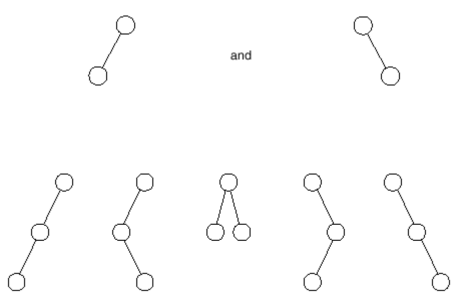
Binary Tree

1

DESIGN & ANALYSIS OF ALGORITHM

Binary Tree

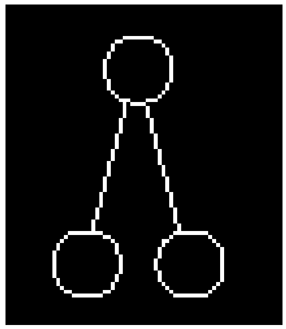
DEFINITION A ***binary tree*** is either empty, or it consists of a node called the ***root*** together with two binary trees called the ***left subtree*** and the ***right subtree*** of the root.



Traversal of Binary Tree

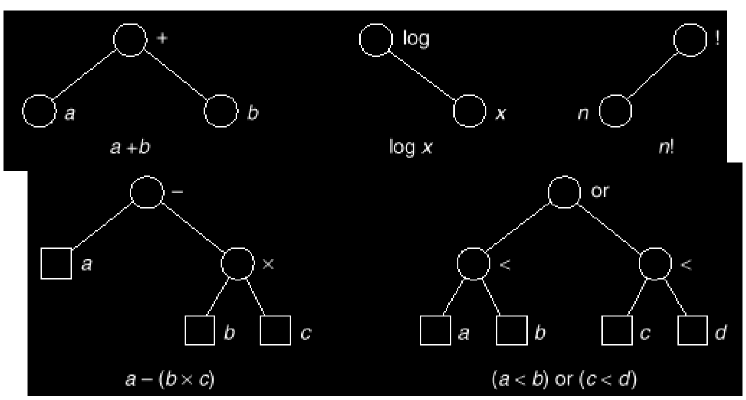
DESIGN & ANALYSIS OF ALGORITHM

- Preorder: VLR
- Inorder LVR
- Postorder LRV



Expression Tree

DESIGN & ANALYSIS OF ALGORITHM

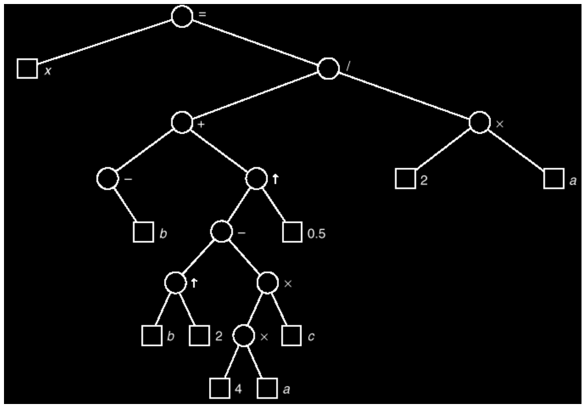


Expression:	$a+b$	$\log x$	$n!$	$a-(b\times c)$	$(a<b)$ or $(c<d)$
Preorder :	$+\ a\ b$	$\log\ x$	$!\ n$	$-\ a\ \times\ b\ c$	or $<\ a\ b\ <\ c\ d$
Inorder :	$a\ +\ b$	$\log\ x$	$n\ !$	$a\ -\ b\ \times\ c$	$a\ <\ b$ or $c\ <\ d$
Postorder :	$a\ b\ +$	$x\ \log$	$n\ !$	$a\ b\ c\ \times\ -$	$a\ b\ <\ c\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

