Comparisons between merge sort and insertion sort execution times

Maria Halvarsson, mahalv@kth.se

To compute the average times of insertion sort and merge sort, a random integer array was used that generated random numbers between 0 and 1000. The average time was calculated by running the sorting algorithm multiple times. For merge sort, the average time out of 1000 iterations was calculated with different sizes of arrays (from 10 – 1 000 000). The same method was used to calculate insertion sort except for when reaching array sizes bigger than 10 000. Arrays bigger than 10 000 took much longer to sort, so, average time was calculated from fewer iterations. The average execution time was also calculated for partially sorted arrays, where half of the array was already sorted.

Looking at the different graphs depicting merge sort and quick sort we can see that insertion sort quickly becomes very slow for large inputs (see figure 2 and 3 below). Since insertion sort has a worst case time complexity of O(n^2), this result was expected. Comparing insertion sort and merge sort we can see it has a much flatter curve than insertion sort. The worst case time complexity for merge sort is O(*n log n*).

Merge sort seem to work better for arrays where the integers are generated in random order. The difference in execution time between merge sort and insertion sort becomes very significant when the array contains more than 100 000 elements. When the array is smaller than 100 elements, insertion sort seems to have an advantage, especially when the array is partially sorted as we can see in table 2.

These advantages of merge sort and insertion sort can be used to optimize the implementation of merge sort by only dividing the array in merge sort until the arrays are small and then using cut-off to insertion.

To conclude, merge sort works better for larger arrays where the data is unsorted or randomized. Whereas insertion sort works better for smaller arrays (< 100 elements) where the data is close to or partially sorted.

Table 1: Execution times of insertion sort and merge sort where the input is random

Number of inputs	Insertion sort (ms)	Merge sort (ms)
10	0.001992	0.004092
100	0.01895	0.01711
1 000	0.1532	0.1130
10 000	12.64	1.089
100 000	1275	11.64
1 000 000	131691	126.1

Table 2: Execution times of insertion sort and merge sort where the input is partially sorted

Number of inputs	Insertion sort (ms)	Merge sort (ms)
10	0.001985	0.003993
100	0.01024	0.01692
1 000	0.09031	0.08744
10 000	3.299	0.8174
100 000	326.2	9.025
1 000 000	94271.299475	113.4

