**TREES**

We use trees to represent a hierarchical organization of information.

All previous data organizations we’ve studied are linear (each element can have only 1 predecessor and successor). We have been accessing all elements in a linear sequence.

Trees are nonlinear.

Tree nodes can have multiple successors but only 1 predecessor.

Trees can represent hierarchical organizations of information:

* class hierarchy (in JAVA)
  + in C++ a class can inherit from 2 or more class so it is graph, not a tree
* disk directory and subdirectories
* family tree

Trees are recursive data structures because they can be defined recursively (tree and subtrees)

Many methods to process trees are written recursively.

Tree Terminology and Applications

Text

Description automatically generatedA tree consists of a collection of elements or nodes with each node linked to its successors.

Text

Description automatically generatedGraphical user interface, text, application

Description automatically generatedDiagram

Description automatically generated

dog --> root node

dog is the ancestor of all others.

cat and wolf are siblings because they have same parent.

cat-canine is left subtree of dog, canine is left subtree of cat.

Leaf nodes also are known as external nodes, and nonleaf nodes are known as internal nodes.

A generalization of the parent-child relationship is the ancestor-descendant relationship.

Each node in a tree has EXACTLY 1 parent except for the root node which has no parent.

Diagram

Description automatically generated

The level of a node is determined by its distance from the root (distance from the root plus 1).

The height of a tree is the number of nodes in the longest path from the root node to a leaf node.

Height of the above tree is 3.

Binary Tree

Each element has at most 2 successors.

Can be represented by arrays and by linked data structures

Searching a binary search tree, an ordered tree, is generally more efficient than searching an ordered list (O(logn) versus O(n))

Each node has 2 subtrees.

A set of nodes T is a binary tree if either of the following is true:

* T is empty
* Its root node has 2 subtrees, and , such that and are binary trees
  + = left subtree; = right subtree

Diagram

Description automatically generatedExpression Tree

Each node contains an operator or an operand

Operands are stored in leaf nodes

Parentheses are not stored in the tree because the tree structure dictates the order of operand evaluation

Operators in nodes at higher tree levels are evaluated after operators in nodes at lower tree levels

You cannot guarantee that which + operation will be done first.

Huffman Tree

Represents Huffman codes for characters that might appear in a text file.

As opposed to ASCII or Unicode, Huffman codes uses different numbers of bits to encode letters; more common characters use fewer bits

Many programs that compress files use Huffman codes

Huffman Tree is a binary tree.

Diagram

Description automatically generated

For example, code for r is 0010.

Diagram

Description automatically generatedBinary Search Tree

Binary search trees:

* All elements in the left subtree precede those in the right subtree

A set of nodes T is a binary search tree if either of the following is true:

* T is empty
* If T is not empty, its root node has 2 subtrees, and , such that and are binary search trees and the value in the root node of T is greater than all values in and is less than all values in .

Value at the root is larger than all the values in left subtree and smaller than all the values in right subtree.

In binary search tree, there is no duplication of an element.

A binary search tree never has to be sorted bc its elements always satisfy required order relationships

When new elements are inserted (or removed) properly, the binary search tree maintains its order

In contrast, a sorted array must be expanded whenever new elements are added, and compacted whenever elements are removed (expanding and contracting are both O(n)).

You can always insert a node without shifting the values to the left or right. So performance is quite good compared to sorted arrays.

When searching a BST, each probe has the potential to eliminate half the elements in the tree, so searching generally can be O(logn). In the very rarely worst case, searching is O(n).

Shape, polygon

Description automatically generatedThis is the worst case, searching 1 is O(n).

This is like searching in an array or linked list.

Actually 1 cannot be there, it is less than 2.

Recursive Algorithm for Searching a Binary Tree

Text

Description automatically generated

if tree is empty 🡪 base case

else if target matches the root node’s data 🡪 best case of the general case ( n > 0 )

Full, Perfect, and Complete Binary Trees

A complete binary tree is a perfect binary tree through level n - 1 with some extra leaf nodes at level n (the tree height), all toward the left.  
Except the last level, tree is perfect. Leaf nodes are filled from left to right.

A picture containing clock, clipart

Description automatically generatedA picture containing clock, clipart

Description automatically generatedA picture containing surface

Description automatically generated

A perfect binary tree is a full binary tree of height n with exacty - 1 nodes.  
Every leaves are at same level.  
In this case, n = 3 and - 1 = 7

A full binary tree is a binary tree where all nodes have either 2 children or 0 children (the leaf nodes).

