1. View the following Data Set here. The column on the left is the original input of strings to be sorted or shuffled; the column on the extreme right are the string in sorted order; the other columns are the contents at some intermediate step during one of the 8 algorithms listed below. Match up each algorithm under the corresponding column. Use each algorithm exactly once: (1) Knuth shuffle (2) Selection sort (3) Insertion sort (4) Mergesort(top-down) (5) Mergesort (bottom-up) (6) Quicksort (standard, no shuffle) (7) Quicksort (3-way, no shuffle) (8) Heapsort.

2. Implement Quicksort using median-of-three to determine the partition element. Compare the performance of Quicksort with the Mergesort implementation and dataset from Q1 (HW2). Is there any noticeable difference when you use N=7 as the cut-off to insertion sort. Experiment if there is any value of "cut-off to insertion" at which the performance inverts.

3. Problem 9.29 From Sedgewick, Algorithms in C++, 3rd Edition: Empirically determine the percentage of time heapsort spends in the construction phase for N=103, 104, 105 and 106.

4. Implement the rank() and select() ordered operations for a BST. Use data set linked below. (i) What is the value of select(7) for the data set? (ii) What is the value of rank(7) for the data set? Data Set for Q4 is: https://sakai.rutgers.edu/access/content/group/9a721e60-ef8e-412a-835b-14c0ab9020f0/HW-Dataset/hw3-q4-select-data.txt

1)

1) Bottom up Merge Sort

2) Quick Sort

3)

4) Top down Merge Sort