UNIT VI

BRANCH AND BOUND

Branch and Bound is another method to systematically search a solution space. Just like backtracking, we will use bounding functions to avoid generating sub trees that do not contain an answer node. However branch and Bound differs from backtracking in two important manners:

1. It has a branching function, which can be a depth first search, breadth first search or based on bounding function.

2. It has a bounding function, which goes far beyond the feasibility test as a mean to prune efficiently the search tree.

3.

Branch and Bound refers to all state space search methods in which all children of the E-node are generated before any other live node becomes the E-node Branch and Bound is the generalization of both graph search strategies, BFS and D- search.

- A BFS like state space search is called as FIFO (First in first out) search as the list of live nodes in a first in first out list (or queue).

- A D search like state space search is called as LIFO (Last in first out) search as the list of live nodes in a last in first out (or stack).
- Definition 1: Live node is a node that has been generated but whose children have not yet been generated.
- Definition 2: E-node is a live node whose children are currently being explored. In other words, an E-node is a node currently being expanded.
- Definition 3: Dead node is a generated node that is not to be expanded or explored any further. All children of a dead node have already been expanded.
- Definition 4: Branch-an-bound refers to all state space search methods in which all children of an E-node are generated before any other live node can become the E-node.

Least Cost (LC) search:

- In both LIFO and FIFO Branch and Bound the selection rules for the next E-node in rigid and blind. The selection rule for the next E-node does not give any preference to a node that has a very good chance of getting the search to an answer node quickly.

Control Abstraction for LC-Search:

- Let t be a state space tree and c() a cost function for the nodes in t. If x is a node in t, then c(x) is the minimum cost of any answer node in the sub tree with root x. Thus, c(t) is the cost of a minimum-cost answer node in t.
- A heuristic
- $c\square$ (.) is used to estimate c(). This heuristic should be easy to compute and
- generally has the property that if x is either an answer node or a leaf node, then
- c(x) =
- $c\square(x)$.
- LC-search uses $c\square$ to find an answer node. The algorithm uses two functions Least() and Add() to delete and add a live node from or to the list of live nodes, respectively.

- Least() finds a live node with least c(). This node is deleted from the list of live nodes and returned.

Bounding:

- A branch and bound method searches a state space tree using any search mechanism in which all the children of the E-node are generated before another node becomes the E-node.
- We assume that each answer node x has a cost c(x) associated with it and that a minimum-cost answer node is to be found.
- Three common search strategies are FIFO, LIFO, and LC. The three search methods differ only in the selection rule used to obtain the next E-node.

Branch and Bound Applications are

1. 0/1 Knapsack Problem-LCBB
2. Travelling Sales Person Problem-LCBB
3. 15 Puzzle Problem-LCBB
4. FIFO Branch & Bound
5. FIFO Branch & Bound Solution

LC Search Algorithm:

```
listnode = record {
    listnode *next, *parent; float cost;
}

Algorithm LCSearch(t)
// Search t for an answer node.
{
    if *t is an answer node then output *t and return;
    E := t; // E-node.
    Initialize the list of live nodes to be empty;
    repeat
    {
        for each child x of E do
        {
            if x is an answer node then output the path
                from x to t and return;
            Add(x); // x is a new live node.
            (x -> parent) := E; // Pointer for path to root.
        }
        if there are no more live nodes then
        {
            write ("No answer node"); return;
        }
        E := Least();
    } until (false);
}
```

live nodes

0/1 Knapsack problem using Least-cost:-

→ 0/1 knapsack problem we learn in previous units also like greedy method, dynamic programming--etc.

→ The main aim of this problem is to get maximized profits

→ For Example we take given problem.

profits	40	10	12	18
weights	2	4	6	9

Total capacity is = 15

→ To fill the bag by those objects, the total weight included in the bag is must be less than [or] equal to bag capacity.

→ In this problem we follow one condition, that is If you take any object then take completly otherwise leave it (or) don't take.

→ For Suppose, in the above example weights are 2, 4, 6 and 9 From these take complete weight like 2, 4, 6, 9 don't take like 1, 3, 4, 5

→ In Branch and Bound the problem is solved by minimization Approach but the solution is maximized.

→ That means, In this all the +ve profits are converted into -ve profits. (maximized → minimization).

→ In Branch and Bound this problem is solved by using state space tree.

→ For this, First of calculate upper Bound and cost of all nodes in the State space tree.

LC 0/1 Knapsack Algorithm (Upper Bound):

```
Algorithm UBound(cp, cw, k, m)
// cp, cw, k, and m have the same meanings as in
// Algorithm 7.11. w[i] and p[i] are respectively
// the weight and profit of the ith object.
{
    b := cp; c := cw;
    for i := k + 1 to n do
    {
        if (c + w[i] ≤ m) then
        {
            c := c + w[i]; b := b − p[i];
        }
    }
    return b;
}
```

Upper Bound :-

→ $\sum_{i=1}^{n} p_i x_i \leq m$ (without Fractions)

cost :- → $\sum_{i=1}^{n} p_i x_i \leq m$ (with Fractions)

Steps :-

① Initially the upper Bound is '∞'
Consider the first node. and costruct state space tree and calculate upper Bound and cost

$\hat{u} = -32$ ① $u = \infty$
$c = -38$ ①

$\hat{u} = 10+10+12 = 32$ $c = 10+10+12+\frac{18}{2}\times 3 = 38$
$2+4+6 \leq m$ $2+4+6$
$15-12 = 3$

After finding upper Bound then $\hat{u}$ is less then u then update (∞) replace $\hat{u}$ in the place of u.

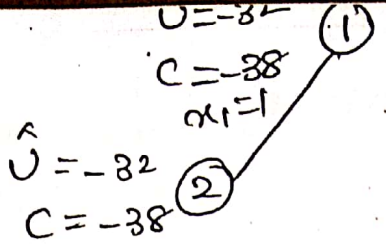
$\hat{u} = -32$ ① $u = -32$
$c = -38$

Note :-

(1) In this problem always check $\hat{u}$ is lesser then u then update (or) replace $\hat{u}$ in the place of u. ($\hat{u} < u$).
② The cost is greater then the 'u' then Stop the node. No need to find out Solution of that node

node.

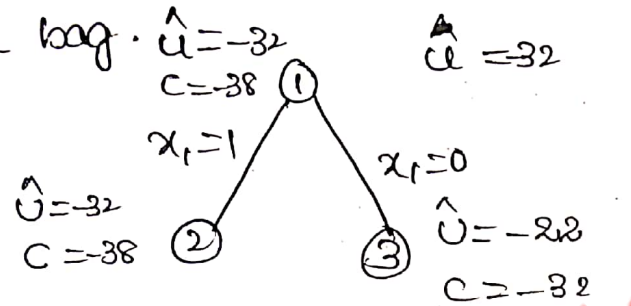
② step :
In this we consider next node that is 2, In this the x_i object weight is included in the bag. ($x_i = 1$) then calculate upper Bound and cost.



→ Based on the least cost then $\hat{u} = -32$ the node will be selected for next step.

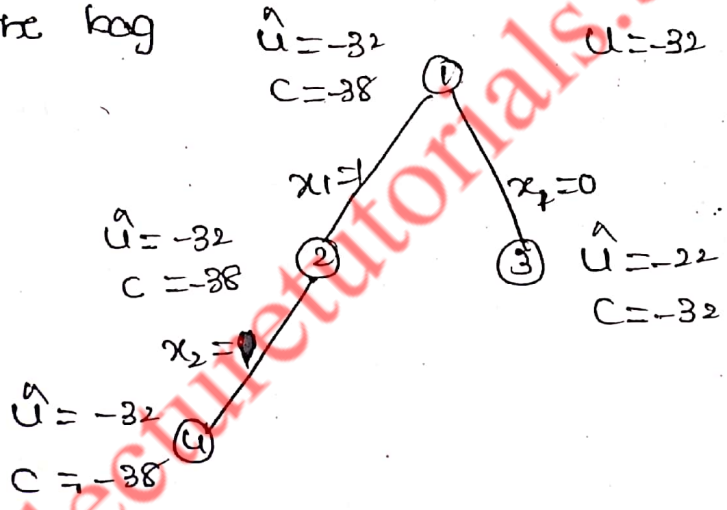
$$c = 10+10+12+\frac{18}{9}\times 3$$
$$2+4+6$$
$$c = -38$$

Next consider node ③ in this the x_1 is not included in the bag.



