	1	2	3	4
Keys →	10	20	30	40
frequency →	4	2	6	3

	0	1	2	3	4
0					
1					
2					
3					
4					

Search Tree

	1	2	3	4
Keys →	10	20	30	40
frequency →	4	2	6	3

$l = j - i = 0$

$0 - 0 = 0$

$1 - 1 = 0$

$2 - 2 = 0$

$3 - 3 = 0$

$4 - 4 = 0$

i \ j

	0	1	2	3	4
0					
1					
2					
3					
4					

Search Tree

	1	2	3	4
Keys →	10	20	30	40
frequency →	4	2	6	3

$l = j - i = 1$

$1 - 0 = 1 \; (0,1)$
$2 - 1 = 1 \; (1,2)$
$3 - 2 = 1 \; (2,3)$
$4 - 3 = 1 \; (3,4)$
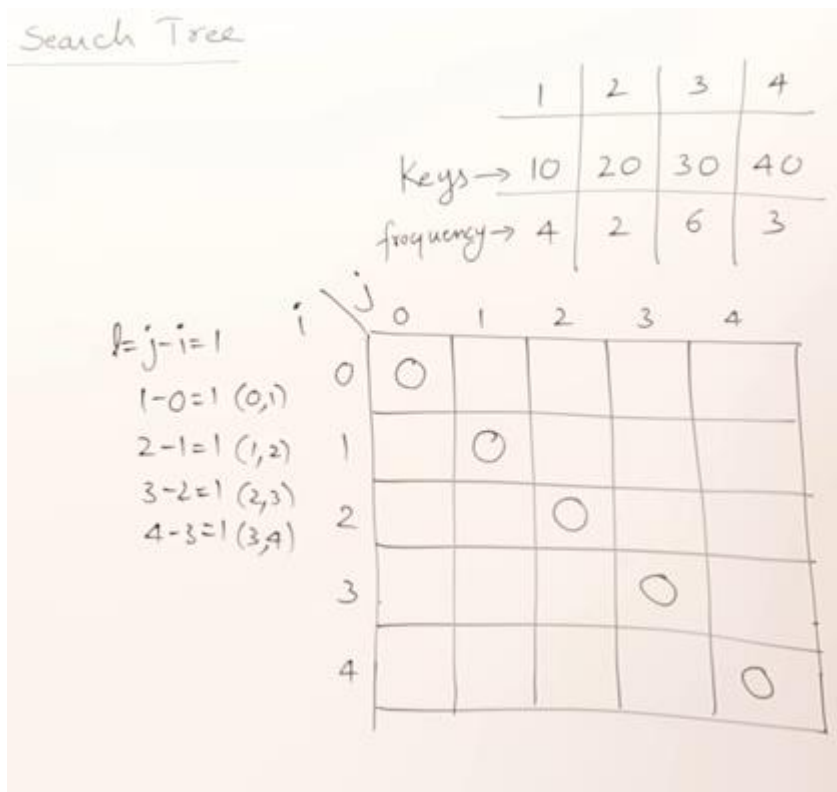
Optimal Binary Search Tree

As we know that in binary search tree, the nodes in the left subtree have lesser value than the root node and the nodes in the right subtree have greater value than the root node.

We know the key values of each node in the tree, and we also know the frequencies of each node in terms of searching means how much time is required to search a node. The frequency and key-value determine the overall cost of searching a node. The cost of searching is a very important factor in various applications. The overall cost of searching a node should be less. The time required to search a node in BST is more than the balanced binary search tree as a balanced binary search tree contains a lesser number of levels than the BST. There is one way that can reduce the cost of a binary search tree is known as an **optimal binary search tree**.

Let's understand through an example.

