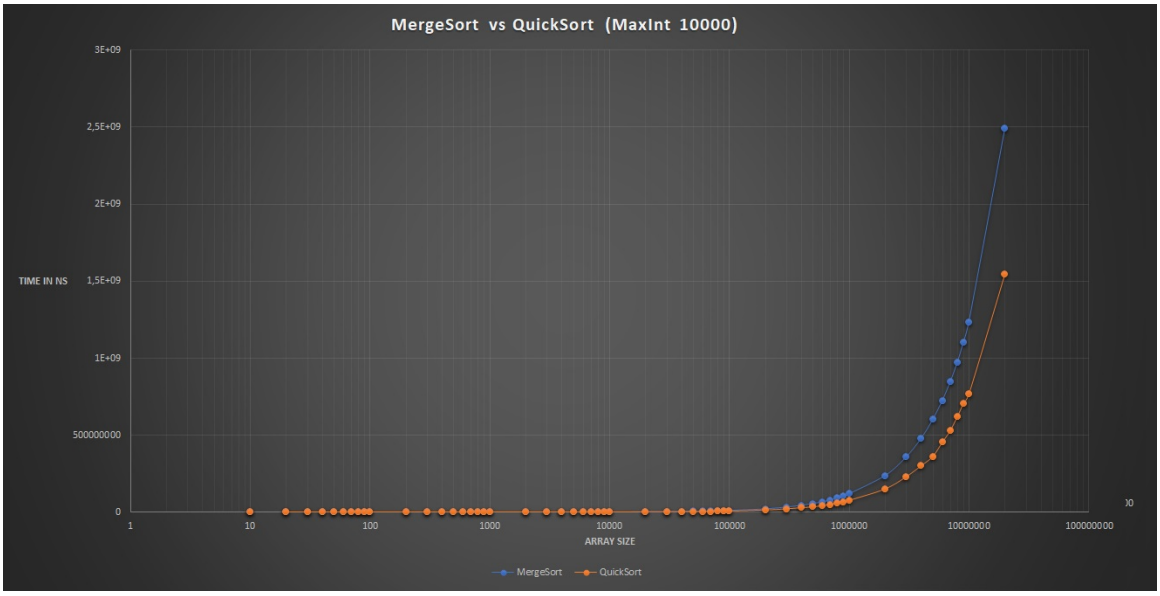
The cutoff to insertion sort from merge sort is useful because insertion sort for array values smaller than 70 is faster than merge sort. This brings some improvements as we can see from the image

above.
The best size for the cutoff value is dependent on the total size of the array, for smaller arrays a value of around 30 is preferable while with big arrays larger values are better. This is due to the recursive divisions in merge sort.



