CS213/293 Data Structure and Algorithms 2023

Lecture 6: Tree

Instructor: Ashutosh Gupta

IITB India

Compile date: 2023-08-20

Topic 6.1

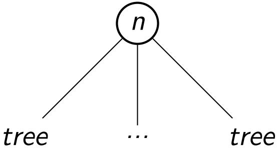
Tree

Tree

Definition 6.1

A tree *is either a node*

or the following structure consisting of a node and a set of children trees.



The above is our first recursive definition.

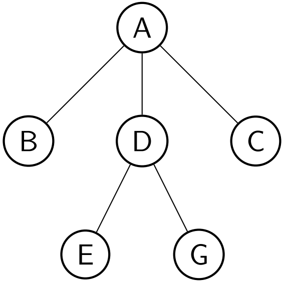
Exercise 6.1

Does the above definition include infinite trees? How would you define an infinite tree?

Example: tree

Example 6.1

An instance of tree.



Some tree terminology(2)

For nodes n_1 and n_2 in a tree T.

Definition 6.2

n_1 is child of n_2 if n_1 is immediately below n_2. We write $n_1 \in children(n_2)$.

Definition 6.3

We say n_2 is parent of n_1 if $n_1 \in children(n_2)$ and write $parent(n_1) = n_2$.
If there is no such n_2, we write $parent(n_1) = \perp$.

Definition 6.4

n_1 is ancestor of n_2 if $n_1 \in parent^*(n_2)$. We write $n_1 \in ancestors(n_2)$.
n_2 is descendant of n_1 if $n_1 \in ancestor(n_2)$. We write $n_1 \in descendants(n_2)$.

Commentary: For a function $f(x)$, we define $f^*(x) = y | y = f(..f(x))$,i.e., the function is applied 0 or more times (informal definition). What would be a mathematically formal definition?

Some tree terminology

Definition 6.5
n_1 and n_2 are siblings if $parent(n_1) = parent(n_2)$.

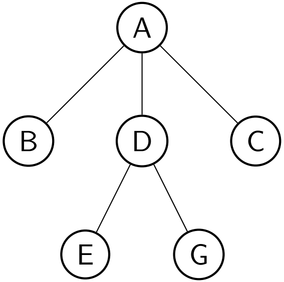
Definition 6.6
n_1 is a leaf if $children(n_1) = \emptyset$.
n_1 is an internal node if $children(n_1) \neq \emptyset$.

Definition 6.7
n_1 is a root if $parent(n_1) = \bot$.

Exercise 6.2
Can the root be an internal node? Can the root be a leaf?

Example: Tree terminology

Example 6.2



B, D, and C are children of A.

D is the parent of G.

A is an ancestor of G and E is a descendant of A.

A is an ancestor of A.

G and E are siblings.

B, E, G, and C are leaves.

A and D are internal nodes.

A is the root.

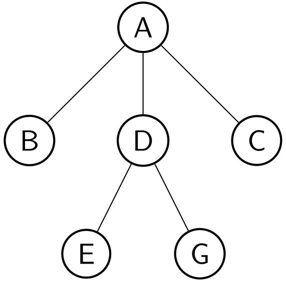
Degree of nodes

Definition 6.8

We define the degree of a node n as follows.

$$degree(n) = |children(n)|$$

Example 6.3



$$dgree(A) = 3$$

$$dgree(B) = 0$$

$$dgree(D) = 2$$

Label of tree

Usually, we store data on the tree nodes.

We define the *label*(n) of a node n as the data stored on the node.

Level/Depth and height of nodes

Definition 6.9

We define the level/depth of a node n as follows.

$$level(n) = \begin{cases} 0 & \text{if } n \text{ is a root} \\ level(n') + 1 & n' \in parent(n) \end{cases}$$

Definition 6.10

We define the height of a node n as follows.

