CS372

Lab2 Report

Quick sort runtime analysis

The quick sort algorithm has a worst-case runtime of O(n^2). When an already sorted array is put through the quick sort algorithm, using the lowest index as the pivot, then the runtime will reach O(n^2). Quick sort has an average case and best case of O(nlogn). For the running time complexities, it depends on how even the partitions are. In a worst case the partitions are most unbalanced as the array is recursively called for n-1 each time. In a best case, the size of the partitions is as evenly distributed as they can be, each subproblem is cut in half each time. As for the average case, there is a 50/50 chance that the partition will be the worst case or 3-to-1 split or better.

Merge sort runtime analysis

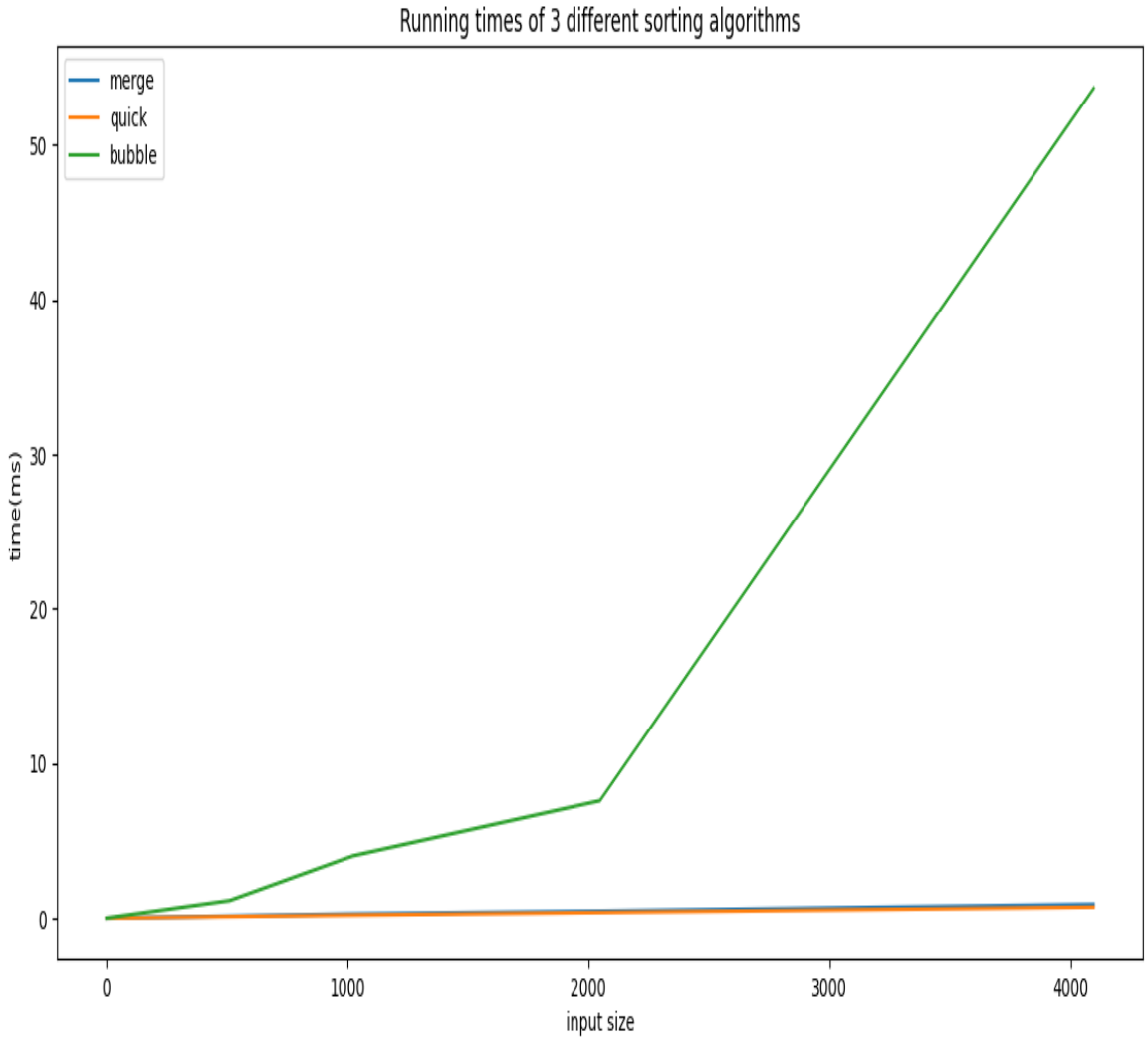
The worst-case runtime for merge sort is O(nlogn). Similarly, the runtimes for the best and average cases are also O(nlogn). Unlike quick sort, merge sort will always divide the subproblems evenly in halves. Similar to a best-case quick sort runtime, each input can only be divided logn amount of times and because it runs through the entire array it is multiplied by n giving it a complexity of O(nlogn).

Sorting 2gb of data

I would use quick sort to sort the 2gb of data. In the average case quick sort would give a runtime of O(nlogn). Quick sort is an in-place sorting algorithm, meaning that it will save memory when executing a large set of data. Merge sort would not be as practical since it creates more arrays to work on its sub problems giving it higher space complexity. Lastly, I would not use bubble sort because its average time complexity is O(n^2).

Observation

Bubble sort shows exponential growth and quick sort and merge sort show a much smaller running time even as time progresses just as it was analyzed previously


Running times of 3 different sorting algorithms
merge
quick
bubble
time(ms)
0
10
20
30
40
50
0
1000
2000
3000
4000
input size