Wednesday

* UNIT - 05 *

* Concepts:-

1. Back tracking
2. Branch and Bound
3. P and NP

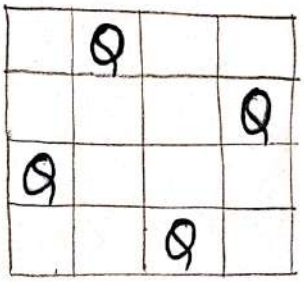
* Back tracking:-

↓

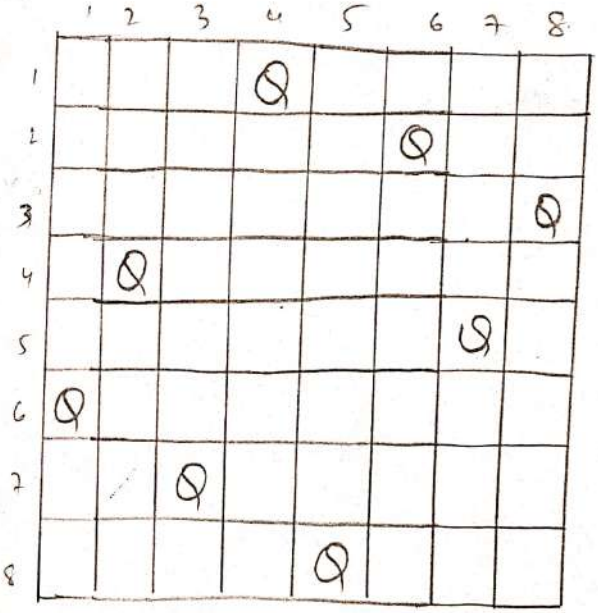
3 problems → n-Queens problem
→ Sum of subsets problem
→ Graph

