

Trees (I)

Joseph Chuang-Chieh Lin (林莊傑)

Department of Computer Science & Engineering,
National Taiwan Ocean University

Fall 2024

Outline

1 Introduction

- Representation of Trees

2 Binary Trees

- Binary Tree Representations

Outline

1 Introduction

- Representation of Trees

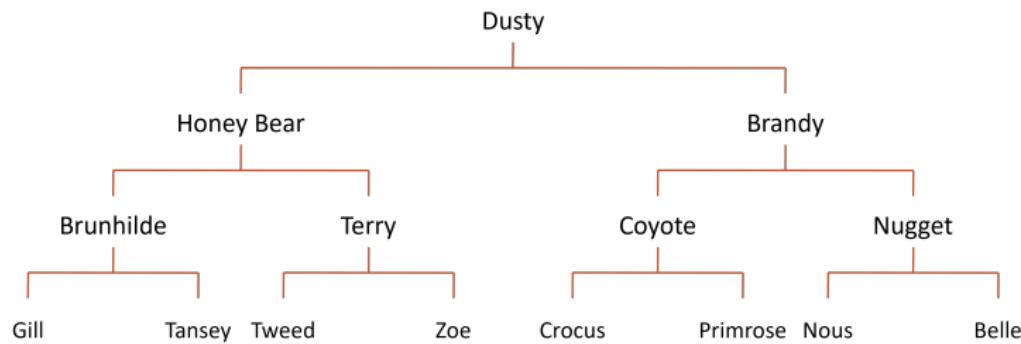
2 Binary Trees

- Binary Tree Representations

Introduction

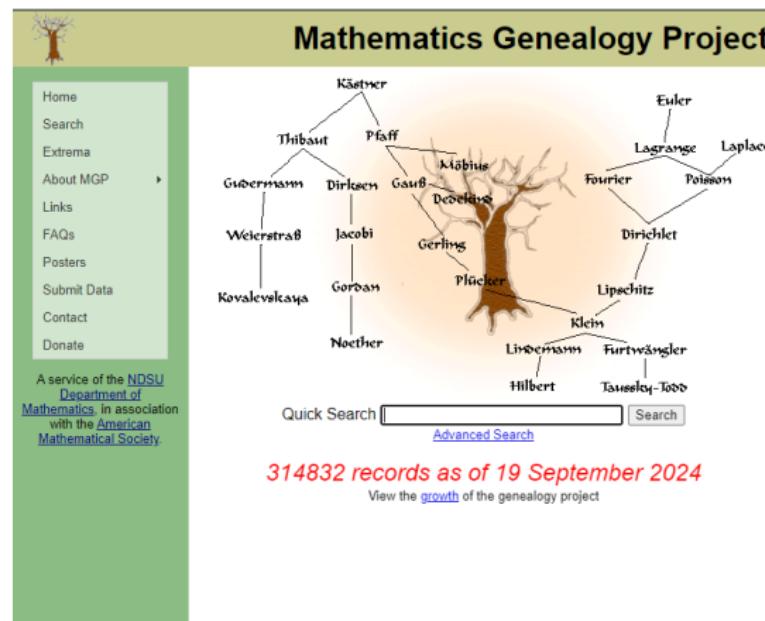
- Intuitively, a **tree** structure organized data in a **hierarchical** manner.

