Wendi Tan
wtan37@ucsc.edu

CSE13s Winter 2021
Assignment 5: Sorting

Pre-lab Part1:

1. 10 rounds.
2. The worst case would be an array in a reverse order, for example {5, 4, 3, 2, 1}. And the comparison would be n*(n-1)/2, where n is the length of the array.
3. Changing the swap condition: if(array[i] < array[i+1]) {swap array[i] and array[i+1]}.

Pre-lab Part2:

1. The best time complexity is O(n*log(n)), and the worst case is O(n*(log n)^2). Since the sequence of gap can determine the round of comparison in shell sort, we need to choose a more efficient way.

Pre-lab Part3:

1. Quick sort is implemented by picking a pivot in the array, and putting smaller numbers on its left, larger numbers on its right, then recursively sort two sub-groups with the same steps. The worst case emerges when the pivot is the largest or the least number in the unsorted array for every round. For example, if we choose the leftmost pivot in a reverse-ordered array (or the rightmost pivot in a sorted array), we need to iterate n times for each sort times n elements need to be sorted, then get the O(n^2). However, we now do this by choosing the middle element in the array and sub-groups as pivots, and this would decrease the chance of encountering the worst case.

Pre-lab Part4:

1. I planned to create a file for public use, containing functions swap, counting moves and comparison. These functions can be called by all source files for sorting methods.

Design

This program can implement four sorting methods: bubble sort, shell sort, quick sort, and heap sort.