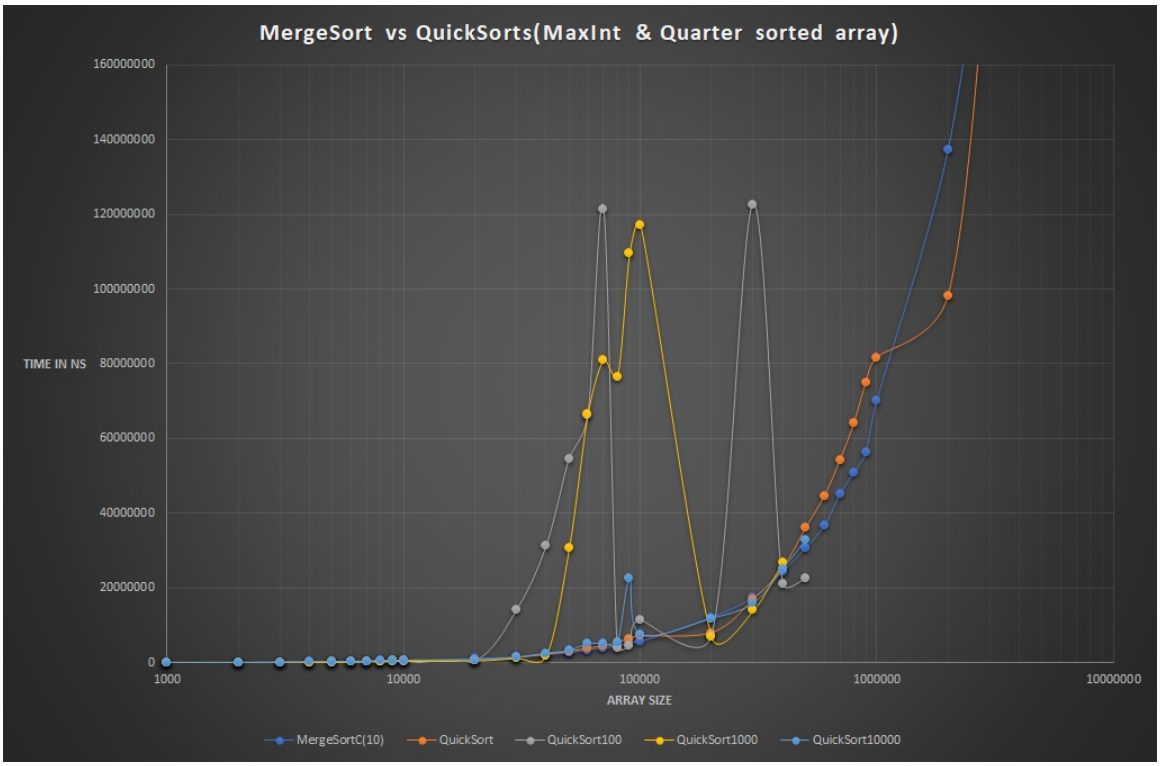
From the graph above we see the better value of 30, even though it's still hard to come up with the best value. In general it's always better to use the cutoff to insertion sort.
It's important to keep the cutoff small though as increasing it will slowly bring the execution time closer to the one of insertion sort.

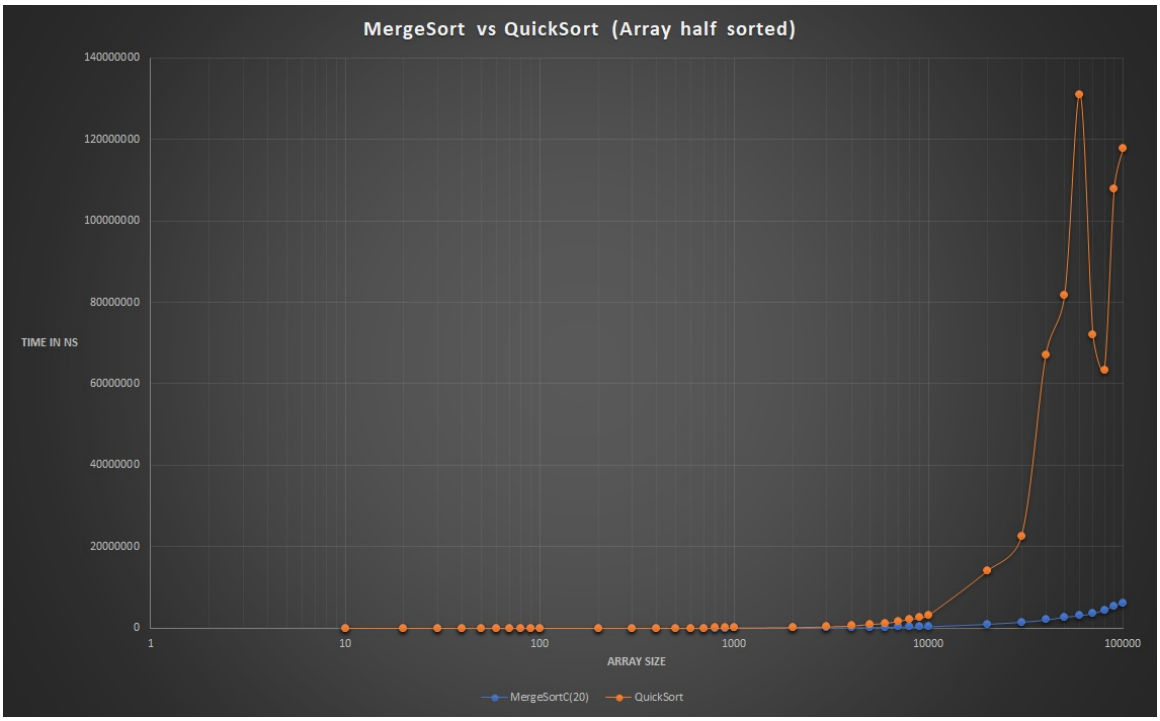
Quick Sort vs Merge Sort:



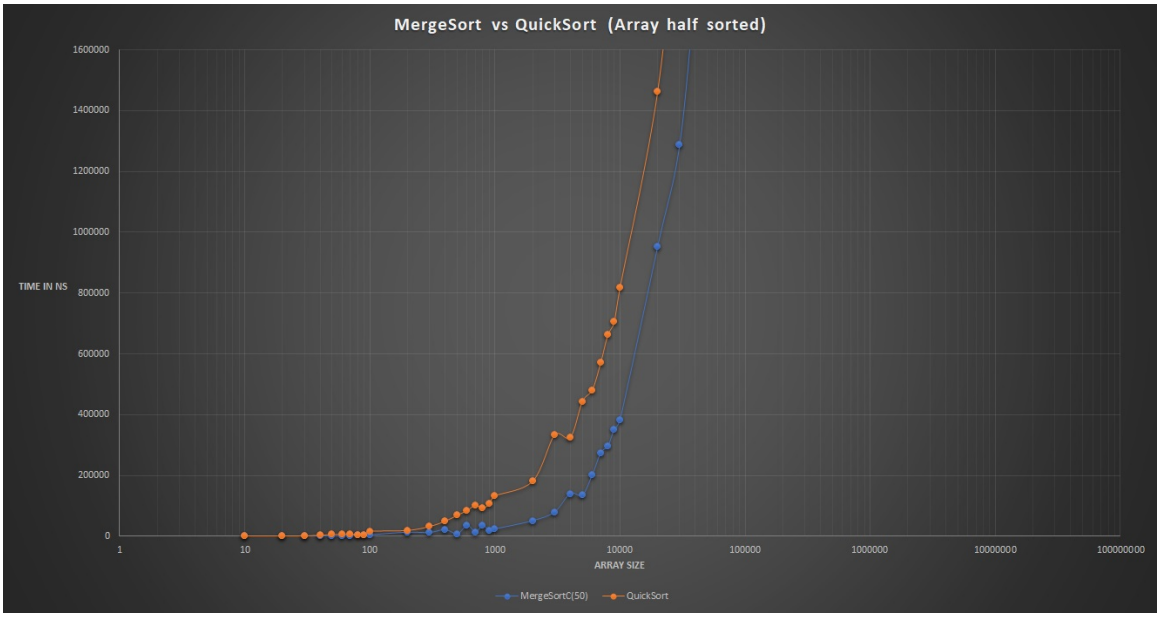
In the general case quick sort is faster than merge sort and that is to be expected. This happens when the data inside the array is not too close to each other and is shuffled, this fact prevents the array from having multiple same elements and degenerating to its worst time complexity which is

$O(N^2)$.
The benefit of quick sort over merge sort is that the algorithm is in place while merge sort always uses $O(N)$ memory. A disadvantage of quick sort is that the algorithm is not stable while merge sort is stable.
The biggest downside of quick sort is that the algorithm degenerates to $O(N^2)$ in the worst case scenario, which happens when the array is partially sorted or the size of the input is very small with a big array.