* Examples-

4x4

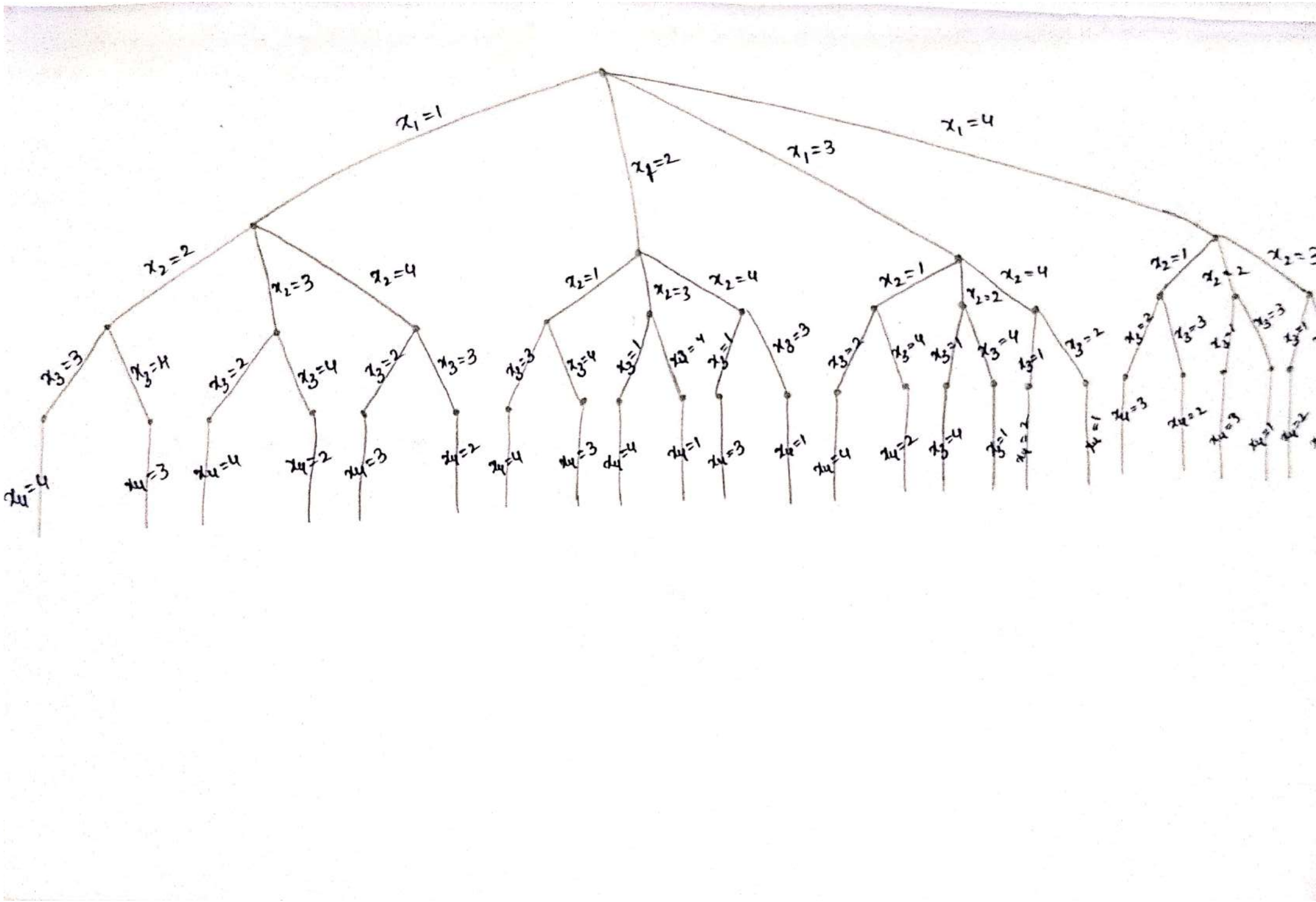


8x8

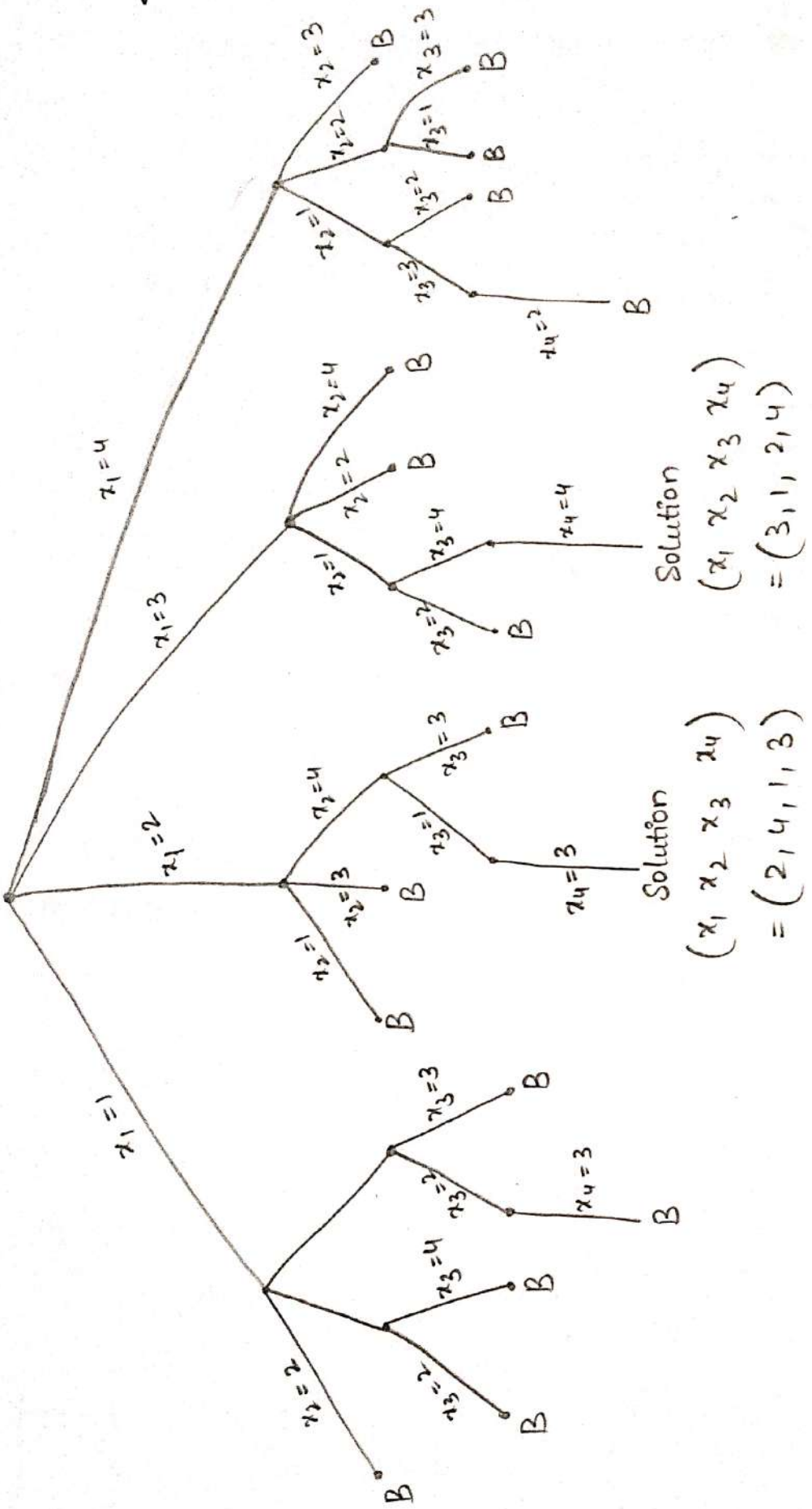


* Solution to 4-Queen's problem:-

Draw the state space tree for 4 Queens problem

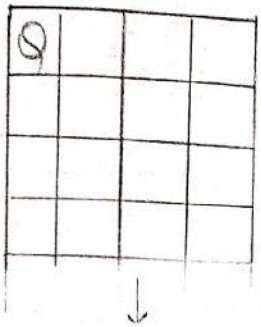


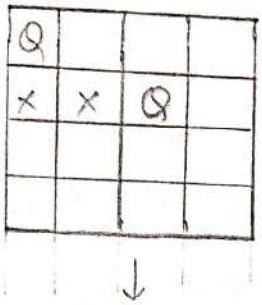
Q2: Draw solution space tree for 4-Queen's problem by using backtracking.

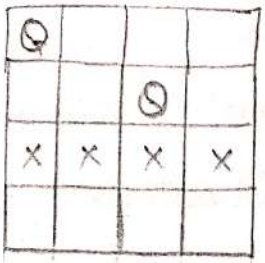


Q: Explain 4 - Queen's problem?

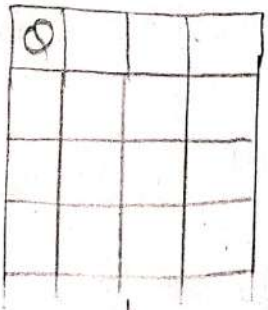
In 4-Queen's problem, we need to place 4 Queen's in 4x4 chessboard in such a way that no two queen's attack.

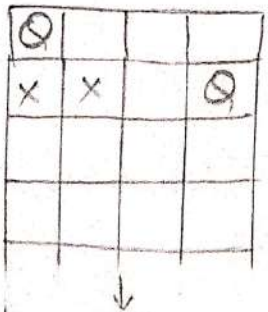


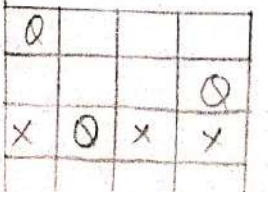


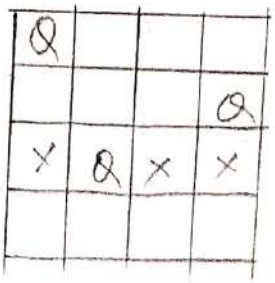


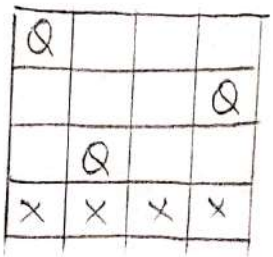
Backtrack to Q_1



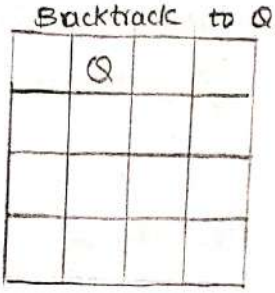


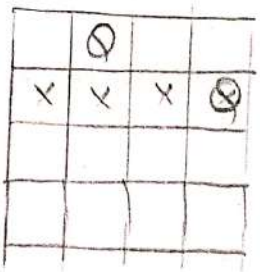


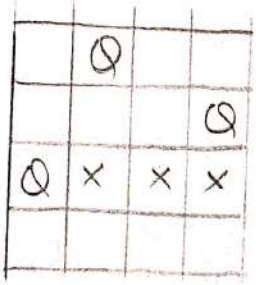


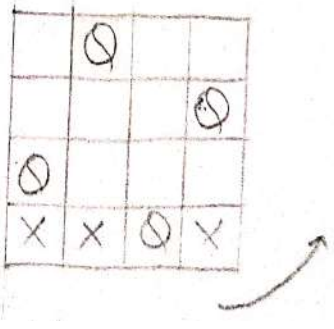


Backtrack to Q_2, Q_1

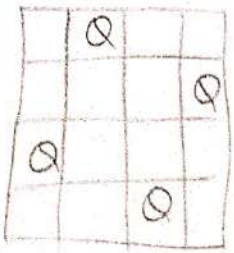






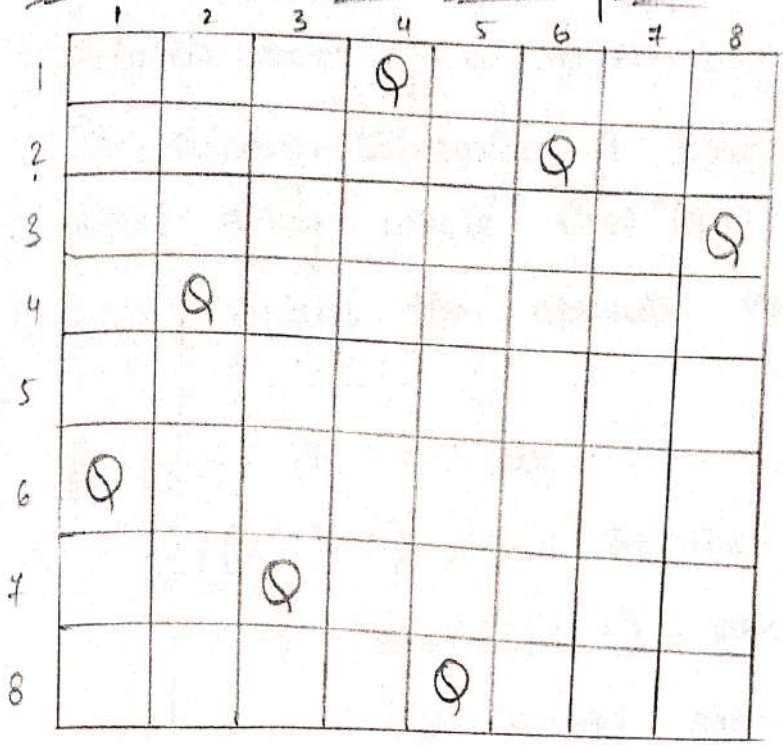


Solution to 4-Queen's



x_1, x_2, x_3, x_4

* Solution to 8-Queens problem:-



$(x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8) = (4, 6, 8, 2, 7, 1, 3, 5)$

* write the algorithm for n-queen's problem.

