fit@hcmus

Tree Structures

1

fit@hcmus

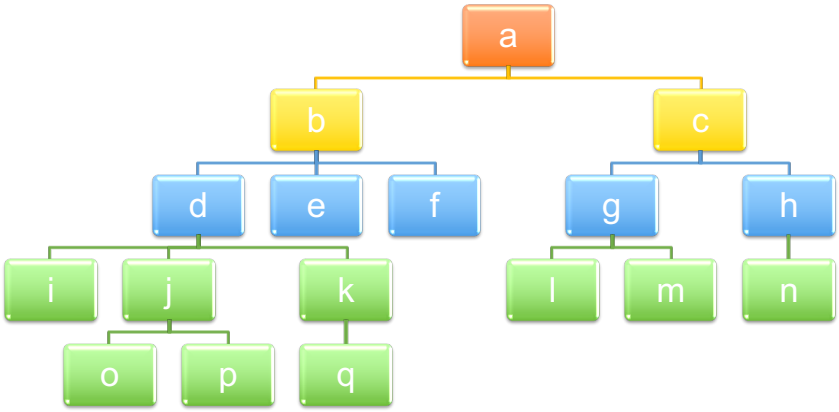
Contents

- Terminologies
- Tree traversals
- Tree representation
- Binary tree
- Binary search tree
- AVL tree
- 2-3 tree, 2-3-4 tree

2

fit@hcmus

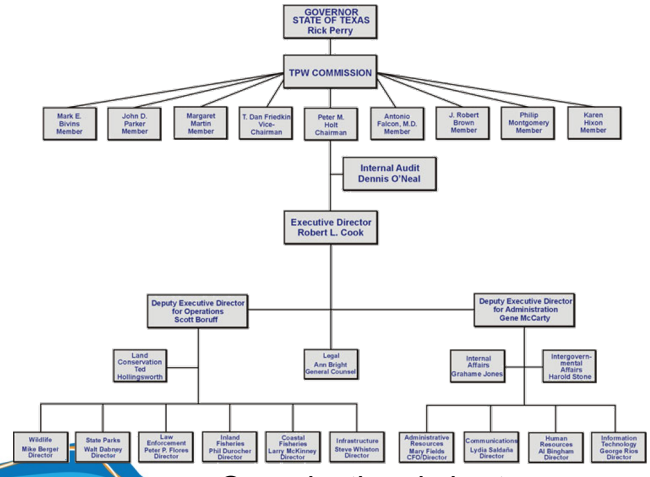
Some Examples



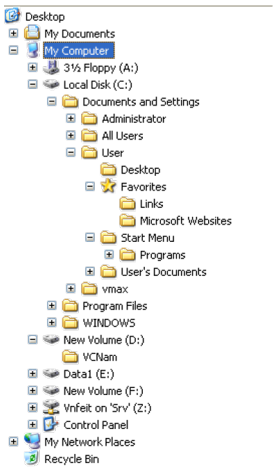
3

fit@hcmus

Some Examples



Organizational chart



Directory tree

4

fit@hcmus

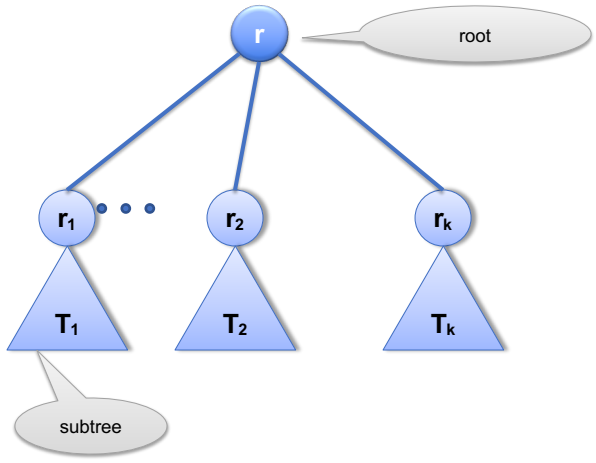
Trees

- Used to represent **relationships**
- ***hierarchical*** in nature
  - "Parent-child" relationship exists between nodes in tree.
  - Generalized to ancestor and descendant
  - Lines between the **nodes** are called **edges**
- A **subtree** in a tree is any node in the tree together with all of its descendants

5

fit@hcmus

Trees



6

fit@hcmus

Terminologies

- node: an item/element in a tree.
- parent (of node *n*): The node **directly above** node *n* in the tree.
- child (of node *n*): The node **directly below** node *n* in the tree.
- root: The only node in the tree with no parent.
- leaf: A node with no children.
- path: A sequence of nodes and edges connecting a nodes with the nodes below it.

