ITS202: Algorithms and Data Structures

Advanced Data Structures

Ms. Sonam Wangmo

Gyalpozhing College of Information Technology
Royal University of Bhutan

October 28, 2020

Trees

Formal Defination

we define a tree T as a set of nodes storing elements such that the nodes have a parent-child relationship that satisfies the following properties:

- If T is nonempty, it has a special node, called the root of T , that has no parent.
- Each node v of T different from the root has a unique parent node w; every node with parent w is a child of w.

Trees

Terms Used in Trees

Two nodes that are children of the same parent are **siblings**. A node v is **external** if v has no children. A node v is **internal** if it has one or more children. External nodes are also known as **leaves**.

Trees

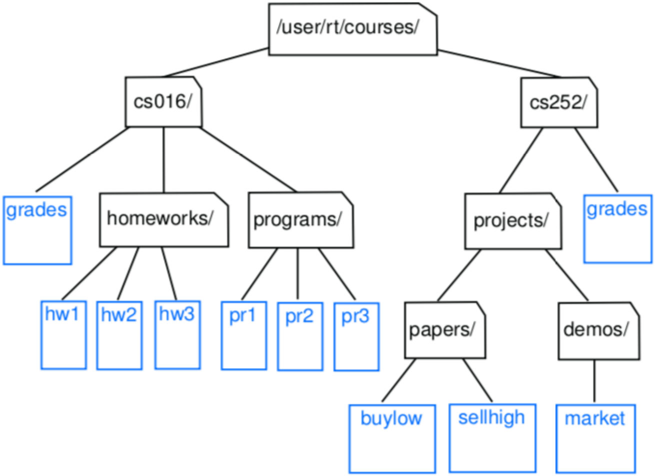

Figure 1: Tree representing a portion of a file system.

A node u is an ancestor of a node v if u is an **ancestor** of the parent of v.
Conversely, we say that a node v is a **descendant** of a node u if u is an ancestor of v.
For example, in Figure 1, cs252/ is an ancestor of papers/, and pr3 is a descendant of cs016/.
The **subtree** of T rooted at a node v is the tree consisting of all the descendants of v in T (including v itself). In Figure 1, the subtree rooted at cs016/ consists of the nodes cs016/, grades, homeworks/, programs/, hw1, hw2, hw3, pr1, pr2, and pr3.

Edges and Paths in Trees

An **edge** of tree T is a pair of nodes (u,v) such that u is the parent of v, or vice versa. A **path** of T is a sequence of nodes such that any two consecutive nodes in the sequence form an edge. // **For example**, the tree in Figure 1 contains the path (cs252/, projects/, demos/, market).

The Tree Abstract Data Type

- getElement():Returns the element stored at this position.
- root():Returns the position of the root of the tree (or null if empty).
- parent(p):Returns the position of the parent of position p (or null if p is the root).
- children(p):Returns an iterable collection containing the children of position p (if any).
- numChildren(p):Returns the number of children of position p.
- isInternal(p):Returns true if position p has at least one child.

The Tree Abstract Data Type

- isExternal(p):Returns true if position p does not have any children.
- isRoot(p):Returns true if position p is the root of the tree.
- size():Returns the number of positions (and hence elements) that are contained in the tree.
- isEmpty():Returns true if the tree does not contain any positions (and thus no elements).
- iterator():Returns an iterator for all elements in the tree (so that the tree itself is Iterable).
- positions():Returns an iterable collection of all positions of the tree.

Computing Depth and Height

Let p be a position within tree T. The depth of p is the number of ancestors of p, other than p itself.
Note that this definition implies that the depth of the root of T is 0. The depth of p can also be recursively defined as follows:

- If p is the root,then the depth of p is 0.
- Otherwise, the depth of p is one plus the depth of the parent of p.

Computing Depth and Height

Formally, we define the height of a position p in a tree T as follows:

- If p is a leaf,then the height of p is 0.
- Otherwise, the height of p is one more than the maximum of the heights of p's children.

Binary Trees

A binary tree is an ordered tree with the following properties:

- Every node has at most two children.
- Each child node is labeled as being either a left child or a right child.
- A left child precedes a right child in the order of children of a node.

Note

The subtree rooted at a left or right child of an internal node v is called a left subtree or right subtree, respectively, of v

The Binary Tree Abstract Data Type

As an abstract data type, a binary tree is a specialization of a tree that supports three additional accessor methods:

- left(p):Returns the position of the left child of p (or null if p has no left child).
- right(p):Returns the position of the right child of p (or null if p has no right child).
- sibling(p):Returns the position of the sibling of p (or null if p has no sibling).