In main function we call n-queens of (1, n)

```
Algorithm NQueens(k, n)
// Using backtracking, this procedure prints all
// possible placements of n queens on an n x n
// chessboard so that they are non-attacking.
{
   for i:=1 to n do
   {
      if Place(k,i) then
      {
         x[k] := i;
         if (k=n) then write (x[1:n]);
            else NQueens (k+1, n);
      }
   }
}
```

→ Algorithm Place (k, i)

// Returns true if a queen can be placed in kth row and
// ith column. Otherwise it returns false. x[] is a
// global array whose first (k-1) values have been set.
// Abs(r) returns the absolute value of r.

```
{
    for j := 1 to k-1 do
        if ((x[j] = i)  // Two in the same column
            or (Abs(x[j] - i) = Abs(j - k)))
                // or in the same diagonal
            then return false;
    return true;
}
```

Q: Explain the general method for Backtracking?

* Let $T(x_1, x_2, \ldots x_i)$ be the set of all possible values for x_{i+1} such that $(x_1, x_2, \ldots x_{i+1})$ is also a path to a problem state.

* If $B_{i+1}(x_1, x_2, \ldots x_{i+1})$ is false then the path cannot be extended to answer node.

→ Algorithm Backtrack(k)

// This schema describes the backtracking process using
// recursion. On entering, the first k-1 values
// x[1], x[2], ..., x[k-1] of the solution vector
// x[1:n] have been assigned. x[] and n are global.

```
{
    for (each x[k] ∈ T(x[1], ...., x[k-1]) do
    {
        if (B_k(x[1], x[2], ...., x[k]) ≠ 0) then
        {
```

if $(x[1], x[2], \ldots, x[k]$ is a path to
node)

then write $(x[1:k])$;

if $(k<n)$ then Backtrack $(k+1)$;

}

}

}

* write iterative Backtracking method.

Algorithm IBacktracking (n)

// This schema describes the backtracking process.

// All solutions are generated in $x[1:n]$ and printed

// as soon as they are determined.

{

$k := 1$;

while $(k \neq 0)$ do

{ if (there remains an untried $x[k] \in T(x[1], x[2], \ldots,$
$x[k-1])$ and $B_k(x[1], \ldots x[k])$ is true) then

{ if $(x[1] \ldots x[k]$ is a path to an answer node)
then write $(x[1:k])$;

$k := k+1$; // consider the next set.

}

else $k := k-1$; // Backtrack to the previous set.

}

}

10/10/2025 *What is sum of subsets problem?

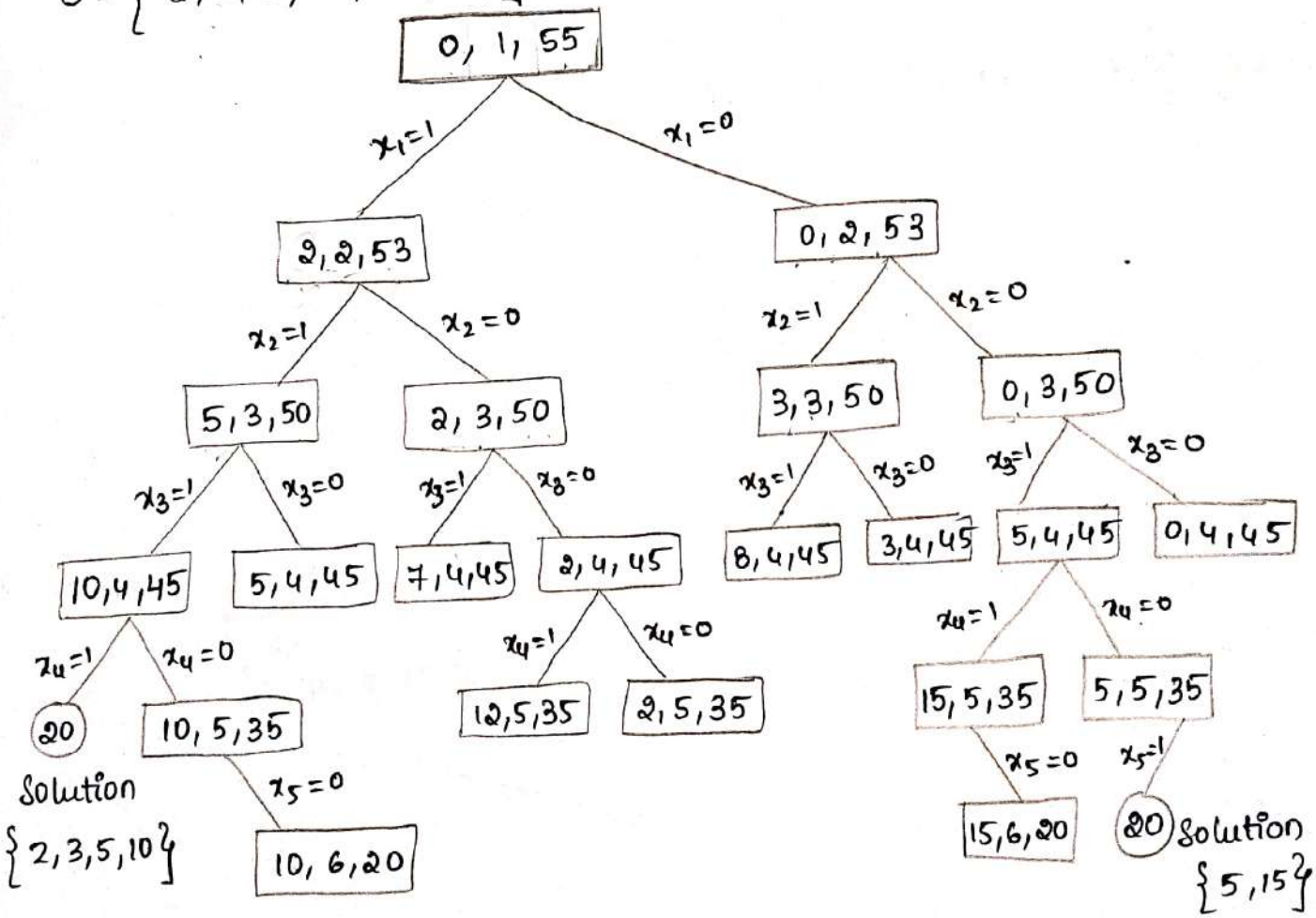
Friday Suppose we are given n distinct positive numbers (usually called weights) and we desire to find all combinations of these numbers whose sums are m. This is called sum of subsets problem.

Example:- $S = \{2, 3, 5, 10, 15, 20\}$ $m = 20$

Subsets : $\{2,3,5,10\}$ $\{5,15\}$ $\{20\}$ $\{2,3,15\}$

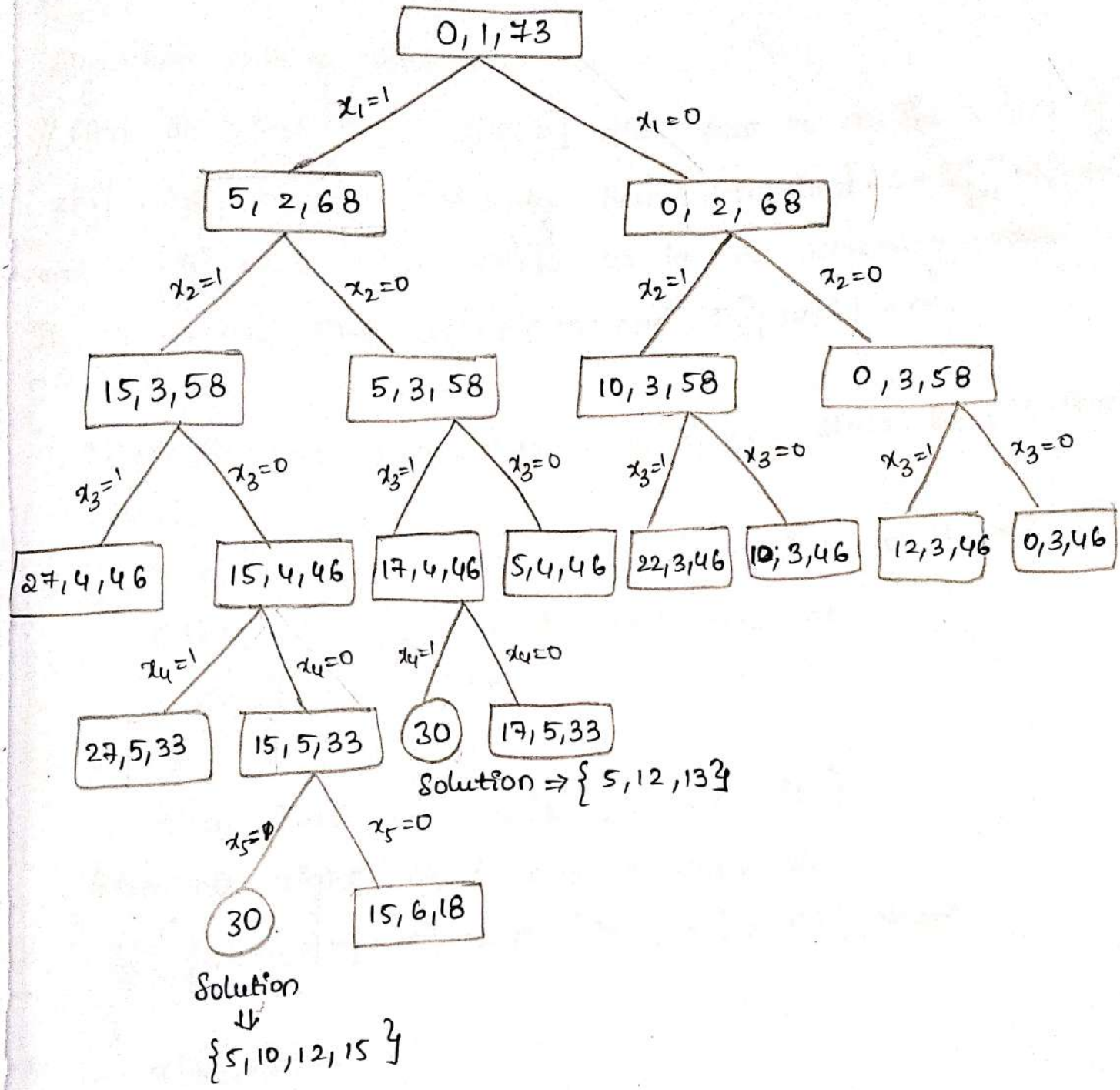
* Solve the following sum of subsets problem using backtracking

$S = \{2, 3, 5, 10, 15, 20\}$ $m = 20$



∴ Solution subsets are $\{2,3,5,10\}$ $\{5,15\}$ $\{2,3,15\}$ $\{20\}$

* Example 2: $S = \{5, 10, 12, 13, 15, 18\}$ $m = 30$



14-10-20
Tuesday

* Algorithm for sum of subsets:-

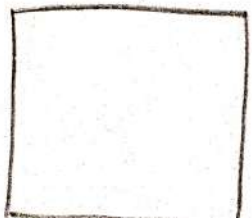
```
Algorithm SumOfSub(s, k, r)
// Find all subsets of w[1:n] that sum to m. The values of
x[j], 1 ≤ j < k, have already been determined. s = Σ_{j=1}^{k-1} w[j]*x[
and r = Σ_{j=k}^{n} w[j]. The w[j]'s are in non decreasing order.
It is assumed that w[1] ≤ m and Σ_{i=1}^{n} w[i] ≥ m.
{
  // Generate left child. Note: s+w[k] ≤ m since B_{k-1} is true
  x[k]:=1;
  if (s+w[k]=m) then write (x[1:k]); // Subset found
     // There is no recursive call here as
        w[j]>0, 1 ≤ j ≤ n.
  else if (s+w[k]+w[k+1] ≤ m)
     then SumOfSub(s+w[k], k+1, r-w[k]);
  // Generate right child and evaluate B_k.
  if ((s+r-w[k] ≥ m) and (s+w[k+1] ≤ m)) then
  {
     x[k]:=0;
     SumOfSub(s, k+1, r-w[k]);
  }
}
```

*what is Graph coloring ? Give an example.

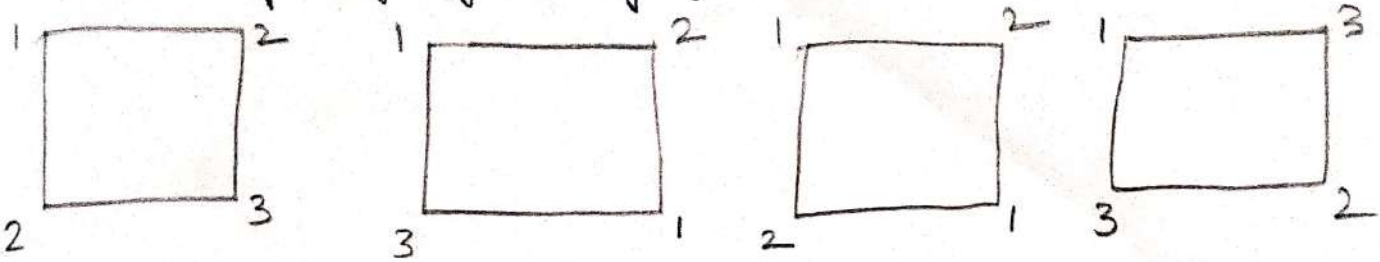
In Graph coloring problem, we need to colour the vertices of a graph in such a way that no two adjacent vertices have same colour.

Consider the graph,

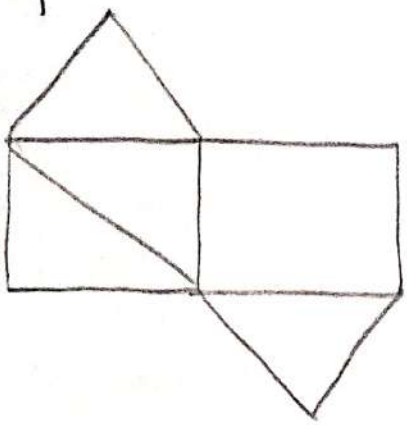
Graph

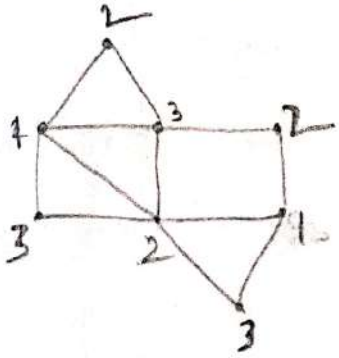


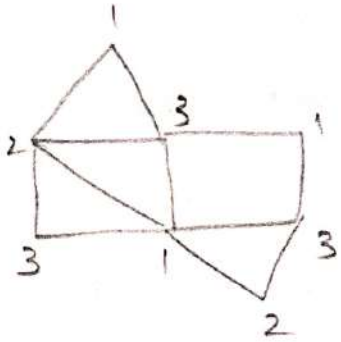
3 -colorings of given graph.



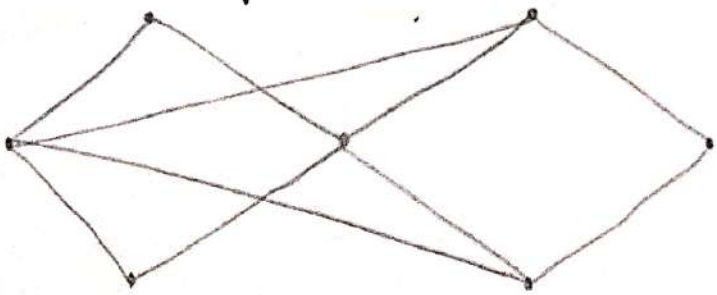
* Draw atleast two possibilities of 3 colorings of the following graph.

