Data Structure
Semester Project(Lab Part)

Rifad Ahamed

ID:242-16-028

Section: 20(A)

Table of Content…… 2

Task-1

Answer to the Q1.

Mr. Sisir's Ticket Booking System

Used Data Structure	Why used
Queue (FIFO)	Ensures fair processing of ticket requests (first come first Input). Who comes first to buy ticket he first gets tickets.
Hash Table	Get available info about the ticket holder with id and number
Graph	Shows all possible travel routes between cities and Calculates cheapest or fastest routes automatically
Tree	For quickly searching the available ticket

Mr. Tushar's Music App

Used Data Structure	Why used
Doubly Linked List	It is easy to add or remove songs anywhere in playlist. It can go to next OR previous song with one click. Flexible for making changes .
Hash Table	Find any song with it's name instantly, quickly accessible .
Binary Search Tree (BST)	Keeps the song sorted with alphabetically or with title. Allows quick searching.

Task- 1

Ans. To the Q2.

An adjacency matrix is asymmetric if matrix[i][j] ≠ matrix[j][i] .
Since Figure 01 represents a directed graph, travel cost from A → B may not equal B → A.
Thus, the matrix is not symmetric, proving it is asymmetric.

Proof :

Nodes (Vertices): DIU, Cox's Bazar, Rangamati, Saint Martin, Kuthibari, Sajek Valley, Ahsan Manzil.

Edges: Represent travel costs between nodes

Suppose we want to create an adjacency matrix by numbering the vertices.
Let's show an example where,

`A [i] [j] ≠ A [j] [i]:`

Example 1: Between Node 1 and Node 2

- `A[1][2] = 25`
- `A[2][1] = -15`

Clearly: `A[1][2] ≠ A[2][1]` → Matrix is **not symmetric**

Example 2: Between Node 4 and Node 5

- `A[4][5] = 25`
- `A[5][4] = 0` (no edge from 5 to 4)

`25 ≠ 0` → Not symmetric

Example 3: Between Node 6 and Node 5

- `A[6][5] = 0`
- `A[5][6] = -11`

`0 ≠ -11` → **Not symmetric**

We prove that:

The adjacency matrix of **Figure 01** is **asymmetric.**

Task- 1

Ans. To the Q3

The instructor said to choose 10 different integers between 20 and 200, where:

1st = 26

4th = XX

7th = 86

9th = 52

XX = (Last two digits of Student ID % 10) + smallest number

Only 2 digits less than 35

Maximum 4 digits more than 110

Example Student ID: 24216028

- Last two digits = 28
- 38 % 10 = 8
- Smallest number = 26
- Then XX = 8 + 26 = 34

Now Select 10 numbers according to the condition

The selected numbers are: [26, 45, 72, 34, 108, 120, 86, 99, 52, 135]

only two numbers are less than 35 □ No more than four numbers exceed 110.

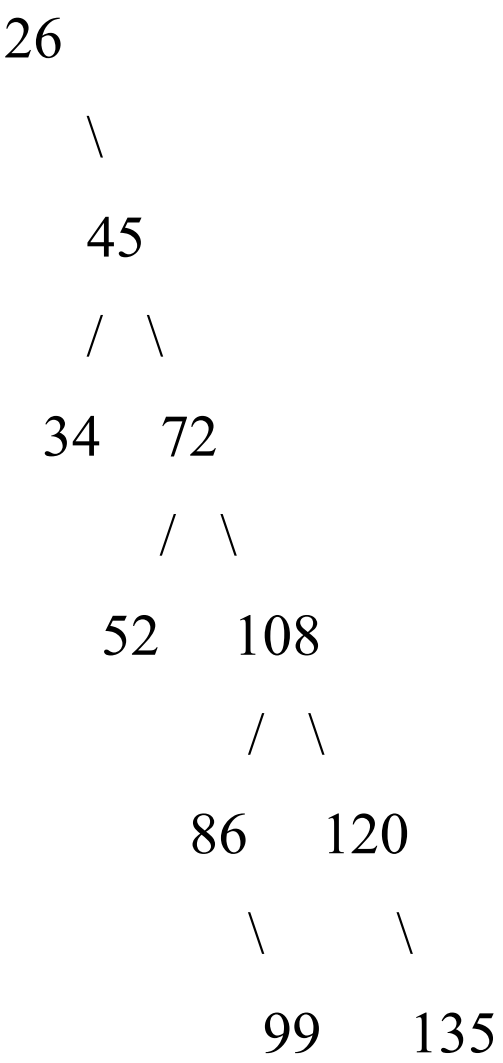
Constructing the Binary Search Tree (BST)

Mr. Arif inserted the selected numbers into a BST in the following order:

1. **26** → Root node
2. **45** → inserted to the **right** of 26
3. **72** → inserted to the **right** of 45
4. **34** → inserted to the **left** of 45
5. **108** → inserted to the **right** of 72

6. **120** → inserted to the **right** of 108
7. **86** → inserted to the **left** of 108
8. **99** → inserted to the **right** of 86
9. **52** → inserted to the **left** of 72
10. **135** → inserted to the **right** of 120

This creates a well-structured **Binary Search Tree**.



