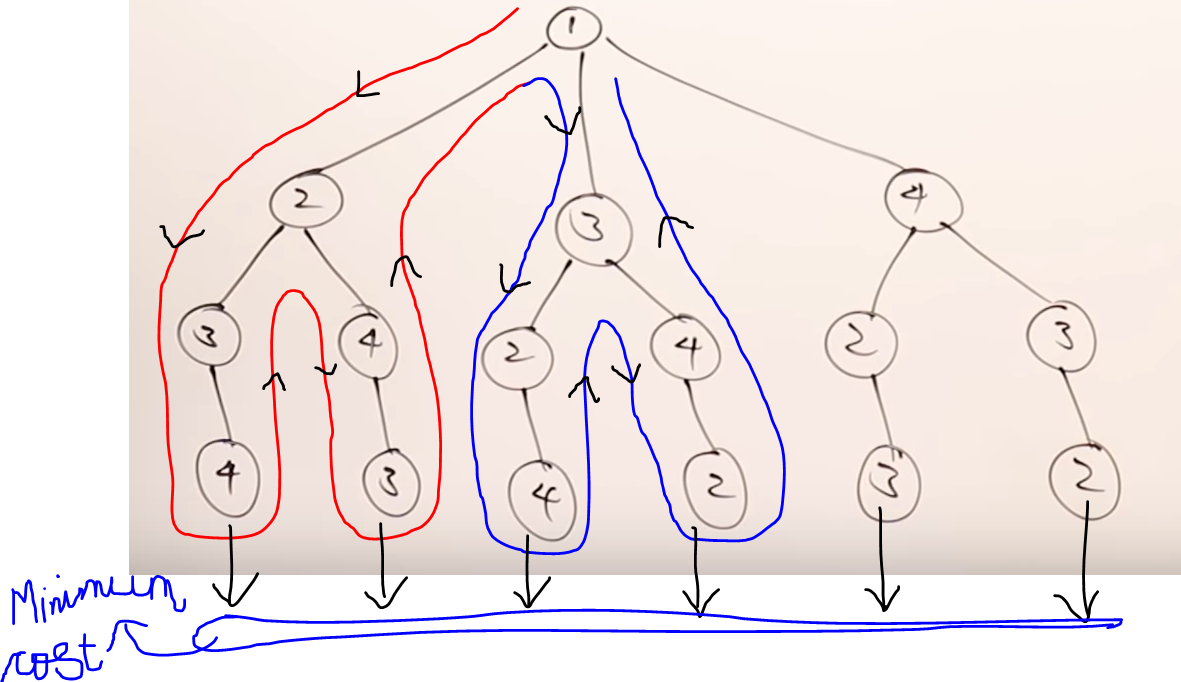
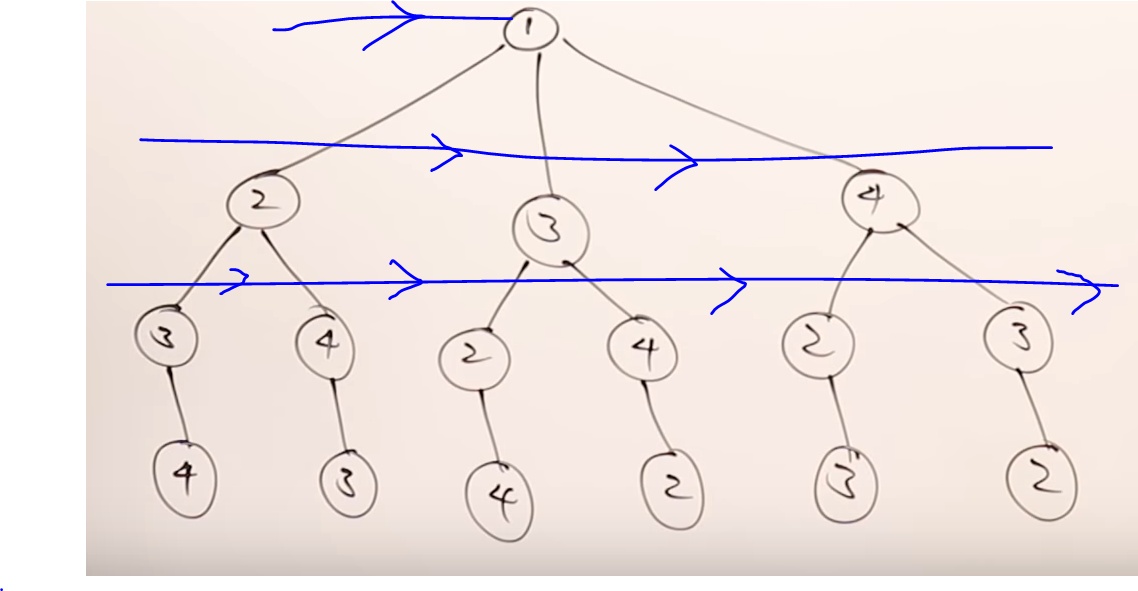
# **Travelling Sales Person problem**

Back-tracking and Branch and Bound are similar. They use the same state space tree.

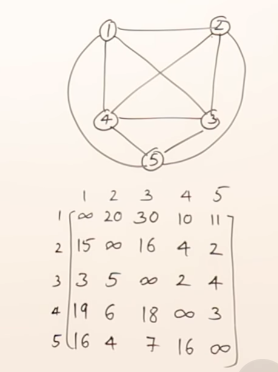
**Back-tracking** 🡪 Used to solve permutations and combination problems. This will give solution to all the paths and from that we should find the minimum. So Time consuming and cost is extremely maximum.  


**Brach bound** 🡪 Used to solve optimization problems (i.e minimization / maximization)

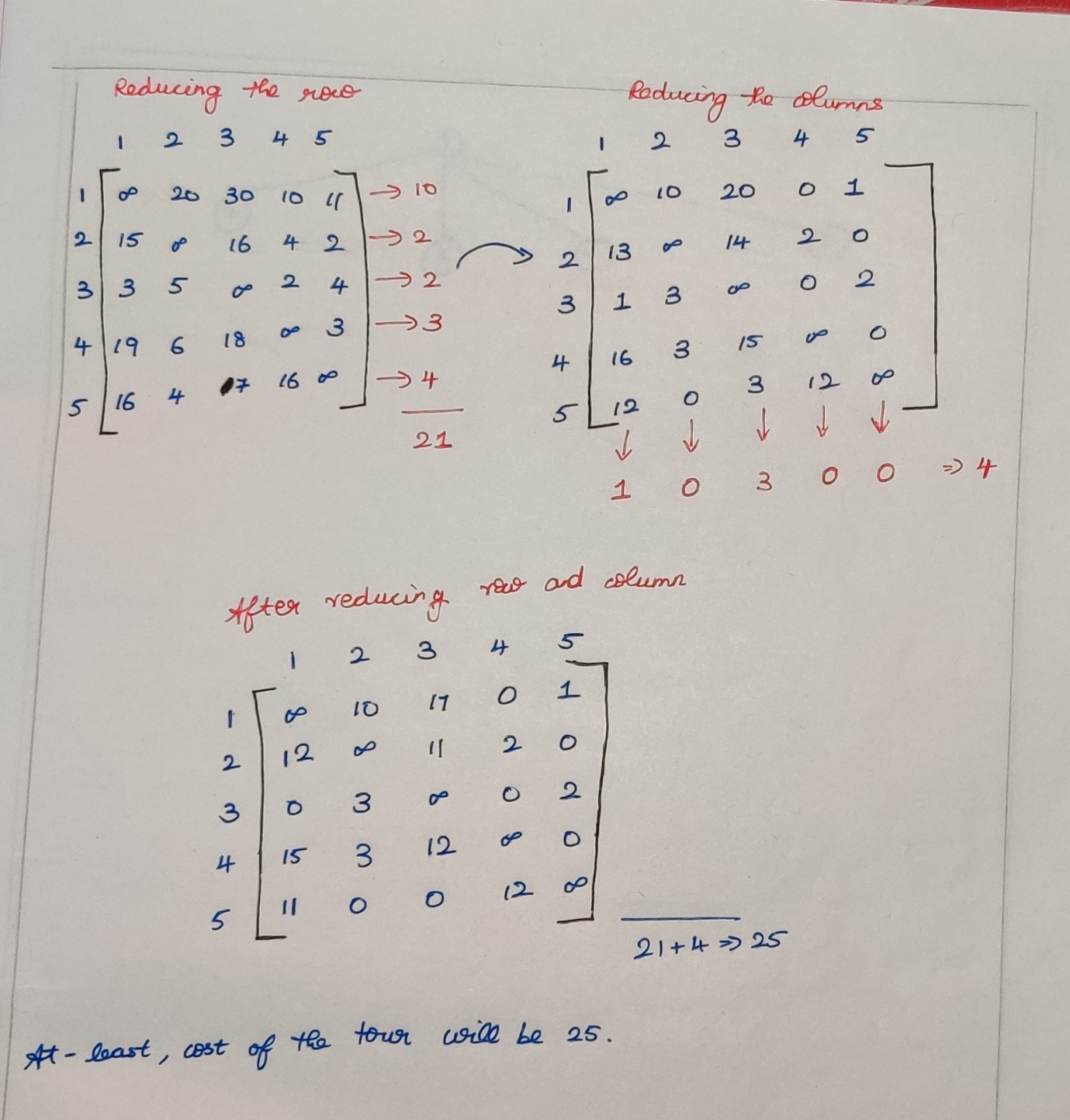


In the 2nd row, will find the minimum between vertex-2,3,4. If the cost of traversing the nodes is minimum at vertex-3, then we will discard the remaining the vertices-2 and 4. So there is no need of going into vertex-2 and vertex-3 and its child.  
So Time consuming and cost is extremely minimum and low.

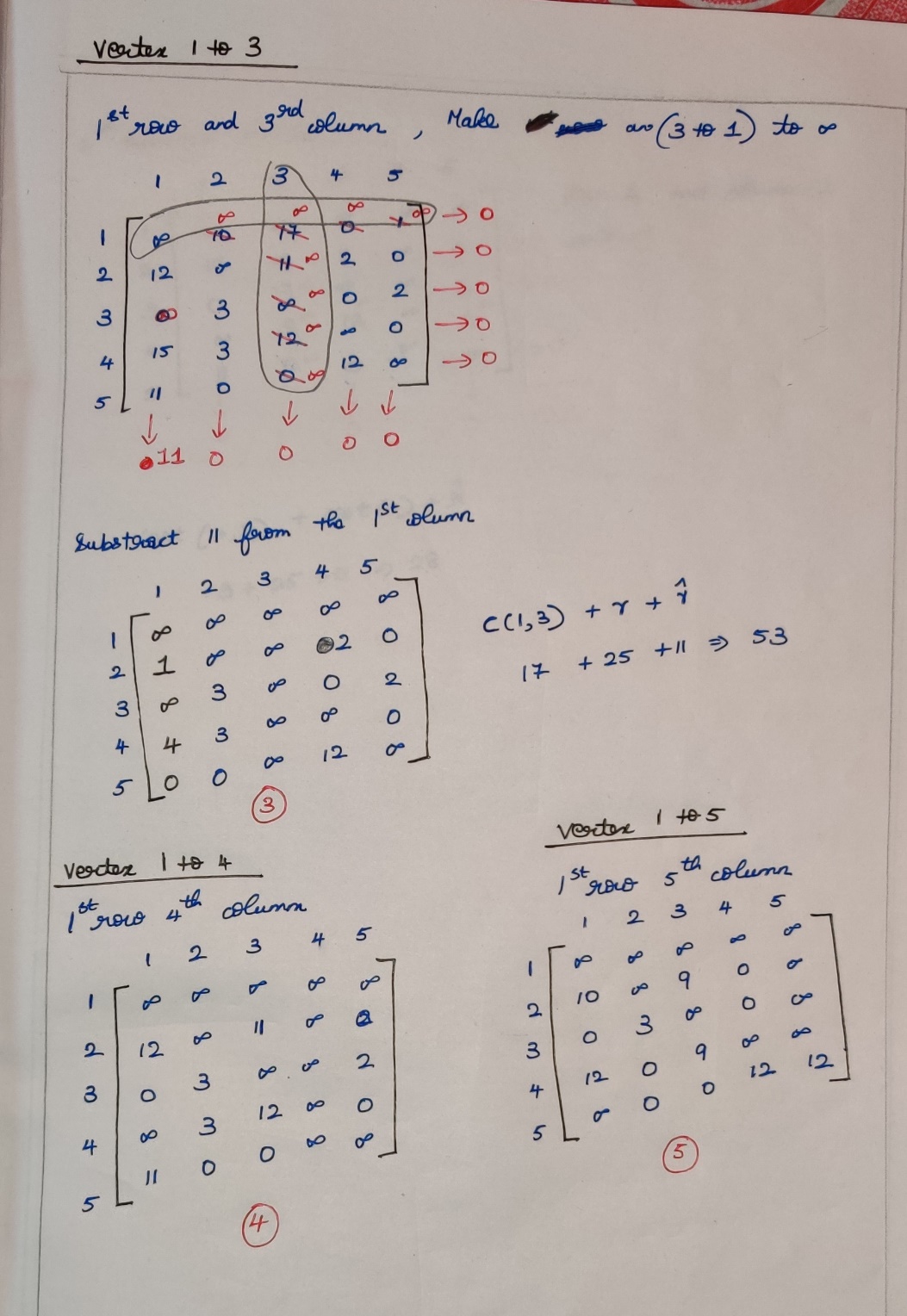
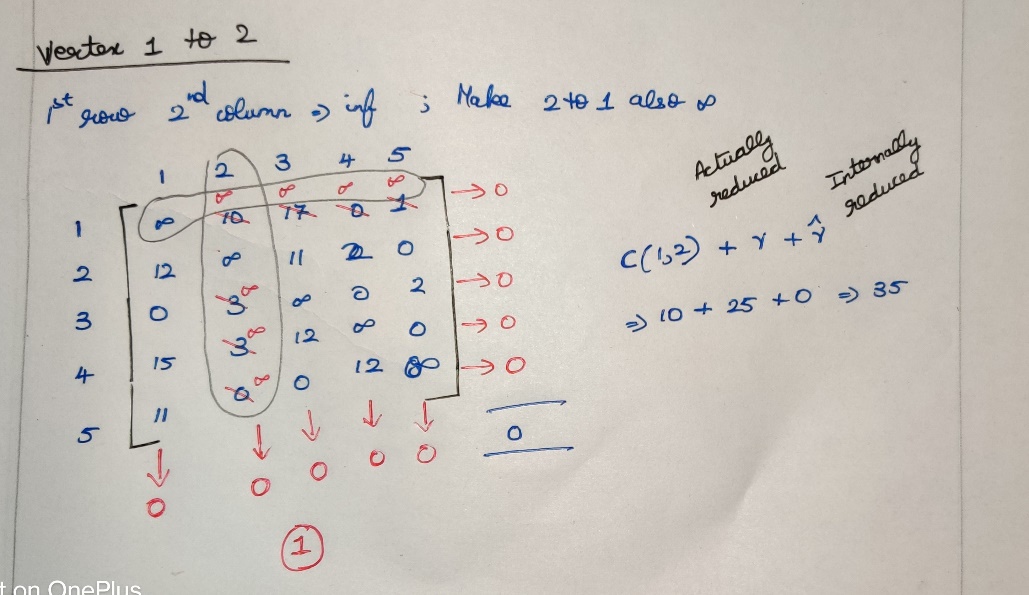
Question

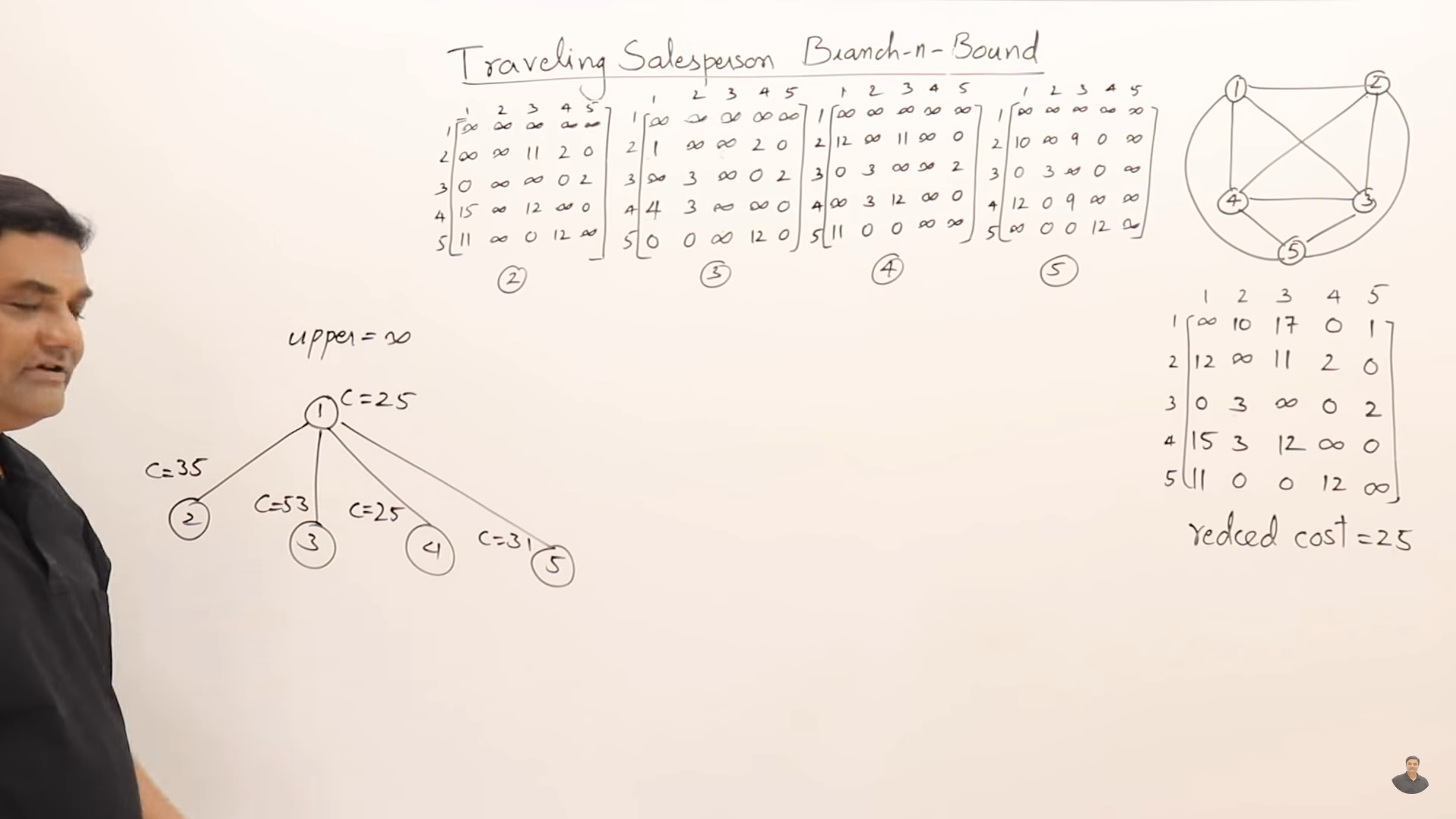


Step-1 Reducing the row and column and subtract   
(take the minimum value out)

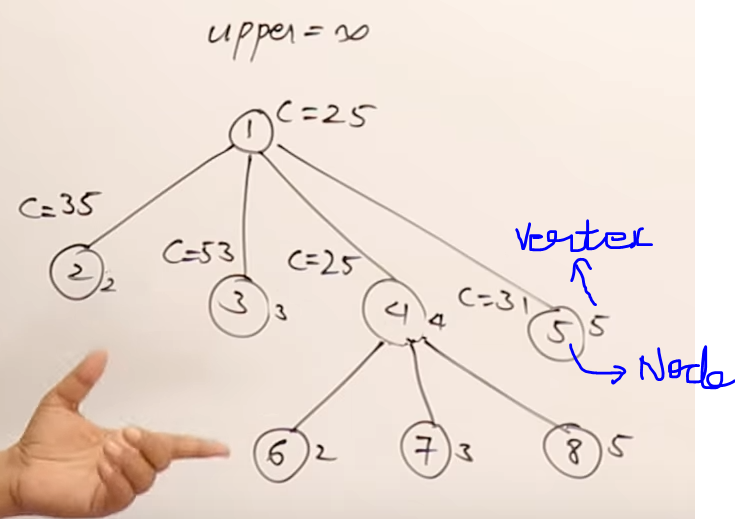


***Now, at-least the cost of the tour is 25.***

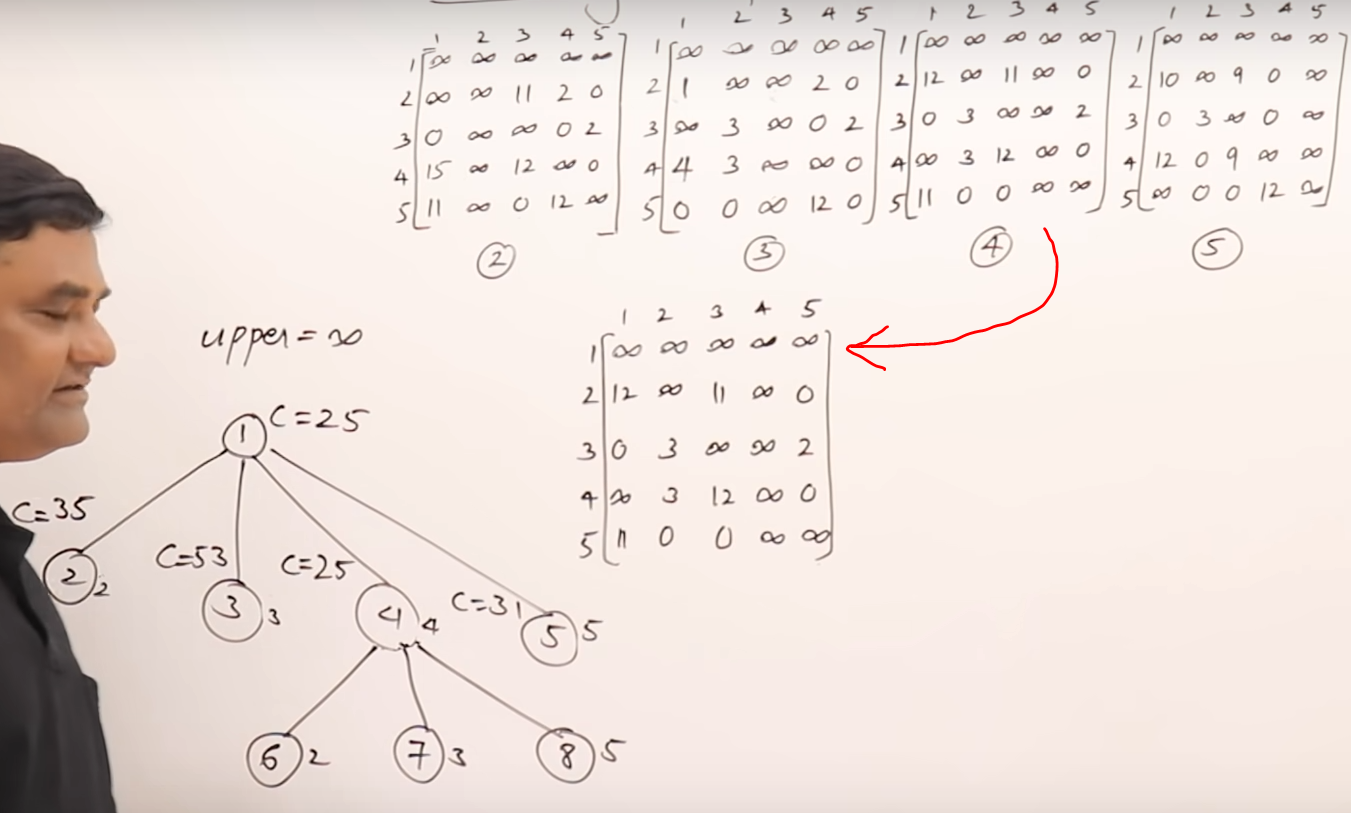


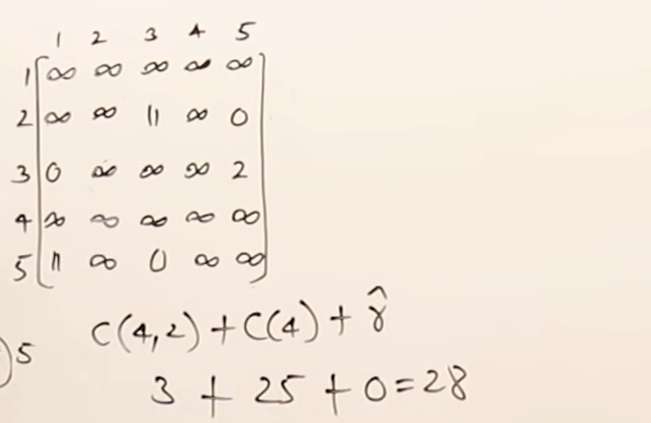
Now from vetex-1 we reached to 2,3,4,5

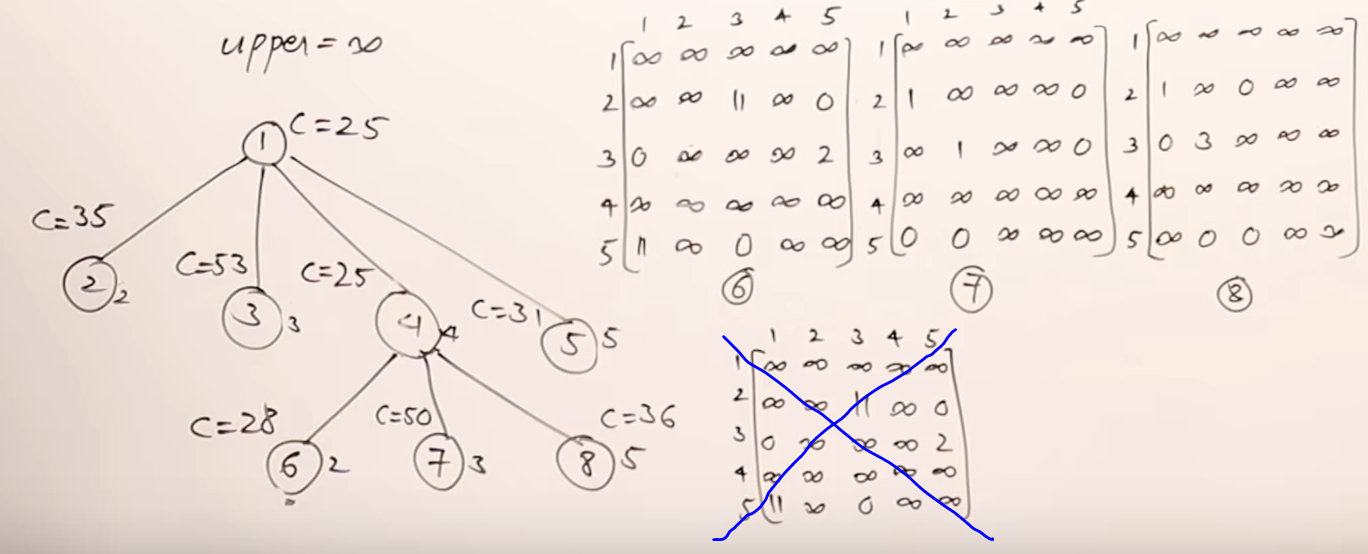
If we start exploring from vertex-2,3,4,5. Then we are following  
***first-first out branch and bound***. But we want optimal solution.  
So we are going to use ***Least Cost branch and bound***.



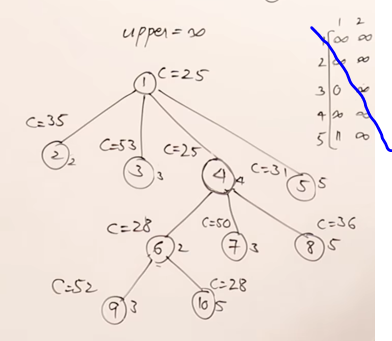
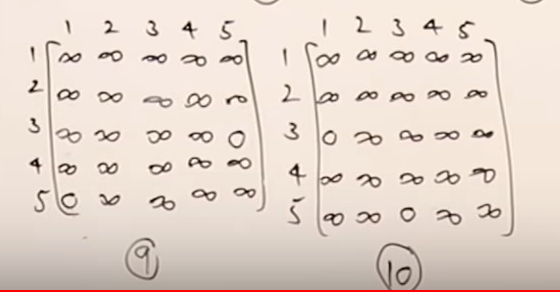
In 1st row (35,53,25,31) 🡪 25 is minimum.   
Now we want to find (4 to 2 , 4 to 3 , 4 to 5) by taking the matrix-4.  
Take the vertex-4 th matrix as the main reference.



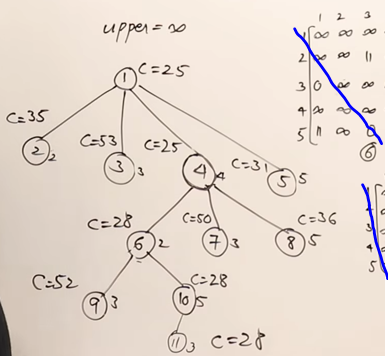
Vertex-4 to Vertex-2 ( 4th row and 2nd column 🡪 inf,  
from 2 to 1 🡪 inf )  
  
Vertex-4 to Vertex-3 ( 4th row and 3rd column 🡪 inf,  
from 3 to 1 🡪 inf )  
Vertex-4 to Vertex-5 ( 4th row and 5th column 🡪 inf,  
from 5 to 1 🡪 inf )



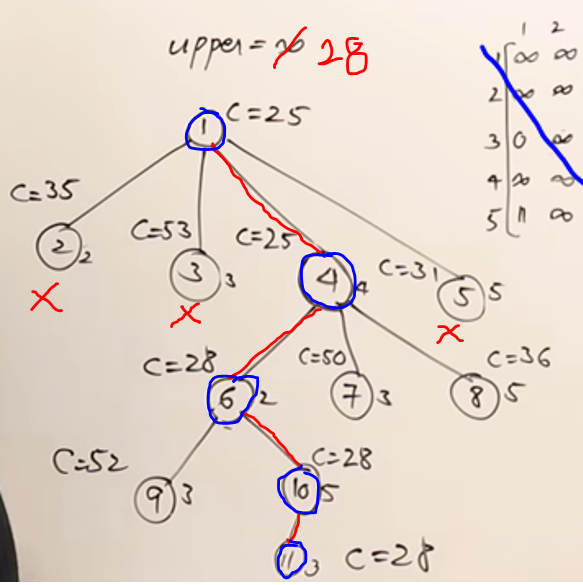
In 2nd row (28,50,36) 🡪 28 is minimum.   
Now we want to find (2 to 3 and 2 to 5)  
Take the vertex-6th matrix as the main reference.

In 3rd row (52,28 ) 🡪 28 is minimum.   
Now we want to find (5 to 3)by taking the matrix-10 (i.e vertex-3 ).  
Since the vertex-3 is only remaining.



Now we got the final graph. (1 🡪 4 🡪 2 🡪 5 🡪 3 and then to 1)   
The final cost=28.



# **0/1 Knapsack Problem**

