COS 364 Floyd Project - FloydProject2020.pdf
Due Date Thursday October 29 by 5 pm

You will be implementing Floyd's algorithm, Algorithm 3.4 on pages 107 to 108 modified by my outline (see the Floyd folder on my google drive) where the path matrix is initialized to -1 and not to 0. This algorithm computes the distance of a shortest path from each vertex in a weighted graph to each of the other vertices. It also creates a shortest path from each vertex to each of the other vertices.

You will create a Project in Visual Studio to implement Floyd. I have given you the skeleton called FloydSkeleton.cs. It is on my google drive in the folder Floyd/Project. Note that in my skeleton the namespace is called GraphFloyd. If you initially use a different name when you create your project, you will have the change the namespace from GraphFloyd to whatever name you use.

You will write the following functions:

The function Floyd: In here you will set up the matrices **D** and the path matrix **P**. D is the matrix discussed in section 3.2, which starts out as the adjacency matrix. P is the path matrix. So set up D and P initially as follows: in here, D gets initialized with the weights of the edges (meaning D is initailly set to be W, the adjacency matrix). The matrix P should get initialized to all -1 values, **not** 0's as they do on page 108 of our text. Next, follow Algorithm 3.4 in your code.

Lastly, do your outputs. (1) Output the final D matrix which will give you the lengths of the shortest paths between each pair of vertices. (2) Output the path matrix P. (3) Output all the shortest paths, utilizing Algorithm 3.5 on page 109 (as modified). Your function will call the path function.

The function path: follows algorithm 3.5 as modified by me. See the Floyd folder on my google drive.

The function printGraph: that I call in the main program. This function should print the number of vertices as well as the adjacency matrix.

The function createGraph: you have to write mostly all of this. Refer to the class example project called InputExampleMatrix.

The Main function: This is totally written by me, meaning you do not add anything. The main reads the command line argument you supply when you run the program. This argument is the filename containing the information for your specific graph. The command line argument is referred to (in your code) by **args[0]**. Using this command line argument, in a try . . . catch block, I created a new StreamReader using this filename as the argument. To get a command line argument into Visual Studio, right click the project and choose properties; you will get a popup with a section entitled **Debug** in the left pane and on the right choose the section **Application arguments**. In Application arguments, type in the name of yoiyur data file, including its extension.

The data file: You must create a data file that contains the vertex and edge information. This file must be in the format described in class so that my code in CreateGraph reads it correctly. Inside the folder for the Floyd project, is a folder called SampleDataFile which contains an example of the structure for your data set. **Make sure to use the graph I drew. This picture is on my google drive in the folder FloydProject.** You must put your data file into the following subfolder of your project: **metcoreapp3.1**; so for example if the project folder is called GraphFloyd, then we have the following structure:

```
GraphFloyd
    GraphFloyd
        bin
            Debug
                netcoreapp3.1
```

What To Turn In:

All is to be emailed or shared (via your google drive) with me.

1. State the complexity of the Floyd algorithm as a function of n, which is the number of vertices and explain in detail why you chose this formula. Focus on the triple nested loop in Floyd.

2.
   i. Write your program to implement Floyd's Algorithm 3.4 on pages 107 to 108 as modified by my outline (where path matrix is initialized to -1 and not to 0). Also in printing the path, you ask if it is equal to -1 instead of to 0.

   ii. Document it appropriately so that someone can understand what you are doing and if in the future they wanted to modify it, they would understand from your documentation enough to easily modify it.

   iii. Turn in the cs file you wrote.

   iv. Turn in your input file that contains the vertex and edge information. Make sure to use the graph I drew.

   v. Turn in your output from your run of the program.

   vi. Write a paper to explain the Floyd algorithm and how you coded it. Also include an explanation of how you designed and coded the function createGraph.