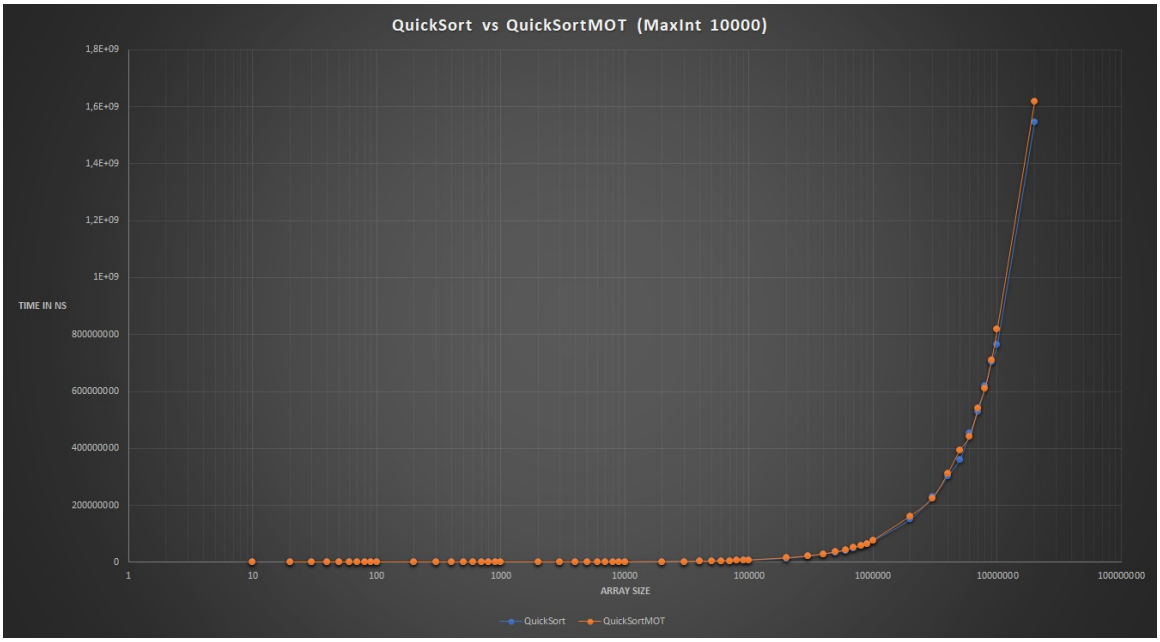
From the graph above we clearly see this scenario where quick sort's time complexity increases drastically.
The first one to degenerate is when the input is limited to a maximum of 100 and the other follow. We can also zoom in for smaller sizes of arrays and we see the same behavior.



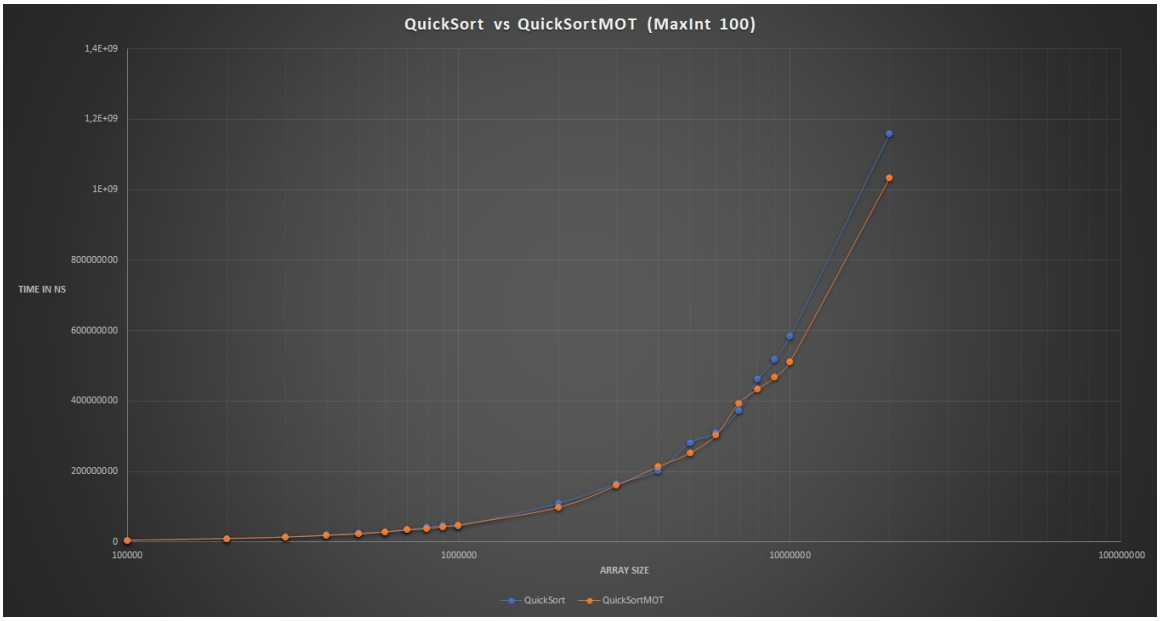
In conclusion when we know the qualities of the input and we that the elements are very large and the array is not partially sorted, we can chose quick sort. We could also shuffle the array but that adds more operations and reduces the efficiency and the advantage of quick sort.
Also when memory complexity is an issue we can implement quick sort being sure to control the input so that it falls under the average case.