Example: Pedigree Chart



Example: Mathematical Genealogy Project

Figure reference: <https://www.mathgenealogy.org/>



Definitions

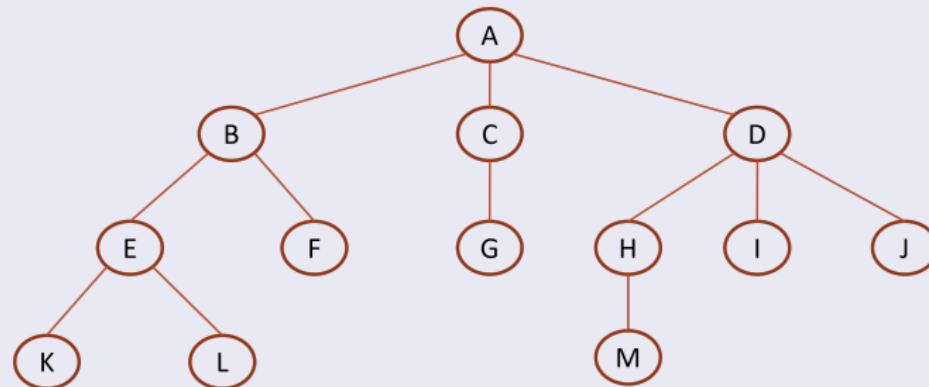
Tree

- A tree is a finite set of one or more nodes such that:
 - There is a specially designated node called **root**.
 - The remaining nodes are partitioned into $n \geq 0$ disjoint sets, T_1, \dots, T_n , where each of these sets is a tree.
 - T_1, \dots, T_n : **subtrees** of the root.

Definitions

Node

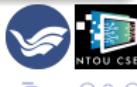
- A node stands for the item of **information** plus the **branches** to other nodes.



Definitions

Degree

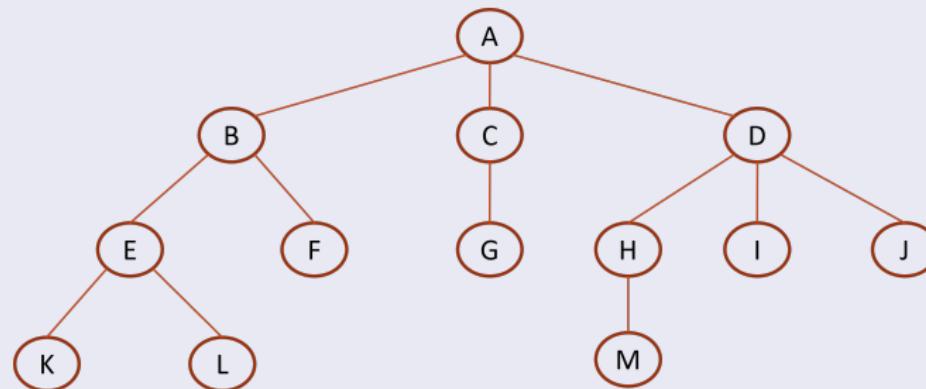
- The number of subtrees of a **node** is called its **degree**.



Definitions

Degree

- The number of subtrees of a node is called its degree.
 - $\deg(A) = 3$, $\deg(C) = 1$, $\deg(F) = 0$.



Definitions

Leaf, children, parent

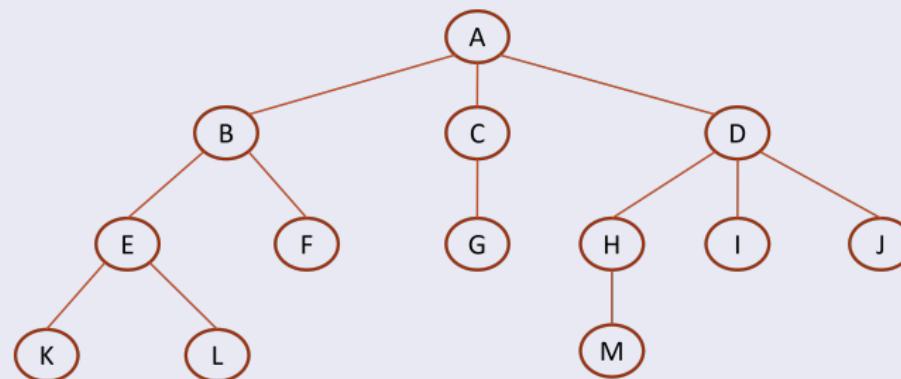
- A node that has degree 0 is called a **leaf** or **terminal**.



Definitions

Leaf, children, parent

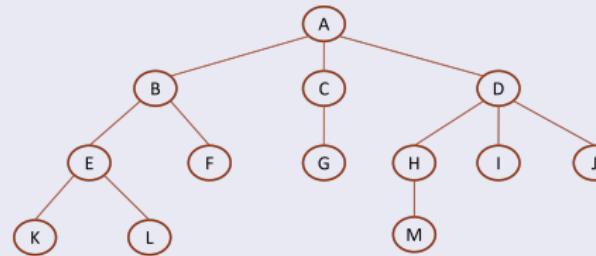
- A node that has degree 0 is called a **leaf** or **terminal**.
- The roots of the subtrees of a node X are the **children** of X . X is the **parent** of its children.



