Introduction to Trees

Class: 4005-800-1
Dr. Leon Reznik

Victor Moudgil
3/29/05

1

Session Goals

- Introduction
- What is a Tree?
  - Trees as Models.
  - Definition of Tree.
  - m-ary trees.
- Binary search tree?
- Operations of binary tree's?
  - Traversal.
  - Min, Max
  - Insertion, Deletion
  - Successor, Predecessor
- Variations of Binary Trees
- References

2

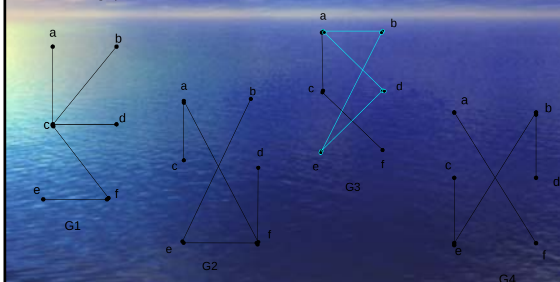
Trees as Models

- Tree of porfery??
  - -Purported to be a the first model in AI.
- Computer File Systems.
- Decision trees.
- Taxonomy.

3

Def of Tree:

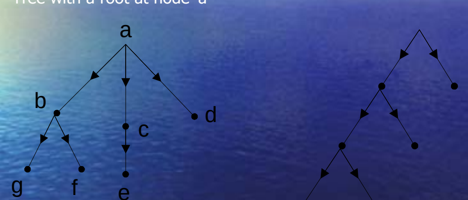
A *connected undirected graph* that contains *no simple circuits*.
G1 and G2 are Trees – Connected with no simple circuts.
G3 is not graph since e,b,a,d,e is a simple circuit
G4 is not a graph since it is not connected



4

Rooted Tree: A rooted tree is one where one vertex has been defined as the root and all other edges are directed away from it.

-Tree with a root at node 'a'



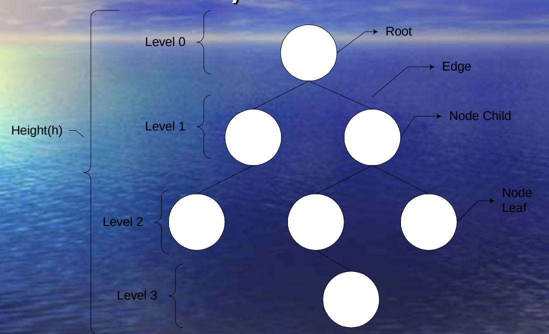
5

m-ary tree

- A rooted tree is called a *m-ary* tree in the case where every internal vertex has no more than *m* children.
- The tree is referred to as a full *m-ary* tree in the event every internal vertex has exactly *m* children.
- A *m-ary* tree where m = 2 the tree is referred to as a Binary Tree.

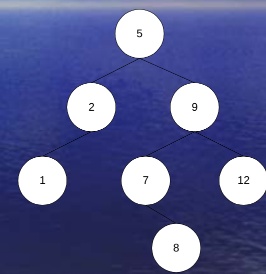
6

Review Binary Tree.



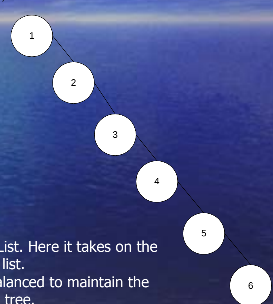
Building a Binary tree

- Add values 5, 9, 7, 2, 12, 1,8 to a Binary Tree.
- 1) Start Root with 5
- 2) Next digit is 9; Larger values on the right
- 3) Next digit is 7; Since 7 is larger than 5 and less than 9 it is placed to the left of 9.
- 4) The next digit is 2; it is placed on the left branch to 5
- 5) Next digit is 12 so placed to the right of 9
- 6) Next is 1 so placed to the left of branch 2
- 7) 8 is placed to the right of 7



Worst case Binary Tree.

1,2,3,4,5,6



-Essentially a Linked List. Here it takes on the properties of a linked list.
-Need to keep tree balanced to maintain the properties of a Binary tree.

Binary Tree Rep.

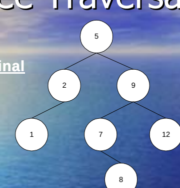
- Each node in a Binary tree is a object
  - Has a key field and possibly satellite data (ex.max val)
  - Each node contains fields for:
    - Left
    - Right
    - Pointer to Parent
    - If data for a particular field is missing than field contains the value NIL.

Operations on Binary Trees.