7

fit@hcmus

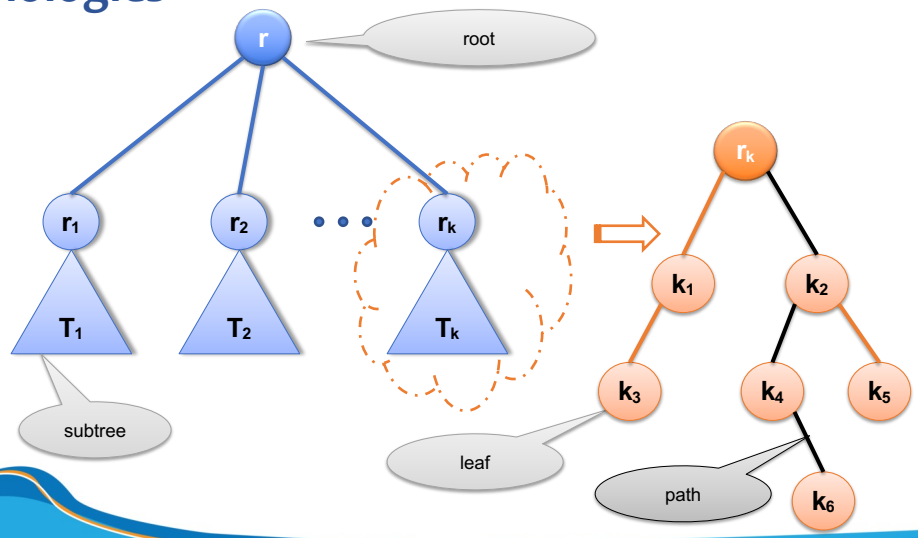
Terminologies

- siblings: Nodes with common parent.
- ancestor (of node *n*): a node on the path from the root to *n*.
- descendant (of node *n*): a node on the path from node *n* to a leaf.
- subtree (of node *n*): A tree that consists of a child (if any) of *n* and the child's descendants.

8

fit@hcmus

Terminologies



9

fit@hcmus

Terminologies

- degree/order
  - Order of node *n*: number of children of node *n*.
  - Order of a tree: the maximum order of nodes in that tree.
- depth/level (of node *n)*
  - If *n* is the root of *T* , it is at level 1.
  - If *n* is not the root of *T* , its level is 1 greater than the level of its parent.

```
if node n is root:
    level(n) = 1
Otherwise:
    level(n) = 1 + level(parent(n))
```


10

fit@hcmus

Terminologies

- Height of tree: number of nodes in the longest path from the root to a leaf.

- Height of a tree T in terms of the levels of its nodes
  - If T is empty, its height is 0.
  - If T is not empty, its height is equal to the maximum level of its nodes.

11

fit@hcmus

Terminologies

- Height of tree *T*:

```
if T is empty:
   height(T) = 0
Otherwise:
   height (T) = max{level(Ni)}, Ni∈T
```