Definition

Siblings, degree, ancestors

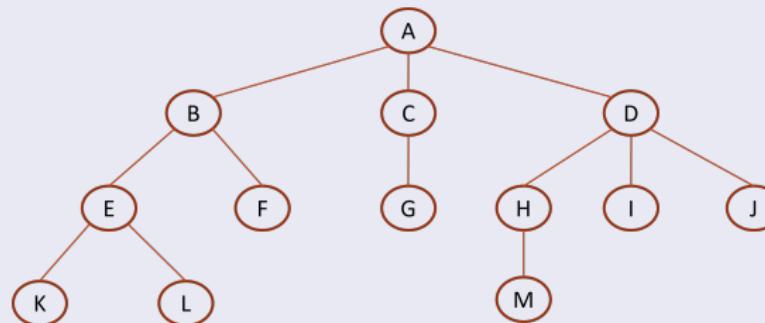
- Children of the same parent are said to be **siblings**.
 - Example: *H, I* and *J* are siblings; *B, C* and *D* are siblings.
- The degree of a **tree** is the **maximum** of the degree of the nodes in the tree.
 - The tree in this example has degree 3.
- The **ancestors** of a node are **all the nodes along the path from the root to that node**.
 - The ancestors of *M* are *A, D*, and *H*.



Definition

Level, height or depth

- The **level** of a node:
 - the root: 1.
 - if a node is at level k , then its children are at level $k + 1$.
 - Example: $\text{level}(A) = 1$, $\text{level}(H) = 3$, $\text{level}(L) = 4$.
- The **height** or **depth** of a tree is defined to be the maximum level of any node in the tree.
 - The depth of the tree in this example is 4.

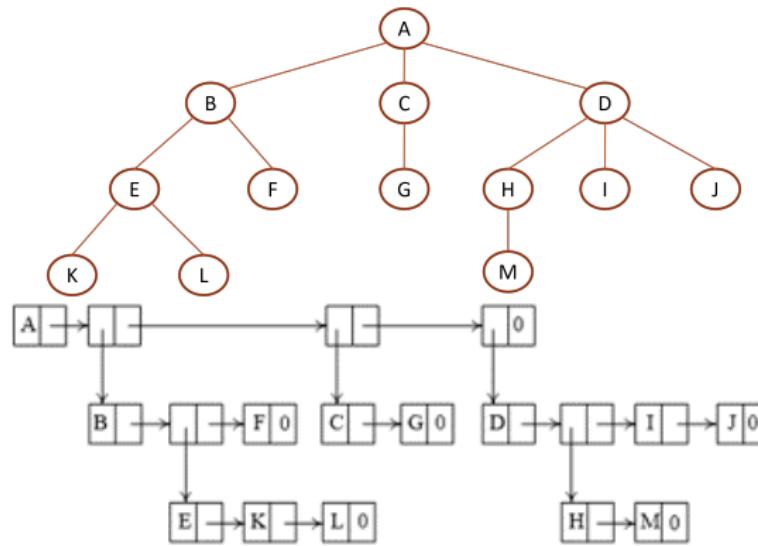


Representation of Trees

- The tree in the example can be written as

$$(A(B(E(K,L),F),C(G),D(H(M),I,J))).$$

- Rule:** root node \rightarrow list of its subtrees.



A Possible Node Structure of a Tree of Degree k

- The degree of each tree node may be different.

A Possible Node Structure of a Tree of Degree k

- The degree of each tree node may be different.
 - we may be tempted to use memory nodes with a varying number of pointer fields.
- However, one only uses nodes of a **fixed size** to represent tree nodes in practice.

data	child 1	child 2	...	child k
------	---------	---------	-----	-----------

A Possible Node Structure of a Tree of Degree k

- The degree of each tree node may be different.
 - we may be tempted to use memory nodes with a varying number of pointer fields.
- However, one only uses nodes of a **fixed size** to represent tree nodes in practice.

data	child 1	child 2	...	child k
------	---------	---------	-----	-----------

- Then, how to choose such a fixed size?