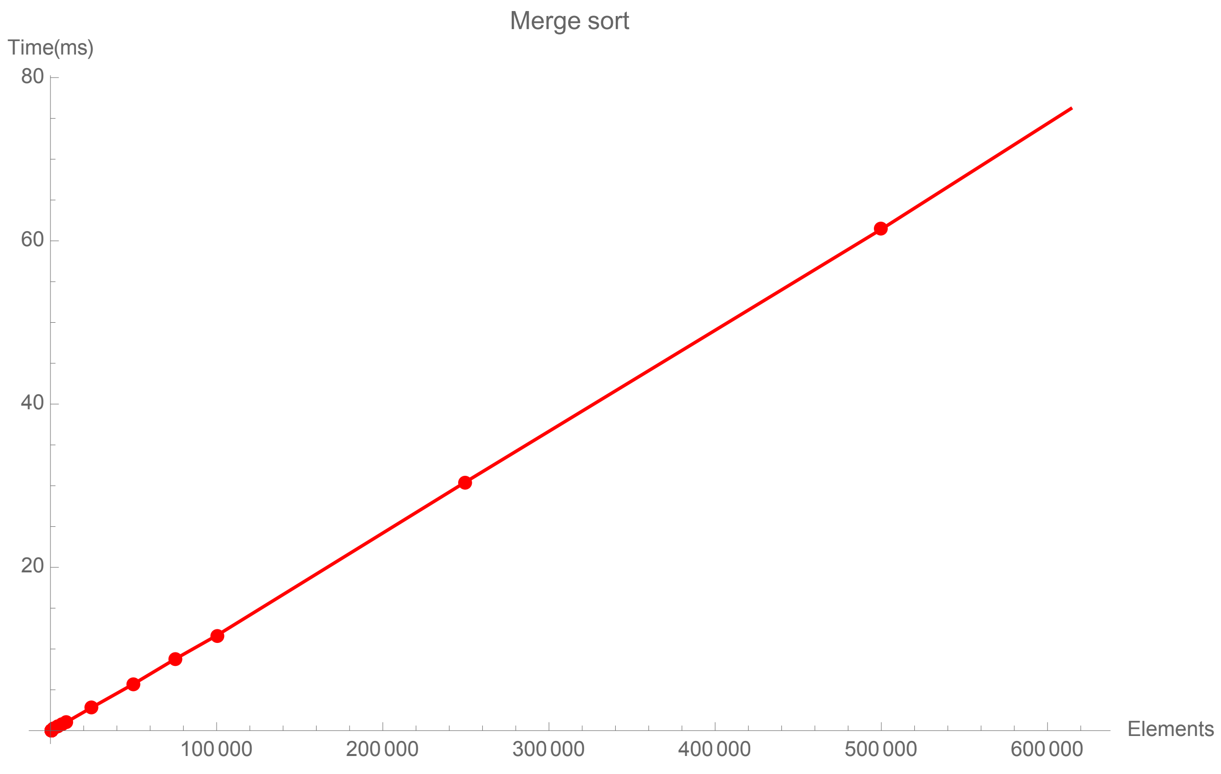

Figure 1: Merge sort execution time

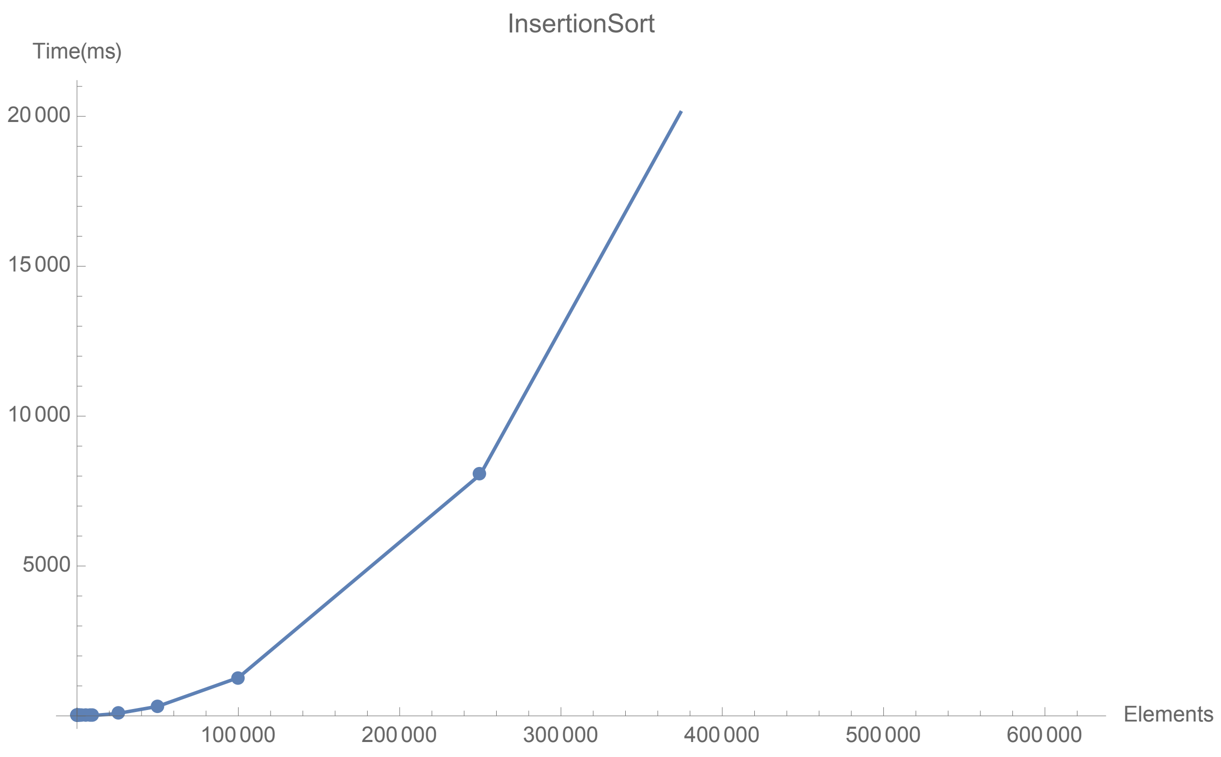

