Implementing Insertion Sort
Lab #2

By
Joshua Matthews

CS 303 Algorithms and Data Structures
September 7, 2015

1. Problem Specification

The goal of this Lab assignment was to implement the Insertion Sort and test the difference in time it takes to sort the input from various sized files passed in to the program.

2. Program Design

This program required one function besides the main function to implement the insertion sort. It had to accept and extract values from a file, test the insertion sort, and display the amount of time it took to complete the insertion sort.

The following steps were needed to develop this program:

A. I needed to setup the environment with the correct c++ libraries included and main function to accept arguments that would be passed into the program.

B. Next I setup the code to take the file passed in and extract the values from the file and push them on to a vector to be used for later.

C. I then created a function which implemented the insertion sort which has a vector passed in as reference so the function can directly mutate the original object without creating a copy of it.

D. I then setup code around where I am invoking the insertion sort function on the data from the file to calculate how long the insertion sort took to complete.

The included libraries were: iostream(for printing out to the screen), fstream(for reading in data from a file), vector(for having a dynamically sizing array), ctime(for calculating program running time in seconds).

3. Testing Plan

The Lab02.cpp file tests the Insertion sort on the values from whatever file is passed in. The program accepts a file and parsers out each number in the file and pushes them back onto the vector object the is created to hold all of the values. The initial values we retrieve from the files are random numbers between 1 and the number of elements that are in the file inclusively. Once the vector is filled with every value from the file the current time is capture in a variable then the insertion sort is invoked with the vector passed as the parameter. The vector is passed in by reference so there is no time being wasted on copying the value over to the function and allows the function to have direct manipulation over the object. Once the insertion sort is done we capture the current time minus the start time to calculate how long the function took in seconds, then display it to the screen for the user to see.

4. Test Cases

The test cases are shown below:

Test Case Number	Input File	Output
1	input100.txt	Start Insertion Sort! Insertion Sort Done! Insertion Sort time to complete: 8.1e-0.5s
2	input_1000.txt	Start Insertion Sort! Insertion Sort Done! Insertion Sort time to complete: 0.003812s
3	input_10000.txt	Start Insertion Sort! Insertion Sort Done! Insertion Sort time to complete: 0.208282s
4	input_100000.txt	Start Insertion Sort! Insertion Sort Done! Insertion Sort time to complete: 17.9711s
5	input_1000000.txt	Start Insertion Sort! Insertion Sort Done! Insertion Sort time to complete: 1835.47s
6	input_5000.txt	Start Insertion Sort! Insertion Sort Done! Insertion Sort time to complete: 0.048379s
7	input_50000.txt	Start Insertion Sort! Insertion Sort Done! Insertion Sort time to complete: 4.57817s
8	input_500000.txt	Start Insertion Sort! Insertion Sort Done! Insertion Sort time to complete: 17.7035s
9	input_5000000.txt	/* This one actually never finished. I left it running over night and it was still not done in the morning so I stopped the program because it had been running for 6+ hours at that point. */

5. Analysis and Conclusion

The results show that the Insertion sort is okay for relatively small numbers of arrays or arrays that are not completely randomized. In the worst case scenario, the sort could be done on a reversed sorted array. This would cause the sort to sort the array back to it's un-reversed self which will be O(n^2) time complexity. Best case is an already sorted array which will be O(n) time complexity.

6. References

The www.cplusplus.com has references and documentation on all of the standard libraries that were used in this program. Input files were supplied by Abhishek Anand.