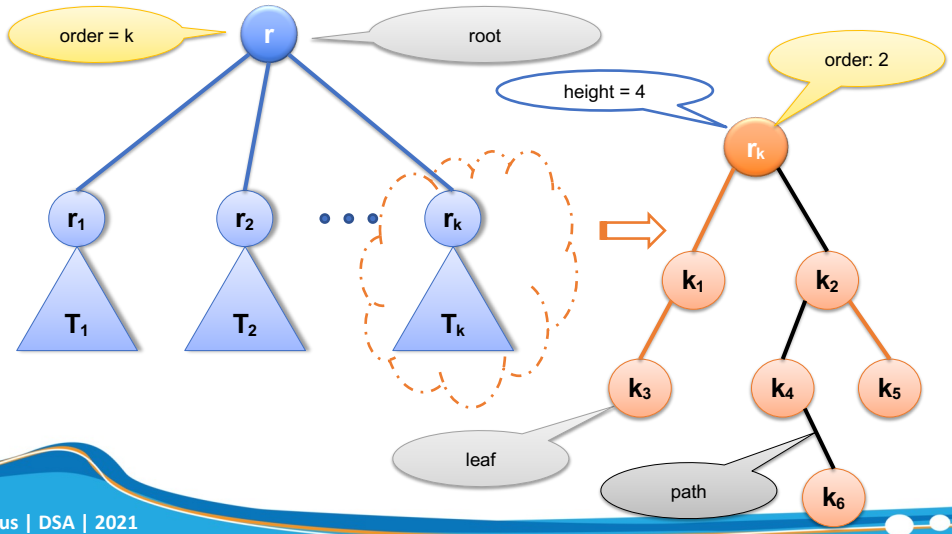
- Height of tree *T*:

```
if T is empty:
   height(T) = 0
Otherwise:
   height(T) = 1 + max{height(Ti)}, Ti is a subtree of T
```


12

fit@hcmus

Terminologies



13

fit@hcmus

Kinds of Trees

14

fit@hcmus

General Tree

- Set T of one or more nodes such that T is partitioned into disjoint subsets
  - A single node r , the root
  - Sets that are general trees, called subtrees of r

15

fit@hcmus

n-ary tree

- set T of nodes that is either empty or partitioned into disjoint subsets:
  - A single node r , the root
  - n possibly empty sets that are n-ary subtrees of r

16

fit@hcmus

Binary tree

- Set T of nodes that is either empty or partitioned into disjoint subsets
  - Single node r , the root
  - Two possibly empty sets that are binary trees, called left and right subtrees of r

17

fit@hcmus

Traversals

18

fit@hcmus

Traversal

- Visit each node in a tree **exactly once**.
- Many operations need using tree traversals.
- The basic tree traversals:
  - Pre-order
  - In-order
  - Post-order

19

fit@hcmus

Pre-order Traversal

```
PreOrder(root)
{
   if root is empty
      Do_nothing;
   Visit root; //Print, Add, ...
   //Traverse every Child_i.
   PreOrder(Child_0);
   PreOrder(Child_1);
   …
   PreOrder(Child_k-1);
}
```


20

fit@hcmus

Post-order Traversal

```
PostOrder(root)
{
   if root is empty
       Do_nothing;
   //Traverse every Child_i
   PostOrder(Child_0);
   PostOrder(Child_1);
   …
   PostOrder(Child_k-1);
   Visit at root; //Print, Add, ...
}
```


21

fit@hcmus

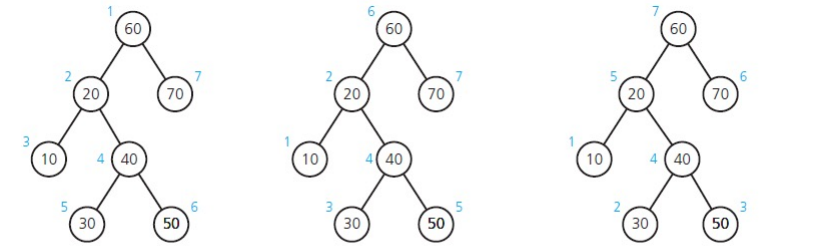
In-order Traversal

```
InOrder(root)
{
   if root is empty
       Do_nothing;
   //Traverse the child at the first position
   InOrder(Child_0);
   Visit at root;
   //Traverse other children
   InOrder(Child_1);
   InOrder(Child_2);
   …
   InOrder(Child_k-1);
}
```


22

fit@hcmus

Traversals



(a) Preorder: 60, 20, 10, 40, 30, 50, 70 (b) Inorder: 10, 20, 30, 40, 50, 60, 70 (c) Postorder: 10, 30, 50, 40, 20, 70, 60

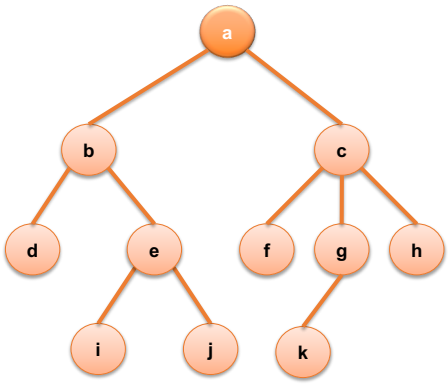
(Numbers beside nodes indicate traversal order.)

