Syed Naqvi

Dec 6th 2023

Professor Smith

CS 2336

The difference in computation of various sorting algorithms

Given different sorting algorithms: Bubble sort, selection sort, insertion sort, quick sort, and merge sort, all of them can be categorized as either O(n^2) or O(log(n)) time complexity. Bubble sort, selection sort, and insertion sort are of O(n^2) time complexity, while quick sort and merge sort are O(log(n)). These categorizations were derived from testing the algorithms on different-size arrays of non-sorted elements, with 10 trials each (n=10). The average completion times in milliseconds are listed below:

	nums[50]	nums[500]	nums[5000]
Bubble sort	0.2	20.3	3258.4
Selection sort	0.1	9.6	1243.8
Insertion sort	0.1	9.7	1224.2
Quick sort	0.1	0.7	14.1
Merge sort	0.1	1.9	31.3

The data shows that as array "nums" grows in size by a factor of 10, the millisecond time to complete each algorithm scales by around N^2 for the first three and around log (n) for the last two. Out of these three N^2 algorithms insertion sort seems to be the quickest, while selection is not far behind, despite this bubble sort seeming to be the most ineffective, around 2.5 times more ineffective than the other two. Comparing the two log(N) algorithms quick sorts seem to be quicker than merge sort by around a factor of two in terms of time. Doing an intergroup comparison between insertion sort, quick sort, and merge sort, we find that the log(N) algorithms are many times more efficient in time when compared to the fastest N^2 sort. Comparing the average number of swaps (s) in these functions:

Swaps | Compares	nums[50]	nums[500]	nums[5000]
Bubble sort	630 | 1225	65049 | 124750	6204912 | 12497500
Selection sort	615 | 1225	62033 | 124750	659834 | 12497500
Insertion sort	325 | 600	32344 | 63736	29945 | 6256203
Quick sort	193 | 401	1202 | 3974	23144 | 46940
Merge sort	| 419	| 4552	| 55156

The swaps and the comparisons together help paint a picture of why some algorithms are faster than others, for example, bubble sort had many more comparisons and swaps relative to quick sort. Bubble sort and selection sort have to iterate and compare elements through the entire array multiple times therefore they have a high number of comparisons. Bubble and Selection sort had similar swaps and compares, growing exponentially (N^2). Insertion sort had around half the number of comparisons of bubble and selection and therefore had fewer average swaps on average. On the other hand, quick and merge sorts had a slower growth with the number of comparisons (slightly faster than linear), this is consistent with N log(N). This is most likely due to their divide-and-conquer nature which speeds up the process similar to storing elements in a binary tree. I expected this difference since they roughly follow their mathematical model in the O(N) complexities given. The differences in time can be found using average comparisons, since O(N) complexity only tells about how well an algorithm scales it does not as easily convey how an algorithm will compare to another of its type. Since bubble sort goes through N items and selection sort goes through N-1, by this small difference, on average, selection sort will be quicker even though they are both O(N^2) complexity. I believe the same is going on with quick and merge sort.

Overall the log (N) algorithms performed significantly faster than N^2 ones, one reason being the reduction in computations required to produce the result, if using a sorting algorithm the best seems to be quick sort since it did slightly outperform merge sort.