BRANCH AND BOUND

This method is generally applied to only optimization problem ie maximization (knapsack) problem or minimization (TSP) problem only

In this method a state space tree of all possible solution is generated.
Then partitioning or branching is done at each node of the tree.
Then compare the node's bound value with the value of the best solution obtained up to now ie current state

If the bound value of some node is better than the best solution then that corr. node is not expanded. It is also called branch pruned here

After computation of bounding value at each node, we compare it with best solution & then expanding the node further can lead to a final solution node.

The branching procedure replaces an original problem by a set of new problem that are:
- Mutually exclusive and exhaustive sub prblm of the original problem
- Partially solved version of the original prblm.
- Smaller problems than original prblm.

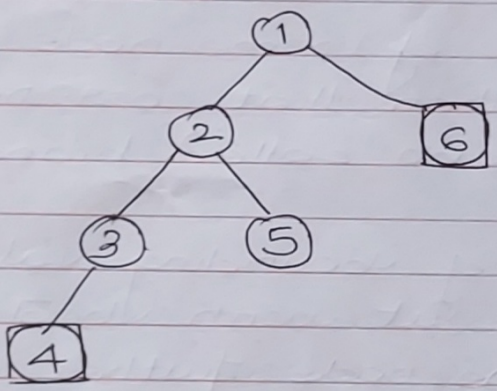
There are various reason for terminating search path in state space tree of branch & bound.
- The value of the node's bound is not better

than the best solution
- There is no feasible soln
- A new better soln can be obtained rather than continuing further.

Terminology

Answer Node

Those solution states S for which path from root to S defines a tuples which is a member of the set of solutions (ie it satisfies the implicit constraints) of the problem.



If 4 and 6 are the answer state (1,2,3,4) + (1,6) are the solution states.

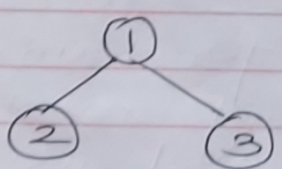
Live Node

A node which has been generated and all of those children have not yet been generated is live node.

Step1

Here 1 is a live node since children of 1 not yet been generated.

Step2

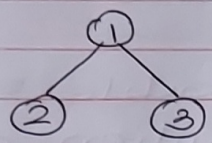


Here 2 & 3 are live node But node 1 is not live since all the children of 1 are generated.

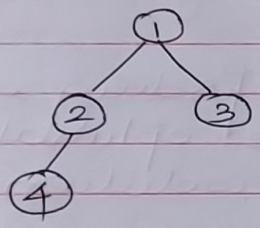
E-node

The live node whose children are currently being generated is called the E-node (node being expanded)

Here 1 is a live node. And its children are currently being expanded. So 1 is the E-node.



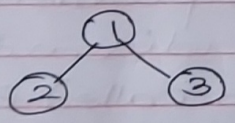
Here 1 is a live node. And its childre are currently being expanded. So 1 is the Enode



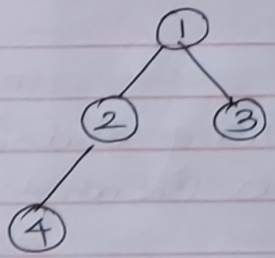
Here 2 & 3 are live nodes. And 2's children are currently being expand So 2 is the Enode.

Dead Node

It is generated node that is either not to be expanded further or one for which all of its children have been generated.



Here 1 is already expanded. Assum 2 & 3 are not expanded. Then 1,2,3 a dead nodes.



Here 1 is already expanded. And 2's children are currently being expanded. So it is Enod Assume that 3 & 4 are not expanded. Then 1,3 & 4 are dead node.

Generally three types of searching strategies used in branch and bound

1) FIFO (First In First Out)
2) LIFO (Last In First Out)
3) LC (Least Count) Search

Branch and Bound requires two tools.

a) Way of covering the feasible solution by several smaller feasible solution called BRANCHING - we are exploring all solns.

b) BOUNDING is a way of finding upper and lower bound for the optimal soln within a feasible sub solution.
When bounding we are eliminating least feasible solution

If the lower bound of a sub problem A from the search tree is greater than the upper bound for any other previously examined subproblem B, than A may be safely discarded from the search. This is called PRUNING

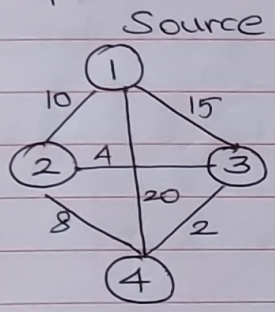
FIFO (First In First Out) Search.

- Breadth first search with QUEUE based Branch & Bound is called FIFO Branch & Bound
State space tree using BFS
- In the FIFO Search, the children of the

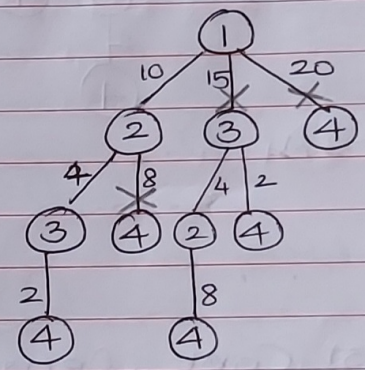
root node are generated in the first iteration
In the next step, children of the root node are generated
If the children not already killed by bounding function then put it in the queue Then the children of second child is explored.

Put all the children in the queue except for those children which are killed.

Example:



To traverse from Source to Destn



1

2 3 4

2 3 4 4 4 4 4

Not expand as dest.

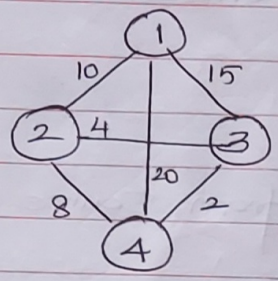
Therefore the path: 1-2-3-4
Cost : 10+4+2 =16

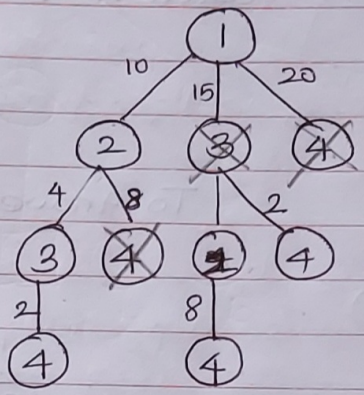
LIFO (Last In First Out) Search

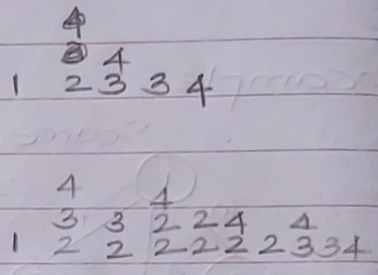
- DFS with stack based Branch & Bound is called LIFO Branch & Bound
- In LIFO search the children of the root node are generated in the first iteration
- In the next step, children of the ~~root node are generated in the first iteration~~ child is generated.

If the children not already killed by bounding function then put it in the stack. Then the children of second child is explored.

- Put all the children in the stack except for those children which are killed.







Path: 1-2-3-4

Cost = 16

LC (Least Cost Search)

- In both FIFO & LIFO Branch & Bound, the selection rule for the next E-node is fixed
- The rigid selection rule does not give any preference to a node which best and can search the answer node quickly
- The search for an answer node can be speeded by using a ranking function for live nodes.
- The next E-node is selected on the basis of this ranking function.

The ideal way do assign ranks would be on the basis of additional computational effort needed to reach the answer node from the live node.