23

fit@hcmus

Examples

- Pre-order
- In-order
- Post-order



24

fit@hcmus

Examples

Pre-order

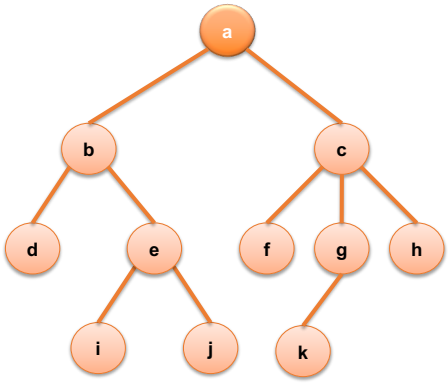
- *a b d e i j c f g k h*

In-order

- d b i e j a f c k g h

Post-order

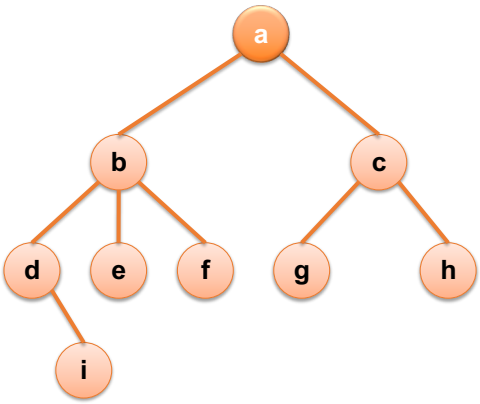
- *d i j e b f k g h c a*



25

fit@hcmus

Examples



26

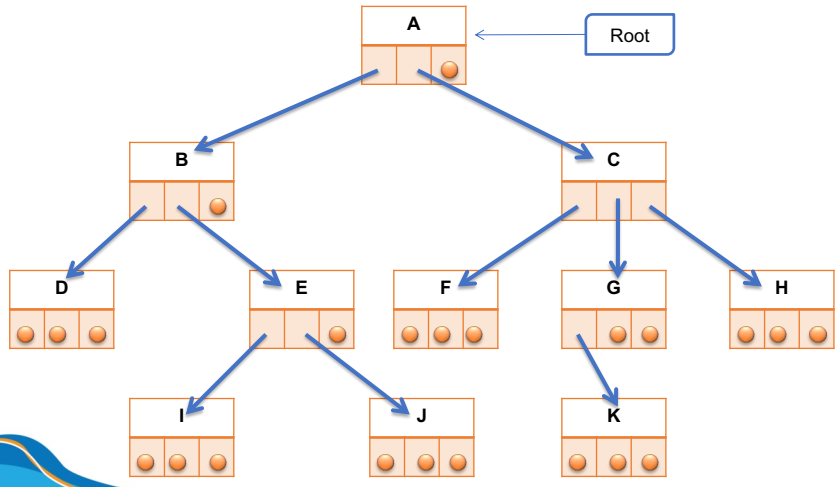
fit@hcmus

Tree Representation

27

fit@hcmus

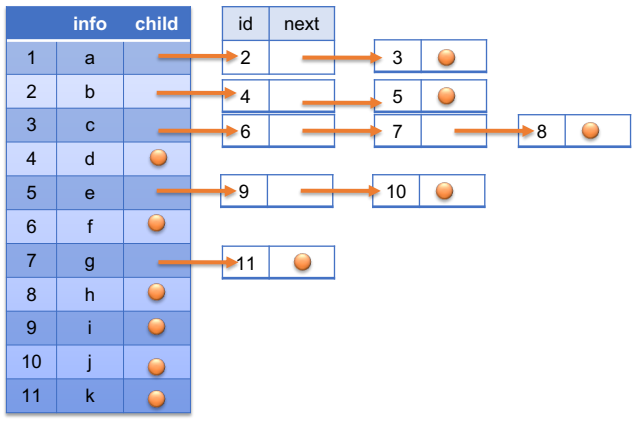
Tree Representation

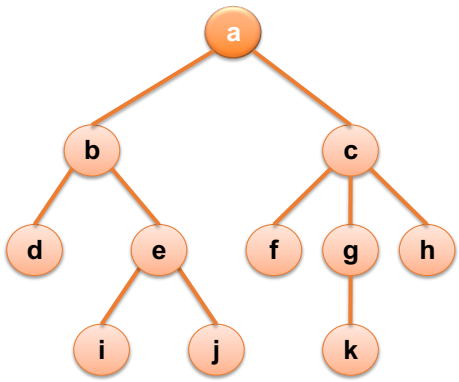