Expression Tree

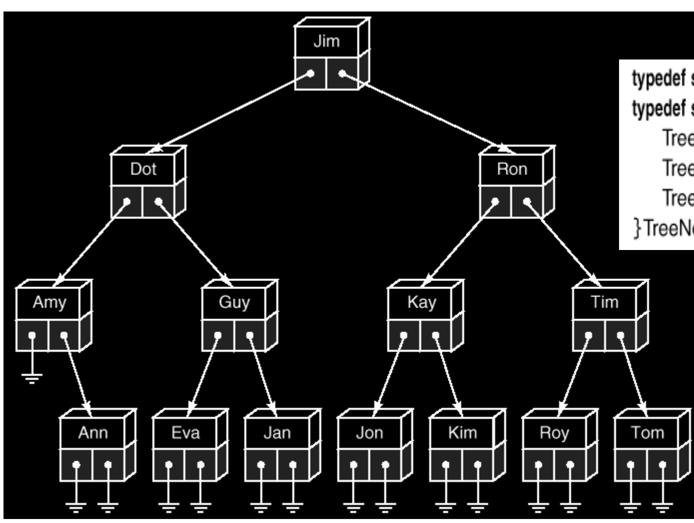
DESIGN &
ANALYSIS OF
ALGORITHM

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



Linked List Implementation

DESIGN &
ANALYSIS OF
ALGORITHM




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

Inorder Traversal Routines

DESIGN &
ANALYSIS OF
ALGORITHM

```
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

Idea

DESIGN & ANALYSIS OF ALGORITHM

- 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)?

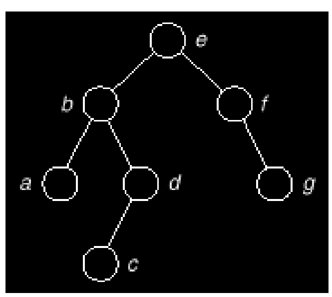
- Solution
  - Binary Search Tree!

Binary Search Tree

DESIGN & ANALYSIS OF ALGORITHM

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.

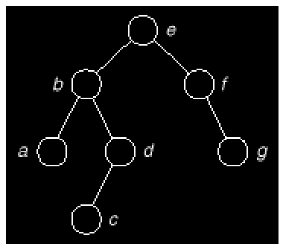


- Assumption: No two entries in a binary search tree may have equal keys.

Op-1: Searching into a BST

DESIGN & ANALYSIS OF ALGORITHM

```
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

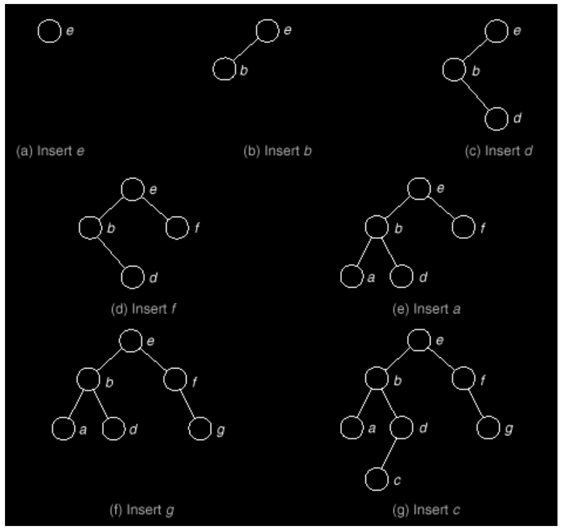
DESIGN & ANALYSIS OF ALGORITHM

```
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;
}
```