Shape, arrow

Description automatically generatedShape, arrow

Description automatically generated Shape, arrow

Description automatically generated

*Neither full nor complete.*

*Full but not complete.*

*Full but not complete.*

Shape

Description automatically generatedShape

Description automatically generatedShape

Description automatically generated

*Neither full nor complete.*

*Complete but not full.*

*Not full nor complete.*

Shape

Description automatically generated

Shape, arrow

Description automatically generatedShape

Description automatically generated

*BOTH FULL AND COMPLETE*

*Complete but not full.*

*Full but not complete.*

General Trees

Diagram

Description automatically generatedWe do not discuss general trees in this chapter, but nodes of a general tree can have any number of subtrees.

Diagram

Description automatically generated with medium confidenceA general tree can be represented using a binary tree

The left branch of a node is the oldest child, and each right branch is connected to the next younger sibling (if any)

**Tree Traversals**

In tree, we can only access to root node directly.

Often, we want to determine the nodes of a tree and their relationship:

* We can do this by walking through the tree in a prescribed order and visiting the nodes as they are encountered
* This process is called *tree traversal*

Three common kinds of tree traversal: INORDER , PREORDER , POSTORDER

A picture containing surface

Description automatically generatedInorder:

* We traverse all the nodes at the left subtree
* We process the root node
* We traverse all the nodes at the right subtree

0 1 2 3 4 5 6 7 9 10 11 12 13 🡪 You get ordered list

A picture containing surface

Description automatically generatedPreorder:

* We process the root node
* We traverse all the nodes at the left subtree
* We traverse all the nodes at the right subtree

7 1 0 3 2 5 4 6 10 9 12 11 13

A picture containing surface

Description automatically generatedPostorder:

* We traverse all the nodes at the left subtree
* We traverse all the nodes at the right subtree
* We process the root node

0 2 4 6 5 3 1 9 11 13 12 10 7

A picture containing text

Description automatically generated

Running times are same ( ) :

* T(n) = θ(1) if n = 0
* T(n) = θ(1) + T(?)

It is not easy to write a recurrence relation for this algorithm bc we don’t know how many elements are there in right subtree or left subtree.

For each subtree (if there are n nodes, there are n subtrees), we perform a recursive call. There are n recursive calls that process the algorithm in “else” part.

Also there are recursive calls for empty subtrees as well. “if tree is empty” case happens for empty subtrees. If # of nodes = n, # of empty subtrees = n+1 (if n is number of nodes, there are n+1 calls for if case).

Running time of each recursive calls is constant time.

All in all; if there are n nodes, there are 2n+1 calls for the traversal algorithms. 🡪 T(n) = θ(n)

How many activation records are put on the system stack during this algorithm’s run? 🡪 2n+1

What is the largest size (number of activation records) of run-time stack? / How many activation records live at the same time? 🡪 We make the second call after the first call is returned. So any recursive call are made to left or right subtree do not live together at the same time. Answer is “Height+1” (+1 is for the empty subtree).

Visualizing Tree Traversals

You can visualize a tree traversal by imagining a mouse that walks along the edge of the tree

If the mouse always keeps the tree to the left, it will trace a rout known as the Euler tour

The Euler tour is the path traced in blue in the figure below:

Shape

Description automatically generated

Each node is reached 3 times.

A Euler tour is a preorder traversal. The mouse visits each node before traversing its subtrees (shown by the downward pointing arrows).

The sequence in this example is : a b d g e h c f i j

If we record a node as the mouse returns from traversing its left subtree (shown by horizontal black arrows in the figure) we get an inorder traversal

The sequence in this example is : d g b h e a i f j c

If we record each node as the mouse last encounters it, we get a postorder traversal (shown by the upward pointing arrows)

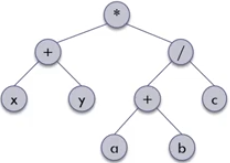
The sequence is : g d h e b i j f c a

Traversals of Binary Search Trees and Expression Trees

Diagram

Description automatically generatedAn inorder traversal of a binary search tree results in the nodes being visited in sequence by increasing data value

canine, cat, dog, wolf

An inorder traversal of an expression tree results in the   
sequence:

x + y \* a + b / c

If we insert parantheses where they belong, we get the   
infix form:

(x + y) \* ((a + b) / c)

---------------------------------------------------------------------------

A postorder traversal of an expression tree results in the  
sequence:

x y + a b + c / \*

This is the postfix or reverse polish form of the expression,   
operators follow operands

---------------------------------------------------------------------------

A preorder traversal of an expression tree results in the   
sequence:

\* + x y / + a b c

This is the prefix or forward polish form of the expression,   
operators precede operands