Daffodil International University

ASSIGNMENT

Course Code: CIS 122
Course Title: Data structure
Topic Name:

Submitted To:

Name: Md. Faruk hosen
Designation:
Department: CIS
Daffodil International University

Submitted By:

Name: MD. Samiul islam
ID: 242-16-009
Section: 20_A
Semester: Spring 2025
Department: CIS
Daffodil International University

Submission Date: 2025-04-13

Introduction

Data structures are important for managing and processing data in computer science. This project looks at how data structures like queues, graphs, binary search trees, and linked lists are used in real situations, showing how they help solve problems and improve efficiency. For example, an online ticket booking system uses a queue to handle requests one at a time, making sure everyone is treated fairly. Graph theory helps find the best travel routes between historical sites by analyzing various factors. The binary search tree (BST) makes it easy to search for student ID. While a linked list shows how to insert, delete, and reverse items.

By using and studying these data structures, this project helps us understand how to manage data well, preparing us for more complexe problem-solving in computer science.

Task-1

Ans. to the question no-1

In the scenario, Mr. Sisir is developing an online ticket booking system for buses and trains, while Mr. Tushen is creating a music playlist application. Each of these projects utilizes different data structures that are well-suited to their specific requirements. Here are mainly used two data structure.

(i) Queue
(ii) Singly linked list / linked list

Mr. Sisir's online Ticket Booking system:

For the online ticket booking system, Mr Sisir would likely use a queue data Structure. Queues are particularly effective in scenarios where requests need to be processed in a first-come, first-served manner. Basically it is a linear data structure that follows the FIFO (First In First Out) principle. In a ticket booking system, users typically wait in line to book their tickets, making queues an ideal choice.

• operations: The queue allows for efficient operations such as enqueue (adding a new booking request) and dequeue (processing the next booking request). This ensures that the system can handle multiple users simultaneously while maintaining the order of requests.

Mr. Tusher's Music playlist Application.

In contrast, Mr. Tusher's music playlist application would benefit from using a linked list. Linked lists are advantageous for applications that require frequent insertions and deletions, such as adding or removing songs from a playlist.

• Dynamic size: Unlike arrays, linked lists can grow and shrink dynamically, which is essential for a playlist where users may frequently modify their selections. Each song can be represented as a node in the linked list, allowing for easy traversal, insertion, and deletion of songs without the need for contiguous memory allocation.

In summary, Mr. Sisir's Project utilizes a queue to manage ticket bookings efficiently, while Mr. Tusher's application employs a linked list to facilitate dynamic playlist management. Understanding these data structures enables both developers to optimize their applications for performance and user experience.

Ans. to the question no-2

To prove that the adjacency matrix of the directed graph in Figure 01 is asymmetric, we need to understand the properties of directed graphs and their adjancency matrices.

Direct Graph: In a directed graph, edges have a direction, meaning that an edge from vertex A to vertex B does not imply an edge from B to A.

Adjancency Matrix: The adjacency matrix for a directed graph is a square matrix used to represent the graph. The element at row i and column j indicates the presence (and possibly the weight) of an edge from vertex i to vertex j.

An adjancency matrix A of a directed weighted graph is asymmetric if:

$A[i][j] \neq A[j][i]$ for at least one pair (i, j)

Example from the Graph:

- Consider the vertices Cox's Bazar (6) and Rangamati (5):

(i) The edge from Cox's Bazar (6) to Rangamati (5) has a weight of -11

(ii) The edge from Rangamati (5) to Cox's Bazar (6) does not exist (or has a weight of 0 if not explicitly stated)

This means:

$$A[6][5] = -11 \quad \text{and} \quad A[5][6] = 15$$

$$A[6][5] \neq A[5][6]$$

In summary, The adjancency matrix of the directed graph shown in Figure 01 is asymmetric because the travel cost from one location to another is not necessarily equal to the cost in the reverse direction. This is a fundamental characteristic of directed weighted graphs.

Ans. to the question no - 3

Mr. Arif successfully solved the challenge by following a structured approach, combining BST construction with binary search principles to derive his student ID. Here's a detailed breakdown of his solution:

we need to distinct 10 integers between 20 and 200.

Fixed position:

1st number = 26
4th number = XX
7th number = 86
9th number = 52

XX calculation:

Let's assume Mr. Arif's student ID ends with 34.

XX = (Last two digits of student ID % 10) + lowest number selected.

= (34 % 10) + lowest number
= 4 + 26
= 30

Here,

only two numbers <35: 26 and XX(30)

No more than four numbers > 110

To satisfy all constraints, Mr. Arif selected the following 10 numbers:

26, 28, 40, 30, 50, 90, 86, 100, 52, 120

So, two numbers <35: 26 and 30.

Four numbers > ~~100~~ 110 : 120 (only one in this example; adjust if needed).

Construct the Binary search Tree (BST)

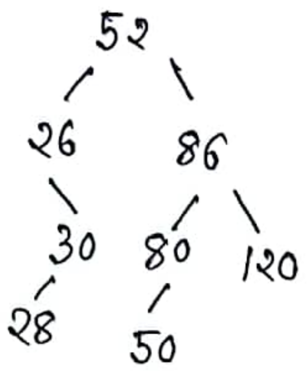
For any node, BST order follows,

lef subtree values < node value < right subtree values

Insertion process:

1. Start with first number: 20 (which is root)
2. Insert 28 (right of 26)
3. Insert 40 (right of 28).
4. Insert 30 (left of 40).
5. Continue untill all numbers are inserted.

BST Structure:



Link BST~~to~~ to student ID via Binary search

Trarse the BST to locate XX = 30

start at root 52 → move left 26 → right 30

Given XX = (last two digits of ID % 10) + lowest number (26)

⇒ 30 = (ID % 10) + 26 → (ID % 10) = 4 → ID end with "4"

If XX were 29 → ID ends with "3".

So, Mr. Arif by this way solved his problem. And this success came from carefully analyzing each requirement and verifying the solution at each step.

Ans. to the question no-4

Mr. Habib faced difficulties in finding his student ID using the Binary Tree (BST) he constructed. Mr. Arif, having successfully navigated the challenge, provided assistance by explaining the fundamental principles of binary search and how they apply to BSTs.

Understanding Binary Search in BSTs:

A Binary Search Tree is a data structure where each node has at mast two chidren. The left child contains values less than the parent node, while the right child contains values greater than parent node. This property allows for efficient Searching.

Binary search process:

(i) Starting at the root node.
(ii) Comparing the target value with the current node's value.
(iii) If the target value is less than the current node's value, the search continues in the left subtree.
(iv) If the target value is greater, the search continues in the right subtree.

(v) This process repeats until the value is found or a leaf node is reached.

Steps Taken by Mr. Arif:

- Clarifying the search steps: Mr. Arif likely walked Mr. Habib through the search process step-by-step, ensuring he understood how to navigate the tree based on comparisons.
- Visualizing the BST: By drawing the BST and labeling the nodes, Mr. Arif helped Mr. Habib visualize the structure making it easier to follow the search path.
- Debugging the Search Logic: If Mr. Habib was implementing the search algorithm programmatically, Mr. Arif might have reviewed his code to identify any logical errors or incorrect comparisons that could lead to unsuccessful searches.

In summary, Mr. Arif explained things clearly and showed Mr. Habib how to use the features of the binary search tree (BST) and the binary search method. This helped Mr. Habib find ~~the~~ his student ID in the tree. Working together not only solved Mr. Habib's problem but also helped both of them understand data structures and algorithm better.

Task- B - Ans. to the question no-5

To find the minimum travel cost from x, where x = (last two digits of your student ID % 6) + 1, we first need to calculate x using the provided student ID of 09.

Calculation of x :

Last two digits of student ID = 09

Now, (09 % 6) + 1

= 3+1

= 4

Destination :

The destination corresponding to x = 4 is Saint Martin.