Waste of Space

Lemma 5.1

If T is a k -ary tree (i.e., a tree of degree k) with n nodes ($n \geq 1$), each having a fixed size, then $n(k - 1) + 1$ of the nk child fields are 0.

data	child 1	child 2	...	child k
------	---------	---------	-----	-----------

Proof

- The number of edges of T : $n - 1$
 - Hence, the number of non-zero child fields in T is exactly $n - 1$.

Waste of Space

Lemma 5.1

If T is a k -ary tree (i.e., a tree of degree k) with n nodes ($n \geq 1$), each having a fixed size, then $n(k - 1) + 1$ of the nk child fields are 0.

data	child 1	child 2	...	child k
------	---------	---------	-----	-----------

Proof

- The number of edges of T : $n - 1$
 - Hence, the number of non-zero child fields in T is exactly $n - 1$.
 - The total number of child fields in a k -ary tree with n nodes is nk .



Waste of Space

Lemma 5.1

If T is a k -ary tree (i.e., a tree of degree k) with n nodes ($n \geq 1$), each having a fixed size, then $n(k - 1) + 1$ of the nk child fields are 0.

data	child 1	child 2	...	child k
------	---------	---------	-----	-----------

Proof

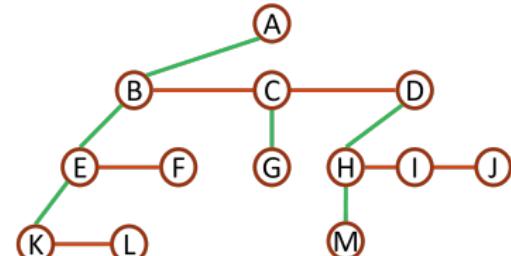
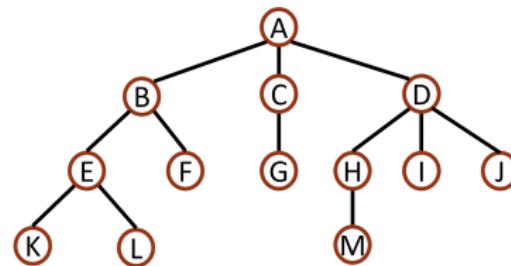
- The number of edges of T : $n - 1$
 - Hence, the number of non-zero child fields in T is exactly $n - 1$.
 - The total number of child fields in a k -ary tree with n nodes is nk .
 - Thus, the number of zero fields is $nk - (n - 1) = n(k - 1) + 1$.



Left Child-Right Sibling Representation

- Every node has ≤ 1 leftmost child and ≤ 1 closest right sibling.
- The **left child field** of each node points to its **leftmost child** (if any)
- The **right sibling field** points to its **closest right sibling** (if any).

data	
left child	right sibling



— left child
— right sibling



Outline

1 Introduction

- Representation of Trees

2 Binary Trees

- Binary Tree Representations

Binary Trees

Binary Trees

A binary tree is a finite set of nodes that

- consists of a root
- two **disjoint binary trees**: the **left** subtree and the **right** subtree.

Trees vs. Binary Trees

Notice

In a binary tree we distinguish between the **order** of the children while in a tree we do not.

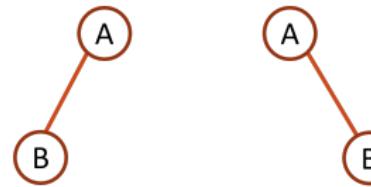
- The following two binary trees are different.
 - the first binary tree has an empty right subtree
 - the second has an empty left subtree.