Properties of Binary Trees

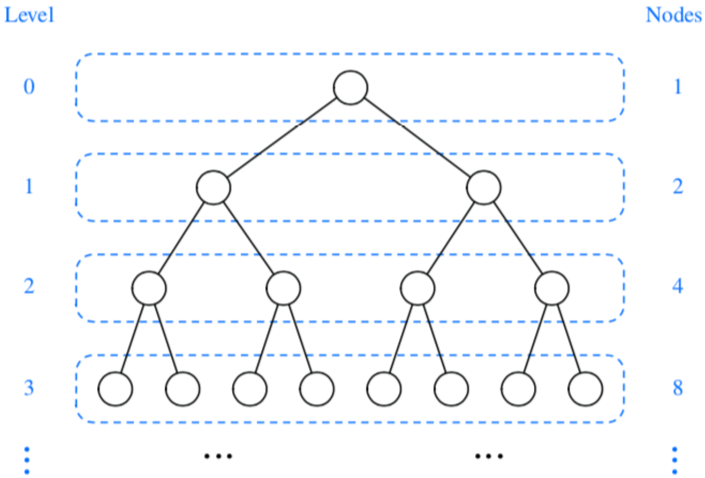

Figure 2: Maximum number of nodes in the levels of a binary tree.

Properties of Binary Trees

Note

We denote the set of all nodes of a tree T at the same depth d as **level** d of T .

In a binary tree, level 0 has at most one node (the root), level 1 has at most two nodes (the children of the root), level 2 has at most four nodes, and so on. In general, level d has at most 2^d nodes.

Tree Traversal Algorithms

A traversal of a tree T is a systematic way of accessing, or "visiting," all the positions of T.

Three way tree traversal of a Binary Tree

1. Inorder Traversal steps
2. Preorder Traversal steps
3. Postorder Traversal steps

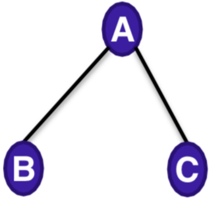

Figure 3: Typical Tree

Tree Traversal Algorithms

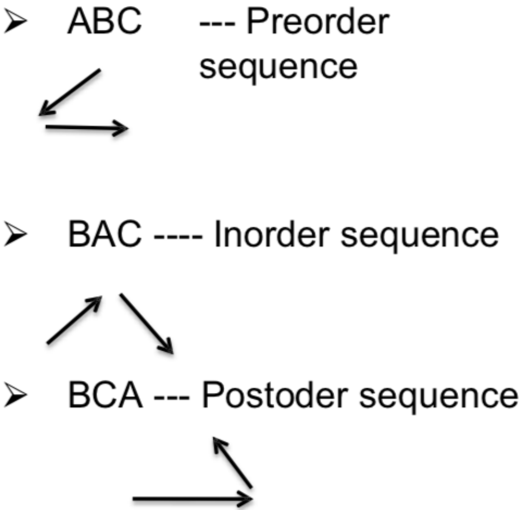

Figure 4: Tree Traversals

Tree Traversal Algorithms

InOrder Traversal Steps

1. Visit the left sub-tree in inorder
2. Visit the root
3. Visit the right sub-tree inorder

PreOrder Traversal Steps

1. Visit the root
2. Visit the left sub-tree in preorder
3. Visit the right sub-tree preorder

PostOrder Traversal Steps

1. Visit the left sub-tree in postorder
2. Visit the right sub-tree postorder
3. Visit the root

Tree Traversals

Figure 5: Example 1

Tree Traversals

Figure 6: Example 2

Tree Traversals

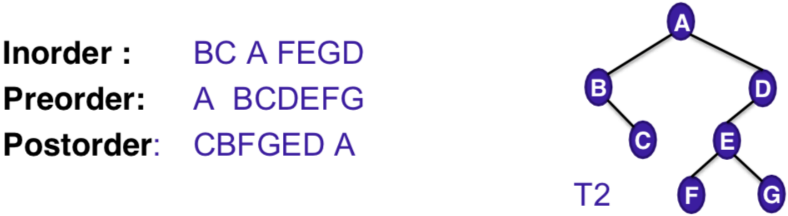

Figure 7: Solution of Example 2