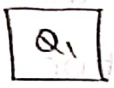
UNIT-IV Backtracking

N-Queens problem:-

→ consider an NxN chess board on which we have to place N-Queens so that, no two queens attack each other by being in the same row or in the same column or in the same diagonal.

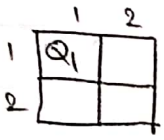
→ The following figures illustrates the N-queens problem for N = 1, 2, 3

⇒ For N = 1, we have to place 1-queen on 1x1 chess board. This problem has a trivial solution, because it contains 1-cell, there is no alternate position for Q_1.



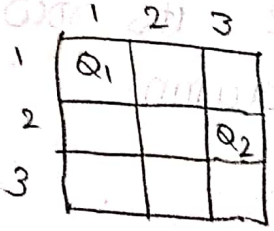
Trivial solution

⇒ For N=2, we have to place 2-queens on 2x2 chess board. Every row contains the corresponding queen and Every column contains 1-queen. There is no solution for N=2 because Q_1 can be placed on (1,1) position, Q_2 cannot be placed.



No solution

⇒ For N=3, we have to place 3-queens on 3x3 chess board. Q_1 can be placed on (1,1) position, Q_2, can be placed on (2,3) position but there is no position for Q_3. Hence, we can say that there is no solution for N=3.



Let us consider, 4-queens problem (N=4) and solve it by using the backtracking technique. We have to place 4-queens on 4x4 chess board. Each queen has to be placed in its own row. i.e; Q_1 can be placed on 1st row Q_2 can be placed on 2nd row, Q_3 can be placed on 3rd row and Q_4 can be placed on 4th row.

→ Our objective is to assign a column number for each queen on 4x4 chess board.

→ This problem has two constraints

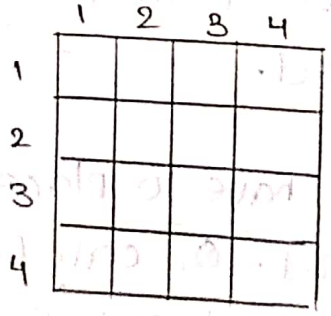
(i) Implicit constraints.

(ii) Explicit constraints.

→ The Implicit constraints for this problem is no two queens are on the same column and no two queens are on the same diagonal.

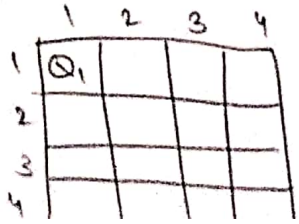
→ The Explicit constraints for this problem is: The finite set $S = \{1, 2, 3, 4\}$, the solution vector x, can take the values from the finite sets.

→ Now, we start with Empty chess board

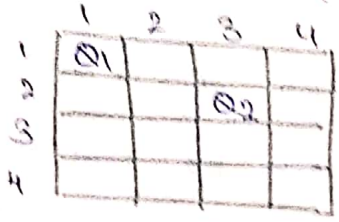


4- Queens problem.

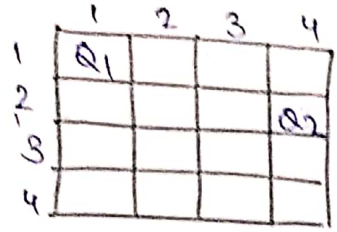
→ Place Q_1 in the first possible position of its row, i.e; on first row and first column.



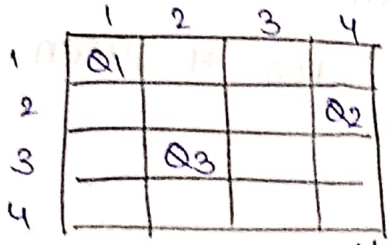
→ Place Q_2 at (2,3) position i.e, 2nd row and 3rd column, after trying unsuccessful positions (2,1) and (2,2)



→ Place Q_3 on 3rd row by trying the possible positions (3,1), (3,2), (3,3), (3,4) all are unsuccessful positions for Q_3. Hence the algorithm backtracks and check the possible position for Q_2. Q_2 has only one possible position i.e, (2,4)

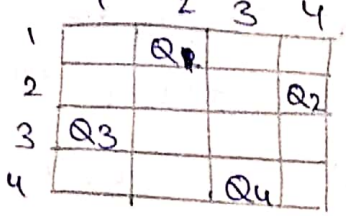


→ Now, Place Q_3 at (3,2) position.



→ Now, Place Q_4 at 4th row, there is no possible position for placing Q_4 on 4th row after trying all unsuccessful positions (4,1) (4,2) (4,3) (4,4). Hence, we backtrack Q_3, Q_2, Q_1

→ Finally, we have to place Q_1 at (1,2) position, Q_2 can be placed at (2,4) position, Q_3 at (3,1), Q_4 can be placed on (4,3) position, after backtracking Q_3, Q_2, Q_1

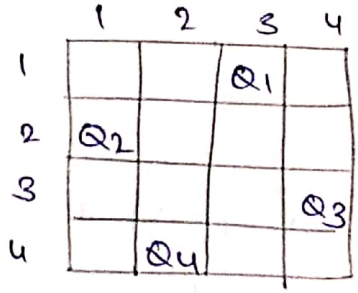


Solution vector $x = (2, 4, 1, 3)$

ie; first Q_1 can be placed on 2nd column, Q_2 can be placed on 4th column, Q_3 can be placed on 1st column, Q_4 can be placed on 3rd column.

→ The another solution for 4-Queens problem is, mirror image of the above solution

$x = (3, 1, 4, 2)$



State space Tree for 4-queens problem:-

Solution vector $x = [x_1, x_2, x_3, x_4] = [2, 4, 1, 3]$ and $[3, 1, 4, 2]$

The above stace space tree, the edges are labelled by possible values of x_i

Edges from level 1 to level 2 nodes specify the values for x_1.

The leftmost subtree contains all solutions with $x_1 = 1$. Edges from level i to level i+1 are labelled with values of x_i the solution space is defined by all paths from the root node to a leaf node.

There are $4! = 24$ leaf nodes in the above stace space tree.

* 8-queens problem:-

→ 8-queens problem is to place 8-queens on 8x8 chess board so that no two queens attack each other i.e; no two queens are on the same row, same column or same diagonal.

→ Let us number the rows and columns of chess-board 1-8. The queens can also numbered from 1-8. since, each queen must be on a different row, assume queen i (Q_i) is to be placed on row i.

→ All solutions to the Eight queens problem represented as 8-tuples $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$ where x_i is the column number on which Q_i is to be placed.

There are two types of constraints must followed by the 8-Queens problem.

(i) Explicit constraints

(ii) Implicit constraints.

→ The explicit constraints using this formulation are $S_i = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$$1 \le i \le 8$$

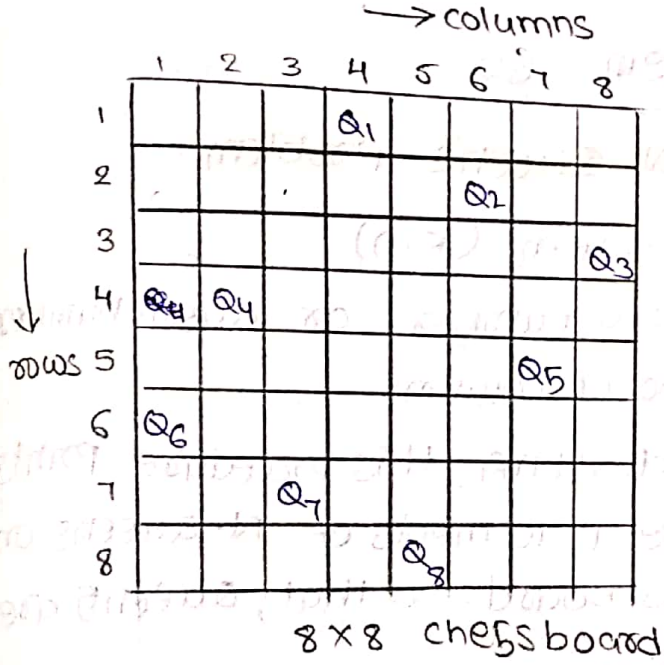
∴ The solution space contains 8^8 tuples

The Implicit constraints. For this problem are

→ no two x_i's can be on same column or on same diagonal.

→ These two constrains implies that all solutions are permutations of the 8-tuple $\{1, 2, 3, 4, 5, 6, 7, 8\}$

→ This realization reduces the size of the solution space from 8^8 tuples to 8! Tuples.



8 × 8 chessboard

Solution vector $x = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]$

$= [4, 6, 8, 2, 7, 1, 3, 5]$

Suppose two queens are placed at positions i, j and k, l these two queens lie on the same diagonal if and only if

$$|j - l| = |i - k|$$

Ex:-

Two queens Q_1 and Q_2 are lie on 1,2 and 3,3.

Both queens are not on the diagonal because

$|2-3| = |1-3|$

$1 \neq 2$

→ For an 8x8 chess board there are $64C_8$ possible ways to place 8 queens. However by allowing only placements of queens on distinct rows and columns we require atmost 8! ways.

→ The total no of nodes in the 8 queens state space tree is 69,821. For four queens problem 65

* Algorithm for N-Queen's problem

```
Algorithm N-Queens (k,n)
// k is the row number or queen number
// n is the no of columns
// using backtracking, this procedure prints
   all possible placements of N-Queens on
   NXN chess board. so that, queens are
   non-attacting
{
    for i:=1 to n do
    {
        if Place(k,i) then
        {
            x[k]:=i;
```

```
        if (k==n) then write (x[1:n]);
        else
            return N-Queens(k+1, n);
    }
}
}
algorithm Place(k, i)
// k is the row number or queen number
// i is the no. of columns.
// This procedure returns true if a queen can
   be placed in k^th row and i^th column
   otherwise it returns false.
// x[ ] is a global array whose first k-1
   values have been set.
// Abs[r] returns absolute value of r
{
    for j := 1 to k-1 do
    // two queens on the same diagonal.
    if ((x[j]=i) or (Abs(x[j]-i)=Abs(j-k))) then
        return false;
    else
        return true;
}
```

// Two queens on the same column

* Backtracking General Method :-

→ Backtracking is one of the most important algorithm design technique.

→ Many problems which deal with searching for a set of solutions or ask for an

optimal solution satisfying some constraints can be solved using the backtracking formulation

→ The name backtrack was defined by D. H. Lemmer in 1950.

→ If the solution is expressible as an n tuples $(x_1, x_2, x_3 \ldots x_n)$ where x_i are choosen from some finite set S_i, then we can apply backtracking.

→ All the solutions using backtracking must satisfy two constraints.

(i) Explicit Constraints.

(ii) Implicit constraints.

→ Explicit constraints are the rules which restrict each x_i to take on values from a given set S_i

Ex:- (i) $x_i \geq 0$ where $S_i = \{$ All non-negative real numbers $\}$

(ii) $x_i = 0$ or $x_i = 1$ where $S_i = \{0, 1\}$

→ Explicit constraints depends on the particular instance of the problem being solved. All tuples that satisfies the explicit constraints define a possible solution space for i.

→ The implicit constraints are rules that determine which of the tuples in the solution space satifies the objective function.

→ Generally backtracking solve three types of problems. They are:-

(i) Enumeration problems.

- In Enumeration problems, we find all the possible, feasible solutions.

(ii) Decision problems

- In decision problems we find whether there is any feasible solution or

not for a given problem. This problem is also called as Yes or no type problems.

(iii) optimization problems

- In optimization problems, we find whether there exist any best solution.

→ All solutions for a given problem. are represented using a tree called stace space tree. Each stace space tree contains 3 types of nodes. They are

(i) Live node
(ii) E-node
(iii) dead node

Live node:- A node which has been generated and whose children have not yet been generated is called a Live node.

E-node :- A live node whose childrens are currently being generated is called a E-node.

deadnode:- A node which is already expanded and there is no use for future.

→ Each node in the stace space tree defines a problem state, All paths from rootnode to leaf node or other nodes defines a state space.

→ In backtracking we have to use bounding functions, they will be used to kill live nodes without generating all their children.

→ The tree organization of the solution space is referred to as the stace space tree.

(or) In backtracking, while solving a

given problem, a tree is constructed based on the choices made. Such a tree with all possible solutions is called a state space tree.

→ In state space tree, there are two types of nodes. They are

(i) Promising node.

(ii) Non-promising node.

→ A node in a state space tree is said to be promising if it corresponds to a partially constructed solution that may lead to a complete solution.

→ A node in a state space tree is said to be non-promising if it cannot lead to a complete solution.

* Algorithm for general method of Backtracking:-

```
Algorithm Backtrack(k)
// This algorithm 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[], n are Global
{
    for (each x[k] ∈ T(x[1], x[2], ..., x[k-1]) do
    {
        if(Bk(x[1], x[2], ...x[k]) ≠ 0) then
        {
            if(x[1], x[2], ..., x[k] is a path to an
                                    answer node) then
```

```
        write (x[1:n]);
        if (k<n) then Backtrack(k+1);
    }
}
}
```

Graph coloring problem:-

Let G be a graph and m be a positive integer. The graph coloring problem is to find whether the nodes of graph G can be coloured in such a way that no two adjacent nodes have the same colour. Yet only m colours are used. It is called as m-colorability decision problem.

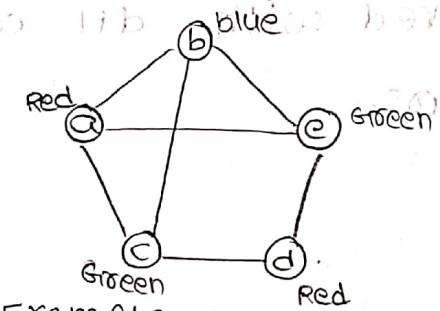
This problem has two versions.

(i) m-colorability decision problem.

(ii) m-colorability optimization problem.

The m-colorability optimization problem ask for the smallest integer m for which the graph G can be coloured.

This integer is referred to as chromatic number of the graph.



Example Graph and its colouring.

The above graph can be coloured with 3-colours red, blue, Green. Three colours are needed to colour all the vertices of the given graph G. Hence, the chromatic number of the given graph G is 3.

chromatic number:-

The minimum number of colours required to colour all the vertices of the given

graph G is called chromatic number of G.

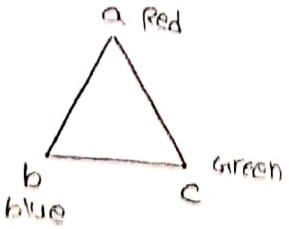
→ If 'd' is the degree of the given graph then it can be coloured with d+1 colours.

Graph G

Degree of vertex a is 2
Degree of vertex b is 2
Degree of vertex c is 2
∴ Degree of graph G is 2

For the above graph, how many number of colours required to colour all the vertices of graph G?

The degree of given graph G is 2.
It can be coloured with d+1 colours i.e;
(2+1) = 3 colours.

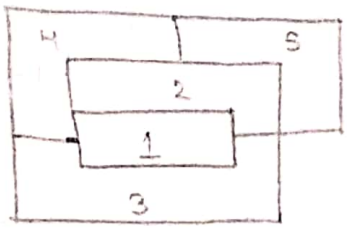


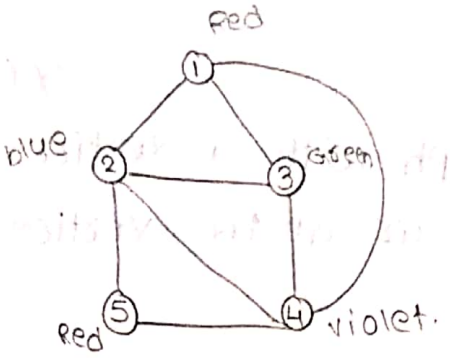
chromatic number of Graph G is 3.

Coloring of planar Graph:-

A Graph G is said to be planar graph if and only if it can be drawn in a plane in such a way that no two edges are cross. each other.

Given graph.





A map and its planar graph representation

→ The above planar graph can be coloured with 4-colours. The chromatic number of the planar graph G is 4.

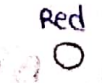
The time complexity of Graph colouring problem is $O(n \cdot m^n)$

The no. of internal nodes in the state space tree is $\sum_{i=0}^{n-1} m_i$. at each internal node $O(mn)$ time is spent by next value to determine the children corresponding to legal colourings.

NOTE :-

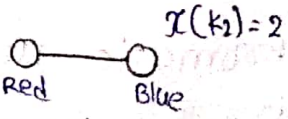
The chromatic number of complete graph can be defined as follows:-

Complete graph K_1



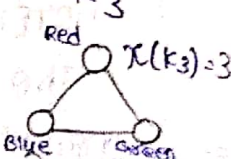
1-colourable

Complete graph K_2



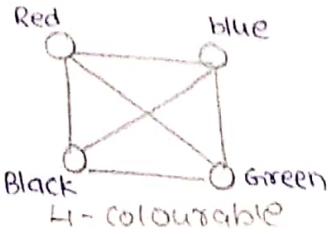
2-colourable

Complete graph K_3



3-colourable

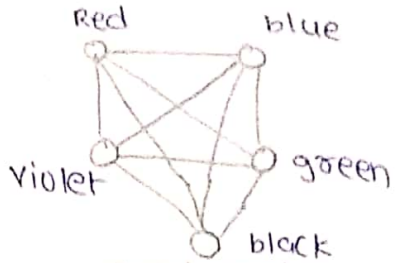
complete graph K_4



4-colourable

$\chi(K_4) = 4$

complete graph K_5



5-colourable

$\chi(K_5) = 5$

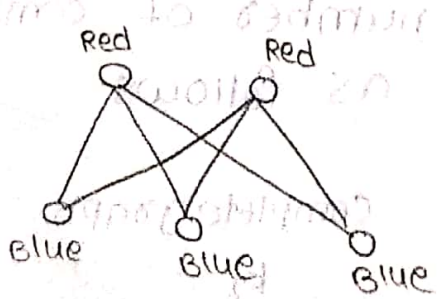
A complete graph with n-vertices (K_n) requires n-colours to colour all the vertices of the given graph.

The chromatic number of a given graph G is denoted by $\chi(G)$

chromatic number of Bipartite Graph:-

In a Bipartite graph we have to divide the given vertices of graph G into two classes. The first coordinate contains no. of vertices in class 1 or group 1. The 2nd coordinate denotes the no. of vertices in class. 2 or group-2.

→ How many no. of colours required to colour all the vertices of given Bipartite graph $K(2,3)$?



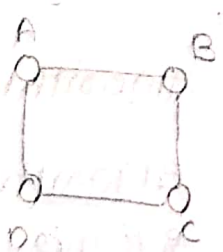
class-I

class-II

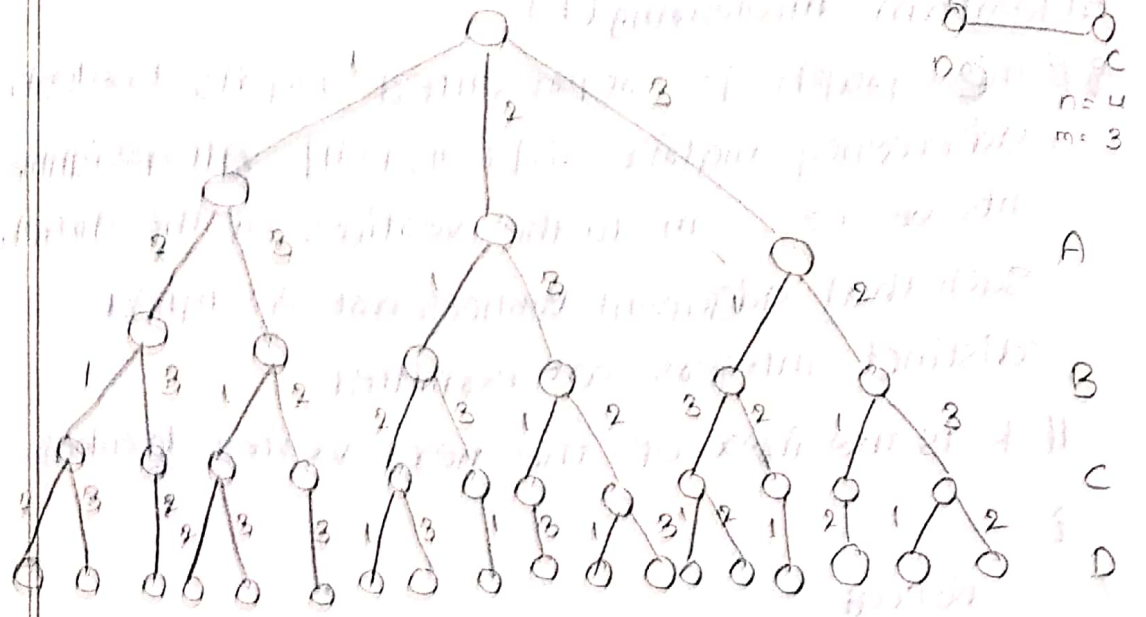
NOTE:-

The chromatic number of any Bipartite graph is 2.

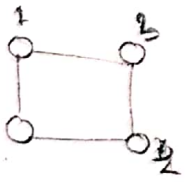
State space tree for Graph colouring problem of Graph G with 4 vertices and 3 colours.



n = 4
m = 3



Draw the state spacetree for graph colouring problem for n=4 and m = 4



* State space tree for graph colouring problem for n=4, m=3

* Algorithm for graph colouring problem:

```
Algorithm mcolouring(k)
// The graph is represented by its boolean
   adjacency matrix G[1:n, 1:n]. All assignments
   -nts of 1,2,....m to the vertices of the graph
   such that adjacent vertices are assigned
   distinct integers are permitted.
// k is the index of the next vertex to colour.
{
    Repeat
    {
        // generate all assignments for x[k]
        Next value (k); // assign to x[k] a
                           legal colour
        if (x[k] = 0) then
            return ; // No new colour possible.
        if (k == n) then
            write (x[1:n]); // Atmost m colours
                               needed to colour
                               n vertices.
        else
            mcolring(k+1);
    } until(false);
}
```

Finding all m colouring of a graph

```
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 colour while maintaining distinctness from the adjacent vertices of vertex k. If no such colour exists, then x[k] is 0

```
{
    repeat
    {
        x[k]:= x[k]+1 mod (m+1);
                    //next highest colour.
        if (x[k]=0) then return;
                    // All colours have been used
        for j:= 1 to n do
        {
            // check if this colour is distinct from
               adjacent colours.
            if (G(k,j)≠0) and (x[k]=x[j]) then
               // if (k,j) is an edge and if
                  adjacent vertices have the same
                  colour
               break;
        }
        if (j:=n+1) then return;
    } until (false); // otherwise try to find
                        another colour;
}
```

Generate the next highest numbered colour

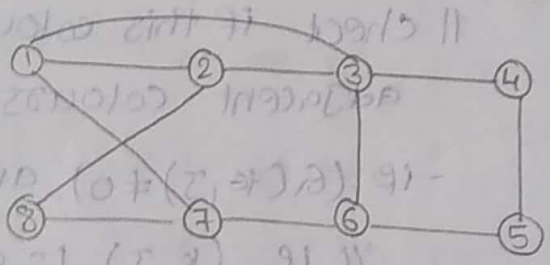
* Hamiltonian cycle/circuit problem:-

Let G(V,E) be a connected graph with 'n' vertices. A hamiltonian cycle is a round trip path along 'n' edges of G that visits every vertex exactly once and returns to the starting vertex.

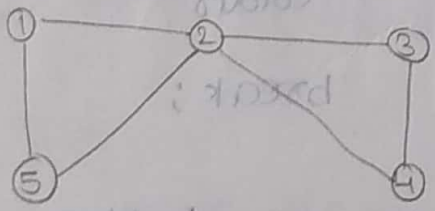
In other words, if a hamiltonian cycle begins at some vertex $v_1 \in G$ and the vertices of G are visited in the order $v_1, v_2, v_3, \ldots v_{n+1}$ then the edges (v_i, v_{i+1}) are in E, $1 \leq i \leq n$ and the v_i are distinct except for v_1 and v_{n+1} which are equal.

Let us consider the following graphs G_1 and G_2. check whether these Graphs containing the hamiltonian cycle or not.

Graph G_1:



Graph G_2:



The first Graph G_1 contains the hamiltonian cycles

1-2-8-7-6-5-4-3-1

1-3-4-5-6-7-8-2-1

By observing above two hamiltonian cycles starting vertex and ending vertex are same. The remaining vertices in that hamiltonian

cycle visits exactly once.

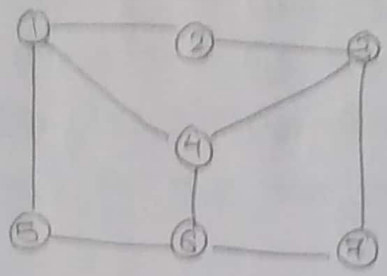
Consider the Graph G2, it doesn't contain any hamiltonian cycle

→ Hamiltonian cycle problem is a NP-Hard problem.

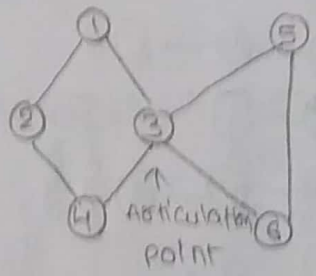
→ If any graph contains the articulation point then we can say that there is no hamiltonian cycle for that Graph.

→ If any graph contains any pendant vertex then we can say that there is no hamiltonian cycle for that Graph.

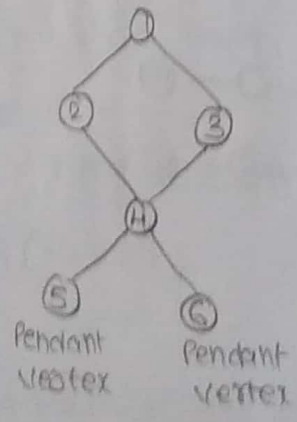
G2:



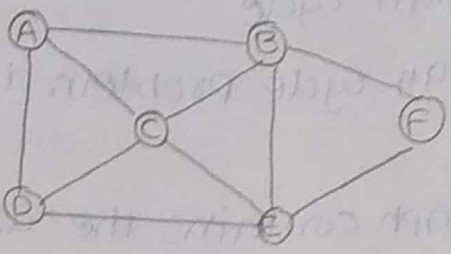
G4:

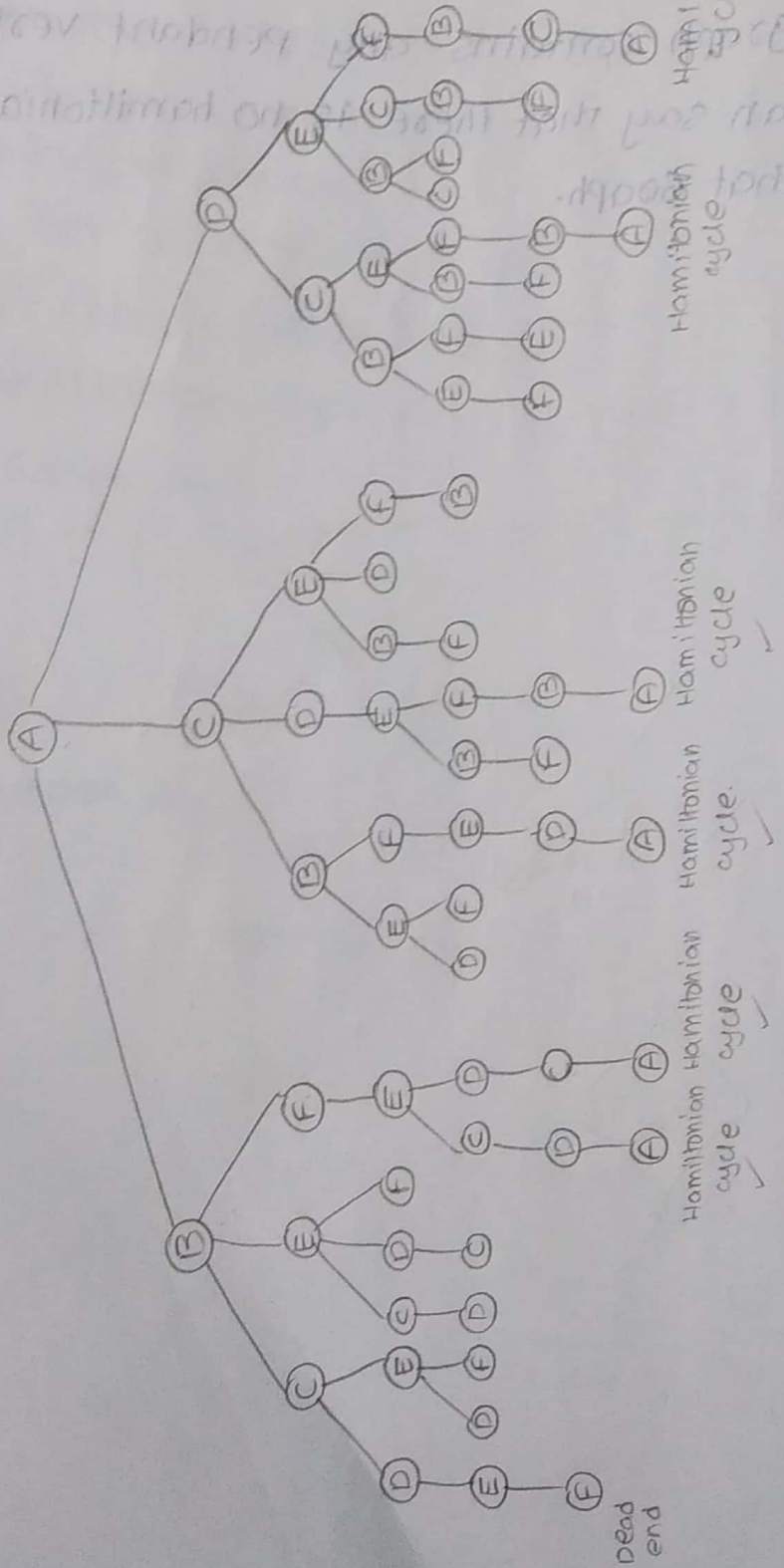


G5:



* Draw the state space tree for finding out hamiltonian cycles for the given graph





The following hamiltonian cycles in the above state space tree are:

1) A - B - F - E - C - D - A
2) A - B - F - E - D - C - A
3) A - C - B - F - E - D - A
4) A - C - D - E - F - B - A
5) A - D - C - E - F - B - A
6) A - D - E - F - B - C - A

Algorithm for Hamiltonian cycle

