Optimal Binary Search Tree

Presentation by:
V. Balasubramanian
SSN College of Engineering

Definition

A ***binary search tree*** is a binary tree of items (ordinarily called keys), that come from an ordered set, such that

1. Each node contains one key.
2. The keys in the left subtree of a given node are less than or equal to the key in that node.
3. The keys in the right subtree of a given node are greater than or equal to the key in that node.

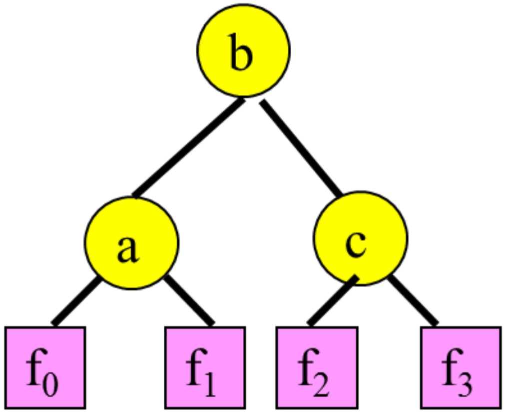
Contd…

- The depth of a node in a tree is the number of edges in the unique path from the root to the node. This is also called the level of the node in the tree.

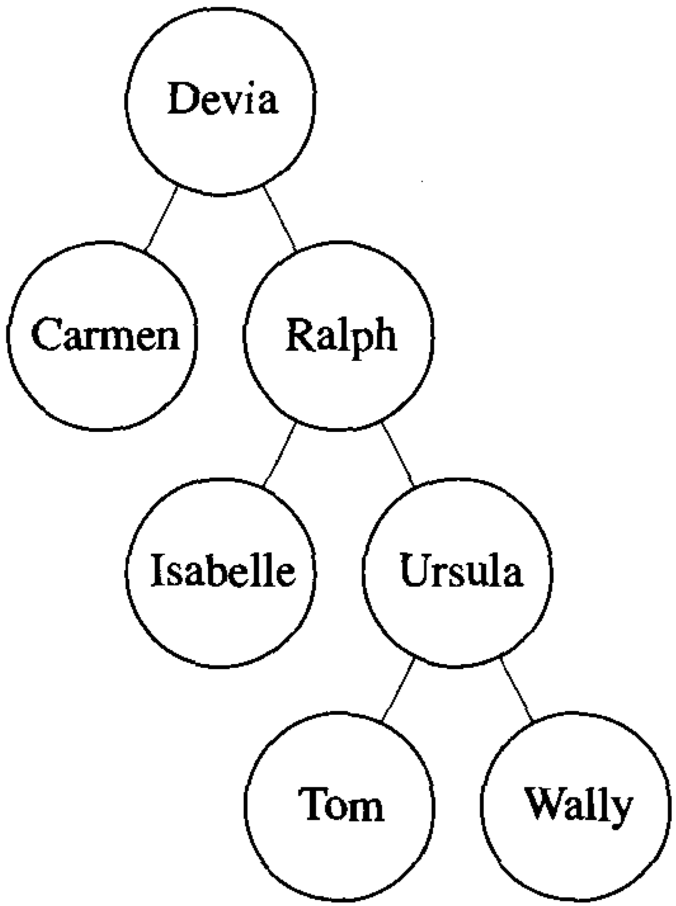
Contd…

- Ordinarily, a binary search tree contains records that are retrieved according to the values of the keys.
- Our goal is to organize the keys in a binary search tree so that the average time it takes to locate a key i minimized.
- A tree that is organized in this fashion is called optimal.

- 20 keys, we have to try out 131,282,408,400 different trees.



