CSE 2216 : Data Structures and Algorithm 1 Lab

Problem Set for Graph and Binary Search Tree

Graph Lab Tasks

Task 1: Represent a Graph Using Adjacency Matrix and Adjacency List

Problem: Write a program to represent a graph using both an adjacency matrix and an adjacency list. Implement functions to add edges and display the graph representation.

Example Input:

- Number of vertices: 4
- Edges: (1, 2), (1, 3), (2, 4)

Example Output:

- Adjacency Matrix:

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- Adjacency List:

```
1: 2, 3
2: 1, 4
3: 1
4: 2
```

Task 2: Implement Depth-First Search (DFS)

Problem: Write a program to perform Depth-First Search (DFS) on a graph and print the traversal order.

Example Input:

- Number of vertices: 5
- Edges: (1, 2), (1, 3), (2, 4), (2, 5)
- Starting node: 1

Example Output:

```
DFS Traversal: 1 → 2 → 4 → 5 → 3
```

Task 3: Check if a Graph is Bipartite

Problem: Write a program to check whether a given graph is bipartite using BFS.

Example Input:

- Number of vertices: 4
- Edges: (1, 2), (2, 3), (3, 4), (4, 1)

Example Output:

```
Graph is Bipartite: Yes
```

Task 4: Find Shortest Path in an Unweighted Graph

Problem: Implement a BFS-based algorithm to find the shortest path between two nodes in an unweighted graph.

Example Input:

- Number of vertices: 6
- Edges: (1, 2), (1, 3), (2, 4), (3, 5), (4, 6), (5, 6)
- Source: 1, Destination: 6

Example Output:

```
Shortest Path: 1 → 3 → 5 → 6
```

Task 5: Detect a Cycle in a Directed Graph

Problem: Write a program to detect if there is a cycle in a directed graph using DFS.

Example Input:

- Number of vertices: 4
- Edges: (1, 2), (2, 3), (3, 4), (4, 2)

Example Output:

```
Graph contains a cycle: Yes
```

Binary Search Tree (BST) Lab Tasks

Task 1: Implement Basic BST Operations

Problem: Write a program to implement insertion and searching in a BST.

Example Input:

- Insert: 50, 30, 70, 20, 40, 60, 80
- Search: 40, 90

Example Output:

```
Search Result:
40: Found
90: Not Found
```

Task 2: Traversals in a BST

Problem: Write a program to perform Preorder, Inorder, and Postorder traversals in a BST.

Example Input:

- Insert: 50, 30, 70, 20, 40, 60, 80

Example Output:

```
Inorder Traversal: 20 → 30 → 40 → 50 → 60 → 70 → 80
Preorder Traversal: 50 → 30 → 20 → 40 → 70 → 60 → 80
Postorder Traversal: 20 → 40 → 30 → 60 → 80 → 70 → 50
```

Task 3: Find the Minimum and Maximum in a BST

Problem: Write a program to find the minimum and maximum value in a BST.

Example Input:

- Insert: 50, 30, 70, 20, 40, 60, 80

Example Output:

```
Minimum Value: 20
Maximum Value: 80
```

Task 4: Implement Deletion in a BST

Problem: Write a program to delete a node in a BST and display the Inorder traversal after deletion.

Example Input:

- Insert: 50, 30, 70, 20, 40, 60, 80
- Delete: 70

Example Output:

```
Inorder Traversal after Deletion: 20 → 30 → 40 → 50 → 60 → 80
```