Mr. Arif then applied Binary Search on the BST to find any given value (e.g., **52**). Here's how the search proceeds:

- Start at **26** → 52 > 26 → move to the right
- At **45** → 52 > 45 → move to the right
- At **72** → 52 < 72 → move to the left

At **52** → **Value found!**

This shows how efficiently Binary Search works on a Binary Search Tree.

Task- 1

Ans. to the Q4

Mr. Habib was able to create a BST (Binary Search Tree) correctly, but he failed to find his Student ID.

The reason for this could be:

- ❖ Not understanding the correct formula for finding XX

(i.e. XX = (last two digits of Student ID % 10) + lowest number)

- ❖ Lack of skill in how to find information from BST using Binary Search

Mr. Arif helped him to teach how to calculate XX using the last two digits of the Student ID

For example:

- If the Student ID is: 242-16-028 → Last two digits = 28
- 28 % 10 = 8
- Let's say the smallest number is 26

- Then XX = 8 + 26 = 34

Explains how Binary Search works in BST

He teaches Mr. Habib how to quickly find a value in BST starting from the root and going right-left. Helps with the search process with simple examples

Mr. Arif then applied Binary Search on the BST to find any given value (e.g., 52). Here's how the search proceeds:

- Start at 26 → 52 > 26 → move to the right
- At 45 → 52 > 45 → move to the right
- At 72 → 52 < 72 → move to the left

The value is found.

He helps with hands-on assistance on how to move right or left in BST step by step.

Task- 2

Ans to the Q5

To find the minimum travel cost from **DIU** to **X**, we follow the given formula:

X = ((Last two digits of Student ID) % 6) + 1

Given, the last two digits of the Student ID = **38**

So,

X = (38 % 6) + 1 = 2 + 1 = 3

Therefore, we need to find the minimum travel cost from **DIU (Node 7)** to **Node 3 (Ahsan Manzil)**.

Available Paths from DIU (Node 7):

1. **Direct path:**
   - 7→ 3 = **13**
2. **Other paths** 7 →4→ 3

- 7 → 4 = 50
- 4→3 = 8
- **Total cost = 50+8 = 58**

3. **Other paths** 7 → 6 → 5 → 3
   - 7 → 6 = 88
   - 6 → 5 = -11
   - 5 → 3 = 15
   - **Total : 88 - 11 + 15 = 92**
4. **Other paths** 7 → 1→ 2 → 4→3
   - 7 → 1 = 94
   - 1 → 2 = -15
   - 2 → 4 = 15
   - 4 → 3 = 8
   - **Total : 94-15 + 15 + 8 = 102**

Minimum Travel Cost: **13**

Therefore, the minimum travel cost from DIU to Ahsan Manzil is: 13

Task- 2

Ans. to the Q6

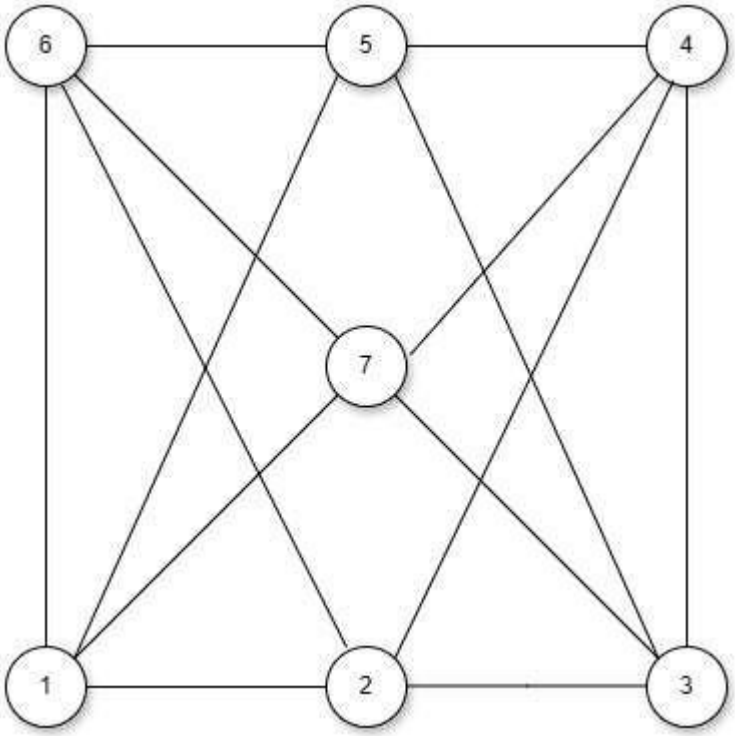
Chromatic Number :-

The Chromatic Number of a graph is the minimum number of colors required to color all the nodes (vertices) of the graph in such a way that:

No two adjacent (connected) nodes share the same color.