Trees vs. Binary Trees

Notice

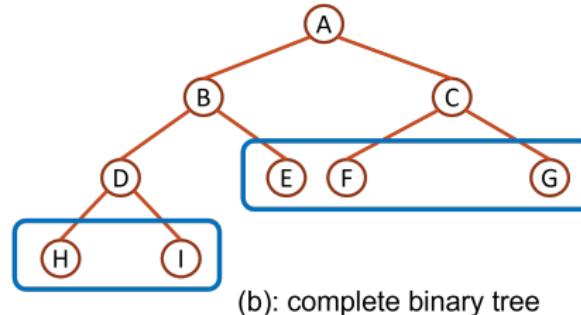
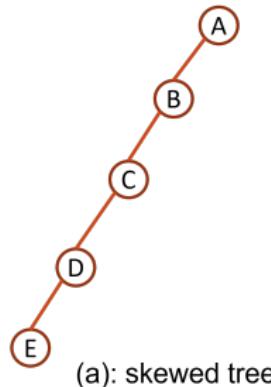
In a binary tree we distinguish between the **order** of the children while in a tree we do not.

- The following two binary trees are different.
 - the first binary tree has an empty right subtree
 - the second has an empty left subtree.



Skew Binary Trees & Complete Binary Trees

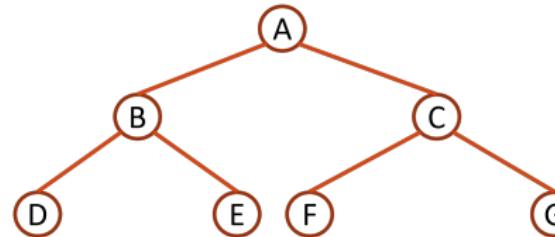
- skew: only left (or right) subtrees for each node
- complete: **all leaf nodes** of these trees are on **two adjacent levels**.



Properties of Binary Trees

Lemma 5.2 [Maximum Number of Nodes]

- The maximum number of nodes on **level** i of a binary tree is 2^{i-1} , for $i \geq 1$.
- The maximum number of nodes in a binary tree of **depth** k is $2^k - 1$, for $k \geq 1$.
- On level 2: 2 nodes; on level 3: 4 nodes.
- Totally $2^3 - 1 = 7$ nodes in the binary tree.



Proof of Lemma 5.2

- Induction Base:
 - The root is the only node on level 1. $2^{1-1} = 2^0 = 1$.
- Induction Hypothesis: Assume that the maximum number of nodes on level $i - 1$ is 2^{i-2} .
- Induction Step:
 - The maximum number of nodes on level $i - 1$ is 2^{i-2} by the induction hypothesis.
 - Since each node in a binary tree has a maximum degree of 2, the maximum number of nodes on level i is $2^{i-2} \cdot 2 = 2^{i-1}$.

Proof of Lemma 5.2

- Induction Base:
 - The root is the only node on level 1. $2^{1-1} = 2^0 = 1$.
- Induction Hypothesis: Assume that the maximum number of nodes on level $i - 1$ is 2^{i-2} .
- Induction Step:
 - The maximum number of nodes on level $i - 1$ is 2^{i-2} by the induction hypothesis.
 - Since each node in a binary tree has a maximum degree of 2, the maximum number of nodes on level i is $2^{i-2} \cdot 2 = 2^{i-1}$.
- The maximum number of nodes in a binary tree of depth k is

$$1 + 2 + 2^2 + \cdots + 2^{k-1} = \sum_{i=1}^{k-1} 2^{i-1} = 2^k - 1.$$

Full Binary Tree

Full Binary Tree

A full binary tree of depth k is a binary tree of depth k having $2^k - 1$ nodes, for $k \geq 0$.

Remark

A binary tree with n nodes and depth k is complete iff its nodes correspond to the nodes numbered from 1 to n in the full binary tree of depth k .

- From Lemma 5.2, we know that

the height of a complete binary tree with n nodes is $\lceil \log_2(n + 1) \rceil$.

Full Binary Tree

Full Binary Tree

A full binary tree of depth k is a binary tree of depth k having $2^k - 1$ nodes, for $k \geq 0$.

Remark

A binary tree with n nodes and depth k is complete iff its nodes correspond to the nodes numbered from 1 to n in the full binary tree of depth k .