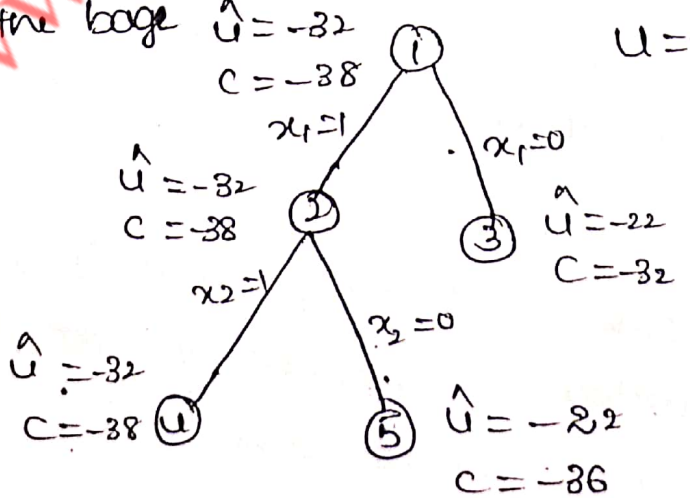
$$\hat{u} = 10+12 = 22$$
$$c = 10+12+\frac{18}{9}\times 5$$
$$= -32$$

Next consider node ① in this the object x_2 is included in the bag



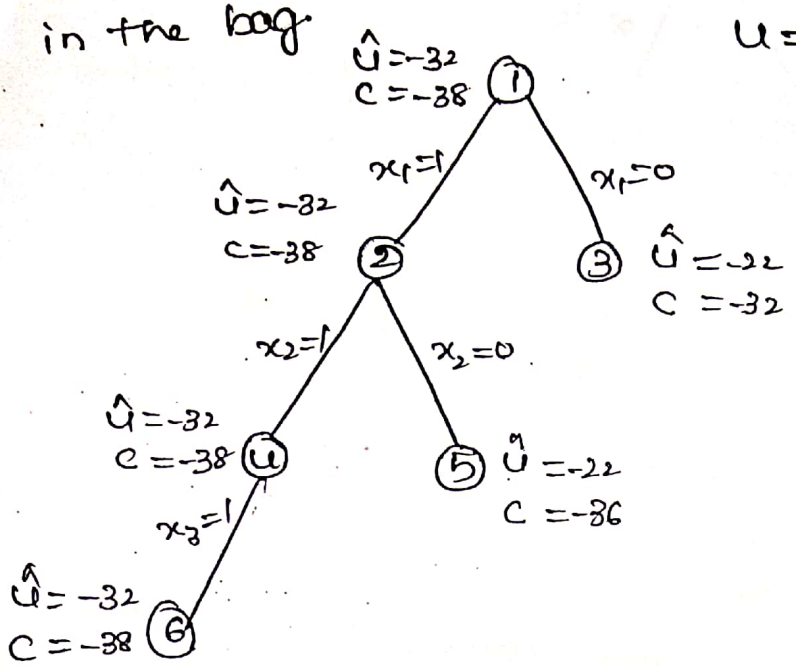
$$\hat{u} = 10+10+12 = 32$$
$$c = 38$$

Next step consider node ⑤ in this x_2 is not included in the bag



$$\hat{u} = 10+12 = 32$$
$$c = 10+12+\frac{18}{9}\times 7 = 38$$

Next step we consider node 6 in this the x_3 is included in the bag

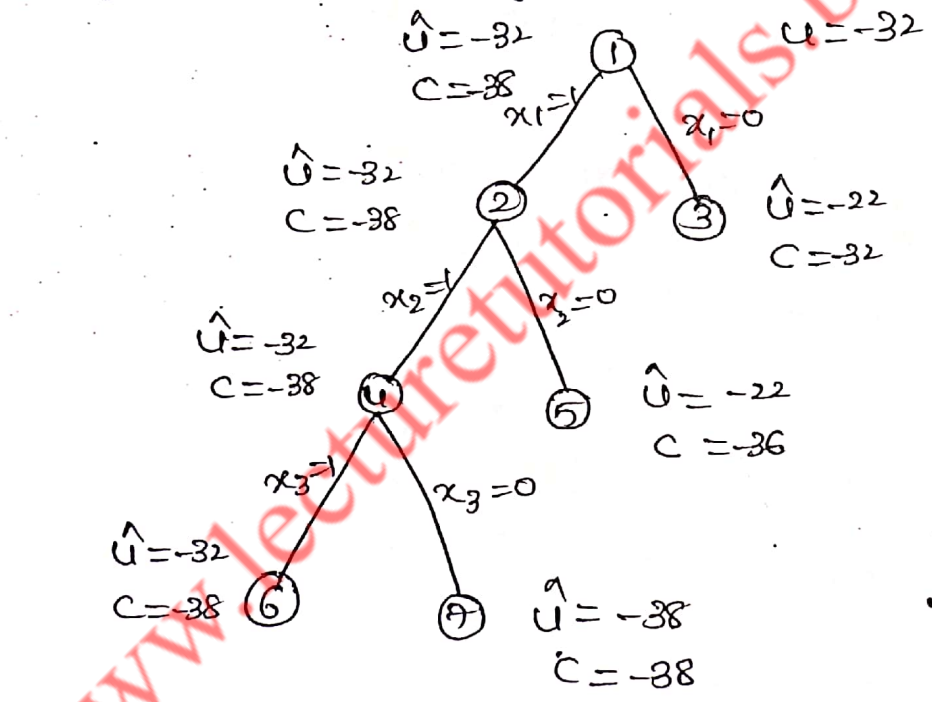


$\hat{u} = 10+10+12 = 32$

$c = 10+10+12+\frac{18}{9}3$

$= 38$

Next step we consider node: 7 in this the x_3 is not included in the bag



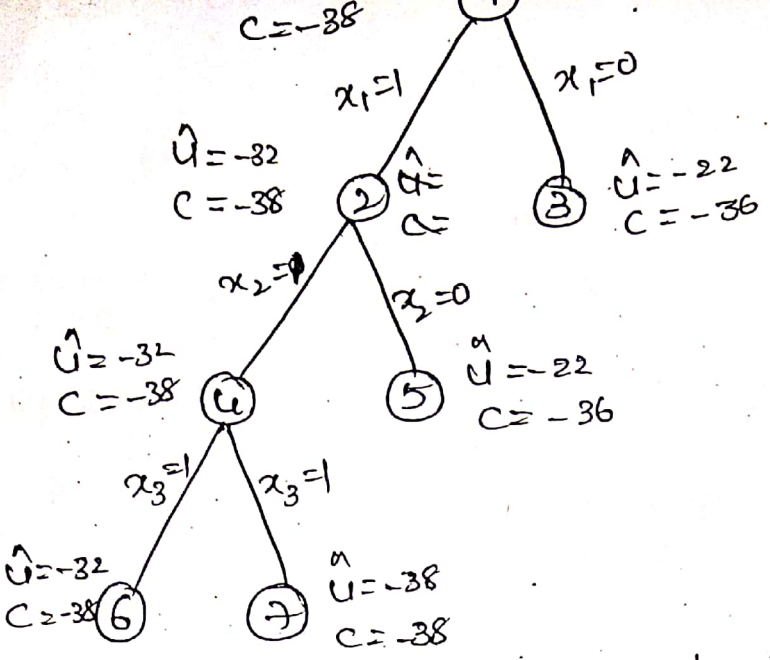
$\hat{u} = 10+10+18 = 38$

$c = 10+10+18 = 38$

In the above one node ② upper bound is lessed than u, then replace u has $\hat{u}$ ~~(u^ etc)~~

