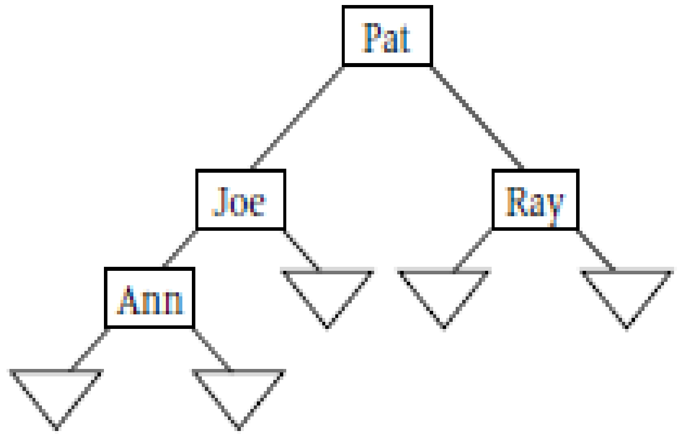
Optimal Binary Search Trees

Textbook Reading:

Section 8.3 Optimal Binary Search Trees, pp. 339-348.

Example Binary Search Tree



The binary search tree has keys 'Ann', 'Joe', 'Pat', 'Ray'. The internal nodes correspond to the successful searches *X* = 'Ann', *X* = 'Joe', *X* = 'Pat', *X*= 'Ray', and the leaf nodes correspond to the unsuccessful searches

X < 'Ann', 'Ann' < *X* < 'Joe', 'Joe' < *X* < 'Pat', 'Pat' < *X* < 'Ray', 'Ray' < *X*.

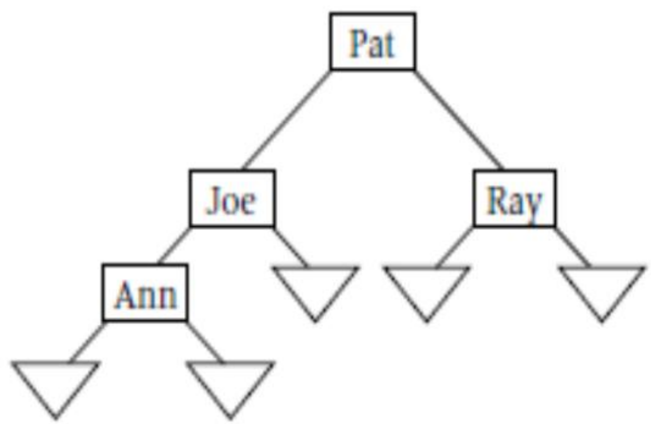
Pseudocode for Searching a Binary Search Tree

```
procedure BinSrchTreeSearch(Root, X, Location, Found)
Input:      Root (→BinaryTreeNode)   //points at root of a binary search tree
            X (KeyType)
Output: Location (→BinaryTreeNode)   //points at occurrence of X, if any
        Found(Boolean)  //.false. if X not in tree, .true. otherwise
            Found ← .false.
            Location ← null
            Current ← Root
            while Current ≠ null .and. .not. Found do
                Location ← Current
                if X = Current→Key then
                    Found ← .true.
                else
                    if X < Current→Key then
                        Current ← Current→LeftChild
                    else
                        Current ← Current→RightChild
                    endif
                endif
            endwhile
end BinSrchTreeSearch
```

Average number of comparisons

Let p_0,p_1,p_2,p_3 be the probability that X = 'Ann', X = 'Joe', X = 'Pat', X = 'Ray', respectively, and let q_0,q_1,q_2,q_3,q_4, denote the probability that