28

fit@hcmus

Tree Representation



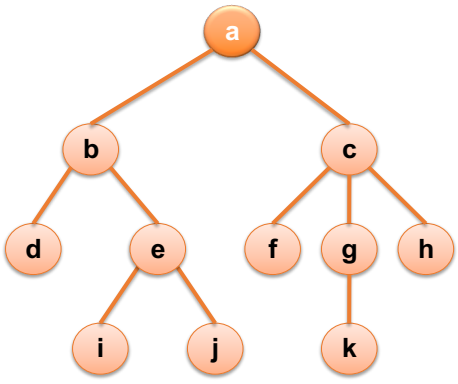


29

fit@hcmus

Tree Representation

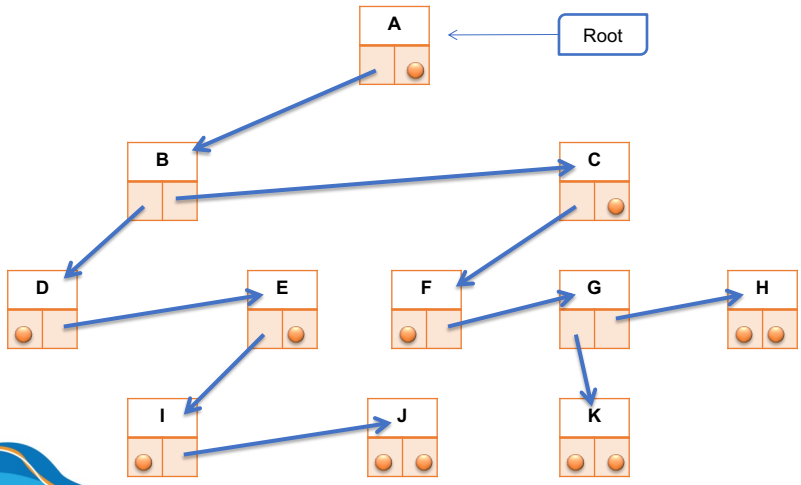
	Info	Eldest Child	Next Sibling
1	a	2	0
2	b	4	3
3	c	6	0
4	d	0	5
5	e	9	0
6	f	0	7
7	g	11	8
8	h	0	0
9	i	0	10
10	j	0	0
11	k	0	0



30

fit@hcmus

Tree Representation

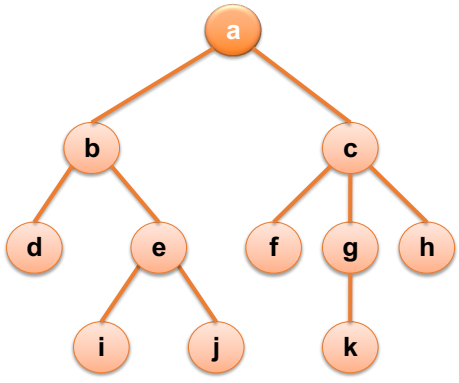


31

fit@hcmus

Tree Representation

	Info	Parent
1	a	0
2	b	1
3	c	1
4	d	2
5	e	2
6	f	3
7	g	3
8	h	3
9	i	5
10	j	5
11	k	7



32