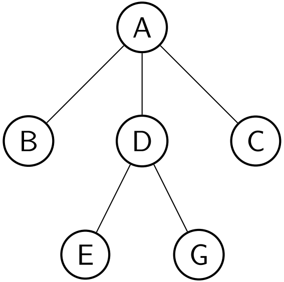
$$height(n) = max(\{height(n') + 1 | n' \in children(n)\} \cup \{0\})$$

Exercise 6.3

Why do we need to take a union with 0 in the definition of height?

Example: Level(Depth) and height of nodes

Example 6.4



$$level(A) = 0$$

$$level(B) = 1$$

$$level(E) = 2$$

$$height(E) = 0$$

$$height(D) = 1$$

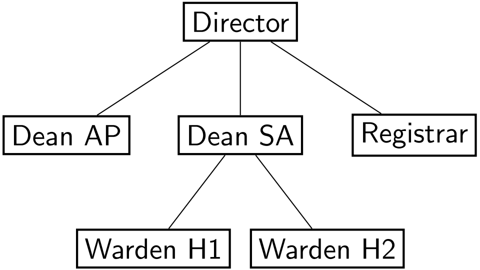
$$\begin{aligned} height(A) &= max(\{height(B) + 1, \\ &\qquad height(D) + 1, \\ &\qquad height(C) + 1\} \cup \{0\}) \\ &= max(\{1, 2, 1\} \cup \{0\}) = 2 \end{aligned}$$

Why do we need trees?

A tree represents a hierarchy.

Example 6.5

- *Organization structure of an organization*

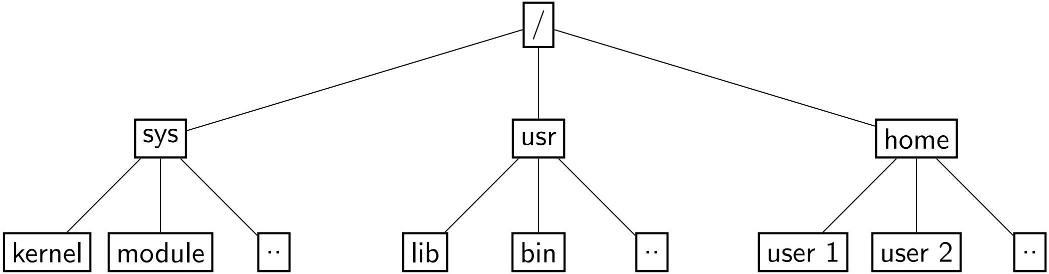


Example: File system

Files are stored in Trees in Linux/Windows.

Example 6.6

Part of a Linux file system.



Topic 6.2

Binary tree

Ordered tree

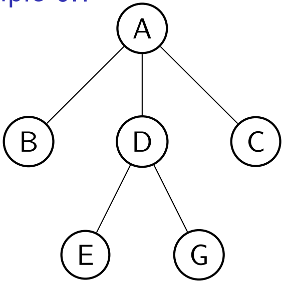
Definition 6.11

A tree is an ordered tree if we assign an order among children.

Definition 6.12

Let n be a node. In an ordered tree, $children(n)$ is a list instead of a set.

Example 6.7



In a tree, we define the children as follows.
$children(A) = \{B, D, C\}$

In an ordered tree, we define the children as follows.
$children(A) = [B, D, C]$

Binary tree

Definition 6.13

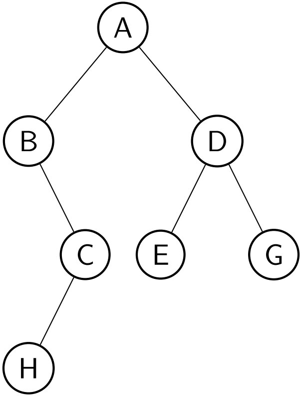
An ordered tree T is a binary tree if $|children(n)| \leq 2$ for each $n \in T$.

We define the left and right child of n as follows.

- *if $chidren(n) = [n_1, n_2]$,*
  - $left(n) = n_1$ *and* $right(n) = n_2$.
- *If $chidren(n) = [n_1]$, n_1 is either left or right child.*
  - $left(n) = n_1$ *and* $right(n) = Null$, *or*
  - $left(n) = Null$ *and* $right(n) = n_1$.
- *If $chidren(n) = []$,*
  - $left(n) = Null$ *and* $right(n) = Null$.

Example: binary tree

Example 6.8

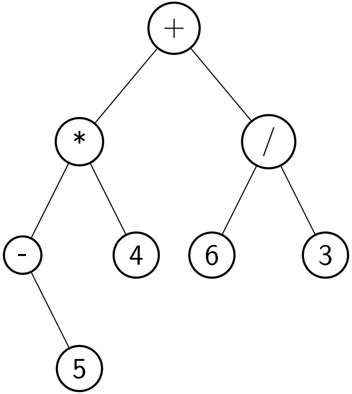


E is the left and G is the right child of D. C is the right child of B. B has no left child.

Usage of binary tree: representing expressions

Example 6.9

Representing mathematical expressions



Exercise 6.4

a. Why do we need an ordered tree?

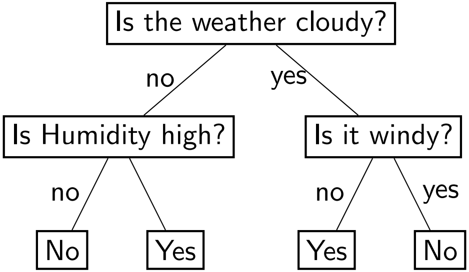
b. How would you evaluate a mathematical expression that is given as a binary tree?

Usage of binary tree: decision trees in AI

Example 6.10

Does one want to play given the weather?

Given the behavior, we may learn the following tree.



Complete binary tree

Definition 6.14

A binary tree is complete if the height of the root is h and every level $i \leq h$ has 2^i nodes.

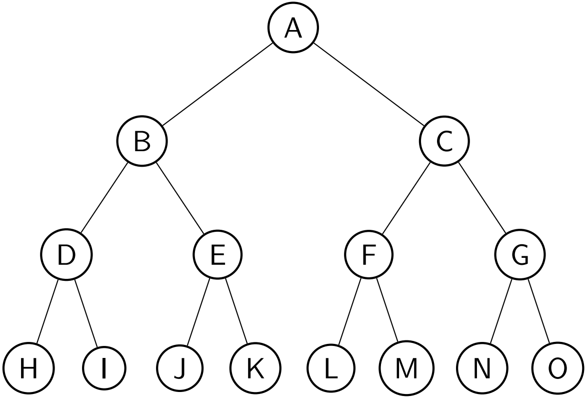
Leaves are only at level h.

The number of leaves $= 2^h$.

Number of internal nodes $= 1 + 2 + ... + 2^{h-1}$
$= 2^h - 1$.

The total number of nodes is $2^{h+1} - 1$.

Example 6.11



Exercise 6.5

a. Prove/Disprove: if no node in the binary tree has a single child, the binary tree is complete.
b. What fraction of nodes are leaves in a complete binary tree?

Maximum and minimum height of a binary tree

Exercise 6.6

Let us suppose there are n nodes in a binary tree.

- *What is the minimum height of the tree?*
- *What is the maximum height of the tree?*

Commentary: For a given height h, a complete binary tree has $2^{h+1} - 1$ nodes. All other binary trees with the height h have fewer nodes. Therefore, $n \leq 2^{h+1} - 1$. Therefore, $log_2 \frac{n+1}{2} \leq h$. The maximum possible height for n nodes is $n - 1$. Therefore, $log_2 \frac{n+1}{2} \leq h \leq n - 1$.

Leaves of binary tree

Theorem 6.1

For a binary tree, $|leaves| \leq 1 + |internal\ nodes|$.

Proof.

We will prove the theorem by induction over the structure of a tree (Recall the recursive definition of a tree).

base case:

We have a single node.
$|leaves| = 1$ and $|\text{internal nodes}| = 0$. Case holds. ...

Commentary: $|A|$ indicates the size of set A.

Leaves of binary tree(2)

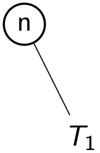
Proof(continued).

induction step:

We have two cases in the induction step: Root has one child or two children.

Case 1:

Let tree T be constructed as follows.



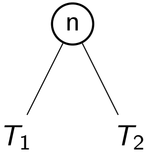
For T_1, let $|leaves| = \ell_1$ and $|\text{internal nodes}| = i_1$.
T has ℓ_1 leaves and $i_1 + 1$ internal nodes.
By the induction hypothesis, $\ell_1 \leq 1 + i_1$.
Therefore, $\ell_1 \leq 1 + i_1 + 1$.
Therefore, $\ell_1 \leq 1 + (i_1 + 1)$. Case holds.

...

Leaves of binary tree(3)

Proof(continued).

Case 2:
Let tree T be constructed as follows.



For T_1, let $|leaves| = \ell_1$ and $|\text{internal nodes}| = i_1$.
For T_2, let $|leaves| = \ell_2$ and $|\text{internal nodes}| = i_2$.
T has $\ell_1 + \ell_2$ leaves and $i_1 + i_2 + 1$ internal nodes.
By induction hypothesis, $\ell_1 \leq 1 + i_1$ and $\ell_2 \leq 1 + i_2$.
Therefore, we have $\ell_1 + \ell_2 \leq 2 + i_1 + i_2$.
Therefore, $\ell_1 + \ell_2 \leq 1 + (i_1 + i_2 + 1)$. Case holds.

Maximum and minimum number of leaves

Let n be the number of nodes in a binary tree T.

Due to the previous theorem, we know $|\text{leaves}| \leq 1 + |\text{internal nodes}|$.

Since $|\text{leaves}| + |\text{internal nodes}| = n$, $|\text{leaves}| \leq 1 + n - |\text{leaves}|$.

$$|\text{leaves}| \leq \frac{(n+1)}{2}.$$

Exercise 6.7

a. When do $|leaves|$ meet the inequality?

b. When is the number of leaves minimum?

Commentary: If T is complete, the number of leaves is $\frac{(n+1)}{2}$.

Topic 6.3

Representing Tree

Container for tree

There is no C++ container for tree.

Trees are the backbone of many abstract data structures.

For some reason, it is not explicitly there.

Exercise 6.8

Why there is no tree container in C++ STL? (Let us ask ChatGPT)

Commentary: I guess that we rarely explicitly need trees in our programming. We usually have higher goals such as stack,queue, set, and map, which may need a tree as an internal data structure, but users need not be exposed. However, there are applications where there is a clear need for trees. For example, the representation of arithmetic expressions. In my programming, whenever I needed a tree. I have implemented it myself.

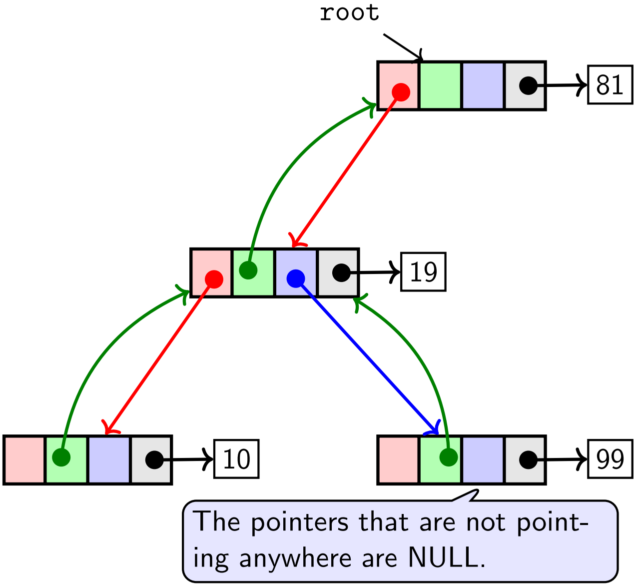
Representation of a binary tree on a computer

Definition 6.15

A binary tree consists of nodes containing four pointer fields.

- *left child*
- *parent*
- *right child*
- *label*

An additional root pointer points to the root of the tree.



Exercise 6.9

Do we need the parent pointer?

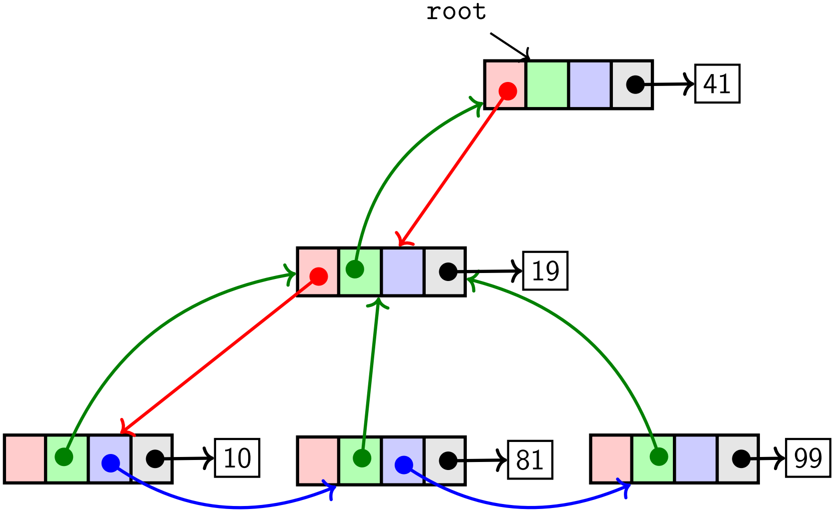
Representation of a tree on a computer

Definition 6.16

A tree consists of nodes containing four pointer fields.

- *first child*
- *parent*
- *next sibling*
- *label*

An additional root pointer points to the root of the tree.



Exercise 6.10

Are we representing an ordered tree or an unordered tree?

Topic 6.4

Problems

Exercise: paths in a tree

Exercise 6.11

Given a tree with a maximum number of children as k. *We give a label between 0 and k-1 to each node with the following simple rules. (i) the root is labeled 0. (ii) For any vertex v, suppose that it has r children, then arbitrarily label the children as 0,..,r-1. This completes the labeling. For such a labeled tree T, and a vertex v, let seq(v) be the labels of the vertices of the path from the root to v. Let* $Seq(T) = \{seq(w) | win T\}$ *be the set of label sequences. What properties does Seq(T) have? If a word w appears what words are guaranteed to appear in Seq(T)? How many times does a word w appear as a prefix of some words in Seq(T)?*

Commentary: From Milind's notes!

Lowest common ancestor(LCA)

Definition 6.17

For two nodes n_1 *and* n_2 *in a tree* T, $LCA(n_1, n_2, T)$ *is a node in* $ancestors(n_1) \cap ancestors(n_2)$ *that has the largest level.*

Exercise 6.12

Write a function that returns $lca(v, w, T)$. *What is the time complexity of the program?*

Exercise: paths in a tree

Exercise 6.13

Given $n \in T$, Let $f(n)$ be a vector, where $f(n)[i]$ is the number of nodes at depth i from n.

- *Give a recursive equation for $f(n)$.*
- *Give a pseudo code to compute the vector $f(root(T))$. How is the time complexity of the program?*

Exercise: mean level

Exercise 6.14

a. Suppose that you are given a binary tree, where, for any node v, the number of children is no more than 2. We want to compute the mean of $ht(v)$, i.e., the mean level of nodes in T. Write a program to compute the mean level.

b. Suppose that we are given the level of all leaves in the tree. Can we compute the mean height? Given a sequence $(n_1, n_2, ..., n_k)$ of the levels of k leaves, is there a binary tree with exactly k leaves at the given levels?

End of Lecture 6