Example



BST

```
struct nodetype
{
    keytype key;
    nodetype* left;
    nodetype* right;
};

typedef nodetype* node_pointer;
```

Search

```
while (! found)
    if (p-> key == keyin)
        found = true;
    else if (keyin < p-> key);
        p = p-> left;
    else
        p = p-> right;
```

$$depth(key) + 1,$$

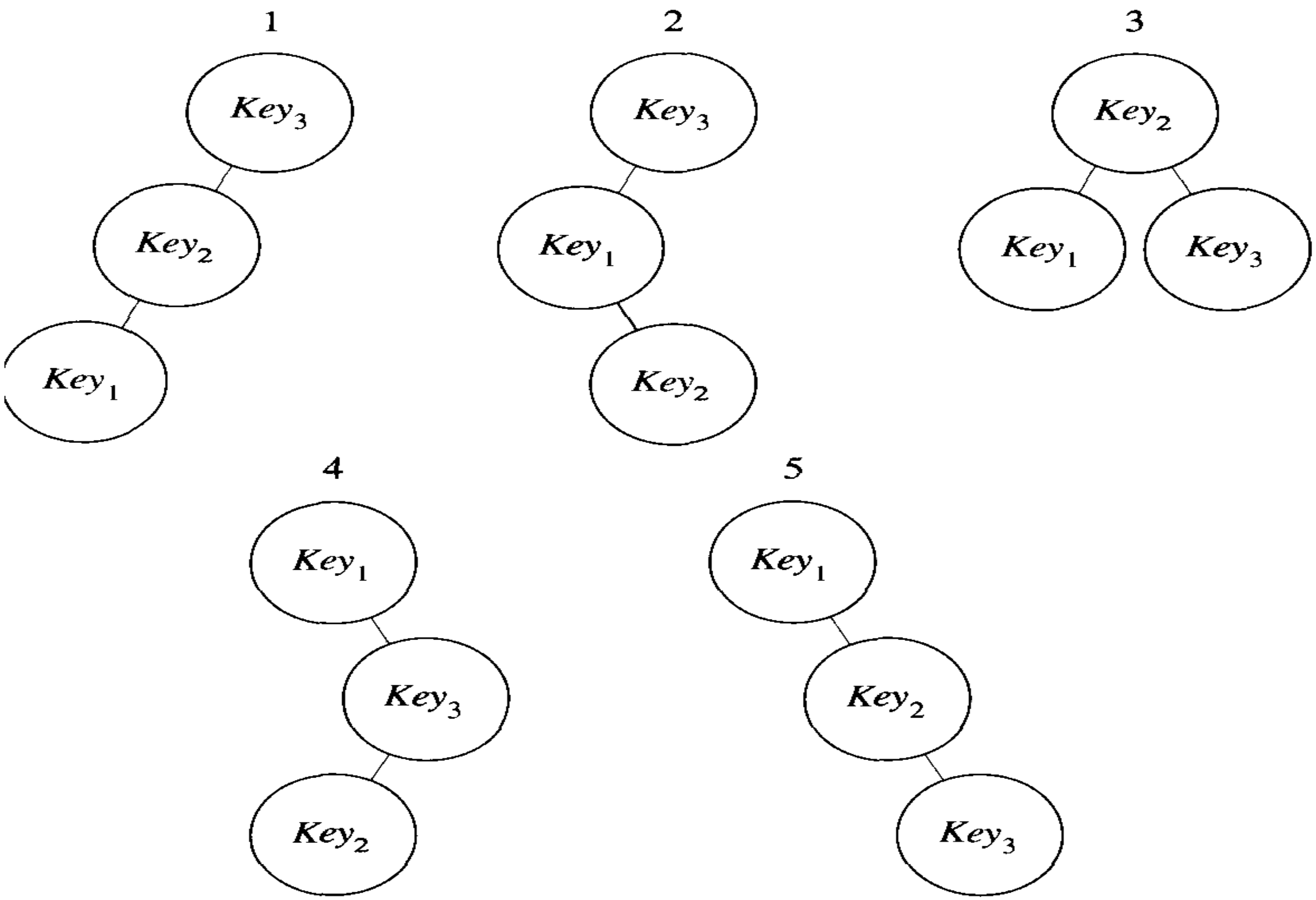
Let $Key_1, Key_2, \ldots, Key_n$ be the n keys in order, and let p_i be the probability that Key_i is the search key. If c_i is the number of comparisons needed to find Key_i in a given tree, the average search time for that tree is

$$\sum_{i=1}^{n} c_i p_i.$$

This is the value we want to minimize.

