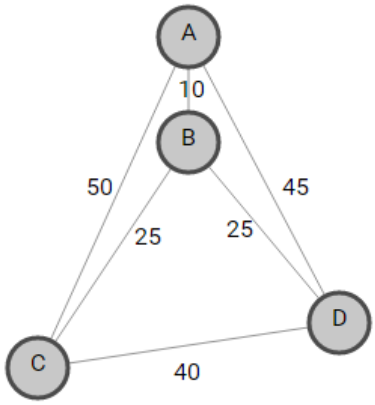
ARTICLE

TRAVELING SALESMAN PROBLEM USING BRANCH AND BOUND

"Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point."

The term *Branch and Bound* refer to all state-space search methods in which all the children of an E–node are generated before any other live node can become the E–node. E–node is the node, which is being expended. A node that has been generated and whose children are not yet been expanded is called a live-node. A node is called a dead node, which has been generated, but it cannot be expanded further. we expand the most promising node, which means the node which promises that expanding or choosing it will give us the optimal solution.

For example, consider the following graph. A TSP tour in the graph is `A –> B –> C –> D –> B –> A`. The cost of the tour is `10 + 25 + 40 + 25 + 10 = 100`.



We have a cost matrix defined by:

`C(i, j) = W(i, j), if there is a direct path from` C_i `to` C_j
`= INFINITY, if there is no direct path from` C_i `to` C_j

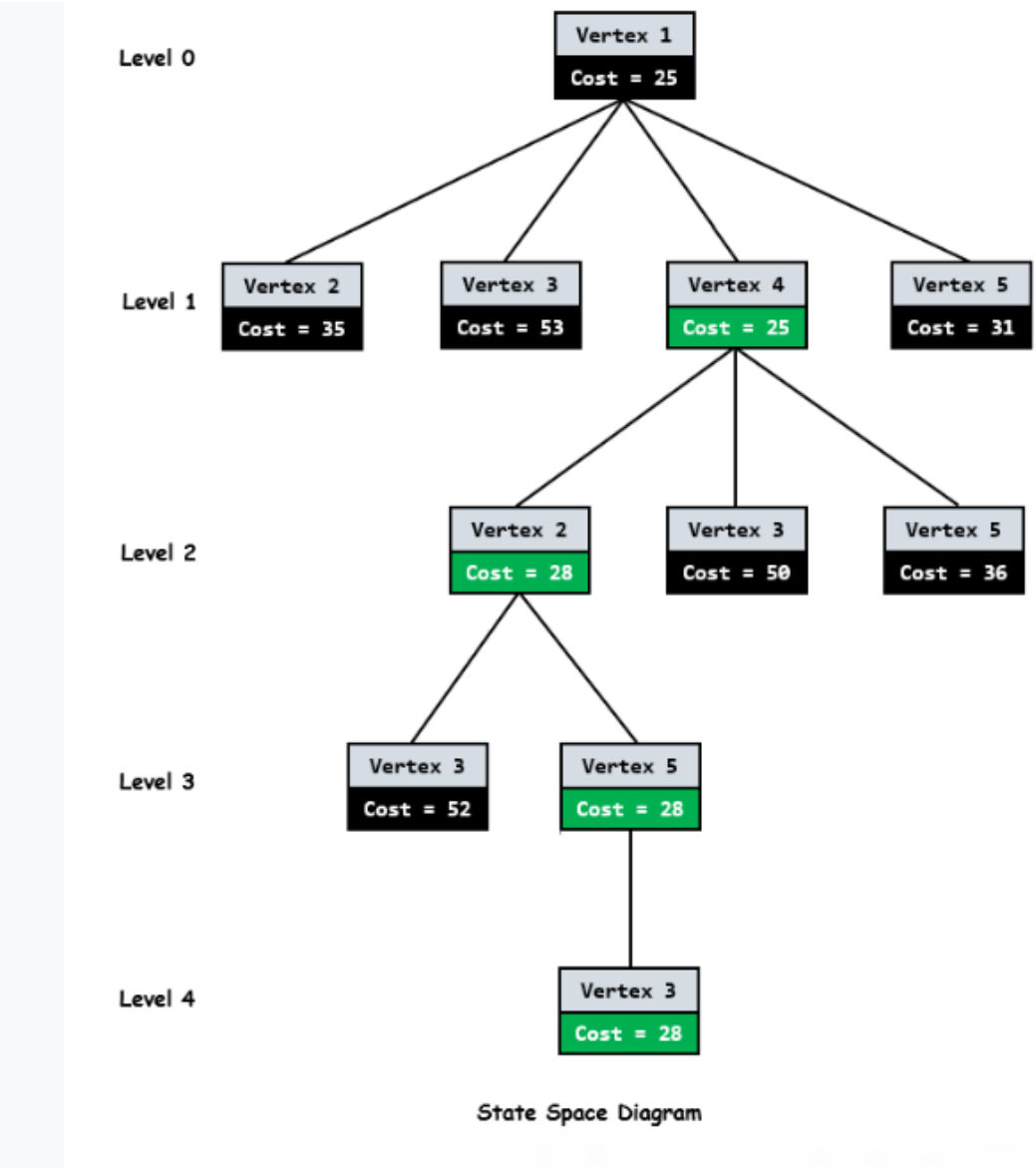
For example, consider the following cost matrix `M`,

	C0	C1	C2	C3	C4
C0	INF	20	30	10	11
C1	15	INF	16	4	2
C2	3	5	INF	2	4
C3	19	6	18	INF	3
C4	16	4	7	16	INF

Here,

```
C(0, 2) = 30
C(4, 0) = 16
C(1, 1) = INFINITY
```

Following is the state-space tree for the above TSP problem, which shows the optimal solution marked in green:



State Space Diagram

As we can see from the above diagram, every node has a cost associated with it. When we go from the city `i` to city `j`, the cost of a node `j` will be the sum of the parent node `i`, the cost of an edge `(i, j)`, and the lower bound of the path starting at node `j`.

