CS 4/56101

Design and Analysis of Algorithms

Kent State University

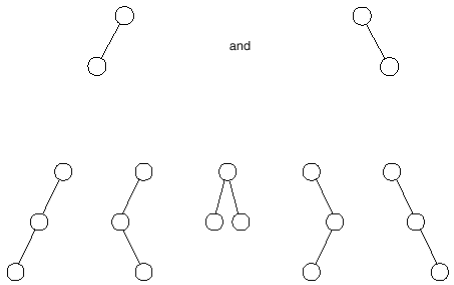
Dept. of Math & Computer Science

LECT-10

Binary Tree

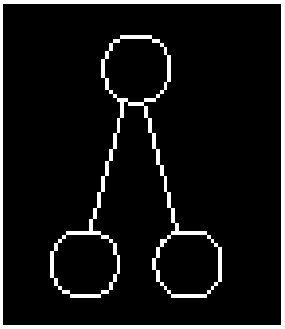
Binary Tree

DEFINITION A ***binary tree*** is either empty, or it consists of a node called the ***root*** together with two binary trees called the ***left subtree*** and the ***right subtree*** of the root.

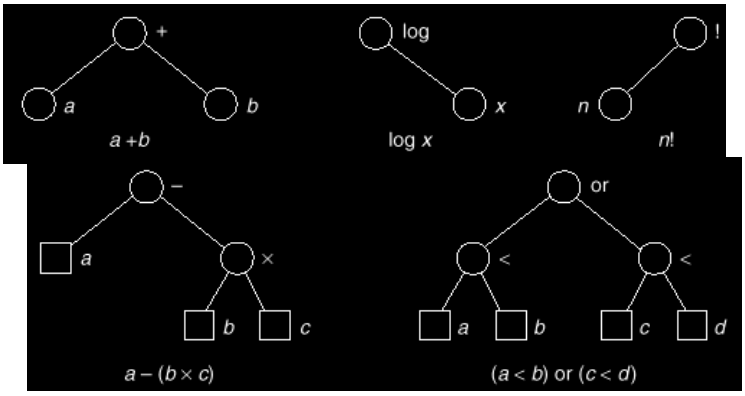


Traversal of Binary Tree

- Preorder: VLR
- Inorder LVR
- Postorder LRV



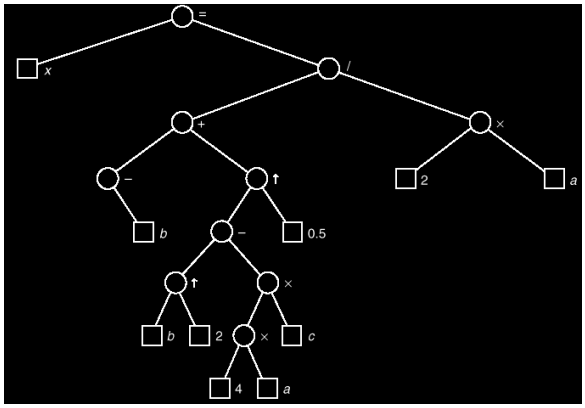
Expression Tree



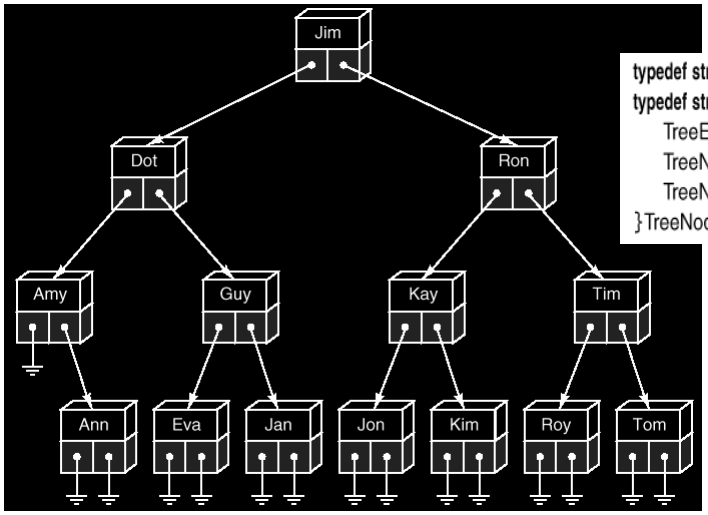
Expression:	$a + b$	$\log x$	$n!$	$a - (b \times c)$	$(a < b)$ or $(c < d)$
Preorder :	$+\ a\ b$	$\log\ x$	$!\ n$	$-\ a \times b\ c$	or $<\ a\ b\ <\ c\ d$
Inorder :	$a + b$	$\log\ x$	$n\ !$	$a - b \times c$	$a\ <\ b$ or $c\ <\ d$
Postorder :	$a\ b\ +$	$x\ \log$	$n\ !$	$a\ b\ c \times -$	$a\ b\ <\ c\ d\ <$ or

Expression Tree

$$x = (-b + (b\uparrow 2 - 4 \times a \times c)\uparrow 0.5)/(2 \times a)$$



Linked List Implementation




```
typedef struct treenode TreeNode;
typedef struct treenode{
    TreeEntry entry;
    TreeNode *left;
    TreeNode *right;
}TreeNode;
```

Inorder Traversal Routines

```
void B(TreeNode *root,
       void (*Visit)(TreeEntry x))
{
  if (root) {
    A(root->left, Visit);
    Visit(root->entry);
    A(root->right, Visit)
  }
}
```