Quick Sort vs Quick Sort Median of three

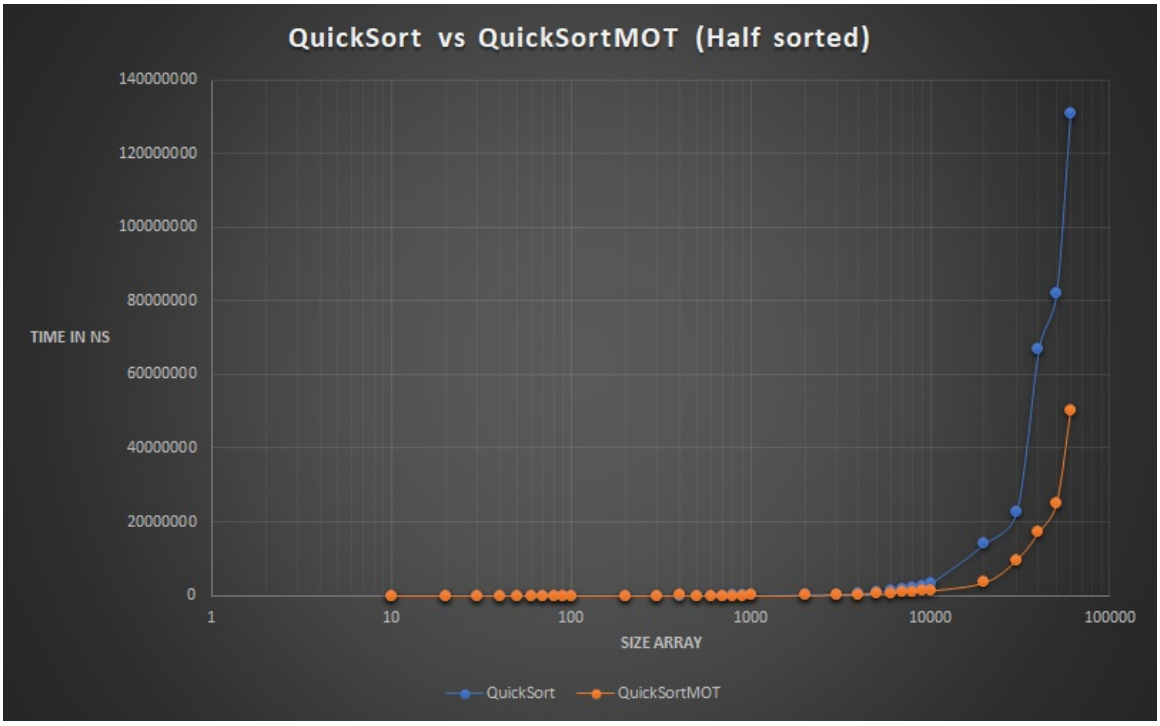
The purpose of the median of three algorithm is to try to minimize the chance of the worst case to happen. The way it works is by choosing the pivot element between the first, middle and last element of the sub arrays. This method comes at an initial cost because we have to go through the elements of the array and swap them so that the pivot ends either in the first position or last. This increases the time complexity for small arrays but it's beneficial in the long run and prevents the algorithm from degenerating.



