COS2021 - Homework 5
Closeness Centrality

Due: Dec 12th Before midnight

Version: 00

Task Overview and Learning Goals

In this assignment, you will practice evaluating graphs programmatically. You will get experience with several graph algorithms, representations and concepts. Graphs are very important to computer science. A strong understanding of graphs and how they can be used to represent a myriad of problems will serve you well in any CS-related job.

Submission and Requirements

Submit your C++ solution (all files) in a single zip file. Submit **all** files related to the assignemnt (except for this pdf). This means submit .hpp, .cpp and the graph .txt files.

Your code should contain a header comment with your name and any issues or bugs I should know about. Remember: most programs contain some type of bug and it is a sign of a mature coder to acknowledge issues (not hide them).

Details

Your program should have the following functionality:

- Your program should ask the user for the name of a graph text file on the command line. An example of the prompt is given below. If the file does not exist, the program should inform the user and either quit gracefully or allow the user to enter the name a second time.
- If the graph file exists, it should load the graph data into your program. You are welcome to use any data structure(s) to store the graph. However, the program must store the graph in memory (as opposed to repeatedly reading the text file).
- Once the graph is loaded, your program should calculate the Closeness Centrality of every node in the graph. This can be done efficiently with Breadth First Search. Pseudocode for BFS is given below. Closeness Centrality for a single vertex x_k is calculated using the following formula:

$$C(x_k) = \sum_{i}^{N-1} \frac{N-1}{d(x_k, x_i)} \quad \text{where } i \neq k \ \forall \ i$$

N is the number of vertices in the graph. $d(x_k, x_i)$ is a function that gives the shortest path distance between two vertices (e.g., via breadth first search). **IMPORTANT**: when calculating the closeness along a path, you should not include the starting vertex in the count. In other words, count the number of edges between two vertices along a shortest path.

- Finally, the program should print out a nicely formatted list of the vertices and the Closeness Centrality. The list should be sorted in **descending** order with respect to centrality.

Graphs

I am providing two graph txt files for you to test your code on. *graphA.txt* is a very simple undirected graph (meaning all edges appear in the edge lists of both endpoint vertices). The second, *graphB.txt*, is a more complex directed graph. Part of the challenge is to use these graphs to verify your code works.

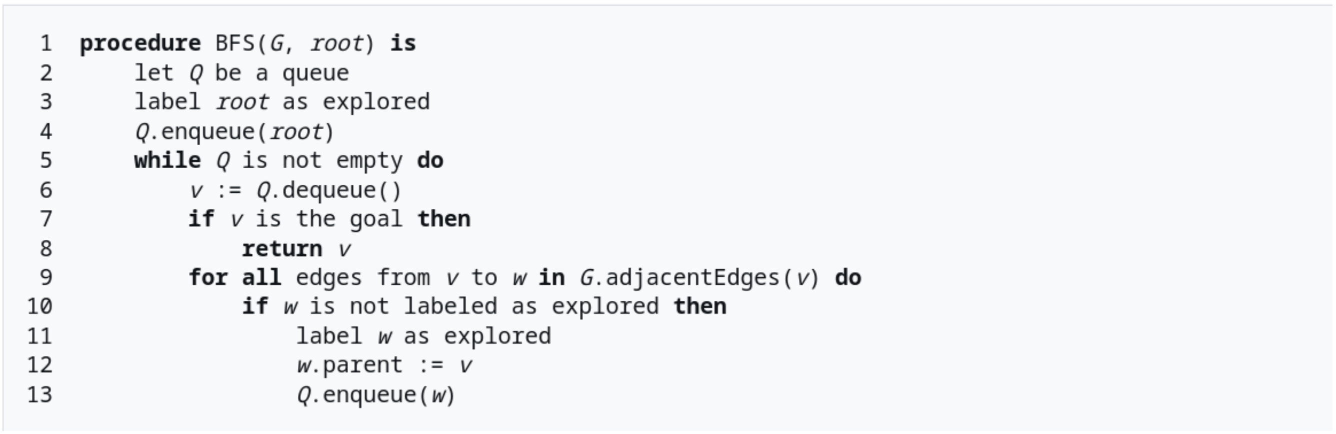
Examples

An example of the prompt for a file:

```
FILENAME:
```

Breadth First Search

Here is pseudocode for Breadth First Search from the Wikipedia page on BFS. The algorithm as written will have to be adapted to your graph implementation and also include a summation of all path distances. Also the algorithm below assumes you are searching for the shortest path to a single node. In fact you are looking for the length of all shortest paths from all nodes.



```
procedure BFS(G, root) is
    let Q be a queue
    label root as explored
    Q.enqueue(root)
    while Q is not empty do
        v := Q.dequeue()
        if v is the goal then
            return v
        for all edges from v to w in G.adjacentEdges(v) do
            if w is not labeled as explored then
                label w as explored
                w.parent := v
                Q.enqueue(w)
```

Figure 1: BFS pseudocode from Wikipedia

Grading

Your program will be graded on how well it solves the above problem. Partial work will count for something. Start early and ask for help. I will grade using both the graph files I provided and a new one you have not seen.

If your program does not work as expected, I advise including a comment to that effect. Being aware of issues and not hiding from them shows maturity.

As I mentioned in class, if your code strongly resembles Generative AI examples you will receive a zero. If your code strongly resembles code of another student and you have not cited them as a co-worker (and they have not cited you), you will receive a zero. Group work is limited to two people. Citations should be done clearly at the top of the main file (or in a Canvas submission comment).