- Binary Trees support many operations.
  - Search
  - Minimum, Maximum
  - Predecessor, Successor
  - Insert, delete
- Tree Traversal
  - Pre-order
  - In-order
  - Post-order

Tree Traversal:

Original



Pre order traversal:

In Pre-order the contents of the root node are printed first, followed by left subtree and finally the right subtree. 5,2,1,9,7,8,12

In order traversal:

In-order the lowest value are printed first, and then increasing in value as it traverses the tree . 1,2,5,7,8,9,12

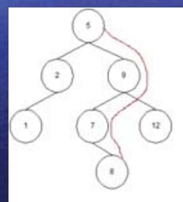
Post order traversal:

Post-order the contents of the left subtree are printed first, followed by right subtree and finally the root node. 1,2,8,7,12,9,5

It takes Θ(n) time to walk an n-node binary tree. After the initial call each node is called recursively twice for each node once for the left and once for the right.

Searching a Binary Tree

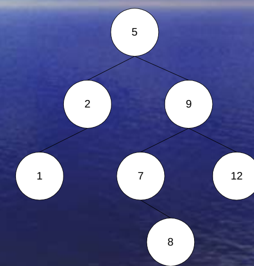
-Searching for a node.
- Given a pointer to the root of the tree
- Given a key k
- A tree Search either returns a pointer to a node with key k or returns a NIL if the key does not exist.
- To search for the Key(8) you would take the path 5->9->7->8.



-Running time is *O(h)*

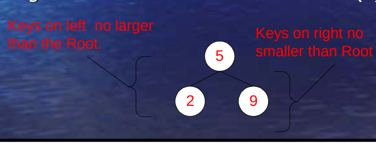
Min., Max

- Minimum.
  - Found by following the *Left* child pointers from the root until a NIL is encountered.
- Maximum.
  - Found by following the *Right* child pointers from the root until a NIL is encountered.
- Running time is *O(h)*



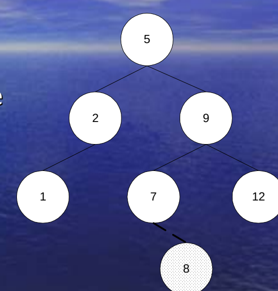
Insertion and deletion

- The Binary tree is modified to reflect the change but the Binary-search-tree property at the conclusion of the insertion or deletion is maintained.
- Binary-Search-tree property.
  - Let x be a node in a binary search tree.
    If y is a node in the left subtree of x, then key[y] < key[x].
    If y is a node in the right subtree of x, then key[x] < key[y].
- Running Time for both insert and delete is O(h).



Insertion

- Straight forward.
- Locate position in tree for the key value to be inserted.
- Locate the nil value for the insertion.
- Place the input item here.
- *Running time is O(h)*



DELETION

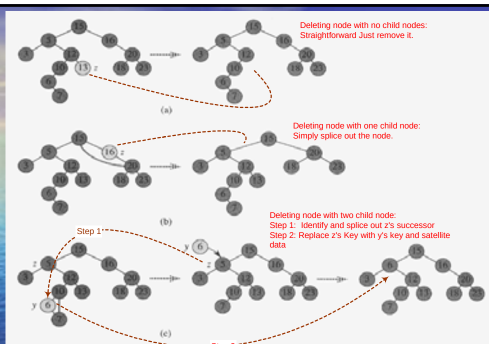
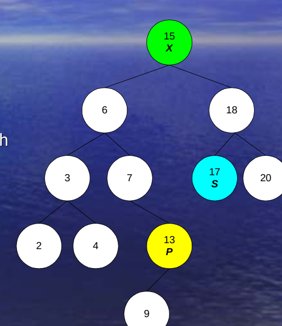


Figure 12.4 Deleting a node z from a binary search tree. Which node is actually removed depends on how many children z has; this node is shown lightly shaded. **(a)** If z has no children, we just remove it. **(b)** If z has only one child, we splice out z. **(c)** If z has two children, we splice out its successor y, which has at most one child, and then replace z's key and satellite data with y's key and satellite data.

Successor

- If the right subtree of node X is nonempty, then the successor of x is the leftmost node in the right subtree.
- The successor of the node with key 15 is node with key 17.
  - Similarly, If *X* has two children, its predecessor is the maximum value in its left subtree. (its successor the minimum value in its right subtree)
- Running time for both successor and predecessor is *O(h).*



Variations on Binary Trees.

- Binary Trees can implement operations such as Search, Predecessor, Successors, Min, Max, Insert, Delete in *O(h) Time.*
  - *Fine for small trees.*
  - *Once the height is large they perform as good as Linked list.*
- AVL
- Red-Black

19

AVL: is a balanced binary tree, named for their inventors Adelson-Velskii and Landis, that maintain a *O(log(n))* for search, insert and delete.

RED and Black Trees: Named for the fact that each node has a one extra bit for storage. The bit identifies if the node is either RED or BLACK. The tree is balanced to ensure a *O(log(n))* for operations such as search, predecessor, successors, insert, delete.

20

References:

- **Cormen, Thomas H. et al. Introduction to Algorithms, 2/e. North America: MIT Press, 2003.**
- **Rosen, Kenneth. Discrete Mathematics and its Applications, 5/e New York: McGraw-Hill, 2003.**
- **AVL:** http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/AVL.html

21

The End.

Q & A.

22