Example

$p_1 = 0.7 \qquad p_2 = 0.2, \qquad \text{and} \qquad p_3 = 0.1,$



1
Key3
Key2
Key1
2
Key3
Key1
Key2
3
Key2
Key1
Key3
4
Key1
Key3
Key2
5
Key1
Key2
Key3

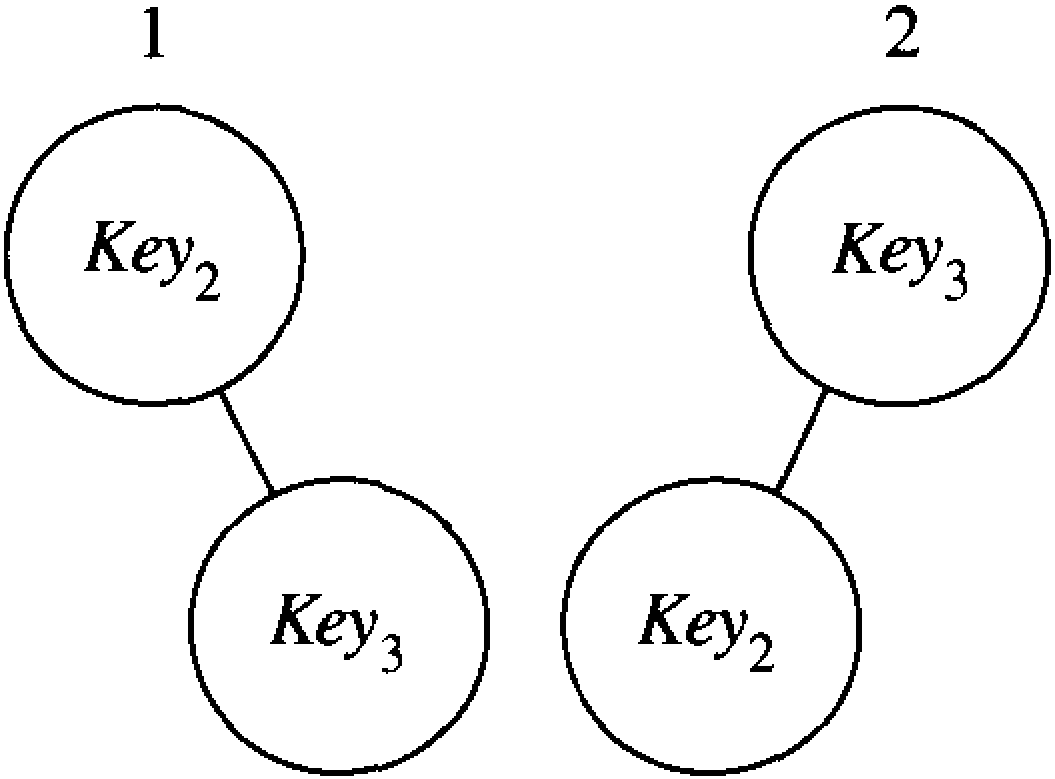
A[i][j]

To that end, suppose that keys Key_i through Key_j are arranged in a tree that minimizes

