2 Trees in Graph Theory

This module covers tree structures, spanning trees, tree traversals, and important properties. The exercises below will help you implement and visualize these concepts using Python.

2.1 Exercise 1: Constructing a Tree

Task: Implement a program to create a tree as an adjacency list.

Hint: A tree is an acyclic connected graph. Ensure no cycles are formed while adding edges.

```python
class Tree:
    def __init__(self):
        self.tree = {}

    def add_edge(self, u, v):
        if u not in self.tree:
            self.tree[u] = []
        if v not in self.tree:
            self.tree[v] = []
        self.tree[u].append(v)
        self.tree[v].append(u)

T = Tree()
T.add_edge(1, 2)
T.add_edge(1, 3)
T.add_edge(2, 4)
T.add_edge(2, 5)
print(T.tree)
```

2.2 Exercise 2: Tree Traversals

Task: Implement BFS and DFS for tree traversal.

Hint: Use a queue for BFS and recursion for DFS.

```python
from collections import deque

def bfs(tree, start):
    queue = deque([start])
    visited = []
    while queue:
        node = queue.popleft()
        if node not in visited:
            visited.append(node)
            queue.extend(tree[node])
    return visited

def dfs(tree, start, visited=None):
    if visited is None:
        visited = []
    visited.append(start)
    for neighbor in tree[start]:
        if neighbor not in visited:
            dfs(tree, neighbor, visited)
    return visited
```

2.3 Exercise 3: Minimum Spanning Tree (MST)

Task: Implement Prim's and Kruskal's algorithms to find an MST of a weighted tree.

Hint: Use priority queues for Prim's and sorting for Kruskal's algorithm.

2.4 Exercise 4: Tree Diameter

Task: Compute the diameter (longest path) of a tree.

Hint: Use two BFS/DFS traversals: one to find the farthest node, and another from there to determine the diameter.

2.5 Exercise 5: Lowest Common Ancestor (LCA)

Task: Implement LCA using Binary Lifting or DFS method.

Hint: Precompute ancestor tables or use depth tracking for efficient queries.

2.6 Exercise 6: Counting Number of Trees in a Forest

Task: Given a graph, count the number of disconnected tree components.

2.7 Bonus Challenge 1: Tree Center

Task: Find the center of a tree (nodes minimizing max distance to any other node).

2.8 Bonus Challenge 2: Random Tree Generation

Task: Generate a random tree of 'n' nodes and analyze its properties.

Hint: Use Prufer sequences or random spanning tree algorithms.