1. How is the graph stored in the provided code? Is it represented as an adjacency matrix or list?
    a. It is represented as a list

2. Which of the 3 graphs are connected? How can you tell?
    a. Graphs 2 and 3 are connected. We can tell because there is a path between every combination of 2 vertices. Graph 1 is not connected. For example, there is no path between vertices 0 and 3 in that graph.

3. Imagine that we ran each depth-first and breadth-first searches in the other direction (from destination to source). Would the output change at all? Would the output change if the graphs were directed graphs?
    a. No, there would be no change because it is not a directed graph. AKA, if there is an edge between 0 and 3, then you can transverse from 3 to 0 and from 0 to 3. If the graph was directed, there is a good likelihood that when the direction of the search is changed, no path would be available. For example, say we have a graph of three vertices, 0, 1, and 2, where the edges are directed in ascending order (0->1->2). In this graph, 0 to 2 would have a path but 2 to 0 would not have a path.

4. What are some pros and cons of DFS vs BFS? When would you use one over the other?
    a. DFS can often find a path faster than BFS because it will fully explore one path before searching a different path. If a path is found on the first try, then it stops and would not need to explore the rest of the graph. However, BFS would equally explore all parts of the graph (in a wave like pattern) until the correct path is found. The disadvantage, is that if a path is endless, the DFS can get caught and will fail. Because the BFS searches all options, it will not get stuck in an endless loop.
    b. Along the same lines, BFS can lead to the shortest path whereas DFS will only find a path that works, not necessarily the shortest. Based on what you want this could determine which search you use
    c. The BFS will use more memory than the DFS, that could also help decide which search to use.

5. What is the Big O execution time to determine if a vertex is reachable from another vertex?
    a. O(number of vertices + number of edges)