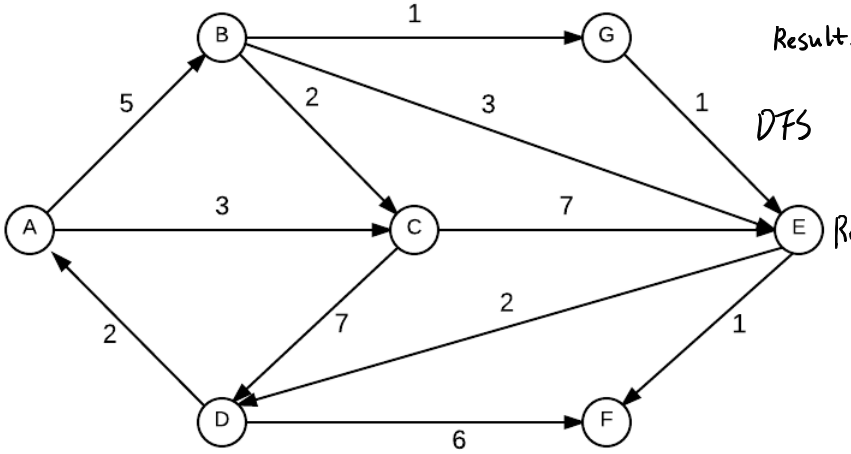
CSE331 Homework 6 (Due by the start of class on April 3)

1. a) Show the in-degree and out-degree for each vertex in the following graph. And, b) using B as the starting vertex, show the BFS tree using the algorithm in our notes. c) Show the DFS tree using B as the source vertex. (a: 5 pts. b: 5 pts. c: 5 pts)



BFS Queue: ~~B G C E D F A~~
Result: B C E G D F A

DFS stack:

Result: B C D A F E G

2. For the above graph, please list the indicated path using the edges in the path. If you choose to draw the paths in the graph, you need to use different colors to differentiate each path. (a. 10 pts, b. 5 pts)
   a) Find the shortest path from *A* to all other vertices.
   b) Find the shortest unweighted path from *B* to all other vertices.

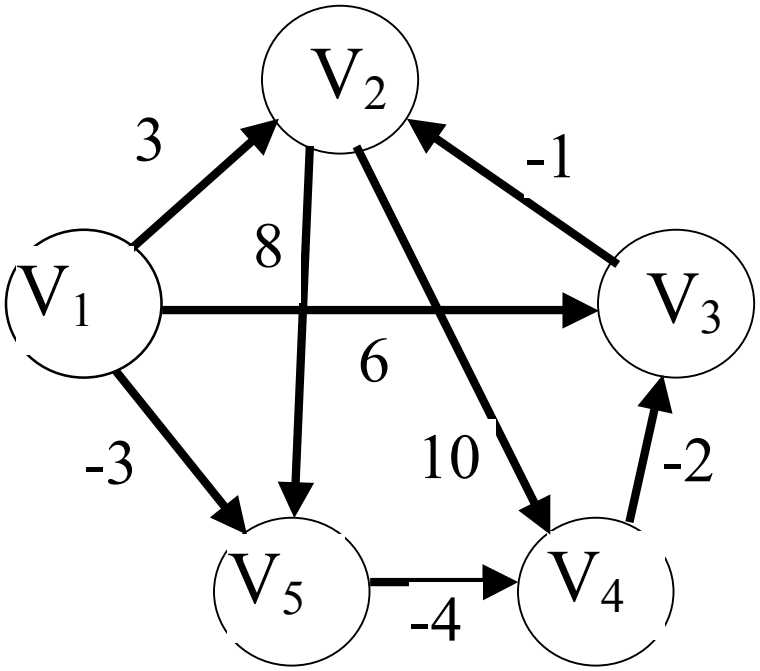
3. Give an example where Dijkstra's algorithm gives the wrong answer in the presence of a negative edge but no negative-cost cycle. Please draw the graph and explicitly show the wrong path. Since you only need to provide one negative example, feel free to draw this path in the figure. (5 pts)

4. Explain how to modify Dijkstra's algorithm to produce a count of the number of different minimum paths from *v* to *w*. (10 pts)

5. How can the output of the Floyd-Warshall algorithm be used to detect the presence of a negative-weight cycle? A negative-cycle is a cycle with total weight being less than zero. For example, A→B→C→A is a cycle. If the total weight of A→B, B→C, and C→A is negative, this is a negative-weight cycle. (5 pts)

6. Using Warshall's algorithm to find all-pairs shortest path in the following graph (on page 2).
   a) Show the matrix D and P for each k. (see the pseudo-code in the lecture notes). For k from 0 to 5, you only need to show matrix cells with changes. (10 pts)

b) Show the final all-pairs shortest paths. Describe each path using edges in them. For example, you could describe a path from V1 to V2 as V1→V3→V2 (5 pts)



CSE331 Homework 6 Programming Problem
Due April 3
Submit source code files and README file via Handin

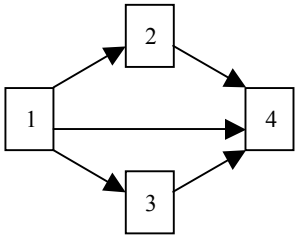
7. Implement the BFS algorithm using the pseudocode shown in lecture 18 and Weiss 2014 chapter 9.

   a) Input: your program must accept **two input parameters**. The first input parameter is an input file name (with relative path). The second input parameter is a string corresponding to the node ID of a node. For example, suppose your program has name "BFS". Then, an example command will be "./**BFS ./test/graph.txt 1**". The input graph file can be located anywhere. And it should be represented using adjacency list. The first line contains the number of nodes in the graph. The other lines contain adjacency lists for all nodes. An example input file is shown below:

   4
   1:2 3 4
   2:4
   3:4
   4:

   There are five lines in this file. Except the first line, other lines start with the node ID and a colon. Then all adjacent nodes are listed and separated using one spacer.

   The corresponding graph (node represented by boxes) is:



   We will run your program as "./**BFS ./test/graph.txt 1**", meaning that the start node is 1. We will try different start nodes.

   b) Output: we are looking for three types of output. 1): the output list (i.e. D in the code). 2) the parent node of each node. (p in the code). 3) the distance of each node from the starting node, (d in the code). For example, a sample output of running BFS to the above graph using 1 as the starting node will be:

   D = 1 2 3 4
   1.p=nil, 2.p=1, 3.p=1, 4.p=1
   1.d=0, 2.d=1, 3.d=1, 4.d=1

c) For your convenience, a sample input file can be found on D2L. Look for the file named "BFSinput1.txt" in the Homework 6 section of the Content pane.

Your program must be able to read files with the given format. We will test your program using different graphs with the same format.

d) When you need to put multiple nodes into the queue, use the order given in the adjacency list. For example, after processing node 1, the nodes in the queue is: (2,3,4).

e) Any programming language is fine. Make sure your program can compile and run at arctic.cse.msu.edu.

f) Total points: 40. You will get 5 pts if your source code is submitted but it cannot compile or run. If your program can run without any problem, it will be graded as follows:

    1) Take two arguments as input. The graph file can be located anywhere. 5 pts.
    2) We will have six test cases (same graph but different starting nodes, and different graphs). Correct output for each test case is worth 5 pts.