Coding Assignment 3

CSE2320

Name your program `Code3_xxxxxxxxxx.c` where xxxxxxxxxx is your student id (not netid). My file would be named `Code3_1000074079.c`.

Please place the following files in a zip file with a name of `Code3_xxxxxxxxxx.zip` and submit the zip file.

`Code3_xxxxxxxxxx.c`

`Graph.txt`

A zip file is used to avoid Canvas's file renaming convention.

Reminder – ANY compiler warning(s) or error(s) when compiled on Omega will result in an automatic 0. This apply no matter what the warning/error is. You will need to code so that your final program will compile cleanly and run on both the VM and Omega.

For example, if the GTA compiles your code on Omega and sees

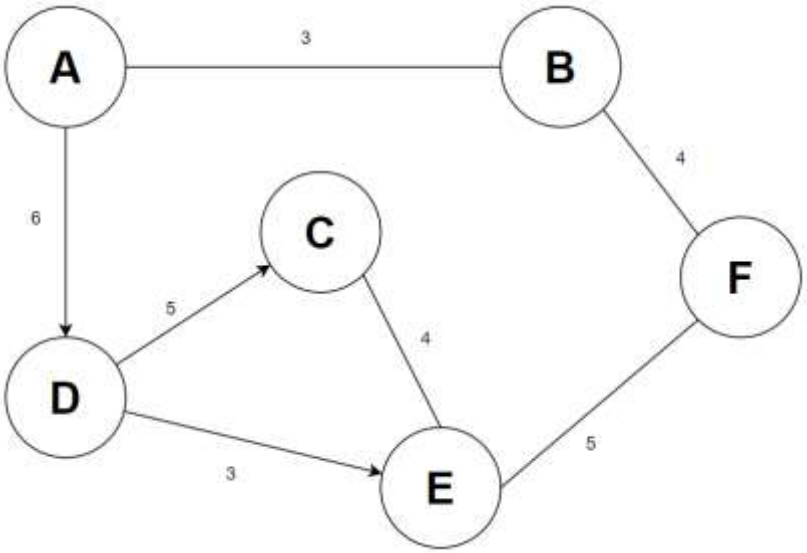
```
[frenchdm@omega CA1]$ gcc Code3_1000074079.c
Code3_1000074079.c:44:2: warning: no newline at end of file
```

you will receive a 0 on the assignment.

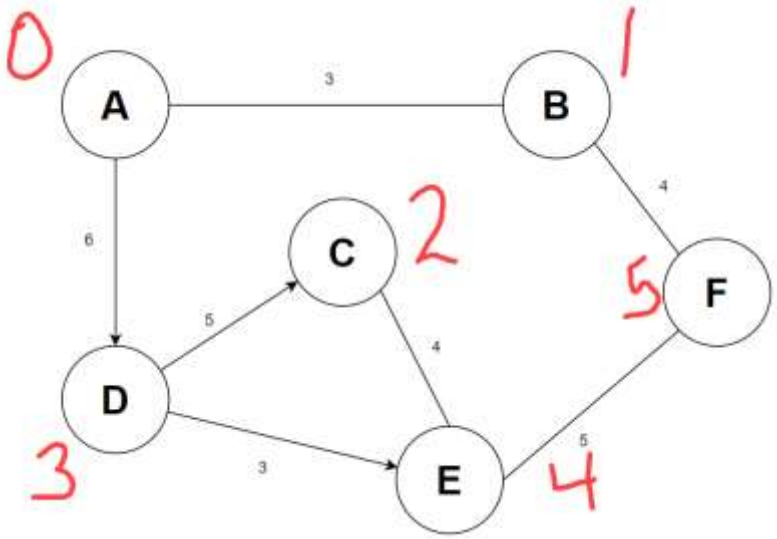
The Program

You are creating a program that can read a file of graph information and run Dijkstra's Algorithm on the graph and produce the path and the length/weight of the path between the starting Vertex and any other Vertex in the graph.

Let's say we start with this graph.



To translate graph to file, we'll first start by numbering the vertices. The number we assign to a vertex will be that vertex's index number in the Vertex Array. Just for convenience, I am going to start with A and number it 0 and go alphabetically from there. This is not a necessary order. You could start with Vertex E and 0 and mark Vertex C as 1, etc. Just pick one to be index 0 and number from there.



Now we can translate this graph to our file format. Our file format assumes that each line in the file represents a vertex and all of its edges and weights. If the file has 5 lines, then the graph has 5 vertices.

The format of a line in the file is

VertexLabel, Adjacent Vertex Index, Weight of Edge

where "Adjacent Vertex Index, Weight of Edge" can be repeated. The first line in our file would be

```
A,1,3,3,6
```

This shows that Vertex A (which is at index 0) has an edge between itself and Vertex B (which is at index 1) and that edge has a weight of 3. Vertex A also has an edge between itself and Vertex D (index 3) and that edge has a weight of 6.