And then, compare the all nodes costs to upper bound

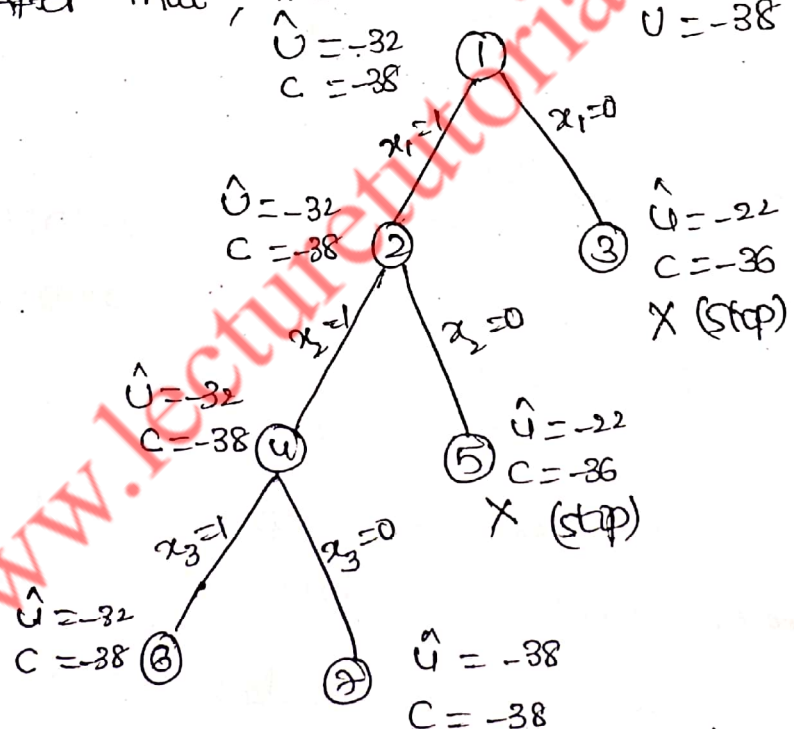
→



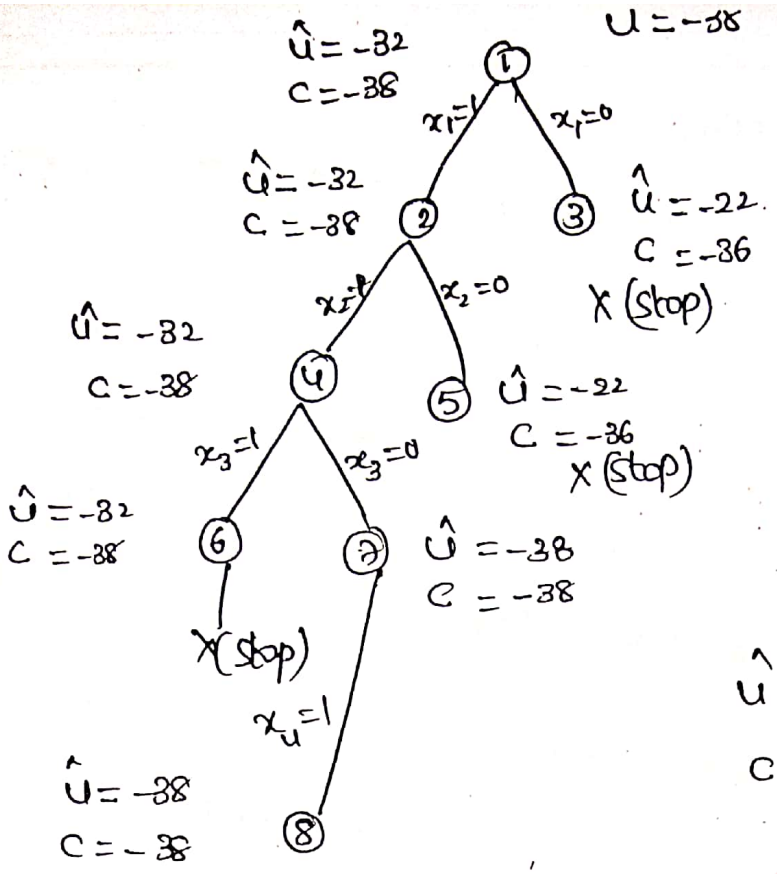
Then compare cost all nodes costs to the upper Bound, The cost is greater than the upper Bound then leave(or) stope that node.

3 and 5 nodes are greater than the upper Bound then leave those nodes.

After that, the state space tree is,



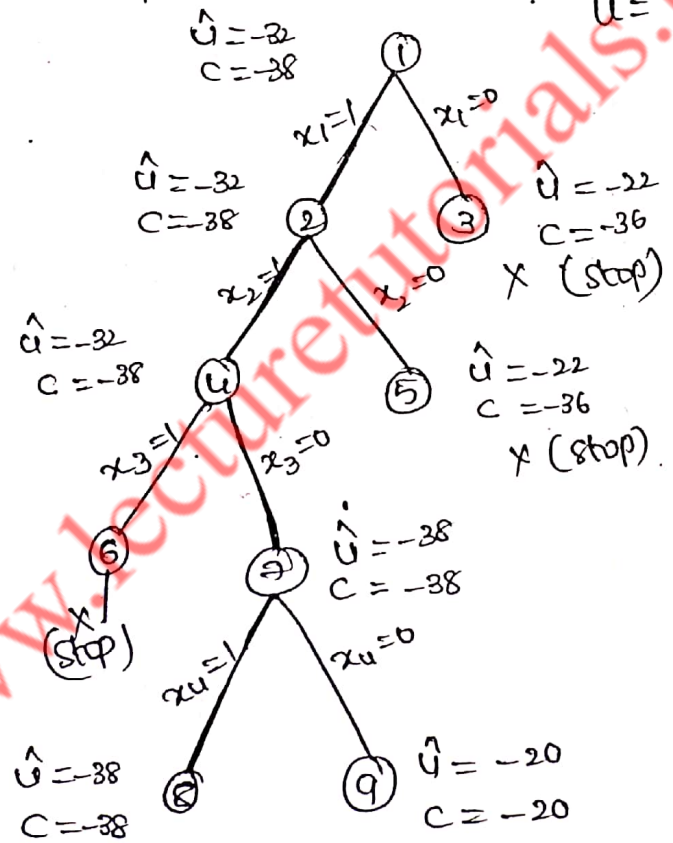
Next step to select 7 why because the node 6 is have weights upto 12 then we have to add one more the bag capacity will be exceed then stop node 6 and select node 7 for Next step.



û = 10+10+18 = 38

c = 38

Then next step x_4 is not included in the bag.



Then $x_1=1$ $x_2=1$ $x_3=0$ $x_4=1$

Then find out $\sum_{i=1}^{n} w_i x_i$ and $\sum_{i=1}^{n} p_i x_i$

→ $\sum_{i=1}^{n} p_i x_i$

→ $p_1 x_1 + p_2 x_2 + p_3 x_3 + p_4 x_4$

→ $10 \times 1 + 10 \times 1 + 12 \times 0 + 18 \times 1$

$= 10 + 10 + 0 + 18$

$= 38$ //

→ $\sum_{i=1}^{n} w_i x_i \leq m$

→ $w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 \leq 15$

→ $2 \times 1 + 4 \times 1 + 6 \times 0 + 9 \times 1 \leq 15$

→ $2 + 4 + 0 + 9 \leq 15$

→ $15 \leq 15$ ✓

$\therefore x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1$

Then maximized profits $= 38$ ✓

Traversal Sales problem in Branch and Bound.

→ consider Graph G(V,E) of a weighted where.

V = Set of vertices

E = Set of Edges.

→ As per the Branch and Bound, the given graph must be converted into equivalent adjabray cost matrix.

→ check whether the matrix what we developed is either reduced matrix (or) not.

