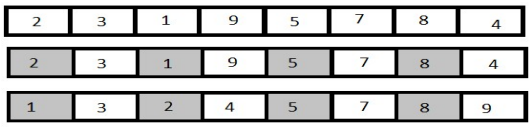
ECE 36800: Data Structures and Algorithms
Project 1 Report
Kunwar Digraj Singh Jain

Project Goal

The goal of this Project is to show the improvement in the performance of **Insertion Sort** when implemented with a gap sequence, which is commonly known as **Shell Sort**, and apply a similar optimization to **Bubble Sort**, to make an improved bubble sort.

Shell Sort with Insertion Sort

To implement the Shell sort with Insertion sort, the Pratt Sequence (2^p3^q) was taken as the gap sequence. The input Array is divided into sub arrays based on the gap sequence, similar to what has been shown below. If the 'gap' is equal to 2, the array is separated in to two different sub-arrays, the white one and the gray one. Subsequently, each sub-array is sorted using insertion sort separately. The end result is a sorted sub-array. Next, the gap value is decreased and the same process is continued. When gap is equal to 1, shell sort becomes a normal insertion sort. The first array is the unsorted array, the second one highlights the sub-arrays and the third array shows the sorted sub-arrays.



Improved Bubble sort

In the improved bubble sort, a similar approach is used. A gap sequence is used to partition the given array into sub-arrays and then the sub-arrays are sorted using bubble sort. One optimization that has been applied is that in the second loop, which creates the sub-arrays, instead of running through the whole array i.e. instead of going from (n-2 down to 0), goes from (0 to gap) and stops when the loop variable equals the gap value since the third nested loop takes care of the rest of the array. This reduces the number of iterations in the second 'for loop' which reduces run-time.

Time and Space complexity of the sequences

No. Elements	Sequence 1 – Shell sort with Insertion		Sequence 2 – Improved Bubble Sort	
	Time	Space	Time	Space
1000	O(n)	O(n), 640 bytes	O(n)	O(n), 320 bytes
10000	O(n)	O(n), 640 bytes	O(n)	O(n), 320 bytes
100000	O(n)	O(n), 1280 bytes	O(n)	O(n), 640 bytes
1000000	O(n)	O(n), 1280 bytes	O(n)	O(n), 640 bytes

Moves, Comparisons and run-time

No. Elements	Sequence 1 – Shell sort with Insertion			Sequence 2 – Improved Bubble Sort		
	Run-time (s)	Moves	Comparison	Run-time (s)	Moves	Comparison
1000	0.000	66221	35266	0.000	12561	23596
10000	0.000	1166240	615529	0.000	184938	329398
100000	0.030	18089535	9484124	0.010	2446419	4197990
1000000	0.430	259684562	135697411	0.160	30249219	51036159

Note: All run-times are computed on the shay.ecn server

Extra memory usage in sub-routines

The sub-routines allocate additional memory for the gap sequences which is equal to **sizeof(long) * (number of gaps)**. The number of gaps are determined by calculating the number of new line characters ('\n') in the file and adding 1 to it since the last gap does have a new line character after it. A temporary variable 'temp' is used in both the sub-routines. The temporary variable facilitates the exchange of the values.