61

Title :- To construct a tree and print the nodes.

Problem statement : A book consists of chapters, chapters consist of subsections construct a tree and print the nodes Find the time and space requirements of your method.

Objectives :- To understand the concept of tree datastructure and understand the features of object oriented programming

Software requirement :- g++ / gcc compiler, 64-bit Fedora, eclipse IDE

Theory :-

Trees :- A tree T is a set of nodes have a parent-child relationship that satisfies the following.

- if T is not empty. T has a special tree called the root that has no parent
- each node 'v' of T different than the root has a unique parent node 'w'; each node with parent w is a child of 'w'
- A tree is non-linear and a hierarchical data structures consisting of collection of nodes that each node of the tree stores a value and a list of references to other nodes

Recursion in trees :-

- T is either empty or consists of a node r (root node) and a probably empty set of trees whose roots are the children of r. Tree is a widely-used data structure that emulates a tree structure with a set of linked nodes, the trees graphically represented most commonly as shown below The circles are nodes and the edges are the links between them

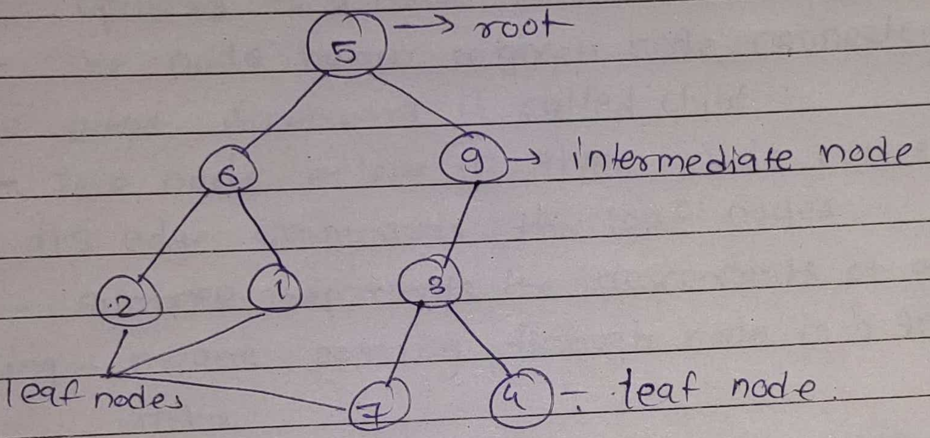


fig. Tree data Structure

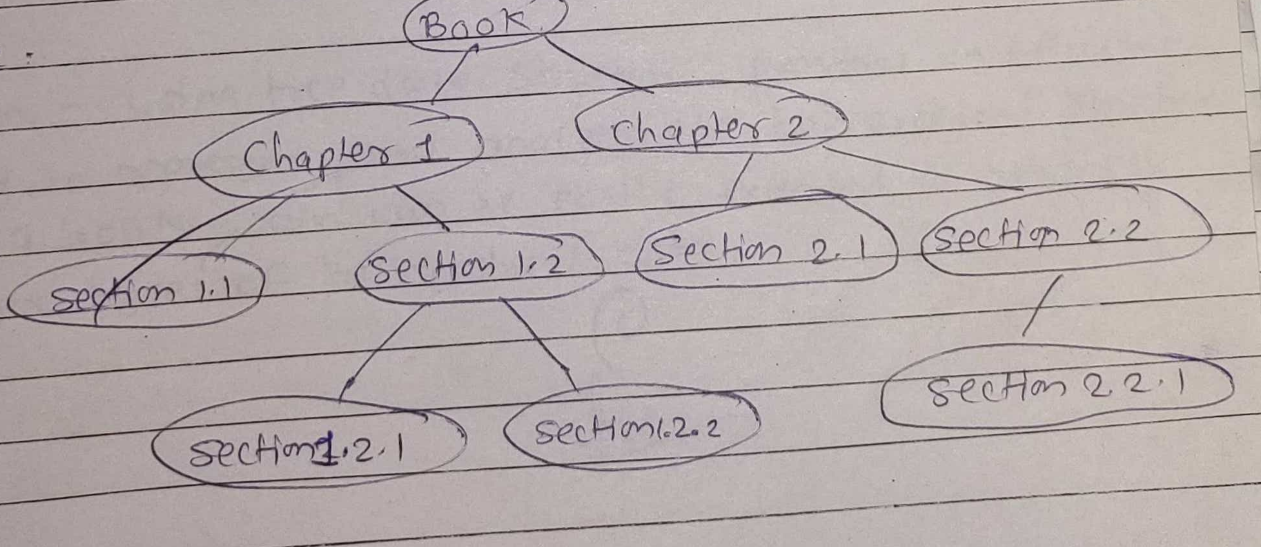
- Trees are usually used to Store and represent data.
In same heirarchical order The data are Stored in the.
nodes from which the tree is considered of.
- A node may contain a value or a condition or represent
a seperate data structure or a tree of its own.
- The topmost node in a tree is called the root node. It
is the node where operations on the tree commonly begins
- All the other nodes can be reached from it by following
edges or links.

Important Terms.

- following are the important terms with respect to tree data structure

1. Path - refers to the sequence of nodes along the edges of the tree
2. Root - The node at the top
3. Parent - Any node except the root node has one edge upward to a node called parent
4. child - The node below a given node connected by its edge downward is called child
5. leaf - The node below a given node connected by its edge downwards. the leaf nodes.
6. subtree - subtree represents the descendants of a node
7. Traversing - means passing through node in a specific order.
8. ~~p~~ Levels - level of an order present the generation of node.
9. Keys - represents a value of a node based an which an operation is carried out.

Example :



- Algorithm

```
1. Start
2. Define an empty tree.
3. Define a node Structure for the tree to hold the data.
4. Read the Book and parse it into chapters, sections
   and subsections.
5. Create a root node for the book and set its value
   to the title of the book.
6. For each chapter, create a child node of the root
   node and set its value to the chapter title.
7. For each section in chapter , create a chid node.
8. for each subsections in section create a child node.
9. Traverse the tree and print the value of each node
10. calculate the time and space complexity of algorithm
11. Stop.
```

- The time complexity of the algorithm is O(n) where n is the total no. of nodes in the tree.
- The space complexity of the algorithm is also O(n) as if it requires storing all nodes in memory

Conclusion :- using tree data structures provides an efficient way to represent and analyze the heirarchical structure of a book, and can be easily extended to handle other types of heirarchical data.