If the keys are 10, 20, 30, 40, 50, 60, 70
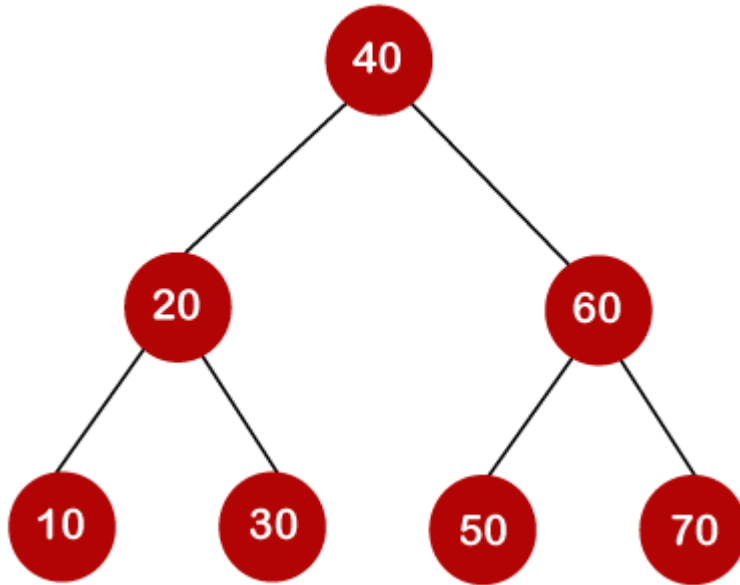
In the above tree, all the nodes on the left subtree are smaller than the value of the root node, and all the nodes on the right subtree are larger than the value of the root node. The maximum time required to search a node is equal to the minimum height of the tree, equal to logn.

Now we will see how many binary search trees can be made from the given number of keys.

For example: 10, 20, 30 are the keys, and the following are the binary search trees that can be made out from these keys.

The Formula for calculating the number of trees:

$$\frac{^{2n}C_n}{n+1}$$

When we use the above formula, then it is found that total 5 number of trees can be created.
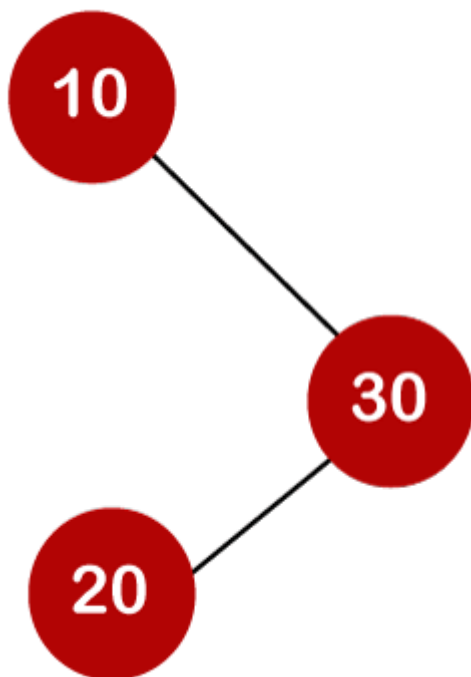
The cost required for searching an element depends on the comparisons to be made to search an element. Now, we will calculate the average cost of time of the above binary search trees.
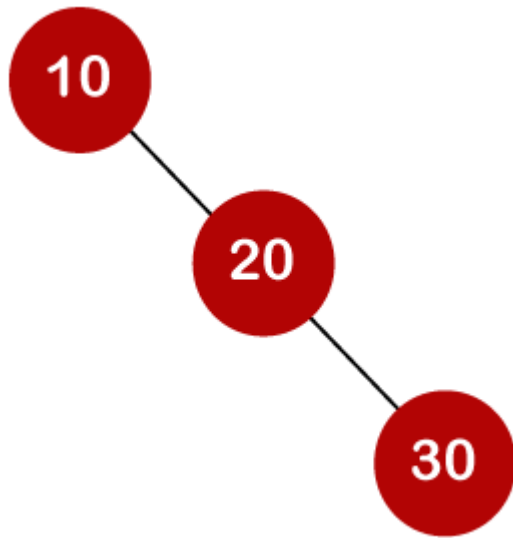
In the above tree, total number of 3 comparisons can be made. The average number of comparisons can be made as:

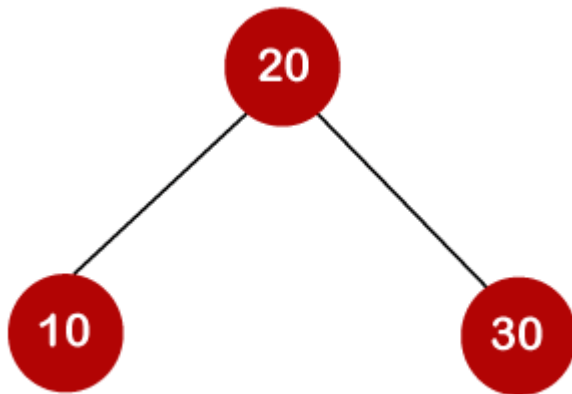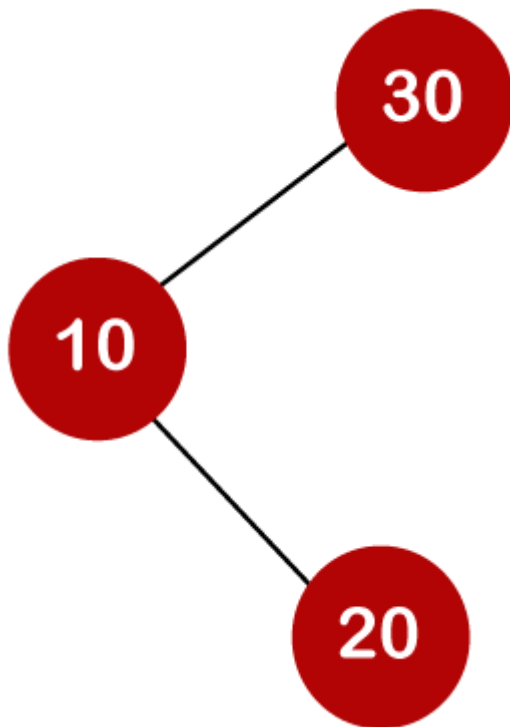$$average\ number\ of\ comparisons = \frac{1+2+3}{3} = 2$$

In the above tree, the average number of comparisons that can be made as:

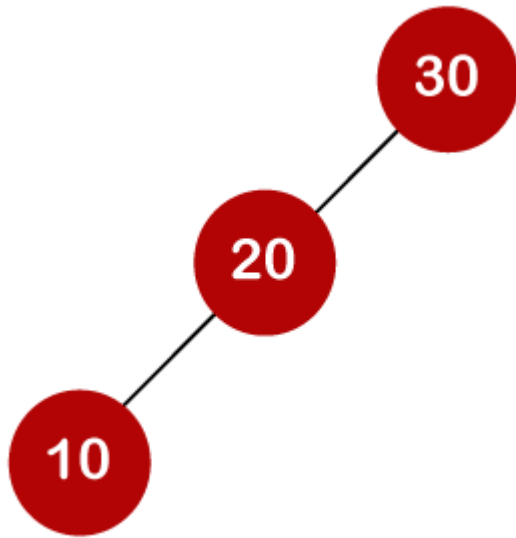$$\textit{average number of comparisons} = \frac{1+2+3}{3} = 2$$

In the above tree, the average number of comparisons that can be made as:

$$\textit{average number of comparisons} = \frac{1+2+2}{3} = 5/3$$

In the above tree, the total number of comparisons can be made as 3. Therefore, the average number of comparisons that can be made as:

$$average\ number\ of\ comparisons = \frac{1+2+3}{3} = 2$$

In the above tree, the total number of comparisons can be made as 3. Therefore, the average number of comparisons that can be made as:

$$average\ number\ of\ comparisons = \frac{1+2+3}{3} = 2$$

In the third case, the number of comparisons is less because the height of the tree is less, so it's a balanced binary search tree.

Till now, we read about the height-balanced binary search tree. To find the optimal binary search tree, we will determine the frequency of searching a key.

Let's assume that frequencies associated with the keys 10, 20, 30 are 3, 2, 5.

The above trees have different frequencies. The tree with the lowest frequency would be considered the optimal binary search tree. The tree with the frequency 17 is the lowest, so it would be considered as the optimal binary search tree.

Dynamic Approach

Consider the below table, which contains the keys and frequencies.

	1	2	3	4
Keys ⟶	10	20	30	40
Frequency ⟶	4	2	6	3

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

First, we will calculate the values where j-i is equal to zero.

When i=0, j=0, then j-i = 0

When i = 1, j=1, then j-i = 0

When i = 2, j=2, then j-i = 0

When i = 3, j=3, then j-i = 0

When i = 4, j=4, then j-i = 0

Therefore, c[0, 0] = 0, c[1 , 1] = 0, c[2,2] = 0, c[3,3] = 0, c[4,4] = 0

Now we will calculate the values where j-i equal to 1.

When j=1, i=0 then j-i = 1

When j=2, i=1 then j-i = 1

When j=3, i=2 then j-i = 1

When j=4, i=3 then j-i = 1

Now to calculate the cost, we will consider only the jth value.

The cost of c[0,1] is 4 (The key is 10, and the cost corresponding to key 10 is 4).

The cost of c[1,2] is 2 (The key is 20, and the cost corresponding to key 20 is 2).

