Thanhbinh Truong
CS261

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

2.Which of the 3 graphs are connected? How can you tell?

Of the 3 graphs, only graph 2 and 3 are connected while graph 1 is not. Within the output of the main program, it is illustrated in graph 2 and 3 that there are BFS and DFS paths for each vertex to every other, while graph 1 has missing paths between vertices.

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?
If we ran each depth-first and breadth-first searches in the other direction, the output would not change because the resulting edges would be crossed going either direction. Interestingly, if the graphs were directed graphs, the output would likely change because the vertices would only have access from one way rather than either directions.

4.What are some pros and cons of DFS vs BFS? When would you use one over the other?
Pros of DFS:
1. Destination could quickly be reached
2. Can traverse large graphs
3. Less memory

Cons of DFS:
1. Possibility of being in an infinite path
2. Might take too much time to process
3. Only allow simple path

Pros of BFS:
1. Will check all proximate paths/check all nodes reachable
2. Will always find path of exists/will determine shortest path

Cons of BFS:
1. Takes up more memory

You would rather use a DFS over a BFS if the graph is too large and deep. On the other hand, a BFS would be used over a DFS if the solution is not far from the root.

5.What is the Big O execution time to determine if a vertex is reachable from another vertex?
Both DFS and BFS are O(E). (*E = number of edges)