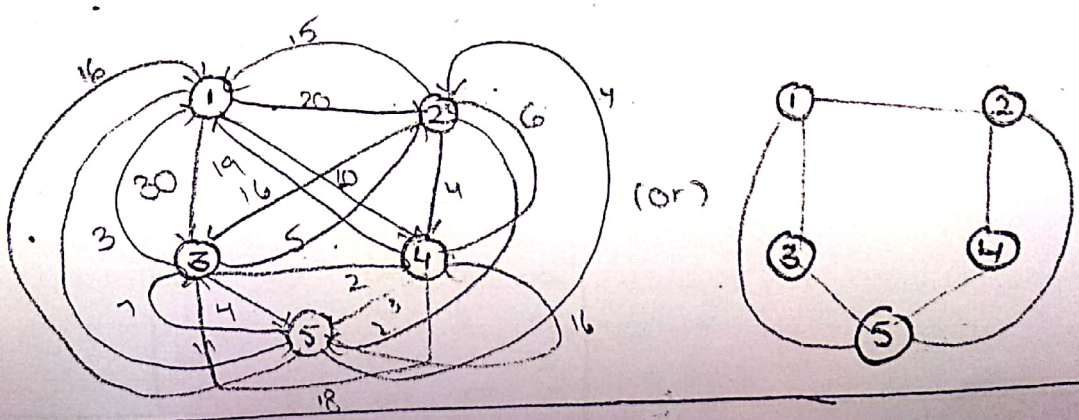
Reduced matrix :- If a matrix contains atleast one '0' either in all raws and in all columns, then it is said to be as "Reduced Matrix"

→ If it is not Reduced, then by following below two steps we can make it is Reduced.

(1) Raw Reduction

(2) column Reduction.

Ex:-



The matrix for the above graph is.

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

→ check whether it is Reduced (or) Not

→ If not then 1) Row Reduction

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \begin{matrix} -10 \\ -2 \\ -2 \\ -3 \\ -4 \end{matrix} \rightarrow \begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix}$$

(21)

Row Reduction

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix} \rightarrow \begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

1 0 3 0 0 = (4)

Column Reduction

Now the final matrix after Reduction is $A=\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$

Total cost of reduction: $c(r)+c(c)$

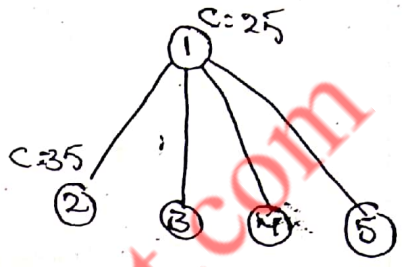
$= 21+4$

$= 25$

We have to find all the costs for each node.

→ As we start from ① node we can visit remaining nodes like ② ③ ④ ⑤.

→ Now find the cost at node ②, so Consider Path 1–2



→ To find the cost at any node, we have to follow the basic Branch and Bound Principle.

ie. (i,j) → Replace ith row elements '∞'

1) → Replace jth column elements '∞'

2) (j,1) → so that we can't go Back.

→ for node ② Path (1,2), $= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix}$ Here the total Reduction cost = 0 ie R(2) = 0

$\therefore \hat{c}(2) = A(1,2) + R(2) + \hat{c}(1)$

$= 10 + 0 + 25 = 35$

$\therefore \boxed{\hat{c}(2) = 35}$

→ Similarly for Path (1,3) $= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 15 & 3 & \infty & \infty & 0 \\ 11 & 0 & \infty & 12 & \infty \end{bmatrix}$

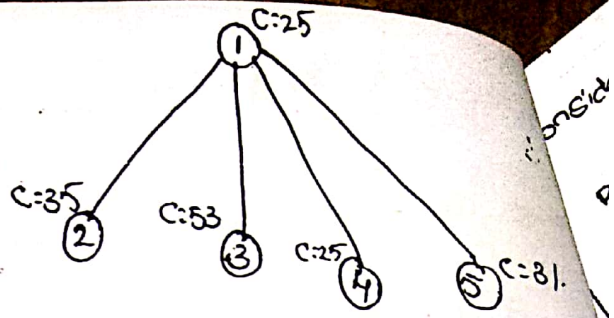
(11)

Column Reduction ⇒ $\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$

$\hat{c}(3) = \hat{c}(1) + A(1,3) + R(3)$

$= 25 + 17 + 11$

$= 53$



for path(1,4):

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & \infty \\ 0 & 3 & \infty & \infty & 0 \\ \infty & 3 & 12 & \infty & 2 \\ 11 & 0 & 0 & \infty & 0 \end{bmatrix}$$

R=0

$\Rightarrow \hat{c}(4) = \hat{c}(1) + A(1,4) + R(4)$

$= 25 + 0 + 0$

$= 25$

for path(1,5):

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 15 & 3 & 12 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix} \begin{matrix} \\ \\ \leftarrow 2 \\ \\ \leftarrow 3 \end{matrix}$$

$\Rightarrow$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$

R = 2+3

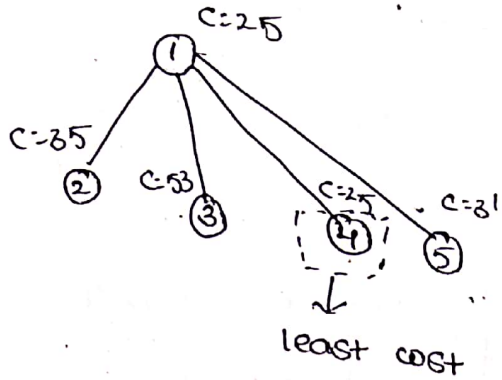
= 5

$\Rightarrow \hat{c}(5) = \hat{c}(1) + A(1,5) + R(5)$

$= 25 + 1 + 5$

$= 31$

$\Rightarrow$



Consider all the nodes, (4) is least cost, so find path from that node by following Branch and Bound Principle.

→ As we start from (1) and next (4), and we are moving from (4), so consider (1,4) path matrix as 'A'.

$$\therefore A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

for (4,2) $= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix}$

$\hat{c}(2) = \hat{c}(4) + A(4,2) + R$

$= 25 + 3 + 0$

$= 28$

for (4,3): $\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ \infty & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$ (-2) $\Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$

(11) Row Reduction

$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 11 & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & 3 & 12 & \infty & 0 \\ 0 & 0 & 0 & \infty & \infty \end{bmatrix}$ then $R(3) = 2 + 11$

$= 13$

Column Reduction

$\hat{c}(3) = \hat{c}(4) + A(4,3) + R(3) = 25 + 12 + 13 = 50$

for (4,5) $= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix}$ (11) $\Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 0 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix}$

$\hat{c}(5) = \hat{c}(4) + A(4,5) + R(5)$

$= 25 + 0 + 11$

$= 36$

$=$

→ Now select least cost and again start the tour.

Consider (4,2) $\Rightarrow$ $A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix}$

for(2,3) $= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix}$ ② $\Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix}$ Row Reduction

① ⑪

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

$R = 2 + 11 = 13$

$=$

$\hat{c}(3) = A(2,3) + R + \hat{c}(2)$

$= 11 + 13 + 28$

$= 24 + 28$

$= 52$

$=$

for(2,5): $\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix}$ $\Rightarrow \hat{c}(5) = \hat{c}(2) + R + A(2,5)$

