Assignment 6

Arshia Sharma
Email: arsharma@chapman.edu

December 13, 2019

1 Bubble Sort

Bubble Sort has a run time of $O(n^2)$, which implies that this algorithm takes a long time to sort every element in a list. Bubble Sort works by each element (i), and compares the next element (i+1), to check which one is smaller. Since it uses a double for loop to check each element with its following, the algorithm checks every element in the list and switch them. It also takes about 2-3 seconds for this method to run.

2 Selection Sort

Selection Sort has a Big O run time of $O(n^2)$, the same as Bubble Sort. This also shows that there is a double nested loop used in this sorting algorithm. Selection sort works by index i looking at each element in the list and switches i with the minimum value. This continues with every element in the list until each element is sorted. This sort takes about 3-5 seconds to run.

3 Insertion Sort

As the other two sorting methods, Insertion Sort also has a run time of $O(n^2)$ because of a double nested loop. This method runs by having each sorted element to the left of the array sorted since it compares index i to every element on the left to see if i is less than any element and puts the element in the correct place. This method takes about 1 second to run.

4 Quick Sort

Unlike the other three algorithms, Quick Sort runs with a Big O of $O(n \log n)$. Quick Sort runs by using divide-and-conquer process, which makes sense due to its run time, and also uses recursion. It runs by having a pivot, which in my algorithm is the last index in the list. The pivot is used by having the program figure out where the pivot should be and since this work recursively, the array is split into smaller arrays to sort the list appropriately. This takes 0.01 seconds to run as well, which is much shorted than the others.

5 Merge Sort

Same as Quick Sort, Merge Sort also has a Big O run time of $O(n \log n)$. Merge Sorts works in a similar way, it is a divide-and-conquer process and is defined recursively. However, this is different than Quick Sort since the array is cut in half and once each element is separated, they are then sorted accordingly. One may say that the arrays are merged together in the right order. This also runs in about 0.01 seconds.

6 Conclusions or Closing Statements

In conclusion, although each algorithm sorts the list from smallest-greatest, the 5 algorithm we discussed all have a different run time. The differences, although just in seconds, were still very different even though they all do the same thing. When running an algorithm is $O(n^2)$, it takes a lot longer to run than $O(n \log n)$, which is expected since that is what we learned in class. The trade off to have a algorithm of $O(n \log n)$, it would be more complex to run, having to use recursion and breaking up the arrays. These programs were written in C++ and was good to work with due to the use of pointers, which other languages don't use. Some shortcomings that came with using this empirical analysis to compare the run times is that depending on the size of the list we are using, the times will change. If it's larger than what we currently have, it would take longer to run and if its smaller, it will take a shorter time to run.