```
Algorithm Hamiltonian(k)
// This algorithm uses the recursive formulation of
   backtracking to find all the Hamiltonian cycles
   of a graph.
// This graph is stored as an adjacency matrix
   G[1:n, 1:n). All the Hamiltonian cycles begin at
   node 1.
{
   repeat
   {
      NextValue(k); // Generate next values to x[k]
                       and assign a next value
                       to x[k].
      if(x[k] == 0) then return;
      if(k==n) then write(x[1:n]);
      else
          Hamiltonian(k+1);
   } until(false);
}
Algorithm NextValue(k)
// x[1:k-1] is a path of k-1 distinct vertices
   if (x[k]) = 0 then no vertex has been
   assigned to x[k]
```

Generating the next vertex in Hamiltonian cycle.

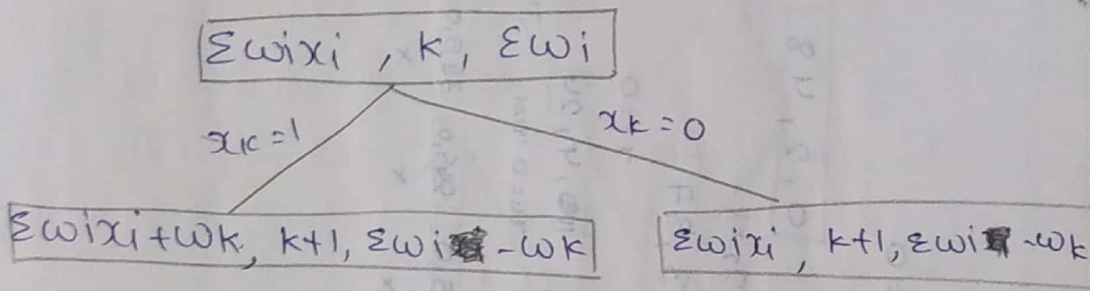
```
// After execution x[k] is assigned to the
next highest numbered vertex which doesnot
already visited in x[1:k-1] and is connected
by an edge to x[k-1] otherwise x[k]=0
// if k=n, then x[k] is connected to x[1]
{
   repeat
    {
       x[k] := (x[k]+1) mod (n+1); // Next
                                      vertex.
       if (x[k]==0) then return; // Backtrack.
       if (G(x[k-1], x[k] ≠ 0)) then // is there
       {                                an edge.
          for J:= 1 to k-1 then do
              if (x[J]==x[k]) then break;
                  // check for vertex
                                  distinctness
              if (J==k) then
              {
                 if ((k<n) or ((k=n) &&
                                 G(x[n], x[1] ≠ 0))) then
                                  // edge between first
                                     and last vertex.
                    return;
              }
       }
    } until false;
}
```

The time complexity of Hamiltonian cycles problem is $O(n^n)$

* Sum of Subsets problem:-

1) Draw the state space tree for the sum of subsets problem instance, when $w[1:4]$ $= \{7, 11, 13, 24\}$, $M = 31$ and $n = 4$

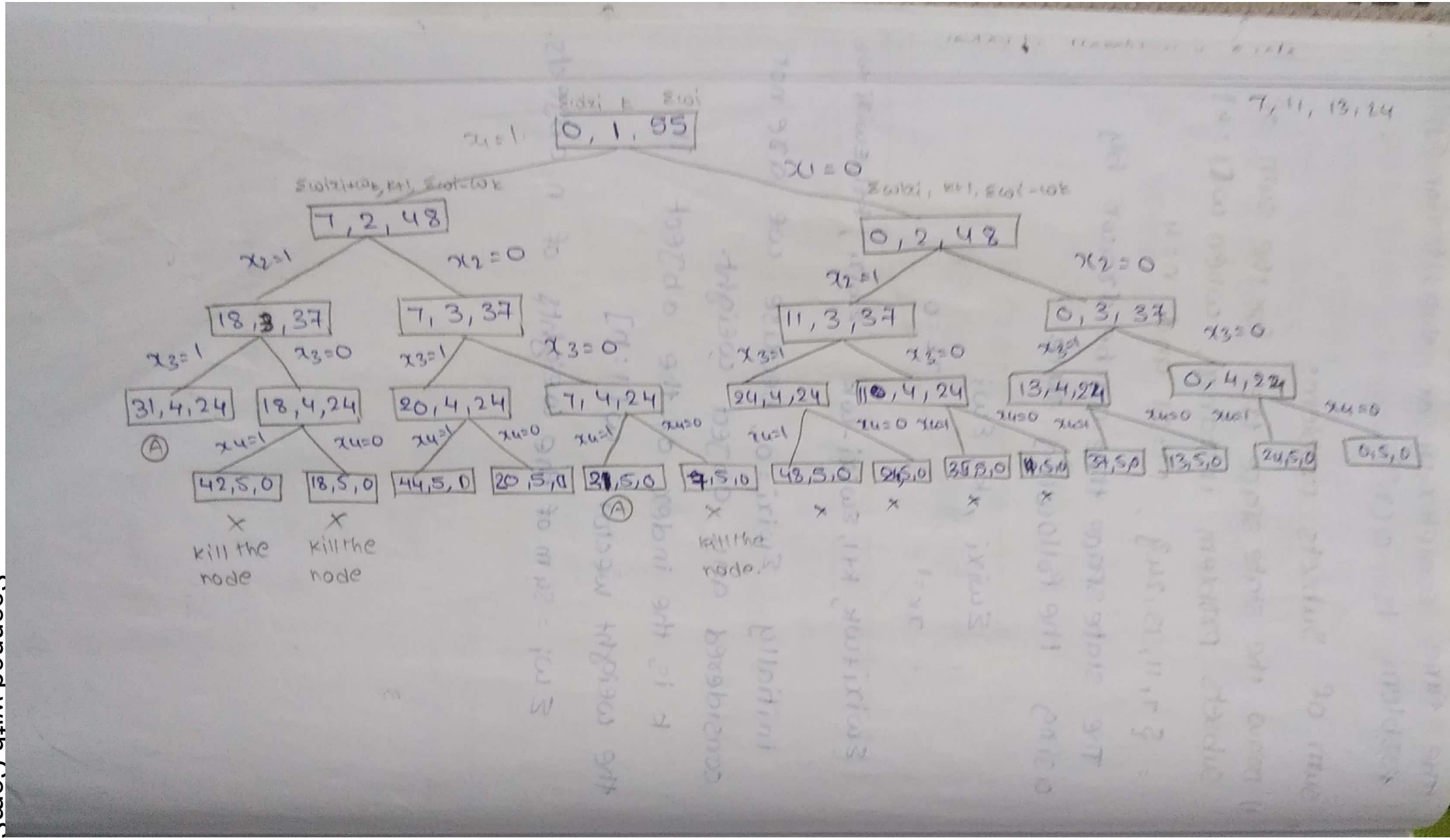
The state space tree can be drawn by using the following rule.



Initially $\Sigma w_i x_i = 0$, because we are not considered any object weight.

k is the index of the object in the weight vector - $w[1:n]$

Σw_i = Sum of the weights of n objects.



Sum of subsets problem Theory:

→ Suppose we are given n distinct positive numbers usually called the weights, and we have to find out all the subsets whose sum=M this problem is called as sum of subsets problem

→ Sum of subsets problem can be solved using either fixed or variable size tuple formulation

→ Here, we consider a backtracking solution using the fixed tuple size strategy.

→ In fixed tuple size strategy, the element x_i of the solution vector is either one or zero depending on whether the weight w_i is included or not.

The children of any node easily generated for a node at level i be the left child corresponding to $x_i = 1$ and the right child corresponding to $x_i = 0$

→ In this problem we have to use bounding functions, they are used to kill the livenodes by satisfying the following conditions. in this problem

⇒ $B_k(x_1, x_2, x_3 \ldots, x_k)$ = True iff $\sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i \geq M$ and $\sum_{i=1}^{k} w_i x_i + w_{k+1} \leq M$

Algorithm:-

Algorithm Sumofsubsets(s, k, r)

// Find out all subsets of w[1:n] whose sum = M. The values of x[J], $1 \leq J \leq k$ have already been determined.

// $s = \sum_{J=1}^{k-1} w[J] * x[J]$ and $r = \sum_{J=k}^{n} w[J]$