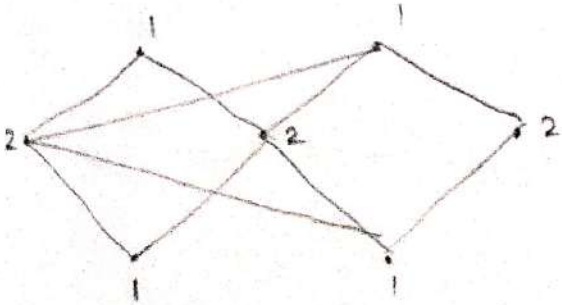


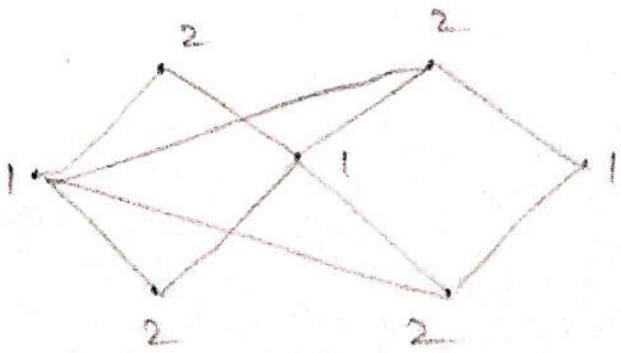




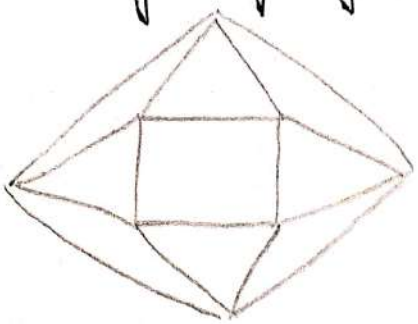
* Draw two colorings of the following graph.

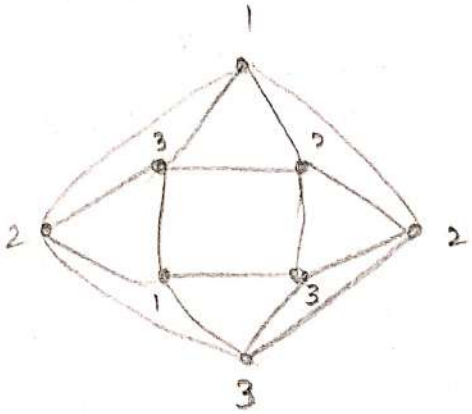






* Draw 3 colorings of following graph.





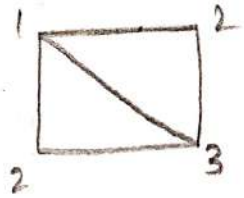
3 -coloring is not possible.

Chromatic number

* Chromatic Number:-

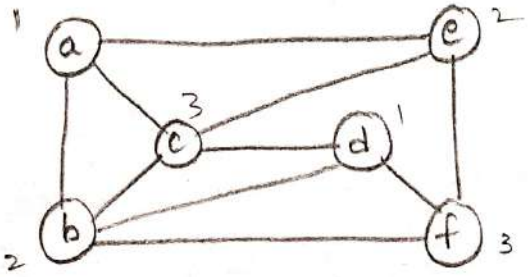
The minimum no. of colours needed to colour the vertices of a graph in such a way that no two adjacent points have same colour is called chromatic number.

Eg:-



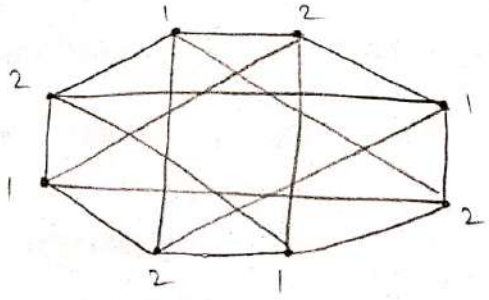
Chromatic number = 3.

36. The chromatic no of the following graph —



CN = 3.

38.



CN = 2

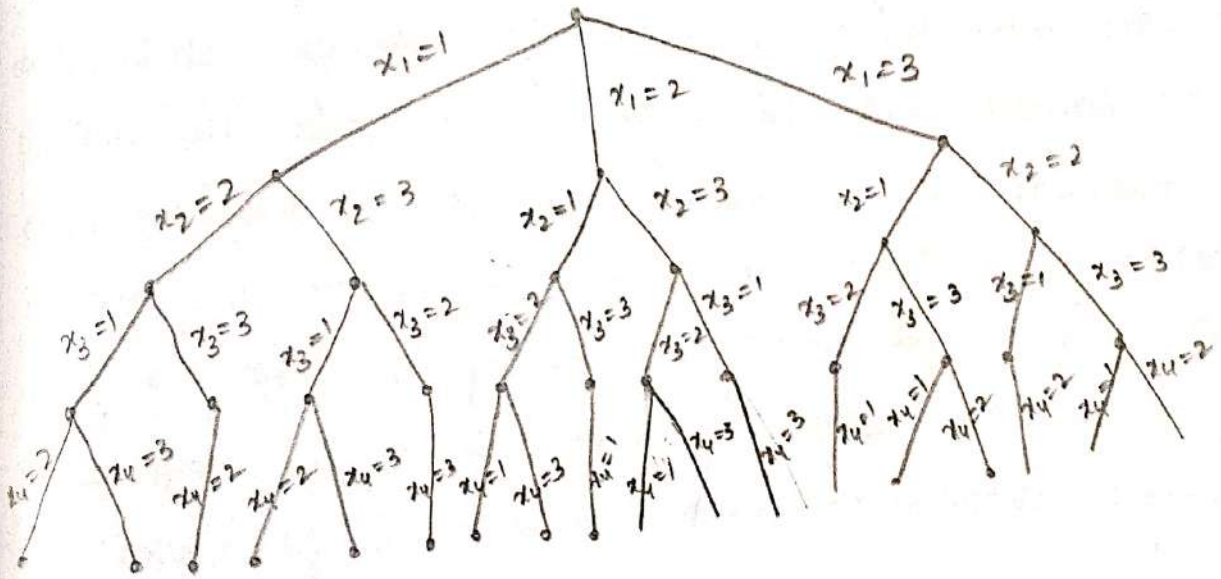
15/10/2025
Wednesday

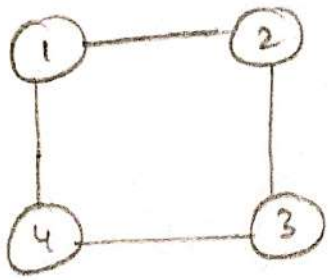
Algorithm mColoring (k)
// This algorithm was formed using the recursive backtracking scheme. The graph is represented by its boolean adjacency matrix G[1:n, 1:n]. All assigments of 1, 2, ..., m to the vertices of the graph such that adjacent vertices are assigned distinct integers are printed. k is the index of the next vertex to color.
{
    repeat {
        // Generate all legal assignments for x[k].
        NextValue (k); // Assign to x[k] a legal color.
        if (x[k] = 0) then return; // No new color possible.
        if (k = n) then // Almost m colors have been used to
                        color the n vertices
            write (x[1:n]);
        else
            mColoring (k+1);
    } until (false);
}

Algorithm NextValue (k)
// x[1], ... x[k-1] have been assigned integer values in the range [1, m] such that adjacent vertices have distinct integers. A value for x[k] is determined in the range [0, m]. x[k] is assigned the next highest numbered color while maintaining distinctness from the adjacent vertices of vertex k. If no such color exists, then x[k] is 0.
{
    repeat {
        x[k] := (x[k]+1) mod (m+1); // next highest color.

```
if (x[k]=0) then return; // All colors have been used.
   for j:=1 to n do {
      // check if this is distinct from adjacent colors.
      if ((G[k,j]≠0) and (x[k]=x[j]))
         // If (k,j) is and edge and if adj vertices
         // have the same color.
            then break;
   }
   if (j=n+1) then return; // new color found.
} until (false); // Otherwise try to find another color.
}
```

Q: Draw state space for m coloring when m=3 for the graph.





x_1 x_2 x_3 x_4

$= (1, 2, 3, 2)$

* what is Branch and Bound Technique?

