**Rishabh Singh**

**CSC382**

**Prof. Kent Slack**

**11/13/2017**

**University of Advancing Technology**

* **Introduction** – In your own words, describe the operation of each of your sorting algorithm.
* The two algorithms I chose for my assignment were Bubble Sort and Selection Sort. The very first one is Bubble sort which compares each pair of adjacent elements from the beginning of a certain array and sorts them if they are in a reversed order. The other is Selection sort algorithm which has multiple array slots and the smallest element is selected from each of them and swaps with the leftmost element, allowing that element to be the part of the sorted array. This continues until the unsorted arrays are moved by one element to the right-hand side.
* **Programmer’s Guide –**Provide detailed documentation describing your implementation of sorts.  For each function, give a usage example.  Also give documentation on the general usage of your program.
* In my header files I created the sorting functions and a separate .cpp file for running all the main functions by including the header file in there. My header file contains the algorithms, starting off with Selection sort I created a template class S (short for selection sort) with a void function below it where it repeatedly finds the minimum element from the unsorted parts and places it together in the beginning. The Subarray stays sorted and the Remaining subarray stays unsorted. The minimum element is picked and shifted to the sorted subarray from the unsorted subarray.
* This is one of the easiest algorithms to build, that repeatedly swaps the adjacent elements if they are not in the right order. It compares first two elements then swaps if its greater than the other number. Keeps repeating that with the other numbers that are under its value and since some elements are already in order it doesn’t swaps them at all. This process keeps repeating until the algorithm knows they’re all sorted.
* **Analysis –**Determine the worst-case time complexity of the algorithm you chose, and document your calculations in this section.  Additionally, include your data table and graphs.  In this section, answer the following questions:
  + Does your calculated run-time complexity match the measured run-time complexity?  Why or why not?
  + The average and worst-case complexity of the bubble sort is O(n^2) which also swaps it in the worst-case. All the sorted arrays give a O(n) estimation.
  + When the unsorted part becomes empty selection sort stops performing. Selection sort makes n steps where the n stands for the elements in array of an outer loop before it stops and requires every step of the outer loop to find out minimum in unsorted parts. However, the algorithm complexity is also related to the Bubble Sort algorithm (O (n^ 2)).
  + Under which dataset sizes does each algorithm perform best?  Why or why not?  Does any algorithm perform best under all cases?
  + All these algorithms are simple to perform under all cases. Since they run on the save case time complexity, the algorithms’ best case occurs when an array is already sorted.
* <http://www.algolist.net/Algorithms/Sorting/Bubble_sort>
* <http://www.algolist.net/Algorithms/Sorting/Selection_sort>