```
// It is assumed that w[1] ≤ M and
   Σ_{i=1}^{n} w[i] ≥ M.
{
    // Generate left child
        x[k] := 1;
        if (s+w[k] = M) then write (x[1:k]);
                            // subset found.
        else if (s+w[k]+w[k+1] ≤ M then
            sumofsubsets (s+w[k], k+1, r-w[k]);
    // Generate right child and evaluate Bk
        if ((s+r-w[k] ≥ M) and (s+w[k+1] ≤ M))
                                        then
        {
            x[k] := 0;
            sumofsubsets (s, k+1, r-w[k]);
        }
}
```

Let $w = \{5, 7, 10, 12, 15, 18, 20\}$ and $M = 35, n = 7$
Find out all possible subsets of w whose sum = M draw the state space tree.

Unit - V

Branch and Bound

1. Travelling sales person problem using Least cost Branch and Bound (LCBB):-

Apply the LC Branch and Bound Technique for the following graph adjacency matrix

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \begin{array}{c} \text{min} \\ 10 \\ 2 \\ 2 \\ 3 \\ 4 \\ \hline 21 \end{array}$$

First we will find the minimum of each row. Find out the min value in each and every row and sum those values to get row reduction cost

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

row reduction cost = 10+2+2+3+4 = 21

→ Find out the minimum value in each and every column and sum those values to get the column reduction cost.

$$= \begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \\ 1 & 0 & 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

column reduction cost = 1+0+3+0+0 = 4

* Total reduction cost = row reduction cost + column reduction cost

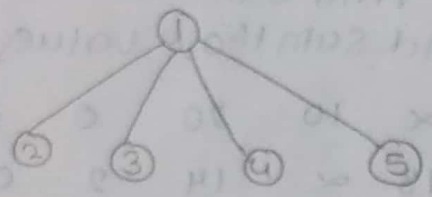
= 21 + 4

= 25

→ The total reduction cost is the cost of the first vertex i.e; $\hat{c}(1) = 25$

After selecting the first vertex, in the remaining vertices we have to select the next minimum cost vertex.

$\hat{c}(1) = 25$



Now consider the reduced matrix

$$A = \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 8 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

consider the path (1,2), make first row and second column as ∞ and set A[2][1] = ∞

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

row reduction cost = 0

column reduction cost = 0

total reduction cost = 0 + 0 = 0

$$c^1(2) = c^1(1) + A(1,2) + 0$$

(matrix 1, edge, reduction cost)

$$= 25 + 10 + 0$$

$$= 35$$

Consider the path (1,3) make first row and third column as ∞ and set $A[3][1] = \infty$

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

row reduction cost = 0

column reduction cost = 11

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & \infty \end{bmatrix}$$

Total reduction cost = 11 + 0 = 11

$$c^1(3) = c^1(1) + A(1,3) + 11$$

$$= 25 + 17 + 11$$

$$= 53$$

Consider the path (1,4) make first row and fourth column as ∞ and set $A[4][1] = \infty$

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

Total reduction cost = 0 + 0 = 0

$$C^{\wedge}(4) = C^{\wedge}(1) + A(1,4) + 0$$

$$= 25 + 0 + 0$$

$$C^{\wedge}(4) = 25$$

Consider the path (1,5) make first row and fifth column as infinity and set $A[5][1] = \infty$

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

row reduction cost = 5

col reduction cost = 0

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

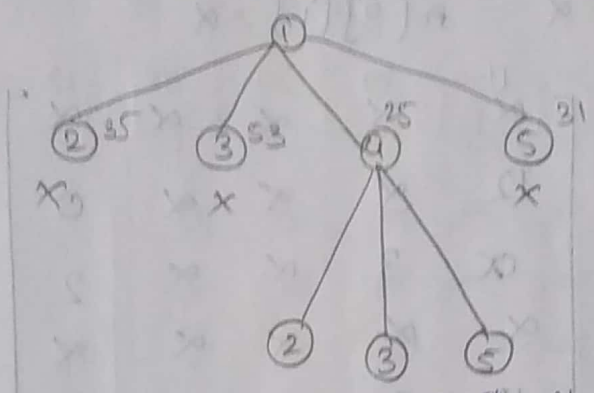
Total reduction cost = 5

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

$$= 25 + 1 + 5$$

$$= 31$$

Among all the nodes 2, 3, 4, 5, node 4 has the minimum value i.e; 25. we will set node 4 as E-Node and generates its children 2, 3 & 5.



consider the path 1-4-2, make first row, 4th row and 2nd column as ∞, set $A[4][1] = \infty$, $A[2][1] = \infty$

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

$$= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix} \begin{matrix} 0 \\ 0 \\ \\ \\ 0 \end{matrix}$$

row reduction cost: 0

Col reduction cost = 0

Total reduction cost = 6+6
= 0

$$c^{\wedge}(2) = c^{\wedge}(4) + A(4,2) + 0$$

$$= 25 + 3 + 0$$

$$= 28$$

Consider the path 1-4-3 make 1st row, 4th row, 3rd col as ∞ and

$A[4][1] = \infty \quad A[3][1] = \infty$

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

Total reduction cost: 0+0 = 0

$$c^{\wedge}(3) = c^{\wedge}(4) + A(4,3) + 0$$

$$= \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \end{bmatrix}$$

Total reduction cost = 11

$$c^{\wedge}(3) = c^{\wedge}(4) + A(4,3) + 11 + 2$$

$$= 25 + 12 + 13$$

$$= 50$$

consider the path 1-4-5 make 1st, 4th rows and 5th col as ∝ and

A[4][1] = ∝ A[5][1] = ∝

$$= \begin{bmatrix} \overset{0}{\infty} & \overset{0}{\infty} & \overset{0}{\infty} & & \\ \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix} \begin{matrix} \\ 11 \\ 0 \\ \\ 0 \end{matrix}$$

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

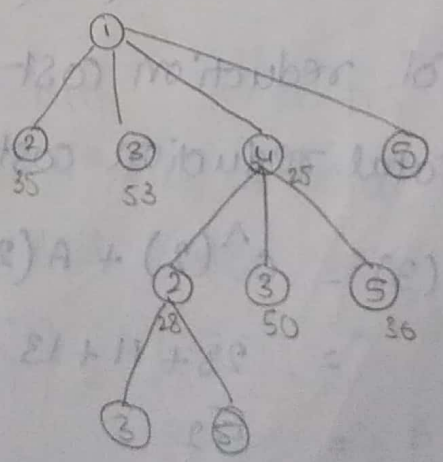
Total reduction cost = 11+0 = 11

$$c\hat{}(5) = c\hat{}(4) + A(4,5) + 11$$

$$= 25 + 0 + 11$$

$$= 36$$

Among the costs of node 2, 3, 5, node 2 is the minimum node 2 is the E-Node. Node 2 generates the children. 3 & 5



The cost matrix obtained at node 2 is the reference matrix now

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

consider the path 1-4-2-3 for node 3.

make first row, 4th row, 2nd row to ∞ and also 3rd column to ∞ set

A[4][1], A[2][1], A[3][1] to ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix}_{11}^{2}$$

row reduction cost = 11+2 = 13

~~col reduction cost = 11+2 = 13~~

~~Total reduction cost = 13+13 = 26~~

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

Col reduction cost = 0

Total reduction cost = 0+13 = 13.

$$c\hat{}(3) = c\hat{}(2) + A(2,3) + 13$$

$$= 28 + 11 + 13$$

$$= 52$$

consider the path 1-4-2-5 for nodes make 1st row, 4th row, 2nd row, 5th col to ∞ and also 5th col to ∞

set A[4][1], A[2][1], A[5][1] = ∞

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

row reduction cost = 0

col reduction cost = 0

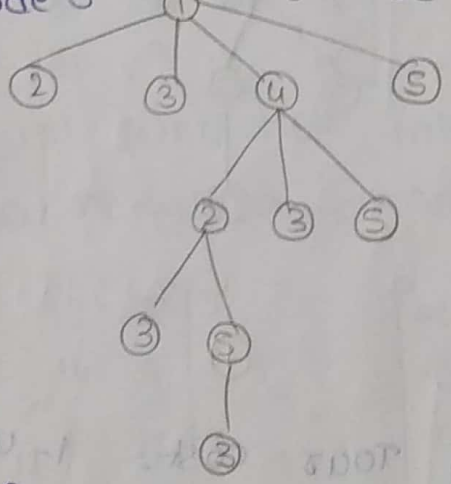
Total reduction cost = 0

$$c^\wedge(5) = c^\wedge(2) + A(2,5) + 0$$

$$= 28 + 0$$

$$c^\wedge(5) = 28$$

Among the costs of 3 and 5. node 5 is minimum node 5 is the E-node it generates the children 3 Now the reference matrix is cost matrix at node 5.



consider the path 1-4-2-5-3 for nodes make 1st row, 4th row, 5th row, 2nd row and 3rd col to ∞ and also set A[4][1], A[2][1], A[5][1], A[3][1] = ∞

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

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

row reduction cost = 0

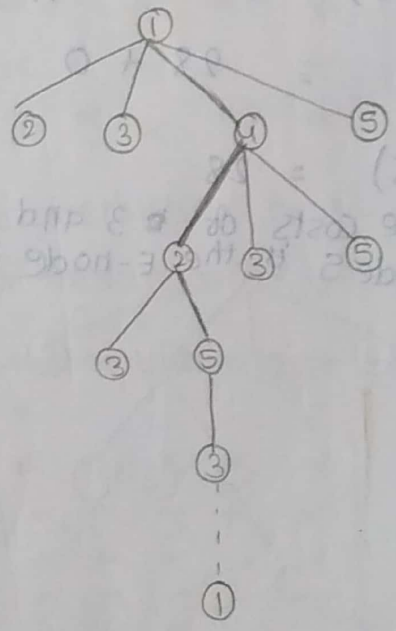
col reduction cost = 0

Total reduction cost = 0

$$C^1(3) = C^1(5) + A(5,3) + 0$$

$$= 28 + 0 + 0$$

$$= 28$$



Minimum Tour cost: 1-4-2-5-3-1

$= 28$

2) consider the travelling sales person instance defined by the cost matrix

$$\begin{bmatrix} \infty & 7 & 3 & 12 & 8 \\ 3 & \infty & 6 & 14 & 9 \\ 5 & 8 & \infty & 6 & 18 \\ 9 & 3 & 5 & \infty & 11 \\ 18 & 14 & 9 & 8 & \infty \end{bmatrix}$$

a) obtain the reduced cost matrix

b) obtain the portion of the state space tree that will be generated by LCBB. label each node by its $\hat{c}$ value. write down the reduced matrices corresponding to each of these nodes.

3) Apply the LCBB Algorithm to solve the travelling salesperson problem for the following cost matrix

$$\begin{bmatrix} \infty & 11 & 10 & 9 & 6 \\ 8 & \infty & 7 & 3 & 4 \\ 8 & 4 & \infty & 4 & 8 \\ 11 & 10 & 5 & \infty & 5 \\ 6 & 9 & 5 & 5 & \infty \end{bmatrix}$$

4) $$A = \begin{bmatrix} \infty & 7 & 3 & 12 & 8 \\ 3 & \infty & 6 & 14 & 9 \\ 5 & 8 & \infty & 6 & 18 \\ 9 & 3 & 5 & \infty & 11 \\ 18 & 14 & 9 & 8 & \infty \end{bmatrix}$$

First we will find the min cost of each row. row reduction cost.

$$\begin{bmatrix} \infty & 7 & 3 & 12 & 8 \\ 3 & \infty & 6 & 14 & 9 \\ 5 & 8 & \infty & 6 & 18 \\ 9 & 3 & 5 & \infty & 11 \\ 18 & 14 & 9 & 8 & \infty \end{bmatrix} \begin{matrix} 3 \\ 3 \\ 5 \\ 3 \\ 8 \end{matrix} \quad \begin{bmatrix} \infty & 4 & 0 & 9 & 5 \\ 0 & \infty & 3 & 11 & 6 \\ 0 & 3 & \infty & 1 & 13 \\ 6 & 0 & 2 & \infty & 8 \\ 10 & 6 & 1 & 0 & \infty \end{bmatrix}$$

row reduction cost = 3+3+5+3+8 = 22

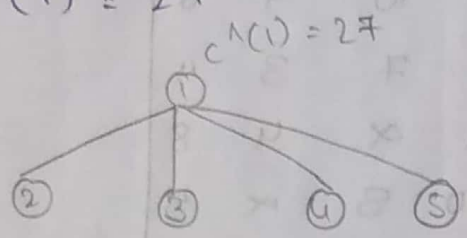
Find the column reduction cost

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

Col reduction cost = 5

Total reduction cost = 22 + 5 = 27

$c\hat{}(1) = 27$



Now consider the reduced matrix

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

consider the path (1,2) make 1st row and 2nd col as ∞ $A[2][1] = \infty$

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

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

row reduction cost = 3

col reduction cost = 0

Total reduction cost = 3

$$c^{\wedge}(2) = c^{\wedge}(1) + A(1,2) + 3$$

$$= 27 + 4 + 3$$

$$= 34$$

Now consider the path 1 - 3 make 1st row, 3rd col as ∞ and A[3][1] = ∞

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

row reduced cost = 1

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

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

col reduced cost = 1

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

Total red cost = 2

$$c^{\wedge}(3) = c^{\wedge}(1) + A(1,3) + 2$$

$$= 27 + 0 + 2$$

$$= 29$$

Now consider the path 1, 4 make 1st row, 4th col to ∞ & A[4][1] = ∞

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

row red cost = 1

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

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 1 \\ 0 & 3 & \infty & \infty & 8 \\ \infty & 0 & 2 & \infty & 3 \\ 9 & 6 & 0 & \infty & \infty \end{bmatrix}$$

col red cost = 1

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 3 & \infty & 0 \\ 0 & 3 & \infty & \infty & 7 \\ \infty & 0 & 2 & \infty & 2 \\ 9 & 6 & 0 & \infty & \infty \end{bmatrix}$$

Total red cost = 2

$$c^1(4) = c^1(1) + A(1,4) + 2$$
$$= 27 + 9 + 2$$
$$= 38$$

Now consider the path 1, 5, make 1st row, 5th col as ∞ and A[5][1] = ∞

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

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

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

row red cost = 0

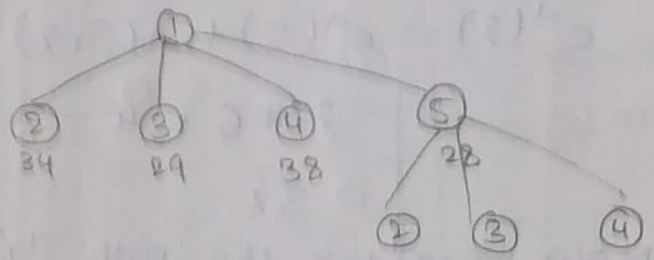
Col red cost = 1

Total red cost = 1

$$c^1(5) = c^1(1) + A(1,5) + 1$$

$$= 27 + 0 + 1$$

$$= 28$$



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

Now consider the path 1-5-2

make 1st row, 5th row and 2nd col as ∞

A[5][1], A[2][1] = ∞

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

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 9 & \infty \\ 0 & \infty & \infty & 1 & \infty \\ 5 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

col reduction cost = 1

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

Total red cost = 3+1 = 4

$c^1(2) = c^1(5) + A(5,2) + 4$

$= 28 + 6 + 4$

$= 38$

Now, consider the path 1,5,3 make 1st row, 5th row and 2nd col as ∞ and

$A[5][1] = \infty$, $A[3][1] = \infty$

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

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 2 & 11 & \infty \\ \infty & \infty & \infty & 0 & \infty \\ 5 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \quad \begin{pmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 11 & \infty \\ \infty & 2 & \infty & 0 & \infty \\ 6 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

row red cost = 1

col red cost = 0

Total red cost = 1

$c^1(3) = c^1(5) + A(5,3) + 1$

$= 28 + 0 + 1$

$= 29$

Now consider the path 1,5,4 make 1st row, 5th row and 4th col as ∞ and $A[5][1] = \infty$ $A[4][1] = \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 2 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} 0 \\ 0 \\ 0 \end{matrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 1 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

row red cost = 0

col red cost = 1

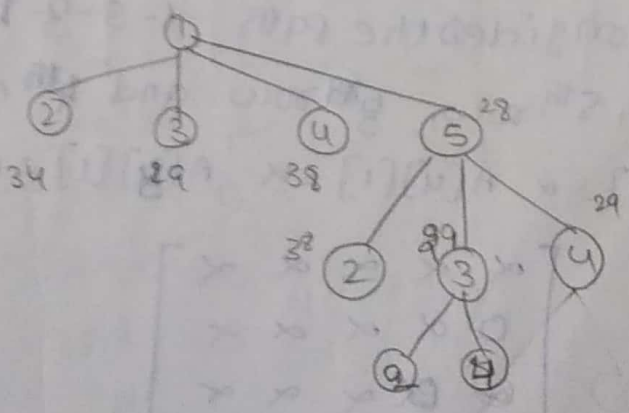
Total red cost = 1

$c^1(4) = c^1(5) + A(5,4) + 1$

$= 28 + 0 + 1$

$= 29$

Among these $c^1(4)$ is minimum



Now consider the path 1-5-4-2 make 1st, 5th, 4th rows as ∝ and 2nd col as ∝

A[5][1] = ∝ A[4][1] = ∝ A[2][1] = ∝

$$A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 10 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} \\ 10 \\ \\ 6 \\ \end{matrix}$$

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

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

row red cost = 16

col red cost = 0

Total red cost = 16

$$c^1(2) = c^1(1) + A(1,2) + 16$$

$$= 29 + 0 + 16$$

$$= 48$$

Now consider the path 1-5-3-4 make 1st row, 5th row, 3rd row and 4th col as ∝

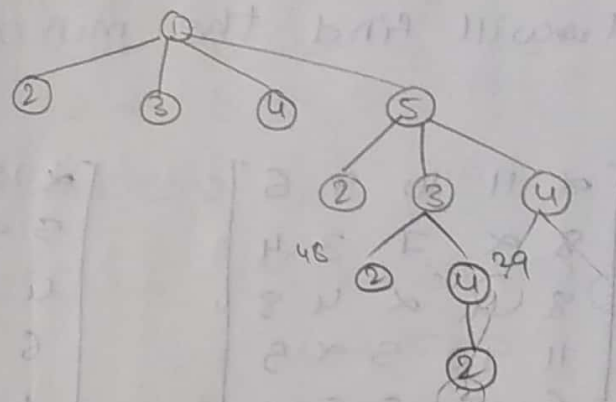
A[5][1] = ∝ A[4][1] = ∝ A[3][1] = ∝

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} \\ 0 \\ \\ 2 \\ \\ \end{matrix}$$

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

Total red cost = 0

$$c^1(3) = c^1(4) + A(4,3) + 0$$
$$= 29 + 0 + 0$$
$$= 29$$

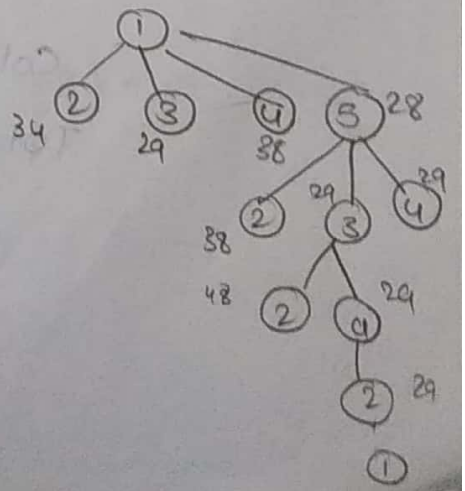


Now consider the path 1-5-4-2-3
make 1st row, 5th row, 4th row, 2nd row, 3rd col as ∞ and $A[5][1] = \infty$ $A[4][1] = \infty$ $A[2][1] = \infty$ $A[3][1] = \infty$

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

Total red cost = 0

$$c^1(2) = 29 + 0 + 0$$
$$= 29$$



1 - 5 - 3 - 4 - 2

3]

$$A = \begin{bmatrix} \infty & 11 & 10 & 9 & 6 \\ 8 & \infty & 7 & 3 & 4 \\ 8 & 4 & \infty & 4 & 8 \\ 11 & 10 & 5 & \infty & 5 \\ 6 & 9 & 5 & 5 & \infty \end{bmatrix}$$

First will find the min cost of each row

$$\begin{bmatrix} \infty & 11 & 10 & 9 & 6 \\ 8 & \infty & 7 & 3 & 4 \\ 8 & 4 & \infty & 4 & 8 \\ 11 & 10 & 5 & \infty & 5 \\ 6 & 9 & 5 & 5 & \infty \end{bmatrix} \begin{matrix} 6 \\ 3 \\ 4 \\ 5 \\ 5 \end{matrix} \qquad \begin{bmatrix} \infty & 5 & 4 & 3 & 0 \\ 5 & \infty & 4 & 0 & 1 \\ 4 & 0 & \infty & 0 & 4 \\ 6 & 5 & 0 & \infty & 0 \\ 1 & 4 & 0 & 0 & \infty \end{bmatrix}$$

row red cost = 6 + 3 + 4 + 5 + 5 = 23

Find the col red cost

1 0 0 0 0

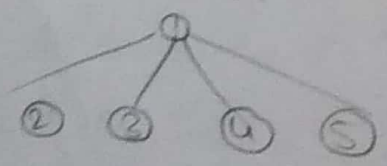
$$\begin{bmatrix} \infty & 5 & 4 & 3 & 0 \\ 5 & \infty & 4 & 0 & 1 \\ 4 & 0 & \infty & 0 & 4 \\ 6 & 5 & 0 & \infty & 0 \\ 1 & 4 & 0 & 0 & \infty \end{bmatrix} \qquad \begin{bmatrix} \infty & 5 & 4 & 3 & 0 \\ 4 & \infty & 4 & 0 & 1 \\ 3 & 0 & \infty & 0 & 4 \\ 5 & 5 & 0 & \infty & 0 \\ 0 & 4 & 0 & 0 & \infty \end{bmatrix}$$

Col red cost = 1

Total red cost = 23 + 1 = 24

$c^{\wedge}(1) = 24$

24



Now consider the reduced matrix

$$A = \begin{bmatrix} \infty & 5 & 4 & 3 & 0 \\ 4 & \infty & 4 & 0 & 1 \\ 3 & 0 & \infty & 0 & 4 \\ 5 & 5 & 0 & \infty & 0 \\ 0 & 4 & 0 & 0 & \infty \end{bmatrix}$$

consider the path 1,2 make 1st row, 2nd col as ∞ and $A[2][1] = \infty$.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 4 & 0 & 1 \\ 3 & \infty & \infty & 0 & 4 \\ 5 & \infty & 0 & \infty & 0 \\ 0 & \infty & 0 & 0 & \infty \end{bmatrix} \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix}$$

Total red cost = 0

$$c^1(2) = c^1(1) + A(1,2) + 0$$
$$= 24 + 5 + 0$$
$$= 29$$

Now consider the path 1,3 make 1st row 3rd col as ∞ and $A[3][1] = \infty$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 4 & \infty & \infty & 0 & 1 \\ \infty & 0 & \infty & 0 & 4 \\ 5 & 5 & \infty & \infty & 0 \\ 0 & 4 & \infty & 0 & \infty \end{bmatrix} \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix}$$

Total red cost = 0

$$c^1(3) = c^1(1) + A(1,3) + 0$$
$$= 24 + 4$$
$$= 28$$

Now consider the path 1,4 make 1st row 4th col as ∞ and A[4][1] = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 4 & \infty & 4 & \infty & 1 \\ 3 & 0 & \infty & \infty & 4 \\ \infty & 5 & 0 & \infty & 0 \\ 0 & 4 & 0 & \infty & \infty \end{bmatrix} \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 3 & \infty & 3 & \infty & 0 \\ 3 & 0 & \infty & \infty & 4 \\ \infty & 5 & 0 & \infty & 0 \\ 0 & 4 & 0 & \infty & \infty \end{bmatrix}$$

Total red cost = 1

$$c^1(4) = c^1(1) + A(1,4) + 1$$

$$= 24 + 3 + 1$$

$$= 28$$

Now consider the path 1,5 make 1st row 5th col as ∞ and A[5][1] = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 4 & \infty & 4 & 0 & \infty \\ 3 & 0 & \infty & 0 & \infty \\ 5 & 5 & 0 & \infty & \infty \\ \infty & 4 & 0 & 0 & \infty \end{bmatrix} \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix}$$

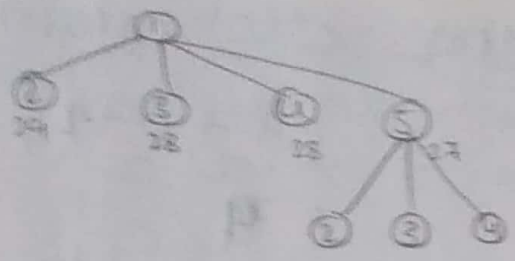
$$A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 4 & 0 & \infty \\ 0 & 0 & \infty & 0 & \infty \\ 2 & 5 & 0 & \infty & \infty \\ \infty & 4 & 0 & 0 & \infty \end{bmatrix}$$

Total red cost = 3

$$c^1(5) = c^1(1) + A(1,5) + 3$$

$$= 24 + 0 + 3$$

$$= 27$$



$$A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 4 & 0 & \infty \\ 0 & 0 & \infty & 0 & \infty \\ 2 & 5 & 0 & \infty & \infty \\ \infty & 4 & 0 & 0 & \infty \end{bmatrix}$$

Now consider the path 1-5-2 make 1st row, 5th row and 2nd col as ∞ and A[5][1] = ∞ A[2][1] = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 4 & 0 & \infty \\ 0 & \infty & \infty & 0 & \infty \\ 2 & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} 0 \\ 0 \\ 0 \end{matrix}$$

Total red cost = 0