Now, how do we calculate the lower bound of the path starting at any node?

In general, to get the lower bound of the path starting from the node, we reduce each row and column so that there must be at least one zero in each row and Column. We need to reduce the minimum value from each element in each row and column.

Let's start from the root node.
We reduce the minimum value in each row from each element in that row. The minimum in each row of cost matrix `M` is marked by blue `[10 2 2 3 4]` below.

```
INF  20  30  10  11
 15 INF  16   4   2
  3   5 INF   2   4
 19   6  18 INF   3
 16   4   7  16 INF
```

After reducing the row, we get the below reduced matrix,

```
INF  10  20   0   1
 13 INF  14   2   0
  1   3 INF   0   2
 16   3  15 INF   0
 12   0   3  12 INF
```

```
Cost = [10 2 2 3 4] + [1 0 3 0 0] = 25
```

Since we have discovered the root node `C0`, the next node to be expanded can be any node from `C1`, `C2`, `C3`, `C4`. Whichever node has a minimum cost will be expanded further. So, we have to find out the expanding cost of each node.

```
INF  10  17   0   1
 12 INF  11   2   0
  0   3 INF   0   2
 15   3  12 INF   0
 11   0   0  12 INF
```

Let's consider an edge from `0 -> 1`.
1. As we add an edge `(0, 1)` to our search space, set outgoing edges for city `0` to `INFINITY` and all incoming edges to city `1` to `INFINITY`. We also set `(1, 0)` to `INFINITY`.
So in a reduced matrix of the parent node, change all the elements in row `0` and column `1` and at index `(1, 0)` to `INFINITY` (marked in red).

```
INF  10  17   0   1
 12 INF  11   2   0
  0   3 INF   0   2
 15   3  12 INF   0
 11   0   0  12 INF
```

The resulting cost matrix is:

```
INF INF INF INF INF
INF INF  11   2   0
  0 INF INF   0   2
 15 INF  12 INF   0
 11 INF   0  12 INF
```

2. We try to calculate the lower bound of the path starting at node `1` using the above resulting cost matrix. The lower bound is `0` as the matrix is already in reduced form, i.e., all rows and all columns have zero value.
Therefore, for node 1, the cost will be:

```
Cost = cost of node 0 +
        cost of the edge(0, 1) +
        lower bound of the path starting at node 1
     = 25 + 10 + 0 = 35
```

Let's consider an edge from `0 –> 2`
1. Change all the elements in row `0` and column `2` and at index `(2, 0)` to `INFINITY` (marked in red).

```
INF  10  17   0   1
 12 INF  11   2   0
  0   3 INF   0   2
 15   3  12 INF   0
 11   0   0  12 INF
```

The resulting cost matrix is:

```
INF INF  INF  INF  INF
12  INF  INF    2   0
INF  3   INF    0   2
15   3   INF  INF   0
11   0   INF   12  INF
```

2. Now calculate the lower bound of the path starting at node `2` using the approach discussed earlier. The resultant matrix will be:

```
INF INF INF INF INF
  1 INF INF   2   0
INF   3 INF   0   2
  4   3 INF INF   0
  0   0 INF  12 INF
```

```
Cost = cost of node 0 +
       cost of the edge(0, 2) +
       lower bound of the path starting at node 2
     = 25 + 17 + 11 = 53
```

Let's consider an edge from `0 –> 3`.
1. Change all the elements in row `0` and column `3` and at index `(3, 0)` to `INFINITY` (marked in red).

```
INF  10  17   0   1
 12 INF  11   2   0
  0   3 INF   0   2
 15   3  12 INF   0
 11   0   0  12 INF
```

The resulting cost matrix is:

```
INF  INF  INF INF  INF
 12  INF  11  INF   0
  0   3   INF  INF   2
 INF  3    12  INF   0
 11   0    0   INF  INF
```

2. Now calculate the lower bound of the path starting at node `3` using the approach discussed earlier. The lower bound of the path starting at node `3` is `0` as it is already in reduced form, i.e., all rows and all columns have zero value.
Therefore, for node 3, the cost will be

```
Cost = cost of node 0 +
       cost of the edge(0, 3) +
       lower bound of the path starting at node 3
     = 25 + 0 + 0 = 25
```

Similarly, we calculate the cost of `0 –> 4`. Its cost will be `31`.
Live nodes `1`, `2`, `3`, and `4` have costs `35`, `53`, `25`, and `31`

The minimum among them is node `3`, having cost `25`. So, node `3` will be expanded further, as shown in the state-space tree diagram. After adding its children to the list of live nodes, find a live node with the least cost and expand it. Continue the search till a leaf is encountered in the space search tree. If a leaf is encountered, then the tour is completed, and we will return to the root node.