Quiz: How the Postorder and Preorder Traversal Routines will be?

Binary Search Tree

9

Idea

- Dilemma
  - In a Linked list there is no obvious way of moving into the middle point of list other than sequential traversal.
  - On the other hand, in a contiguous list it is difficult to add/delete entries!

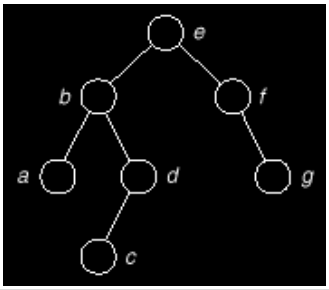
> Can we find an implementation for ordered lists in which we can search quickly (as with binary search on a contiguous list) and in which we can make insertions and deletions quickly (as with a linked list)?

- Solution
  - Binary Search Tree!

Binary Search Tree

DEFINITION A ***binary search tree*** is a binary tree that is either empty or in which every node contains a key and satisfies the conditions:

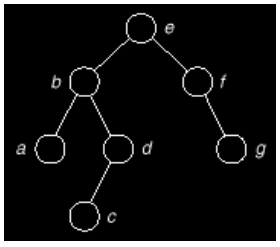
1. The key in the left child of a node (if it exists) is less than the key in its parent node.
2. The key in the right child of a node (if it exists) is greater than the key in its parent node.
3. The left and right subtrees of the root are again binary search trees.



- Assumption: No two entries in a binary search tree may have equal keys.

Op-1: Searching into a BST


```
TreeNode *TreeSearch(TreeNode *root, KeyType target)
{
    if (root)
        if (LT(target, root->entry.key))
            root = TreeSearch(root->left, target);
        else if (GT(target, root->entry.key))
            root = TreeSearch(root->right, target);
    return root;
}
```