For any node x, this cost could be:

- The number of nodes in the subtree x that needed to be generated before an answer node is generated
- The no. of levels the nearest answer node is from 'x'.

The ranking function is

$$c(x) = f(x) + g(x)$$

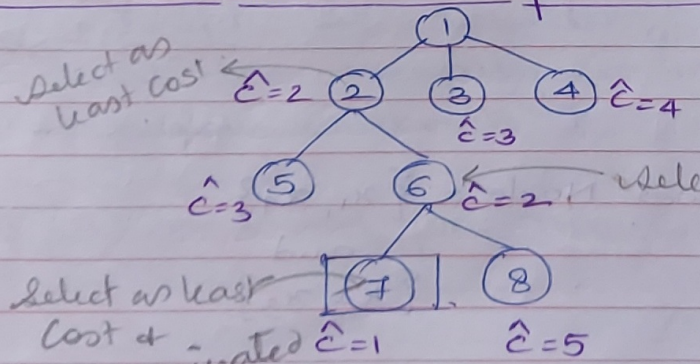
where $c(x)$ is the cost of x

$f(x)$ is the cost of reaching x from root & is non-decreasing

$g(x)$ is an estimate of the additional effort needed to reach an answer node x.

To select the next E-node would always choose for its next E-node a live node with least $c(x)$

Control Abstractions for LC Search



Example

Subset sum problem

$M = 13$

$n = 4, \quad w_1 w_2 w_3 w_4 = (7, 4, 9, 6)$

Queue

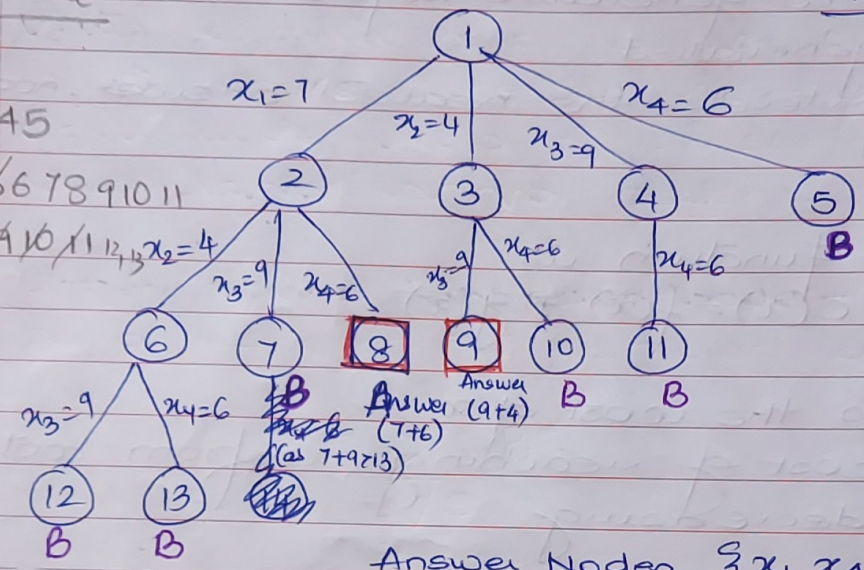
1

2 3 4 5

3 4 5 6 7 8 9 10 11

4 5 6 7 8 9 10 11 12 13 $x_2 = 4$

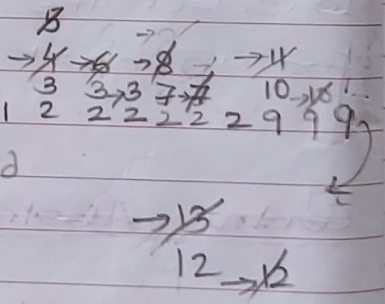
FIFO Branch & Bound using Queue

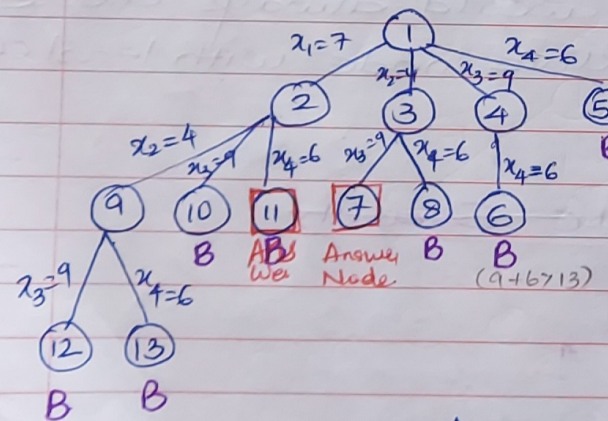


Answer Nodes $\{x_1, x_4\} + \{x_2, x_3\}$

LIFO Branch & Bound using Stack.

Stack

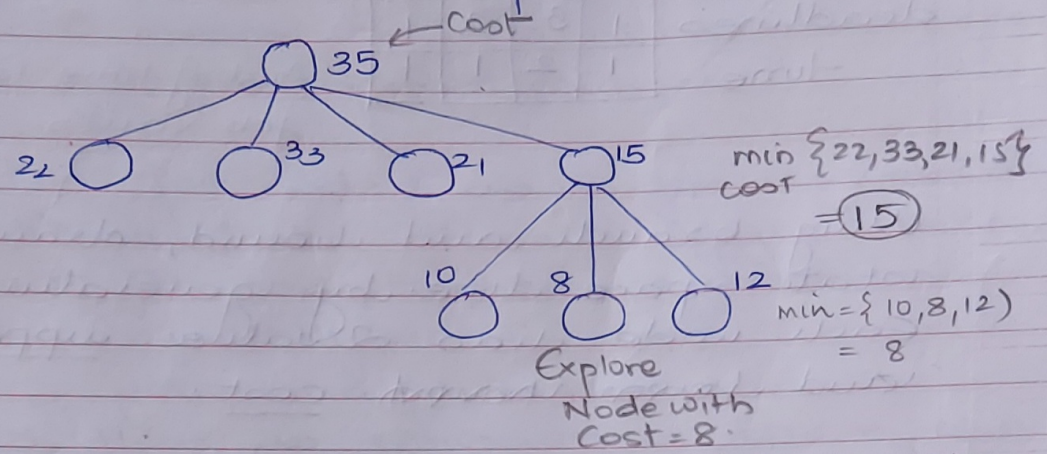




Answer Node = $\{x_1, x_4\}$ and $\{x_2, x_3\}$

Least Count Search. B & B (priority Queue)

Consider the below state space tree



For Control Abstraction using LC Search Refer DAA Notes pdf.

Job sequencing with deadlines

Given 'n' jobs and 1 processor
Each job j_i has a 3-tuple associated with it represented by (p_i, d_i, t_i)
- Job J_i requires t_i units of processing time
- If processing is not completed by deadline d_i, a penalty p_i is incurred.

Objective is to select a subset J of 'n' jobs such that all jobs in J can be completed by their deadline
- A penalty will incur only on jobs not in J.
- For optimal solution, J should be such that the penalty is minimized among all possible subsets J.

B & B → used for minimization prblm

Problem

	J_1	J_2	J_3	J_4
Penalty	5	10	6	3
deadlines	1	3	2	1
time	1	2	1	1

Minimizes penalty

Amt of time taken for comp job varies.

