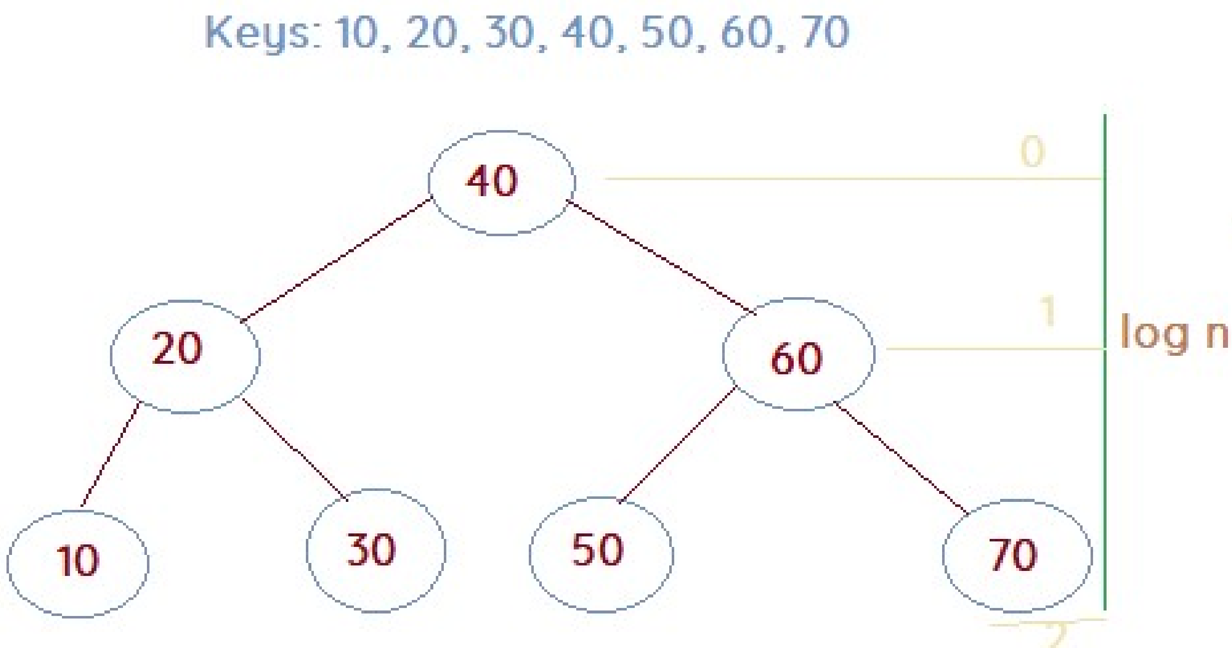
Lecture 21

Dynamic Programming: Optimal Binary Search Trees & its Time Complexity.

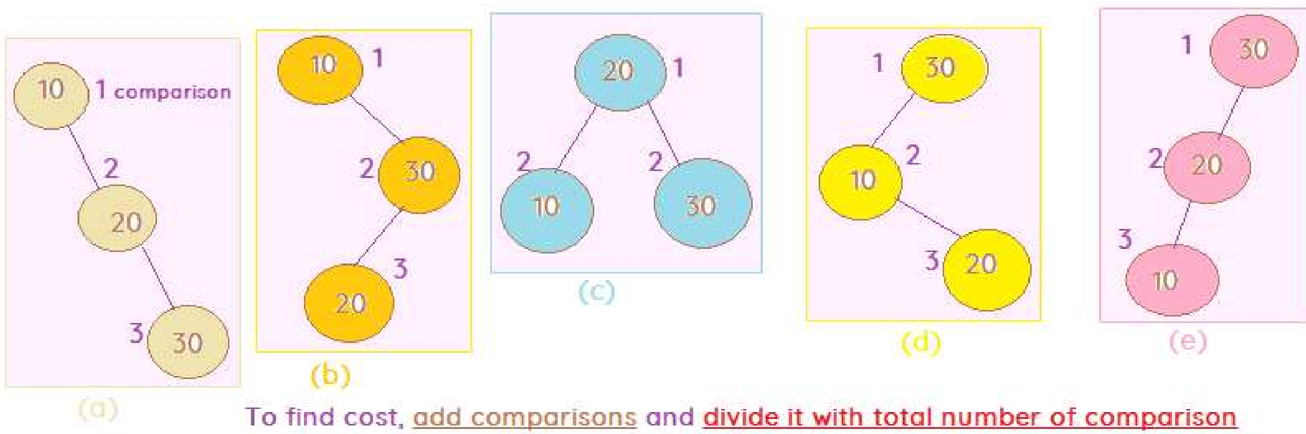
Binary Tree



How many different Binary Search Trees?

