Name: Carlos Carrillo
Date: 02/29/2016
CS 261
Assignment 7

GRAPH TRAVERSALS

1. How is the graph stored in the provided code -- adjacency matrix or edge list?

All the graphs are stored in *edge list* representation. This is possible to see since the *Warshall's algorithm* for the adjacency matrix representation is not applying in any of the graphs provided.

2. Which of the graphs are connected? How can you tell?

All the graphs are connected except for Graph #3. It can be seen by observing the reachability among vertexes. In Graph #3, we have several unreachable vertexes, whereas all the vertexes are reachable in the other graphs. I makes sense since Graph #3 has fewer edges than vertexes, which makes impossible for al the vertexes to be connected.

3. *Image that we ran each search in the other direction (from destination to source, instead of source to destination) -- would the output change at all? What if the graphs were directed graphs?*

The output would change only if the graphs were directed. In this case, the label given to a vertex does not matter since we are manipulating an indirect graph, which means that all of connected vertices are reachable to each other in any direction.

4. *What are a few pros and cons of DFS vs. BFS?*

	PROS	**CONS**
DFS	• DFS may take less because you don't necessarily have to store all of the child pointers at each level.	• It's like a single person working a maze. • If solution is rare and may be deep, DFS might look a long time.
BFS	• It is like a wave flowing through a maze. • If solution is close to the root, BFS might be better and will find it quickly.	BFS may take up more space because it looks at all paths of a specific length at once.

5. *What's the Big O execution time to determine if a node is reachable from another node?*

   Taking into account that the total execution time of both DFS and BFS is O(Vertexes+Edges), I believe the execution time to determine if a node is reachable from another node depends on the number of neighbors that the vertex being evaluated has. Something like O(n), where n is the number of neighbors a particular node has.