$$C^1(2) = C^1(5) + A(5,2) + 0$$

$$= 27 + 4$$

$$= 31$$

Now consider the path 1-5-3 make 1st row 5th row 2nd col as ∞ and A[5][1] = ∞ A[3][1] = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 0 & \infty \\ \infty & 0 & \infty & 0 & \infty \\ 2 & 5 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} 0 \\ 0 \\ 2 \end{matrix}$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 4 & 0 & \infty \\ \infty & 0 & \infty & 0 & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Total red cost = 2

$$c^1(3) = c^1(5) + A(5,3) + 1$$

$$= 27 + 0 + 1$$

$$= 28$$

Now consider the path 1-5-4 make 1st row, 5th row 4th col as ∝ and A[5][1] = ∝ A[4][1] = ∝

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 4 & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \\ \infty & 5 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} \\ 1 \\ 0 \\ 0 \\ \end{matrix}$$

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

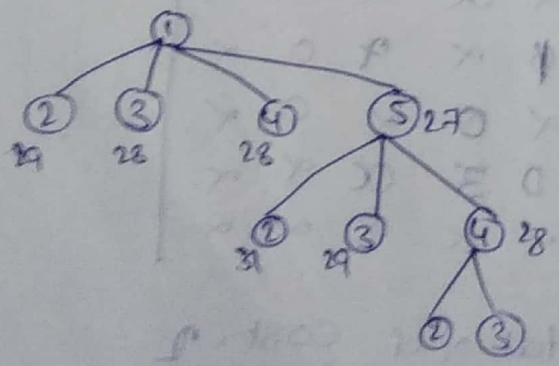
Total red cost = 1

$$c^1(4) = c^1(5) + A(5,4) + 1$$

$$= 27 + 0 + 1$$

$$= 28$$

Among 2, 3, 4 node 4 is minimum. It generates children nodes node2, node3



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

consider the path 1-5-4-2 make 1st, 5th, 4th rows and 2nd col as ∞ A[5][1] = ∞

A[4][1] = ∞ A[2][1] = ∞

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 3 & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \begin{matrix} 3 \\ 0 \end{matrix} \qquad \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Total red cost = 3

$$c^1(2) = 28 + 5 + 3$$

$$= 36$$

consider the path 1-5-4-3 make 1st, 5th, 4th rows and 3rd col as ∞ A[5][1] = ∞, A[4][1] = ∞

A[3][1] = ∞

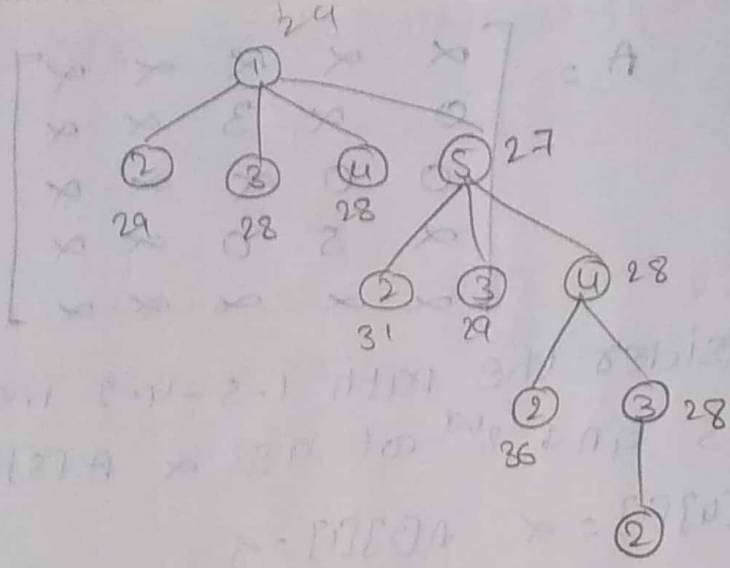
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$$c^1(3) = 28 + 0 + 0$$

$$= 28$$

Node 3 is minimum

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

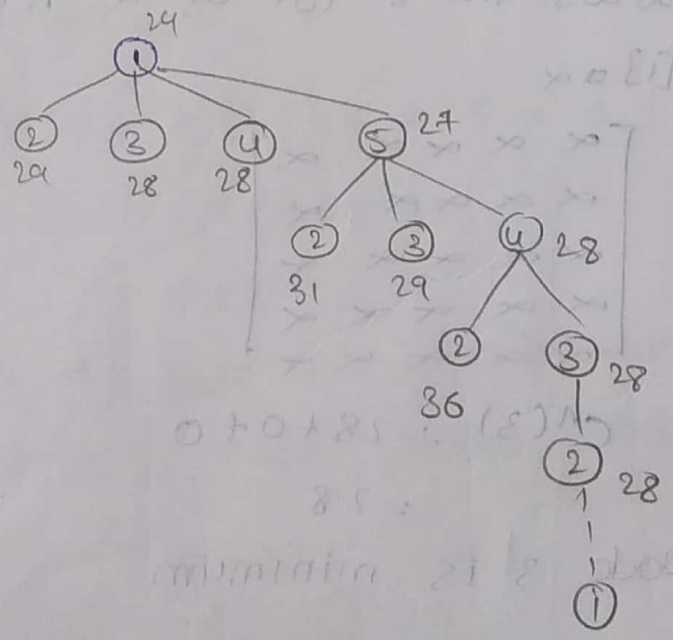


consider 1-5-4-3-2 make 1st row, 5th row 4th row, 3rd row 2nd col as ∝.

A[5][1] = ∝ A[4][1] = ∝ A[3][1] = ∝ A[2][1] = ∝

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

$$C^{\wedge}(2) = 28 + 0 + 0 = 28$$



= 1 - 5 - 4 - 3 - 2 = 28

Knapsack problem using Branch & Bound:-

Knapsack problem is a maximization problem because we always require the maximum profit from the knapsack problem.

But knapsack problem using LCBB (Least cost Branch & Bound) is a minimization problem.

By using minimization problem we have to find out the solution for maximization problem.

→ To convert maximization problem into minimization problem. first convert all profits into negative profits by putting -ve sigh before each profit value

→ knapsack problem using LCBB, we have to calculate two bounds for each node.

→ one is lower bound ($c\hat{}$). second one is upper bound ($\hat{c}$)

→ while calculating lower bound for a node, it allows fractions.

→ while calculating upper bound, we doesn't allow fractions.

→ In Branch & Bound, there are two variations of knapsack problem

1st one is 1) LCBB knapsack.
2) FIFOBB knapsack.

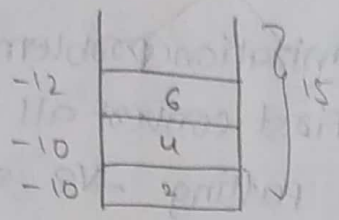
1) Draw the portion of the statespace tree generated by LCBB knapsack, for the knapsack instances $n=4$, $m=15$, $(P_1, P_2, P_3, P_4) = (10, 10, 12, 18)$ $(w_1, w_2, w_3, w_4) = (2, 4, 6, 9)$

First convert the all profits into negative profits by putting negative sign infront of each profit value

$\therefore (P_1, P_2, P_3, P_4) = (-10, -10, -12, -18)$

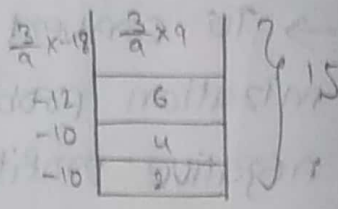
we have to calculate lower bound $\hat{c}$ and upper bound $\hat{u}$ for each node

For node 1:-



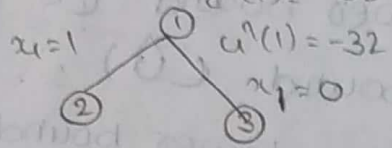
$\hat{u}(1) = -10 - 10 - 12$

$= -32$



$\hat{c}(1) = -10 + (-10) + (-12) + \left(\frac{3}{9} \times -18\right)$

$= -38$

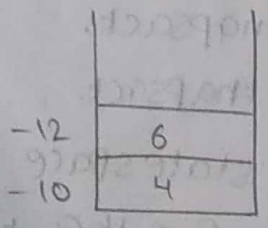


for node 2: $x_1 = 1$

$\hat{c}(2) = -38$

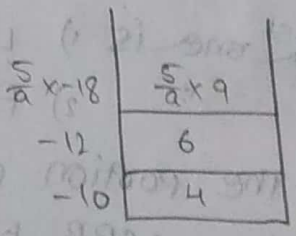
$\hat{u}(2) = -32$

for node 3: $x_1 = 0$



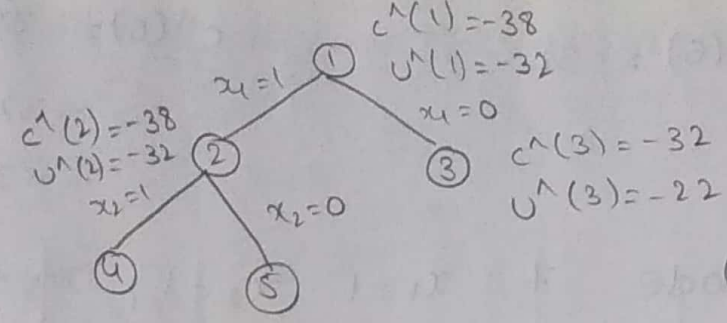
$\hat{u}(3) = -10 - 12$

$= -22$



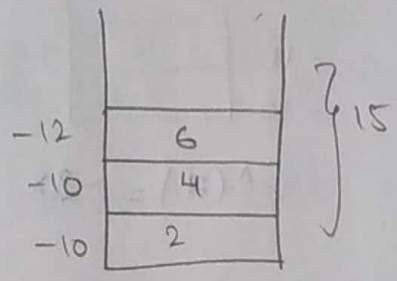
$\hat{c}(3) = -10 - 12 + \frac{5}{9} \times -18$

$= -32$



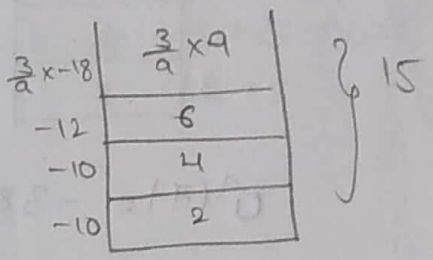
(least lowerbound is considered)

For node 4:- $x_1 = 1$ $x_2 = 1$



$u^(4) = -10-10-12$

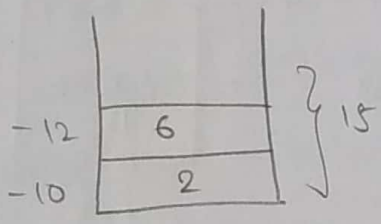
$= -32$



$c^(4) = -10-10-12+\frac{3}{9}x-18$

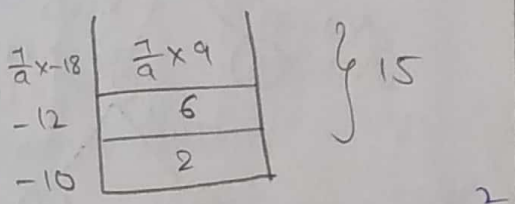
$= -38$

For node 5:- $x_1 = 1$, $x_2 = 0$



$u^(5) = -10-12$

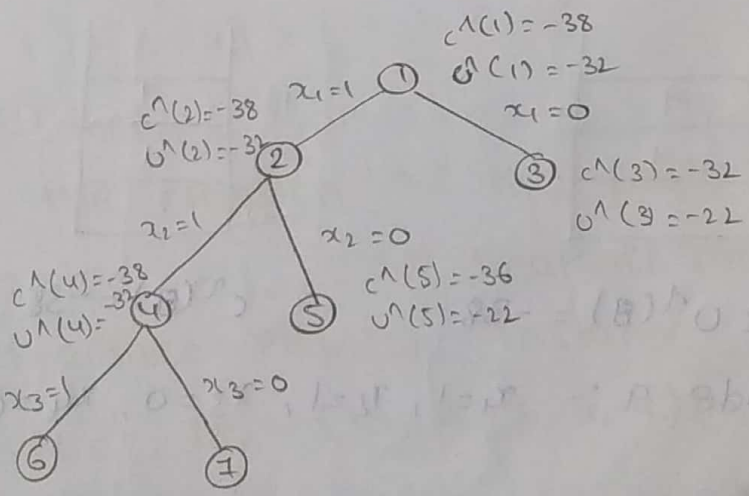
$= -22$



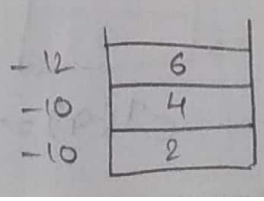
$c^(5) = -10-12+\frac{7}{9}x-18$

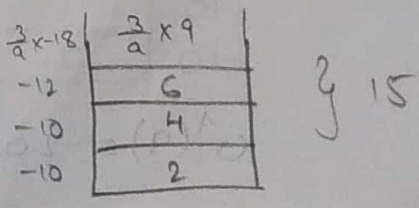
$= -22 - 14$

$= -36$



For node 6: $x_1 = 1$, $x_2 = 1$ $x_3 = 1$





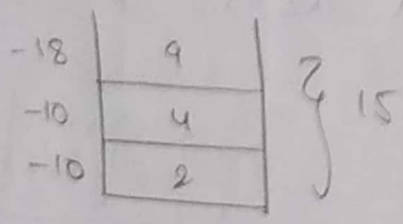
$U^1(6) = -32$

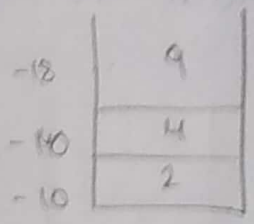
$C^1(6) = -32 + \frac{8}{2} \times 1$

$= -32 - 6$

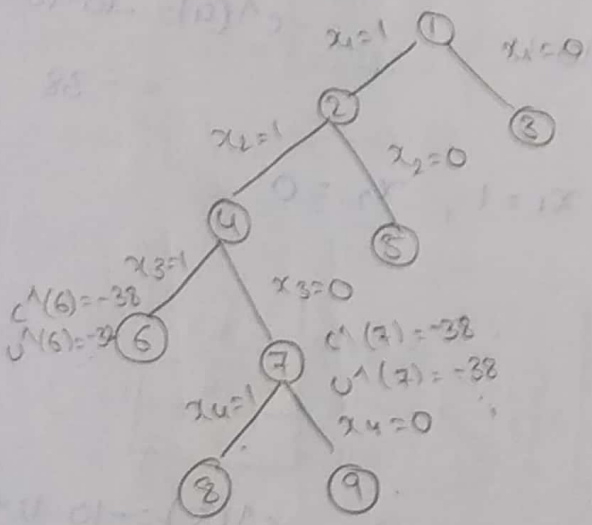
$= -38$

for node 7 : $x_1 = 1$ $x_2 = 1$, $x_3 = 0$

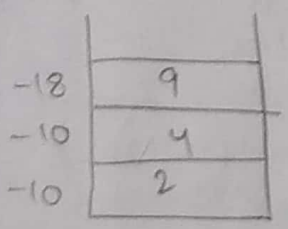


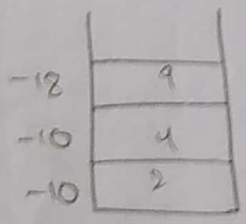


$U^1(7) = -38$ $C^1(7) = -38$



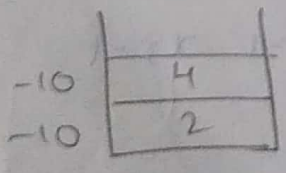
for node 8 : $x_1=1$, $x_2=1$, $x_3=0$, $x_4=1$

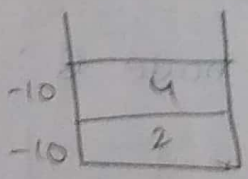




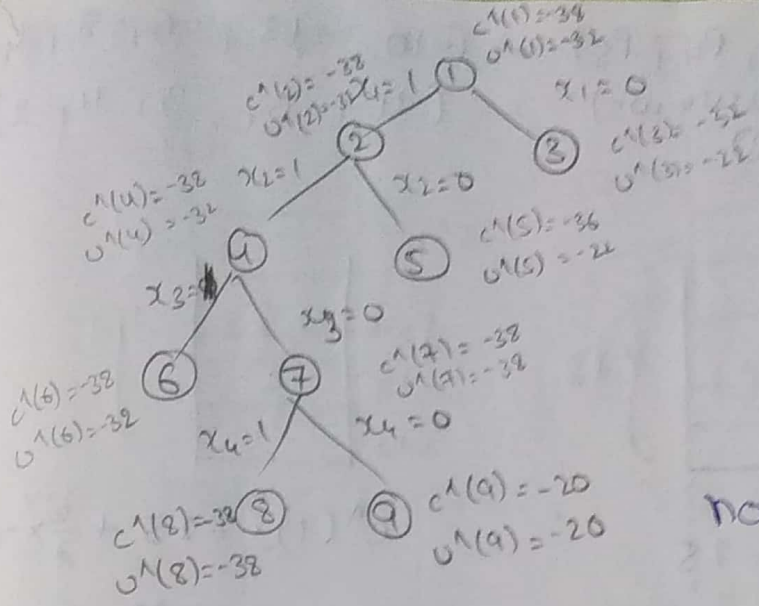
$\therefore U^1(8) = -38$ $C^1(8) = -38$

for node 9 :- $x_1=1$, $x_2=1$, $x_3=0$, $x_4=0$





$U^1(9) = -20$ $C^1(9) = -20$



node 8 is E-node

Active node = 8

Path = 1 - 2 - 4 - 7 - 8

Solution vector

$x[1:4] = [1, 1, 0, 1]$

Profit = $-10 - 10 - 18 = -38$

Weight = $2 + 4 + 9 = 15$

To convert the minimization problem into maximization problem we have to put + sign before each profit value

$x[1:4] = [1, 1, 0, 1]$

Profit = $10 + 10 + 0 + 18 = 38$

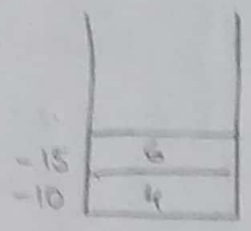
Draw the portion of the state space tree generated by LCBB knapsack problem for the knapsack problem instance $n = 5$, $m = 12$, $(P_1 - P_5) = (10, 15, 6, 8, 4)$ $(w_1 - w_5) = (4, 6, 3, 4, 2)$

first convert all profits into negative values.

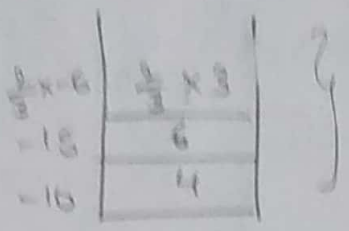
$(P_1, P_2, P_3, P_4, P_5) = (10, 15, 6, 8, 4)$

$(w_1, w_2, w_3, w_4, w_5) = (4, 6, 3, 4, 2)$

For node (1)

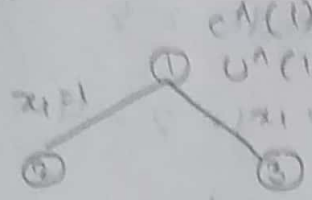


$U^1(1) = -25$

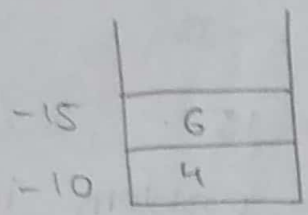


$c^1(1) = -25 + \frac{2}{3} \times -6$

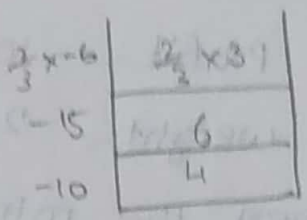
$= -29$



For node 2 :- $x_1 = 1$

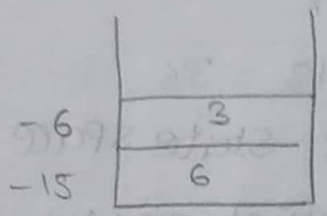


$U^1(2) = -25$

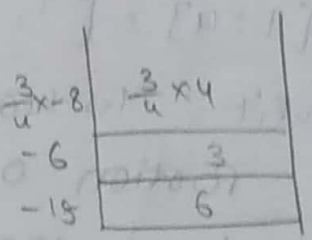


$c^1(2) = -29$

For node 3 :- $x_1 = 0$



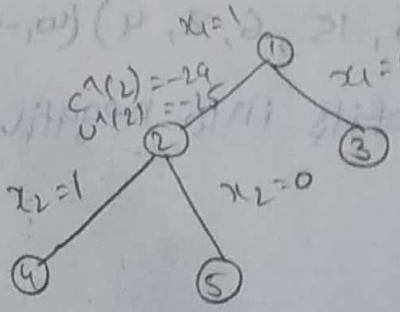
$U^1(3) = -21$



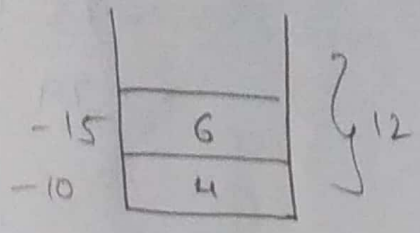
$c^1(3) = -21 + \frac{3}{4} \times -8$

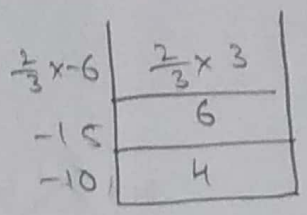
$= -21 - 6$

$= -27.$



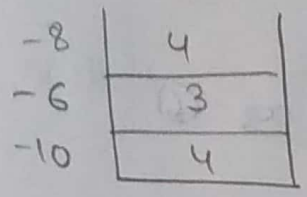
$x_1 = 1 \quad x_2 = 1$

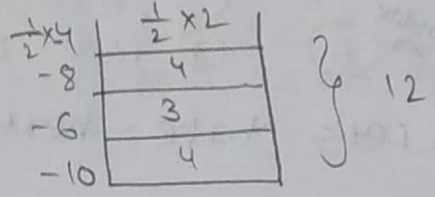




$u^\wedge(4) = -25 \qquad c^\wedge(4) = -29$

for node 5 : $x_1 = 1, \quad x_2 = 0$



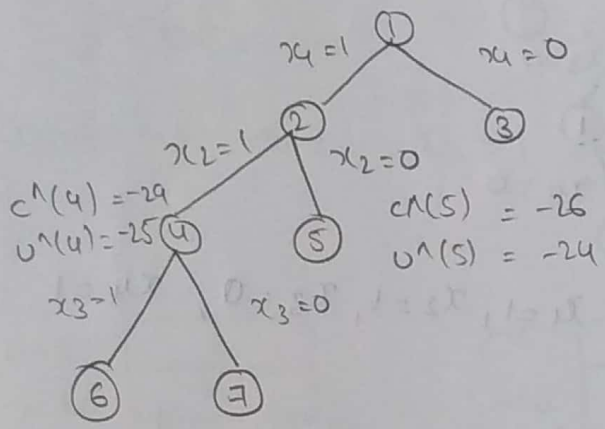


$u^\wedge(5) = -10-6-8$

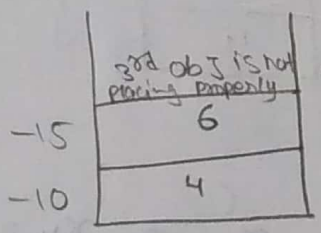
$= -24$

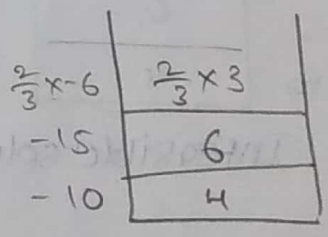
$c^\wedge(5) = -24 + \frac{1}{2}x-4$

$= -26$



for node 6 :- $x_1 = 1, \; x_2 = 1, \; x_3 = 1$

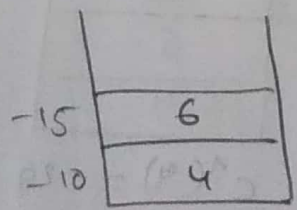


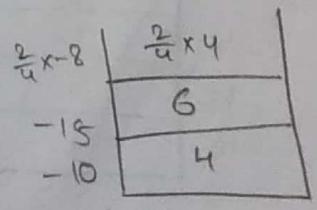


$u^\wedge(6) = -25 \qquad c^\wedge(6) = -29$

Infeasible solution ($x_3 = 1$)

for node 7 :- $x_1 = 1, \; x_2 = 1, \; x_3 = 0$



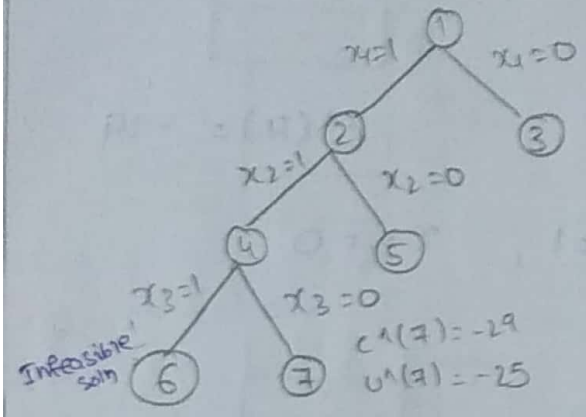


$\hat{u}(7) = -25$

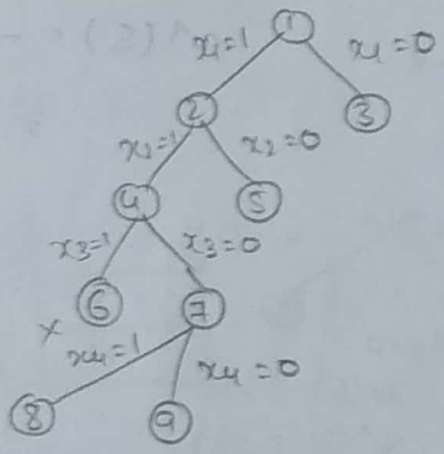
$\hat{c}(7) = -25 + \frac{2}{4} \times 4$

$= -25 - 4$

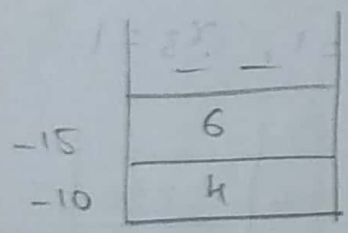
$= -29$

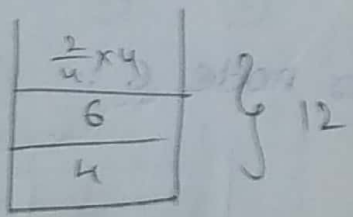


(i) Let's take next node as (6)



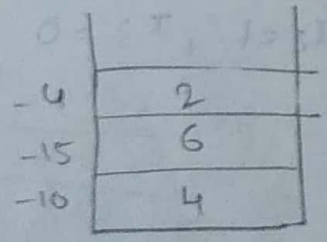
for node 8 : $x_1=1, x_2=1, x_3=0, x_4=1$



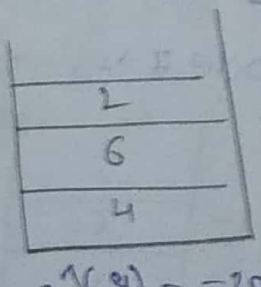


Infeasible solution.

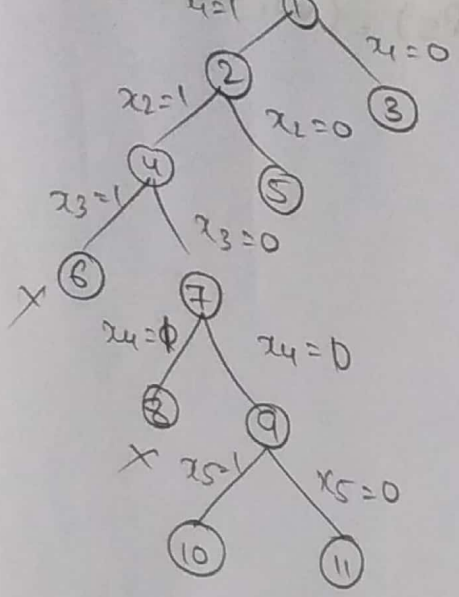
for node 9 :- $x_1=1, x_2=1, x_3=0, x_4=0$



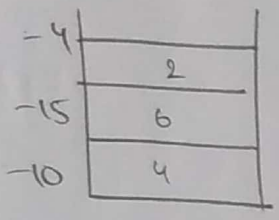
$\hat{u}(9) = -29$



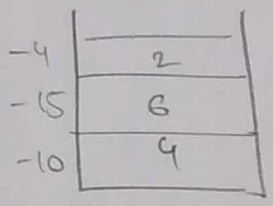
$\hat{c}(9) = -29$



For node 10 : $x_1=1,\ x_2=1,\ x_3=0,\ x_4=0,\ x_5=1$

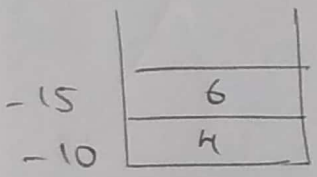


$U^{\wedge}(10) = -29$

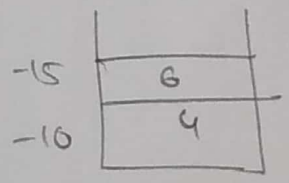


$C^{\wedge}(10) = -29$

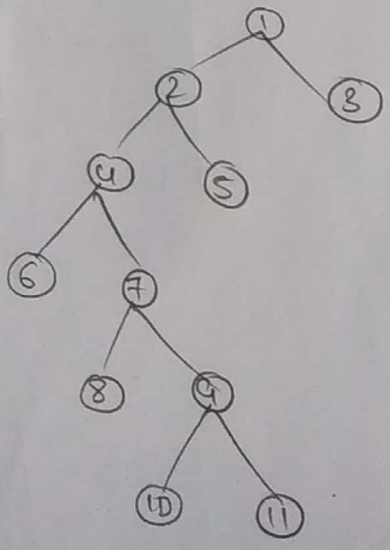
for node 11 : $x_1=1,\ x_2=1,\ x_3=x_4=x_5=0$



$U^{\wedge}(11) = -25$



$C^{\wedge}(11) = -25$



$x = [1\ 1\ 0\ 0\ 1]$

General method of Branch and Bound:

Branch and Bound is a systematic method for solving optimization problems

Branch and Bound technique applied where the greedy method and dynamic programming fail.

However, Branch and Bound is much slower indeed it often leads to exponential time complexities in the worst case.

→ On the other hand, if applied carefully it can lead to algorithms that run reasonably fast on average.

→ The general idea of Branch and Bound is a BFS like search for the optimal solution, but not all nodes get expanded. Rather, a carefully selected criterion determines which node to expand and when, and another criterion tells the algorithm when a optimal solution has been found.

→ Branch and Bound refers to all statespace search methods in which all children of an E-node are generated before any other live node can becomes the E-node.

→ Both BFS and DFS generalized to Branch and Bound strategies.

→ BFS is an FIFO search in terms of live nodes. List of live nodes are represented in the form of a queue.

Bfs is an LIFO search in terms of live nodes. List of live nodes are represented in the form of a stack.

→ Just like Backtracking, we will use bounding functions to avoid generating sub trees that donot contain an answer node

→ Branch and Bound method of algorithm design involves 2 steps.

1. Tree organization of the solution search space.
2. use of bounding functions to limit the search i.e; To help avoid the generation of subtrees that donot contain an answer node.

Examples for B&B method:-

1. 0/1 Knapsack Problem
2. Travelling Sales Person problem (TSP)

Search Techniques used in B&B:-

1. Bfs and DFS
2. Least cost search Techniques.

→ Bfs and Dfs are used for storing the live nodes and selection of next E-node among the live nodes based on FIFO (queue) and LIFO (stack) respectively.

→ In B&B method,

Branch: Break down the problem into sub-problems.

Bound: Compute the bounds in every sub-problem.

→ In the solution of every B & B problem can be represented in the form of a tree called state space tree. In state space tree there are 3 types of nodes used

(i) Live node
(ii) E-node
(iii) dead-node

(i) Live node :-

A node which has been generated but childrens have not yet been generated. is called a live node.

(ii) E-node :-

A live node whose childrens are currently being generated is called a E-node.

(iii) Dead node :-

A node which is already expanded and there is no use in future is called a dead node.

→ Bounding functions are used to kill live nodes without generating their children.

* Differences b/w Backtracking & Branch and bound.

Backtracking	Branch & Bound
1) In backtracking, DFS Technique is used for tracking the solution for the given problem.	1) In B & B, BFS Technique is used for Tracking the solution for the given problem.
2) Typically decision problems (Yes/no) can be solved using backtracking	2) typically optimization problems can be solved using B & B.

3) ~~Dfs~~ Finding the solution to the given problem, bad choices are also evaluated	3) B&B proceeds only on optimal or better solutions.
4) The state space tree is searched until the solution is obtained.	4) The state space tree is needs to be searched completely as there may be chances of being optimal solution anywhere in the state space tree.
5) Applications of backtracking are n-queen problem, graph colouring problem, hamiltonian cycle problem, sum of subsets problem	5) Applications of B&B are 0/1 knapsack problem, Travelling sales person problem

* 0/1 knapsack problem using FIFO Branch & Bound:

→ 0/1 knapsack problem is a maximization problem but FIFO Branch and Bound is a minimization problem. To convert this maximization problem into minimization problem, change the +ve profits into -ve profits by putting the -ve sign infront of each profit.

→ In 0/1 knapsack FIFO BB problem, we have to calculate two bounds.

1. lower bound ($c^\wedge$)
   - lower bound allows fractions.
2. upper bound ($\hat{u}$)
   - upper bound doesnot allow fractions.

→ while solving the 0/1 knapsack problem using FIFO BB, the following steps we have to follow.

1) Set the node 1 upperbound as global upper bound ($\hat{U}$)
2) If the lower bound of any node is greater than the global upperbound, kill that node, otherwise the node can be expanded.
3) If any node upperbound is less than the global upperbound, then update the global upper bound value.
4) If the objects are considered one by one, some of those objects greater than the capacity of the knapsack, then the node is killed. Hence, that node has infeasible solution.

4) Draw the state space tree for the following 0/1 knapsack instance using FIFO BB

$n = 4$ $(P_1, P_2, P_3, P_4) = (10, 10, 12, 18)$

$(w_1, w_2, w_3, w_4) = (2, 4, 6, 9)$ $M = 15$

0/1 knapsack problem is a maximization problem but we require FIFO BB knapsack problem is a minimization problem.

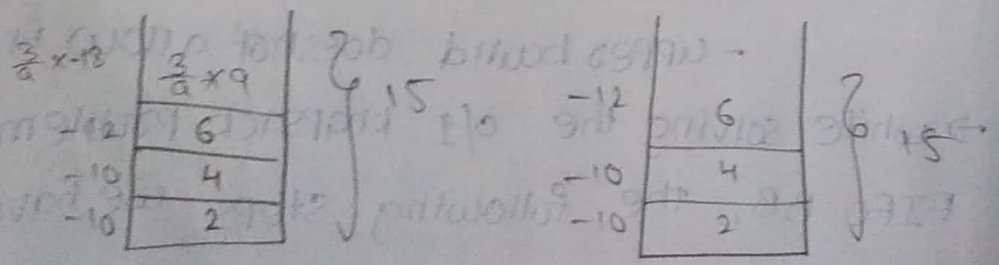
Hence, we convert the +ve profits into -ve profits. Hence, the problem is converted from maximization problem to minimization problem.

$(P_1, P_2, P_3, P_4) = (-10, -10, -12, -18)$

$(w_1, w_2, w_3, w_4) = (2, 4, 6, 9)$

$M = 15$ $n = 4$

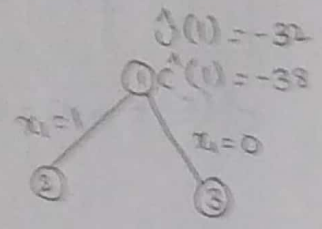
for node 1:



$c^\wedge(1) = -10-10-12-6$
$= -38$

$\hat{u}(1) = -10-10-12$
$= -32$

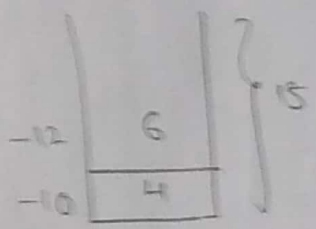
$\hat{u} = -32$ as global upper bound. $-38 > -32$
false
so extend node ①

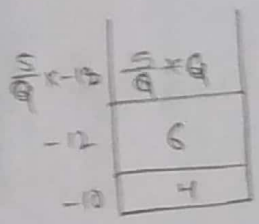


for node 2:-

$c^\wedge(2) = -38$
$u^\wedge(2) = -32$

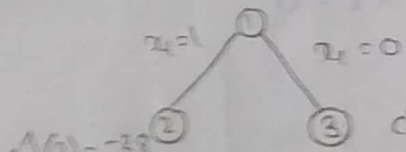
for node 3:- $x_1 = 0$





$\hat{u}(3) = -10-12 = -22$

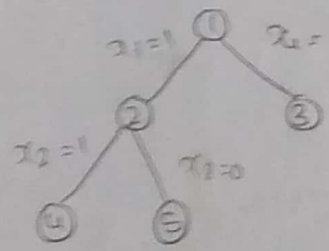
$c^\wedge(3) = -10-12-10 = -32$



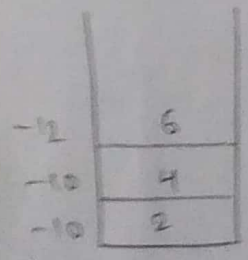
$c^\wedge(2) = -38$
$u^\wedge(2) =$

$c^\wedge(3) = -32$
$u^\wedge(3) = -22$

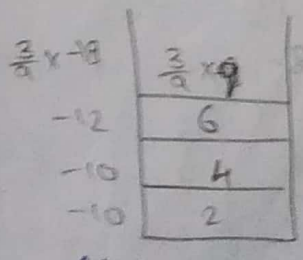
$-38 > -32$ false. Extend node ②



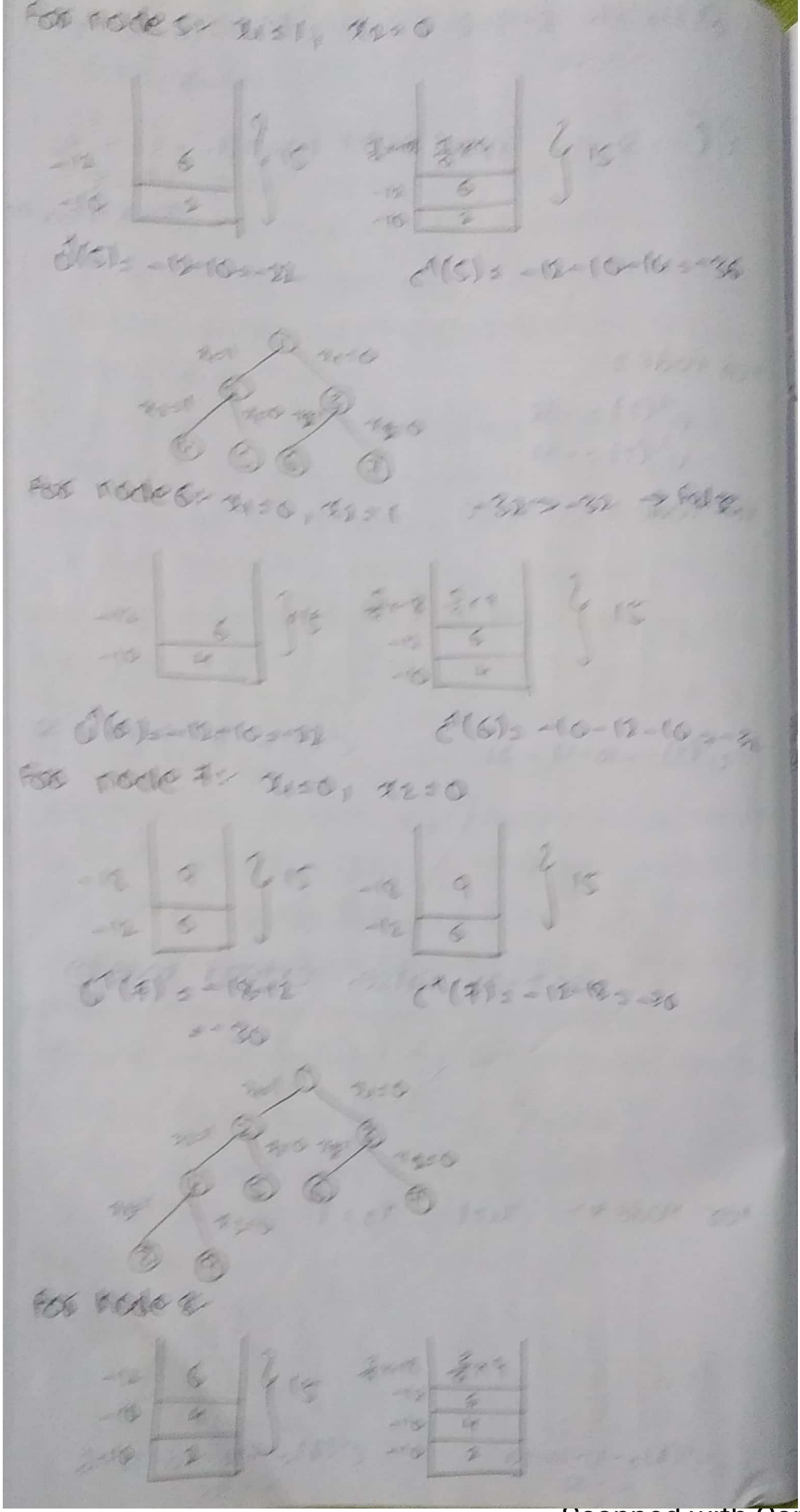
for node 4:- $x_1 = 1$, $x_2 = 1$



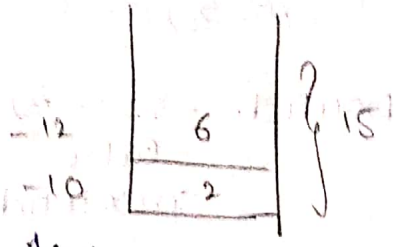
$u^\wedge(4) = -12-10-10$
$= -32$

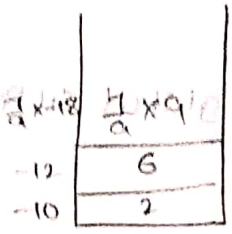


$c^\wedge(4) = -10-10-12-6$
$= -38$

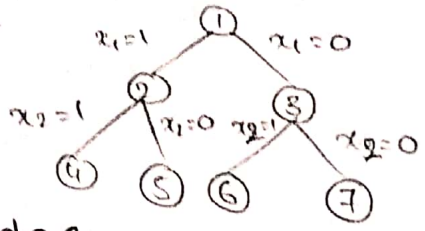


For node 5:- $x_1 = 1$, $x_2 = 0$

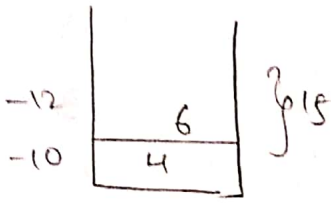


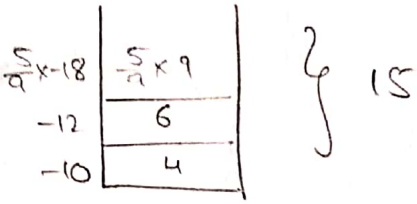


$U^1(5) = -12-10 = -22$ $\quad C^1(5) = -12-10-14 = -36$



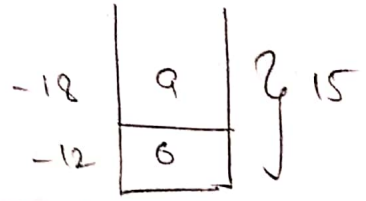
For node 6:- $x_1 = 0$, $x_2 = 1$ $\quad -32 > -32 \rightarrow$ False

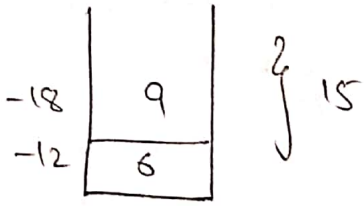




$U^1(6) = -12-10 = -22$ $\quad C^1(6) = -10-12-10 = -32$

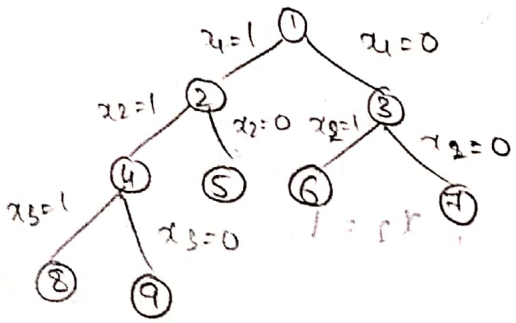
For node 7:- $x_1 = 0$, $x_2 = 0$



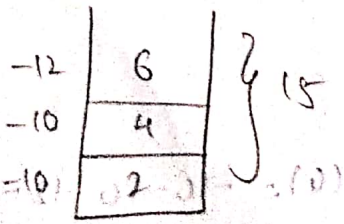


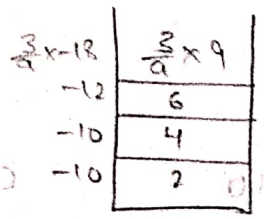
$U^1(7) = -18-12$
$= -30$

$C^1(7) = -12-18 = -30$



For node 8:-





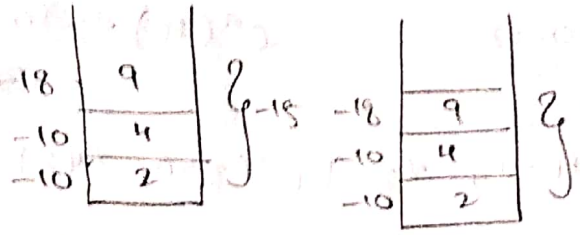
$u^1(8) = -10-10-12$

$= -32$

$c^1(8) = -10-10-12-6$

$= -38$

for node 9: $x_1=1,\ x_2=1,\ x_3=0$



$u^1(9) = -10-10-18$

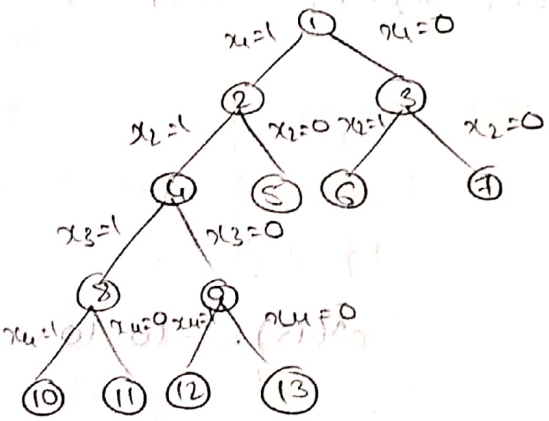
$= -38$

$c^1(9) = -10-10-18$

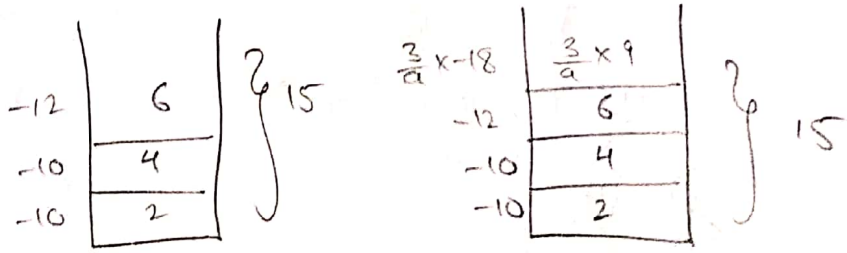
$= -38$

Here the upper bound -38 is greater than global upperbound -32. so, we have to update the global upper bound.

$u^1 = -38$ as global upper bound



for node 10:- $x_1=1,\ x_2=1,\ x_3=1,\ x_4=1$



$u^1(10) = -10-10-12$

$= -32$

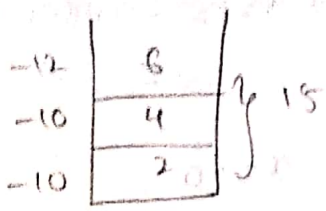
$c^1(10) = -10-10-12-6$

$= -38$

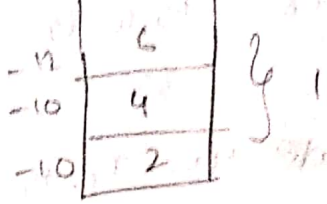
Infeasible solution

for node 11:-

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

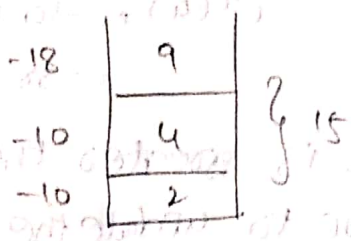


$u^\wedge(11) = -10-10-12$
$= -32$

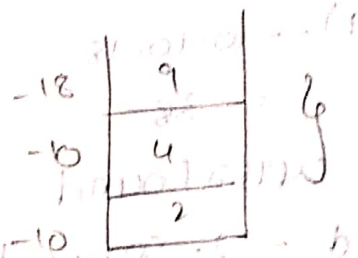


$c^\wedge(11) = -10-10-12$
$= -32$

For node 12:- $x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$

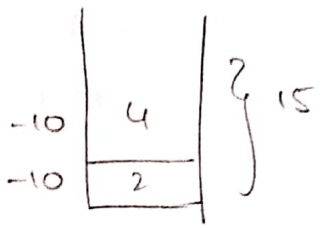


$u^\wedge(12) = -10-10-18$
$= -38$

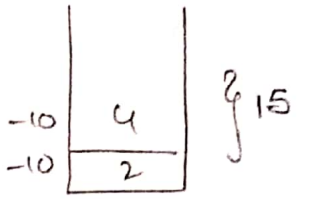


$c^\wedge(12) = -10-10-18$
$= -38$

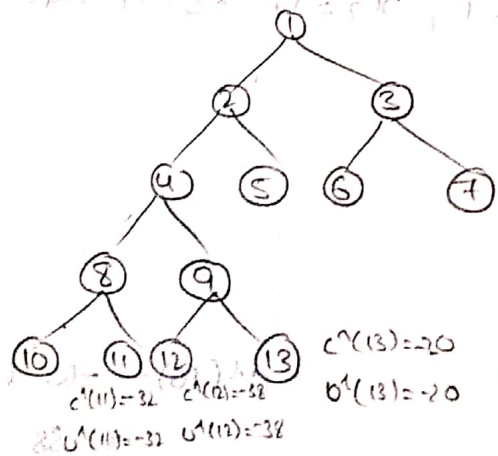
For node 13:- $x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 0$



$u^\wedge(13) = -10-10$
$= -20$



$c^\wedge(13) = -10-10 = -20$



→ If the lower bound is greater than global upper bound kill that node.

→ If lower bound is less than global upper bound extend that node.

active node is 12

solution vector $[x_1, x_2, x_3, x_4] = [1, 1, 0, 1]$

Max profit $= -10 - 10 - 18 = -38$

weight $= 2 + 4 + 0 + 9 = 15$