› Keys: 10, 20, 30; For n=3 Keys, 5 different binary trees are possible

$$Total\#\ of\ trees = \frac{1}{n+1}\binom{2n}{n}\ for\ n > 0, C(0) = 1$$



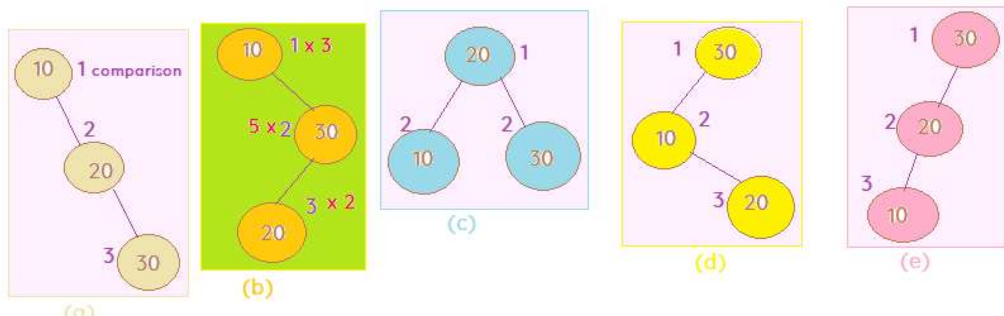
To find cost, add comparisons and divide it with total number of comparison
i.e. for (a) (1 + 2 + 3)/3=2, (b) 2, (c) 5/3, (d) 2 and (e) 2

› Cost of searching an element: Average Comparisons in a tree

› Average number of comparison for (C) are less, so it is a balanced binary tree due its less height.

Optimal Binary Search Tree

- In OBST, we consider frequencies of searching of Keys
  - Keys: 10, 20, 30
  - Freqs: 3, 2, 5 (Supposition of frequencies)
- To find the optimal tree, multiply freqs with comparisons, (as shown in (b))



- (a) 22, (b) 19 (c) 18 (d) 17 (e) 18
- Thus, Tree (d) is optimal due to minimum cost based on freqs.

Optimal BST: Dynamic Programming (Example)

	1	2	3	4
KEYS	10	20	30	40
FREQUENCY	4	2	6	3

Keys are from 1 onward, while table is 0 onward, suitable for formula

Optimal BST: Dynamic Programming (Example)

i\j	0	1	2	3	4
0	0				
1		0			
2			0		
3				0	
4					0

l=j-i=0	I, j	C[0, 0]	C[1, 1]	C[2, 2]	C[3, 3]	C[4, 4]
0-0	(0,0)	0	0		0	0
1-1	(1,1)					
2-2	(2,2)					
3-3	(3,3)					
4-4	(4,4)					

i\j	0	1	2	3	4
0	0	4			
1		0	2		
2			0	6	
3				0	3
4					0

l=j-i=1	I, j		C[0,1]	C[1,2]	C[2, 3]	C[3, 4]
1-0	(0,1)	Key	10	20	30	40
2-1	(1,2)	freq	4	2	6	3
3-2	(2,3)					
4-3	(3,4)					

Optimal BST: Dynamic Programming (Example)

i\j	0	1	2	3	4
0	0	4	8^1		
1		0	2	10^3	
2			0	6	12^3
3				0	3
4					0

			C[0,2]		C[1,3]	C[2,4]
l=j-i=2	l,j	Key	10	20	start from 1 rather 0	
2-0	(0,2)	freq	4	2		
3-1	(1,3)					
4-2	(2,4)					

Total Cost=8

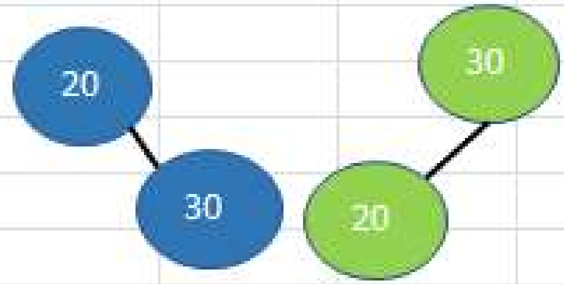
Total Cost=10

w(0,4)= $\sum_{i=1}^{4} f(i)$

root 1 as in tree-1 (LHS)	C[0,0]+	C[1,2]+	w[0,2]	In table to take cost and weight to take from Keys & freqs
	0	2	4+2	Total=8
				<
root 1 as in tree-1 (LHS)	C[0,1]+	C[2,2]+	w[0,2]	
	4	0	4+2	Total=10

Optimal BST: Dynamic Programming (Example)

C[1,3]			Tree Cost	
start from 2 rather 1			1	2
Key	20	30	2x1	1x6
freq	2	6	6x2	2x2
Different possible trees for 20 and 30			14	10
3rd Key as a root				

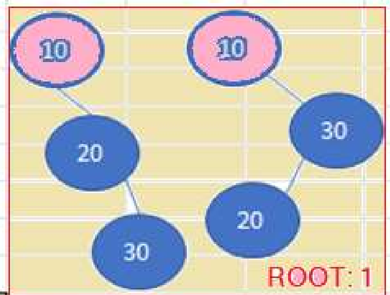


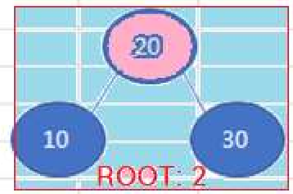
C[2,4]				
			Tree Cost	
start from 2 rather 1			1	2
Key	30	40	6x1	3x1
freq	6	3	3x2	6x2
Different possible trees for 20 and 30			12	15

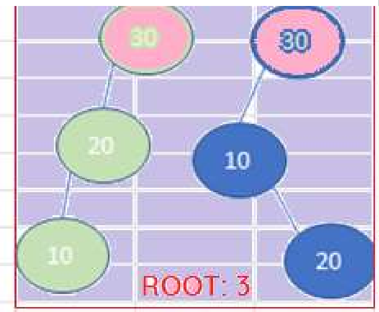
Optimal BST: Dynamic Programming (Example)

i\j	0	1	2	3	4	l=j-i=3	l,j
0	0	4	8^1	20^3		3-0	(0,3)
1		0	2	10^3	16^3	4-1	(1,4)
2			0	6	12^3		
3				0	3		
4					0		

^3 shows the node as minimum cost for a given tree







C[0,3] start from 1 to 3

C[0,3]=min { C[0,0]+C[1,3]+w[0,3], C[0,1]+C[2,3]+w[0,3], C[0,2]+C[3,3]+w[0,3]}

C[0,3]=min{0+10+12, 4+6+12, 8+0+12}

C[0,3]=min{22, 22, 20} w[0,3]=4+2+6=12

C[0,3]=20

C[1,4] Start from 2 to 4

C[1,4]=min { C[1,1]+C[2,4]+w[1,4], C[1,2]+C[3,4]+w[1,4], C[1,3]+C[4,4]+w[1,4]}

C[1,4]=min{0+12+11, 2+3+11, 10+0+11}

C[1,4]=min{23, 16, 21} w[1,4]=2+6+3=11

C[1,4]=16 Root is 3 here

Optimal BST: Dynamic Programming (Example)

i\j	0	1	2	3	4	l=j-i=4	I,j
0	0	4	8^1	20^3	26^3	4-0	(0,4)
1		0	2	10^3	16^3		
2			0	6	12^3		
3				0	3		
4					0		

	1	2	3	4	**w(0,4)**
Key	10	20	30	40	**15**
freq	4	2	6	3	

^3 shows the node as minimum cost for a given tree

C[0,4]=min { C[0,0]+C[1,4]+w[0,4], C[0,1]+C[2,4]+w[0,4], C[0,2]+C[3,4]+w[0,4], C[0,3]+C[4,4]+w[0,4]}