Figure 01 is a Directed Weighted Graph. However, when determining the Chromatic Number, the direction of the arrows is not important.

Therefore, we treat it as an Undirected Graph — meaning, if there is an edge 7 → 1, we consider it simply as 7— 1 (a connection exists between 7 and 1 regardless of direction).



Undirected Graph

Assuming the connections are Undirected:

7 — 1, 3, 4, 6

1 — 2, 6, 5

2 — 3, 4, 6

3 — 2, 4, 5

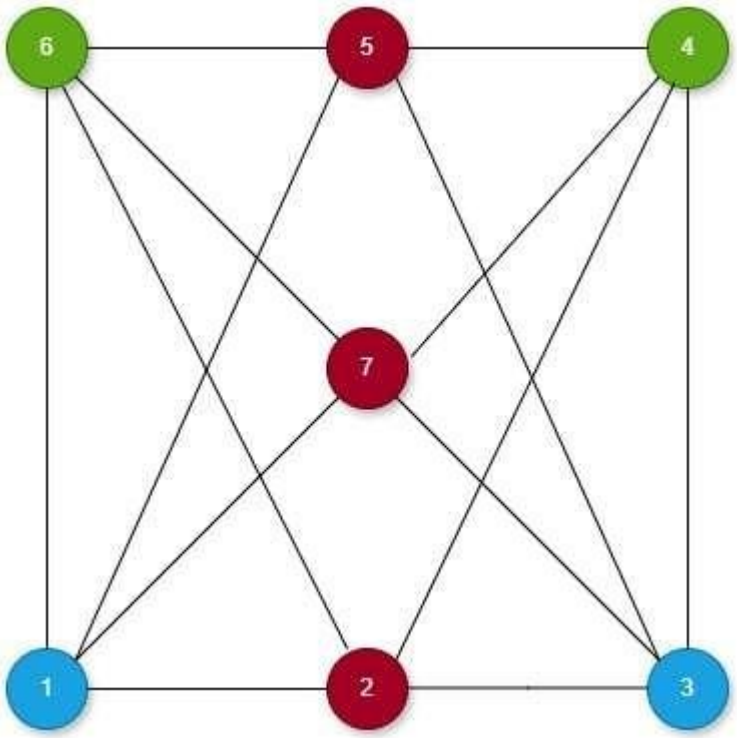
4 — 3, 5

5 — 1, 3, 4, 6

6 — 5, 2

Greedy Coloring Algorithm .

We will color each vertex in such a way that it does not match any of its connected vertices.



Chromatic Number of Figure 01 = 3

Because, it is possible to color all vertices using a minimum of 3 colors in such a way that no two connected vertices have the same color.

Task- 2

Ans. to the Q7 (QA)

QA: Pseudocode to delete ‘XX’ from the linked list

Mr. Tamim was asked to delete a specific element XX from a singly linked list. Given that his CGPA was 3.05, the value of XX was calculated as:

XX = CGPA + 3.21 = 3.05 + 3.21 = 6.26

To solve this, Mr. Tamim used pointer manipulation in a singly linked list to search for and delete the node with value 6.26. His pseudocode is given below:

```
Function deleteNode(head, key = 6.26):
    if head is NULL:
        return NULL

    # Case 1: The node to delete is the head node
    if head.data == 6.26:
        temp = head
        head = head.next
        delete temp
        return head

    # Case 2: Traverse to find the node before the target
    current = head
    while current.next != NULL and current.next.data != 6.26:
        current = current.next

    if current.next == NULL:
        print "Node with value 6.26 not found"
        return head

    temp = current.next
    current.next = current.next.next
    delete temp
```

return head

QB: Is it possible to display the elements in reverse order?

Yes, it is possible to display a singly linked list in reverse order. Mr. Tamim used recursion to achieve this.

Pseudocode (using recursion):

plaintext

Copy

Edit

algorithm

```
Function printReverse(node):
    if node is NULL:
        return
    printReverse(node.next)
    print node.data
```

Explanation:

The function first moves to the end of the list by calling itself recursively.

When it reaches the last node, it starts printing the values during the return phase of each function call.

This naturally prints the list in reverse order.

Alternatively, he mentioned that a stack could also be used to reverse the order, since stacks follow Last In First Out (LIFO), but recursion is more elegant for this task.