Now, convert the minimization problem into maximization problem by changing -ve profit sighs to +ve profit sighs

solution vector $[x_1, x_2, x_3, x_4] = [1, 1, 0, 1]$

Max profit $= 10 + 10 + 18 = 38$

2) Draw the portion of the state space tree generated by the FIFO B B technique for the knapsack instance of $n=5$, $m=12$, $(P_1 - P_5) = (10, 15, 6, 8, 4)$ $(w_1 - w_5) = (4, 6, 3, 4, 2)$

3) Apply the least cost Branch and Bound technique for the following adjacency matrix using travelling sales person problem.

$$\begin{bmatrix} \infty & 7 & 3 & 12 & 8 \\ 3 & \infty & 6 & 14 & 9 \\ 5 & 8 & \infty & 6 & 18 \\ 9 & 3 & 5 & \infty & 11 \\ 18 & 14 & 9 & 8 & \infty \end{bmatrix}$$

UNIT - III

Dynamic programming:

* Travelling sales person problem:-(TSP)

→ Let $G=(V,E)$ be a directed graph with edge cost c_{ij} where 'V' denotes the set of vertices and 'E' denotes the set of edges. The edges are given along with their costs c_{ij} the cost $c_{ij} > 0$ for all i and j.

$$cost[i,j] = \begin{cases} 0 & \text{if } i=j \\ c_{ij} & (i,j) \in E \\ \infty & (i,j) \notin E \end{cases}$$

→ A tour for the graph G is a directed cycle that includes every vertex in 'V' exactly once. The cost of the tour is the sum of the cost of the edges on the tour and the cost must be minimum.

→ The travelling salesperson problem objective is to find out the tour of the minimum cost.

→ Let us assume that the tour starts from vertex '1' and ends at vertex 1. Every tour consists of an edge $(1,k)$ for some $k \in V-\{1\}$ and a path from vertex k to vertex 1.

→ The path from vertex k to vertex 1 goes through each vertex in $V-\{1,k\}$ exactly once.

→ Let $G(i,S)$ be the length of a shortest path starting at vertex i, going through all the vertices S and terminating edge vertex 1.

→ $g(i,S) = \min_{j \in S} \{c_{ij} + g(j, S-\{j\})\}$

where $S = V-\{i\}$

Example:-

→ suppose we have route a postal van to pickup mails from mail boxes located at 'n' different locations in a city.

→ n+1 vertex graph can be used to represent this situation. one vertex represents the post office from which the postal van starts and to which it must return.

→ Edge (i,j) is assigned a cost equal to the distance from location i to location j. The root taken by the postal van is a tour and we are interested in finding a tour of minimum cost.

→ let us consider a graph with 4 vertices, whose cost adjacency matrix of the graph given below

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

Find out the optimal tour cost.

$$g(i,S) = \min_{j \in S} \{c_{ij} + g(j, S-\{j\})\}$$

i) $|S| = \phi$

$$g(1,\phi) = c_{11} = c(i,1) = c(1,1) = 0$$

$$g(2,\phi) = c_{21} = c(2,1) = 5$$

$$g(3,\phi) = c_{31} = c(3,1) = 6$$

$$g(4,\phi) = c_{41} = c(4,1) = 8$$

b) $|S| = 1$

$g(2, \{3\}) = \min_{j \in 3} \{c_{23} + g(3, \{3\} - \{3\})\}$

$= \min \{9 + g(3, \phi)\}$

$= \min \{9 + 6\}$

$= 15$

$g(2, \{4\}) = \min_{j \in 4} \{c_{24} + g(4, \{4\} - \{4\})\}$

$= \min \{10 + g(4, \phi)\}$

$= \min \{10 + 8\}$

$= 18$

$g(3, \{2\}) = \min_{j \in 2} \{c_{32} + g(2, \{2\} - \{2\})\}$

$= \min \{13 + g(2, \phi)\}$

$= \min \{13 + 5\}$

$= 18$

$g(3, \{4\}) = \min_{j \in 4} \{c_{34} + g(4, \{4\} - \{4\})\}$

$= \min \{12 + g(4, \phi)\}$

$= \min \{12 + 8\}$

$= 20$

$g(4, \{2\}) = \min_{j \in 2} \{c_{42} + g(2, \{2\} - \{2\})\}$

$= \min \{8 + g(2, \phi)\}$

$= \min \{8 + 5\}$

$= 13$

$g(4, \{3\}) = \min_{j \in 3} \{c_{43} + g(3, \{3\} - \{3\})\}$

$= \min \{9 + g(3, \phi)\}$

$= \min \{9 + 6\}$

$= 15$

3) $|S| = 2$

$g(2, \{3,4\}) = \min_{j \in 3,4} \{ c_{2,3} + g(3, \{3,4\} - \{3\}),\ c_{2,4} + g(4, \{3,4\} - \{4\}) \}$

$= \min\{9 + g(3, \{4\}), 10 + g(4, \{3\})\}$

$= \min\{9+20, 10+15\}$

$= \min\{29, 15\}$

$= 25$

$g(2, \{4,3\}) = \min_{j \in 4,3} \{ c_{2,4} + g(4, \{3,4\} - \{4\}),\ c_{2,3} + g(3, \{3,4\} - \{3\}) \}$

$= \min\{10 + g(4, \{3\}), 9 + g(3, \{4\})\}$

$= \min\{10+15, 9+20\}$

$= \min\{25, 29\}$

$= 25$

$g(3, \{2,4\}) = \min_{j \in 2,4} \{ c_{32} + g(2, \{2,4\} - \{2\}),\ c_{34} + g(4, \{2,4\} - \{4\}) \}$

$= \min\{13 + g(2, \{4\}), 12 + g(4, \{2\})\}$

$= \min\{13+18, 12+13\}$

$= \min\{32, 25\}$

$= 25$

$g(3, \{4,2\}) = \min_{j \in 4,2} \{ c_{34} + g(4, \{2,4\} - \{4\}),\ c_{32} + g(2, \{2,4\} - \{2\}) \}$

$= \min\{12 + g(4, \{2\}), 13 + g(2, \{4\})\}$

$= \min\{12+13, 13+18\}$

$= \min\{25, 32\}$

$= 25$

$g(4, \{2,3\}) = \min_{j \in 2,3} \{ c_{42} + g(2, \{2,3\} - \{2\}),\ c_{43} + g(3, \{2,3\} - \{3\}) \}$

$= \min\{8 + g(2, \{3\}), 9 + g(3, \{2\})\}$

$= \min\{8+15, 9+18\} = \min\{23, 27\}$

$= 23$

$$g(4,\{3,2\}) = \min_{j \in 3,2} \{ c_{43} + g(3,\{2,3\}-\{3\}),\ c_{42} + g(2,\{2,3\}-\{2\}) \}$$

$$= \min\{9+g(3,\{2\}),\ 8+g(2,\{3\})\}$$

$$= \min\{9+18,\ 8+15\}$$

$$= \min\{27, 23\}$$

$$= 23$$

4) $|S| = 3$

$$g(1,\{2,3,4\}) = \min_{j \in 2,3,4} \begin{cases} c_{12} + g(2,\{2,3,4\}-\{2\}) \\ c_{13} + g(3,\{2,3,4\}-\{3\}) \\ c_{14} + g(4,\{2,3,4\}-\{4\}) \end{cases}$$

$$= \min\begin{cases} 10+g(2,\{3,4\}) \\ 15+g(3,\{2,4\}) \\ 20+g(4,\{2,3\}) \end{cases}$$

$$= \min\{10+25,\ 15+25,\ 20+23\}$$

$$= \min\{35, 40, 43\}$$

$$= 35$$

Optimal tour = 35

Path = 1-2-4-3-1 with cost = 35

	1	2	3	4	5
1	0	20	30	10	11
2	15	0	16	4	2
3	3	5	0	2	4
4	19	6	18	0	3
5	16	4	17	16	0

1. $|S| = \phi$

$g(1,\phi) = c_{11} = c(1,1) = c(1,1) = 0$

$g(2,\phi) = c_{21} = c(2,1) = 15$

$g(3,\phi) = c_{31} = c(3,1) = 3$

$g(4,\phi) = c_{41} = c(4,1) = 19$

$g(5,\phi) = c_{51} = c(5,1) = 16$

2. $|S| = 1$

$g(2, \{3\}) = \min_{j \in 3} \{ c_{2,3} + g(3, \{3\} - \{3\}) \}$

$= \min \{ 16 + g(3, \phi) \}$

$= \min \{ 16 + 3 \}$

$= 19$

$g(2, \{4\}) = \min_{j \in 4} \{ c_{2,4} + g(4, \{4\} - \{4\}) \}$

$= \min \{ 4 + g(4, \phi) \}$

$= \min \{ 4 + 19 \}$

$= 23$

$g(2, \{5\}) = \min_{j \in 5} \{ c_{2,5} + g(5, \{5\} - \{5\}) \}$

$= \min \{ 2 + g(5, \phi) \}$

$= \min \{ 2 + 16 \}$

$= 18$

$g(3, \{2\}) = \min_{j \in 2} \{ c_{3,2} + g(2, \{2\} - \{2\}) \}$

$= \min \{ 5 + g(2, \phi) \}$

$= \min \{ 5 + 15 \}$

$= 20$

$g(3, \{4\}) = \min_{j \in 4} \{ c_{3,4} + g(4, \{4\} - \{4\}) \}$

$= \min \{ 2 + g(4, \phi) \}$

$= \min \{ 2 + 19 \}$

$= 21$

$g(3, \{5\}) = \min_{j \in 5} \{ c_{3,5} + g(5, \{5\} - \{5\}) \}$

$= \min \{ 4 + g(5, \phi) \}$

$= \min \{ 4 + 16 \}$

$= 20$

$g(4, \{2\}) = \min_{j \in 2} \{ c_{4,2} + g(2, \{2\} - \{2\}) \}$

$= \min \{ 6 + g(2, \phi) \}$

$= \min \{ 6 + 15 \}$

$= 21$

$g(4,\{3\}) = \min_{j\in 3}\{c_{4,3} + g(3,\{3\}-\{3\})\}$

$= \min\{18 + g(3,\phi)\}$

$= \min\{18+3\}$

$= 21$

$g(4,\{5\}) = \min_{j\in 5}\{c_{4,5} + g(5,\{5\}-\{5\})\}$

$= \min\{3 + g(5,\phi)\}$

$= \min\{3+16\}$

$= 19$

$g(5,\{2\}) = \min_{j\in 2}\{c_{5,2} + g(2,\{2\}-\{2\})\}$

$= \min\{4 + g(2,\phi)\}$

$= \min\{4+15\}$

$= 19$

$g(5,\{3\}) = \min_{j\in 3}\{c_{5,3} + g(3,\{3\}-\{3\})\}$

$= \min\{7 + g(3,\phi)\}$

$= \min\{7+3\}$

$= 10$

$g(5,\{4\}) = \min_{j\in 4}\{c_{5,4} + g(4,\{4\}-\{4\})\}$

$= \min\{16 + g(4,\phi)\}$

$= \min\{16+19\}$

$= 35$

3. $|S| = 2$

$g(2,\{3,4\}) = \min_{j\in 3,4}\{c_{2,3} + g(3,\{3,4\}-\{3\}),\ c_{2,4} + g(4,\{3,4\}-\{4\})\}$

$= \min\{16 + g(3,\{4\}),\ 4 + g(4,\{3\})\}$

$= \min\{16+21,\ 4+21\}$

$= \min\{37, 25\}$

$= 25$

$g(2, \{4,3\}) = \min\{25, 37\}$

$= 25$

$g(2,\{4,5\}) = \min_{j \in 4,5} \{c_{2,4} + g(4, \{4,5\} - \{4\}), c_{2,5} + g(5, \{4,5\} - \{5\})\}$

$= \min\{4 + g(4, \{5\}), 2 + g(5, \{4\})\}$

$= \min\{4+19, 2+35\}$

$= \min\{23, 37\}$

$= 23$

$g(2, \{5,4\}) = \min\{37, 23\}$

$= 23$

$g(2, \{3,5\}) = \min_{j \in 3,5} \{c_{2,3} + g(3, \{3,5\} - \{3\}), c_{2,5} + g(5, \{3,5\} - \{5\})\}$

$= \min\{16 + g(3, \{5\}), 2 + g(5, \{3\})\}$

$= \min\{16+20, 2+10\}$

$= \min\{36, 12\}$

$= 12$

$g(2, \{5,3\}) = \min\{12, 36\}$

$= 12$

$g(3, \{2,4\}) = \min_{j \in 2,4} \{c_{3,2} + g(2, \{2,4\} - \{2\}), c_{3,4} + g(4, \{2,4\} - \{4\})\}$

$= \min\{5 + g(2, \{4\}), 2 + g(4, \{2\})\}$

$= \min\{5+23, 2+21\}$

$= \min\{28, 23\}$

$= 23$

$g(3, \{4,2\}) = \min\{23, 28\}$

$= 23$

$g(3, \{2,5\}) = \min_{j \in 2,5} \{c_{3,2} + g(2, \{2,5\} - \{2\}), c_{3,5} + g(5, \{2,5\} - \{5\})\}$

$= \min\{5 + g(2, \{5\}), 4 + g(5, \{2\})\}$

$= \min\{5+18, 4+19\}$

$= \min\{23, 23\}$

$= 23$

$g(3,\{5,2\}) = \min\{23, 23\}$

$= 23$

$g(3,\{4,5\}) = \min_{j \in 4,5}\{c_{3,4} + g(4,\{4,5\}-\{4\}), c_{3,5} + g(5,\{4,5\}-\{5\})\}$

$= \min\{2+g(4,\{5\}), 4+g(5,\{4\})\}$

$= \min\{2+19, 4+35\}$

$= \min\{21, 39\}$

$= 21$

$g(3,\{5,4\}) = \min\{39, 21\}$

$= 21$

$g(4,\{2,3\}) = \min_{j \in 2,3}\{c_{4,2} + g(2,\{2,3\}-\{2\}), c_{4,3} + g(3,\{2,3\}-\{3\})\}$

$= \min\{6+g(2,\{3\}), 18+g(3,\{2\})\}$

$= \min\{6+19, 18+20\}$

$= \min\{25, 38\}$

$= 25$

$g(4,\{3,2\}) = \min\{38, 25\}$

$= 25$

$g(4,\{2,5\}) = \min_{j \in 2,5}\{c_{4,2} + g(2,\{2,5\}-\{2\}), c_{4,5} + g(5,\{2,5\}-\{5\})\}$

$= \min\{6+g(2,\{5\}), 3+g(5,\{2\})\}$

$= \min\{6+18, 3+19\}$

$= \min\{24, 22\}$

$= 22$

$g(4,\{5,2\}) = \min\{22, 24\}$

$= 22$

$g(4,\{3,5\}) = \min_{j \in 3,5}\{c_{4,3} + g(3,\{3,5\}-\{3\}), c_{4,5} + g(5,\{3,5\}-\{5\})\}$

$= \min\{18+g(3,\{5\}), 3+g(5,\{3\})\}$

$= \min\{18+20, 3+10\} = \min\{38, 13\}$

$= 13$

$g(4,\{5,3\}) = \min\{13, 38\}$

$= 13$

$g(5,\{2,3\}) = \min_{j \in 2,3} \{c_{5,2} + g(2,\{2,3\}-\{2\}),\ c_{5,3} + g(3,\{2,3\}-\{3\})\}$

$= \min\{4 + g(2,\{3\}), 7 + g(3,\{2\})\}$

$= \min\{4+19, 7+20\}$

$= \min\{23, 27\}$

$= 23$

$g(5,\{3,2\}) = \min\{27, 23\}$

$= 23$

$g(5,\{2,4\}) = \min_{j \in 2,4} \{c_{5,2} + g(2,\{2,4\}-\{2\}),\ c_{5,2} + g(4,\{2,4\}-\{4\})\}$

$= \min\{4 + g(2,\{4\}),\ 16 + g(4,\{2\})\}$

$= \min\{4+23, 16+21\}$

$= \min\{27, 37\}$

$= 27$

$g(5,\{4,2\}) = 27$

$g(5,\{3,4\}) = \min_{j \in 3,4} \{c_{5,3} + g(3,\{3,4\}-\{3\}),\ c_{5,4} + g(4,\{3,4\}-\{4\})\}$

$= \min\{7 + g(3,\{4\}),\ 16 + g(4,\{3\})\}$

$= \min\{7+21, 16+21\}$

$= \min\{28, 37\}$

$= 28$

$g(5,\{4,3\}) = \min\{37, 28\}$

$= 28$

4. $|S| = 3$

$g(2,\{3,4,5\}) = \min_{j \in 3,4,5} \{c_{2,3} + g(3,\{3,4,5\}-\{3\}),\ c_{2,4} + g(4,\{3,4,5\}-\{4\}),\ c_{2,5} + g(5,\{3,4,5\}-\{5\})\}$

$= \min\{16 + g(3, \{4,5\}), 4 + g(4, \{3,5\}),$

$2 + g(5, \{3,4\})\}$

$= \min\{16+21, 4+15, 2+28\}$

$= \min\{37, 17, 30\}$

$= 17.$

$g(2, \{4,3,5\}) = 17$

$g(2, \{5,3,4\}), g(2, \{4,5,3\}), g(2, \{5,4,3\}),$

$g(2, \{5,3,4\}) = 17$

$g(3, \{2,4,5\}) = \min_{j \in 2,4,5} \{c_{3,2} + g(2, \{2,4,5\} - \{2\}),$

$c_{3,4} + g(4, \{2,4,5\} - \{4\}),$

$c_{3,5} + g(5, \{2,4,5\} - \{5\})\}$

$= \min\{5 + g(2, \{4,5\}), 2 + g(4, \{2,5\}), 4 + g(5, \{2,4\})\}$

$= \min\{5+23, 2+22, 4+27\}$

$= \min\{28, 24, 31\}$

$= 24$

$g(3, \{2,5,4\}), g(3, \{4,2,5\}), g(3, \{4,5,2\}), g(3, \{5,2,4\}),$

$g(3, \{5,4,2\}) = 24$

$g(4, \{2,3,5\}) = \min_{j \in 2,3,5} \{c_{4,2} + g(2, \{2,3,5\} - \{2\}),$

$c_{4,3} + g(3, \{2,3,5\} - \{3\}),$

$c_{4,5} + g(5, \{2,3,5\} - \{5\})\}$

$= \min\{6 + g(2, \{3,5\}), 18 + g(3, \{2,5\}),$

$3 + g(5, \{2,3\})\}$

$= \min\{6+12, 18+23, 3+23\}$

$= \min\{18, 41, 26\}$

$= 18$

$g(4, \{2,5,3\}), g(4, \{3,2,5\}), g(4, \{3,5,2\}),$

$g(4, \{5,2,3\}), g(4, \{5,3,2\}) = 18$

$g(5,\{2,3,4\}) = \min_{j\in\{2,3,4\}} \{ c_{5,2} + g(2,\{2,3,4\}-\{2\}),\ c_{5,3} + g(3,\{2,3,4\}-\{3\}),\ c_{5,4} + g(4,\{2,5,4\}-\{4\})\}$

$= \min\{4+g(2,\{3,4\}),\ 7+g(3,\{2,4\}),\ 6+g(4,\{2,3\})\}$

$= \min\{4+25,\ 7+23,\ 16+25\}$

$= \min\{29, 30, 41\}$

$= 29$

$g(5,\{2,4,3\}), g(5,\{3,2,4\}), g(5,\{3,4,2\}), g(5,\{4,2,3\})$

$g(5,\{4,3,2\}) = 29$

5. $|S| = 4$

$g(1,\{2,3,4,5\}) = \min_{j\in\{2,3,4,5\}} \{ c_{12}+g(2,\{2,3,4,5\}-\{2\}),\ c_{13}+g(3,\{2,3,4,5\}-\{3\}),\ c_{14}+g(4,\{2,3,4,5\}-\{4\}),\ c_{15}+g(5,\{2,3,4,5\}-\{5\})\}$

$= \min\{20+g(2,\{3,4,5\}), 30+g(3,\{2,4,5\}), 10+g(4,\{2,3,5\}), 11+g(5,\{2,3,4\})\}$

$= \min\{20+17, 30+24, 10+18, 11+29\}$

$= \min\{37, 54, 28, 40\}$

$= 28$

1-4-2-5-3-1 with cost 28.

* All pairs shortest path problem (Floyd's / Warshall's Algorithm):-

Let $G=(V,E)$ be a directed graph with n vertices. Let cost be the cost of a adjacency matrix for graph G such that $c(i,i) = 0$, $i \le j \le n$.

cost(i,j) be the length (or) cost of the edge(i,j) if $(i,j) \in E(G)$ and $cost(i,j) = \infty$, if $i \neq j$ and $(i,j) \notin E(G)$.

All pairs shortest path problem
a matrix A such that A(i,j) is the length of a shortest path from i to j.

* The time complexity of all pairs shortest path problem is $O(n^3)$ using the principle of optimality.

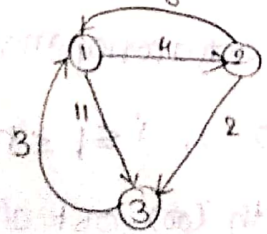
* Let us Examine a shortest path from i to j in G (i ≠ j). This path originates at vertex i and goes through some intermediate vertices and terminates at vertex j. We can assume that this path contains no cycle. If 'k' is an intermediate vertex on this shortest path, then the subpath from i to k and k to j must be shortest paths respectively. Hence, the principle of optimality holds.

* If 'k' is an intermediate vertex with highest index, then i to k path is a shortest path in G going through no vertex with index > k-1.

* $A^k(i,j)$ is the length of a shortest path from i to j going through no vertex of index > k.

$$A^k(i,j) = \min\{A^{k-1}(i,j), A^{k-1}(i,k) + A^{k-1}(k,j)\}$$

1) Consider the following graph and find out the shortest distance between every pair of vertices using dynamic programming.



Take k=0

$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix} \end{array}$$

take k=1

$A^1(i,j) = \min\{A^{1-1}(i,j), A^{1-1}(i,k) + A^{1-1}(k,j)\}$

$A^1(1,2) = \min\{A^{1-1}(1,2), A^{1-1}(1,1) + A^{1-1}(1,2)\}$

$= \min\{A^0(1,2), A^0(1,1) + A^0(1,2)\}$

$= \min\{4, 0+4\}$

$= \min\{4,4\}$

$= 4$

$A^1(1,3) = \min\{A^0(1,3), A^0(1,1) + A^0(1,3)\}$

$= \min\{11, 0+11\}$

$= 11$

$A^1(2,1) = \min\{A^0(2,1), A^0(2,1) + A^0(1,1)\}$

$= \min\{6, 6+0\}$

$= 6$

$A^1(2,3) = \min\{A^0(2,3), A^0(2,1) + A^0(1,3)\}$

$= \min\{2, 6+11\}$

$= \min\{2,17\}$

$= 2$

$A^1(3,1) = \min\{A^0(3,1), A^0(3,1) + A^0(1,2)\}$

$= \min\{3, 3+0\}$

$= 3$

$A^1(3,2) = \min\{A^0(3,2), A^0(3,1) + A^0(1,2)\}$

$= \min\{\infty, 3+4\}$

$= \min\{\infty, 7\}$

$= 7$

$$A^1 = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Take $k=2$

$A^2(i,j) = \min\{A^{2-1}(i,j), A^{2-1}(i,k) + A^{2-1}(k,j)\}$

$A^2(1,2) = \min\{A^{2-1}(1,2), A^{2-1}(1,2) + A^{2-1}(2,2)\}$

$= \min\{A^1(1,2), A^1(1,2) + A^1(2,2)\}$

$= \min\{4, 4+0\}$

$= 4$

$A^2(1,3) = \min\{A^1(1,3), A^1(1,2) + A^1(2,3)\}$

$= \min\{11, 4+2\}$

$= 6$

$A^2(2,1) = \min\{A^1(2,1), A^1(2,2) + A^1(2,1)\}$

$= \min\{6, 0+6\}$

$= 6$

$A^2(2,3) = \min\{A^1(2,3), A^1(2,2) + A^1(2,3)\}$

$= \min\{2, 0+2\}$

$= 2$

$A^2(3,1) = \min\{A^1(3,1), A^1(3,2) + A^1(2,1)\}$

$= \min\{3, 7+6\}$

$= 3$

$A^2(3,2) = \min\{A^1(3,2), A^1(3,2) + A^1(2,2)\}$

$= \min\{7, 7+0\}$

$= 7$

$$A^2 = \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Take $k=3$

$A^3(i,j) = \min\{A^{3-1}(i,j), A^{3-1}(i,k) + A^{3-1}(k,j)\}$

$A^3(1,2) = \min\{A^2(1,2), A^2(1,3) + A^2(3,2)\}$

$= \min\{4, 6+7\}$

$= 4$

$A^3(1,3) = \min\{A^2(1,3),\ A^2(1,3) + A^2(3,3)\}$

$= \min\{6,\ 6+0\}$

$= 6$

$A^3(2,1) = \min\{A^2(2,1),\ A^2(2,3) + A^2(3,1)\}$

$= \min\{6,\ 2+3\}$

$= \min\{6, 5\}$

$= 5$

$A^3(2,3) = \min\{A^2(2,3),\ A^2(2,3) + A^2(3,3)\}$

$= \min\{2,\ 2+0\}$

$= 2$

$A^3(3,1) = \min(A^2(3,1),\ A^2(3,3) + A^2(3,1)\}$

$= \min\{3,\ 0+3\}$

$= 3$

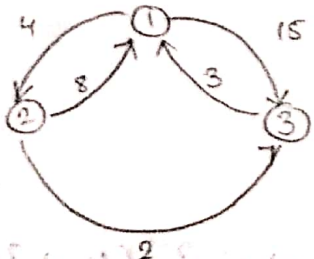
$A^3(3,2) = \min\{A^2(3,2),\ A^2(3,3) + A^2(3,2)\}$

$= \min\{7,\ 0+7\}$

$= 7$

$$A^3 = \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

4)



Take $k = 0$

$$A^0 = \begin{bmatrix} 0 & 4 & 15 \\ 8 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix}$$

Take $k = 1$

$A^1(i,j) = \min\{A^{1-1}(i,j),\ A^{1-1}(i,k) + A^{1-1}(k,j)\}$

$A^1(1,2) = \min\{A^0(1,2), A^0(1,1) + A^0(1,2)\}$

$= \min\{4, 0+4\}$

$= 4$

$A^1(1,3) = \min\{A^0(1,3), A^0(1,1) + A^0(1,3)\}$

$= \min\{15, 0+15\}$

$= 15$

$A^1(2,1) = \min\{A^0(2,1), A^0(2,1) + A^0(1,1)\}$

$= \min\{8, 8+0\}$

$= 8$

$A^1(2,3) = \min\{A^0(2,3), A^0(2,1) + A^0(1,3)\}$

$= \min\{2, 8+15\}$

$= \min\{2, 23\}$

$= 2$

$A^1(3,1) = \min\{A^0(3,1), A^0(3,1) + A^0(1,1)\}$

$= \min\{3, 3+0\}$

$= 3$

$A^1(3,2) = \min\{A^0(3,2), A^0(3,1) + A^0(1,2)\}$

$= \min\{\infty, 3+4\}$

$= 7$

$$A^1 = \begin{bmatrix} 0 & 4 & 15 \\ 8 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Take $k=2$

$A^2 = \min\{A^{2-1}(i,j), A^{2-1}(i,k) + A^{2-1}(k,j)\}$

$A^2(1,2) = \min\{A^1(1,2), A^1(1,2) + A^1(2,2)\}$

$= \min\{4, 4+0\}$

$= 4$

$A^2(1,3) = \min\{A^1(1,3), A^1(1,2) + A^1(2,3)\}$

$= \min\{15, 4+2\}$

$= 6$

$A^2(2,1) = \min\{A^1(2,1), A^1(2,2) + A^1(2,1)\}$

$= \min\{8, 0+2\}$

$= 8$

$A^2(2,3) = \min\{A^1(2,3), A^1(2,2) + A^1(2,3)\}$

$= \min\{2, 0+2\}$

$= 2$

$A^2(3,1) = \min\{A^1(3,1), A^1(3,2) + A^1(2,3)\}$

$= \min\{3, 7+8\}$

$= 3$

$A^2(3,2) = \min\{A^1(3,2), A^1(3,2) + A^1(2,2)\}$

$= \min\{7, 7+0\}$

$= 7$

$$A^2 = \begin{bmatrix} 0 & 4 & 6 \\ 8 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Algorithm:-