The cost of c[2,3] is 6 (The key is 30, and the cost corresponding to key 30 is 6)

The cost of c[3,4] is 3 (The key is 40, and the cost corresponding to key 40 is 3)

	0	1	2	3	4
0	0	4			
1		0	2		
2			0	6	
3				0	3
4					0

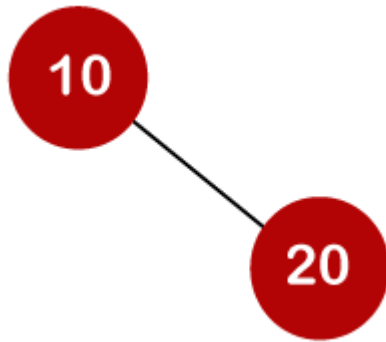**Now we will calculate the values where j-i = 2**

When j=2, i=0 then j-i = 2

When j=3, i=1 then j-i = 2

When j=4, i=2 then j-i = 2

In this case, we will consider two keys.

- When i=0 and j=2, then keys 10 and 20. There are two possible trees that can be made out from these two keys shown below:

In the first binary tree, cost would be: $4*1 + 2*2 = 8$

In the second binary tree, cost would be: $4*2 + 2*1 = 10$

The minimum cost is 8; therefore, $c[0,2] = 8$

	0	1	2	3	4
0	0	4	8		
1		0	2		
2			0	6	
3				0	3
4					0

- When i=1 and j=3, then keys 20 and 30. There are two possible trees that can be made out from these two keys shown below:

In the first binary tree, cost would be: $1*2 + 2*6 = 14$

In the second binary tree, cost would be: $1*6 + 2*2 = 10$

The minimum cost is 10; therefore, $c[1,3] = 10$

- When i=2 and j=4, we will consider the keys at 3 and 4, i.e., 30 and 40. There are two possible trees that can be made out from these two keys shown as below:

In the first binary tree, cost would be: $1*6 + 2*3 = 12$

In the second binary tree, cost would be: $1*3 + 2*6 = 15$

The minimum cost is 12, therefore, $c[2,4] = 12$

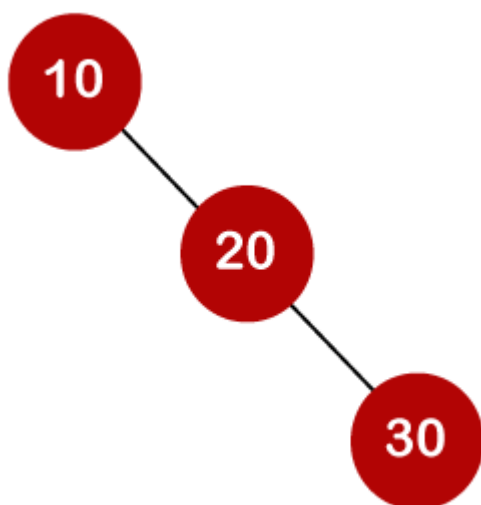| i $\diagdown$ 1	0	1	2	3	4
0	0	4	8^1		
1		0	2	10^3	
2			0	6	12^3
3				0	3
4					0

Now we will calculate the values when j-i = 3

When j=3, i=0 then j-i = 3

When j=4, i=1 then j-i = 3

- When i=0, j=3 then we will consider three keys, i.e., 10, 20, and 30.

The following are the trees that can be made if 10 is considered as a root node.

In the above tree, 10 is the root node, 20 is the right child of node 10, and 30 is the right child of node 20.
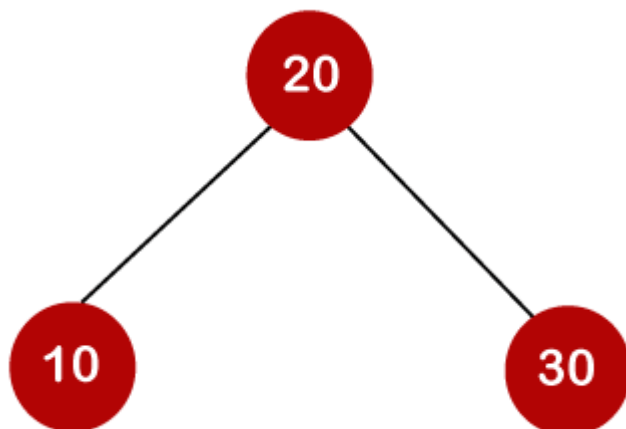
Cost would be: 1*4 + 2*2 + 3*6 = 26
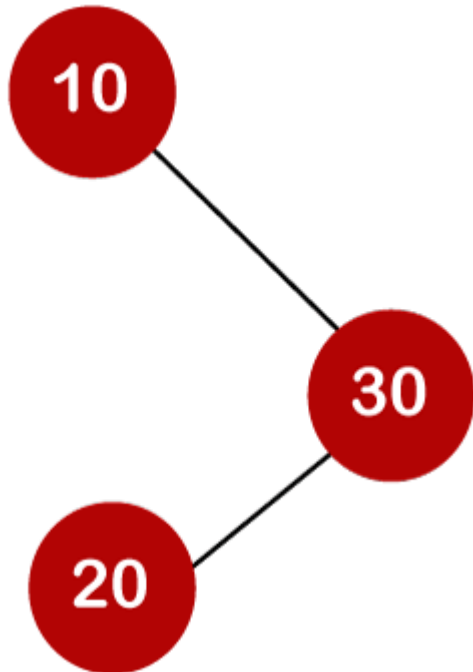
In the above tree, 10 is the root node, 30 is the right child of node 10, and 20 is the left child of node 20.
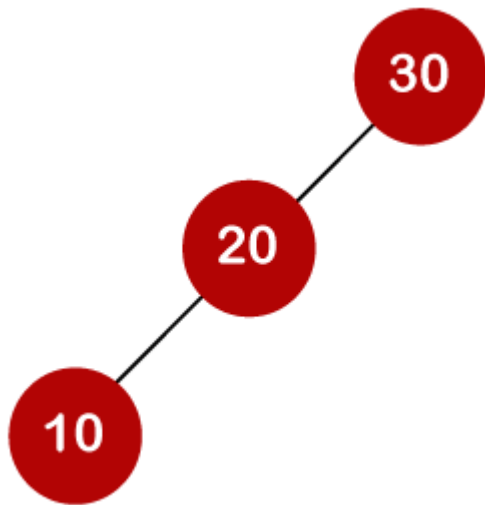
Cost would be: 1*4 + 2*6 + 3*2 = 22

The following tree can be created if 20 is considered as the root node.

In the above tree, 20 is the root node, 30 is the right child of node 20, and 10 is the left child of node 20.
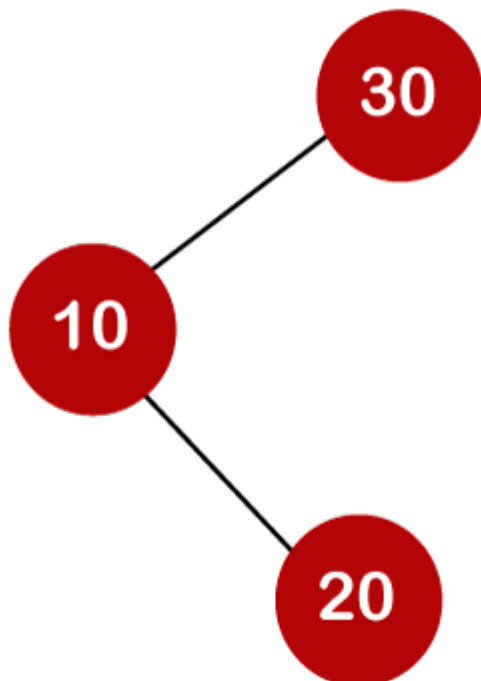
Cost would be: 1*2 + 4*2 + 6*2 = 22

The following are the trees that can be created if 30 is considered as the root node.

In the above tree, 30 is the root node, 20 is the left child of node 30, and 10 is the left child of node 20.

Cost would be: 1*6 + 2*2 + 3*4 = 22

In the above tree, 30 is the root node, 10 is the left child of node 30 and 20 is the right child of node 10.

Cost would be: 1*6 + 2*4 + 3*2 = 20

Therefore, the minimum cost is 20 which is the 3rd root. So, c[0,3] is equal to 20.

