Alan Osborn

Final Project

CS 260 / Jess

June 7, 2022

README

**Final Project:** Graph

**Part 1: Create some test before you start coding…**

**-**I should have some code created to print out elements of the vectors…

-When I wrote this I had pseudo code for it but I most deleted somewhere along the way..

-This is what that became:

Text

Description automatically generated

-When we have a add\_vertex and add\_edge function we should have it displayed similar to an array

-This was provided with the in-class example:

Text

Description automatically generated

-There are other miscellaneous print functions.. and I had a random\_testing.cpp file that was used for initial set up

**Part 2: Implement a graph class with at least:**

I’ll start this part by saying I used the in-class example as a starting point. It was easy to create the nodes and edges from the code you created but when it came to deal with the shortest\_path algorithm I’ve been stuck on it for days now. Here is what I started with, and I’ve typed of random things trying to get through it.

Text, letter

Description automatically generated

I’ve overcomplicated it at this point but I basically stuck at updating a vector I created to hold and update the new shortest distance.. I’ve been at this for 3 days and progress is so slow I decided to just turn in what I have.

For the minimum spanning tree my friend created a C program and walked me through how to do that Kruskal problem. I never got to my own implementation because of struggling through the shortest distance problem. I don’t expect any points for that portion I just included it to show I at least got another run down on the problem.

**Part 3: Analysis the complexity**

The graph created is an adjacency list therefore to add a vertex it’s as simple as adding a graph node to our vector. The edge is also very fast because the two classes are intertwined therefore both are O(1) operations.

I’ll analysis my unfinished shortest path algorithm. First the way I approached it was very space inefficient because I created 4 vectors that tracked information and directly access them as opposed to using pointers. I did create them in the function so at least they aren’t allocated after the function call. I then iterate through every node to find the source node costing O(V) time complexity. I for loop then through the neighbors and in worst case could be another O(V) operation. According to my design it would take a couple more lines but overall, the time complexity would be O(V^2) + some miscellaneous operations.

The last thing about complexity is the Kruskal algorithm is O(E log E). The way it was explained to me by my friend is that it has to go through all edges then verify is each edge is safe. Because some are connected it’s a similar cutting the operation down which is where the log behavior comes from.

**Closing remarks:**

This isn’t my best work. I felt comfortable all term with the assignments and took this project for granted. I feel really defeated because it took many hours to get what I produced, and it wasn’t even working. A retrospect is that I should have implemented my own graph to understand it better. I struggled with using classes and vectors in this course and at least I got a lot of practice with both in this project.

It’s been a pleasure learning so much from you over the last year. Thank you.