$= 28 + 0 + 0$

$= 28$

$\hat{c}(5) = 28$

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

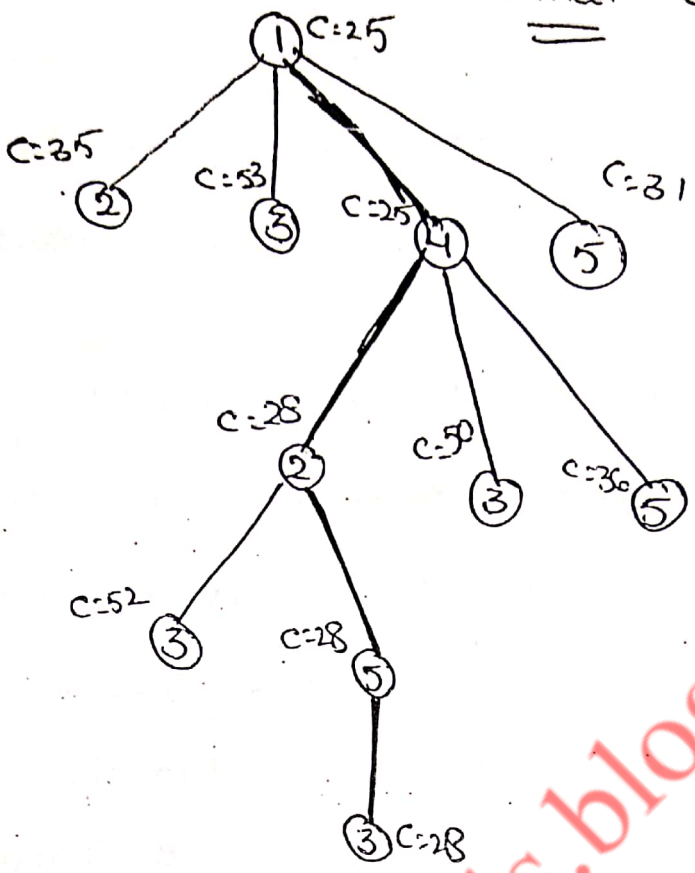
Again from ⑤ $\Rightarrow$ Path (5,3): $\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$ $\hat{c}(3) = A(5,3) + R + \hat{c}(5)$

$= 0 + 0 + 28$

$= 28$

$=$

Final State Space Tree



∴ The Path is 1-4-2-5-3-1

Cost=28

BOUNDING:-

A feasible solution, which either Maximizes (or) minimizes a given objective function is called optimal solution... Depth first node generation with bound function is called Back tracking

→ Branch and Bound discovers branches with in the complete search space by using estimated bounds to limit the number of possible solutions.

→ The different types define different strategies to explore the search space and generate branches.

FIFO (FIRST IN FIRST OUT):-

The oldest node in the queue is used to extend the branch. This leads to breadth-first search, where all nodes at depth d are visited first, before any nodes at depth d+1 are visited.

LC (Least count):-

The branch is extended by the node which adds the lowest additional costs, according to given cost function. The strategy of traversing the search space is therefore defined by the cost function.

BRANCH AND BOUND:-

It is Breadth first search and similar to Back tracking.

→ The solution is also represented in the state space tree.

→ This method is useful to solve the optimization and minimization problem.

→ The solution represented as two ways.

(1) variable size solution

(2) fixed size solution

→ In variable size solutions changes are possible.

→ But in fixed size solutions changes are not possib

ex:-

→ Consider Jobs = { x_1, x_2, x_3, x_4 }

profits = { 10, 5, 18, 13 }

weights = { 1, 2, 1, 2 }

(1) solution = { x_1, x_2, x_3 }

↓

variable size solutions.

→ Here we can add (or) remove the variables

(2) solution = { 0, 1, 0, 1 }

↓

fixed size solution.

→ If we once fix the solution. we cannot change the values.

* In Least Count and FIFO the two factors are same :-

(1) Calculating the upper bound &

(2) cost

→ In upper bound, we should not consider the fraction values. It can be represented as '$\hat{U}$'

→ $\hat{U}$ = It is the sum of all profits according to the suitable weights.

an cost, consider the fraction. values.

→ It can be denoted as '$\hat{c}$'

→ $\hat{c}$ = sum of the total profits with fraction according to their weights.

→ Bound values and cost always represented as negation to the actual result

→ The global bounding value is 'infinity' (∞). Th value must be changed if gets smaller upper boun value.

FIFO B

FIFO Branch and Bound:-

Bag capacity (m) = 15

Eg:-

10	12	15	18
2	3	6	9

→ Upper bound is not contain the fraction values

$\hat{u}$ = 2 + 3 + 6

= 10 + 12 + 15 = 37

∴ we can't consider the last profit because of exceed bag capacity

→ Cost is contain the fraction values

$\hat{c}$ = 2 + 3 + 6

$= 10 + 12 + 15 + \frac{18}{9} \times 4$

= 10 + 12 + 15 + 8 = 45

→ we can take the four parts in profit 18

Step1:- we can take Global upper bound is ∞

① $\hat{u} = -37$

$\hat{c} = -45$

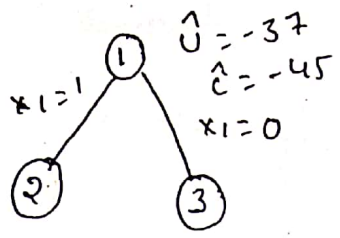
∴ upperbound is minimum than Global upperbound, so we can change or replace it.

∴ Global upperbound = ϕ

= -37

∴ So, we can extend the node ①

Step2:-



Global UB = ∞
= -37

→ In node ② , x_1 is include so we can replace same values

→ In node ③ we cannot include x_1

10	12	15	18
2	3	6	9

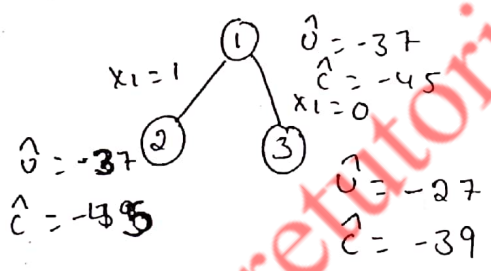
⇐ ($x_1 = 0$)

$\hat{c}$ = 12 + 15 + $\frac{18}{9}^2$ x 6

= 3 + 6 + 9

= 12 + 15 + 12 = 39

$\hat{u}$ = 12 + 15 = 27



Global UB = -37

→ node ② and node ③ upper bounds are maximum we cannot replace it.

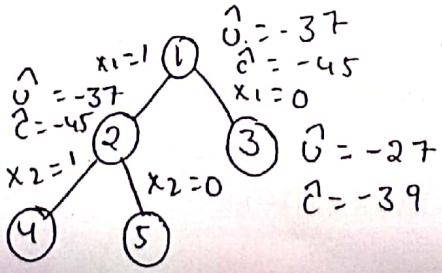
→ Now, we can check cost > Global U.B

node ②:- -45 > -37 false

node ③:- -39 > -37 false

∴ So, we can't kill node ② and node ③. So we can extended present node ⑤

Step3:-



→ In node (4) we can include x_2 so we can replace

→ In node (5) we can't include x_2

$(x_1=1, x_2=0)$

10	12	15	18
2	3	6	9

$\hat{u} = 2+6$

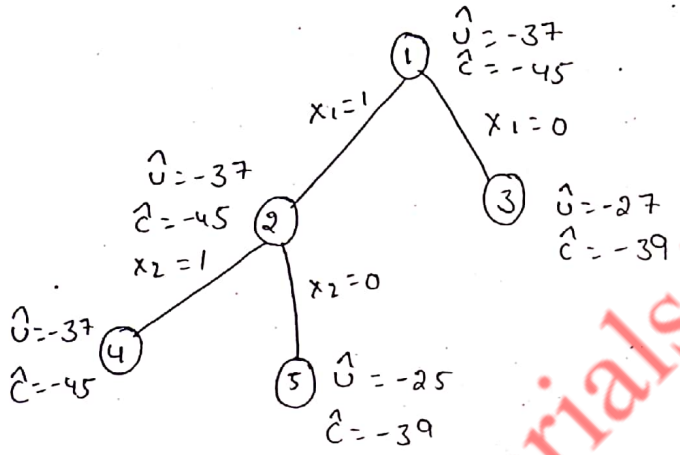
$= 10+15 = 25$

$\hat{c} = 2+6$

$= 10+15+\frac{18}{9}^{2} \times 7$

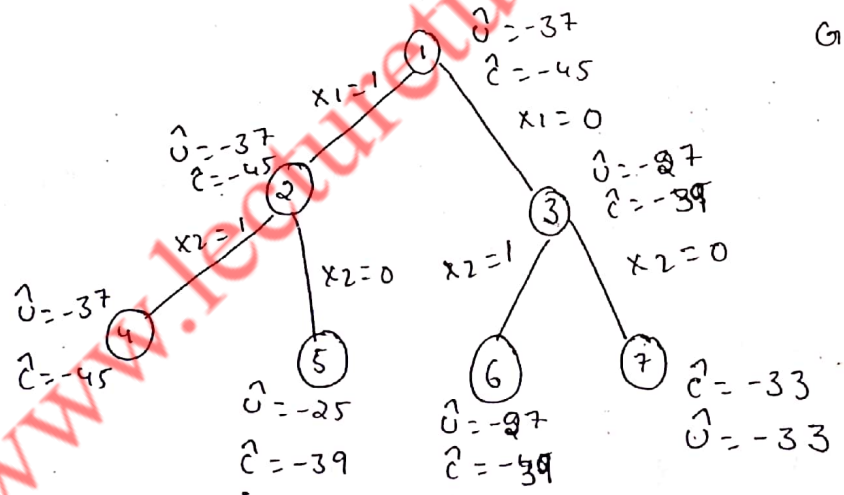
$= 39$

Global UB = -37



∴ Now we can extend the node (3)

Global UB = -37



→ In node (6) we can replace same as above values

→ In node (7) we can find

$(x_1=0, x_2=0)$

10	12	15	18
2	3	6	9

$\hat{c} = 15+18$

$= -33$

$\hat{u} = 15+18$

$= -33$

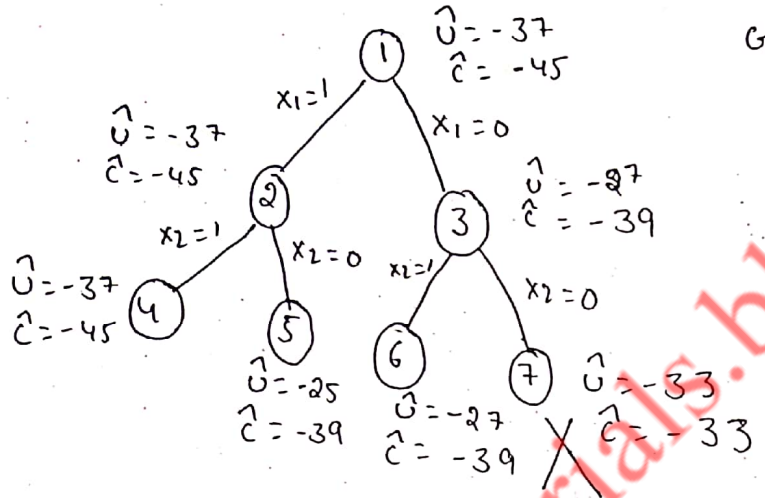
∴ All nodes upper bounds are maximum than global upperbou

So. cannot replace it

→ we can check the cost > Global U.B

node ④	-45 > -37	false
node ⑤	-25 > -37	false
node ⑥	-39 > -37	false
node ⑦	-33 > -37	true

