Abhimanyu Agarwal
ECE-36800 – Data Structures in C
Project 1

Project-1 (Milestone-1)

Objective: The objective is to understand and implement two different types of sorting algorithms that undergo a similar optimization based on two gap sequences and based on the obtained results, possibly analyze the two algorithms performance.

Shell sort: The Shell sort is basically an optimization to Insertion Sort. As discussed in lecture, Shell sort takes an array of n inputs and sorts it by breaking it down to logical sub-arrays. These sub-arrays are sorted based on the 'gap' that in general scenarios is empirically derived. However, with respect to the project the gap here is essentially determined based on an array of 'gap' sequences (Sequence 1).

Sequence 1 = $\{2^{p'}3^{q'}, \ldots, 2^{p}3^{q}, \ldots, 16, 12, 9, 8, 6, 4, 3, 2, 1\}$

The shell sort would take the very last value of this array and perform insertion sort on each sub-array. The same process would be repeated for different and smaller 'gap' values until the gap value is one. Assume our 'gap' value is X, then our logical sub-arrays would look like:

{0, N, 2N, 3N, ... , XN}where, XN < length(array)

{1, N+1, 2N+1, 3N+1, ... , XN+1}where, XN + 1 < length(array)

Once the gap value is 1, the shell sort is basically performing the insertion sort. By that point, the array should be sorted.

Bubble Sort: The Bubble Sort expects a similar optimization as the Shell Sort. The difference being that Sequence 2 would be used in order to determine the gap values. Notable aspect being, that each sub-array is sorted using bubble sort which essentially exchanges elements. In Sequence 2, it is observed that gap value in the array is the floor value of the previous integer gap followed by a division by 1.3 until the value is 1. In general, the sequence is more like a geometric progression with common ratio: 1.3.

Sequence 2 = $\{N_1=N/1.3, N_2=N_1/1.3, N_3=N_2/1.3, \ldots, 1\}$

Using the 'gap' values from the very last value (until gap value is 1) we generate sub-arrays which are sorted using Bubble sort based on the gap values. At any point when there are no comparisons made, it could be assumed that the sub-array is sorted.

Analysis: On implementation of both the sorting algorithms, we wish to obtain the number of comparisons, moves and runtime and analyze the algorithms performance and express our opinion on it.