The term 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 can become the E-node

In branch and bound methods, bounding functions are used to help avoid the generation of subtrees that do not contain an answer node.

* Branch and Bound technique needs two values

1) A bound value of the objective function for every node of state space tree.

2) The value of the best solution so far is compared with a node's bound and if the node's bound value is not better than the value of best solution set so far then that node is terminated.

* what is reduced matrix?

A row of a matrix is said to be reduced if it contains atleast one zero and all remaining entries are non-negative.

A matrix is reduced if every row and column is reduced.

Eg: Consider the following matrix subtract 10, 2, 2, 3, 4 from rows 1, 2, 3, 4, 5 respectively. Then we get a matrix

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

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

Subtract 1,3 from columns 1,3 respectively (columns 2,4,5 are already reduced). Then we get a reduced matrix M of given matrix M.

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

* Solve the following TSP by using branch and bound.

Given traveling sales Person problem matrix is

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} \infty & 7 & 3 & 12 & 8 \\ 3 & \infty & 6 & 14 & 9 \\ 5 & 8 & \infty & 6 & 18 \\ 9 & 3 & 5 & \infty & 11 \\ 18 & 14 & 9 & 8 & \infty \end{bmatrix} \end{array}$$

Subtract 3,3,5,3,8 from rows 1,2,3,4,5 respectively.

$$A = \begin{bmatrix} \infty & 4 & 0 & 9 & 5 \\ 0 & \infty & 3 & 11 & 6 \\ 0 & 3 & \infty & 1 & 13 \\ 6 & 0 & 2 & \infty & 8 \\ 10 & 6 & 1 & 0 & \infty \end{bmatrix} \begin{array}{l} -3 \\ -3 \\ -5 \\ -3 \\ -8 \end{array}$$

22

Subtract 5 from column respectively.

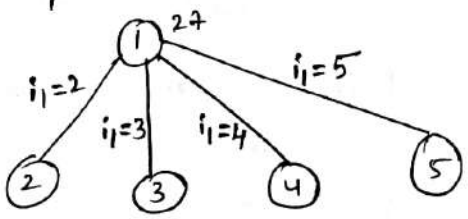
$$A_1 = \begin{bmatrix} \infty & 4 & 0 & 9 & 0 \\ 0 & \infty & 3 & 11 & 1 \\ 0 & 3 & \infty & 1 & 8 \\ 6 & 0 & 2 & \infty & 3 \\ 10 & 6 & 1 & 0 & \infty \end{bmatrix}$$

5

cost for node 1 = 22 + 5 = 27 = $\hat{c}(1)$

cost.

portion of state space tree is as shown below.



Node 2:-

Consider 1-2 path

Set 1st row, 2nd column entries to ∞.

Set $A_1[2,1]$ to ∞. Then the resultant matrix is

$$A_2 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 3 & 11 & 1 \\ 0 & \infty & \infty & 1 & 8 \\ 6 & \infty & 2 & \infty & 3 \\ 10 & \infty & 1 & 0 & \infty \end{bmatrix}$$

Subtract 1, 2 from rows 2, 4 respectively.

$$A_2 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 2 & 10 & 0 \\ 0 & \infty & \infty & 1 & 8 \\ 4 & \infty & 0 & \infty & 1 \\ 10 & \infty & 1 & 0 & \infty \end{bmatrix}$$

cost for node 2 $\hat{c}(2) = \hat{c}(1) + A_1[1,2] + 8$

$= 27 + 4 = 31$

Node 3:- Consider path 1-3

Set 1st row, 3rd column entries to ∞.

Set $A_1[3,1]$ to ∞. Then the resultant matrix is.

$$A_3 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & 1 \\ \infty & 3 & \infty & 1 & 8 \\ 6 & 0 & \infty & \infty & 3 \\ 10 & 6 & \infty & 0 & \infty \end{bmatrix}$$

Subtract 1 from row 3 respectively

$$A_3 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & 1 \\ \infty & 2 & \infty & 0 & 7 \\ 6 & 0 & \infty & \infty & 3 \\ 10 & 6 & \infty & 0 & \infty \end{bmatrix}$$

Subtract 1 from column 5 respectively.

$$A_3 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & 0 \\ \infty & 2 & \infty & 0 & 6 \\ 6 & 0 & \infty & \infty & 2 \\ 10 & 6 & \infty & 0 & \infty \end{bmatrix}$$

1

Cost for node 3 $\hat{c}(3) = \hat{c}(1) + A_1[1,3] + 2$

$= 27 + 0 + 2 = 29$.

Node 4:- Consider path 1-4

1st row, 4th column entries to ∞

$A_1[4,1]$ to ∞. Then

$$A_4 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 1 \\ 0 & 3 & \infty & \infty & 8 \\ \infty & 0 & 2 & \infty & 3 \\ & & & & \end{bmatrix}$$

from row 5 respectively.

$$A_4 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 1 \\ 0 & 3 & \infty & \infty & 8 \\ \infty & 0 & 2 & \infty & 3 \\ 9 & 5 & 0 & \infty & \infty \end{bmatrix}$$

Subtract 1 from column 5 respectively.

cost.

$$A_4 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 0 \\ 0 & 3 & \infty & \infty & 7 \\ \infty & 0 & 2 & \infty & 2 \\ 9 & 5 & 0 & \infty & \infty \end{bmatrix}$$

cost for Node 4 $\hat{c}(4) = \hat{c}(1) + A_1[1,4] + 2$

$= 27 + 9 + 2 = 38.$

Node 5:- Consider path 1-5

1st row, 5th column to ∞

$A_1[5,1]$ to ∞.

$$A_5 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & 11 & \infty \\ 0 & 3 & \infty & 1 & \infty \\ 6 & 0 & 2 & \infty & \infty \\ \infty & 6 & 1 & 0 & \infty \end{bmatrix}$$

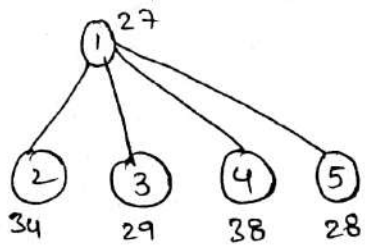
Subtract 1 from column 3 respectively.

$$A_5 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 2 & 11 & \infty \\ 0 & 3 & \infty & 1 & \infty \\ 6 & 0 & 1 & \infty & \infty \\ \infty & 6 & 0 & 0 & \infty \end{bmatrix}$$

Cost for Node 5 $\hat{c}(5) = \hat{c}(1) + A_1[1,5] + 8$

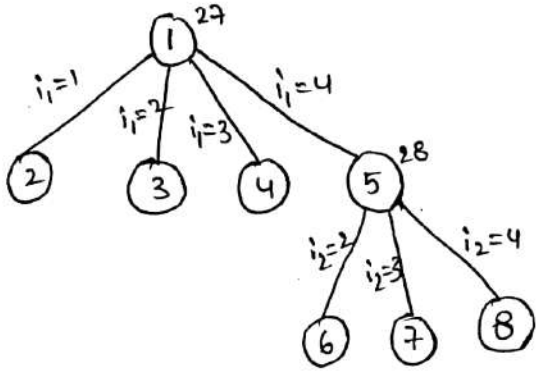
$= 27 + 0 + 1 = 28$

portion of state space tree is as shown below.



Among the nodes 2, 3, 4, 5, node (5) is having least cost.

