Apollo Truong
CS014 – Summer 2016
Professor Rusich
July 20, 2016

Assignment 4 Report

==========

Quicksort

==========

1. What is the worst-case running time of the sorting algorithm? Explain.

The worst running time of the sorting algorithm would have to be O(n^2) in a quicksort. This being said, choosing a maximum, or minimum of a data set as its pivot will force the program to run recursively and pivot as many times as possible, giving quick sort a bad runtime at O(n^2), although this is highly unlikely, but definitely possible.

2. What is the average-case running time of the sorting algorithm? Explain.

The average-case would definitely be O(nlog(n)) runtime complexity. The reason is that choosing a good pivot relies on picking the 2nd or 3rd quarter of a data set (imagine the data set split into quarters. I am referring to the two middle quarters of the data.) Quick sort is effective if a good and/or decent pivot is chosen. The range of a decent pivot is the two quarters of the data set. Also, if the pivot ends up being in the first or last quarter (bad pivot) quicksort will recursively run again, with a 50% chance to choose a good pivot. Giving quicksort generally a very effective runtime.

3. What is the best-case running time of the sorting algorithm? Explain.

The best-case running time of the sorting algorithm would be O(n log(n)). This is if the pivot chosen every time was a perfect median of the data set, and its partitions. This would give the minimal amount of recursive calls and the fastest run time.

4. Is the sorting algorithm stable? If not, why?

The sorting algorithm is not stable. For it to be stable would mean that quicksort would not swap equal keys, but it does. When a key is equal to the pivot, that key is moved towards the right of the pivot (or left, depends on the code).

==========

Mergesort

==========

1. What is the worst-case running time of the sorting algorithm? Explain.

The worst-case running time of the sorting algorithm is O(n log(n)). An example would be running the sort on an array, {0,2,4,6,1,3,5,7}.

This would split the array into two sub arrays, {0,2,4,6} and {1,3,5,7}. This is the MAXIMUM amount of comparisons and swaps you would have to do.

2. What is the average-case running time of the sorting algorithm? Explain.

The average-case running time of the sorting algorithm is O(n log(n)). mergesort is generally very fast and will split the data set into sets that are of size (n/2), recursively.

3. What is the best-case running time of the sorting algorithm? Explain.

The best-case running time of the sorting algorithm is O(n log(n)). This case would be if the data set is already sorted. Allowing for no comparisons.

{1,2,3,4,5,6,7,8,9}
would result in not having to be sorted.

4. Is the sorting algorithm stable? If not, why?

Mergesort is a stable algorithm. When a data set is sorted already, the same data set will be in place, even the existing duplicate elements.