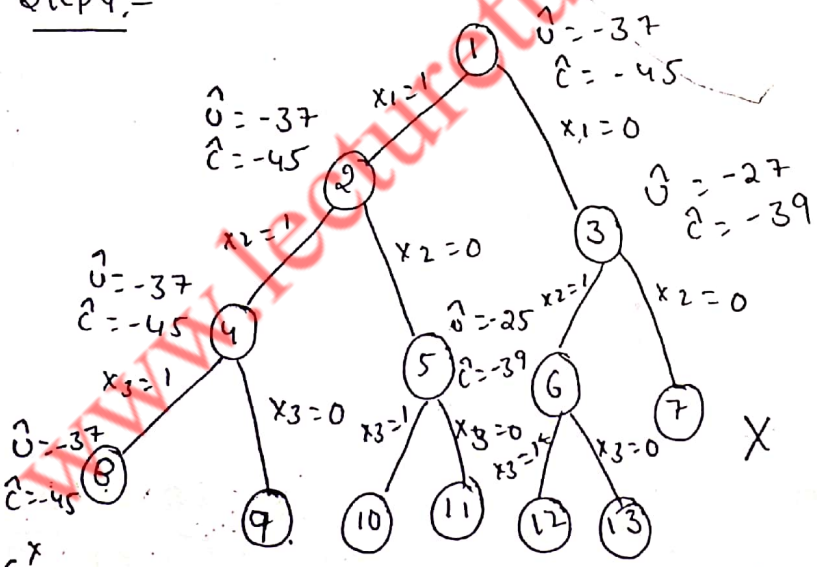
∴ So, we can kill the node ⑦



Global UB = -37

∴ So, we can extended the nodes ④, ⑤, ⑥

Step4:-



Global UB = -37

~~On node ⑥ we can including so we can place as it is.~~ X

~~→ On node ⑦ we can't include x3~~

→ node ⑧ we can include x3, as same above we can replace

$(x_1=1, x_2=1, x_3=0)$

10	12	~~15~~	18
2	3	6	9

$\Rightarrow$

node (9):- $\hat{u} = 10+12+18 = 22+18 = 40$

$\hat{c} = 10+12+\frac{18}{9}\times 9 = 40$

node (8):- $\hat{c} = 10+15+\frac{18}{9}\times 7$

$= 25+14 = 39$

$\hat{u} = 10+15 = 25$

$(x_1=1, x_2=0, x_3=1)$

10	~~12~~	15	18
2	3	6	9

node (11):- $\hat{c} = 2+9$

$= 10+18 = 28$

$\hat{u} = 10+18 = 28$

$(x_1=1, x_2=0, x_3=0)$

10	~~12~~	~~15~~	18
2	3	6	9

node (12):- $\hat{c} = 3+6$

$= 12+15+\frac{18}{9}^{2}\times 6$

$= 39$

$\hat{u} = 12+15$

$= 27$

$(x_1=0, x_2=1, x_3=1)$

~~10~~	12	15	18
2	3	6	9

node (13):- $\hat{c} = 12+18$

$= 30$

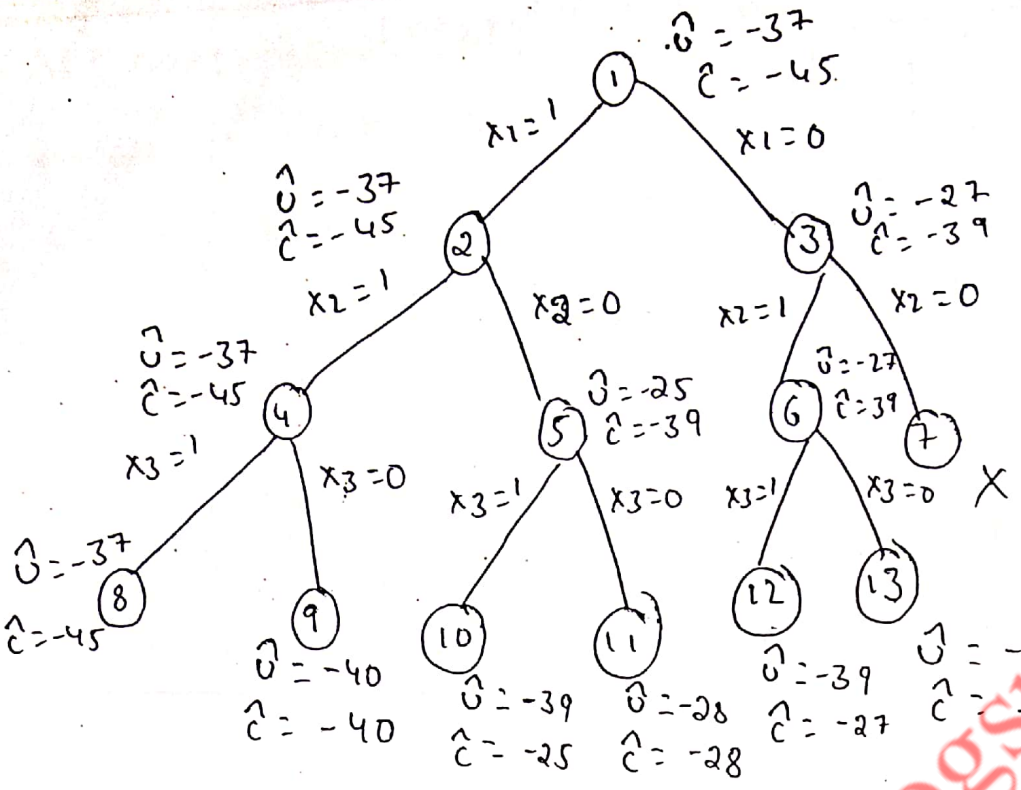
$\hat{u} = 12+18$

$= 30$

$(x_1=0, x_2=1, x_3=0)$

~~10~~	12	~~15~~	18
2	3	6	9

Global UB = -3
-40



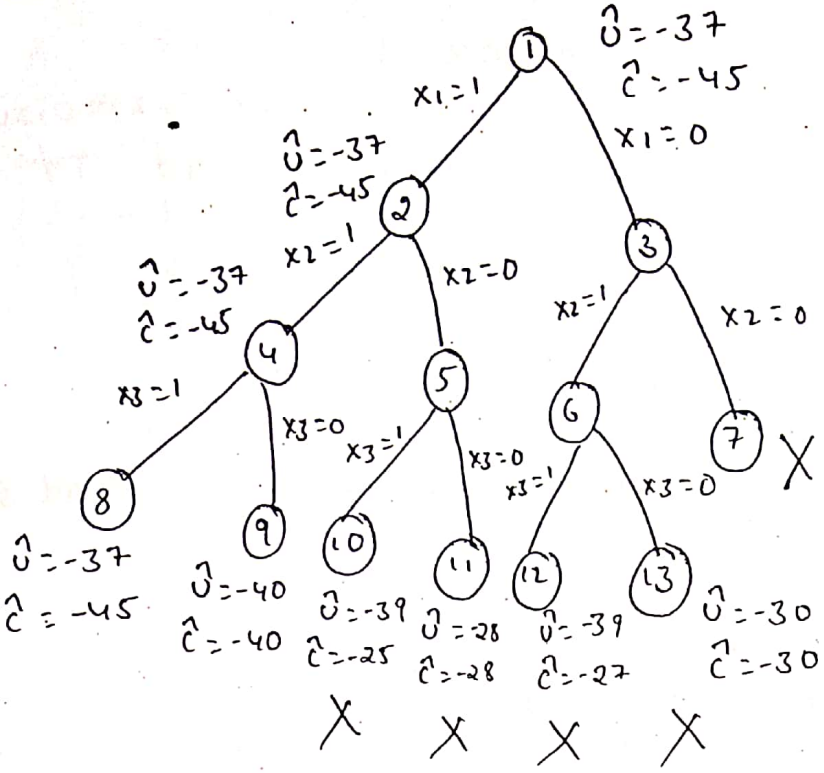
∴ If we check UB values in node (9) UB value is -40 i.e., is minimum. so we can change Global UB is -40

→ we can check cost > Global UB

node (8)	-45 > -40	false
node (9)	-40 ≥ -40	false
node (10)	-39 > -40	true
node (11)	-28 > -40	true
node (12)	-27 > -40	true
node (13)	-30 > -40	true

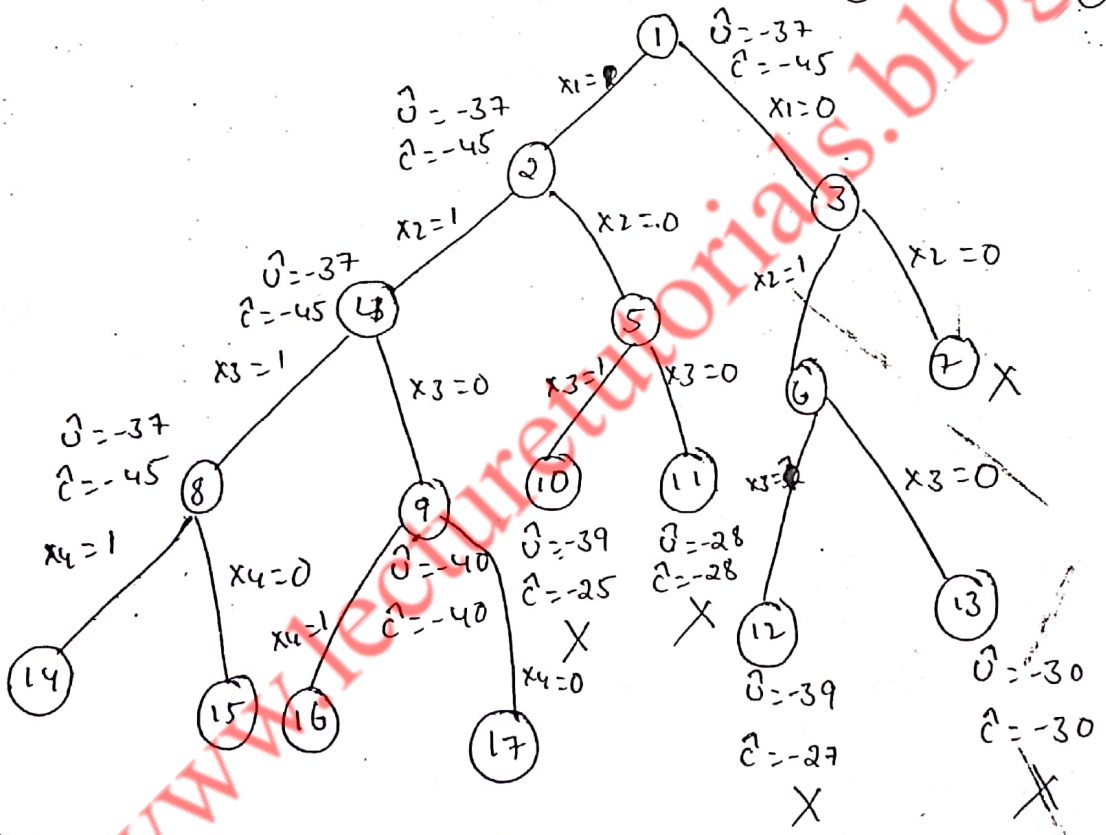
→ So, we can kill the nodes 10, 11, 12, 13

→ Now we can extending the two nodes are 8 and 9



Global UB = -40

Steps:- We can extended the node (8) and node (9)



