HW10 - Graphs

Prasun Surana
CS5008, Summer 1

1. The space complexity of an adjacency list is O(V+E), where V is the number of vertices in the graph and E is the number of edges. This is because the adjacency list stores a list of all vertices, with each edge associated with each vertex. Hence, to allocate memory for all vertices takes O(V) space, and allocating memory for all edges takes O(E), hence the total space complexity is O(V+E).

2. The space complexity of an adjacency matrix is O(V^2). This is because the adjacency matrix stores a grid of relationships between all vertices. If there are V vertices in a graph, this means that the adjacency matrix has to represent each vertex's relationship with V other vertices (V because it includes itself), and it has to do this for V vertices. Hence, to store all the relationships between vertices, it requires O(V^2) space.

3. The time complexity of a DFS in a graph depends on its implementation. If the graph is implemented using an adjacency matrix, the DFS will take O(V^2) time, because for each node in the matrix, we have to traverse the entire row to find all of its outgoing edges. Since there are V vertices, it will take O(V^2) time. However, if the graph is implemented using an adjacency list, where each node only stores the other nodes to which is has an outgoing edge, we only have to check the other vertices with which our vertex has a connection. The total number of edges in the graph is E, and since there are V vertices, the time complexity for DFS in an adjacency list is O(V+E).

4. The time complexity of a BFS in a graph depends on its implementation, but has the same time complexities as a DFS. Using an adjacency matrix, the algorithm still has to go through the entire row for each vertex to find all outgoing edges. Hence, it would be O(V^2). Using an adjacency list, the algorithm would have to go through all vertices once and all edges, so it would be O(V+E).