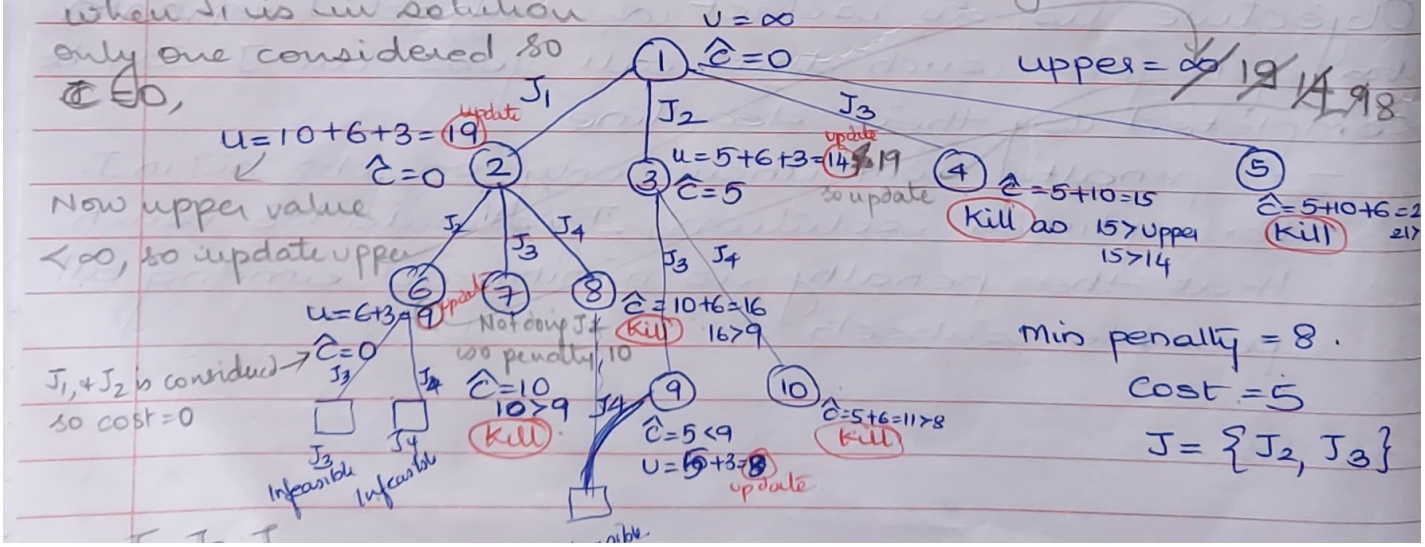
Using branch and bound, draw the state space tree by generating nodes. Each node has 2 values upper bound and cost.

Upper bound for each node - sum of all penalties except that included in solution

$$u = \sum_{i \notin S} P_i$$

Cost, $\hat{c}$ = sum of penalties till the last job consid

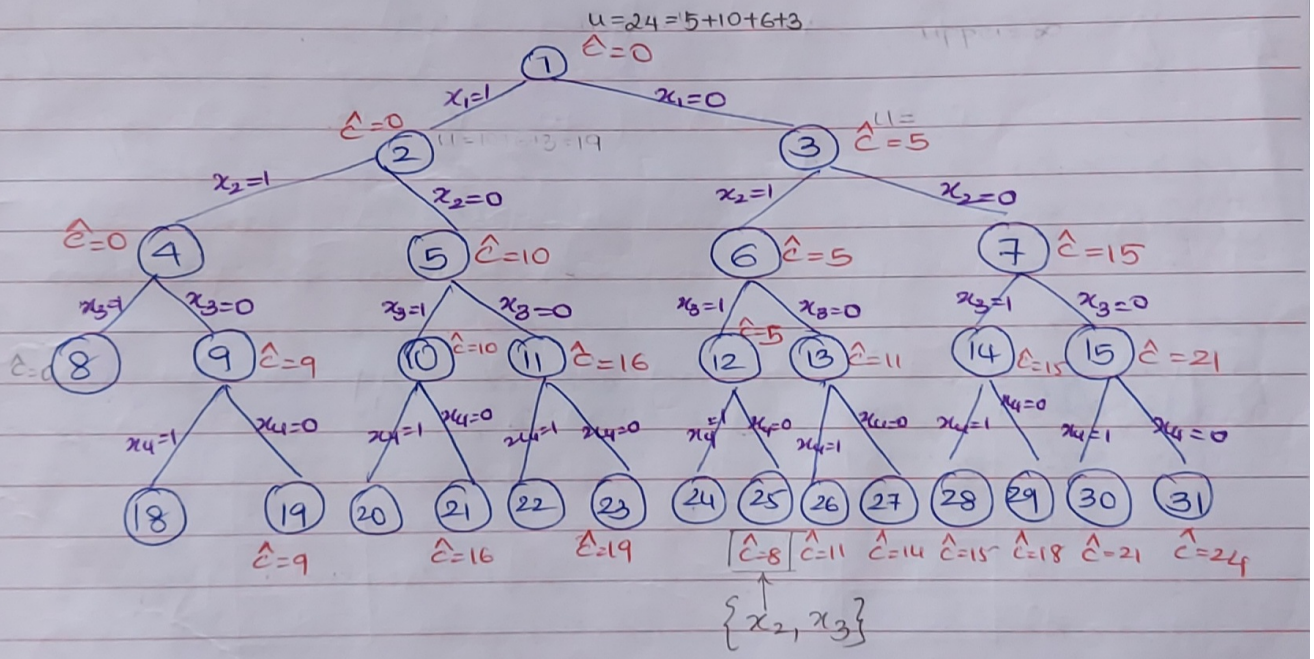
$$\hat{c} = \sum_{i \in S_k} P_i$$ upto last k^{th} job



Fixed size tuple formulation

In fixed tuple size i/p, $x_i=1$ means job 'i' is selected $x_i=0$ means job i is not selected for formulation of J (the optimal subset).

Initially $\hat{c}=0$ for node 1 as no job is selected. For node 3, it indicates omission of job 1. Hence penalty is 5. For node 5 there is omission of job 2. Hence $\hat{c}$ = penalty = 10. For node 7 there is omission of job1 and job2. Hence $\hat{c}=15$. For node 13 there is omission of job 1 + job 3. Hence penalty is 11. Therefore $\hat{c}=11$. Continuing in this $\hat{c}$ is computed.



Min penality = 8

0/1 knapsack problem using branch & bound.

There are n objects and the capacity of the knapsack is M. We have to select some objects from 'n' objects in such a way that it should not exceed the capacity of the knapsack 'M' and the maximum profit can be earned

So, the knapsack is maximization problem ~~But branch and bound deals only with minimization problem. So we~~
Let M be the capacity. 'n' objects are to placed in the knapsack. whose weights are $w_1, w_2, \ldots w_n$ resp. such sum of weights of objects $\leq$ capacity of knapsack giving maximize profit

ie maximize $\sum_{i=1}^{n} P_i x_i$ such that $\sum_{i=1}^{n} w_i x_i \leq M$.
as max profit to be obtained

here $x_i = 0/1$ (So 0/1 knapsack is max. problem)
But branch and bound deals only with minimization problem. So, we modify the gi knapsack problem to be minimization problem.
The modified problem is

$$\min Z = -P_1 x_1 - P_2 x_2 - \ldots - P_n x_n$$

The constraints are $w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \leq M$.
and $x_i = 0$ (or) 1

In Branch and bound, we calculate the lower bound and upper bound for each and every node in state space tree.

In lower bound, fractions are allowed whereas in upper bound fractions are not allowed.

