Data Structures

Lab -7

Date: 20.08.25

Observation Questions

1. Differentiate between a Threaded Binary Tree and a regular Binary Tree. What problem does threading solve?
2. Explain how a priority queue differs from a regular queue. Why is heap an efficient data structure to implement it?
3. State the differences between BFS and DFS in terms of traversal order, memory usage, and applications.
4. With an example, illustrate how an in-order traversal is performed using a Threaded Binary Tree without recursion/stack.
5. Mention two real-world applications each of BFS and DFS, justifying why they are suitable for those scenarios.

Execution Questions

1. Implement a threaded binary tree to store book IDs in a library. Perform in-order traversal without recursion or stack to display books in ascending order of ID.

- **Input**: Book IDs = [40, 20, 60, 10, 30, 50, 70]
- **Output**: In-order Traversal = 10 20 30 40 50 60 70

2. Implement a max-heap-based priority queue where patients are assigned numbers based on severity. Higher severity gets higher priority.

- **Input**: Patients (ID, Severity):
- (P1, 5), (P2, 2), (P3, 8), (P4, 6)
- **Output**: Order of Treatment = P3 → P4 → P1 → P2

3. Model a city as a graph where nodes represent places and edges represent roads. Use BFS to find the shortest path between two given places.

- **Input**: Graph Edges:
- A-B, A-C, B-D, C-D, D-E
- Start = A, Destination = E
- **Output**: Shortest Path = A → B → D → E

4. Given a graph representing a social network (nodes = people, edges = connections), use DFS to detect if there exists a cycle of friendships (mutual connections forming a loop).

- **Input**: Graph Edges:
- A-B, B-C, C-D, D-A, C-E
- **Output**: Cycle Found = A → B → C → D → A

5. Simulate task scheduling using a min-heap priority queue where smaller values indicate higher priority (e.g., deadline in hours).

- **Input**: Tasks:
- (T1, 4), (T2, 1), (T3, 3), (T4, 2)
- **Output**: Execution Order = T2 → T4 → T3 → T1