C[0,4]=min{0+16+15, 4+12+15, 8+3+15, 20+0+15}

C[0,4]=min{31, 31, 26, 35}

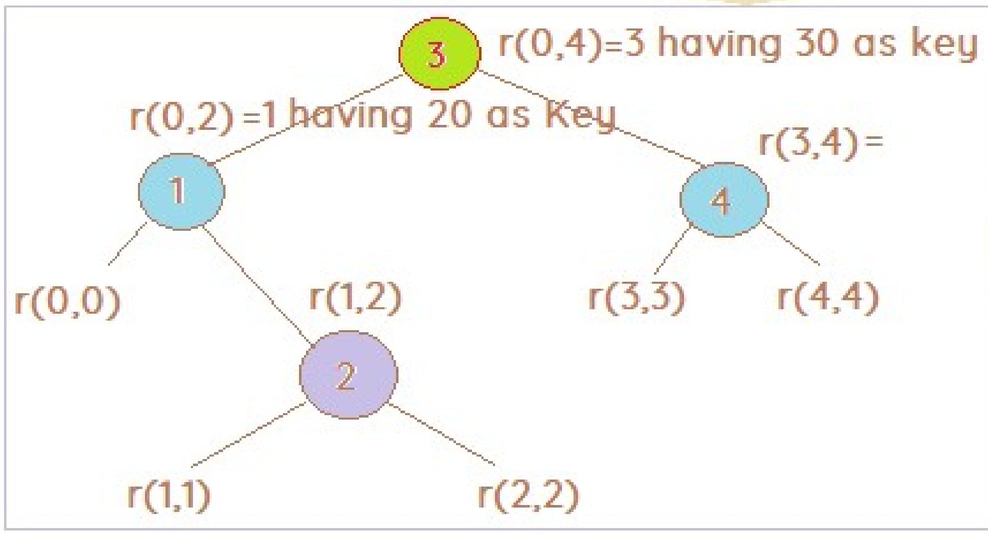
w[1,4]=2+6+3=11

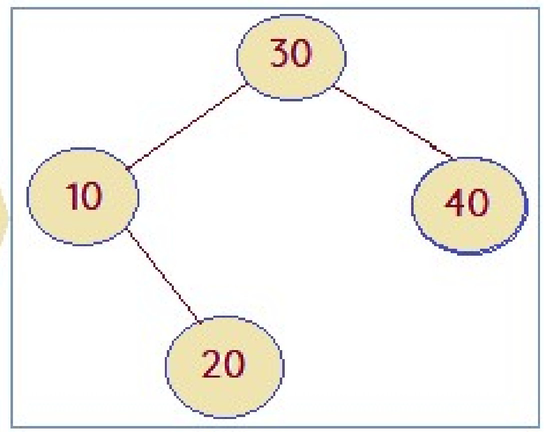
C[0,4]=26

Root is 3 here

Optimal Binary Search Tree

i\j	0	1	2	3	4
0	0	4	8^1	20^3	26^3
1		0	2	10^3	16^3
2			0	6	12^3
3				0	3
4					0





Optimal binary search trees

- n identifiers : $a_1 < a_2 < a_3 < \ldots < a_n$

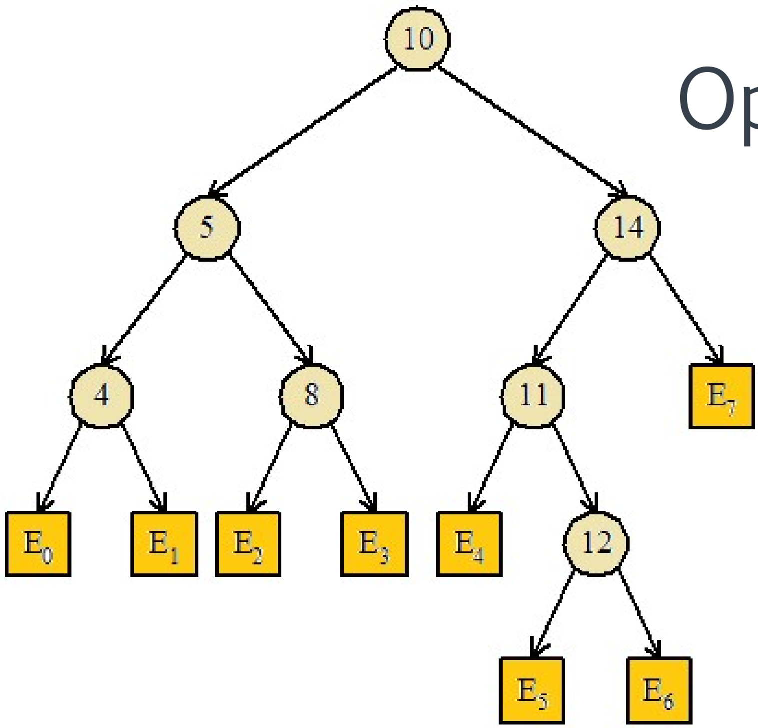
  P_i, $1 \le i \le n$: the probability that a_i is searched.

  Q_i, $1 \le i \le n$: the probability that x is searched

  where $a_i < x < a_{i+1}$ ($a_0 = -\infty$, $a_{n+1} = \infty$).

$$\sum_{i=1}^{n} P_i + \sum_{i=1}^{n} Q_i = 1$$

Optimal binary search trees



- Identifiers : 4, 5, 8, 10, 11, 12, 14
- Internal node : successful search, P_i
- External node : unsuccessful search, Q_i

- The expected cost of a binary tree:

$$\sum_{n=1}^{n} P_i * \text{level}(a_i) + \sum_{n=0}^{n} Q_i * (\text{level}(E_i) - 1)$$

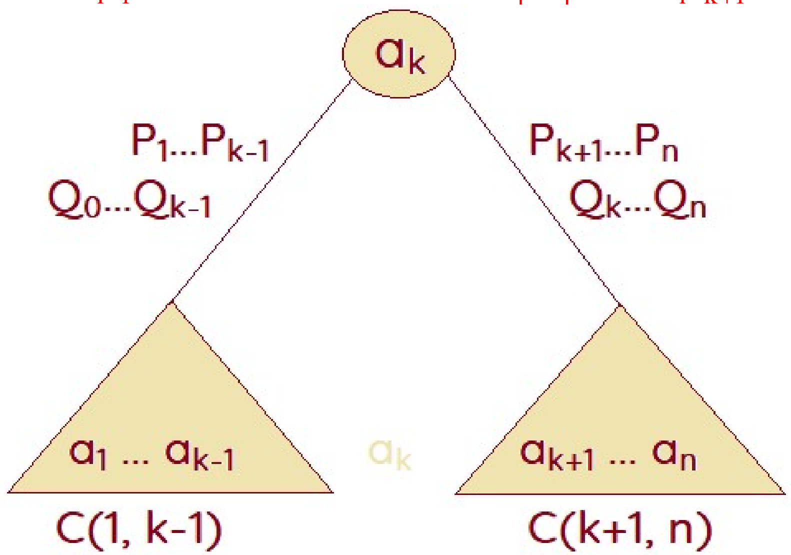
- The level of the root : 1

The dynamic programming approach

› Let C(i, j) denote the cost of an optimal binary search tree containing $a_i,\ldots,a_j$.

› The cost of the optimal binary search tree with a_k as its root :

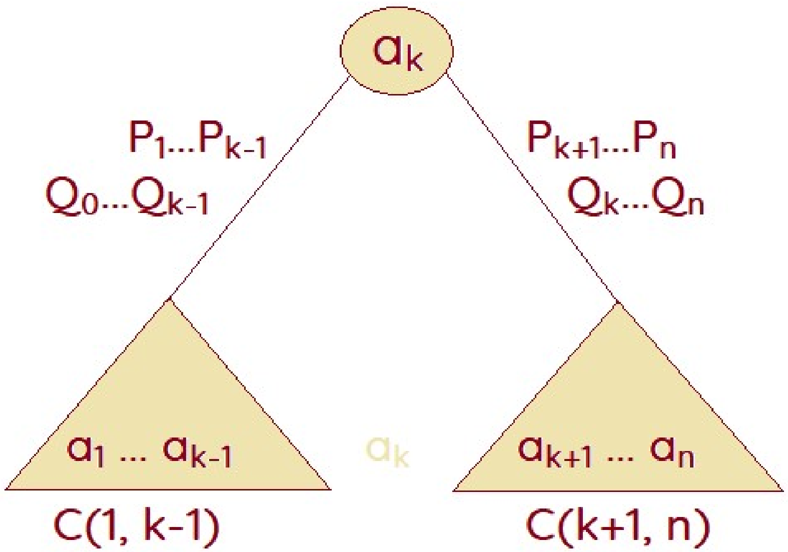
$$C(1,n) = \min_{1\le k\le n}\left\{P_k + \left[Q_0 + \sum_{i=1}^{k-1}(P_i + Q_i) + C(1,k-1)\right] + \left[Q_k + \sum_{i=k+1}^{n}(P_i + Q_i) + C(k+1,n)\right]\right\}$$



General formula

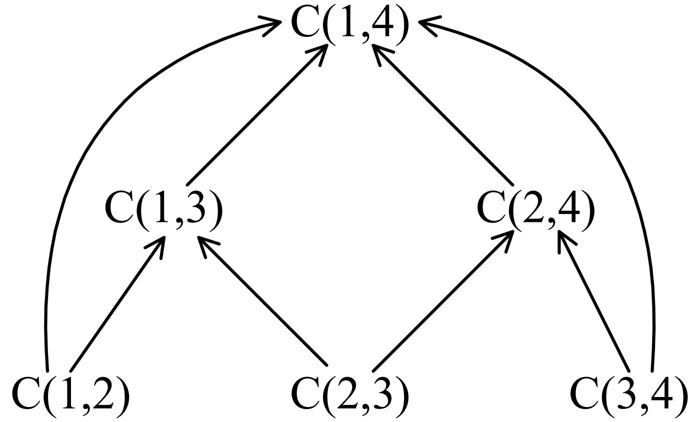
$$C(i, j) = \min_{i \le k \le j}\left\{P_k + \left[Q_{i-1} + \sum_{m=i}^{k-1}(P_m + Q_m) + C(i, k-1)\right] + \left[Q_k + \sum_{m=k+1}^{j}(P_m + Q_m) + C(k+1, j)\right]\right\}$$

$$= \min_{i \le k \le j}\left\{C(i, k-1) + C(k+1, j) + Q_{i-1} + \sum_{m=i}^{j}(P_m + Q_m)\right\}$$



Computation relationships of subtrees

- e.g. n=4



- Time complexity : $O(n^3)$

  when j-i=m, there are (n-m) C(i, j)'s to compute.

  Each C(i, j) with j-i=m can be computed in O(m) time.

$$O\left(\sum_{1\le m\le n} m(n-m)\right) = O(n^3)$$

Algorithm OBST
and its time complexity

$$T(n) = \sum_{m=1}^{n} \sum_{i=1}^{n-m+1} \sum_{j=i}^{n-1+1} \Theta(1)$$

$$= \sum_{m=1}^{n} \sum_{i=1}^{n-m+1} n = \sum_{m=1}^{n} n^2$$

$$= \Theta(n^3)$$

```
Algorithm OBST(p, q, n)
// e[1…n+1, 0…n ] : Optimal sub tree
// w[1…n+1,  0…n] : Sum of probability
// root[1…n, 1…n] : Used to construct OBST
for i ← 1 to n + 1 do
    e[i, i − 1] ← qi − 1
    w[i, i − 1] ← qi − 1
end
for m ← 1 to n do
    for i ← 1 to n − m + 1 do
        j ← i + m − 1
        e[i, j] ← ∞
        w[i, j] ← w[i, j − 1] + pj + qj
        for r ← i to j do
            t ← e[i, r − 1] + e[r + 1, j] + w[i, j]
            if t < e[i, j] then
                e[i, j] ← t
                root[i, j] ← r
            end
        end
    end
end
return (e, root)
```

Thank You!!!

Have a good day