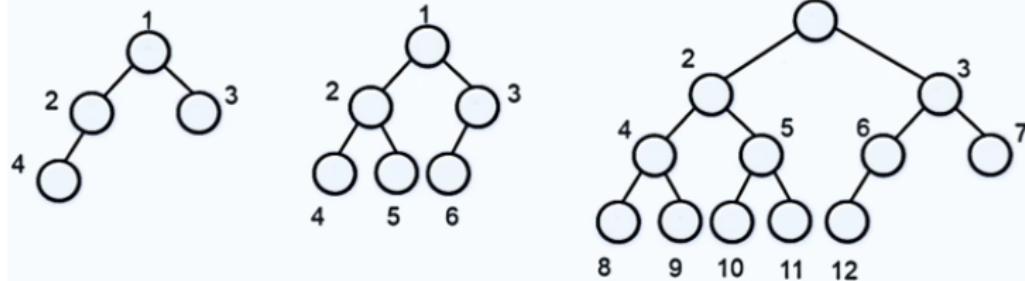
- From Lemma 5.2, we know that

the height of a complete binary tree with n nodes is $\lceil \log_2(n + 1) \rceil$.

- Note:** A complete binary tree is NOT necessarily a full binary tree!



Complete Binary Tree

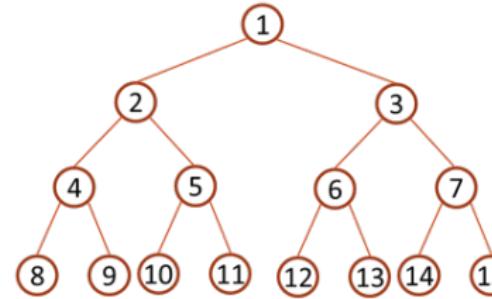


Binary tree Array Representation

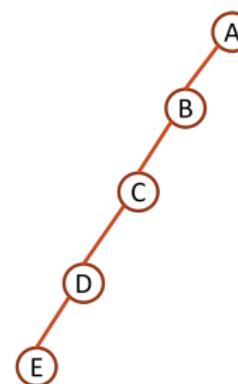
Lemma 5.4

If a complete binary tree with n nodes is represented sequentially, then for any node with index i , $1 \leq i \leq n$, we have

- $\text{parent}(i)$ is at $\lfloor i/2 \rfloor$ if $i \neq 1$. If $i = 1$, i is at root so it has no parent.
- $\text{leftChild}(i)$ is at $2i$ if $2i \leq n$. If $2i > n$, then i has no left child.
- $\text{rightChild}(i)$ is at $2i + 1$ if $2i + 1 \leq n$. If $2i + 1 > n$, then i has no right child.