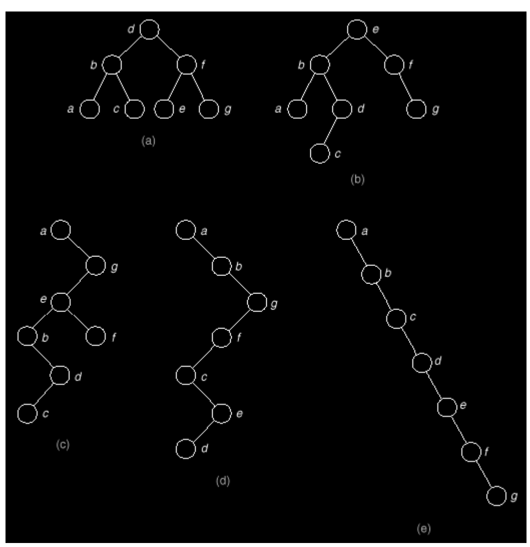
Quiz: insert e,b,d,f,a,g and c

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



DESIGN & ANALYSIS OF ALGORITHM

Uniqueness of BST

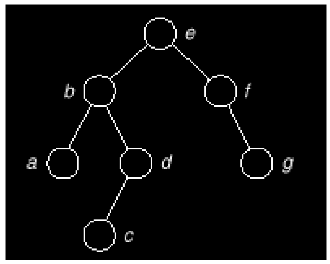


DESIGN & ANALYSIS OF ALGORITHM

Tree Sort

DESIGN & ANALYSIS OF ALGORITHM

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



Tree Sort & Quick Sort

DESIGN & ANALYSIS OF ALGORITHM

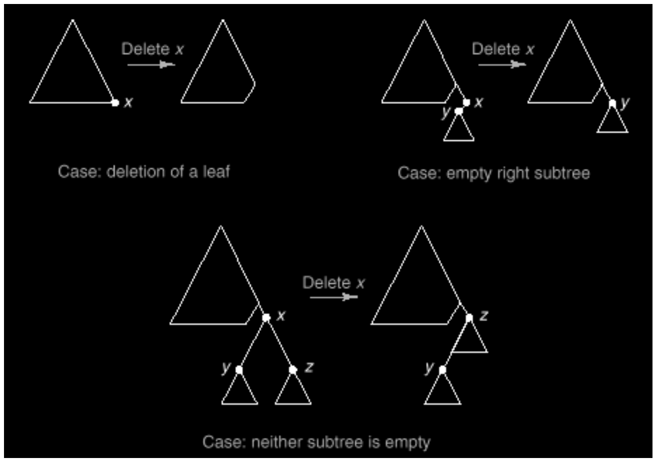
- In BST
  - The first key becomes the root of the tree.
  - During search the key is compared with the root.
- In Quick Sort
  - The first key is compared with the first pivot...

> 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.

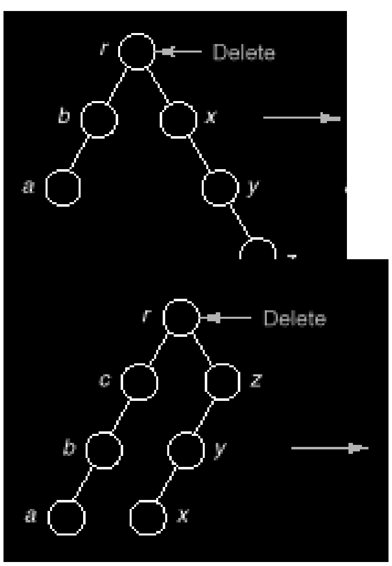
Op-3: Deletion from BST

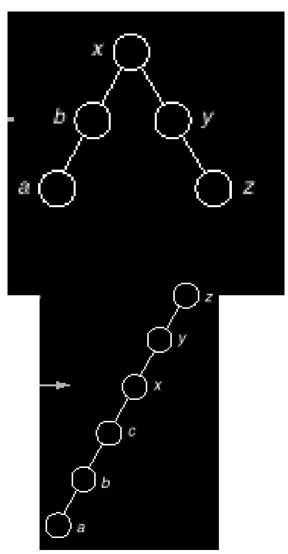


DESIGN & ANALYSIS OF ALGORITHM

LECT-10, S-17
ALG00S, javed@kent.edu
Javed I. Khan@1999

Example





DESIGN & ANALYSIS OF ALGORITHM

LECT-10, S-18
ALG00S, javed@kent.edu
Javed I. Khan@1999

Today’s Math
Rabbit Counting

19

Evaluation of Fibonacci Series

DESIGN & ANALYSIS OF ALGORITHM

- Let us consider an infinite series:

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

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

- Therefore:

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

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

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

LECT-10, S-20
ALG00S, javed@kent.edu
Javed I. Khan@1999

Evaluation (continued..)

DESIGN & ANALYSIS OF ALGORITHM

- From the original definition:

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

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

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

- 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!