Assignment 03

Graphs & Trees

Flight Reservation System and Flood Fill Algo

Time Management

The combined estimated time to complete assignment is **20 hours**.

[2 HOURS] Requirement Understanding

1. **Event Scheduling System:**
    - Understand the requirements for event scheduling:
        - Key functionalities: add, update, delete events, find overlaps, calculate free slots, and print schedules.
        - Maintain a binary tree structure based on event start date and time.
        - Avoid overlapping events, even across multiple days.
    - Identify inputs and outputs:
        - Inputs: Event ID, name, start date & time, duration.
        - Outputs: Event schedule, overlapping events, free time slots.
    - Analyze constraints and scenarios:
        - Prevent event overlaps during addition or updates.
        - Rearrange the binary tree upon deletion.
2. **Flood Fill Algorithm:**
    - Grasp the concept of flood fill as a graph traversal problem.
    - Understand the traversal approaches:
        - DFS (recursive) and BFS (iterative).
    - Identify inputs and outputs:

        - Inputs: Image matrix, starting pixel (sr, sc), new color.
        - Output: Modified image matrix.
    - Analyze constraints:
        - Ensure all connected regions of the same color are replaced.
        - Avoid infinite recursion by handling boundary cases.

[4 HOURS] Software Design

1. **Event Scheduling System:**
    - Plan user interactions:
        - Add, update, delete events, find overlaps, and calculate free slots.
        - Print the schedule in chronological order.
    - Design the binary tree structure:
        - Node details: Event ID, name, start date & time, duration.
        - Binary tree property based on start date and time.
    - Define modular functions:
        - Add event: Ensure no overlaps and maintain binary tree properties.
        - Update event: Modify details without introducing overlaps.
        - Delete event: Remove nodes and rebalance the tree.
        - Overlap detection: Identify conflicts within a given time range.
        - Free time slots: Calculate available slots for a specific day or multiple days.
        - Print schedule: Use in-order traversal to display events chronologically.
2. **Flood Fill Algorithm:**
    - Choose traversal approach:
        - DFS for recursion simplicity or BFS for iterative implementation.
    - Design base cases:
        - Out-of-bounds pixels.
        - Pixels already filled with the new color.
    - Plan helper functions:

        - Recursive or iterative traversal.
        - Color replacement logic.

[10 HOURS] Software Coding

1. **Event Scheduling System:**
    - Implement binary tree operations:
        - Add event: Insert nodes while ensuring no overlapping events.
        - Update event: Modify nodes while maintaining tree properties.
        - Delete event: Remove nodes and rebalance the tree as needed.
    - Implement utility functions:
        - Find overlaps for a given time range.
        - Calculate free time slots by analyzing gaps between events.
        - Print the schedule using in-order traversal.
    - Integrate the operations into a unified system.
2. **Flood Fill Algorithm:**
    - Implement flood fill:
        - DFS approach: Use recursion to traverse and replace connected regions.
        - BFS approach: Use a queue for iterative traversal.
    - Handle edge cases:
        - Pixels on the boundary or already filled with the new color.
        - Large matrices with complex connected regions.
    - Test with sample image matrices of varying sizes.

[3 HOURS] Software Testing

1. **Event Scheduling System:**
    - Test binary tree operations with edge cases:
        - Adding events with conflicting time ranges.
        - Updating events to create or resolve overlaps.

        - Deleting nodes with different configurations (no child, one child, two children).
    - Validate outputs for:
        - Overlapping event detection.
        - Free time slot calculation across multiple days.
        - Chronological schedule printing.

2. **Flood Fill Algorithm:**
    - Test flood fill on various image matrices:
        - Small and large sizes.
        - Starting pixels on edges or corners.
        - Already-filled regions and disconnected regions.
    - Compare results for DFS and BFS implementations.

[1 HOUR] Software Documentation

1. **Document the design and implementation:**
    - Explain binary tree operations and flood fill logic.
    - Describe modular functions for both problems.
2. **Add comments for clarity:**
    - Highlight key steps in each function.
    - Note edge cases and testing scenarios.
3. **Format the code with proper indentation and naming conventions.**

Problem 1

Event Scheduling System

You are tasked with building an Advanced Multi-Day Event Scheduling System using a binary tree. This system should handle events across multiple days, where each node in the binary tree represents an event and stores the following information:

1. Event ID (integer): A unique identifier for the event.
2. Event Name (string): A short description of the event.

3. Event Start Date & Time (string): The starting date and time in the format YYYY-MM-DD HH:MM.
4. Event Duration (integer): The duration of the event in minutes.
5. Left Child: Represents an event scheduled before the current event (based on start date and time).
6. Right Child: Represents an event scheduled after the current event (based on start date and time).

Functional Requirements

1. **Add Event:**
   Write a function to add an event to the tree. Ensure:
   - Events do not overlap, even if they span multiple days.
   - The binary tree property is maintained based on start date and time.
2. **Update Event:**
   Write a function to update the details of an existing event (e.g., its name, start time, or duration). Ensure the updated event does not cause overlaps.
3. **Delete Event:**
   Write a function to delete an event by its ID. Rearrange the tree to maintain the binary tree property.
4. **Find Overlapping Events:**
   Write a function that, given a specific time range (input: start time and end time), identifies all overlapping events.
5. **Find Free Time Slots Across Days:**
   Write a function to calculate all free time slots for a specific day or multiple days. Return these as ranges of available times.
6. **Print Full Schedule:**
   Write a function to print the entire schedule in chronological order (in-order traversal), including Event ID, Name, Start Date & Time, and Duration.

Problem 2

Flood Fill Algo

You are given a 2D grid representing an image, where each element in the grid represents the color of a pixel. A pixel is defined by its row and column indices (sr, sc), and its color is given by image[sr][sc].

Your task is to perform a flood fill on the image starting from the pixel (sr, sc) and replace all pixels in the connected region (including the starting pixel) with a new color newColor.

Key Details:

1. **Connected Region:**
   A region is considered connected if:
   - The pixel has the same color as the starting pixel.
   - It is directly adjacent to another pixel in the region (vertically or horizontally).
2. **Behavior:**
   - Replace the color of the starting pixel and all connected pixels with newColor.
   - Do not replace pixels that are not part of the connected region.
3. **Image Dimensions:**
   The image is a 2D array of integers, where:
   - image[i][j] represents the color of the pixel at position (i, j).

Input Format:

1. image (2D vector of integers): A matrix of size m x n where 1 <= m, n <= 50.
2. sr (integer): Row index of the starting pixel.
3. sc (integer): Column index of the starting pixel.
4. newColor (integer): The color to apply to the connected region.

Output Format:

Return the modified image after performing the flood fill operation.

Example 1:
Input:

image = [

[1, 1, 1],

[1, 1, 0],

[1, 0, 1]

]

sr = 1, sc = 1, newColor = 2

Output:

[

[2, 2, 2],

[2, 2, 0],

[2, 0, 1]

]

The pixel at (1, 1) has color 1. All connected pixels with color 1 are replaced with 2, as shown: