

## 8.2 Rooted Trees & Binary Search Trees

Notebook: Discrete Mathematics [CM1020]

Created: 2019-10-07 2:31 PM

Updated: 2020-01-03 6:03 PM

Author: SUKHJIT MANN

### Cornell Notes

**Topic:**  
8.2 Rooted Trees & Binary Search Trees

Course: BSc Computer Science

Class: Discrete Mathematics-Lecture

Date: January 03, 2020

### Essential Question:

What are rooted trees & binary search trees?

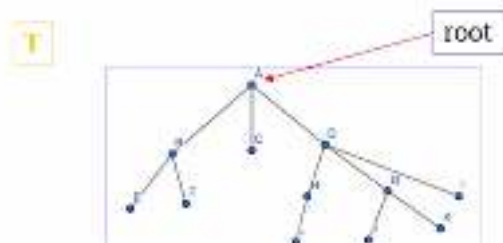
### Questions/Cues:

- What is a rooted tree?
- How is a directed tree represented as a rooted tree?
- What is some terminology associated with rooted trees?
- What are the depth & height in a rooted tree?
- What are some special rooted trees?
- When is m-ary tree considered to be regular?
- What are some properties of m-ary rooted trees?
- What is isomorphism in trees & some properties related to this?
- What is isomorphism in rooted trees?
- What is a binary search tree?
- What is an application of binary search trees?
- What is the height of a binary search tree?
- What is the binary search algorithm?

### Notes

## Definition of rooted trees

A rooted tree is a **directed tree** having one **distinguished** vertex  $r$ , called a root, such that for every vertex  $v$  there is a **directed path** from  $r$  to  $v$ .



# Theorem

A directed tree is represented as a rooted tree **if and only if one vertex** has in-degree **0** whereas **all other vertices** have **in-degree 1**.



## Terminology of rooted trees

A is the **root** of the tree

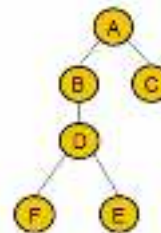
B is called the **parent** of D

E and F are the **children** of D

B and A are **ancestors** of E and F (E and F are **siblings**)

B and D are called **internal** nodes

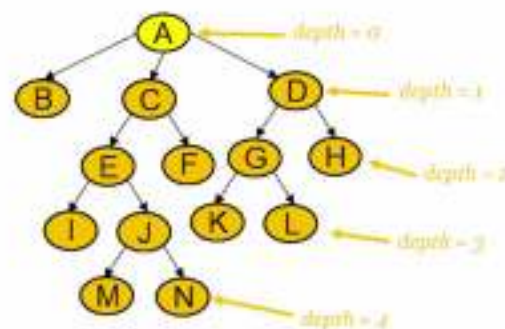
C, E and F are called **external** nodes.



## Depth and height in a tree

The **depth** or **path length** of a node in a tree is the number of edges from the root to that node.

The **height** of a node in a tree is the longest path from that node to a leaf.



The **depth or the height** of a tree is the maximum path length across all its nodes.

The depth (height) of this tree is **4**.

## Special trees

### Binary Trees

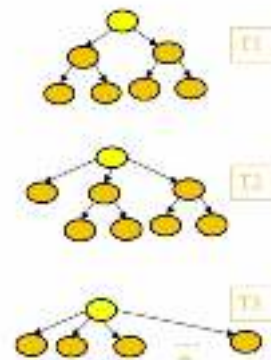
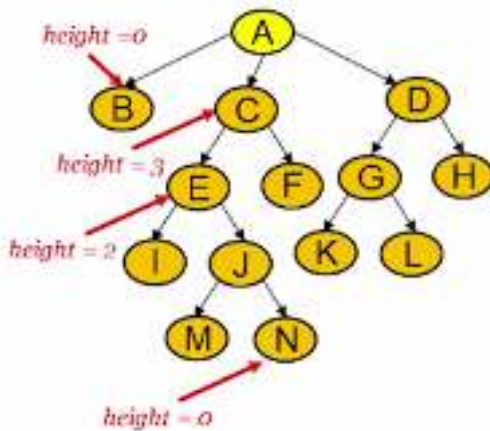
A binary tree is a rooted tree in which every vertex has 2 or fewer children.

### Ternary Trees

A ternary tree is a rooted tree in which every vertex has 3 or fewer children.

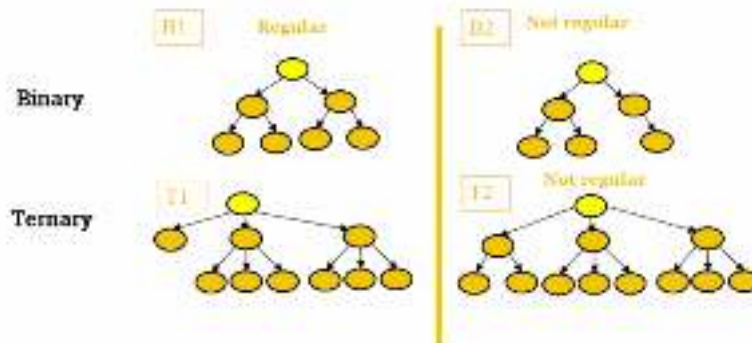
### m-ary Trees

A m-ary tree is a rooted tree in which every vertex has m or fewer children.



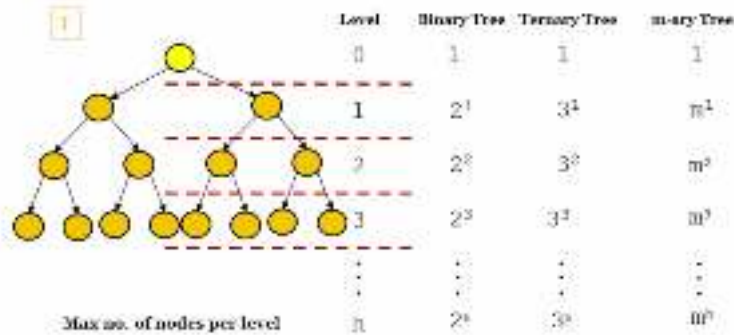
## Regular rooted trees

An **m-ary** tree is **regular** if every one of its **internal nodes** has **exactly m** children.



# Properties

An  $m$ -ary tree has at most  $m^h$  vertices at level  $h$ .



**Note\*\*** Also the maximum number of edges in an  $m$ -ary of  $h$  levels =  $\frac{m^{h+1}-1}{m-1}$

## Isomorphic trees

Two trees  $T_1$  and  $T_2$  are isomorphic if there is a **bijection**:

$$f: V(T_1) \rightarrow V(T_2)$$

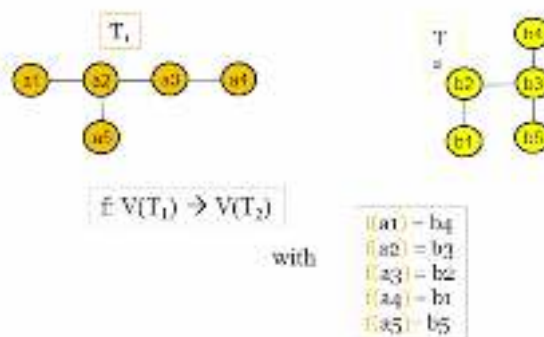
which **preserves adjacency** and **non-adjacency**.

That is, if  $uv$  is in  $E(T_1)$  and  $f(u)f(v)$  is in  $E(T_2)$ .

**Notation:**

$T_1 \cong T_2$  means that  $T_1$  and  $T_2$  are isomorphic.

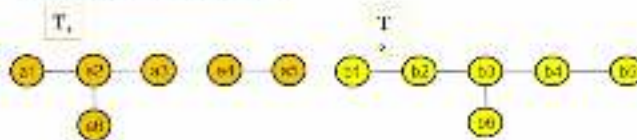
## Example



# Properties

Two trees with **different degree sequences** are **not isomorphic**.

Two trees with the **same degree sequence** are **not necessarily isomorphic**.



$T_1$  and  $T_2$  have the **same degree sequence**: 3, 2, 2, 1, 1.  
 $T_1$  and  $T_2$  are **not isomorphic**.

# Isomorphic rooted trees

Two isomorphic trees are **isomorphic as rooted trees** if and only if there is a **bijection** that maps the **root** of one tree to the root of the other.

# Properties

Isomorphic trees **may** or **may not** be **isomorphic as rooted trees**.



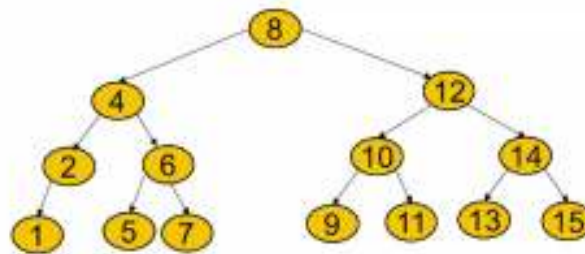
$T_1$  and  $T_2$  are **isomorphic as graphs** but **not isomorphic as rooted trees**.