X < 'Ann', 'Ann' < X < 'Joe', 'Joe' < X < 'Pat', 'Pat' < X < 'Ray', 'Ray' < X,



The average (expected) number of comparisons made by *BinSrchTreeSearch* for the binary search tree above is

$$3p_0 + 2p_1 + p_2 + 2p_3 + 3q_0 + 3q_1 + 2q_2 + 2q_3 + 2q_4.$$

General Binary Search Tree

n internal nodes correspond to n keys

$$K_0 < K_1 < \ldots < K_{n\text{-}1}$$

with associated probabilities $\mathbf{p} = (p_0, p_1, \ldots, p_{n\text{-}1})$

$n + 1$ leaf nodes correspond to the $n + 1$ intervals

I_0: $X < K_0$, I_1: $K_0 < X < K_1$,…,I_{n-1}: $K_{n-2} < X < K_{n\text{-}1}$, I_n: $X > K_{n\text{-}1}$

with associated probabilities $\mathbf{q} = (q_0, q_1, \ldots, q_n)$.

Number of comparisons performed by *BinSrchTreeSearch*

- Let d_i denote the depth of the internal node corresponding to K_i, $i = 0,\ldots,n - 1$.
- Let e_i denote the depth of the leaf node corresponding to the interval I_i, $i = 0, 1,\ldots,n$.
- If $X = K_i$, then *BinSrchTreeSearch* traverses the path from the root to the internal node corresponding to K_i.
- Thus, it terminates after performing $d_i + 1$ comparisons.
- On the other hand, if X lies in I_i, then *BinSrchTreeSearch* traverses the path from the root to the leaf node corresponding to I_i and terminates after performing e_i comparisons.

Formula for average number of comparisons

It follows from the previous slide that the average number of comparisons performed by *BinSrchTreeSearch* is given by:

$$A(T, n, \mathbf{p}, \mathbf{q}) = \sum_{i=0}^{n-1} p_i\,(d_i + 1) + \sum_{i=0}^{n} q_i e_i$$

Optimal Binary Search Tree

The problem we need to solve is to

$$minimize\ A(T, n, \mathbf{p}, \mathbf{q})$$

over all binary search trees *T* for n keys $K_0 < K_1 < \ldots < K_{n\text{-}1}$ and associated probabilities $\mathbf{p} = (p_0, p_1, \ldots, p_{n\text{-}1})$ and $\mathbf{q} = (q_0, q_1, \ldots, q_n)$.

The binary search tree T that realizes the minimum is called an **optimal binary search tree**.

Combine Formula

Let

A(T) = A(T,n,**p**,**q**)

$\sigma(i,j) = p_i + \ldots + p_j + q_i + \ldots + q_{j+1}$

L and R be the left and right subtrees of T.

Observing that the depth of L and R are both one less than the depth of T, it follows that

$$A(T) = A(L) + A(R) + \sigma(i,j).$$

Note that the sum of the p and q probabilities in L and R will not sum to 1, although, this is true for the initial input tree. We drop this constraint in the design of our algorithm.

Principle of Optimality

PSN. Prove that the Principle of Optimality holds.

Optimal Binary Search Trees Involving Subset of the Keys

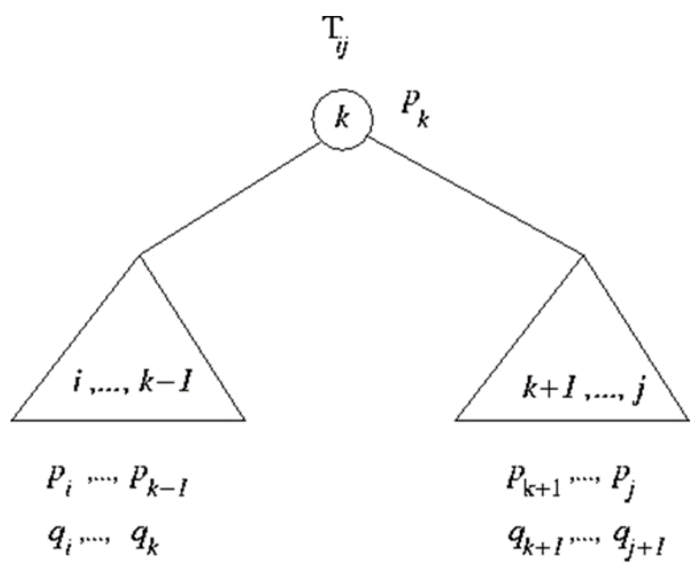
Let T_{ij} be an optimal search tree involving keys

$$K_i < K_{i+1} < \ldots < K_j \, .$$

Principle of Optimality Holds

Principle of Optimality:

T_{ij} is optimal implies L and R are both optimal, i.e., $L = T_{i,j\text{-}1}$ and $R = T_{k+1,j}$



Recurrence Relation

If the root of T_{ij} is *k*, then

$$A(T_{ij}) = A(T_{i,k-1}) + A(T_{k+1,j}) + \sigma(i,j)$$

Thus,

$$A(T_{ij}) = \min_{k \in \{i, i+1, \ldots, j\}} \{A(T_{i,k-1}) + A(T_{k+1,j})\} + \sigma(i,j).$$

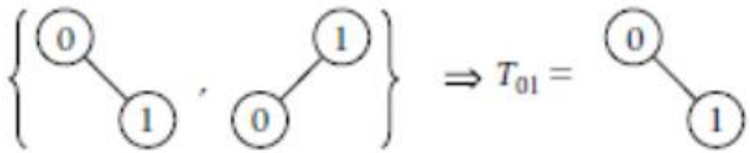
Initial Condition: $A(T_{ii}) = \sigma(i,i) = p_i + q_i + q_{i+1}$

Action for an example

i	0	1	2	3	4
p_i	.15	.1	.2	.3	
q_i	.05	.05	0	.05	.1

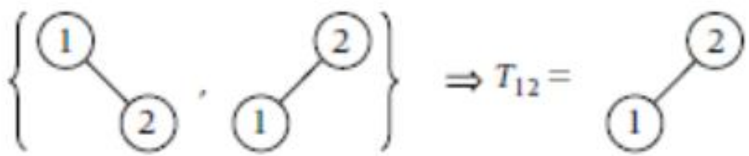
$T_{00} = (0) \quad T_{11} = (1) \quad T_{22} = (2) \quad T_{33} = (3)$

$A(T_{00}) = .25 \quad A(T_{11}) = .15 \quad A(T_{22}) = .25 \quad A(T_{33}) = .45$



$A(T_{01}) = .5$

min $\{0 + .15 = .15, .25 + 0 = .25\} + .35 = .5$

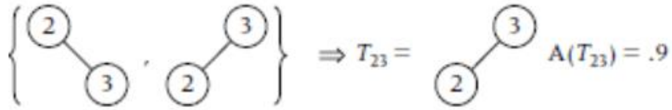


$A(T_{12}) = .55$

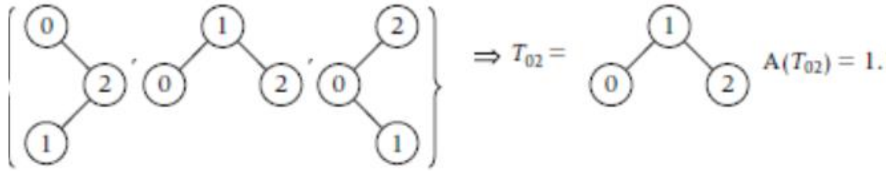
min $\{0 + .25 = .25, .15 + 0 = .15\} + .4 = .55$

