Alan García

1. What is the big-Oh space complexity of an adjacency list? Justify your answer.

   On the average case, on a non complete graph is the number of nodes + number of edges. The absolute worst case is still O(v^2) on a complete graph, because we would have n edges for every node, n being the number of nodes. Making it as inefficient as the matrix.

2. What is the big-Oh space complexity of an adjacency matrix? Justify your answer.

   A matrix has a constant space complexity of O(N^2), because we are performing a column by row of the same size operation.

3. What is the big-Oh time complexity for searching an entire graph using *depth-first search* (DFS)? Does the representation of the graph make a difference? Justify your answer.

   DFS takes O(V+E) vertices + edges to explore an entire graph (worst case) on an adjacency list. This is logical since we need to traverse each node one by one regardless of order.

   Since it is required for us to explore all of the current node's neighbors, we have to go through a matrix row or an adjacency list to figure out which are its neighbors, this would take longer on an adjacency matrix.

   If we have 8000 nodes and just 3 neighbors, then it just might be easier to go through an adjacency list rather than exploring the whole row of an 8000^2 items' adjacency matrix. A matrix would take around O(V^2) to explore the entire graph in the worst case.

4. What is the big-Oh time complexity for searching an entire graph using *breadth-first search* (BFS)? Does the representation of the graph make a difference? Justify your answer.

The same applies to BFS, in reality BFS and DFS are not that different except when we compare space complexity. Each algorithm has to go through each and every node, the way we access the data depends on the implementation.
BFS takes O(V + E) using an adjacency list vs O(V^2) when using a Col x row matrix.
It's different to access an already filtered list of items than checking for all neighbors again and again, for every node exists in the graph..