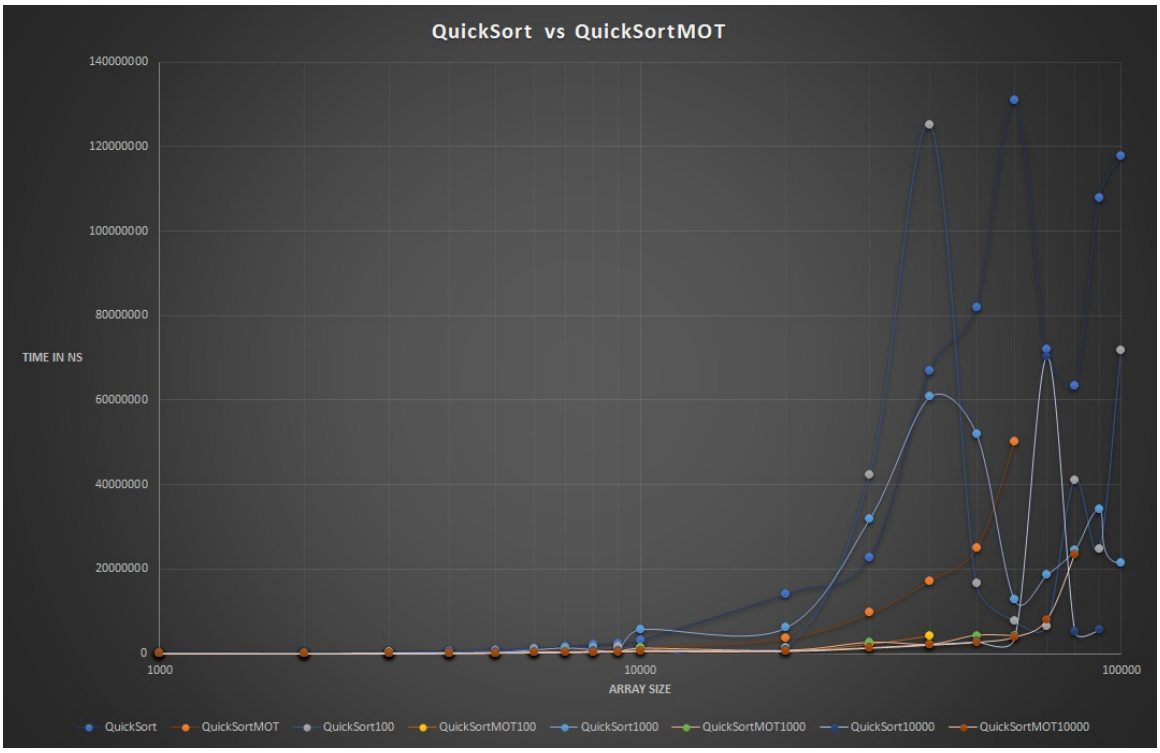
In the case where the array is shuffled and the range of the inputs is large, the Median of three method doesn't bring any advantages.



In the above scenario the array is shuffled but the max range of the inputs is 100 and here we start seeing the benefits of choosing one method over the other. In the next graph we'll analyze how a half sorted array affects the two algorithms.



We can immediately observe how the array size is limited in this case. We went from an array of 10 million elements to one of less than 100 thousands.



From the above graph we have the case where the array is half sorted and different cases for the max range of the input. In all the cases the Median of three algorithm is always better compared to the regular quick sort. The max range affects how early the algorithm will degenerate.