So expand node (5)



Node 6: Consider 1-5-2 path

Set 5th row, 2nd column to ∞

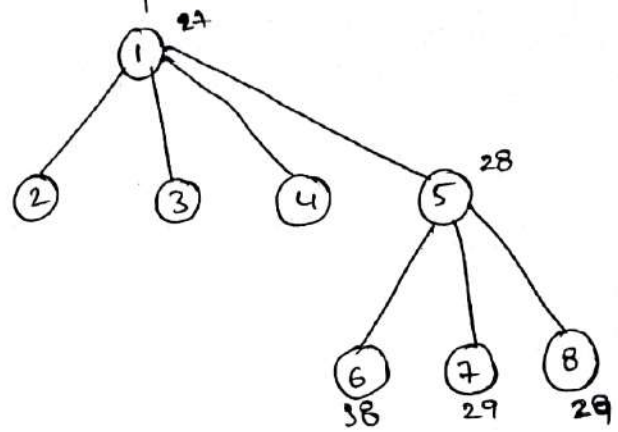
$A_5[2,1]$ to ∞

$$A_6 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 2 & 11 & \infty \\ 0 & \infty & \infty & 1 & \infty \\ 6 & \infty & 1 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

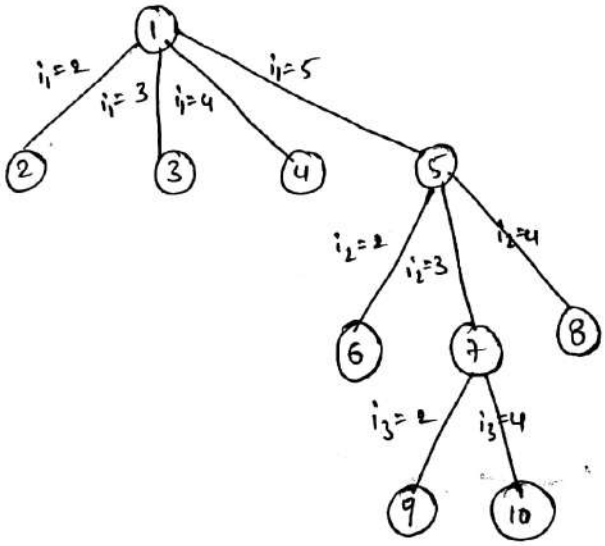
Subtract 2, 1 from rows 2, 4 respectively.

$$A_6 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 9 & \infty \\ 0 & \infty & \infty & 1 & \infty \\ 5 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} \\ -2 \\ \\ -1 \\ \\ \hline 3 \end{matrix}$$

portion of state space tree is as shown below.



Among the nodes 6, 7, 8, node (7) and (8) are having least cost. So expand node (7).



Node 9:- Consider 1-5-3-2 path

3^{rd} row, 2^{nd} column to ∞

$A_7[2,1]$ to ∞.

$$A_9 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 11 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Subtract 11 from row 2

$$A_9 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Subtract 6 from column 1

$$A_9 = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

cost for node 9 $\hat{c}(9) = \hat{c}(7) + A_7[3,2] + r$

$= 29 + 2 + 17 = 48.$

Node 10:- Consider 1-5-3-4 path.

3rd row, 4th column to ∞

$A_7[4,1]$ to ∞.

$$A_{10} = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

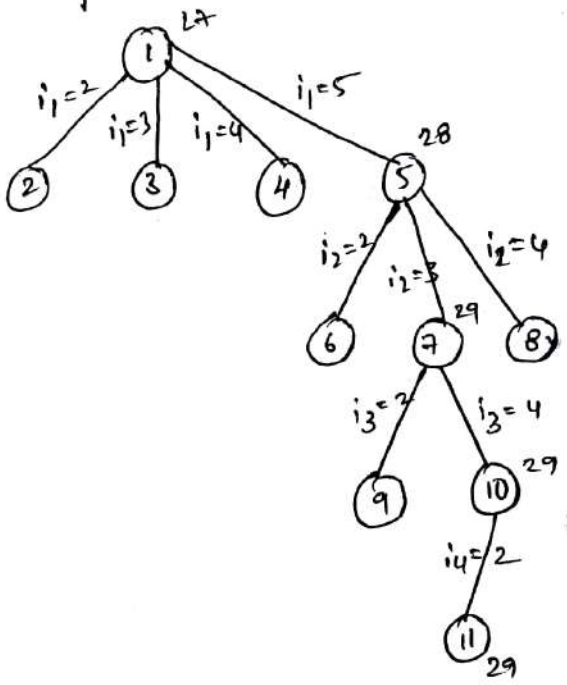
cost for node 10 $\hat{c}(10) = \hat{c}(4) + A_7[3,4] + r$

$= 29 + 0 + 0 = 29.$

Node 10, is the next line node with least cost so it becomes our E-node.

As the node (10) only child being generated is node (11) we set path as 1,5,3,4,2. The tour length for this node is $\hat{c}(11) = 29$ and upper is updated to 29. To complete the tour, we need to return back to 1. Hence optimal tour is 1-5-3-4-2-1 and cost is 29.

Portion of state space tree is as shown below.



* Explain the formation of 0/1 knapsack problem using branch and bound?

To use the branch and bound technique for 0/1 knapsack problem, we need to replace the objective function $\sum P_i x_i$ by the function $-\sum P_i x_i$ clearly $\sum P_i x_i$ is maximized if $-\sum P_i x_i$ is minimized to modified knapsack problem is stated as follows.

$$\text{minimize} \quad -\sum_{i=1}^{n} P_i x_i$$

$$\text{Subject to} \quad \sum_{i=1}^{n} w_i x_i \leq m$$

$x_i = 0$ or 1, $1 \le i \le n$

Note: 1) For every node, we calculate $\hat{c}(x)$, $U(x)$ where $\hat{c}(x)$ will be calculated by the procedure of greedy knapsack.

$E(x)$ will be calculated for 0/1 knapsack.

2) If any item doesn't fit into knapsack then skip and go to next item.

3) $U(x)$ will be used only to update upper value.

4) Every we expand the node with least $\hat{c}(x)$ value.

5) If $\hat{c}(x) >$ upper we kill that node.

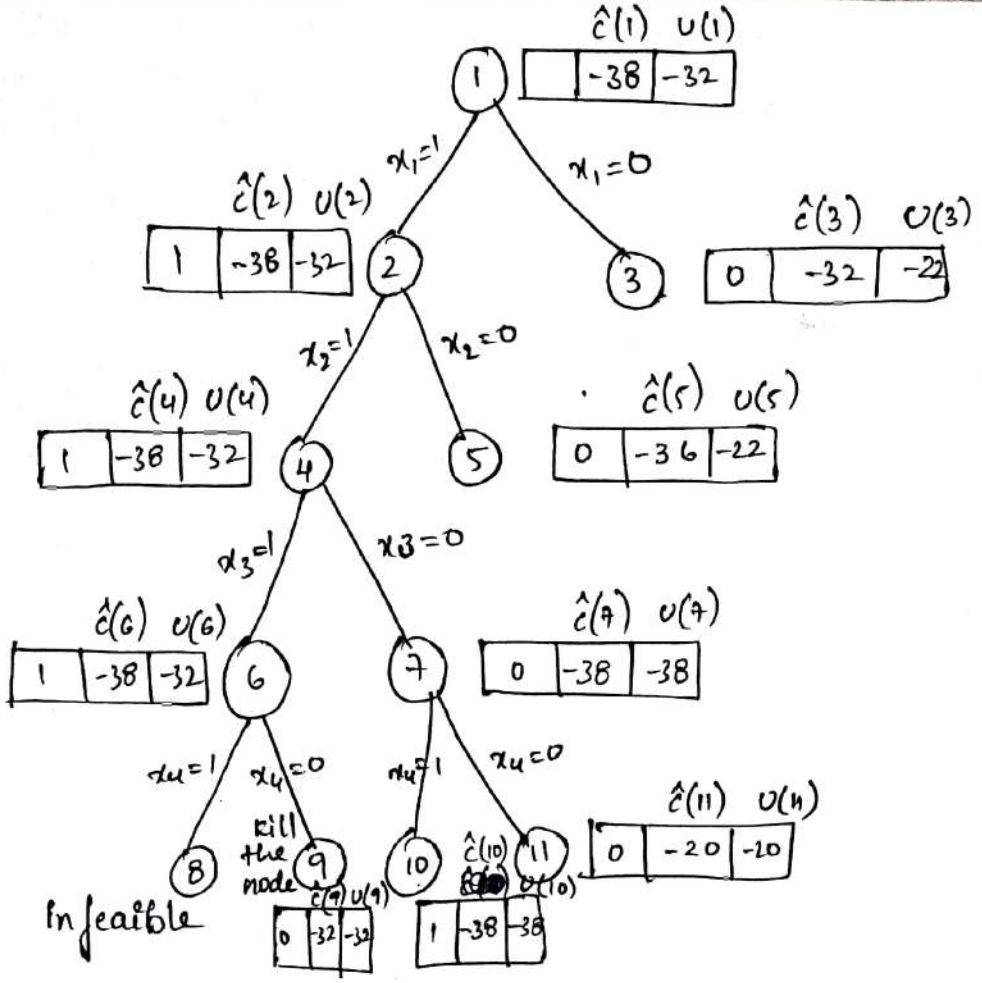
6) The leaf node for which $\hat{c}(x) =$ upper is our solution node.