```
Algorithm AllPairsshortest (cost, A, n)
// cost[1:n, 1:n] is the cost adjacency matrix of a
   graph with n vertices.
// A[i,j] is the cost of a shortest path from vertex i
   to vertex j.
// cost[i,i] = 0 for 1 ≤ i ≤ n
{
    for i:=1 to n do
      for j:=1 to n do
          A[i,j] = cost[i,j];          } // cost adjacency matrix

    for k:=1 to n do
      for i:=1 to n do
        for j:=1 to n do
            A[i,j] := min{A(i,j), A[i,k] + A[k,j]};
}
```

Time complexity:-

→ In the above algorithm the first two for loops are used for construction of cost adjacency matrix is $O(n^2)$

→ The next 3 for loops are used for finding out the shortest path between every pair of vertices that take is $O(n^3)$

∴ The time complexity of all pairs shortest path is

$$= \{O(n^2) + O(n^3)$$

$$= O(n^3)$$

Multistage graph:-

A multistage graph $G = (V, E)$ is a directed weighted graph. In this graph, all the vertices are partitioned into k stages where $k \geq 2$.

In multistage graph problem, we have to findout the shortest pathfrom source to sink.

The cost of the path is calculated by using weights given along the edges. The cost of a path from source (s) to sink (T) is the sum of the cost of the edges on the path.

→ In multistage graph problem, we have to find out the shortest path from s to T. There is a set of vertices in each stage.

→ Edges are connecting vertices from one stage to next stage. starting stage contains only one vertex is called source vertex (s) and ending stage contains only one vertex is called sink (T) vertex).

→ Various paths from source to sink, now we have to select the optimal path.

The multistage graph problem can be solved by using two approaches.

1) Forward approach.
2) Backward approach.

Forward approach:-

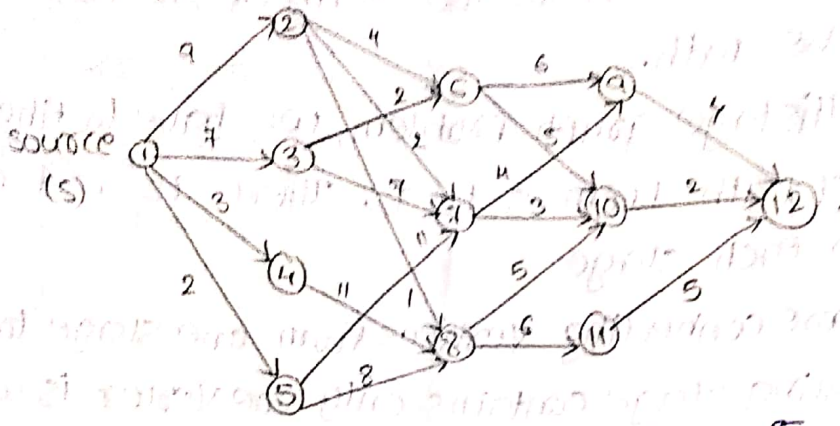
Let c(i,j) be a minimum cost path from vertex j in stage i to vertex sink. Let cost(i,j) be the cost of its path, then

$$cost(i,j) = \min_{\substack{(j,l) \in E \\ l \in V_{i+1}}} \{ c(j,l) + cost(i+1,l) \}$$

where <j,l> is the cost to reach a node l in the level i+1 D(i,j) be the value of l that minimi-ses c(j,l) and cost(i+1, l)

$$D(i,j) = \min \{ c(j,l) + cost(i+1,l) \}$$

1) Find out the minimum shortest path for the following graph using forward approach.



stages:- 1 2 3 4 5

5th stage:-

$cost(5,12) = 0$

4th stage:-

$cost(4,9) = c(9,12) = 4$

$cost(4,10) = c(j,l) = c(10,12) = 2$

$cost(4,11) = c(j,l) = c(11,12) = 5$

3rd stage:-

$$cost(3,6) = \min \begin{Bmatrix} c(6,9) + cost(4,9) \\ c(6,10) + cost(4,10) \end{Bmatrix}$$

$= \min\{6+4, 5+2\}$

$= 7$

$D(3,6) = 10$

$$cost(3,7) = \min \begin{Bmatrix} c(7,9) + cost(4,9) \\ c(7,10) + cost(4,10) \end{Bmatrix}$$

$= \min\{4+4, 3+2\}$

$= 5$

$D(3,7) = 10$

$$cost(3,8) = \min \begin{Bmatrix} c(8,10) + cost(4,10) \\ c(8,11) + cost(4,11) \end{Bmatrix}$$

$= \min\{5+2, 6+5\}$

$= 7$

$D(3,8) = 10$

2nd stage:-

$$cost(2,2) = \min\begin{Bmatrix} c(2,6) + cost(3,6) \\ c(2,7) + cost(3,7) \\ c(2,8) + cost(3,8) \end{Bmatrix}$$

$$= \min\{4+7, 2+5, 1+7\}$$

$$= 7$$

$D(2,2) = 7$

$$cost(2,3) = \min\begin{Bmatrix} c(3,6) + cost(3,6) \\ c(3,7) + cost(3,7) \end{Bmatrix}$$

$$= \min\{2+7, 7+5\}$$

$$= 9$$

$D(2,3) = 6$

$$cost(2,4) = \min\{c(4,8) + cost(3,8)\}$$

$$= \min\{11+7\}$$

$$= 18$$

$D(2,4) = 18$

$$cost(2,5) = \min\begin{Bmatrix} c(5,7) + cost(3,7) \\ c(5,8) + cost(3,8) \end{Bmatrix}$$

$$= \min\{11+5, 8+7\}$$

$$= 15$$

1st stage:-

$$cost(1,1) = \min\begin{Bmatrix} c(1,2) + cost(2,2) \\ c(1,3) + cost(2,3) \\ c(1,4) + cost(2,4) \\ c(1,5) + cost(2,5) \end{Bmatrix}$$

$$= \min\{9+7, 7+9, 3+18, 2+15\}$$

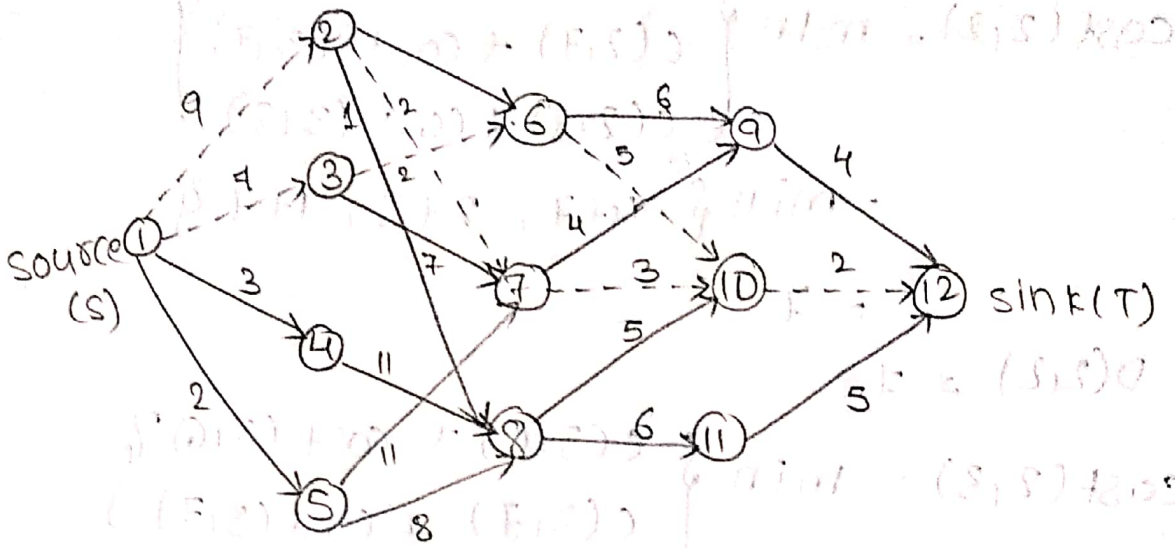
$$= \min\{16, 16, 21, 17\}$$

$$= 16$$

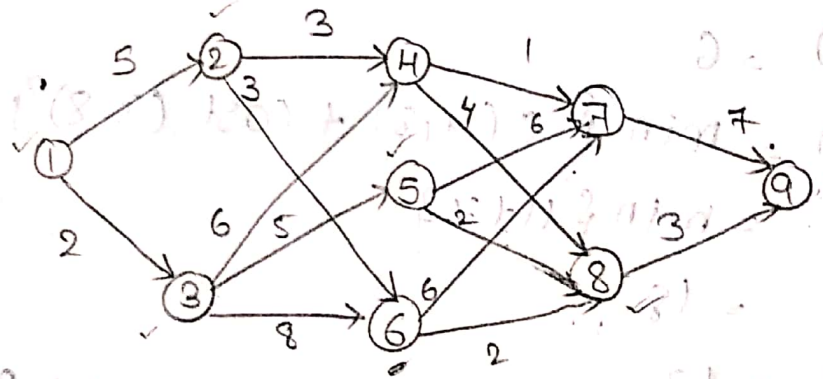
$D(1,1) = 2$, $D(1,1) = 3$

Shortest path

$D(1,1) = 2$	$D(1,1) = 3$
$D(2,2) = 7$	$D(2,3) = 6$
$D(3,7) = 10$	$D(3,6) = 10$
$D(4,10) = 12$	$D(4,10) = 12$
$1-2-7-10-12 = 16$	$1-3-6-10-12 = 16$



2) Find the minimum shortest paths for the following multistage graph using forward approach.



$\boxed{3 + 2 + 5 + 2}$

Backward Approach:-

Let c(i,j) be a minimum cost path from vertex j in stage i to vertex source. Let cost(i,j) be the cost of its path then

$$bcost(i,j) = \min\{bcost(i-1,l) + c(l,j)\}$$

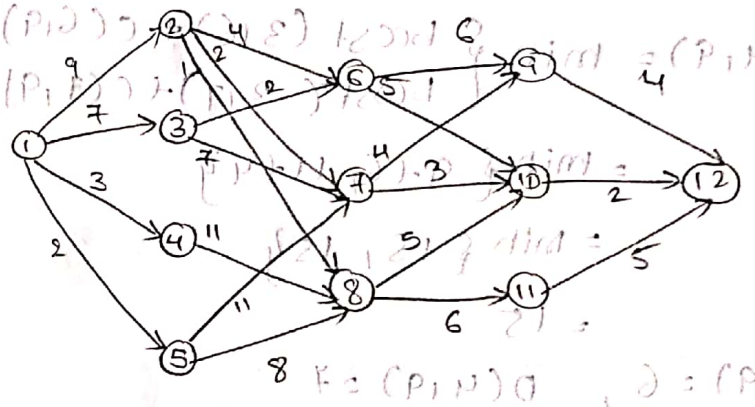
$$l \in V_{i-1}$$

$$(l,j) \in E$$

where

(l,j) is an edge cost to reach a node j in the level $i-1$. $D(i,j) = l$.

Find out the minimum shortest paths for the following multistage graph using backward approach.



Stages:- 1 2 3 4 5

1st stage:-

$bcost(1,1) = 0$

2nd stage:-

$bcost(2,2) = c(i,j) = c(1,2) = 9$

$bcost(2,3) = c(1,3) = 7$

$bcost(2,4) = c(1,4) = 3$

$bcost(2,5) = c(1,5) = 2$

3rd stage:-

$$bcost(3,6) = \min\{bcost(2,2) + c(2,6),\ bcost(2,3) + c(3,6)\}$$

$$= \min\{9+4,\ 7+2\}$$

$$= 9$$

$D(3,6) = 3$

$bcost(3,7) = \min \begin{cases} bcost(2,2) + c(2,7) \\ bcost(2,3) + c(3,7) \\ bcost(2,5) + c(5,7 \end{cases}$

$= \min\{9+2, 7+7, 2+11\}$

$= 11$

$D(3,7) = 2$

$bcost(3,8) = \min \begin{cases} bcost(2,2) + c(2,8) \\ bcost(2,4) + c(4,8) \\ bcost(2,5) + c(5,8) \end{cases}$

$= \min\{9+1, 3+11, 2+8\}$

$= 10$

$D(3,8) = 2, D(3,8) = 5$

4th stage:-

$bcost(4,9) = \min \begin{cases} bcost(3,6) + c(6,9) \\ bcost(3,7) + c(7,9) \end{cases}$

$= \min\{9+6, 11+4\}$

$= \min\{15, 15\}$

$= 15$

$D(4,9) = 6, \quad D(4,9) = 7$

$bcost(4,10) = \min \begin{cases} bcost(3,6) + c(6,10) \\ bcost(3,7) + c(7,10) \\ bcost(3,8) + c(8,10) \end{cases}$

$= \min\{9+5, 11+3, 10+5\}$

$= 14$

$D(4,10) = 6, \quad D(4,10) = 7$

$bcost(4,11) = \min\{bcost(3,8) + c(8,11)\}$

$= \min\{10+6\}$

$= 16$

$D(4,11) = 8$

5th stage:-

$bcost(5,12) = \min \begin{cases} bcost(4,9) + c(9,12) \\ bcost(4,10) + c(10,12) \\ bcost(4,11) + c(11,12) \end{cases}$

$= \min\{15+4, 14+2, 16+5\}$

$= \min\{19, 16, 21\}$

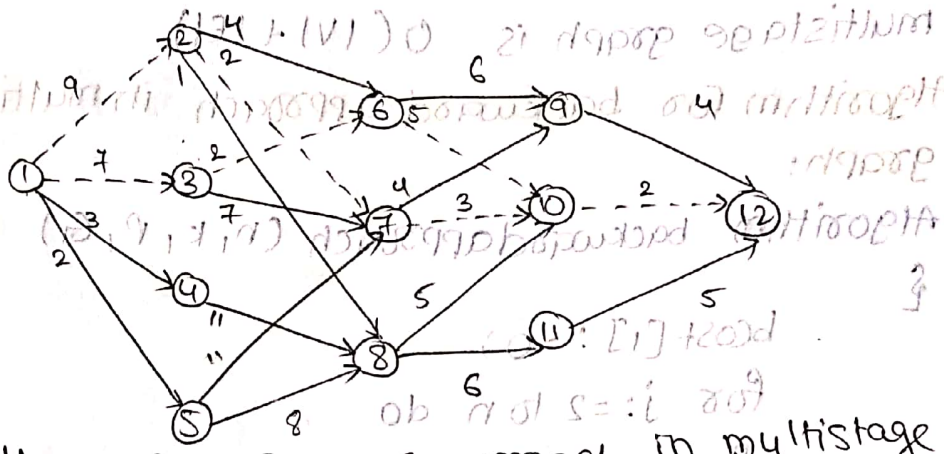
$= 16$

$D(5,12) = 10$

shortest Path:

$D(5,12) = 10$	$D(5,12) = 10$
$D(4,10) = 6$	$D(4,10) = 7$
$D(3,6) = 3$	$D(3,7) = 2$
$D(2,3) = 1$	$D(2,2) = 1$
$1-3-6-10-12$	$1-2-7-10-12$

∴ shortest paths are $1-3-6-10-12$, $1-2-7-10-12$



* Algorithm for forward approach in multistage graph:-