# Definition

A binary search tree is a **binary tree** in which the vertices are **labelled** with items so that a **label of a vertex is greater than** the labels of all vertices in the **left subtree** of this vertex and **is less than** the labels of all vertices in the **right subtree** of this vertex.



## Example



## Applications

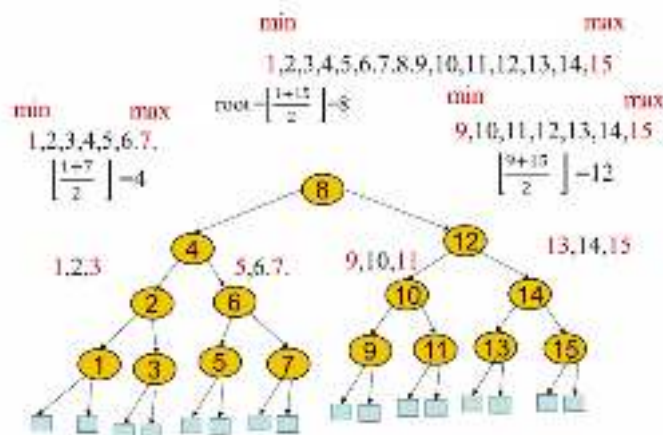
The use applies in the case where we want to **store a modifiable collection** in a **computer's memory** and be able to **search, insert** or **remove** elements from the collection in an efficient way.

**Binary search trees** can be used to solve these kind of **problems**.

## Example

Build a binary search tree to store 15 records and find the height of the this trees.

## Solution



# Height of the tree

Method 1  $2^{h-1} < 1 + N \leq 2^h$

=

$$h-1 < \log_2(1+N) \leq h$$

=

Method 2  $h = \lceil \log_2(N + 1) \rceil$

For example: if  $N=15$  then  $h = 4$

$$2^{4-1} < 1 + 15 \leq 2^4$$

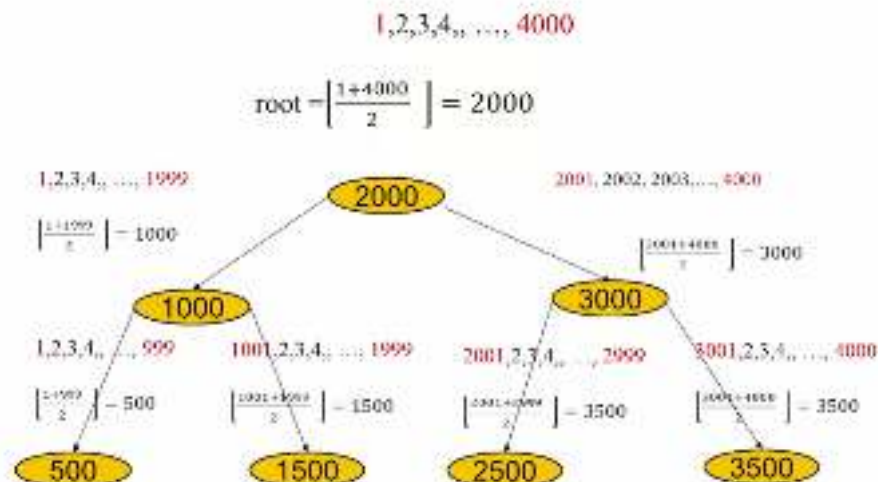
$$h = \lceil \log_2(15 + 1) \rceil = \lceil \log_2(16) \rceil = 4$$

## Exercise

Find the first 3 level of a binary search tree to store 4000 records.

Find the height of this tree.

## Solution



# Height of the tree

## Method 1

$$2^{h-1} < 1 + N \leq 2^h$$

$$2^{12-1} < 1 + 4000 \leq 2^{12}$$

$$h = 12$$

## Method 2

$$h = \lceil \log_2 (N + 1) \rceil$$

$$h = \lceil \log_2 (4000 + 1) \rceil = \lceil \log_2 (4001) \rceil = 12$$

# Binary search algorithm

The algorithm starts by comparing the searched element to the middle term of the list.

The list is then split into two smaller sub-lists of the same size, or where one of these smaller lists has one fewer term than the other.

The search continues by restricting the search to the appropriate sub-list based on the comparison of the searched element and term in the middle.

## Example

Search for **21** in the list of :



## Summary

In this week, we learned what rooted tree is, special rooted trees, properties & terminology associated with rooted trees, what m-ary trees are & what a binary search tree is.