1. **How is the graph stored in the provided code -- adjacency matrix or edge list?**
   Based on the formatting, it appears to be an edge list.
2. **Which of the graphs are connected? How can you tell?**
   Graphs 1, 2, 4, 5 are connected. Graph 3 was not. Via observation, I could readily tell if each vertex could be visited. Due to the gaps in graph 3, not every vertex could be hit. In graph 3, there are fewer edges than vertices, which means that not every single one can be visited and thus it is not connected.
3. **Imagine 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?**

   No, the output would not change, given the vertices are still connected to each other. If the graphs were directed, the output would change due to the fact of one way paths.

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

	PROS	CONS
DFS	Can theoretically find a solution faster. Uses less memory than BFS.	Is not guaranteed to find a workable path, for example, if it starts down an infinite path, it will continue to go down it. It can hit dead ends and can use trackbacks.
BFS	Will find a solution no matter what. Will give the shortest possible path.	Not fast. Not as memory efficient as a DSF.

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

The Big O Execution time for a graph problem is O(E+V), where e is the number of edges and v is the number of vertices. Sometimes written as O(n+m), but the former is preferred.