Implementing Quick Sort
Lab #3

By
Joshua Matthews

CS 303 Algorithms and Data Structures
September 29, 2015

1. Problem Specification

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

2. Program Design

This program required two functions besides the main function to implement the Quick sort. It had to accept and extract values from a file, test the Quick sort, and display the amount of time it took to complete the Quick 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 Quick sort which has one vector passed in as reference and a start and end integer position. This function handled calling the partition function to organize the array by a pivot index and then split the array in two by that pivot index. It also had two recursive calls to itself which organized the lower half of the array and the upper half of the array.
D. I then create the function for partition which took in the array by reference and took in a starting and end point integer values. The function then assigns the pivot value to a local variable and stores the starting index number. A for loop is created and iterates through one plus the starting value the one less than the ending value, and if the current indexed value is less than or equal to the pivot value you then take the current value and exchange it with starting value plus however many times this if statement has been called. Outside of the for loop we exchange the value at whatever the last exchanged value was with the pivot value and then return the last exchanged value index.
E. I then setup code around where I am invoking the Quick sort function on the data from the file to calculate how long the Quick sort took to complete.

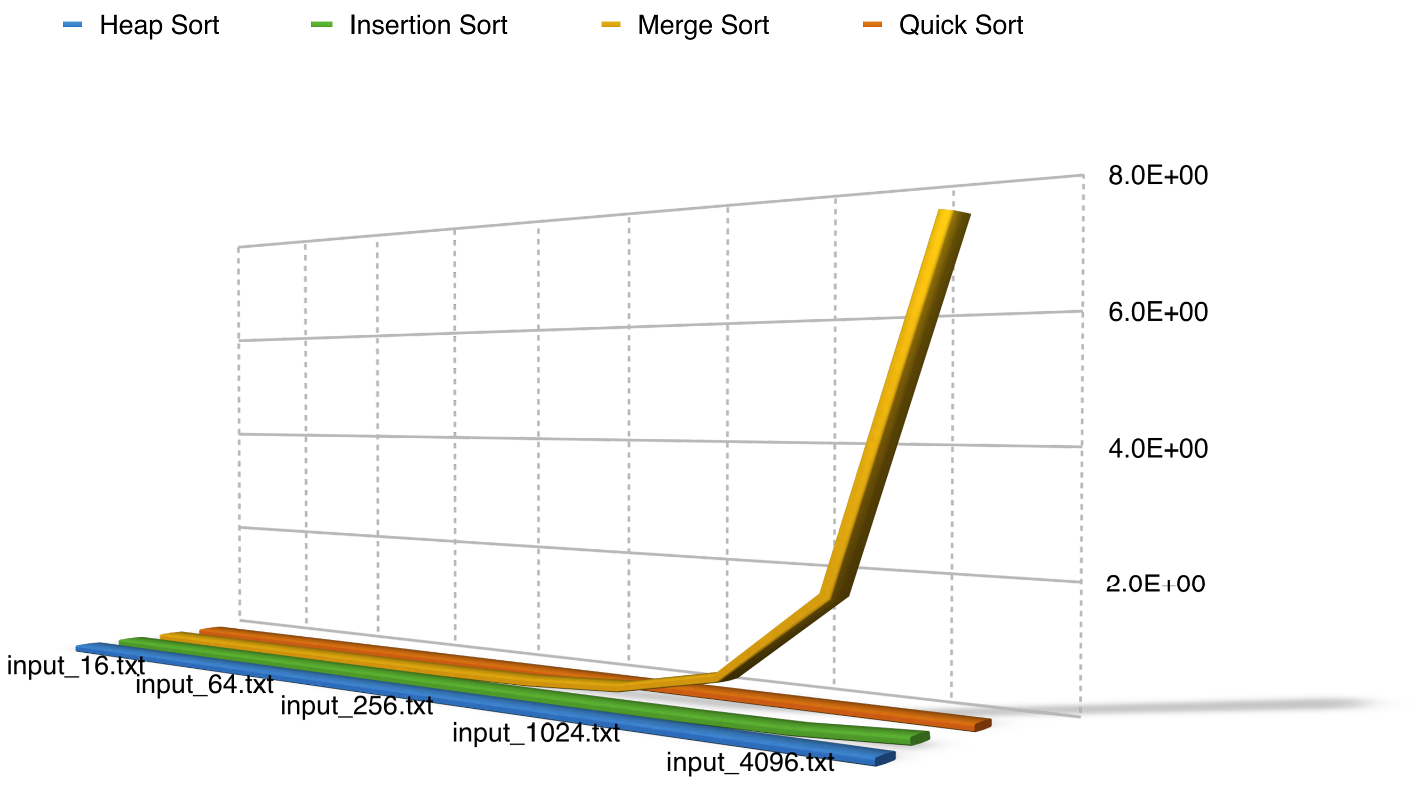
The included libraries were: iostream(for printing out to the screen), fstream(for reading in data from a file), vector(for having a dynamically sizing array), ctime(for calculating program running time in seconds).

3. Testing Plan

The Lab05.cpp file tests the Quick 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 that was 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 captured in a variable then the Quick sort is invoked. The vector is passed in by reference so there is no time being wasted on copying the values over to the function and allows the function to have direct manipulation over the object. Once the Quick 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. The graph shows how long the sorts took from input_16.txt to input_8192.txt.



5. Analysis and Conclusion

The results show that the Quick sort has a time complexity of O(nlogn) in both best case and Average case scenarios, but had O(n^2) in worst case scenario. The Quick sort seems to be faster than most of the sorting algorithms that we have learned about so far. However, though the Quick sort is fast, if you were to get unlucky and end up with it's worst time complexity your program could really suffer. This algorithm is good for programs with little memory to spare with the space complexity of O(logn). I was unable to complete the median of 3 version of quick sort, for some reason I would always get it to where a couple of values were out of place and I could never get it perfect.

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.