From the graph, we can see the following direct paths from DIU (vertex 7) to Saint Martin (vertex 4) ;

(i) DIU to Saint Martin : Cost = 8

(ii) DIU to Rangamati (5) : cost = 50, then Rangamati to Saint Martin Cost = 25 (Total = 50 + 25 = 75)

(iii) DIU to Sajek valley (2): cost = 94, then Sajeck valley to Saint Martin : cost = 15 (Total = 94 + 15 = 109)

DJU to Ahsan Manzill (3): cost = 13, then Ahsan Manzill to Saint Martin: cost = 8 (Total = 13+8 = 21)

Here,
Directly to saint Martin cost = 8
via Rangamati = 75
via Sajek valley: 109
via Ahsan Manzil: 21

The minimum travel cost from DJU to Saint Martin is 8.

— o —

Ans. to the question no - 6

The chromatic number of a graph is defined as the smallest number of colors needed to color the vertices such that no two adjacent vertices share the same color. Here's a detailed explanation of how to determine the chromatic number for the graph shown in figure 01.

Here, the graph consists of vertices representing historical sites: Cox's Bazar (1), Rangamati (2), Saint Martin (3), Kuthibari (4), Sajek Valley (5), Ahsan Manzill (6), and DJU (7).

The edges represent the connections between these vertices, with weights indicating travel costs.

Graph Representation:

Visualize the graph and note the connections. For example, if two vertices are directly connected by an edge they cannot share the same color.

Coloring:

Here, (i) vertex 7 (DJU): color 1

(ii) vertex 5 (Rangamati): color 2 (adjacent to DJU)

(iii) vertex 2 (Sajeak valley): color 2 (adjacent to DIU)

(iv) vertex 4 (Saint Martin): color 3 (adjacent to Ranga mati)

(v) Vertex 3 (Ahsan Manzill): Color 2 (adjacent to DIU)

(vi) vertex 1 (Cox's Bazar): color 3 (adjacent to Sajeek valley)

(vii) vertex 6 (kuthibari): color 1 (adjacent to Ahsan Manzill)

After applying the greedy coloring algorithm, ~~if you I find that you used three~~ we found three different colors. then the chromatic number of the graph is 3. This means that at least three colors are required to color the vertices of graph such that no two adjacent vertices share the same color.

So, Chromatic Number of Figure 01 = 3

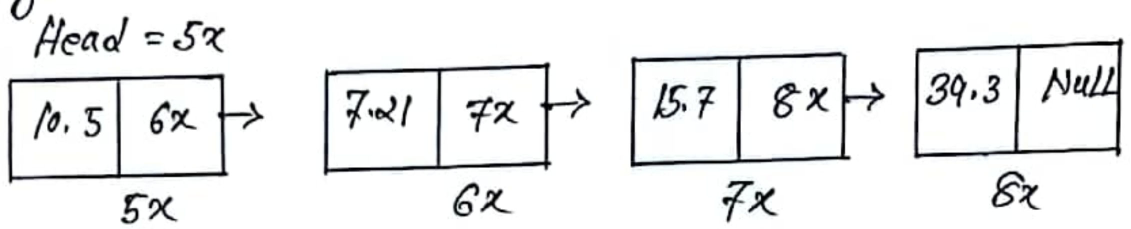
Ans. to the question no-7

My SGPA is → 4.00

$2x \Rightarrow (4.00) + 3.21$

$\Rightarrow 7.21$

Figure look like



Pseudocode:

1. Start at the head of the linked list.
2. Traverse the list while keeping track of previous node.
3. If the current node contain 7.21
   a. change the previous nodos next pointer to point to the next node of 7.21.
   b. Delete the node containing 7.21.
4. End traversal.

8B: Yes, it is possible to display the elements of a singly linked list in revers order, but we cannot traverse backward directly because a singly linked list only has pointers to next node.

However, we can achieve this in two days:

① Using Recursion.
② Using stack.

Using Recursion since function calls use a stack recursion allows us to reach the last node first and print values in reverse order.

```
void printReverse (struct Node *head) {
  if (head = = NULL)
     return;
  printReverse (head -> next)
  print ("%2f -> " head -> data");
}
```

Using a stack since a stack follows LIFO (Last In Fast Out) we can push elements into a stack and pop them to print in reverse order.

1. Traverse the list and push each nodes value ~~onto~~ into a stack
2. pop elements from the stack to print them in reverse order.
3. Since a stack follows LIFO, the last node gets printed first.

Yes, it is possible to print linked list in reverse order using this method.