```
Algorithm Forwardapproach(n, k, p, G)
// The input for the multistage graph is a k-stage graph
G(V,E) with n vertices indexed in order of stages
E is a set of edges and c[i,j] is the cost(i,j).
P[1:k] is a minimum cost path
{
    cost[n] := 0;
    for j := n-1 to 1 do
    {
        // compute cost[j]
        // let l be a vertex such that <j,l> is
           an edge of G and c[j,l]+cost[l]
```

is minimum

```
        cost[j] := c[j, r] + cost(r);
        d[j] := r;
    }
    // Find a minimum cost path
    p[1] := 1;
    p[k] := n;
    for j := 2 to k-1 do
        p[j] := d[p[j-1]];
}
```

Time complexity:

The time complexity of forward approach in multistage graph is $O(|V| + |E|)$

* Algorithm for backward approach in multistage graph:

```
Algorithm backwardapproach (n, k, p, G)
{
    bcost[1] := 0;
    for j := 2 to n do
    {
        // compute bcost[j]
        // Let r be a vertex such that <r, j> is an
        edge of G and bcost(r) + c(r, j) is
        minimum.
        bcost[j] := c(r, j) + bcost(r);
        d[j] := r;
    }
    // find a minimum cost path
    p[1] := 1;
    p[k] := n;
    for j := k-1 to 2 do
        p[j] := d[p[j+1]];
}
```

Time complexity :-

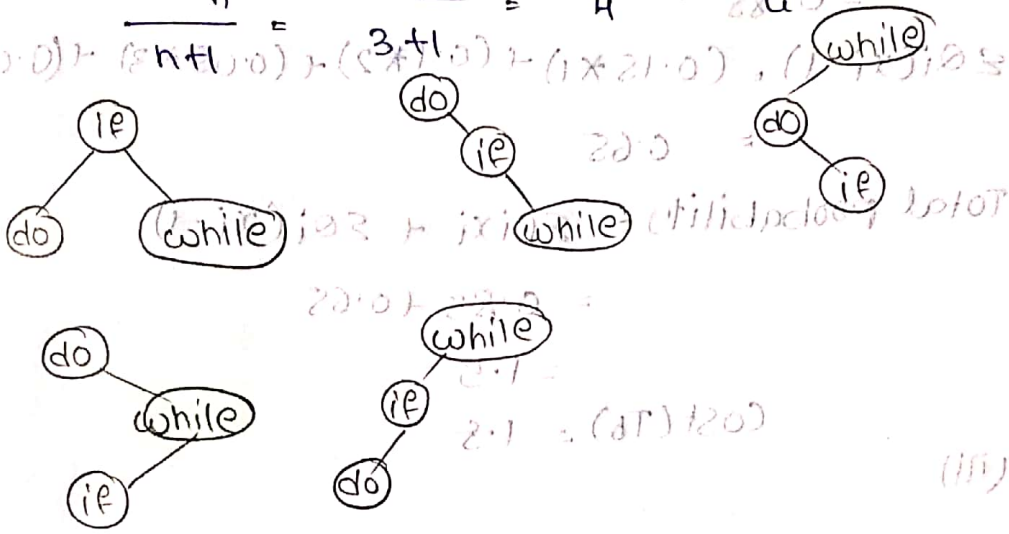
The time complexity of backward approach in multistage graph is $O(|V|+|E|)$

Optimal Binary search Tree (OBST) :-

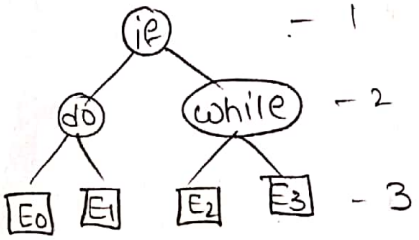
Find out optimal binary search tree for a set $(a_1, a_2, a_3) = (do, if, while)$ $(P_1, P_2, P_3) = (0.5, 0.1, 0.05)$ $(Q_0, Q_1, Q_2, Q_3) = (0.15, 0.1, 0.05, 0.05)$

The possible number of binary search tree with $n = 3$ is

$$\frac{2nC_n}{n+1} = \frac{2*3C_3}{3+1} = \frac{6C_3}{4} = \frac{20}{4} = 5$$



i)



$P(1) = 0.1*1 + 0.5*2 + 0.05*2$

$= 0.1 + 1 + 0.1$

$= 1.2$

$Q(1) = 0.15*2 + 0.1*2 + 0.05*2 + 0.05*2$

$= 0.7$

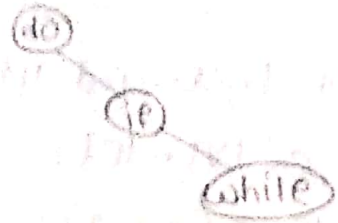
total probability $= \sum P_i x_i + \sum Q_i (x_i - 1)$

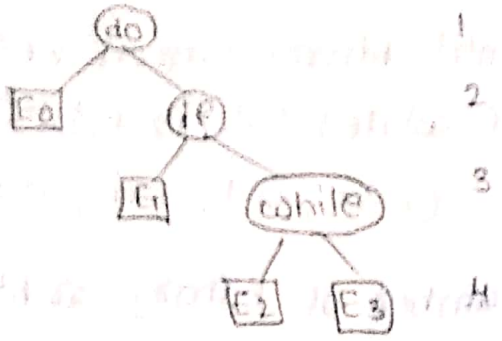
$= 1.2 + 0.7$

$= 1.9$

$Cost(T_a) = 1.9$

(ii)





$\Sigma Pixi = (0.5*1) + (0.1*2) + (0.05*3)$

$= 0.85$

$\Sigma Qi(xi-1) = (0.15*1) + (0.1*2) + (0.05*3) + (0.05*3)$

$= 0.65$

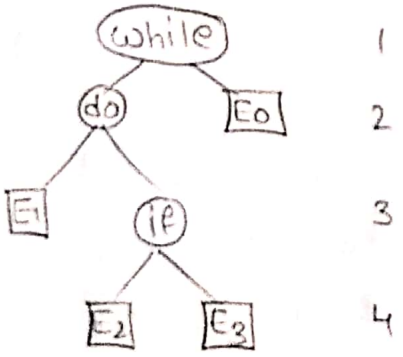
Total probability = $\Sigma Pixi + \Sigma Qi(xi-1)$

$= 0.85 + 0.65$

$= 1.5$

$Cost(T6) = 1.5$

(iii)



$\Sigma Pixi = (0.05*1) + (0.5*2) + (0.1*3)$

$= 1.35$

$\Sigma Qi(xi-1) = (0.15*1) + (0.1*2) + (0.05*3) + (0.05*3)$

$= 0.65$

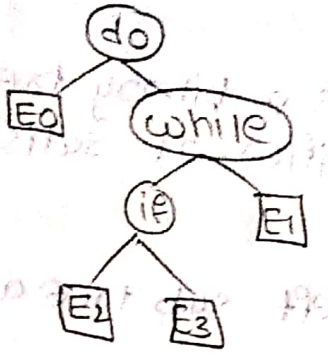
Total probability = $\Sigma Pixi + \Sigma Qi(xi-1)$

$= 1.35 + 0.65$

$= 2$

$Cost(T6) = 2$

(iv)



$\Sigma Pixi = (0.5*1) + (0.05*2) + (0.1*3)$

$= 0.9$

$\Sigma Qi(xi-1) = (0.15*1) + (0.1*2) + (0.05*3) + (0.05*3)$

$= 0.65$

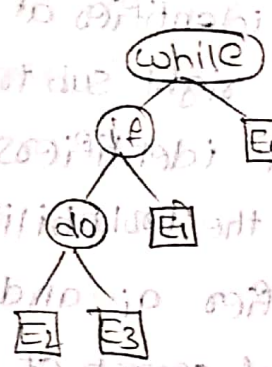
Total probability $= \Sigma Pixi + \Sigma Qi(xi-1)$

$= 0.9 + 0.65$

$= 1.55$

$Cost(T_4) = 1.55$

(v)



$\Sigma Pixi = (0.05*1) + (0.1*2) + (0.5*3)$

$= 1.75$

$\Sigma Qi(xi-1) = (0.15*1) + (0.1*2) + (0.05*3) + (0.05*3)$

$= 0.65$

Total probability $= \Sigma Pixi + \Sigma Qi(xi-1)$

$= 1.75 + 0.65$

$= 2.4$ $Cost(T_5) = 2.4$

$Cost(T_1) = 1.9$

$Cost(T_2) = 1.5$

$Cost(T_3) = 2$

$Cost(T_4) = 1.55$

$Cost(T_5) = 2.4$

Among all the 5 binary search trees tree-2 is optimal tree with cost = 1.5

Theory:-

A binary search tree is a binary tree in which each node is an identifier. It satisfies the following properties:-

1) All identifiers in left sub tree are less than identifier at root node.

2) All identifiers in right sub tree are greater than the identifier at root node.

3) To find whether an identifier x is present or not using the following steps

(i) x is compared with the root.

(ii) If x is less than identifier at root node, then search continuously in left sub tree.

(iii) If x is greater than identifier at root node, then search continuous in right sub tree.

→ Let us assume given set of identifiers $\{a_1, a_2, a_3 \ldots a_n\}$ and let P(i) be the probability of successful search of an identifier a_i and Q(i) be the probability of unsuccessful search of an identifier a_i.

$\therefore \sum P(i) + \sum Q(i) = 1$

→ To obtain a cost function for binary search tree, it is useful to add an imaginary node or empty node.

→ If n identifiers are there, there will be 'n' internal nodes and n+1 external nodes. Every external node represents an unsuccessful search.

→ If a successful search terminates at an internal node at level l. Hence, the expected cost contribution from the internal node for identifier a_i is

P(i) * level(ai)

→ unsuccessful search terminates at an external node at level i. Hence, the expected cost contribution for the external node is Q(i) * level((Ei)-1)

→ The expected cost of a binary search tree is

$$\Sigma(P(i) * level(a_i)) + \Sigma(Q(i) * level(E_i - 1))$$

where P(i) is the probability of successful search and Q(i) is the probability of unsuccessful search

Definition of OBST:-

→ For a given set of identifiers, we have different binary search trees. Among all those which requires least number of comparisons for searching a particular identifier called OBST (or) optimal Binary search Tree.

→ Let $a_1, a_2, a_3, \ldots, a_n$ are different identifiers with $a_1 < a_2 < a_3 < \ldots < a_n$.

→ successful search probabilities P(i), $1 \le i \le n$

→ unsuccessful search probabilities Q(i), $0 \le i \le n$

→ For the given set of n identifiers we have to construct the optimal binary search tree T_{on}, where T_{on} is the optimal binary search tree

→ Wij is the weight of each Tij

Initial condition:-

$$C(i,i) = 0$$

$$r(i,i) = 0$$

$$W(i,i) = q_i \text{ (or) } q(i)$$

Add node by node to the BST by calculating

$$C(i,j) = \min_{i<k\le j} \{C(i,k-1) + C(k,j)\} + w(i,j)$$

$w(i,j) = w(i,j-1) + p(j) + q(j)$

$r(i,j) = k$.

1) Consider $n = 4$ and $(q_1, q_2, q_3, q_4) = (do, if, int, while)$ $(p_1, p_2, p_3, p_4) = (3, 3, 1, 1)$, $q(0-4) = (2, 3, 1, 1, 1)$

Construct the OBST.

We have to construct the OBST $T_{0n} = T_{04}$

	0	1	2	3	4
$\|j-i\|=0$	0	$w(0,0)=2$ $c(0,0)=0$ $r(0,0)=0$	$w(1,1)=3$ $c(1,1)=0$ $r(1,1)=0$	$w(2,2)=1$ $c(2,2)=0$ $r(2,2)=0$	$w(3,3)=1$ $c(3,3)=0$ $r(3,3)=0$	$w(4,4)=1$ $c(4,4)=0$ $r(4,4)=0$
$\|j-i\|=1$	1	$w(0,1)=8$ $c(0,1)=8$ $r(0,1)=1$	$w(1,2)=7$ $c(1,2)=7$ $r(1,2)=2$	$w(2,3)=3$ $c(2,3)=3$ $r(2,3)=3$	$w(3,4)=3$ $c(3,4)=3$ $r(3,4)=4$	
$\|j-i\|=2$	2	$w(0,2)=12$ $c(0,2)=19$ $r(0,2)=1$	$w(1,3)=9$ $c(1,3)=12$ $r(1,3)=2$	$w(2,4)=5$ $c(2,4)=8$ $r(2,4)=3,4$		
$\|j-i\|=3$	3	$w(0,3)=14$ $c(0,3)=25$ $r(0,3)=2$	$w(1,4)=11$ $c(1,4)=19$ $r(1,4)=2$			
$\|j-i\|=4$	4	$w(0,4)=2$ $c(0,4)=32$ $r(0,4)=16$				

Step-1:- $|j-i| = 0$

$c(i,i) = 0$

$r(i,i) = 0$

$w(i,i) = q(i)$

$c(0,0) = 0$

$r(0,0) = 0$

$w(0,0) = q(0) = 2$

$c(1,1) = 0$

$r(1,1) = 0$

$w(1,1) = q(1) = 3$

$c(2,2) = 0$

$r(2,2) = 0$

$w(2,2) = q(2) = 1$

$c(3,3) = 0$

$r(3,3) = 0$

$w(3,3) = q(3) = 1$

$c(4,4) = 0$

$r(4,4) = 0$

$w(4,4) = q(4) = 1$

Step-2:- $|j-i| = 1$

$\rightarrow w(0,1) = w(0,0) + P(1) + Q(1)$

$= 2+3+3$

$= 8$

$c(0,1) = \min_{0<k\leq 1} \{c(0,1-1) + c(1,1)\} + w(0,1)$

$= \min_{k=1} \{c(0,0) + c(1,1) + 8$

$= \min \{0+0\} + 8$

$= 8$

$r(0,1) = 1$

$\rightarrow w(1,2) = w(1,1) + P(2) + Q(2) = 3+3+1 = 7$

$c(1,2) = \min_{1<k\leq 2} \{c(1,2-1) + c(2,2)\} + w(1,2)$

$= \min_{k=2} \{c(1,1) + c(2,2)\} + 7$

$= \min_{k=2} \{0+0\} + 7$

$= 7.$

$r(1,2) = 2$

$\rightarrow w(\overset{i}{2},\overset{j}{3}) = w(2,2) + P(3) + Q(3) = 1+1+1 = 3$

$c(2,3) = \min_{2<k\leq 3} \{c(2,2) + c(3,3)\} + w(2,3)$

$= \min_{k=3} \{0+0\} + 3$

$= 3$

$r(2,3) = 3$

$\rightarrow w(3,4) = w(3,3) + P(4) + Q(4) = 1+1+1 = 3$

$c(3,4) = \min_{3<k\leq 4} \{c(3,3) + c(4,4)\} + w(3,4)$

$= \min_{k=4} \{0+0\} + 3 = 3$

$r(3,4) = 4$

Step-3:-

$|j-i| = 2$

$\rightarrow w[0,2] = w(0,1) + p(2) + q(2) = 8+3+1 = 12$

$$c(0,2) = \min_{0<k\le 2} \begin{Bmatrix} c(0,0)+c(1,2) \\ c(0,1)+c(2,2) \end{Bmatrix} + w(0,2)$$

$k = 1, 2$

$$= \min \{ 0+7, 8+0 \} + 12$$

$$= \min \{ 7, 8 \} + 12$$

$$= \overset{k=1}{7} + 12$$

$$= 19$$

$r(0,2) = 1$

$\rightarrow w(1,3) = w(1,2) + p(3) + q(3) = 7+1+1 = 9$

$$c(1,3) = \min_{1<k\le 3} \begin{Bmatrix} c(1,1)+c(2,3) \\ c(1,2)+c(3,3) \end{Bmatrix} + w(1,3)$$

$k = 2$
$k = 3$

$$= \min \{ \overset{k=2}{0+3}, \overset{k=3}{7+0} \} + 9$$

$$= 3+9$$

$$= 12$$

$r(1,3) = 2$

$\rightarrow w(2,4) = w(2,3) + p(4) + q(4) = 3+1+1 = 5$

$$c(2,4) = \min_{2<k\le 4} \begin{Bmatrix} c(2,2)+c(3,4) \\ c(2,3)+c(4,4) \end{Bmatrix} + w(2,4)$$

$k = 3$
$k = 4$

$$= \min \{ 0+3, 3+0 \} + 5$$

$$= \min \{ 3, 3 \} + 5$$

$$= 3+5 = 8$$

$r(2,4) = 3, 4$

step-4:-

$\Rightarrow w(0,3) = w(0,2) + P(3) + q(3) = 12+1+1 = 14$

$c(0,3) = \min_{0<k\le 3} \{c(0,0)+c(1,3),\ c(0,1)+c(2,3),\ c(0,2)+c(3,3)\} + w(0,3)$

$= \min\{0+12,\ 8+3,\ 19+0\} + 14$

$= \min\{12, 11, 19\} + 14$

$= 11 + 14$

$= 25$

$r(0,3) = 2$

$w(1,4) = w(1,3) + P(4) + q(4) = 9+1+1 = 11$

$\Rightarrow c(1,4) = \min_{1<k\le 4} \{c(1,1)+c(2,4),\ c(1,2)+c(3,4),\ c(1,3)+c(4,4)\} + w(1,4)$

$= \min\{0+8,\ 7+3,\ 12+0\} + 11$

$= \min\{8, 10, 12\} + 11$

$= 8 + 11 = 19$

$r(1,4) = 2$

~~step-5:-~~

~~$w(0,4) = w(0,3) + P(4) + q(4) = 12+1+1 = 14$~~

$w(0,4) = w(0,3) + P(4) + q(4) = 14+1+1 = 16$

$c(0,4) = \min_{0<k\le 4} \{c(0,0)+c(1,4),\ c(0,1)+c(2,4),\ c(0,2)+c(3,4),\ c(0,3)+c(4,4)\} + w(0,4)$

$$= \min \begin{Bmatrix} 0+19 \\ 8+8 \\ 19+3 \\ 25+0 \end{Bmatrix} + 16$$

$$= \min\{19, 16, 22, 25\} + 16$$

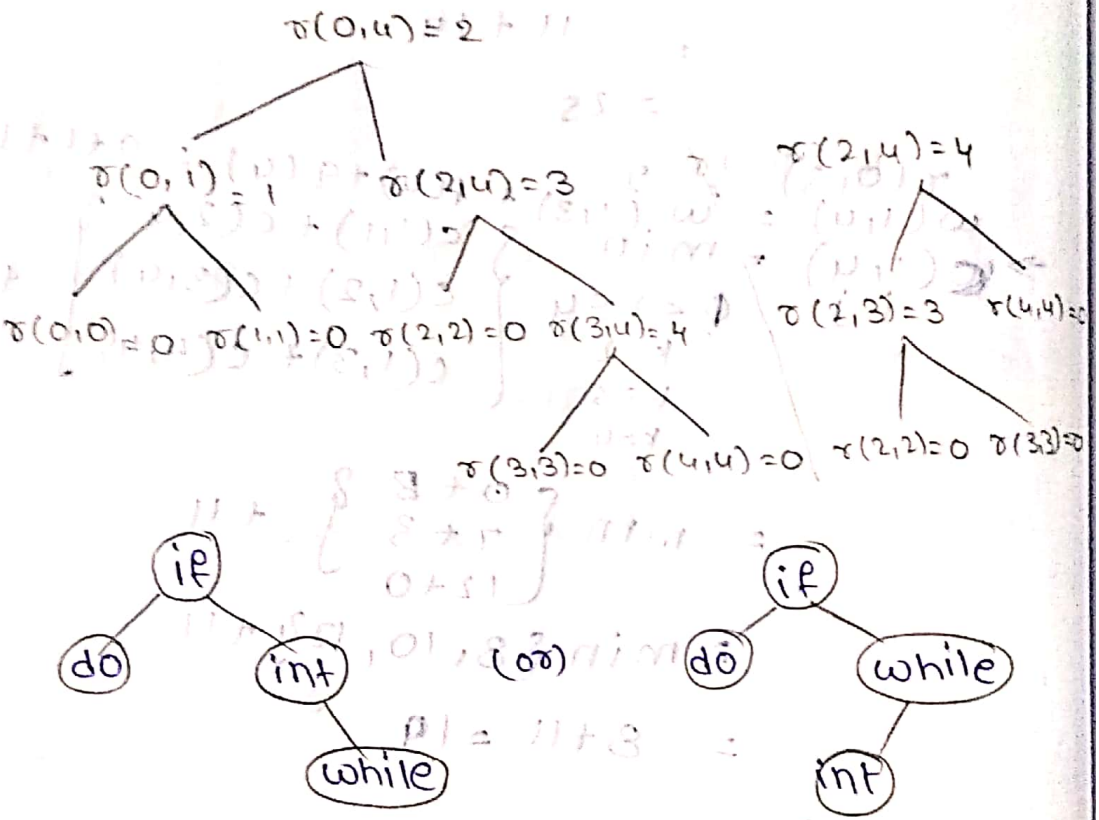
$$= 16 + 16$$

$$= 32$$

$r(0,4) = 2$

$r(i,j) = k$

$r(i,k-1) \quad r(k,j)$



using OBST Algorithm findout the optimal Binary search tree for the identifier set (a_1, a_2, a_3, a_4) = (end, goto, print, stop)

with $P(1) = \frac{1}{20}$, $P(2) = \frac{1}{5}$, $P(3) = \frac{1}{10}$, $P(4) = \frac{1}{20}$

$q(0) = \frac{1}{5}$, $q(1) = \frac{1}{10}$ $q(2) = \frac{1}{5}$ $q(3) = \frac{1}{20}$

$q(4) = \frac{1}{20}$

we will multiply the values of P's & q's by 20 (maximum) denominator.

find OBST for n=4, (q1, q2, q3, q4) = (cout, print, stop) p1 = 1/4 p2 = 1/8 p3 = 1/16 p4 = 1/16

q0 = 1/4 q1 = 3/16 q2 = 1/16 q3 = 1/16 q4 = 1/16

0/1 knapsack problem in Dynamic programming:

* we are given n objects and a knapsack in which the object i has to be placed with a weight w_i, then the profit can be earned is $p_i * x_i$. The knapsack has a capacity 'M'.
* The objective of the 0/1 knapsack problem is filling the knapsack with the objects that has a maximum profit.
* The main objective of 0/1 knapsack problem is ⇒ maximize $\sum_{i=1}^{n} p_i x_i$ subjected to the constraint $\sum_{i=1}^{n} w_i x_i \leq M$ where $1 \leq i \leq n$, $x_i = 0/1$, n is the total number of objects.
* The Greedy method doesn't work for this problem, because in Greedy method, it allows 0, 1, fraction of the objects.
* In Dynamic programming, 0/1 knapsack problem allows only two values - 0 and 1, but it doesn't allow fractions.
* Dominance rule or Purging rule

if S^{i+1} contains two pairs (P_j, w_j) and (P_k, w_k). These two pairs satisfies the two conditions $(P_j \leq P_k)$ and $(w_j \geq w_k)$, then (P_j, w_j) pair can be eliminated. This rule is called as Purging rule or Dominance rule.

In purging or dominance rule, the dominated tuples get purged. i.e; Remove the pair with less profit and more weight.

Solve the 0/1 knapsack problem instance for $n=3$ and $m=6$. Let P_i and w_i are shown below

i	P_i	w_i
1	1	2
2	2	3
3	5	4

Let us build the sequence of decisions S^0, S^1, S^2, S^3.

→ Initially * $S^0 = \{(0,0)\}$

→ while building S_1^0, we select the next (P,w) pair, this pair can be added to S^0

$$S_1^0 = \{(0+1, 0+2)\} = \{(1,2)\}$$

→ S^1 = Merge S^0 and S_1^0

* $S^1 = \{(0,0)\} \cup \{(1,2)\} = \{(0,0), (1,2)\}$

While building S_1^1, we have to select the next (P,w) pair that can be added to S^1

$$S_1^1 = \{(0+2, 0+3), (1+2, 2+3)\}$$

$$= \{(2,3), (3,5)\}$$

S^2 = merge S^1 and S_1^1

$$= \{(0,0), (1,2)\} \cup \{(2,3)(3,5)\}$$

* $S^2 = \{(0,0)(1,2)(2,3)(3,5)\}$

while building S_1^2, we have to select the next (P,w) pair, that can be added to

the order pairs in S^2

$S_1^2 = \{(0+5, 0+4), (1+5, 2+4), (2+5, 3+4), (3+5, 5+4)\}$

$= \{(5,4), (6,6), (7,7), (8,9)\}$

S^3 = Merge S^2 and S^1

$= \{(0,0), (1,2), (2,3), (3,5)\} \cup \{(5,4), (6,6), (7,7), (8,9)\}$

$S^3 = \{(0,0), (1,2), (2,3), (3,5), (5,4), (6,6), (7,7), (8,9)\}$

The order pairs (7,7) and (8,9) are removed for S^3 because the weight coordinates in the order pair greater than the capacity of the knapsack(m=6). Hence, these two pairs are eliminated

purging rule :- (3,5) (P_j w_j), (5,4) (P_k w_k); $P_j \leq P_k$, $w_j \geq w_k$ && $3 \leq 5$ && $5 \geq 4$

$S^3 = \{(0,0), (1,2), (2,3), (5,4), (6,6)\}$ eliminate ($P_j w_j$) pair i.e; (3,5)

As m=6, we will find the tuple denoting the weight 6. It is (6,6)

$(6,6) \in S^3$

$(6,6) \notin S^{3-1}$ $\therefore x_3 = 1$

$(6-P_3, 6-w_3) = (6-5, 6-4) = (1,2) \in S^2$

$= (1,2) \in S^{2-1}$ $\therefore x_2 = 0$

$(1,2) \in S^1$

$(1,2) \notin S^{1-1}$ $\therefore x_1 = 1$

$\therefore$ solution vector $x[1:3] = \{1, 0, 1\}$

2) Findout the optimal solution for 0/1 knapsack problem instance, n=3, m=6

$(P_1, P_2, P_3) = (1, 2, 4)$ $(w_1, w_2, w_3) = (2, 3, 3)$

let us build the seaquence S^0, S^1, S^2, S^3

Initially $S^0 = \{(0,0)\}$

while building S_1^0 we select the next (P,W) pair this pair can be added to S^0

$S_1^0 = \{(0+1, 0+2)\} = (1,2)$

S^1 = Merge S^0 and S_1^0

$= \{(0,0)\} \cup \{(1,2)\}$

$S^1 = \{(0,0)(1,2)\}$

while building S_1^1 we have to select the next (P,W) pair that can be added to the order pairs of S^1.

$S_1^1 = \{(0+2, 0+3), (1+2, 2+3)\}$

$= \{(2,3)(3,5)\}$

S^2 = Merge S^1 and S_1^1

$= \{(0,0)(1,2)\} \cup \{(2,3)(3,5)\}$

$= \{(0,0)(1,2)(2,3)(3,5)\}$

while building S_2^1 we have to select the next (P,W) pair that can be added to the order pairs of S^2.

$S_2^1 = \{(0+4, 0+3)(1+4, 2+3)(2+4, 3+3)(3+4, 5+3)\}$

$S_2^1 = \{(4,3)(5,5)(6,6)(7,8)\}$

S^3 = Merge S^2 and S_2^1

$= \{(0,0)(1,2)(2,3)(3,5)\} \cup \{(4,3)(5,5)(6,6)(7,8)\}$

$S^3 = \{(0,0)(1,2)(2,3)(3,5)(4,3)(5,5)(6,6)(7,8)\}$

The order pair (7,8) is removed for S^3 because the weight coordinates in the order

pair greater than the capacity of the knapsack (m=6). Hence it is eliminated

$S^3 = \{(0,0)(1,2)(2,3)(3,5)(4,3)(5,5)(6,6)\}$

purging rule $(3,5)$ $(4,3)$ (P_j w_j, P_k w_k)

$P_j \leq P_k$ and $w_j \geq w_k$

eliminate $(P_j w_j)$ pair i.e $(3,5)$

$S^3 = \{(0,0)\ (1,2)(2,3)(4,3)(5,5)(6,6)\}$

As m=6, we will find the tuple denoting the weight 6. It is (6,6)

$(6,6) \in S^3$

$(6,6) \notin S^{3-1}$ $\therefore x_3 = 1$

$(6-P_3, 6-w_3) = (6-4, 6-3) = (2,3) \in S^2$

$(2,3) \notin S^{2-1}$

$\therefore x_2 = 1$

$(2-P_2, 3-w_2) = (2-2, 3-3)$

$= (0,0) \in S^1$

$= (0,0) \in S^0$ $\therefore x_1 = 0$

solution vector $x[1:3] = \{0,1,1\}$

3) Solve the following 0/1 knapsack problem using dynamic programming $n=4$, $m=40$
$(P_1-P_4) = (11, 21, 31, 33)$ $(w_1-w_4) = (2, 11, 22, 15)$

4) Find out the optimal solution for 0/1 knapsack problem for $n=4$, $m=30$ $(w_1-w_4) = (10, 15, 6, 9)$
$(P_1-P_4) = (2, 5, 8, 1)$

Let us build the sequence of decisions
s^0, s^1, s^2, s^3, s^4

Initially $s^0 = \{(0,0)\}$

while building s_1^0 we select the next (P,w) pair this pair can be added to s^0

$s_1^0 = \{(0+11, 0+2)\}$
$= \{(11, 2)\}$

s^1 = merge of s^0 and s_1^0
$= \{(0,0)(11,2)\}$

while building s_1^1 we select the next (P,w) pair this pair can be added to s^1

$s_1^1 = \{(0+21, 0+11)(11+21, 2+11)\}$

$s_1^1 = \{(21, 11)(32, 13)\}$

s^2 = merge of s^1 & s_1^1

$= \{(0,0)(11,2)(21,11)(32,13)\}$

while building s_2^2 we select the next (P,w) pair this pair can be added to s^2

$s_2^1 = \{(0+31, 0+22)(11+31, 2+22)(21+31, 11+22)$
$(32+31, 13+22)\}$

$s_2^1 = \{(31,22)(42,24)(52,33)(63, 35)\}$

s^3 = Merge of s^1 and s^2

s^3 = {(0,0) (11,1) (21,11) (32,13) (31,22)
(42,24) (52,33) (63,35)}

while building s^3, we select the next (r,w) pair this pair can be added to s^3

s^3_1 = {(0+33, 0+15) (11+33, 2+15) (21+33, 11+15)
(32+33, 13+15) (31+33, 22+15)
(42+33, 24+15) (52+33, 33+15) (63+33, 35+15)}

s^3_1 = {(33,15) (44,17) (54,26) (65, 28)
(64,37) (75, 39) (85, 48) (96,50)}

s^4 = Merge s^3 and s^3_1

= {(0,0) (11,2) (21,11) (32,13) (31,22) (42,24)
(52,33) (63,35) (33,15) (44,17) (54,26)
(65,28) (64,37) (75,39) (85,48) (96,50)}

the order pairs, (85,48) (96,50) are removed

s^4 = {(0,0) (11,2) (21,11) (32,13) (31,22) (42,24)
(52,33) (63,35) (33,15) (44,17) (54,26)
(65,28) (64,37) (75,39)}

= {(0,0) (11,2) (21,11) (31,22) (32,13) (33,15) (42,24)
(44,17) (52,33) (54,26) (63,35) (63,37)
(65,28) (75,39)}

= {(0,0) (11,2) (21,11) (32,13) (33,15) (44,17)
(54,26) (63,35) (65,28) (75,39)}

s^4 = {(0,0) (11,2) (21,11) (32,13) (33,15) (44,17)
(54,26) (65,28) (75,39)}

As m=40 we denote the tuple containing

General method:-

Dynamic programming is an algorithm design method that solves a given complex problem by breaking down into subproblems, solving each of these subproblems just once and storing their result in an array.

Dynamic programming follows a principle called "principle of optimality".

Principle of optimality states that an optimal sequence of decisions has the property that whatever the initial state and decisions are, but the remaining desicions must constitute an optimal decision sequence with regard to the state resulting from the first decision.

Two properties of a problem that suggests

the given problem can be solved using dynamic programming.

1. optimal substructure
2. overlaping subproblems.

optimal substructure:-

A given problem has optimal substructure property, when the optimal solution of the given problem can be obtained by using optimal solution of its subproblems.

Ex:- The shortest path problem has the optimal substructure property.

if a node x, lies in the shortest path from source node u to a destination node v. The shortest path from u to v is divided into two sub-problems. They are:

1) The shortest path from u to x
2) The shortest path from x to v.

overlaping subproblems:-

Like divide & conquer, dynamic programming combines the solutions of the subproblems.

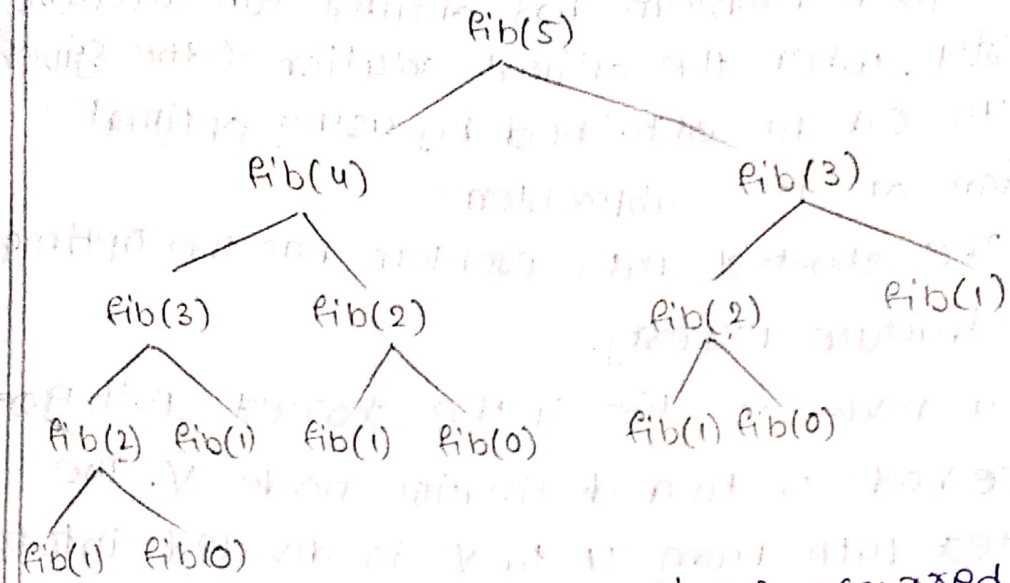
Dynamic programming is mainly used when the solutions of the same (overlaping) sub problems are needed again and again.

The computed solutions of sub problems are stored in a table, so that these subproblems do not have to be re computed

Ex:- If we consider the recursive program for fibonacci numbers, there are many sub-problems which can be solved again and again