Global UB = -40

→ we can include x_4 that exceed the bag capacity so. that can be ignored

$(x_1=1, x_2=1, x_3=1, x_4=0)$

10	12	15	18
2	3	6	9

→ node 15, $\hat{c} = 10+12+15$

$= -37$

$\hat{u} = 10+12+15$

$= -37$

node (16) same as above we can replace it

$(x_1=1, x_2=1, x_3=0, x_4$

10	12	15	18
2	3	6	9

node (17) $\hat{v} = 10+12$

$= -22$

$\hat{c} = 10+12$

$= -22$

→ All upper bounds are greater than Global upper bound so we can't replace it

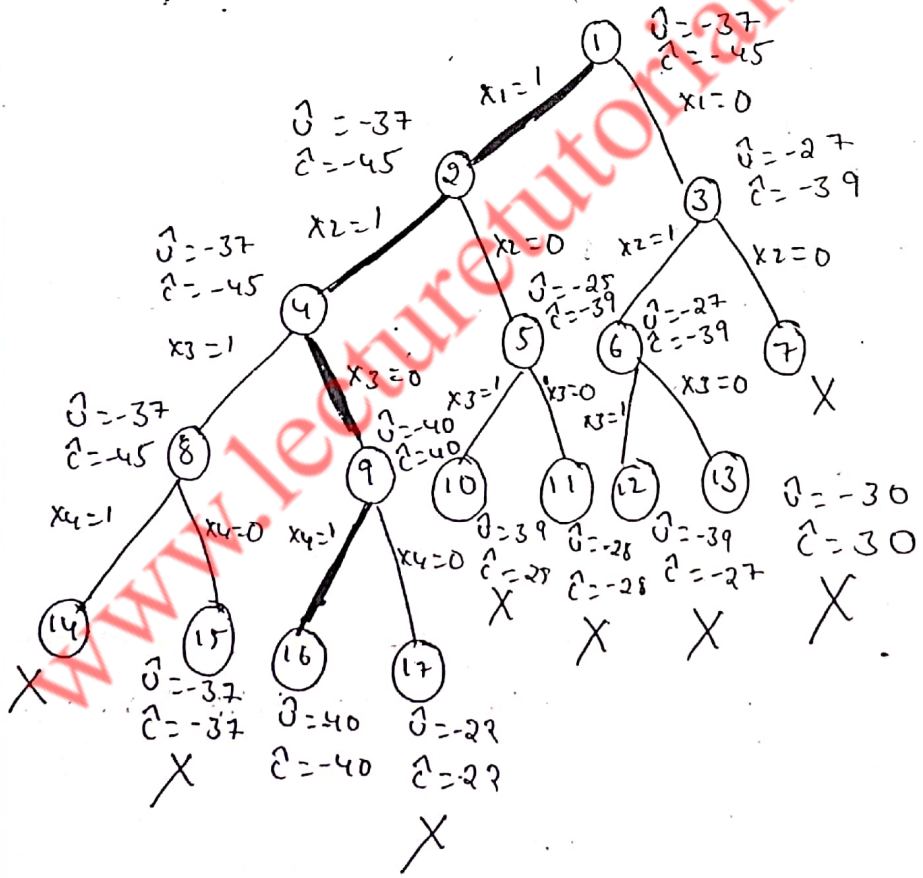
→ Now we check cost > Global UB

node (15) $-37 > -40$ true

node (16) $-40 > -40$ false

node (17) $-22 > -40$ true

So, we can kill the nodes 15, and 17



Global UB = -40

→ We can find out the feasable solution is

1 – 2 – 4 – 9 – 16

$x_1 = 1$

$x_2 = 1$

$x_3 = 0$

$x_4 = 1$

$$\sum_{i=1}^{n} w_i x_i = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$

$$= 9 \times 1 + 3 \times 1 + 6 \times 0 + 9 \times 1$$

$$= 2 + 3 + 0 + 9$$

$$= 14$$

$$\sum_{i=1}^{n} P_i x_i = P_1 x_1 + P_2 x_2 + P_3 x_3 + P_4 x_4$$

$$= 10 \times 1 + 12 \times 1 + 15 \times 0 + 18 \times 1$$

$$= 10 + 12 + 0 + 18$$

$$= 40 //$$

∴ Maximum Profit is 40 //

15-Puzzle problem

* This 15-puzzle problem can follows state Space tree algorithm.
* The state Space of an initial state consists of all stat that can be reached from initial state.
* Given 4×4 board with 15 numbered tiles and 1 empty Space or (empty spot).
* By using this empty Space we can arrange numbered value tiles in Sequence order
* We can Slide 4 adjacent (left, right, above & below) tiles into the empty space.
* It is easy to see that there are 16! ($16! \approx 20.9 \times 10^{12}$) different arrangements of the tiles.
* Each node x in the Search tree is associated with a cost. It is useful to determine the next node which contains least cost.
* In 15-puzzle Algorithm can follows the below estimation cost.

$$\text{Estimated cost } \hat{c}(x) = f(x) + \hat{g}(x)$$

where

$f(x)$ = length of the path from root node to x

$\hat{g}(x)$ = No. of non blank tiles not in their goal position

for example :-

We are considering 4×4 board with 15 numbered tiles and one empty space.

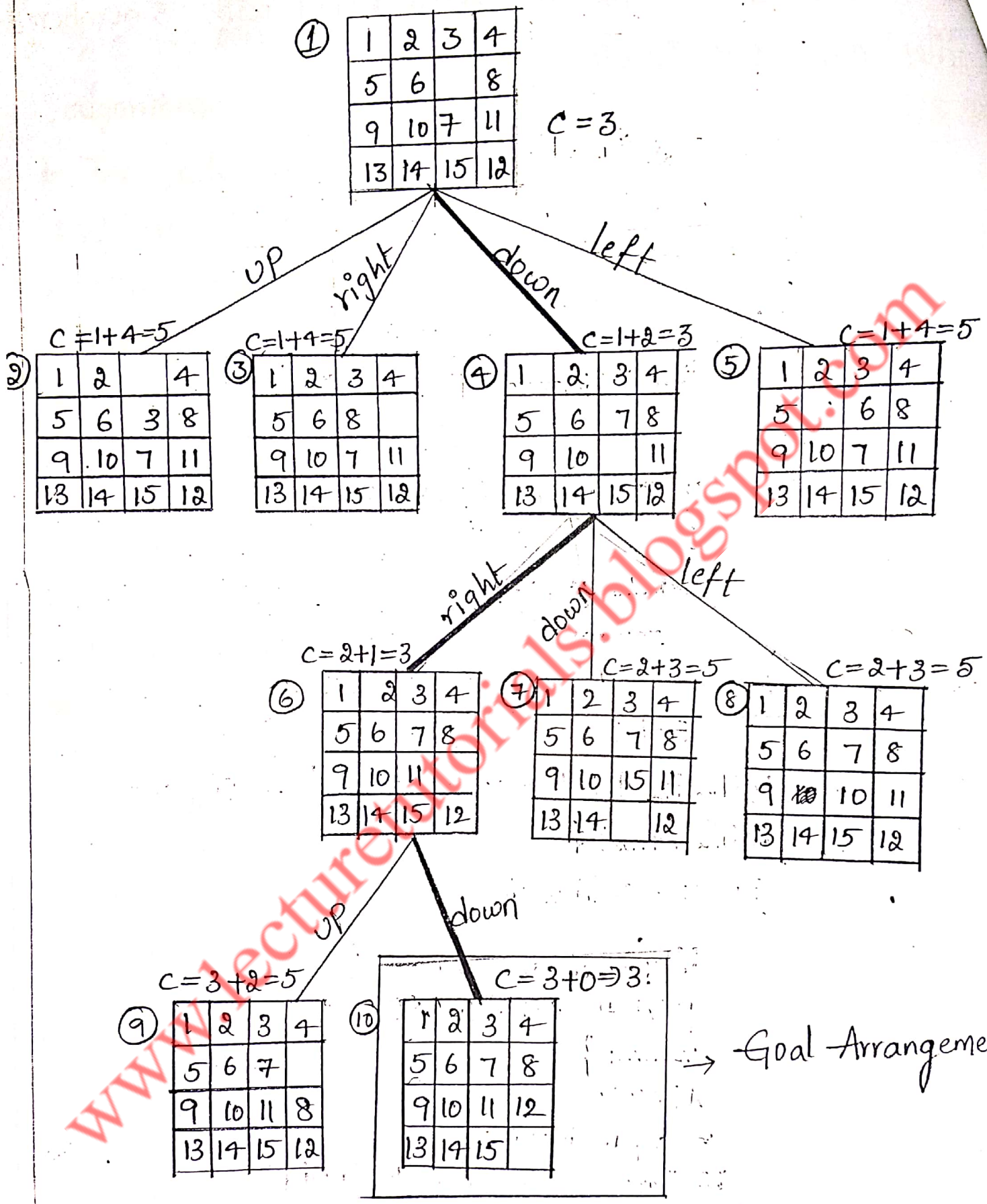
1	2	3	4
5	6		8
9	10	7	11
13	14	15	12

Initial Arrangement

* By using the above initial arrangement we can find goal arrangement by using 15-puzzle problem.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Goal Arrangement



Space tree path = 1 – 4 – 6 – 10

Estimation Cost = 3 + 0 ⇒ 3

Algorithm :-

```
Struct list_node
{
 list_node *next;
 list_node *parent;
 float cost;
};
 algorithm LCsearch (list_node *t)
 {
      if (*t is an answer node)
      {
        print (*t);
        return;
 }
      E=t;
 Initialize the list of live nodes to be empty;
 while (true) {
 for each child x of E.
 {
      if x is an answer node {
      Print the path from x to t;
      return; }
     Add(x);
     x -> parent = E; }
    if there are no more live nodes
   {
    Print (".No answer node");
   } return;
   E = Least(); }
```