Unit III

Optimal Binary Search Tree

Optimal Binary Search Trees

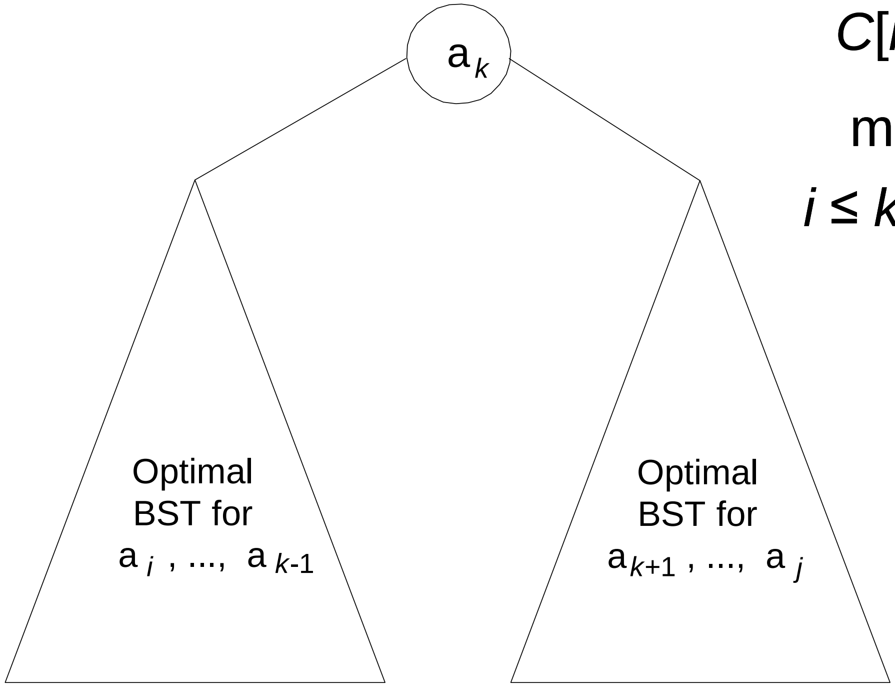
Problem: Given n keys $a_1 < \ldots < a_n$ and probabilities $p_1 \leq \ldots \leq p_n$ searching for them, find a BST with a minimum average number of comparisons in successful search.

Since total number of BSTs with n nodes is given by $C(2n,n)/(n+1)$, which grows exponentially, brute force is hopeless.

Example: What is an optimal BST for keys A, B, C, and D with search probabilities 0.1, 0.2, 0.4, and 0.3, respectively?

DP for Optimal BST Problem

Let $C[i,j]$ be minimum average number of comparisons made in T[i,j], optimal BST for keys $a_i < \ldots < a_j$, where $1 \leq i \leq j \leq n$. Consider optimal BST among all BSTs with some a_k ($i \leq k \leq j$) as their root; T[i,j] is the best among them.



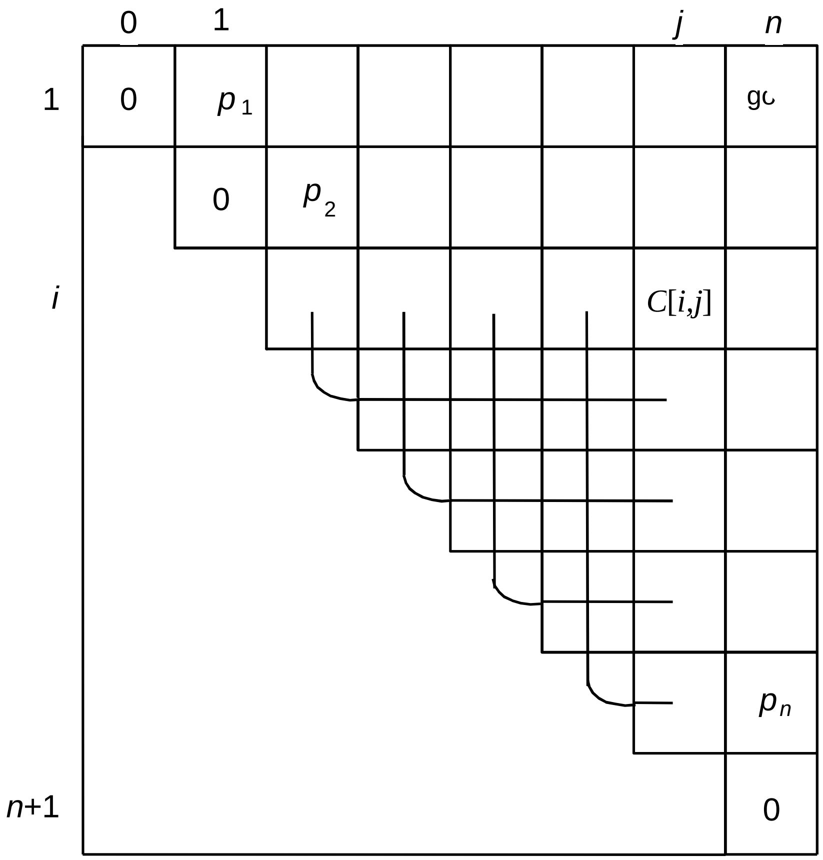
$$C[i,j] = \min_{i \leq k \leq j} \{p_k \cdot 1 + \sum_{s=i}^{k-1} p_s \,(\text{level } a_s \text{ in } T[i,k-1] + 1) + \sum_{s=k+1}^{j} p_s \,(\text{level } a_s \text{ in } T[k+1,j] + 1)\}$$

DP for Optimal BST Problem (cont.)

After simplifications, we obtain the recurrence for $C[i,j]$:

$$C[i,j] = \min_{i \le k \le j} \{C[i,k-1] + C[k+1,j]\} + \sum_{s=i}^{j} p_s \quad \text{for } 1 \le i \le j \le n$$

$$C[i,i] = p_i \quad \text{for } 1 \le i \le j \le n$$



Example:	key	*A*	*B*	*C*	*D*
	probability	0.1	0.2	0.4	0.3

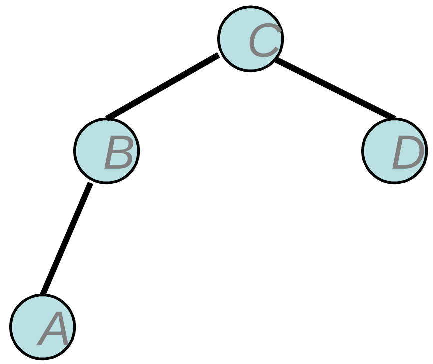
The tables below are filled diagonal by diagonal: the left one is filled using the recurrence

$$C[i,j] = \min_{i \le k \le j} \{C[i,k-1] + C[k+1,j]\} + \sum_{s=i}^{j} p_s, \quad C[i,i] = p_i;$$

the right one, for trees' roots, records *k*'s values giving the minima

i \ *j*	0	1	2	3	4
1	0	.1	.4	1.1	1.7
2		0	.2	.8	1.4
3			0	.4	1.0
4				0	.3
5					0

i \ *j*	0	1	2	3	4
1		1	2	3	3
2		a	2	3	3
3				3	3
4					4
5					



optimal BST

Optimal Binary Search Trees

```
ALGORITHM  OptimalBST(P[1..n])
    //Finds an optimal binary search tree by dynamic programming
    //Input: An array P[1..n] of search probabilities for a sorted list of n keys
    //Output: Average number of comparisons in successful searches in the
    //        optimal BST and table R of subtrees' roots in the optimal BST
    for i ← 1 to n do
        C[i, i − 1] ← 0
        C[i, i] ← P[i]
        R[i, i] ← i
    C[n + 1, n] ← 0
    for d ← 1 to n − 1 do //diagonal count
        for i ← 1 to n − d do
            j ← i + d
            minval ← ∞
            for k ← i to j do
                if C[i, k − 1] + C[k + 1, j] < minval
                    minval ← C[i, k − 1] + C[k + 1, j]; kmin ← k
            R[i, j] ← kmin
            sum ← P[i]; for s ← i + 1 to j do sum ← sum + P[s]
            C[i, j] ← minval + sum
    return C[1, n], R
```

Analysis DP for Optimal BST Problem

Time efficiency: $\Theta(n^3)$ but can be reduced to $\Theta(n^2)$ by taking advantage of monotonicity of entries in the root table, i.e., $R[i,j]$ is always in the range between $R[i,j\text{-}1]$ and $R[i+1,j]$

Space efficiency: $\Theta(n^2)$

Method can be expended to include unsuccessful searches