Algorithm for LC Branch & Bound

1) Draw a state space tree and set upper = ∞
2) Compute $\hat{c}(x)$, $\hat{u}(x)$ for each node
3) $\hat{u}(x) = -\sum P_i$

$$\hat{c}(x) = u(x) - \frac{[\text{m - current total weight}] * [\text{actual profit of remaining obj}]}{[\text{actual weight of remaining obj}]}$$

4) If u(x) is min than upper then upper will set to u(x)
5) If $\hat{c}(x) >$ upper kill node x
6) Otherwise the min cost $\hat{c}(x)$ becomes Enode & generate children for Enode
7) Repeat steps 2 to 6 until all the nodes get covered
8) The minimum cost $\hat{c}(x)$ becomes the answer node. Trace the path in backward direction from x to root for solution set.

Algorithm 0/1 Knapsack - FIFO B&B

1) Draw a state space tree & set upper = ∞
2) Compute $\hat{c}(x)$, $\hat{u}(x)$ for each node
3) $\hat{u}(x) = -\sum P_i$

$$\hat{c}(x) = u(x) - \frac{[\text{m - current total weight}] * [\text{actual profit of rem. obj}]}{[\text{actual weight of remaining object}]}$$

4) If u(x) is min than upper then upper will set to u(x)
5) If $\hat{c}(x) >$ upper, Kill node x
6) Next live node becomes Enode & generate children for Enode
7) Repeat 2 to 6 until all the nodes get covered
8) The min cost $\hat{c}(x)$ becomes the answer node. Trace the path in backward direction from x to root for solution subset.

Differences betn Backtracking and B&B.

Backtracking	Branch and Bound
1) In backtracking, DFS technique is used for tracing the solution for the given problem	1) In BnB, BFS technique is used for tracing the soln for the given problem
2) Typically decision problem can be solved using backtrack	2) Typically optimizatn problem solving using BnB.
3) While finding the solutions to the prblm, bad choices can be made	3) It proceeds only on optima or better solutions.
4) The state space tree is searched until the soln is obtained	4) The state space tree need to be searched completely as there may be chances of being an optimal soln anywh in state space tree.
5) Appl's of backtracking are - N-Queen prblm - Graph colouring - Hamiltonian cycle	5) Appl's of branch & bound - 0/1 knapsack prblm - Travelling Salesperson prb

0/1 Knapsack Problem (LCBB)

$M=15$

$n=3$

$(w_1, w_2, w_3) = (8, 12, 6)$

$(P_1, P_2, P_3) = (32, 24, 6)$

Solution

(I) $\frac{P_1}{w_1} = \frac{32}{8}$; $\frac{P_2}{w_2} = \frac{24}{12}$; $\frac{P_3}{w_3} = \frac{6}{6}$

$= 4$; $= 2$; $= 1$

$\frac{P_1}{w_1} > \frac{P_2}{w_2} > \frac{P_3}{w_3} > 1$

(II) Convert the given profits to negative profits

$\therefore (P_1, P_2, P_3) = (-32, -24, -6)$

(III) Set $\boxed{\text{upper} = \infty}$

(IV) For Node 1

Lower Node.

$M=15$

$\frac{7}{12} \times 24$
8

-14

-32

$\hat{c}(1) = -32 - 14$

$\boxed{\hat{c}(1) = -46}$

Upper bound

$M=15$

6
8

-6

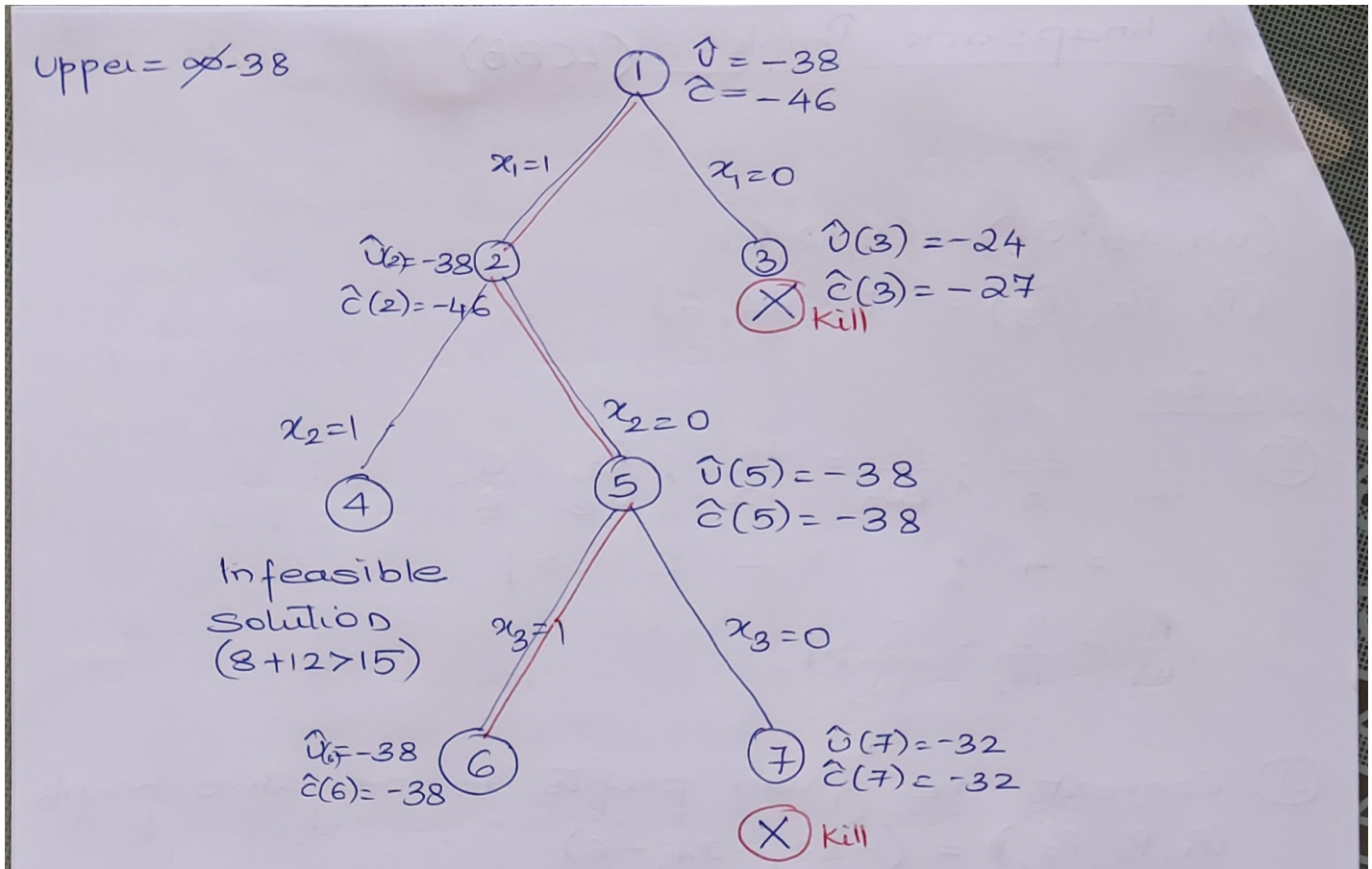
-32

$\hat{u}(1) = -32 - 6$

$\boxed{\hat{u}(1) = -38}$

Check $\hat{u}(1) < \text{upper} = -38 < \infty = -38$

$\hat{c}(1) > \text{upper} = -46 > -38$ (false), so explore node 1



(V) **For Node 2 $x_1=1$**

Lower bound $\hat{c}(2)$ =

8	7/12 x -24
-32	-14

$\Rightarrow$ $\hat{c}(2) = -46$

Upper bound $\hat{u}(2)$ =

8	6
-32	-6

$\Rightarrow$ $\hat{u}(2) = -38$

Check $\hat{u}(2) <$ upper $= -38 < -38$ (false)

$\hat{c}(2) >$ upper $= -46 > -38$ (false), Explore Node2

(VI) **For Node 3 $x_1=0$**

Lower bound $\hat{c}(3)$ =

12	3/6 x -6
-24	-3

$\Rightarrow$ $\hat{c}(3) = -24 - 3 = -27$

Upper bound $\hat{u}(3)$ =

12
-24

$\Rightarrow$ $\hat{u}(3) = -24$

Check $\hat{u}(3) <$ upper $= -24 < -38$ (false.