* Solve the following 0/1 knapsack by using Branch and Bound.

$n = 4$, $m = 15$ $(P_1, P_2, P_3, P_4) = (10, 10, 12, 18)$ $(w_1, w_2, w_3, w_4) = (2, 4, 6, 9)$.

Given knapsack instance is $n = 4$ $m = 15$

$(P_1, P_2, P_3, P_4) = (10, 10, 12, 18)$; $(w_1, w_2, w_3, w_4) = (2, 4, 6, 9)$.



Node 1:- $\hat{c}(1) = 10 + 10 + 12 + \frac{3}{9}(18) = 38$

$\hat{c}(1) = -38$

$\upsilon(1) = 10 + 10 + 12 = 32$

$\upsilon(1) = -32$

Node 2:- $\hat{c}(2) = 10 + 10 + 12 + \frac{3}{9}(18) = 38$ $\hat{c}(2) = -38$

$\upsilon(2) = 10 + 10 + 12 = 32$

$\upsilon(2) = -32$

Node 3:- $\hat{c}(3) = 10 + 12 + \frac{5}{9}(18) = 32$ $\hat{c}(3) = -32$

$\upsilon(3) = 10 + 12 = 22$

$\upsilon(3) = -22$

Among nodes 2, 3 node ② is having least cost. So expand it

node 4:- $\hat{c}(4) = 10+10+12+\frac{3}{9}(18) = 38 \Rightarrow \boxed{\hat{c}(4) = -38}$

$u(4) = 10+10+12 = 32 \Rightarrow \boxed{u(4) = -32}$

node 5:- $\hat{c}(5) = 10+12+\frac{7}{9}(18) = 36 \Rightarrow \boxed{\hat{c}(5) = -36}$

$u(5) = 10+12 = 22 \Rightarrow \boxed{u(5) = -22}$

Among 3, 4, 5 node ④ is having least cost. So expand

node 6:- $\hat{c}(6) = 10+10+12+\frac{3}{9}(18) = 38 \Rightarrow \boxed{\hat{c}(6) = -38}$

$u(6) = 10+10+12 = 32 \Rightarrow \boxed{u(6) = -32}$

node 7:- $\hat{c}(7) = 10+10+18 = 38 \Rightarrow \boxed{\hat{c}(7) = -38}$

$u(7) = 10+10+18 = 38 \Rightarrow \boxed{u(7) = -38}$

Among nodes 3, 5, 6, 7 node 6, 7 is having least cost so expand it.

node 9:- $\hat{c}(9) = 10+10+12 = 32 \Rightarrow \boxed{\hat{c}(9) = -32}$

$u(9) = 10+10+12 = 32 \Rightarrow \boxed{u(9) = -32}$

node 10:- $\hat{c}(10) = 10+10+18 = 38 \Rightarrow \boxed{\hat{c}(10) = -38}$

$u(10) = 10+10+18 = 38 \Rightarrow \boxed{u(10) = -38}$

node 11:- $\hat{c}(11) = 10+10 = 20 \Rightarrow \boxed{\hat{c}(11) = -20}$

$u(11) = 10+10 = 20 \Rightarrow \boxed{u(11) = -20}$

This is solution node $(x_1, x_2, x_3, x_4) = (1, 1, 0, 1)$.

The tag sequence for the path ⑩ ⑨ ④ ② ① is 1 0 1 1 so $x_4 = 1$, $x_3 = 0$, $x_2 = 1$, $x_1 = 1$ and optimal profit ~~solution~~ is 38.

Q) $n = 5$, $m = 12$ $(P_1, P_2, P_3, P_4, P_5) = (12, 10, 5, 9, 3)$

$(w_1, w_2, w_3, w_4, w_5) = (3, 5, 2, 5, 3)$

Q) write control abstraction for least cost (LC) search?

Q) Explain Branch and Bound algorithm.

```
// Search t for an answer node.
Algorithm LCSearch(t){
    if (*t is an answer node) then
        output *t and return;
    E := t; // E-node
    Initialise the list of line 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);
        }
        if (there are no more live nodes) then {
            write ("No answer node");
            return;
        }
        E := least();
    } until (false);
}
```

* what is deterministic algorithms?

Algorithms in which the result of every operation is uniquely defined, are called deterministic algorithms.

Example:
```
Algorithm Sum()
{
    x := 5;
    y := 10;
    z := x+y;
    return(z);
}
```

* what is Non-deterministic algorithms?

Algorithms which contain operations whose outcomes are not uniquely defined but are limited to specified set of possibilities are called NDA.

Example:
```
Algorithm Sum()
{
    x := Choice(5, 7);
    y := Choice(10, 20);
    z := x+y;
    return(z);
}
```

* Non-Deterministic algorithms in general uses the functions Choice(S), Failure(), Success()

1) Choice(S) arbitrarily chooses one of the elements of set S.
2) Failure() signals an unsuccessful completion.
3) Success() signals a successful completion.

* Non-Deterministic Search Algorithm:

```
//a: Array of size n
//x: Element to be searched.
Algorithm Search(a, n, x)
{
```

```
        i := choice(1,n);
        if (a[i] = x)
        {
            print(i);
            Success();
        }
        else
        {
            print(0);
            Failure();
        }
    }
```

* what is decision algorithm?

Any problem for which the answer is either true or false is called a decision problem.

An algorithm that is used to solve a decision problem is termed as decision algorithm.

* what is optimization algorithm?

Any problem that involves the identification of an optimal (either minimum or maximum) value of a given cost function is known as optimization problem.

An algorithm that is used to solve a optimization problem is termed as optimization algorithm.

* Reducibility between problems:-

Let L_1 and L_2 be problems. We say that problem L_1 reduces to problem L_2 if there is a way to solve L_1 by a deter-ministic polynomial time algorithm using a deterministic algorithm that solves L_2 in polynomial time.

If problem L_1 reduces to problem L_2, then denote it by $L_1 \propto L_2$.

* P and NP classes:-

Q: write about P, NP, NP-complete and NP-hard?

Class P: Class P is the set of all decision problems soluable by deterministic algorithms in polynomial time.

Class NP: class P is the set of all decision problems solvable by non-deterministic algorithms in polynomial time.

class NP-complete: A decision problem is in NP-complete if it is in NP and every problem in NP is reducible to it in polynomial time.

A NP-complete problem is solved by a polynomial time deterministic algorithm if and only if all other NP-complete problems are solved by a polynomial time deterministic algorithm.

class NP-hard: A decision problem is in NP-hard if every problem in NP is reducible to it in polynomial time.

If a NP-hard problem is solved by a polynomial time deterministic algorithm, then all other NP-complete problems are solved by a polynomial time deterministic algorithm.

* Relationship among P, NP, NP-C, NP-H classes is: