Project 1 Report

Guanshi He

Description of algorithms

Shell sort by using Insertion sort

First get the sequence of the gap for shell sort by getting the biggest gap first and assign the value for the gap stack from the first one(1). The second smallest gap value will be 1.3 times of the former one. Whenever the value is 9 or 10, assign it with the new value 11. After we get the sequence of the gap, use the gap for the array to implement the shell sort. For shell sort, basically it is to arrange the data sequence in a two-dimensional array and sort by separate the data array using the gap and then sorting the column of the separated array by using Insertion sort.

Shell sort by using bubble sort

First get the sequence of the gap for shell sort by getting the biggest gap first and assign the value for the gap stack from the first one(1). To get the next gap value, we need to do the exponential of 3 first because it will give us the right order. After we get the sequence of the gap, use the gap for the array to implement the shell sort. For shell sort, basically it is to arrange the data sequence in a two-dimensional array and sort by separate the data array using the gap and then sorting the column of the separated array by using Bubble sort.

Analysis of time complexity and space complexity

To generate the first sequence for insertion sort
The time complexity of getting the first sequence is O(n^2) because I used a two-level for loop in the self-defined function for generating the first sequence.
The space complexity of getting the first sequence is O(n) because I assign the value directly to the array of sequence each time.

To generate the second sequence for the bubble sort
The time complexity of getting this sequence is O(n) since only one loop is used.
The space complexity is also O(n) because I go through the entire stack for assigning each gap value.

Data for using two ways of sorting algorithm

By using Insertion Sort

Array Size	Run time	Number of Comparisons	Number of Moves
10	0.000000	31	12
1000	0.000000	92859	89881
10000	0.020000	8644017	8614046
100000	1.830000	841858849	841858885
1000000	198.370000	83593674516	83590674568

By using Shell sort

Array Size	Run time	Number of Comparisons	Number of Moves
10	0.000000	210	54
1000	0.000000	4253529	403965
10000	0.61	429248770	39398976
100000	66.350000	42974654662	3915376104
1000000	**N/A**	**N/A**	**N/A**

The bubble sort used much larger number of comparisons and moves than the insertion sort as the amount of the data increased and it consumed much more time than the insertion sort.
The number of comparisons and moves increased faster using bubble sort than using insertion sort. Further more, the time consumption of bubble sort is exponentially increasing but the time consumption.

Summary of the space complexity of sorting routines

For insertion sort, it will be O(1) auxiliary and O(n) in total.

For bubble sort, it is O(1) auxiliary and O(n) in total.