$\hat{c}(3) >$ upper $= -27 > -38$ (true) kill Node3

For node 4, $x_1=1$, $x_2=1$

Lower bound $\hat{c}(4) \Rightarrow$

8	12
-32	-24

(Infeasible solution)

VIII) For node 5, $x_1=1$, $x_2=0$

Lower bound $\hat{c}(5) \Rightarrow$

8	6
-32	-6

$\Rightarrow \hat{c}(5) = -38$

Upper bound $\hat{u}(5) \Rightarrow$

8	6
-32	-6

$\Rightarrow \hat{u}(5) = -38$

Check $\hat{u}(5) < \text{upper} = -38 < -38$ false

$\hat{c}(5) > \text{upper} = -38 > -38$ false, so expand node 5

IX) For node 6, $x_1=1$, $x_2=0$, $x_3=1$

Lower bound $\hat{c}(6) \Rightarrow$

8	6
-32	-6

$\Rightarrow \hat{c}(6) = -38$

Upper bound $\hat{u}(6) \Rightarrow$

8	6
-32	-6

$\Rightarrow \hat{u}(6) = -38$

Check $\hat{u}(6) < \text{upper} = -38 < -38$ false

$\hat{c}(6) > \text{upper} = -38 < -38$ false, expand Node 6.

But no item is remaining to be considered.

X) Node 7, $x_1=1$, $x_2=0$, $x_3=0$

Lower Bound $\hat{c}(7) \Rightarrow$

8
-32

$\Rightarrow \hat{c}(7) = -32$

Upper bound $\hat{u}(7) \Rightarrow$

8
-32

$\Rightarrow \hat{u}(7) = -32$

Check if $\hat{u}(7) < \text{upper} = -32 < -38$ false

$\hat{c}(7) > \text{upper} = -32 > -38$ true, kill Node 7

XI) Solution Vector is $x(x_1, x_2, x_3) = (1, 0, 1)$ Max profit = -38, weight = 14

1) for Node 6, $x_1=1$, $x_2=1$, $x_3=1$

Lower bound $\hat{c}(6)$ = | 2 | 4 | 6 |
|---|---|---|
| -10 | -18 | -12 |

Here $2+4+6>8$, so it will lead to a infeasible solution

0) For Node 7, $x_1=1$, $x_2=1$, $x_3=0$

Lower bound $\hat{c}(7) \Rightarrow$ | 2 | 4 |
|---|---|
| -10 | -18 |

$\Rightarrow \hat{c}(7) = -28$

Upper bound $\hat{u}(7) \Rightarrow$ | 2 | 4 |
|---|---|
| -10 | -18 |

$\Rightarrow \hat{u}(7) = -28$

Check $\hat{u}(7) < \text{upper} = -28 < -28$ (false)

$\hat{c}(7) > \text{upper} = -28 > -28$ (false) Expand Node 7.

But all items are considered so stop.

∴ The solution vector $x(x_1\ x_2\ x_3) = (1,1,0)$

Max Profit = $10\times1 + 18\times1 + 12\times0 = 28$

Max Weight = $2\times1 + 4\times1 + 6\times0 = 6$.

For Node 2 $x_1 = 1$

Lower Bound $\hat{c}(2) \Rightarrow$

02	4	2/6X-12
-10	-18	-4

$\Rightarrow -32$

Upper bound $\hat{u}(2) \Rightarrow$

2	4
-10	-18

$\Rightarrow -28$

Check $\hat{u}(2) < \text{upper} = -28 < -28$ (false)

$\hat{c}(2) > \text{upper} = -32 < -28$ (false) expand Node 2

6) For Node 3, $x_1 = 0$

Lower Bound $\hat{c}(3) \Rightarrow$

4	4/6X-12
-18	-8

$\Rightarrow -26$

Upper Bound $\hat{u}(3) \Rightarrow$

4
-18

$\Rightarrow \hat{u}(3) = -18$

Check $\hat{u}(3) < \text{upper} = -18 < -28$ (false)

$\hat{c}(3) > \text{upper} = -26 > -28$ (true), Kill Node 3.

7) Node 4, $x_1 = 1$, $x_2 = 1$

Lower Bound $\hat{c}(4) =$

2	4	2/6X-12
-10	-18	-4

$\Rightarrow -32$

Upper bound $\hat{u}(4) =$

2	4
-10	-18

$\Rightarrow -28$

Check $\hat{u}(4) < \text{upper} = -28 < -28$ false

$\hat{c}(4) > \text{upper} = -32 > -28$ (false), Expand Node 4

8) Node 5, $x_1 = 1$, $x_2 = 0$

Lower bound $\hat{c}(5) =$

2	6
-10	-12

$\Rightarrow -22$

Upper bound $\hat{u}(5) =$

2	6
-10	-12

$\Rightarrow -22$

Check $\hat{u}(5) < \text{upper} = -22 < -28$ (false)

$\hat{c}(5) > \text{upper} = -22 > -28$ (true) Kill Node 5

(2) M=08, n=3, $(w_1 w_2 w_3) = (2, 4, 6)$ (LCBB)
$(P_1 P_2 P_3) = (10, 18, 12)$

Sol: 1) Arrange the items in P/W order
$(P_1/w_1, P_2/w_2, P_3/w_3) = (10/2, 18/4, 12/6)$
$= (5, 4.5, 2)$

2) Convert each profit value to negative profits
$(P_1 P_2 P_3) = (-10, -18, -12)$