Continued

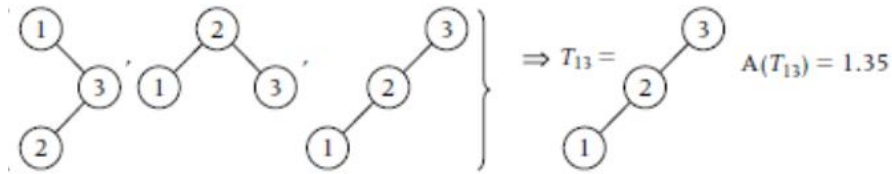
Example continued



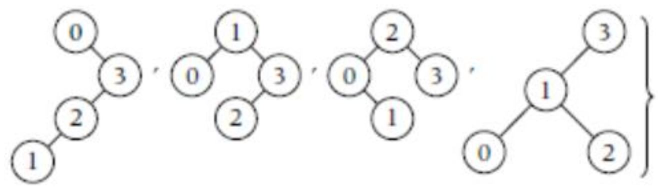
$\min\{0 + .45 = .45, .25 + 0 = .25\} + .65 = .9$



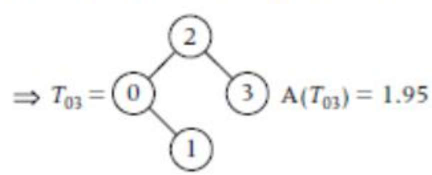
$\min\{0 + .55 = .55, .25 + .25 = .5, .5 + 0 = .5\} + .6 = 1.1$



$\min\{0 + .9 = .9, .15 + .45 = .6, .55 + 0 = .55\} + .8 = 1.35$



$\min\{0 + 1.35 = 1.35, .25 + .9 = 1.15, .5 + .45 = .95, 1.1 + 0 = 1.1\} + 1 = 1.95$



```
procedure OptimalSearchTree(P[0:n – 1],Q[0:n],Root[0:n – 1,0:n – 1],A[0:n – 1,0:n – 1])
Input:    P[0:n – 1] (an array of probabilities associated with successful searches)
          Q[0:n]   (an array of probabilities associated with unsuccessful searches)
Output:   Root[0:n – 1,0:n – 1]   (Root[i,j] is the key of the root node of T_ij)
          A[0:n – 1,0:n – 1]   (A[i,j] is A(T_ij) )
      for i ← 0 to n – 1 do
          Root[i,i] ← i
          Sigma[i,i] ← p[i] + q[i ] + q[i+1]
          A[i,i] ← Sigma[i,i]
      endfor
      for Pass ← 1 to n – 1  do
           for i ← 0 to n – 1 – Pass  do
                 j ← i + Pass
   //Compute σ(p_i,...,p_j,q_i,...q_j+1)
                 Sigma[i,j] ← Sigma[i,j – 1] + p[j] + q[j+1]
                 Root[i,j] ← i
                 Min ← A[i + 1,j]
                 for k ← i + 1 to j do
                       Sum ← A[i,k – 1] + A[k + 1,j]
                       if Sum < Min then
                            Min ← Sum
                            Root[i,j] ← k
                       endif
                 endfor
                 A[i,j] ← Min + Sigma[i,j]
           endfor
      endfor
end OptimalSearchTree
```

PSN. Solve action of *OptimalSearchTree* for following instance. Instead of drawing trees compute $Root[i, j]$ each stage

i	0	1	2	3
p_i	.25	.25	.05	
q_i	.2	0	0	.25

Complexity Analysis

In pseudocode for OptimalSearchTree

- *Pass* varies from 1 to $n - 1$
- i varies from 0 to $n - 1 - Pass$
- k varies from $i + 1$ to $i + Pass$

Setting $t = Pass$, it follows that $W(n)$ is given

$$\sum_{t=1}^{n-1} \sum_{i=0}^{n-1-t} \sum_{k=i+1}^{i+t} 1$$

$$= \sum_{t=1}^{n-1} (n-t)t$$

$$= n\sum_{t=1}^{n-1} t - \sum_{t=1}^{n-1} t^2$$

$$= n\left[(n-1)\frac{n}{2}\right] - \left[(n-1)n\frac{(2n-1)}{6}\right] \in \Theta(n^3).$$

How can you identify a dogwood tree?

By its bark.