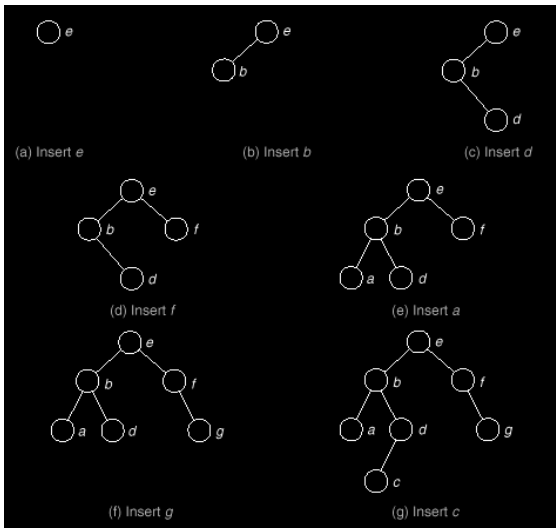


Op-2: Insertion into a BST

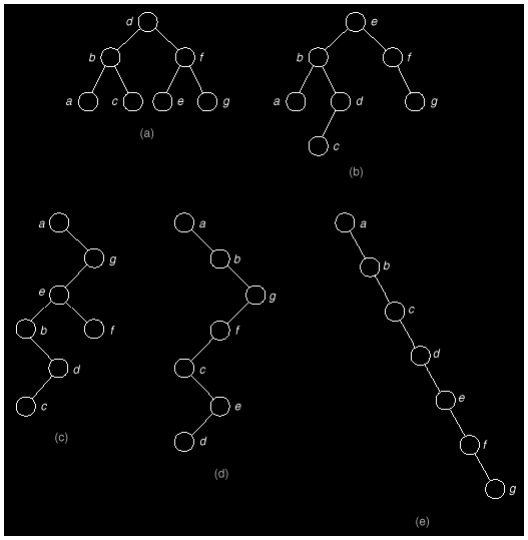

```
TreeNode *InsertTree(TreeNode *root, TreeNode *newnode)
{
    if (!root) {
        root = newnode;
        root->left = root->right = NULL;

    } else if (LT(newnode->entry.key, root->entry.key))
        root->left = InsertTree(root->left, newnode);
    else
        root->right = InsertTree(root->right, newnode);
    return root;
}
```

Quiz: insert e,b,d,f,a,g and c

Example: Insert e,b,d,f,a,g,c

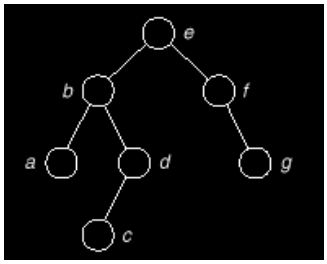


Uniqueness of BST



Tree Sort

- Quiz: How to get a sorted list from a BST?



Tree Sort & Quick Sort

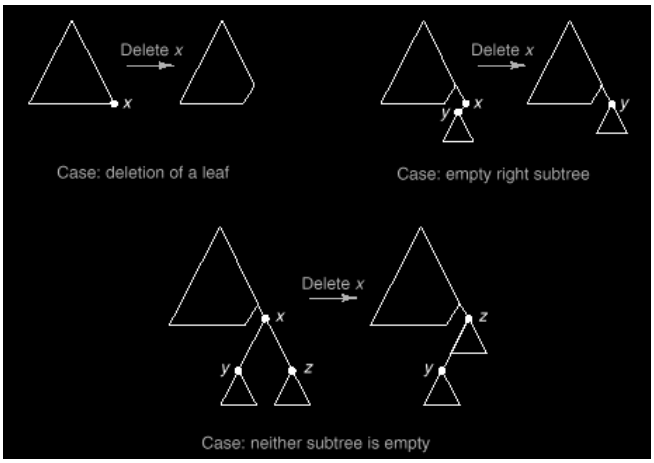
- In BST
  - The first key becomes the root of the tree.
  - During search the key is compared with the root.
- In Quick Sort
  - The first key is compared with the first pivot...

THEOREM 9.1 Treesort makes exactly the same comparisons of keys as does quicksort when the pivot for each sublist is chosen to be the first key in the sublist.