3) Set upper = ∞

4) For Node 1
M = 8

Lower bound $\hat{c}(1) \Rightarrow$

2	4	2/6 X -12
-10	-18	-4

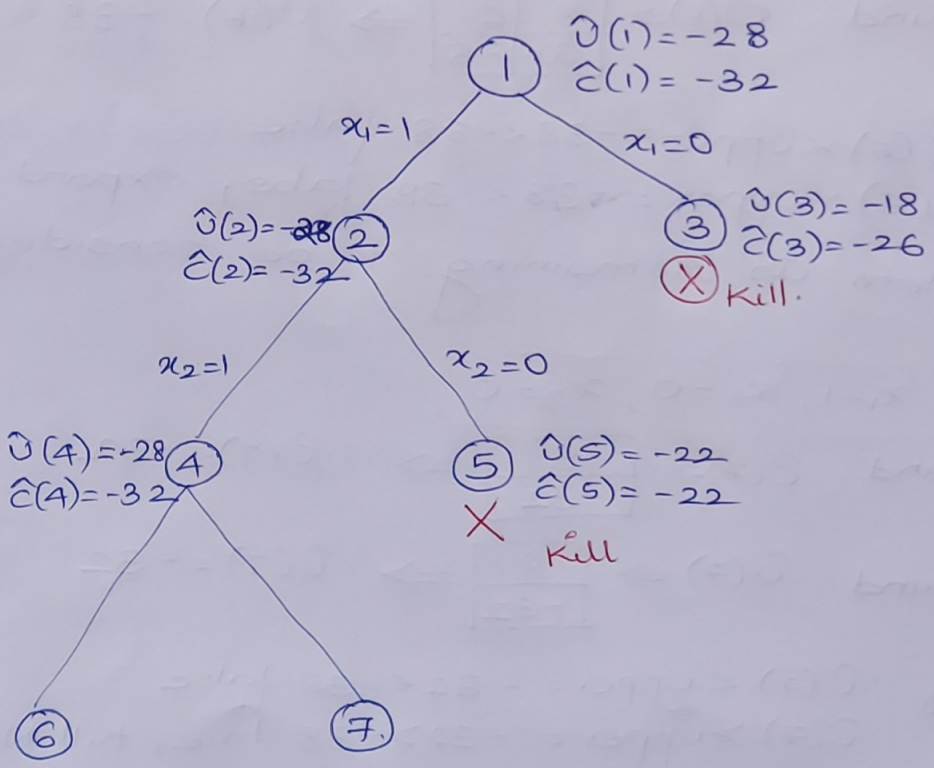
$\Rightarrow -32$

Upper bound $\hat{u}(1) \Rightarrow$

2	4
-10	-18

$\Rightarrow -28$

Check $\hat{u}(1) <$ upper $= -28 < \infty$, upper $= -28$
$\hat{c}(1) >$ upper $= -32 > -28$, (false) Expand Node 1



TRAVELLING SALESPERSON USING BRANCH AND BOUND

If there are 'n' cities and cost of travelling from one city to another city are given. A Salesman has to start from one city and has to visit all the cities exactly once and has to return to the starting place with shortest distance or minimum cost.

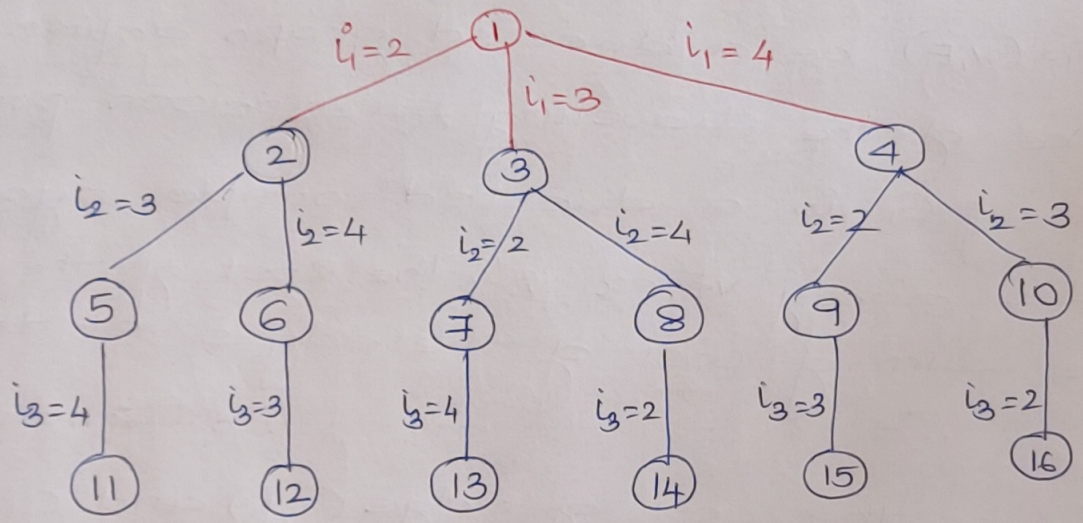
Let $G=(V,E)$ be a directed graph defining an instance of the traveling salesperson problem. A tour of G is a directed simple cycle that includes every vertex 'v'. The cost of the tour is the sum of cost of the edges on the tour.

Let c_{ij} be the cost of edge (i,j) and $c_{ij}=\infty$ if $(i,j) \notin E(G)$ and $|V|=n$

Assume that every tour starts and ends at vertex 1. So the solution space S is given by $S=\{(1,\pi,1) \mid \pi$ is a permutation of $(2,3\cdots n)\}$.
Then $|S|=(n-1)!$
The size of S can be reduced by restricting S so that $(1,i_1,i_2,\cdots,i_{n-1},1) \in S$ iff $(i_j,i_{j+1}) \in$ $0 \le j \le n-1$ and $i_0=i_n=1$.

S can be organized into a state space tree. Consider a tree organization given below for the case of a complete graph with $|V| = 4$. Each leaf node L is a solution node + represents the tour defined by the path from the root to L.

Eg: Node 14 represents the tour $i_0 = 1$, $i_1 = 3$, $i_2 = 4$, $i_3 = 2$ and $i_4 = 1$.



State space tree for travelling salesperson problem with $n = 4$ + $i_0 = i_4 = 1$

Matrix Reduction

A row or column is said to be reduced if it contains atleast one zero and all the remaining entries are non-negative in that row/column.

A matrix is called reduced iff every row and columns are reduced.

This matrix reduction to compute cost fn in TSP using BB.

Example for matrix reduction

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$$

Row reduction ⇒

					Min value
∞	20	30	(10)	11	10
15	∞	16	4	(2)	2
3	5	∞	(2)	4	2
19	6	18	∞	(3)	3
16	(4)	7	16	∞	4

⇓

After row reduction ⇒

(∞-10)	(20-10)	(30-10)	(10-10)	(11-10)	
∞	10	20	0	1	All rows are reduced as each row entry has atleast one zero
13	∞	14	2	0	
1	3	∞	0	2	
16	3	15	∞	0	
12	0	3	12	∞	

Now, check the column for reduction

These columns are reduce as atleast a single zero

∞	10	20	0	1
13	∞	14	2	0
1	3	∞	0	2
16	3	15	∞	0
12	0	3	12	∞
Min Value	1	0	3	0	0

⇒

∞	10	17	0	1
12	∞	11	2	0
0	3	∞	0	2
15	3	12	∞	0
11	0	0	12	∞

Reduced matrix

So, the total reduced cost = total row Reduction Cost + total column Reduction Cost

$= (10+2+2+3+4) + (1+0+3+0+0)$

$= 21 + 4$

$= (25)$

In all tour, for all j there is an exactly one edge (i,k) for $k=1,2,\cdots n$ and there is an exactly one edge (k,j) for $k=1,2,\cdots n$

(leaving vertex) (entering vertex)

eg: 1 2 3 4 1

If $j=3$, one edge (j,k) where j is leaving vertex $(3,4)$
for all one edge (k,j) where j is entering vertex $(2,3)$

If t constant is chosen to be minimum entry in row 'i' or column 'j', then subtracting it from all entries in row i or column j will introduce a zero into row 'i' or column 'j'.

The total amount subtracted from the columns and rows is lower bound on the length of a minimum cost tour and can be used as $\hat{c}(x)$ value for the root of the state space tree.

(lower bound)

Original Matrix

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$$

Reduce Matrix

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

Consider the tour T_1 : 1-2-3-4-5-1

$T_1 = c_{12}+c_{23}+c_{34}+c_{45}+c_{51}$

$= 20+16+2+3+16$

$= 57$

$T_1 = c_{12}+c_{23}+c_{34}+c_{45}+c$

$= 10+11+0+0+1$

$= 32$

∴ Total Reduction Cost = 25 from previous reduction process

We may note = 57 − 32 = 25.

Hence, All tours in the original graph have a length at least 25.

NOTE:

Since every tour on the graph includes exactly one edge (i,j) with $i=K$, $1 \le K \le n$ and exactly one edge (i,j) with $j=K$, $1 \le K \le n$, subtracting a constant t from every entry in one column or one row of the cost matrix reduced the length of every tour by exactly t. A minimum cost tour remains a minimum cost tour following this subtract^n operation.

Matrix Original cost of tour — Tour Cost of Reduced Matrix

57 — 32 ⇒ 25

Conclusion – If we subtract a min amount from the row/col and we reduce to matrix then total amt subtracted will reduce the cost of the tour by that amount.

$57 - 32 = 25$

$32 + 25 = 57$

∴ Cost of every tour will be reduce by 23

Min. cost of the tour will not be changed, so this will give the lower bound ie cost of the tour in this graph cannot be less than 25. So here this amount(25) is used as cost function and associate the reduction matrix with every node of the state space tree

⇒ The reduced cost matrix is associated with every node in the travelling salesperson state space tree

⇒ Let A be the reduced cost matrix for node R. Let S be a child of R such that the tree edge (R,S) corresponds to including edge (i,j) in the tour.

⇒ If S is not a leaf, then the reduced cost matrix for S may be obtained as follows:

(1) Change all entries in row i and column j of A to ∞.

(2) Set A(j, 1) to ∞

(3) Apply row reduction & column reduction except for rows & columns containing ∞

(4) The total cost of node 'S' can be calculated as $\hat{c}(S) = \hat{c}(R) + A(i, j) + r$

where 'r' is the total amount subtracted in Step 3.

For upper bound fn u, use $u(R) = \infty$ for all nodes R

Solve the following instance of TSP using LCBB

$$\begin{bmatrix} \infty & 16 & 11 & 6 \\ 8 & \infty & 13 & 16 \\ 4 & 7 & \infty & 9 \\ 5 & 12 & 2 & \infty \end{bmatrix}$$

Soln **Row Reduction**

Subtract row minimum value from corr. row

Min

$$\begin{bmatrix} \infty & 16 & 11 & 6 \\ 8 & \infty & 13 & 16 \\ 4 & 7 & \infty & 9 \\ 5 & 12 & 2 & \infty \end{bmatrix} \begin{matrix} 6 \\ 8 \\ 4 \\ 2 \end{matrix} \Rightarrow \begin{bmatrix} \infty & 10 & 5 & 0 \\ 0 & \infty & 5 & 8 \\ 0 & 3 & \infty & 5 \\ 3 & 10 & 0 & \infty \end{bmatrix}$$

Row reduction Cost = 6+8+4+2 = 20

Column Reduction

Subtract column minimum value from corr. col.

$$\begin{bmatrix} \infty & 10 & 5 & 0 \\ 0 & \infty & 5 & 8 \\ 0 & 3 & \infty & 5 \\ 3 & 10 & 0 & \infty \end{bmatrix}$$

Min 0 3 0 0

$\Rightarrow$

Column Reduction Cost = 3

$$\begin{bmatrix} \infty & 7 & 5 & 0 \\ 0 & \infty & 5 & 8 \\ 0 & 0 & \infty & 5 \\ 3 & 7 & 0 & \infty \end{bmatrix}$$

Total Reduction Cost

= Row Reduction Cost + Colum Reduction Cost

= 20+3

= (23)

Reduction Matrix A is $\begin{bmatrix} \infty & 7 & 5 & 0 \\ 0 & \infty & 5 & 8 \\ 0 & 0 & \infty & 5 \\ 3 & 7 & 0 & \infty \end{bmatrix}$

$$\hat{c}(1) = 23$$

Steps to find the cost corresponding to every vertex

Suppose edge (i,j) is added

1) Make all entries in row i and column j as ∞

2) Make entry $(j,1)$ as ∞

3) Reduce cost matrix by subtracting amt r

$$\hat{c}(s) = \hat{c}(R) + A[i,j] + r$$

For node 2, consider path $(1,2)$ (i,j)

Make all entries in row 1 and column 2 as ∞ and $A[2,1]$ ($j,1$) as ∞

$$\begin{matrix} & 1 & 2 & 3 & 4 \\ 1 & \infty & 7 & 5 & 0 \\ 2 & 0 & \infty & 5 & 8 \\ 3 & 0 & 0 & \infty & 5 \\ 4 & 3 & 7 & 0 & \infty \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & 5 & 8 \\ 0 & \infty & \infty & 5 \\ 3 & \infty & 0 & \infty \end{bmatrix}$$

Perform row reduction & col^n reduction except for row & col. containing ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & 5 & 8 \\ 0 & \infty & \infty & 5 \\ 3 & \infty & 0 & \infty \end{bmatrix} \begin{matrix} \text{Min} \\ 5 \\ 5 \\ 0 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 3 \\ 0 & \infty & \infty & 5 \\ 3 & \infty & 0 & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 0 \\ 0 & \infty & \infty & 2 \\ 3 & \infty & 0 & \infty \end{bmatrix}$$

Row Red^n → Min 0 0 3 → Col^n Reduct^n.

Total Red^n Cost = Total Row Red^n Cost + Total Col^n Red^n Cost

$= 5 + 3$

$= 8$

$\therefore \quad \hat{c}(2) = \hat{c}(1) + A[1,2] + r$

$= 23 + 7 + 8$

$= 38$

For node 3, consider path (1,3)

Make all entries in row 1 and column 3 of A as ∞ and A[3,1] as ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 8 \\ \infty & 0 & \infty & 5 \\ 3 & 7 & \infty & \infty \end{bmatrix}$$

$$A = \begin{bmatrix} \infty & 7 & 5 & 0 \\ 0 & \infty & 5 & \infty \\ 0 & 0 & \infty & 5 \\ 3 & 7 & 0 & \infty \end{bmatrix}$$

Perform row redⁿ & Column redⁿ except for row & col. containing ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 8 \\ \infty & 0 & \infty & 5 \\ 3 & 7 & \infty & \infty \end{bmatrix} \begin{matrix} \text{Min} \\ 0 \\ 0 \\ 3 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 8 \\ \infty & 0 & \infty & 5 \\ 0 & 4 & \infty & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 03 \\ \infty & 0 & \infty & 0 \\ 0 & 4 & \infty & \infty \end{bmatrix}$$

Min 0 0 5

Row Reduction → Col. Reduction →

Total Redⁿ Cost (r) = 3 + 5 = 8

$$\therefore \hat{c}(3) = \hat{c}(.) + A[1,3] + r$$
$$= 23 + 5 + 8$$
$$\boxed{\hat{c}(3) = 36}$$

For Node 4, consider path (1,4)

Make all entries in row 1 & column 4 of A as ∞ and A[4,1] = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 5 & \infty \\ 0 & \infty & \infty & \infty \\ \infty & 7 & 0 & \infty \end{bmatrix}$$

Perform row reductn & column reductions except for row & col. containing ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 5 & \infty \\ 0 & 0 & \infty & \infty \\ \infty & 7 & 0 & \infty \end{bmatrix} \begin{matrix} \text{Min} \\ 0 \\ 0 \\ 0 \end{matrix}$$

Min: 0 0 0

As rows & columns are reduced.
Total red^n cost (r) = 0

$$\hat{c}(4) = \hat{c}(1) + A(1,4) + r$$
$$= 23 + 0 + 0$$
$$\boxed{\hat{c}(4) = 23}$$

As LCBB is used among the three cost live nodes
$\min(\hat{c}(2), \hat{c}(3), \hat{c}(4)$ is used for exploring
$= \min(38, 36, 23)$
$= 23.$

∴ Select node 4.