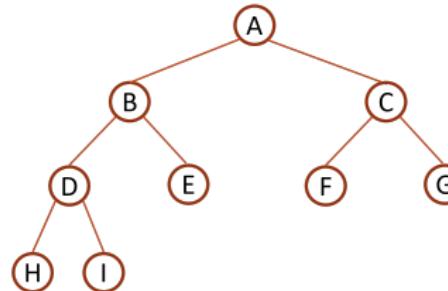


Binary Tree Representation: Examples



<i>tree</i>
[0]
[1] A
[2] B
[3]
[4] C
[5]
[6]
[7]
[8] D
[9]
...
[16] E

Binary Tree Representation: Examples



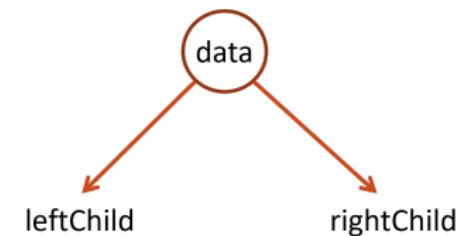
<i>tree</i>
[0]
A
B
C
D
E
F
G
H
I

Drawbacks of the Array Representation

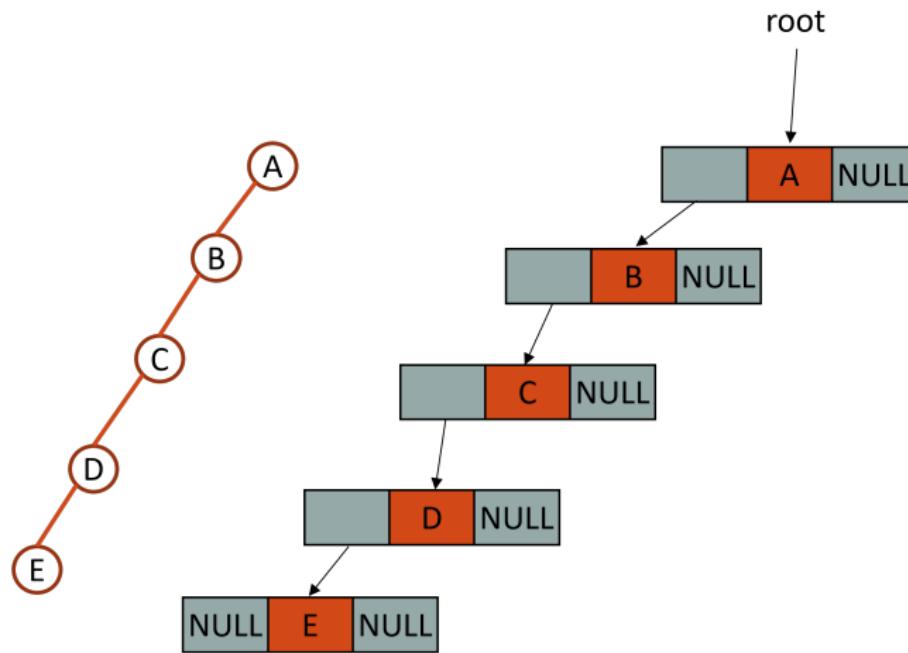
- Waste memory space for most binary trees.
- In the worst case, a skewed tree of depth k requires $2^k - 1$ spaces.
 - Only k spaces is occupied.
- Insertion or deletion of nodes from the middle of a tree requires the movement of potentially many nodes.

Try Linked List Representation

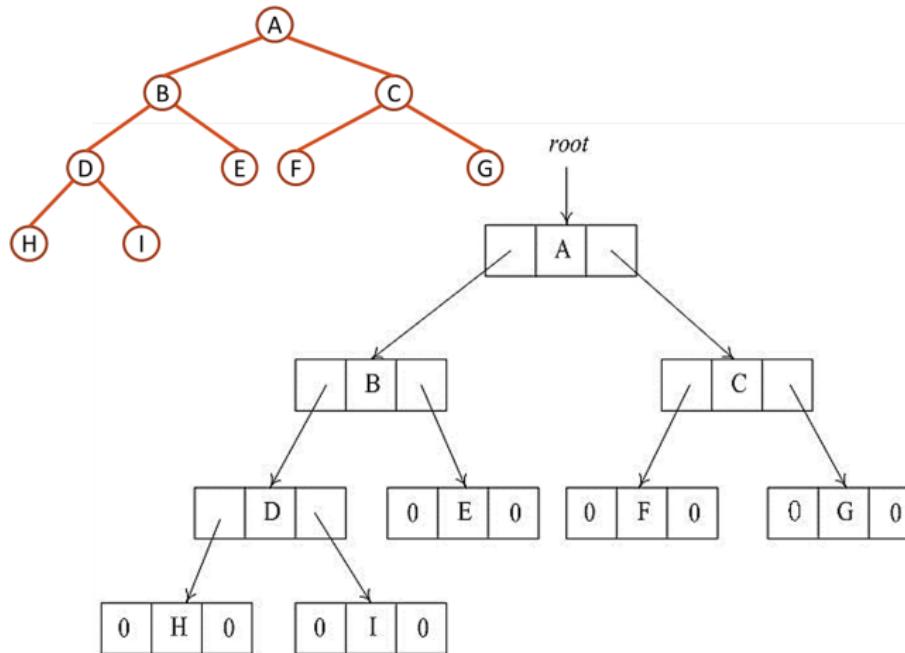
```
typedef struct node *treePointer;
typedef struct node {
    int data;
    treePointer leftChild, rightChild;
};
```



Example



Example



Discussions