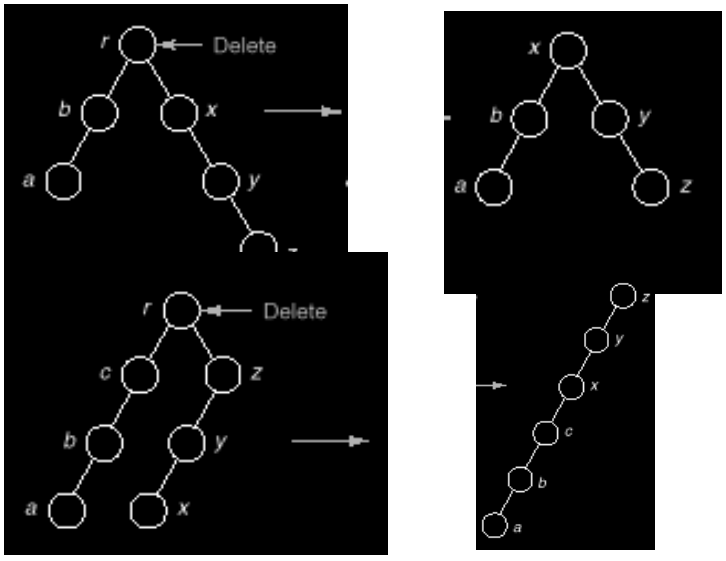
COROLLARY 9.2 In the average case, on a randomly ordered list of length n, treesort does $2n \ln n + O(n) \approx 1.39n \lg n + O(n)$ comparisons of keys.

- One advantage of Tree sort: It does not require all the key to be present throughout the sorting process. But, Quick Sort requires access to all the items.

Op-3: Deletion from BST



Example



Today's Math
Rabbit Counting

Rabbit Counting!

QUIZ: How many pairs of rabbits can be produced from a single pair in a year? We start with a single newly born pair; it takes one month for a pair to mature, after which they produce a new pair each month, and the rabbits never die.

-LEONARDO FIBONACCI, 1202

- In the first month there is 1, in the second month there is still one, but matured. In the third month there are now two, one is matured. Only the mature one reproduces...

$$|F_0| = 0 \quad |F_1| = 1$$

$$|F_n| = |F_{n-1}| + |F_{n-2}|$$

Evaluation

- Let us consider an infinite series:

$$F(x) = F_0 + F_1 x + F_2 x^2 + ... + F_n x^n + ...$$

$$\begin{aligned} F(x) &= F_0 + F_1 x + F_2 x^2 + ... + F_n x^n + ... \\ x.F(x) &= \quad + F_0 x + F_1 x^2 + ... + F_{n-1} x^n + ... \\ x^2.F(x) &= \quad\quad + F_0 x^2 + ... + F_{n-2} x^n + ... \end{aligned}$$

- Therefore:

$$(1 - x - x^2)F(x) = F_0 + (F_1 - F_0)x = x$$

$$F(x) = \frac{x}{(1 - x - x^2)} = \frac{1}{\sqrt{5}}\left(\frac{1}{1 - Ax} - \frac{1}{1 - Bx}\right)$$

$$A = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618034 \quad B = \frac{1}{2}(1 - \sqrt{5}) \approx -.618034$$

Evaluation (continued..)

- From the original definition:

$$F(x) = F_0 + F_1 x + F_2 x^2 + ... + F_n x^n + .$$

$$= \frac{1}{\sqrt{5}}(1 + Ax + A^2 x^2 + ...) - \frac{1}{\sqrt{5}}(1 + Bx + B^2 x^2 + ...)$$

$$F_n = \frac{1}{\sqrt{5}}(A^n - B^n)$$

- B is always smaller than 1, therefore:

$$F_n \approx \frac{1}{\sqrt{5}} A^n = \frac{1}{\sqrt{5}} \left[\frac{1+\sqrt{5}}{2} \right]^n \quad \text{rounded to nearest integer}$$

F(n) is always integer!

The number A has been studied since the time of the Greeks- is called golden mean-the ratio of A:1 gives the most pleasing shape of rectangle.

The Parthenon and many other Greek buildings have sides with this ratio!