1 & 4 are selected so remaining nodes are 2 & 3. So now find cost of 5 & 6 in the state space tree

Now, after performing 1 to 4, now the reductn matrix is obtain from node 4 ie

$$\text{Reduction Matrix } A = \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 5 & \infty \\ 0 & 0 & \infty & \infty \\ \infty & 7 & 0 & \infty \end{bmatrix}$$

Consider path(4,2), for node 5

Make all entries row 4, row 2 and col 2 of A as ∞

as $A[2,1] = \infty$.
$A[4,1] = \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & 5 & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Perform row red^n + col red^n except for row + col containing ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 5 & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} \xrightarrow{\text{Min } 5} \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 0 & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Min 0 0

Total Red^n Cost ($\hat{r}$) = 5 + 0 = 5

$$\hat{c}(5) = \hat{c}(4) + A[4,2] + \hat{r}$$
$$= 23 + 7 + 5 = 35$$

$$\boxed{\hat{c}(5) = 35}$$

Node 6, consider path (4, 3)

Make all entries of row 4 & col 3 of A = ∞ & A[4, 1] = ∞ & A[3, 1] = ∞

$$A = \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 5 & \infty \\ 0 & 0 & \infty & \infty \\ \infty & 7 & 0 & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Perform row + Col. Reduction

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} \text{Min} \\ 0 \\ 0 \\ \text{Min } 0 \end{matrix} \quad 0$$

As row & cols are reduced total red^n cost ($\hat{r}$) = 0

$$\therefore \hat{c}(6) = \hat{c}(4) + A(4,3) + \hat{r}$$
$$= 23 + 0 + 0$$

$$\boxed{\hat{c}(6) = 23}$$

Among node to explored further is the node with min cost value. ∴ from State space tree we can see Node 6 is min & Can be explored.

For Node 7, path is ~~(3,2)~~ (1,4,3,2)

Make all entries of row 1, row 4, row (3) & col 2 of A as ∞ & A[2,1] = ∞, A[4,1] = ∞, A[3,1] = ∞

$$A = \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Perform Row & Col Redn.

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Min (row) — Min (col)

No row & col are reduced

$r = 0$

$\therefore \hat{c}(7) = \hat{c}(6) + A[3,2] + r$

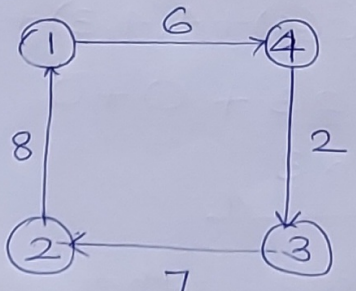
$= 23 + 0 + 0$

$\hat{c}(7) = 23$

The path As the the solution tour is obtained ie 1-4-3-2-1 with cost = 23.

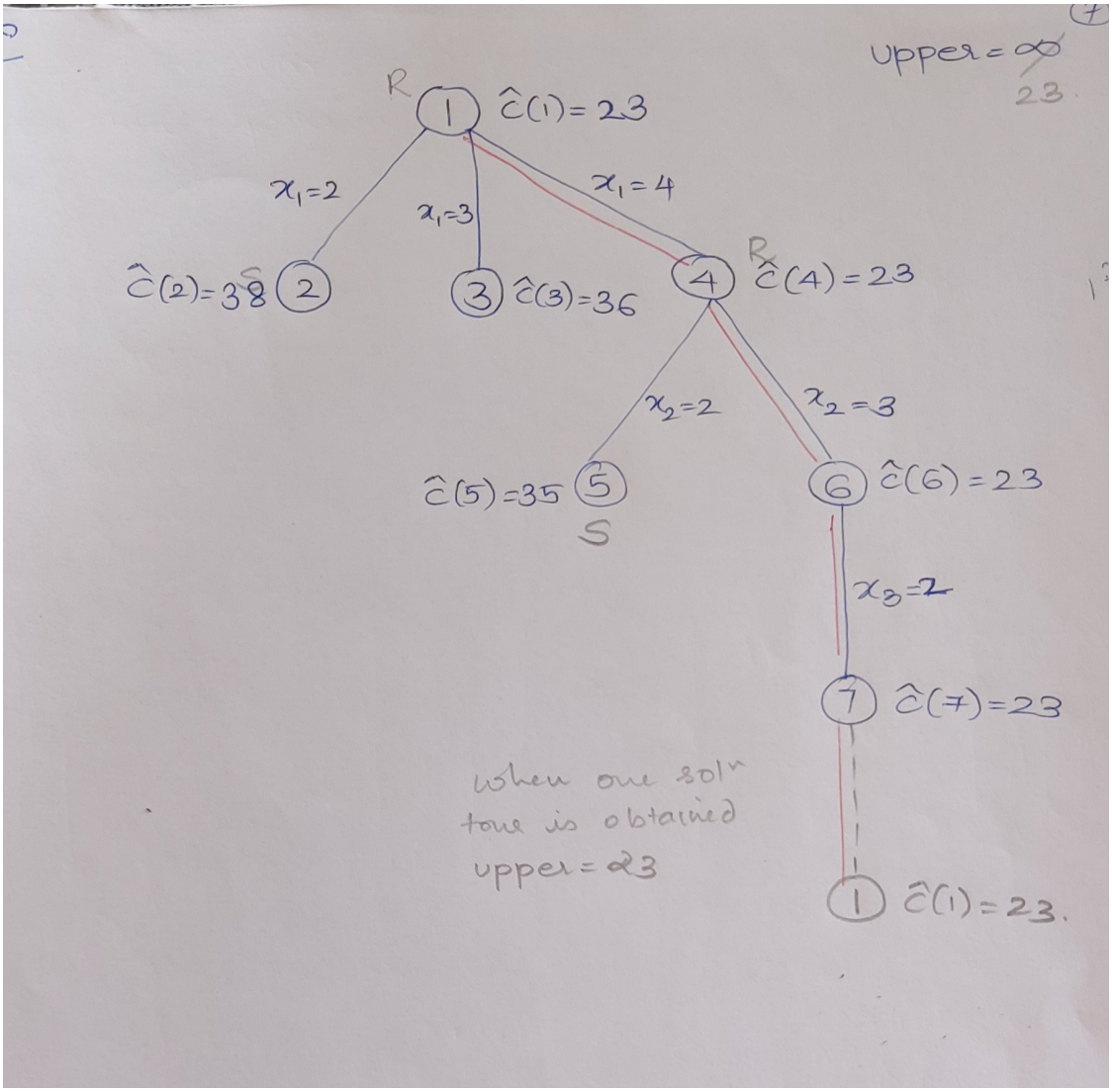
is reached, upper is updated to 23.

Check other live nodes ie 2, 3, 5. Next Enode is Node 5 & have cost greater than 23. So this is only path.



Cost = 8+6+2+7

= 23.


upper = ∞
23
R
1 ĉ(1) = 23
x1 = 2
x1 = 3
x1 = 4
ĉ(2) = 38 2
3 ĉ(3) = 36
R
4 ĉ(4) = 23
x2 = 2
x2 = 3
ĉ(5) = 35 5
S
6 ĉ(6) = 23
x3 = 2
7 ĉ(7) = 23
when one soln tree is obtained upper = 23
1 ĉ(1) = 23.