```
int fib(int n)
{
```

```
if (n<=1) then
    return n;
else
    return fib(n-1)+fib(n-2);
}
```



In the above function fib(0) occurred 3 times fib(1) occurred 5 times, fib(2) occurred 3 times fib(3) occurred 2 times.

→ Instead of calculating fib(1) 5 times, fib(2) 3 times, fib(3) times, fib(0) 3 times these functions are calculated only once, what ever the result we are getting from these results, that results are stored in a table, next time the same function is repeated we have to use the fuction stored in the table without re-calculating that.

There are 2 different ways to store the values in a table so that, the values can be reused in the problem.

1. Tabulation.
2. Memorization

Tabulation:-

For a given problem, this method builds a table in bottom-up fashion and returns the last entry from the table.

Memorization:-

This method follows top-down approach. In this method, we initialize a table with initial values as -1. whenever we need the solution to a subproblem, we first look into the table. If the precomputed value is there, then we return that value. otherwise, we calculate that value and put the result in the table so that it can be reused later.

Differences blw divide & conquer and dynamic programming.

Divide & conquer.	Dynamic programming
1) In this technique, the given problem is divided into small sub problems. These sub problems are solved independently. Finally, all the solutions of sub problems are collected together to get the final solution to the given problem.	1) In dynamic programming, many decision sequences are generated and all the overlapping sub instances are considered.
2) In this method, duplication in sub problems may be obtained.	2) In this method, duplications in solutions of sub-problems is avoided totally.
3) Divide and conquer is less efficient because of recomputations on solutions. (duplications)	3) Dynamic programming is efficient than divide & conquer strategy.
4) Divide and conquer uses top-down approach of problem solving.	4) Dynamic programming uses bottom-up approach of problem solving.

5) Divide & conquer splits its input at specific deterministic point usually in the middle.	5) Dynamic programming splits its input for every possible split point rather than at specific point. After trying all split points, it determines the optimal split point.
6) Applications.	

* Differences b/w greedy method & Dynamic programming :-

Greedy Method.	Dynamic programming.
1) It is used for obtaining optimal solution	1) It is also used for obtaining optimal solution to the given problem.
2) In Greedy method, a set of feasible solutions are generated, among them, we pickup the optimal solution.	2) There is no special set of feasible solutions in this method.
3) In Greedy method, we may find out the optimal solution without revising the previously generated solutions.	3) Dynamic programming considers all possible sequences inorder to obtain the optimal solution.
4) In greedy method, there is no as such guarantee of getting of getting optimal solution.	4) In dynamic programming, it is guaranteed that dynamic programming will generate the optimal solution.
5) Only one decision sequence is generated	5) Multiple desicion sequences are generated.

single source shortest path problem (Bellmanford Algorithm) :-

The difference between single source shortest path problem using Dijkstras Algorithm and single source shortest path alg. problem using Bellmanford Algorithm is:

1) Dijkstra's Algorithm works only with positive edges, where as Bellmanford Algorithm works only with positive & negative edge weights.

2) Bellmanford Algorithm produce correct result In this trying out all possible solutions, then we take the optimal solution.

3) In Bellmanford Algorithm, relaxing all the edges upto (n-1) times. suppose if the graph contains 7 edges and 7 vertices, relaxing all the edges (n-1) = (7-1) = 6 times

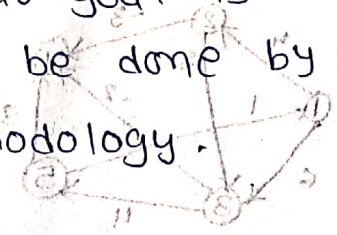
→ The drawback of Bellmanford Algorithm is that there is a negative weight cycle i.e; total weight of a cycle is negative, it cannot be solved using Bellmanford, Algorithm.

→ Let dist[U] be the length of a shortest path from source vertex V to destination vertex U under the constraint that the shortest path contains atmost 'i' edges.

→ dist'[U] is the length of an source vertex V to destination vertex U contains only one edge

$$dist'[U] = Cost[V, U]$$

→ our goal is to compute $dist^{n-1}[U]$, this can be done by using the dynamic programming methodology.



→ If the shortest path from V to U with almost K edges (K>1), then it is made up of a shortest path from V to some vertex followed by the edge ⟨J, U⟩. The path from V to J has K-1 edges and its length is $dist^{K-1}[J]$. All the vertices i, such that the edge ⟨i, U⟩ is in the graph are candidates for J. since we are interested in a shortest path, vertex i that minimizes $dist^{K-1}[i] + cost[i,U]$ is the correct value for J.

→ These observations result in the following formula for dist

$$dist^{K}[U] = \min\{dist^{K-1}[U], \min\{dist^{K-1}[i] + cost[i,U]\}\}$$

This formula can be used to compute $dist^{K}[U]$ from $dist^{K-1}[U]$ for $K = 2, 3, \ldots n-1$.

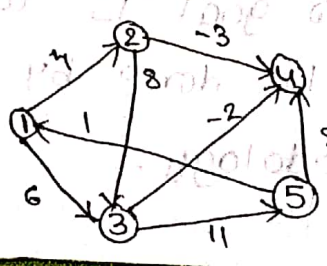
→ The time complexity of Bellmanford Algorithm is $O(|V| * |E|) = O(n * n-1)$

$\therefore O(n^2)$

→ If the given graph is a complete graph then the time complexity of Bellman-ford algorithm is $O(n^3)$

complete graph = $\frac{n(n-1)}{2}$ edges

1) Find out the shortest path from source vertex to remaining all the vertices in the given graph using Bellmanford Algorithm in dynamic programming.



step-1 :- k = 1

$dist^k[v] = dist^1[v] = cost[v,v]$

$dist^1[1] = \overset{cost}{dist}[1,1] = 0$

$dist^1[2] = cost[1,2] = 4$

$dist^1[3] = cost[1,3] = 6$

$dist^1[4] = \infty$

$dist^1[5] = \infty$

step-2 :- k = 2

$$\begin{bmatrix} dist^2[2] = cost[2,2] = 0 \\ dist^2[3] = cost \end{bmatrix}$$

$$dist^2[2] = \min\left\{ dist^{2-1}[2], \min\left\{ \begin{matrix} dist^{2-1}[3] + cost[3,2] \\ dist^{2-1}[4] + cost[4,2] \\ dist^{2-1}[5] + cost[5,2] \end{matrix} \right\} \right\}$$

$$= \min\{ 4, \min\{ 6+\infty, \infty+\infty, \infty+\infty \} \}$$

$$= \min\{ 4, \infty \}$$

$$= 4$$

$$dist^2[3] = \min\left\{ dist^{2-1}[3], \min\left\{ \begin{matrix} dist^{2-1}[2] + cost(2,3) \\ dist^{2-1}[4] + cost(4,3) \\ dist^{2-1}[5] + cost(5,3) \end{matrix} \right\} \right\}$$

$$= \min\{ 6, \min\{ 4+8, \infty+\infty, \infty+\infty \} \}$$

$$= 6$$

$$dist^2[4] = \min\left\{ dist^{2-1}[4], \min\left\{ \begin{matrix} dist^{2-1}[2] + cost(2,4) \\ dist^{2-1}[3] + cost(3,4) \\ dist^{2-1}[5] + cost(5,4) \end{matrix} \right\} \right\}$$

$$= \min\{ \infty, \min\{ 4+(-3), 6+(-2), \infty+8 \} \}$$

$$= \min\{ \infty, \min\{ 1, 4, \infty \} \}$$

$$= 1$$

$$dist^2[5] = \min\left\{ dist^{2-1}[5], \min\left\{ \begin{matrix} dist^{2-1}[2] + cost(2,5) \\ dist^{2-1}[3] + cost(3,5) \\ dist^{2-1}[4] + cost(4,5) \end{matrix} \right\} \right\}$$

$$= \min\{ \infty, \min\{ 4+\infty, 6+11, \infty+\infty \} \}$$

$$= 17$$

step-3:- k=3

$dist^3[2] = \min\{dist^{3-1}[2], \min\{dist^{3-1}[3]+cost(3,2), dist^{3-1}[4]+cost(4,2), dist^{3-1}[5]+cost(5,2)\}\}$

$= \min\{8, \min\{6+\infty, 1+\infty, 17+\infty\}\}$

$= 4$

$dist^3[3] = \min\{dist^{3-1}[3], \min\{dist^{3-1}[2]+cost(2,3), dist^{3-1}[4]+cost(4,3), dist^{3-1}[5]+cost(5,3)\}\}$

$= \min\{6, \min\{4+8, 1+\infty, 17+11\}\}$

$= \min\{6, \min\{12, \infty, 28\}\}$

$= 6$

$dist^3[4] = \min\{dist^{3-1}[4], \min\{dist^{3-1}[2]+cost[2,4], dist^{3-1}[3]+cost(3,4), dist^{3-1}[5]+cost(3,4)\}\}$

$= \min\{1, \min\{4+(-3), 6+(-2)+17+8\}\}$

$= 1$

$dist^3[5] = \min\{dist^{3-1}[5], \min\{dist^{3-1}[2]+cost(2,5), dist^{3-1}[3]+cost(3,5), dist^{3-1}[4]+cost(4,5)\}\}$

$= \min\{17, \min\{4+\infty, 6+11, 1+\infty\}\}$

$= 17$

step-4:- k=4

$dist^4[2] = \min\{dist^3[2], \min\{dist^3[3]+cost(3,2), dist^3[4]+cost(4,2), dist^3(5)+cost(5,2)\}\}$

$= \min\{4, \min\{6+\infty, 1+\infty, 17+\infty\}\}$

$= 4$

$dist^4[3] = \min\{ dist^3[3], \min\{ dist^3[4]+cost(4,3), dist^3[5]+cost(5,3) \} \}$

$= \min\{6, \min\{4+8, 1+\infty, 17+\infty\}\}$

$= 6$

$dist^4[4] = \min\{ dist^3[4], \min\{ dist^3[2]+cost(2,4), [3] (2,3), [5] (2,5) \} \}$

$= \min\{4, \min\{4+(-3), 6+8, 17+\infty\}\}$

$= 1$

$dist^4[5] = \min\{ dist^3[5], \min\{ dist^3[2]+cost(2,5), [3] + (3,5), [4] + (4,5) \} \}$

$= \min\{17, \min\{4+\infty, 6+11, 1+\infty\}\}$

$= 17.$

The shortest paths from source vertex 1 to remaining all the vertices 2, 3, 4, 5 are shown below

$1 \to 2 = 1-2 = 4$

$1 \to 3 = 1-3 = 6$

$1 \to 4 = 1-2-4 = 4-3 = 1$

$1 \to 5 = 1-3-5 = 6+11 = 17$

Algorithm BellmanFord(V, cost, dist, n)

{

for i := 1 to n do

dist[i] := cost(V, i);

for k := 2 to n-1 do

{ for (each u such that u ≠ v and u has

atleast one incoming edge) do

for each <i,v> in the graph do

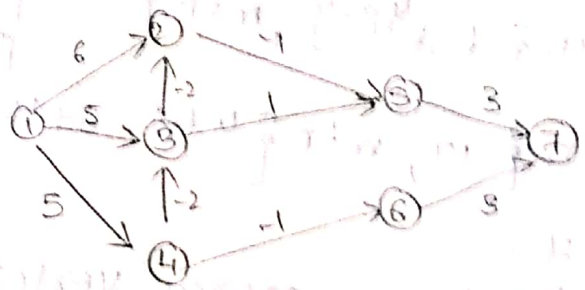
relaxation { if [dist[v] > dist[i] + cost[i,v]]

dist[v] = dist[i] + cost[i,v];

}

}

2) Apply the Bellmanford Algorithm for the following graph and find out the shortest distance from source vertex to remaining all the vertices in the given graph.



Step-1 :- K=1

$dist^k[v] = dist'[v] = cost[v,v]$

$dist'[1] = cost[1,1] = 0$

$dist'[2] = 6$ $cost[1,2] = 6$

$dist'[3] = cost[1,3] = 5$

$dist'[4] = cost[1,4] = 5$

$dist'[5] = \infty$

$dist'[6] = \infty$

$dist'[7] = \infty$

Step-2:- K=2

$dist^2[2] = \min\{ dist^{2-1}[2], \min\{ dist'[3] + cost(3,2), dist'[4] + (4,2), dist'[5] + (5,2), dist'[6] + (6,2), dist'[7] + (7,2) \} \}$

$= \min\{ 6, \min\{ 5+(-2), 6+\infty, \infty+\infty, \infty+\infty \} \}$

$= 3$

$dist^2[3] = \min\{dist^{2-1}[3], \min\{dist^1[2] + cost(2,3), (4,3), (5,3), (6,3), (7,3)\}\}$

$= \min\{5, \min\{6+\infty, 5+(-2), \infty, \infty, \infty\}\}$

$= \min\{5, \min\{\infty, 3, \infty, \infty, \infty\}\}$

$dist^2[4] = \min\{dist^{2-1}[4], \min\{dist^1[2] + cost(2,4), [3] (3,4), [5] (5,4), [6] (6,4), [7] (7,4)\}\}$

$= \min\{5, \min\{6+\infty, 5+\infty, \infty, \infty, \infty\}\}$

$= 5$

$dist^2[5] = \min\{dist^{2-1}[5], \min\{dist^1[2] + cost(2,5), [3] (3,5), [4] (4,5), [6] (6,5), [7] (7,5)\}\}$

$= \min\{\infty, \min\{6+(-1), 5+1, 5+\infty, \infty, \infty\}\}$

$= 5$

$dist^2[6] = \min\{dist^{2-1}[6], \min\{dist^1[2] + cost(2,6), [3] + cost(3,6), [4] + cost(4,6), [5] (5,6), [7] (7,6)\}\}$

$= \min\{\infty, \min\{6+\infty, 5+\infty, 5+(-1), \infty, \infty\}\}$

$= \min\{\infty, \min\{\infty, \infty, 4, \infty, \infty\}\}$

$= 4$

$dist^2[7] = \min\{dist^{2-1}[7], \min\{dist^1[2] + cost(2,7), [3] (3,7), [4] (4,7), [5] (5,7), [6] (6,7)\}\}$

$= \min\{\infty, \min\{6+\infty, 5+\infty, 5+\infty, \infty+3, \infty+3\}\}$

$= \infty$

$dist^3[2] = \min\{dist^2[2], \min\{dist^2[3] + cost(3,2)$
$[4] \quad (4,2)$
$[5] \quad (5,2)$
$[6] \quad (6,2)$
$[7] \quad (7,2)$

$= \min\{3, \min\{3+(-2), 5+\infty, 5+\infty, 4+\infty, \infty\}\}$

$= 1$

$dist^3[3] = \min\{dist^2[3], \min\{dist^2[2] + cost(2,3)$
$[4] \quad (4,3)$
$[5] \quad (5,3)$
$[6] \quad (6,3)$
$[7] \quad (7,3)$

$= \min\{3, \min\{3+\infty, 5+(-2), 5+\infty, 4+\infty, \infty\}\}$

$= 3$

$dist^3[4] = \min\{dist^2[4], \min\{dist^2[2] + cost(2,4)$
$[3] \quad (3,4)$
$[5] \quad (5,4)$
$[6] \quad (6,4)$
$[7] \quad (7,4)$

$= \min\{5, \min\{3+\infty, 3+\infty, 5+\infty, 4+3, \infty\}\}$

$= 5$

$dist^3[5] = \min\{dist^2[5], \min\{dist^2[2] + cost(2,5)$
$[3] \quad (3,5)$
$[4] \quad (4,5)$
$[6] \quad (6,5)$
$[7] \quad (7,5)$

$= \min\{5, \min\{3+(-1), 3+1, 5+\infty, 4+\infty, \infty\}\}$

$= 2$

$dist^3[6] = \min\{dist^2[6], \min\{dist^2[2] + cost(2,6)$
$[3] \quad (3,6)$
$[4] \quad (4,6)$
$[5] \quad (5,6)$
$[7] \quad (7,6)$

$= \min\{4, \min\{3+\infty, \infty, 5+(-1), \infty, \infty\}\}$

$= 4$

$dist^3[7] = \min\{dist^2[7], \min\{dist^2[2]+cost(2,7), dist^2[3]+cost(3,7), dist^2[4]+cost(4,7), dist^2[5]+cost(5,7), dist^2[6]+cost(6,7)\}\}$

$= \min\{\infty, \min\{\infty, \infty, \infty, 5+3, 4+3\}\}$

$= 7$

$dist^4[2] = \min\{dist^3[2], \min\{dist^3[3]+(3,2), dist^3[4]+(4,2), dist^3[5]+(5,2), dist^3[6]+(6,2), dist^3[7]+(7,2)\}\}$

$= \min\{1, \min\{3+(-2), \infty, \infty, \infty, \infty\}\}$

$= 1$

$dist^4[3] = \min\{dist^3[3], \min\{dist^3[2]+cost(2,3), dist^3[4]+(4,3), dist^3[5]+(5,3), dist^3[6]+(6,3), dist^3[7]+(7,3)\}\}$

$= \min\{3, \min\{\infty, 5+(-2), \infty, \infty, \infty\}\}$

$= 3$

$dist^4[4] = \min\{dist^3[4], \min\{dist^3[2]+cost(2,4), dist^3[3]+(3,4), dist^3[5]+(5,4), dist^3[6]+(6,4), dist^3[7]+(7,4)\}\}$

$= \min\{5, \min\{\infty, \infty, \infty, \infty, \infty\}\}$

$= 5$

$dist^4[5] = \min\{dist^3[5], \min\{dist^3[2]+cost(2,5), dist^3[3]+(3,5), dist^3[4]+(4,5), dist^3[6]+(6,5), dist^3[7]+(7,5)\}\}$

$= \min\{2, \min\{\infty, 3+1, \infty, \infty, \infty\}\}$

$= 2$

$dist^4[6] = \min\{dist^3[6], \min\{dist^3[2]+cost(2,6), dist^3[3]+(3,6), dist^3[4]+(4,6), dist^3[5]+(5,6), dist^3[7]+(7,6)\}\}$

$= \{4, \min\{\infty, \infty, 5+(-1), \infty, \infty\}\}$

$= 4$

$dist^4[7] = \min\{dist^3[7], \min\{dist^3[2] + cost(2,7)$
$[3] \quad (3,7)$
$[4] \quad (4,7)$
$[5] \quad (5,7)$
$[6] \quad (6,7)\}$

$= \min\{7, \min\{\infty, \infty, \infty, 2+3, 4+3\}\}$

$= 5$

$dist^5[2] = \min\{dist^4[2], \min\{dist^4[3] + cost(3,2)$
$[4] \quad (4,2)$
$[5] \quad (5,2)$
$[6] \quad (6,2)$
$[7] \quad (7,2)\}$

$= \min\{1, \min\{3+(-2), \infty, \infty, \infty, \infty\}\}$

$= 1$

$dist^5[3] = \min\{dist^4[3], \min\{dist^4[2] + cost(2,3)$
$[4] \quad (4,3)$
$[5] \quad (5,3)$
$[6] \quad (6,3)$
$[7] \quad (7,3)\}$

$= \min\{3, \min\{1+\infty, 5+(-2)+\infty, \infty, \infty\}\}$

$= 3$

$dist^5[4] = \min\{dist^4[4], \min\{dist^4[2] + cost(2,4)$
$[3] \quad (3,4)$
$[5] \quad (5,4)$
$[6] \quad (6,4)$
$[7] \quad (7,4)\}$

$= \min\{5, \min\{1+\infty, \infty, \infty, \infty, \infty\}\}$

$= 5$

$dist^5[5] = \min\{dist^4[5], \min\{dist^4[2] + cost(2,5)$
$[3] \quad (3,5)$
$[4] \quad (4,5)$
$[6] \quad (6,5)$
$[7] \quad (7,5)\}$

$= \min\{2, \min\{1+1, 3+1, \infty, \infty, \infty\}\}$

$= 0$

$dist^5[6] = \min\{dist^4[6], \min\{dist^4[2] + cost(2,6)$
$[3] \quad (3,6)$
$[4] \quad (4,6)$
$[5] \quad (5,6)$
$[7] \quad (7,6)\}$

$= \min\{4, \min\{\infty, \infty, 5-1, \infty, \infty\}\}$

$= 4$

$dist^5[7] = \min\{dist^4[7], \min\{dist^4[2] + (2,7)$
$[3] \quad (3,7)$
$[4] \quad (4,7)$
$[5] \quad (5,7)$
$[6] \quad (6,7)\}$

$= \min\{5, \min\{\infty, \infty, \infty, 2+3, 4+3\}\}$

$= 5$

$dist^6[2] = \min\{dist^5[2], \min\{dist^5[3] + \quad (3,2)$
$[4] \quad (4,2)$
$[5] \quad (5,2)$
$[6] \quad (6,2)$
$[7] \quad (7,2)\}$

$= \min\{1, \min\{3+(-2), 5+\infty, \infty, \infty, \infty\}\}$

$= 1$

$dist^6[3] = \min\{dist^5[3], \min\{dist^5[2] + \quad (2,3)$
$[4] \quad (4,3)$
$[5] \quad (5,3)$
$[6] \quad (6,3)$
$[7] \quad (7,3)\}$

$= \min\{3, \min\{\infty, 5+(-2), \infty, \infty, \infty\}\}$

$= 3$

$dist^6[4] = \min\{ dist^5[4], \min\{dist^5[2] + (2,4)$
$[3] \quad (3,4)$
$[5] \quad (5,4)$
$[6] \quad (6,4)$
$[7] \quad (7,4)\}$

$= \min\{5, \min\{\infty, \infty, \infty, \infty, \infty\}\}$

$= 5$

$dist^6[5] = \min\{ dist^5[5], \min\{dist^5[2] + (2,5)$
$[3] \quad (3,5)$
$[4] \quad (4,5)$
$[6] \quad (6,5)$
$[7] \quad (7,5)$

$= \min\{0, \infty - 1, 3+1, 5+\infty, \infty, \infty\}$

$= 0$

$dist^6[6] = \min\{ dist^5[6], \min\{dist^5[2] + (2,6)$
$[3] \quad (3,6)$
$[4] \quad (4,6)$
$[5] \quad (5,6)$
$[7] \quad (7,6)$

$= \min\{4, \min\{\infty, \infty, 5-1, \infty, \infty\}\}$

$= 4$

$dist^6[7] = \min\{ dist^5[7], \min\{dist^5[2] + (2,7)$
$[3] \quad (3,7)$
$[4] \quad (4,7)$
$[5] \quad (5,7)$
$[6] \quad (6,7)$

$= \min\{5, \min\{\infty, \infty, \infty, 0+3, 4+3\}\}$

$= 3$

$1 \to 2 = 1-3-1 = 1$

$1 \to 3 = 1-4-3 = 3$

$1 \to 4 = 1-4 = 5$

$1 \to 5 = 1-4-3-2-5 = 0$

$1 \to 6 = 1-4-6 = 5-1 = 4$

$1 \to 7 = 1-4-3-2-5-7 = 3$