The file for this graph will be

```
A,1,3,3,6
B,0,3,5,4
C,4,4
D,2,5,4,3
E,2,4,5,5
F,1,4,4,5
```

The ordering of the vertex,edge pairs does not matter – `0,3,5,4` is the same as `5,4,0,3`. You need to create a file that represents a graph you have created – it should be different than the graph shown in the assignment. Name the file `Graph.txt` and submit it with your assignment. Your program will be tested with various files/graphs.

IMPORTANT : make NO assumptions about the ordering or the labeling of the vertices in the graph. The VertexLabel could be up to 5 characters and the vertices can be listed in any order in the file.

Getting Started

Create a define that will be set the to maximum size of your graph – the maximum number of vertices that your program will load and process. Create your Vertex Array and your Adjacency Matrix using that define. Initialize the Adjacency Matrix to -1.

See **File Handling** for details on how to populate the Vertex array and the Adjacency Matrix. Prompt for the starting vertex. Your program should be able to use any vertex in the graph as the starting vertex.

File Handling

As in the previous assignments, open the file from the command line using `argv[1]`. While using `fgets()` to read through the file, use `strtok()` to parse each line into its parts. Use the parts to add vertices to the Vertex Array and edges to the Adjacency Matrix.

```
#ifdef PRINTIT
printf("\n");
for(i = 0; i < VertexCount; i++)
{
    for(j = 0; j < VertexCount; j++)
        printf("%5d\t", AdjMatrix[i][j]);
    printf("\n");
}
#endif
```

Conditional Compile

Add a conditional compile statement `PRINTIT` to print out the Adjacency Matrix after you have populated it (using values from the file). So, if your code is compiled as

```
gcc Code3_xxxxxxxxxx.c -D PRINTIT
```

then your adjacency matrix will be printed to the screen.

```
#ifdef PRINTIT
printf("\n\nI\tL\tD\tP\tV\n");
for (i = 0; i < VertexCount; i++)
{
    printf("%d\t%s\t%d\t%d\t%d\n", i,
        VertexArray[i].label, VertexArray[i].distance,
        VertexArray[i].previous, VertexArray[i].visited);
}
printf("\n");
#endif
```

```
   -1     3    -1     6    -1
    3    -1    -1    -1    -1
   -1    -1    -1    -1     4
   -1    -1     5    -1     3
   -1    -1     4    -1    -1
```

The `PRINTIT` conditional compile should also be used to print out your Vertex Array after running Dijkstra.

```
I     L     D     P     V
0     A     0     -1    1
1     B     3     0     1
2     C     11    3     1
3     D     6     0     1
4     E     9     3     1
```

I = **I**ndex, L = **L**abel, D = **D**istance, **P** = Previous, **V** = Visited

This feature will be used to grade your program. The printing should be controlled with the compiler directive (not by commenting or uncommenting code). The output of your adjacency matrix and vertex array **must** have these formats.

Printing and Prompts

The adjacency matrix must be printed AFTER being populated from the file and the vertex array must be printed AFTER running Dijkstra's Algorithm. The print outs must be after prompting for the starting vertex. The prompting for the end vertex and printing of the path may be before or after the print but must be after running Dijkstra's Algorithm on the data. The path MUST be printed in order starting with the start vertex and ending with the end/destination vertex. The path should not print from ending to starting – it must be in order.

Dijkstra's Algorithm

I suggest you use the Dijkstra code shown in class. It closely mirrors the manual method we learned. If you so choose to use the Internet as your source, be sure you are getting the C version of Dijkstra (and not some other graph traversal/variation of Dijkstra). Your Dijkstra code, regardless of where you source it, **MUST** use an adjacency matrix and a vertex array so that you can create the required output for grading.

Program Input/Output

```
student@cse1325:/media/sf_VM2320$ gcc Code3_1000074079.c -g -D PRINTIT
student@cse1325:/media/sf_VM2320$ ./a.out Graph.txt

   -1     3    -1     6    -1    -1
```

```
    3    -1    -1    -1    -1     4
   -1    -1    -1    -1     4    -1
   -1    -1     5    -1     3    -1
   -1    -1     4    -1    -1     5
   -1     4    -1    -1     5    -1

What is the starting vertex? A


I     L     D     P     V
0     A     0     -1    1
1     B     3     0     1
2     C     11    3     1
3     D     6     0     1
4     E     9     3     1
5     F     7     1     1

What is the destination vertex? E
The path from A to E is A->D->E and has a length of 9
```

```
student@cse1325:/media/sf_VM2320$ gcc Code3_1000074079.c -g
student@cse1325:/media/sf_VM2320$ ./a.out Graph.txt

What is the starting vertex? E
What is the destination vertex? A
The path from E to A is E->F->B->A and has a length of 12
```