26/2/24

4. Back Tracking

Back Tracking:

→ Back Tracking is one of the technique which is supported by dynamic programming approach.

→ In Back Tracking at the time of finding optimal solution if any step is wrong there is a possibility to move backward from the current step.

→ The Applications of Back Tracking are

1. n-Queen Problem
2. Graph coloring problem
3. Hamiltonion Cycle
4. Sum of Subsets problem

Now,

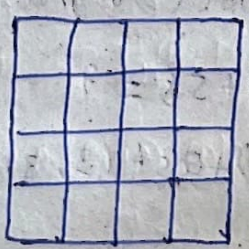
1. n-Queen Problem:

In this problem we are having 'n' no. of Queens and nxn chessboard, the problem is you have to place all the Queens into chessboard without colliding no two Queens as Row wise, Column wise, Diagonal wise.

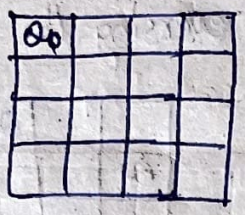
The Application of n-Queen Problem is 4-Queen problem, 8-Queen Problem.

4-Queen Problem:

In this problem we are having 4 no. of Queens (Q_0, Q_1, Q_2, Q_3) and 4x4 chessboard, we have to place all the Queens into chessboard.



Step-1: Now, we are placing Q_0 at (1,1) [1st row, 1st column]



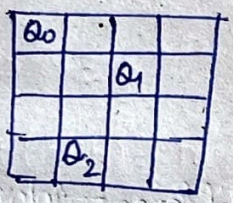
Step-2:

Q_1 is unable to place at 1st row, 1st column and (2,2)

Now, we will place Q_1 at (2,3)

Step-3:

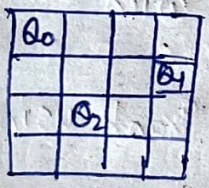
Now, we want to place Q_2 at (4,2)



Step-4:

we are unable to place Q_3 into chessboard, we have to apply backtracking on Q_2 and Q_1.

Q_1 is placed at (2,4) and Q_2 is placed at (3,2)



Step-5:

Now, we want to place Q_3 into chessboard, there is no cell for placing Q_3 without collision.

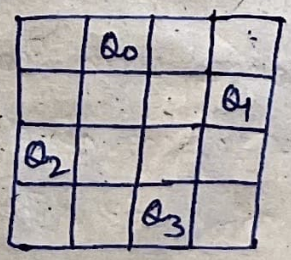
Now we are applying backtracking on Q_2, Q_1, Q_0

Q_0 is Placed at (1,2)

Q_1 is Placed at (2,4)

Q_2 is Placed at (3,1)

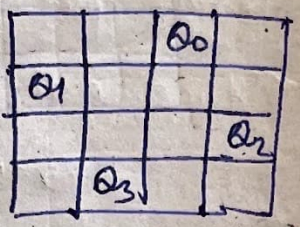
Now, Q_3 is Placed at (4,3)



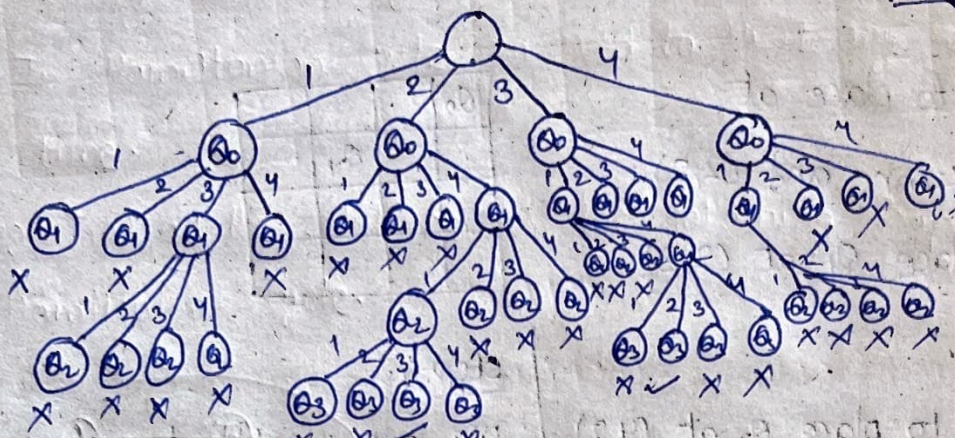
$\therefore (Q_0, Q_1, Q_2, Q_3) = (2, 4, 1, 3)$

The optimal solution is $(Q_0, Q_1, Q_2, Q_3) = (2, 4, 1, 3)$

We will get another optimal solution by reversing the above solution. $(Q_0, Q_1, Q_2, Q_3) = (3, 1, 4, 2)$



State Space Tree



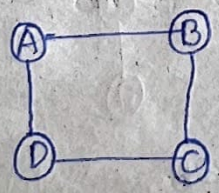
Ilzley

~~4. Sum of subset~~

2. Graph Coloring Problem:

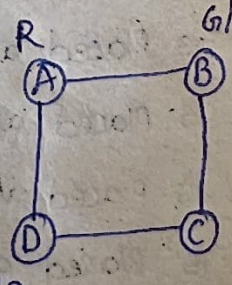
In Graph Coloring Problem we are having different colors and we need to color the graph by obeying the following conditions. "No two adjacent vertices are colored with same color".

eg: Find all possible solutions for the following graph using Graph coloring where colors are Red, Green, Blue



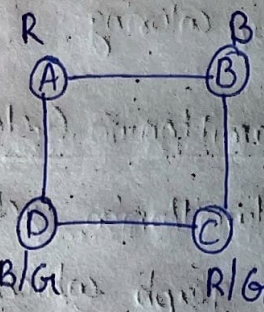
Step-1:

Vertex A is colored with Red color then the adjacent vertices of A are not allowed to color with Red, (B, D) are colored with either Green / Blue color.



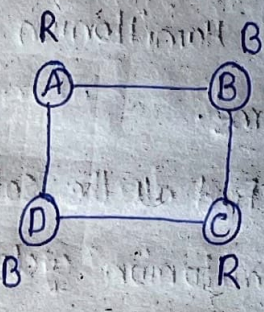
Step-2:

If B is colored with Blue color then the adjacent vertices of B are not allowed to color with Blue, the vertex C is colored with either Green/Red.

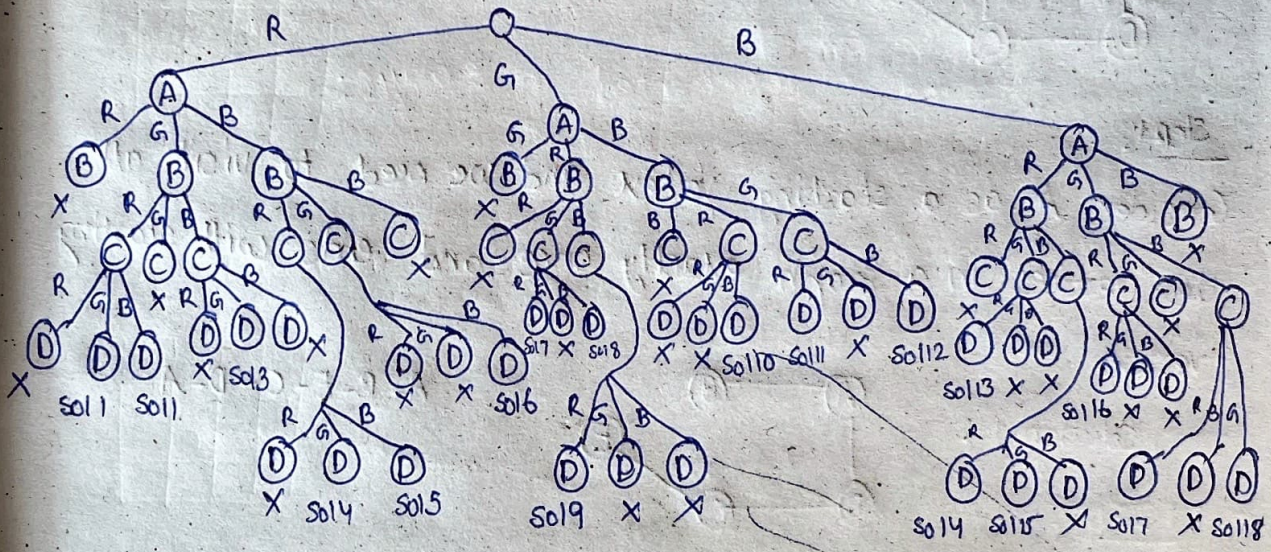


Step-3:

If C is colored with Red color then, the adjacent vertices of C are not allowed to color with Red, D is colored with either Green/Blue color



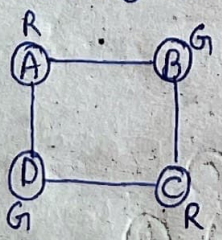
→ For finding all possible solutions we are constructing a state space tree



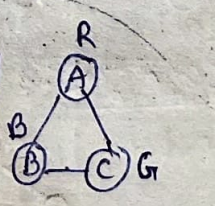
Chromatic Number:

The minimum no. of colors required to color the graph is known as a chromatic number.

eg:



Chromatic number = 2



Chromatic number = 3

Applications of Graph coloring:

1) Sudoku Puzzle
2) Registers Allocation strategies.

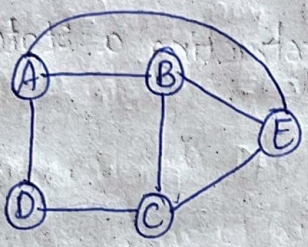
3) Map coloring

3. Hamiltonion Cycle:

→ Hamiltonion cycle is nothing but a closed loop formed by the graph which contains all the vertices in a graph and starting and ending vertices should be same.

→ In Hamiltonion cycle no need to visit any vertices second time.

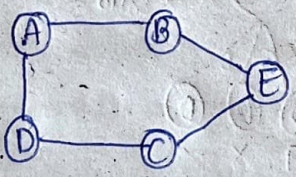
eg: Find all the solutions for the following graph using Hamiltonion cycle.



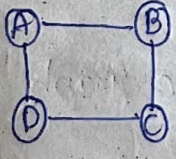
Step 1:

Consider A as a starting vertex and we need to visit all the vertices in a graph exactly once and ends with starting vertex.

Solution-1 :

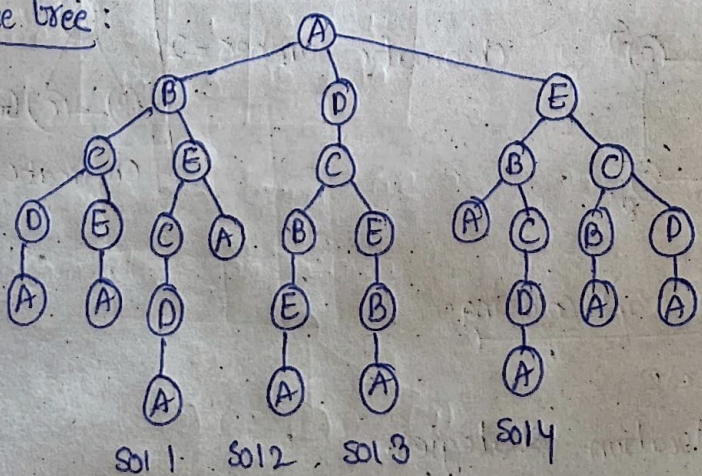


A - B - E - C - D - A



It is not a Hamiltonion cycle because (E) is not visited.

State Space tree:



4, Sum of subsets problem

→ In this problem, we are going to add subsets and find all the possible solutions when the sum of subsets is equal to m, where m is maximum capacity.

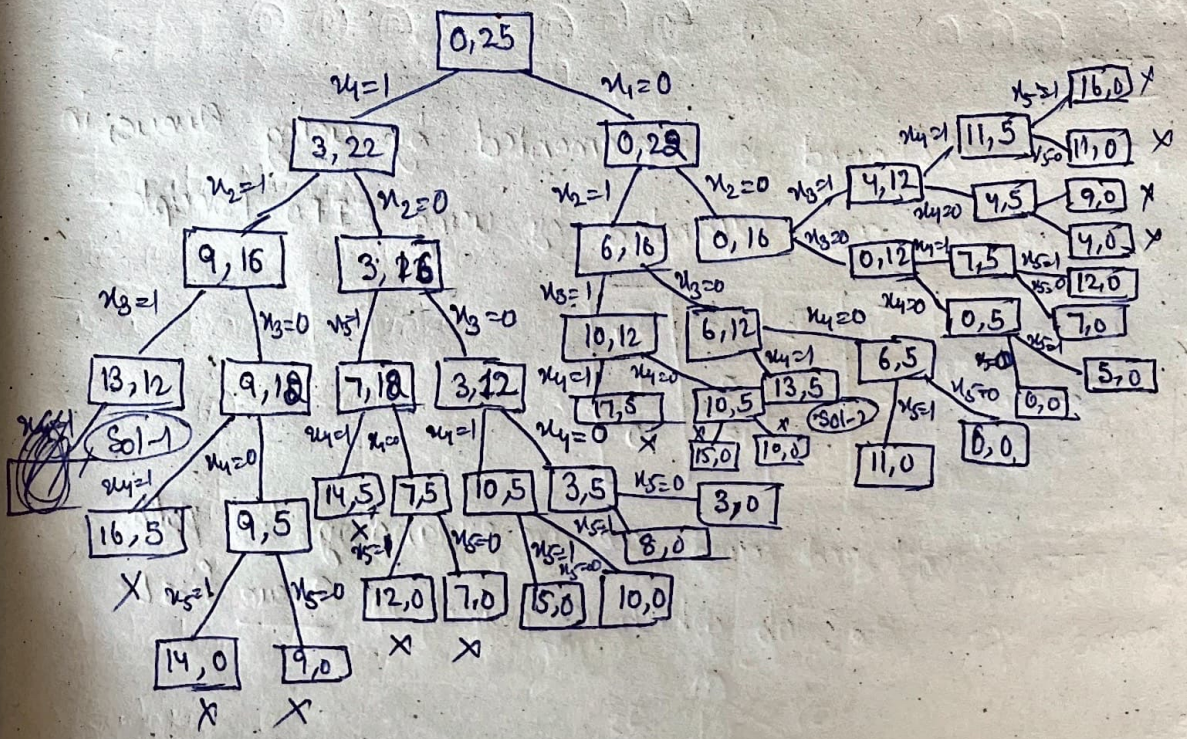
→ If the subset is included into sum then it is represented as 1 otherwise represented as 0.

→ For finding all the possible solutions, we need to construct a state space tree, the initial node of state space tree contains ~~sum~~ Sum, total sum of subsets. Initially (0, Σ)

→ If the subset is included then sum is represented the value of subset and total sum is reduced from the value of subset, if the subset is not considered then its value is represented as 0, it is not considered in total sum of subsets.

Likewise, we need to find all the possible solutions.

eg: Find all the possible solutions for the following subsets where m = 13, $\{x_1, x_2, x_3, x_4, x_5\} = \{3, 6, 4, 7, 5\}$



11/3/24

5. Branch & Bound

Branch and Bound:

→ Branch & Bound is one of the technique which is used for finding optimal solution to any problem.

→ Branch & Bound is used to minimize ~~the~~ a problem whereas Greedy & Dynamic programming is working for both minimal and maximum solutions.

→ In Branch & Bound, we need to construct a State Space tree for solving any problem, State space tree is constructed ~~Depth wise~~ Bedth wise (BFS). Whereas in Backtracking we're using DFS.

→ Branch & Bound is implemented in 3 ways:

1) FIFO Branch & Bound
2) LIFO Branch & Bound
3) Least Cost Branch & Bound

Now,

1) FIFO:

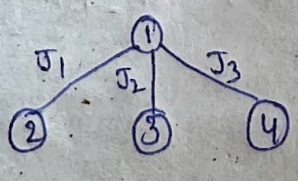
→ FIFO Branch & Bound is implemented by using Queue, in this the nodes are expanded by using FIFO principle.

eg:

Job	J_1	J_2	J_3	J_4
Deadline	8	1	2	1
Profit	10	5	15	10

⇒ In FIFO branch and bound we are expanding the path depends upon the job which is placed in Queue first.

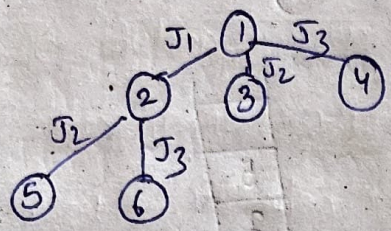
eg:



node ② is expanded first because it is placed first in Queue

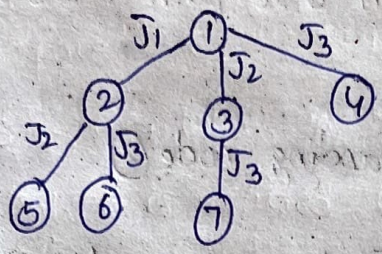
4	3	2

Now,



6	5	4	3

Now we are expanding node ③

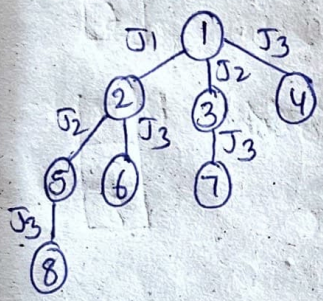


7	6	5	4

Now we are expanding node ⑦, there are no jobs to expand in that path.

7	6	5	

. Now we need to expand node ⑤



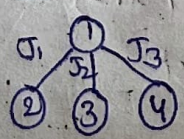
8	7	6

The nodes ⑥, ⑦ & ⑧ are having no jobs to expand in Corresponding Path.

2) LIFO

→ LIFO Branch & Bound is implemented by using stack, in LIFO Branch & Bound the nodes are expanded which are inserted last.

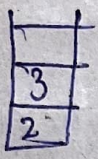
eg:



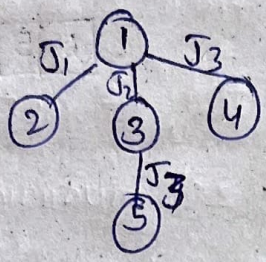
4
3
2

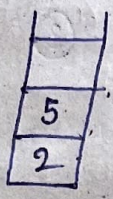
Now, we're expanding node ④ first, there is no jobs

to expand in that path.



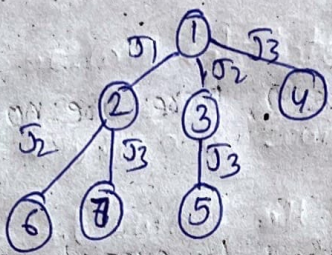
Now we're expanding node ③

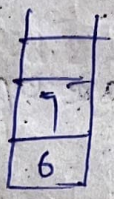




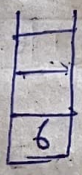
Now we're expanding node ⑤, in that path there are no jobs to expand.

Now we're expanding node ②

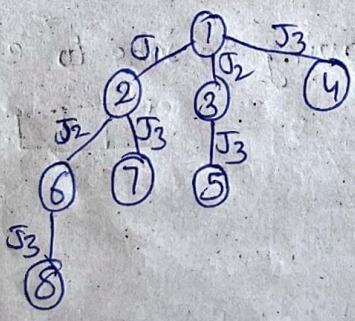




Now, we need to expand node ⑦, there are no jobs to expand in that path.



Now, we need to expand node ⑥ [~~there are no jobs to expand in that path~~]

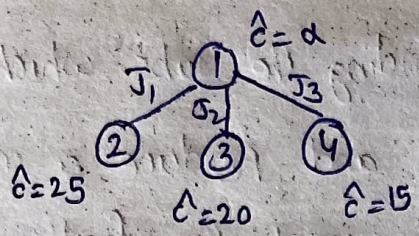


Now, node ⑧ but no jobs to expand so

3) LC Branch & Bound

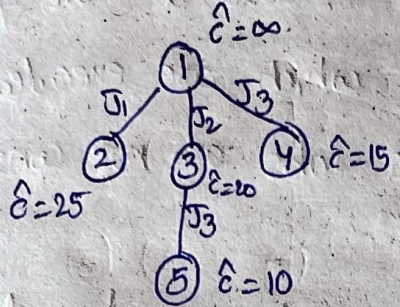
→ LC Brand & bound depends upon cost of nodes, the nodes which is having less cost is expanded first.

eg:



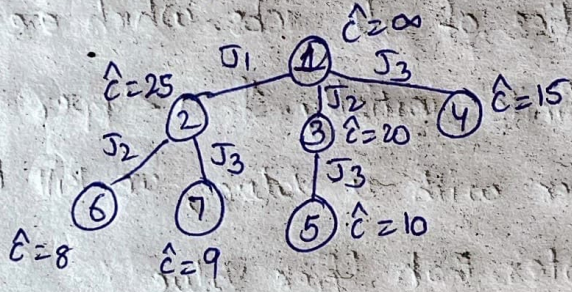
node ④ is having less cost so it is expanded first, in this path there are no jobs to expand.

node ③ is having least cost we need to expand it

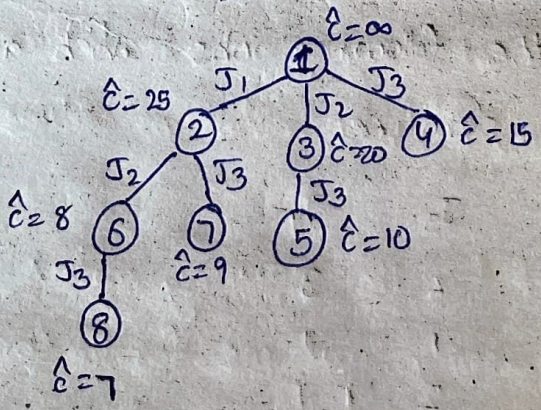


node ⑤ is having least cost so we need to expand it, there are no jobs to expand in that path.

node ② is having least cost we need to expand it



node ⑥ is having least cost we need to expand it



node ⑦ & node ⑧ are having no jobs to expand in their corresponding paths

Job Sequencing with Deadlines using Branch & Bound

→ In this problem we are finding the jobs which are executed within its deadline and produce a maximum profit, the branch & bound is used for finding a minimum optimal solution for that purpose in this problem we're calculating minimum loss instead of maximum profit.

→ For solving this problem we will construct a State Space tree which represents the jobs which are executed.

→ If job is executed its previous jobs are not considered to execute.

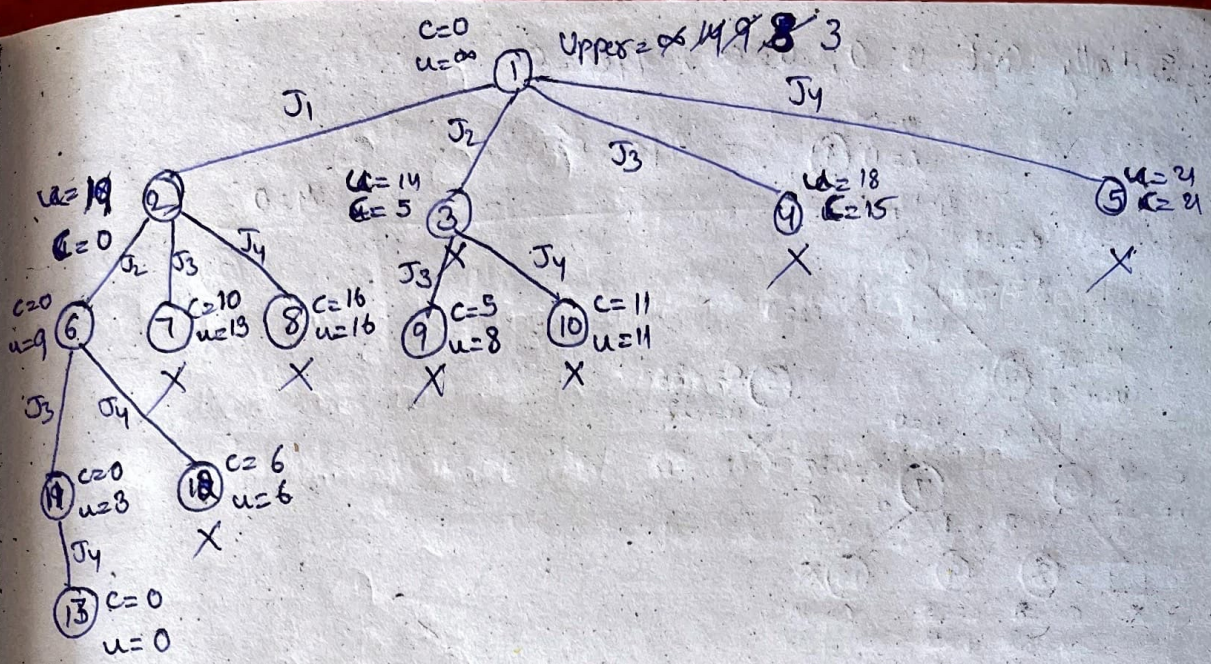
→ At every node we will calculate cost, upper bound (u) if we use LC Branch and Bound technique.

→ u is nothing but sum of Penalties of all jobs and c is sum of Penalties of all jobs which are leaving, Upper limit is infinite initially. If $u < Upper$ then we will update Upper with u value, we kill the nodes if cost value is greater than Upper value

Pblm

Find optimal solution for the following jobs which are executed in corresponding deadlines

Jobs	J_1	J_2	J_3	J_4
Penalty	5	10	6	3
Deadline	1	2	1	1
Execution time	1	3	2	1



19/3/24

Knapsack problem using Branch and Bound

→ In this problem we have to ~~find~~ place objects into bag and find an optimal solution which is maximum profit, the working principle of Branch and Bound is minimal optimization technique so we have to take negative values for cost, upper limit & Upper Bound.

→ Cost is nothing but sum of all Profits and which allows fractional values.

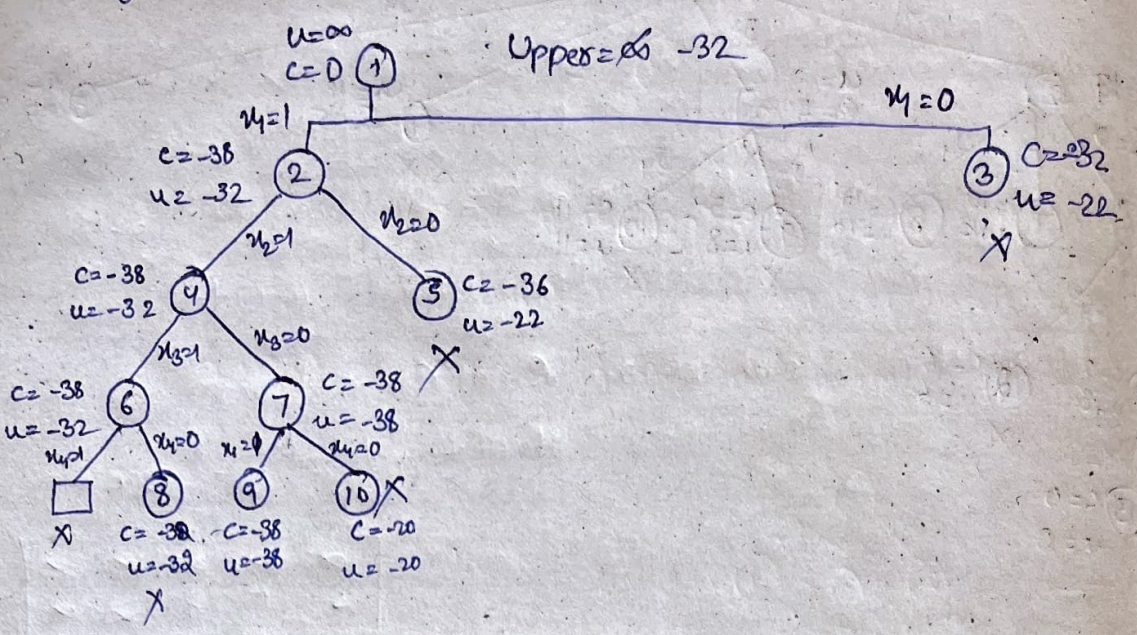
→ Upper Bound is nothing but sum of all profits which doesn't allows fraction values.

eg: If Cost value exceeds Upper Bound then we kill that node, if upper limit is less than upper bound then upper bound is updated with upper limit (W).

eg: Find Optimal Solution for the following objects using Knapsack Problem and produced optimal solution is minimum

object	1	2	3	4
Profit	10	10	12	18
weight	2	4	6	9

Initially Cost is 0, $u=\infty$, Upper $=\infty$



Answer is $x_1=1, x_2=1, x_3=0, x_4=1$

20/3/24

Travelling Salesperson problem using Branch and Bound:

In this problem we have to find a minimal optimal solution to visit a salesperson all the vertices in a graph. Salesperson start from one vertex and ends stop at same vertex.

For finding optimal solution the following steps are used

Step-1: Construct adjacency matrix for the given graph

Step-2: Find minimum values at every row and column in the adjacency matrix, subtract with minimum values every row and every column.

Step-3: Calculate reduced cost for the adjacency matrix.

Step-4: Construct all the adjacency matrices for the corresponding edges by making corresponding column and row infinite.

Step-5: Cost of vertex is calculated by using

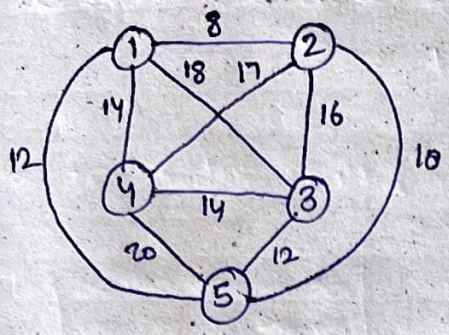
$$c(\text{Vertex}) = c(i,j) + r + \bar{r}$$

Where, r is reduced cost for the corresponding matrix

$\bar{r}$ is reduced cost for the starting adjacency matrix.

Step-6: Upper value initially infinite and it is updated whenever we reach Leaf node.

eg: Find optimal solution for the following graph using Travelling Salesperson- Branch and Bound.



Step-1:

Adjacency matrix =

$$\begin{bmatrix} \infty & 8 & 18 & 14 & 12 \\ 8 & \infty & 16 & 17 & 10 \\ 18 & 16 & \infty & 14 & 12 \\ 14 & 17 & 14 & \infty & 20 \\ 12 & 10 & 12 & 20 & \infty \end{bmatrix}$$

Find minimum value from every row and ~~reduce~~ subtract every element with minimum value row wise.

$$\begin{bmatrix} \infty & 8 & 18 & 14 & 12 \\ 8 & \infty & 16 & 17 & 10 \\ 18 & 16 & \infty & 14 & 12 \\ 14 & 17 & 14 & \infty & 20 \\ 12 & 10 & 12 & 20 & \infty \end{bmatrix} \begin{matrix} -8 \\ -8 \\ -12 \\ -14 \\ -10 \end{matrix} \Rightarrow \begin{bmatrix} \infty & 0 & 10 & 6 & 4 \\ 0 & \infty & 8 & 9 & 2 \\ 6 & 4 & \infty & 2 & 0 \\ 0 & 3 & 0 & \infty & 6 \\ 2 & 0 & 2 & 10 & \infty \end{bmatrix}$$

$$\begin{matrix} \cdot & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 2 & 0 \end{matrix}$$

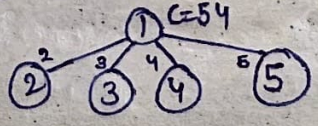
$$\begin{bmatrix} \infty & 0 & 10 & 4 & 4 \\ 0 & \infty & 8 & 7 & 2 \\ 6 & 4 & \infty & 0 & 0 \\ 0 & 3 & 0 & \infty & 6 \\ 2 & 0 & 2 & 8 & \infty \end{bmatrix} -(1)$$

Total row wise = 52

Total column wise = 2

$\bar{r} = 52 + 2 = 54$

* Cost of 1 = $c(1,2) + \bar{r} + \bar{r}$



Now Salesperson will move any one of the vertex 2, 3, 4, 5

Now we will calculate cost at every vertex from adjacency matrix of vertex1

Make 1st row and 2nd column all values infinite from matrix ① remaining all values unchanged.

If we move from ① to ② we are unable to move from ② to ① so make (2,1) is also infinite.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 8 & 7 & 2 \\ 6 & \infty & \infty & 0 & 0 \\ 0 & \infty & 0 & \infty & 6 \\ 2 & \infty & 2 & 8 & \infty \end{bmatrix}$$

Find minimum values at every row

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 8 & 7 & 2 \\ 6 & \infty & \infty & 0 & 0 \\ 0 & \infty & 0 & \infty & 6 \\ 2 & \infty & 2 & 8 & \infty \end{bmatrix} \begin{matrix} \infty \\ 2 \\ 0 \\ 0 \\ 2 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 6 & 5 & 0 \\ 6 & \infty & \infty & 0 & 0 \\ 0 & \infty & 0 & \infty & 6 \\ 0 & \infty & 0 & 6 & \infty \end{bmatrix}$$

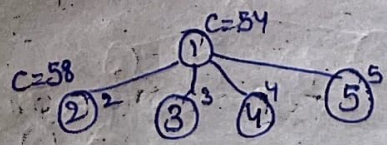
$$\begin{matrix} \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & \infty & 0 & 0 & 0 \end{matrix}$$

Total = 4

Total = 0

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 6 & 5 & 0 \\ 6 & \infty & \infty & 0 & 0 \\ 0 & \infty & 0 & \infty & 6 \\ 0 & \infty & 0 & 6 & \infty \end{bmatrix} \text{-②}$$

$$C(2) = d(1,2) + \gamma + \bar{\gamma}$$
$$= 0 + (4+0) + 54$$
$$= 58$$



from matrix ① make 1st row, 3rd column and (3,1) values as ∞

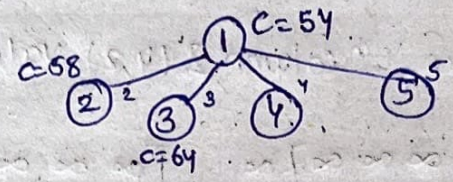
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 7 & 2 \\ \infty & 4 & \infty & 0 & 0 \\ 0 & 3 & \infty & \infty & 6 \\ 2 & 0 & \infty & 8 & \infty \end{bmatrix}$$

Find minimum values at every row

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 7 & 2 \\ \infty & 4 & \infty & 0 & 0 \\ 0 & 3 & \infty & \infty & 6 \\ 2 & 0 & \infty & 8 & \infty \end{bmatrix} \begin{matrix} -\infty \\ -0 \\ -0 \\ -0 \\ -0 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 7 & 2 \\ \infty & 4 & \infty & 0 & 0 \\ 0 & 3 & \infty & \infty & 6 \\ 2 & 0 & \infty & 8 & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & \infty & 0 & 0 \end{bmatrix}$$

Total = 0 Total = 0

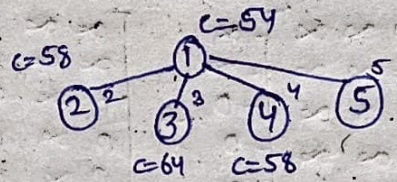
$c(3) = d(1,3) + r + \bar{r} = 10 + 0 + 54 = 64$



from matrix ① make 1st row, 4th column and (4,1) values as ∞
Find minimum values at every row

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 8 & \infty & 2 \\ 6 & 4 & \infty & \infty & 0 \\ \infty & 3 & 0 & \infty & 6 \\ 2 & 0 & 2 & \infty & \infty \end{bmatrix} \rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 8 & \infty & 2 \\ 6 & 4 & \infty & \infty & 0 \\ \infty & 3 & 0 & \infty & 6 \\ 2 & 0 & 2 & \infty & \infty \end{bmatrix} \begin{matrix} -\infty \\ -0 \\ -0 \\ -0 \\ -0 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 8 & \infty & 2 \\ 6 & 4 & \infty & \infty & 0 \\ \infty & 3 & 0 & \infty & 6 \\ 2 & 0 & 2 & \infty & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & \infty & 0 \end{bmatrix}$$

$c(4) = d(1,4) + r + \bar{r} = 4 + (0+0) + 54 = 58$

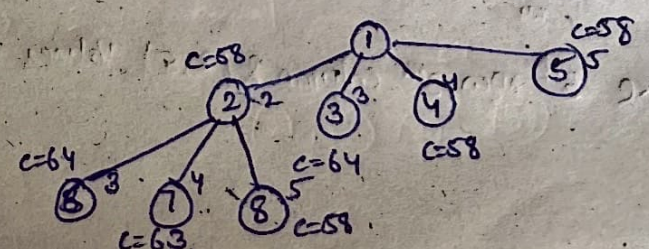


From matrix ① make 1st row, 5th column & (5,1) values as ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 8 & 7 & \infty \\ 6 & 4 & \infty & 0 & \infty \\ 0 & 3 & 0 & \infty & \infty \\ \infty & 0 & 2 & 8 & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 8 & 7 & \infty \\ 6 & 4 & \infty & 0 & \infty \\ 0 & 3 & 0 & \infty & \infty \\ \infty & 0 & 2 & 8 & \infty \end{bmatrix} \begin{matrix} -\infty \\ -0 \\ -0 \\ -0 \\ -0 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 8 & 7 & \infty \\ 6 & 4 & \infty & 0 & \infty \\ 0 & 3 & 0 & \infty & \infty \\ \infty & 0 & 2 & 8 & \infty \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & \infty \end{bmatrix}$$

Total = 0 Total = 0

$c(5) = d(1,5) + r + \bar{r} = 4 + 0 + 54 = 58$



From matrix ② make 2nd row, 3rd column & (3,2) values as ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & 0 & 0 \\ 0 & \infty & \infty & \infty & 6 \\ 0 & \infty & \infty & 6 & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & 0 & 0 \\ 0 & \infty & \infty & \infty & 6 \\ 0 & \infty & \infty & 6 & \infty \end{bmatrix} \begin{matrix} -\infty \\ -\infty \\ -0 \\ -0 \\ -0 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & 0 & 0 \\ 0 & \infty & \infty & \infty & 6 \\ 0 & \infty & \infty & 6 & \infty \\ 0 & \infty & \infty & 0 & 0 \end{bmatrix}$$

Total=0 Total=0

$$c(3) = d(2,3) + \gamma + \bar{\gamma} = 6 + 0 + 58 = 64$$

From matrix ② make 2nd row, 4th column & (4,2) values as ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & 0 \\ 0 & \infty & 0 & \infty & 6 \\ 0 & \infty & 0 & \infty & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & 0 \\ 0 & \infty & 0 & \infty & 6 \\ 0 & \infty & 0 & \infty & \infty \end{bmatrix} \begin{matrix} -\infty \\ -\infty \\ -0 \\ -0 \\ -0 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & 0 \\ 0 & \infty & 0 & \infty & 6 \\ 0 & \infty & 0 & \infty & \infty \\ 0 & \infty & 0 & \infty & 0 \end{bmatrix}$$

Total=0 Total=0

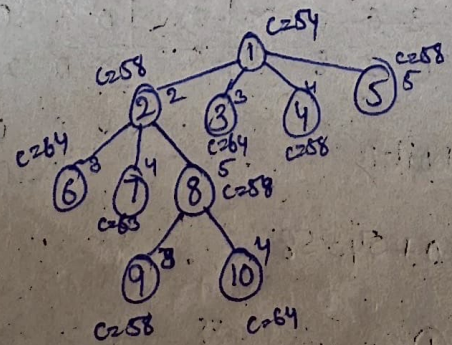
$$c(4) = d(2,4) + \gamma + \bar{\gamma} = 5 + (0) + 58 = 63$$

From matrix ② make 2nd row, 5th column & (5,2) values as ∞

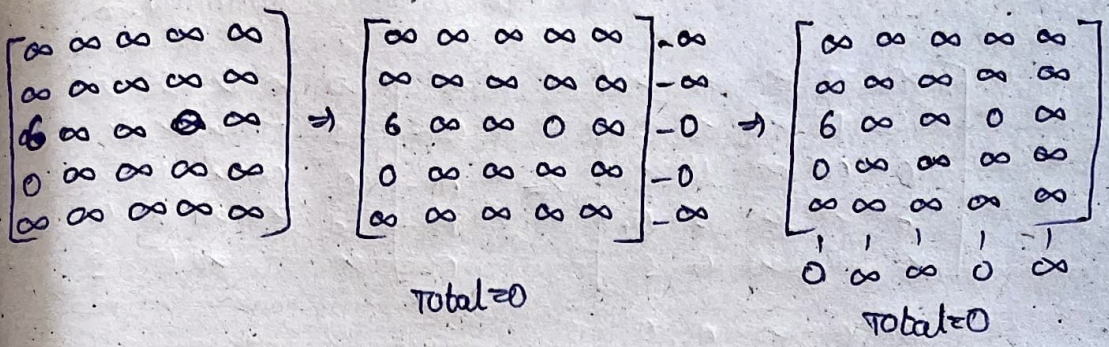
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & 0 & \infty \\ 0 & \infty & 0 & \infty & \infty \\ 0 & \infty & 0 & 6 & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & 0 & \infty \\ 0 & \infty & 0 & \infty & \infty \\ 0 & \infty & 0 & 6 & \infty \end{bmatrix} \begin{matrix} -\infty \\ -\infty \\ -0 \\ -0 \\ -0 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & 0 & \infty \\ 0 & \infty & 0 & \infty & \infty \\ 0 & \infty & 0 & 6 & \infty \\ 0 & \infty & 0 & 0 & \infty \end{bmatrix} - ③$$

Total=0 Total=0

$$c(5) = d(2,5) + \gamma + \bar{\gamma} = 0 + 0 + 58 = 58$$

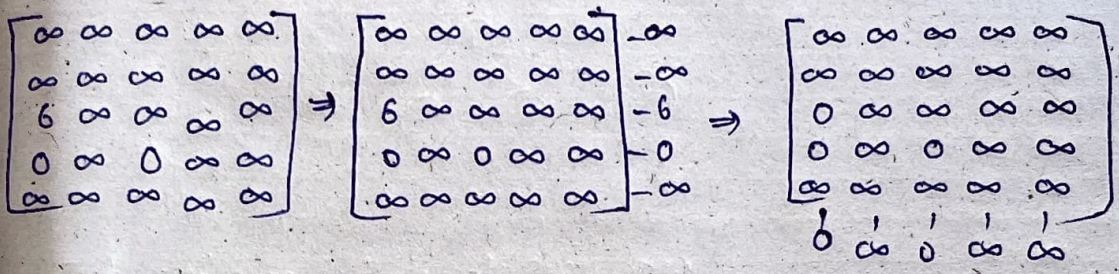


From matrix ③ make 3rd row, 3rd column & (3,5) values as ∞



$c(3) = d(5,3) + \gamma + \bar{\gamma} = 0 + 6 + 58 = 58$

From matrix ③ make 5th row, 4th column & (4,5) values as ∞



$c(4) = d(5,4) + \gamma + \bar{\gamma} = 0 + 6 + 58 = 64$