3/4/18
Evan Pomponio
Description: HW3 an analysis of different variants of the QuickSort Algorithm.

Quicksort:

1. The Quicksort Algorithm performed much better on random arrays than almost sorted arrays with respect to the number of comparisons and swaps.
2. The Quicksort Algorithm had its worse case with respect to the number of comparisons and swaps on almost sorted arrays.
3. This is consistent with the conclusion found on https://www.toptal.com/developers/sorting-algorithms/nearly-sorted-initial-order

Randomized Quicksort:

1. The Randomized Quicksort uses a random index for the pivot.
2. The Randomized Quicksort Algorithm performed much better on random arrays than almost sorted arrays with respect to the number of comparisons and swaps.
3. The Randomized Quicksort Algorithm reached its worse case with respect to the number of comparisons and swaps on almost sorted arrays.

Insertion QuickSort:

1. The implementation of Insertion Quicksort involves transitioning to a non-recursive sorting algorithm that performs fewer swaps, comparisons or other operations on small arrays.
2. Contrary to the traditional and randomized quicksort algorithms, the Insertion Quicksort Algorithm performed much better on almost sorted arrays than randomly generated arrays.
3. The Insertion Quicksort Algorithm reached its worse case with respect to the number of comparisons and swaps on random arrays.

Median of 3 Quicksort:

1. The Median of 3 Quicksort algorithm chooses the middle index of the partition or the median of the first, middle and last element of the partition for the pivot .
2. The Median of 3 Quicksort Algorithm performed much better on random arrays than almost sorted arrays with respect to the number of comparisons and swaps.
3. The Median of 3 Quicksort Algorithm reached its worse case with respect to the number of comparisons and swaps on almost sorted arrays.

Conclusion

Based off this experiment, I do not think that the variations of the Quicksort algorithm should be evaluated by number of comparisons and number of swaps alone. In choosing this approach, the results do not reflect actual performance due to the fact that some variations

technically use more comparisons and swaps, such as the Median of 3 variant. However these comparisons and swaps are made in an effort to optimize the choice of the pivot value. The effect of these variants in most cases improve the balance of partitioning at the expense of increase in the expected number of swaps.

It is my opinion that the final results found in this experiment are misleading. Though the analysis show that almost sorted input increases the amount of swaps and comparisons, the variants of the Quicksort algorithm implemented in this instance should not be compared to each other as a means of evaluating performance. A better approach would be to supplement this data with timing statistics and also measuring the balance of the partitions during runtime in order to draw a more accurate conclusion about the performance of the various Quicksort Algorithms implemented in this experiment.