Assignment: 11

- Title: Department maintain a student information. The file contain roll number, name, division and address. Allow user to add, delete information of student. Display information of particular employee.

- Objectives: The objective help to
  1. Add a student's information to the file.
  2. Delete a student's record from the file based on the roll number.
  3. Display the details of a specific student when their roll no is provided.
  4. Display an appropriate message if record does not exist when searching.

- Learning outcome:
  1. understand how sequential file storage work in the context of handling student record.
  2. Implement basic file operation such as adding, deleting and searching for record in sequential file.

- Theory:

1. Sequential files:
   It is type of file where record are stored one after another and to access a specific record, the system must read file sequentially from beginning to the desired location. This is simplest form of file storage

- Adding a record: This involve appending new student data at the end of the file.

Sundaram
FOR EDUCATIONAL USE

- Deleting a record: Deleting a record in sequential file typically involve reading the file, removing the specified record and then rewriting the remaining record back to file.

- Searching for a record: To find a student's record the system read the file sequentially and compare the roll no. until it find the match.

- File Handling
File Handling in programming refers to the process of reading and writing data from / to files. In Java, file handling can be done using classes like filewriter, buffered writer, filereader and the bufferedreader.

- Challenges:
1. Deletion: Deleting a record from sequential file require rewriting the entire file, which can be inefficient for large files.

2. Search: Searching for a record involve scanning through all the record until the desired one is found.

FOR EDUCATIONAL USE

- Input: The user will be prompted to provide input
1. student Data for adding a new student
   - roll number
   - Name
   - Division
   - Address
2. Roll Number for
   - Deleting a student's record.
   - Displaying the details of a student.

- output: The program should provide the output based on the operation performed.
1. for adding a student:
   - confirmation msg: "Student added successfully"
   - updated list of student record after insertion.
2. for Deleting a student:
   - confirmation msg: "student record deleted successfully!"
   - updated list of student record after deletion.
3. for displaying student info:
   - If the student is found: display the student's details.
   - If the student is not found: Display an error message: "Record not found."

Sundaram
FOR EDUCATIONAL USE

- conclusion : In this practical, we learned how to manage student information using sequential files. we implemented a system that allow for adding, deleting and displaying student record, demonstrating file handling operation in Java.

17/04/25

Assignment : 9

- Title : A dictionary store keyword and its meaning provide facility for adding new keywords, deleting keywords, updating value of any entry. provide facility to display whole data sorted in ascending or discending order. Also find how many maximum comparison may require for finding any keyword. use Height balance tree and find the complexity for finding a keyword.

- objective :
  1. To add a new keyword and its meaning to the dictionary.
  2. To delete a keyword from the dictionary.
  3. To update the meaning of an existing keyword.
  4. Display the entire dictionary sorted either in ascending or descending order.

- Learning outcome :
  1. understand the concept of Height balanced tree and its importance in maintaing balanced structure for efficient operation.
  2. Implement and use an AVL Tree for dictionary management to perform various operation like insertion, deletion and search.

- Theory :

1 Height Balanced tree (AVL tree)

The AVL tree is self balancing binary search tree.

In an AVL tree, the difference between the height of left and right subtrees of any node is at most 1. This balance ensure that tree remain approximately balanced, allowing for logarithmic time complexity for operation like search, insertion and deletion.

- Insertion : Insert a new node while ensuring that the tree remain balanced after each insertion.

- Deletion : Remove a node and re-balance the tree.

- Rotation : Re-balancing is performed using rotations to maintain the AVL property

- Input :

1. Keyword : A string representing the keyword

2. Meaning : The definition or meaning associated with the keyword.

3. operation :
   - Add a keyword and its meaning.
   - Delete a keyword.
   - update the meaning of an existing keyword.
   - search for a keyword.

4. sorting : Display all the dictionary entries in ascending or descending order.

Sundaram
FOR EDUCATIONAL USE

- Conclusion:

In this practical, we implemented a dictionary management system using an AVL tree. The key operation were efficiently handled using the AVL tree, ensuring logarithmic time complexity for these operation.

29/03/26

FOR EDUCATIONAL USE

Assignment No: 6

- Title: Represent a given graph using an adjacency matrix to perform DFS & using an adjacency list to perform BFS. use map of the area around college as graph. Identify the prominent landmark as node and perform DFS and BFS on that.

- Objectives: 1. To understand the concept of graph traversal technique.
2. To learn and implement DFS and BFS

- Learning outcomes:
1. Define a class for graph using OOP concept
2. Analyze the working of DFS and BFS function
3. Represent real-world scenarios using graphs.

- Theory:
Graph are fundamental data structure used to model relationships between different entities. They consist of nodes and edges.

Graph Representation:
1. Adjacency matrix:
- A 2D array where matrix[i][j] is 1 if there is an edge between node i and node j. otherwise 0.
- Best for dense graph but consumes more space for sparse graph.

2. Adjacency list:
- Each node maintain a list of all its neighboring node.

Sundaram
FOR EDUCATIONAL USE

- Efficient for sparse graphs, using less memory than an adjacency matrix.

Graph traversal techniques:

1. Depth-First search

DFS is recursive graph traversal algorithm that explore as far as possible along one branch before backtracking.

- Algorithm
- Start at given node
- mark the node as visited and explore an adjacent unvisited node.
- Repeat process for each unvisited neighbor until no more node remain.
- Backtrack if no adjacent unvisited node are left
- Time complexity: O (V+E)

2. Breadth-first search.

BFS explore all neighbor of node before moving to the next level of neighbors. it uses a queue for traversal.

- Algorithm
- Start at given node and enqueue it.
- Dequeue a node, mark it as visited and enqueue all its unvisited neighbors.
- Repeat until the queue is empty.
- Time complexity: O (V+E)

daram
FOR EDUCATIONAL USE

* Graph Representation for given problem:
- For this assignment, we cdr a real-world map around the college where:
- Nodes represent prominent landmarks.
- Edges represent roads connecting these landmark
- Ex.
  - Node (Landmark)
    - A : college
    - B : Library
    - C : canteen
    - D : Bus stop
    - E : Hostel
  - Edges
    - A ⟷ B
    - A ⟷ C
    - B ⟷ D
    - C ⟷ D
    - D ⟷ E

Graph representation

1. Adjacency matrix representation (for DFS)

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 0 & 0 \\ B & 1 & 0 & 0 & 1 & 0 \\ C & 1 & 0 & 0 & 1 & 0 \\ D & 0 & 1 & 1 & 0 & 1 \\ E & 0 & 0 & 0 & 1 & 0 \end{array}$$

Sundaram
FOR EDUCATIONAL USE

2. Adjacency list representation (for BFS)

A → B, C

B → A, D

C → A, D

D → B, C, E

E → D.

- Input :

1. No. of landmark
2. connection between landmark

- output

1. Adjacency matrix representation
2. Adjacency list representation.

- conclusion :

This program provide insight into graph traversal technique and their real-world application. DFS is implemented using an adjacency matrix or list and BFS is implemented using an adjacency list for efficient memory usage.

14/02/25

Sundaram FOR EDUCATIONAL USE

Assignment No: 5

- Title: convert given binary tree into a threaded binary tree and analyze time and space complexity.

- Objectives: 1. understand the concept of threaded binary tree.
2. Learn the implementation of converting a binary tree into a one-way threaded binary tree.

- Introduction: A binary tree is a hierarchical data structure where each node has at most two children: left and right. Traditional binary tree require additiona storage and traversal technique to efficiently nevigate through the tree. A threaded binary tree helps in optimizing tree traversal by utilizing empty child pointers to store in-order sucessors, thus reducing the need for stack or recursive traversal.

A one-way threaded binary tree is special type of binary tree where the right NULL pointers are converted into threads pointing to the in-order suceessor of the node.

- Theory:
1. Threading in binary trees:
- A null right child is replaced with a pointer to its in order successor.
- This allow an efficient in-order traversal without recursion.

2. properties of one-way threaded binary trees:
- Right threads replace null pointer and point to the in-order successor.
- left pointer remain unchanged.
- Traversal is more efficient compared to a traditional binary tree.

3. Advantages:
- Eliminate the need for stack during traversal.
- Reduces space complexity.
- Improves traversal efficiency.

Implementation Details : The given c++ implementation follows these steps:

Node structure: Each node consist of:
data : store the value
left : pointer to the left child.
right : pointer to the right child.
right thread : Boolean flag indicating if right is thread.

Insertion:
- If the right pointer is NULL, it is replaced with a thread pointing to the inorder successor.
- The tree is traversed using standard binary search tree insertion logic.

FOR EDUCATIONAL USE

3. Leftmost function:
find the leftmost node in a subtree

4. Inorder traversal:
- Start from leftmost node and follow in-order traversal using thread
- If a node has a thread, it follows it instead of recursively traversing

• Algorithm Analysis:
1. Time complexity:
- insertion: O(h), where h is height of the tree
- traversal: O(n), as each node is visited once

2. Space complexity:
- O(n) for storing n nodes.
- O(1) additional space for traversal since no recursion is used.

• Conclusion:
A one-way threaded binary tree optimize traversal by replacing NULL right pointers with in-order successor links, reducing space complexity while maintaining efficient traversal operations.

12/02/25

Sundaram
FOR EDUCATIONAL USE

Assignment : 3

- Title : A book consist of chapters, chapter consist of sections and section consist of subsections. Construct a tree and print the nodes. Find the time and space requirement of your method.

- Objectives : To understand concept of tree data structure to understand the concept and feature of OOP.

- Learning outcome : i) To understand concept of class.
  ii) To understand concept and feature of OOP.
  iii) To understand concept of tree data structure.

- Theory : Introduction to Tree :

  A Tree T is set of nodes storing elements such that nodes have parent child relationship that satisfy following if T is not empty, T has special tree called the root that has no parent each node v of T differen than the root has unique parent node w, each node with parent w is child of w.

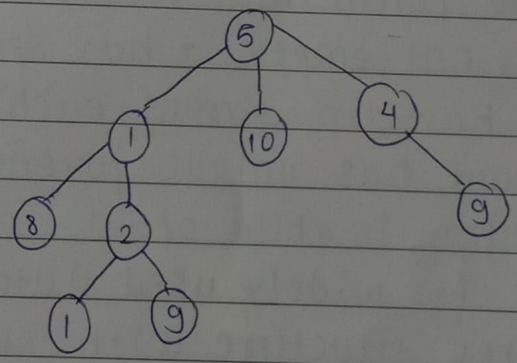
  Tree is widely used data structure that emulate tree structure with set of linked nodes. The tree graphically is represented most commonly as on picture. The circles are nodes and edges are link between them. Trees are usually used to store and represent data in some hierarchical order. The data is stored in nodes, from which the tree is consisted of.

Sundaram
FOR EDUCATIONAL USE

A node may contain a value represent separate data structure or tree of its own. Each node in tree has its own zero or more child nodes, which are one level lower in tree hierarchy. A node that has child is called child's parent node. A node has atmost one parent. A node that has no child is called a leaf and node is of course at bottommost level of the tree.

The height of node is length of longest path to a leaf from that node.

i. The height of tree with no element is 0.
ii. The height of tree with 1 element is 1.
iii. The height of tree with >1 element is equal to 1 + height of its tallest subtree.



- Important Terms

following are the important terms with respect to tree.

1. path - refers to sequence of nodes along the edges of tree.

FOR EDUCATIONAL USE

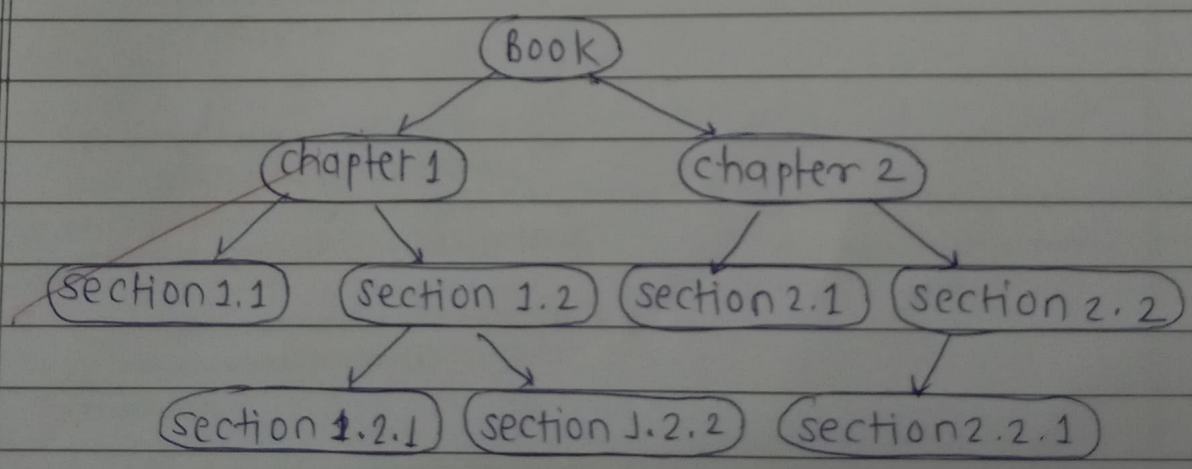
2. Root - The node at top of tree is called root.

3. parent - Any node except the root node has one edge upward to a node called parent.
4. child :- The node below given node connected by its edge downward is called its child node.
5. Leaf - Node which doesn't have any child node.
6. subtree - Represent descendant of node.
7. Visiting - checking value of node when control is on the node.
8. Traversing - passing through node in a specific order
9. Keys - value of node based on which a search operation is to be carried out for a node

- Advantage of tree

Trees are so useful and frequently used, becau they have some very serious adv.

i) Reflect structure relationship in data.
ii) used to represent hierarchies.
iii) provide efficient searching & insertion



Sundaram
FOR EDUCATIONAL USE

- Software Required : g++ / gcc compiler.
- Input : Book name and its no. of sections and subsection along with name.
- output : formation of tree structure for book and its sections.
- conclusion : This program gives us the knowledge tree data structure.

29/01/25