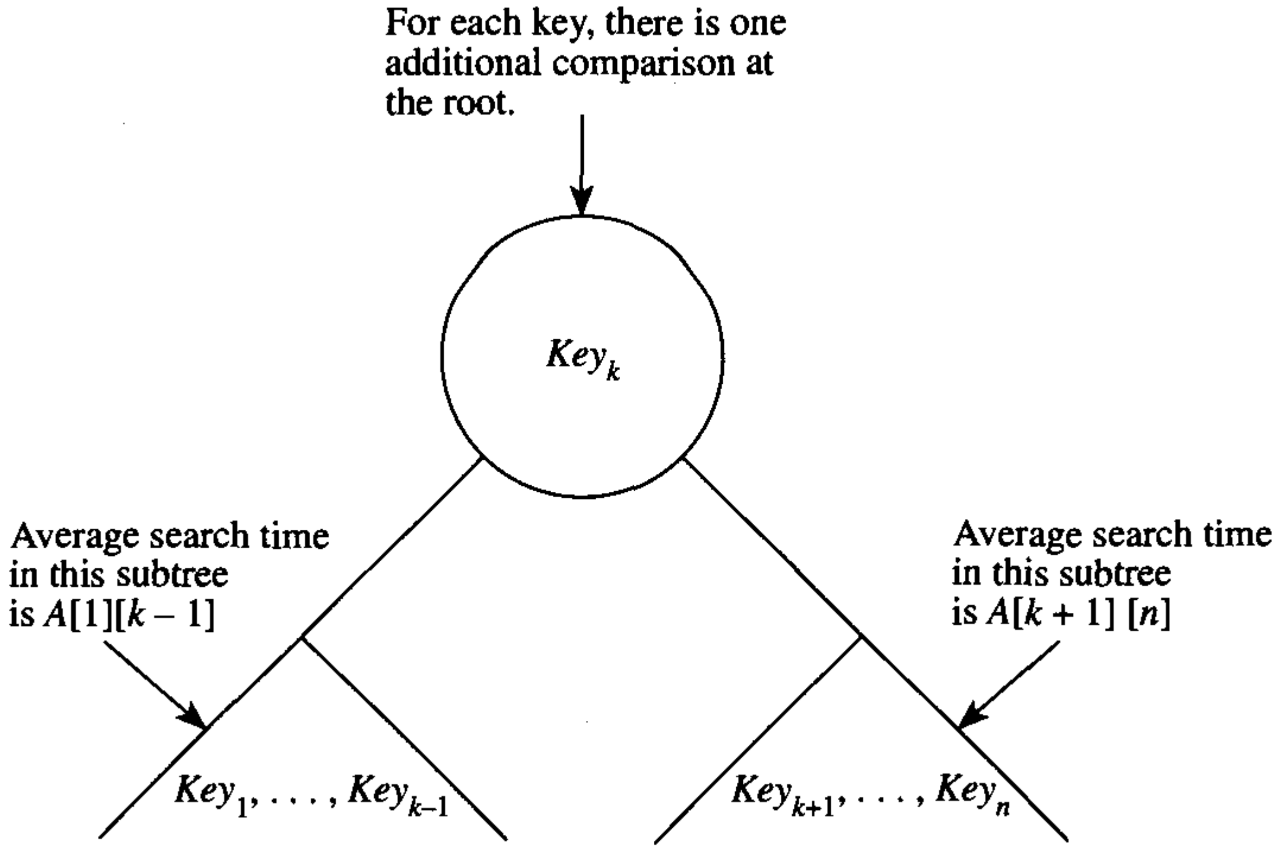
$$\sum_{m=i}^{j} c_m p_m,$$

Because it takes one comparison to locate a key in a tree containing one key, $A[i][i] = p_i$.

A[2][3]



OBST



For each key, there is one additional comparison at the root.
Key_k
Average search time in this subtree is $A[1][k-1]$
Average search time in this subtree is $A[k+1][n]$
$Key_1, \ldots, Key_{k-1}$
$Key_{k+1}, \ldots, Key_n$

$$\underbrace{A[1][k-1]}_{\substack{\text{Average time}\\ \text{in left}\\ \text{subtree}}} + \underbrace{p_1 + \cdots + p_{k-1}}_{\substack{\text{Additional time}\\ \text{comparing at root}}} + \underbrace{p_k}_{\substack{\text{Average time}\\ \text{searching}\\ \text{for root}}} + \underbrace{A[k+1][n]}_{\substack{\text{Average time}\\ \text{in right}\\ \text{subtree}}} + \underbrace{p_{k+1} + \cdots + p_n}_{\substack{\text{Additional time}\\ \text{comparing at root}}},$$

which equals

$$A[1][k-1] + A[k+1][n] + \sum_{m=1}^{n} p_m.$$

Recurrence equation

$$A[1][n] = \underset{1\le k\le n}{minimum}(A[1][k-1] + A[k+1][n]) + \sum_{m=1}^{n} p_m,$$

$$A[i][j] = \underset{i\le k\le j}{minimum}(A[i][k-1] + A[k+1][j]) + \sum_{m=i}^{j} p_m \qquad i < j$$

$$A[i][i] = p_i$$

$A[i][i-1]$ and $A[j+1][j]$ are defined to be 0.