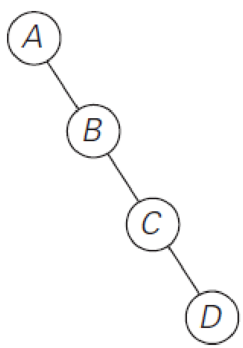
Dynamic Programming

Optimal Binary Search Tree

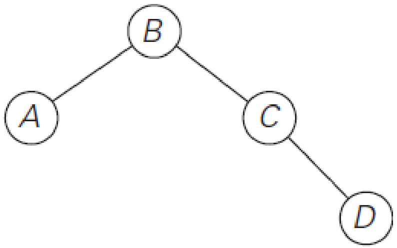
- Implementing dictionaries
- with operations (search, insert, delete)

- If probabilities of searching for elements of a set are known!
- How to make an optimal tree with minimal average number of comparisons for a random successful search?



$0.1 * 1 + 0.2 * 2 + 0.4 * 3 + 0.3 * 4 = 2.9$

keys	A	B	C	D
Prob.	0.1	0.2	0.4	0.3



$0.1 * 2 + 0.2 * 1 + 0.4 * 2 + 0.3 * 3 = 2.1$

Optimal Binary Search Tree

- Brute-Force approach:
  - Create all possible binary trees
  - Find the optimal one
  - Not feasible!

- Total number of possible binary search trees with n keys:

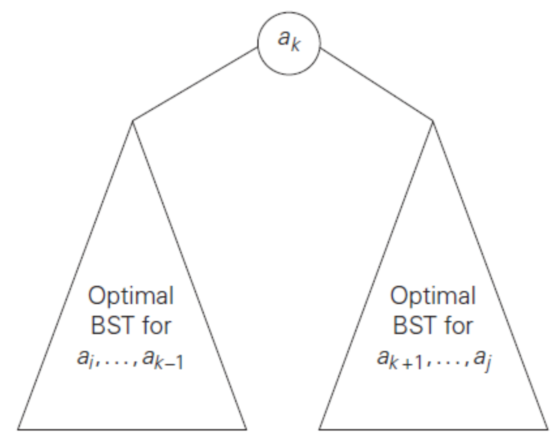
$$C(n) = \frac{1}{n+1}\binom{2n}{n}$$

$$C(0) = 1$$

Optimal Binary Search Tree

- Dynamic Programming approach:
- $a_1, \dots, a_n \rightarrow$ distinct key ordered from smallest to largest
- $p_1, \dots, P_n \rightarrow$ probabilities of searching for each key
- $C(i,j)$ = optimal number of comparisons in a tree made of keys $a_i, \dots, a_j$
  - Find optimal root a_k among keys $a_i, \dots, a_j$
  - Find optimal left subtree using keys $a_i, \dots, a_{k-1}$
  - Find optimal right subtree using keys $a_{k+1}, \dots, a_j$

- $C(j,j) = \min\{c(i,k-1) + C(k+1,j)\} + \sum_{s=i}^{j} p_s, \; i \leq k \leq j$
- $C(i,i) = P_i$
- $C(i,i-1) = 0$
- $C(1,n) = GOAL$



Optimal Binary Search Tree

- Example

keys	A	B	C	D
Number	1	2	3	4
Prob.	0.1	0.2	0.4	0.3

main table

	0	1	2	3	4
1	0	0.1			
2		0	0.2		
3			0	0.4	
4				0	0.3
5					0

root table

	0	1	2	3	4
1		1			
2			2		
3				3	
4					4
5					

Optimal Binary Search Tree

- Example

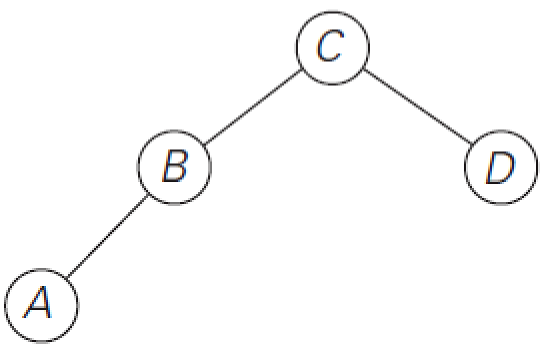
keys	A	B	C	D
Number	1	2	3	4
Prob.	0.1	0.2	0.4	0.3

main table

	0	1	2	3	4
1	0	0.1			
2		0	0.2		
3			0	0.4	
4				0	0.3
5					0

root table

	0	1	2	3	4
1		1			
2			2		
3				3	
4					4
5					



main table

	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

root table

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					