Figure 2: Insertion sort execution time

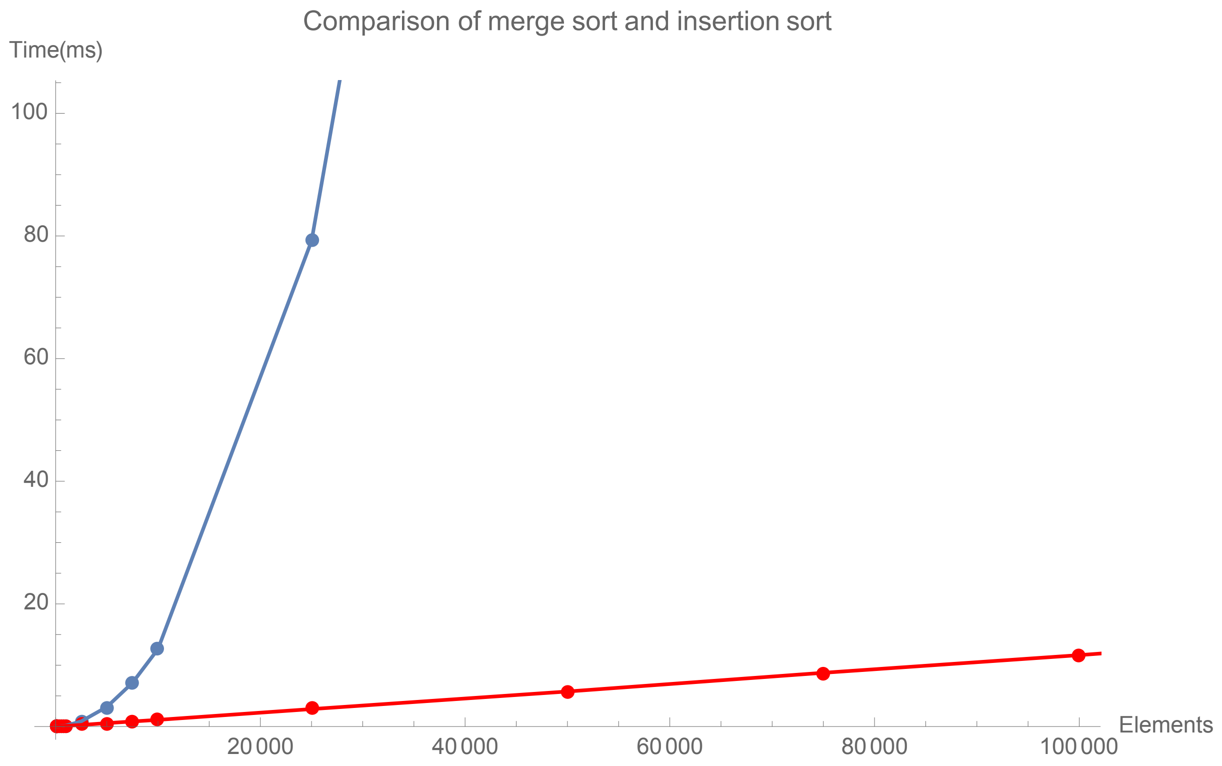

Figure 3: Comparison between merge sort (red) and insertion sort (blue)