Alycia Riese

Project 1 Report

List of Files

- components.cpp
    - Main code file for the project. All code is contained in this single file
- graph.dat
    - Data file provided.
- a.exe
    - Windows executable
- a.out
    - Linux output file (executable)
- .git
    - Folder that contains git information

Note: I used the command "`g++ components.cpp`" to compile the code and then used "`./a.out`" to run the resulting output file. This was done in a Linux terminal.

Summary

This project creates a vector of lists from a file that the user has given, and asks the user to merge 2 of the lists together. The file contains lines of integers, and the lists are formed using these integers. If the 2 lists provided by the user share a common integer, then they are able to be merged. Then, the code determines which list is smaller by comparing the sizes of the lists. The smaller list is then merged into the larger list, and the smaller list is cleared then deleted from the main vector of lists.

Questions

a. Give the worst case runtime to determine where a node should be inserted into an adjacency list and explain/justify your analysis.

- The worst case runtime to determine where a node should be inserted is O(n) because the code iterates through the list 1 time, and finds where the node should go.

b. Give the worst case runtime to build the entire adjacency list and explain/justify your analysis of this runtime.

- The worst case runtime for the entire adjacency list would be O(n^3) because there is 1 while loop looping through the lines in the file, 1 loop for reading the characters in each line, and 1 loop for iterating through the list to determine where the node goes

c. 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.

- The runtime for part a would remain O(n) because all we are doing in that function is finding where it goes, not actually inserting, so the runtime wouldn't change.
- The runtime for part b would increase to O(n^4) because 1 loop for getting the lines in the file, 1 loop for getting the characters from the lines, 1 loop for iterating through each element to find where it goes, and 1 more n because inserting into a vector is linear.