- When i=1 and j=4 then we will consider the keys 20, 30, 40

c[1,4] = min{ c[1,1] + c[2,4], c[1,2] + c[3,4], c[1,3] + c[4,4] } + 11

= min{0+12, 2+3, 10+0}+ 11

= min{12, 5, 10} + 11

The minimum value is 5; therefore, c[1,4] = 5+11 = 16

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

- **Now we will calculate the values when j-i = 4**

When j=4 and i=0 then j-i = 4

In this case, we will consider four keys, i.e., 10, 20, 30 and 40. The frequencies of 10, 20, 30 and 40 are 4, 2, 6 and 3 respectively.

w[0, 4] = 4 + 2 + 6 + 3 = 15

If we consider 10 as the root node then

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

= min {0 + 16} + 15= 31

If we consider 20 as the root node then

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

= min{4 + 12} + 15

= 16 + 15 = 31

If we consider 30 as the root node then,

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

= min {8 + 3} + 15

= 26

If we consider 40 as the root node then,
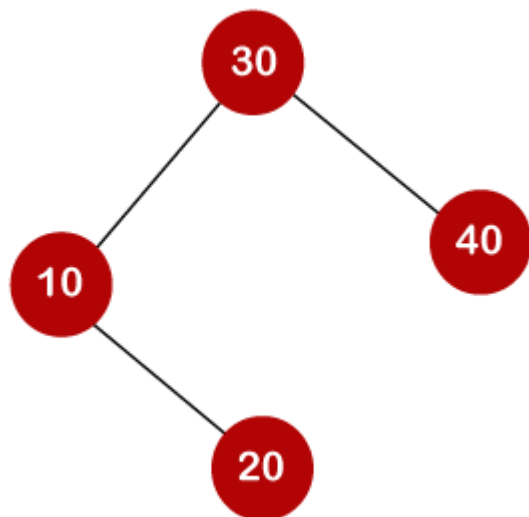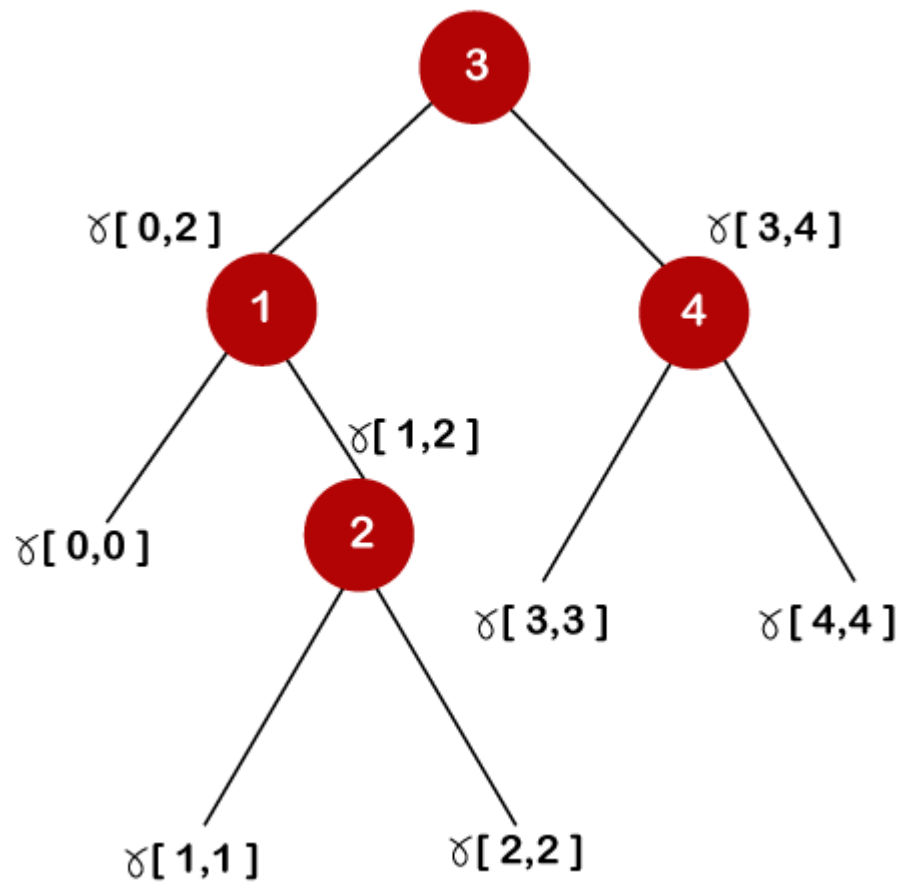
C[0,4] = min{c[0,3] + c[4,4]} + w[0,4]

= min{20 + 0} + 15

= 35

In the above cases, we have observed that 26 is the minimum cost; therefore, c[0,4] is equal to 26.

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

The optimal binary tree can be created as:

General formula for calculating the minimum cost is:

$C[i,j] = \min\{c[i, k-1] + c[k,j]\} + w(i,j)$