ECE 368 Report

Shell Sort				Bubble Sort			
File Size	Comparisons	Moves	Run Time (s)	File Size	Comparisons	Moves	Run Time (s)
1000.txt	4313	66221	0	1000.txt	20707	13197	0
10000.txt	64819	1166240	0	10000.txt	296729	186408	0
100000.txt	878714	18089535	0.05	100000.txt	3766747	2438898	0.02
1000000.txt	11710261	259684562	0.74	1000000.txt	46666768	30144183	0.3

In the shell sort algorithm, I implemented the basic shell sort algorithm, but I generated the three smooth numbers based on the array size to sort no adjacent elements in the array. Before the function does any sorting or generates the 3 smooth numbers I have a loop to check to see if the array is already sorted. If the array is already sorted it will exit the function.

In the improved bubble sort algorithm, I improved the run time by comparing no adjacent elements in the array. In order to determine the gap size, I generated the numbers according to seq2 format. After that I had two pointers one at the beginning of the array and the other pointer started at the first gap size. My function basically compares those non adjacent elements and increments both pointers. Additionally, I check to see if the array is already sorted before any sorting or generating seq2. If the array is already sorted, I exit the array.

The time complexity of the seq1 is O(n) since there is only one loop and it iterates based on the size of the input array. The space complexity is O(n) as well because the array that contains seq1 has to resize itself every time in the loop until it reaches the ending condition in the while loop.

The time complexity of the seq1 is O(n) since there is only one loop and it iterates based on the size of the input array. The space complexity is O(n) as well because the array that contains seq1 has to resize itself every time in the loop until it reaches the ending condition in the while loop.

The average and worst case time complexity of the shell sort is $O(n^2)$ since there are two loops where one of them is nested inside the other loop. However, the best case time complexity will be O(n) since my shell sort initially checks to see if the array is already sorted, or if the size of the input array is just two elements. Since shell sort does in place sortation's therefore the space complexity is just O(1).

The average and worst case time complexity of the improved bubble sort is $O(n^2)$ since there are two loops where one of them is nested inside the other loop. However, the best case time complexity will be O(n) since my improved bubble sort initially checks to see if the array is already sorted, or if the size of the input array is just two elements. Since shell sort does in place sortation's therefore the space complexity is just O(1).