Willia Hardy
October 1, 2018
Data Structures Project
Student ID: 2905326

Project 1 Report

File List

- Component.cpp
    - Main project code. Includes function definitions and implementation
- Graph.dat
    - File that is being read from, sorted, and data that is merged
- CMakeLists.txt
    - File used to compile the project
    - I used the "cmake .." command and make in order to create the executable

Summary of Project

This project was intended to use vectors, iterators, and lists to produce output that merged data from a file. The files contents contained information that related to a graph data structure.

The implementation is done by first displaying the data and then sorting the lists as they are merged. If a list had at least one common element with another, the lists could be merged. Once a list was merged into another list, that list was then deleted. The final result is one list that contained each element in the data file.

Answers to Questions

1. Per number 4 in the Coding Requirements, give the worst-case runtime to determine where a node should be inserted into an adjacency list and explain/justify your analysis.
    a. The worst-case runtime is $O(n)$ because once the location is determined, an element can simply be inserted after looping using the position of the value whether or not the value is found
2. Per number 4 in the Coding Requirements, give the worst-case runtime to build the entire adjacency list and explain/justify your analysis of this runtime.
    a. The worst-case runtime to build the entire adjacency list is $O(n^3)$. This is the case because we have one loop to loop at each line in the graph.dat file which is linear. We then have another for loop to loop over each element in each line of the file, again, linear. Then we have one last search (as mentioned in the above question, linear) to insert the newly merged list elements into the merged list.
3. What if vector of vector is used for the adjacency list instead of a vector of lists, how would the runtimes for parts a and b change? Explain/justify your analysis.
    a. The runtimes will change in Question 2, not in Question 1. The insert runtime would not change if it is only inserted into a vector of vectors. However, if it becomes a vector of a vector, the data structure will need to find the position to insert first (the initial loop),

then it will need to insert the element (another extra loop). This brings the runtime to $O(n^4)$ instead of $O(n^3)$ to account for the extra loop.