# Written Exercise for HW7 Graphs

Name: Rong Huang

Date:03/26/2024

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

Complexity: O(V+E), where V is the number of vertices (or nodes) and E is the number of edges in the graph.

In an adjacency list, each vertex has a list containing all of its adjacent vertices. This structure means that the space required is proportional to the total number of vertices V plus the total number of edges E. For each vertex, there is an overhead of maintaining a list of connections, and for each edge, there is an entry in one of these lists. This is particularly efficient for sparse graphs where E is much less than V^2, as it avoids allocating space for edges that do not exist.

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

Complexity: O(V^2), where V is the number of vertices in the graph.

An adjacency matrix is a V×V matrix where the cell at row i and column j indicates the presence of an edge between vertices i and j. The space required for this matrix does not depend on the number of edges E but on the number of vertices V, and it is squared because we have a row and a column for each vertex. This approach is space-inefficient for sparse graphs as it allocates memory for potential edges that do not exist, particularly when V is large and E is small.

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.

Complexity: O(V+E), where V is the number of vertices and E is the number of edges.

DFS explores as far as possible along each branch before backtracking. In terms of time complexity, it visits every vertex once and explores every edge in the worst case. For an adjacency list representation, this is efficient because each vertex's neighbors are directly accessible, and each edge is considered once. However, in an adjacency matrix, to find all adjacent vertices of a given vertex, one must traverse an entire row in the matrix, which takes O(V) time per vertex, leading to O(V^2) in the worst case for dense graphs. Thus, while the theoretical complexity remains O(V+E), the practical performance can be less optimal for dense graphs when using an adjacency matrix.

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.

Complexity: O(V+E), where V is the number of vertices and E is the number of edges.

BFS explores the neighbor nodes at the current depth prior to moving on to the nodes at the next depth level. Similar to DFS, the time complexity is O(V+E) because it needs to visit every vertex and explore each edge connecting to unvisited vertices. With an adjacency list, BFS is efficient as it quickly accesses all neighbors of a vertex. In contrast, with an adjacency matrix, checking for the presence of an edge to every other vertex takes O(V) time for each vertex, which can lead to O(V^2) time complexity for very dense graphs. Therefore, while the theoretical time complexity is the same for both representations, the actual efficiency of BFS can vary based on the graph's density and the representation used.