Unit - 4

Backtracking :-

Recursion is the key in Backtracking programming. We start with one possible move out of many possible move and try to solve the problem. If we are able to solve the problem with the selected move then print the solution else backtrack and select some other move and try to solve it.

If all moves does not work then there is no solution for the problem.

Backtracking algorithm determine problem solution by systematically searching the solution space for the given problem instance.

This search is done by using Tree organisation for the solution space.

It is based on Depth-first-Search with bounding function.

N-Queen's Problem

There are N Queens and we have to arrange these N Queens in a N x N matrix in such a manner that no two Queens should attack each other (i.e Not in attacking position).

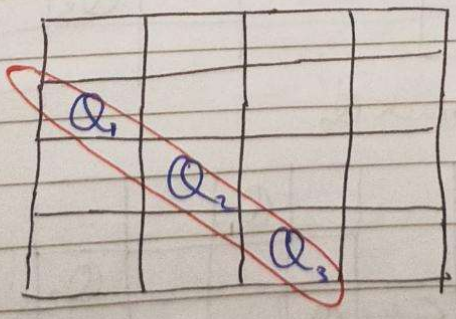
Now, attacking position means :-

1) No two Queen should be in same row.
2) No two Queen should be in same column
3) Two Queen should not be in diagonal position.

Common Diagonal Condition :-

1) Queens placed from upper left to lower right on same diagonal then we have same (row - column) value.

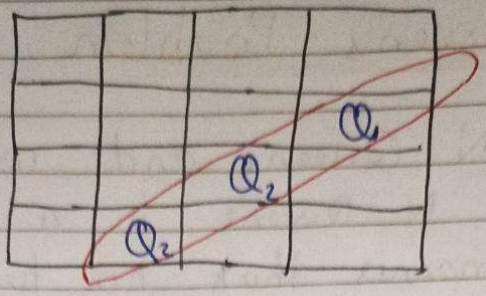
$Q_1 \rightarrow (2,1)$
$Q_2 \rightarrow (3,2)$
$Q_3 \rightarrow (4,3)$



$\Rightarrow$ Same value of row - column.

$\Rightarrow (i-j) = (k-l)$.

2) Queen placed from upper ~~left~~ Right to Lower left on same diagonal, then we have same row + Column value.



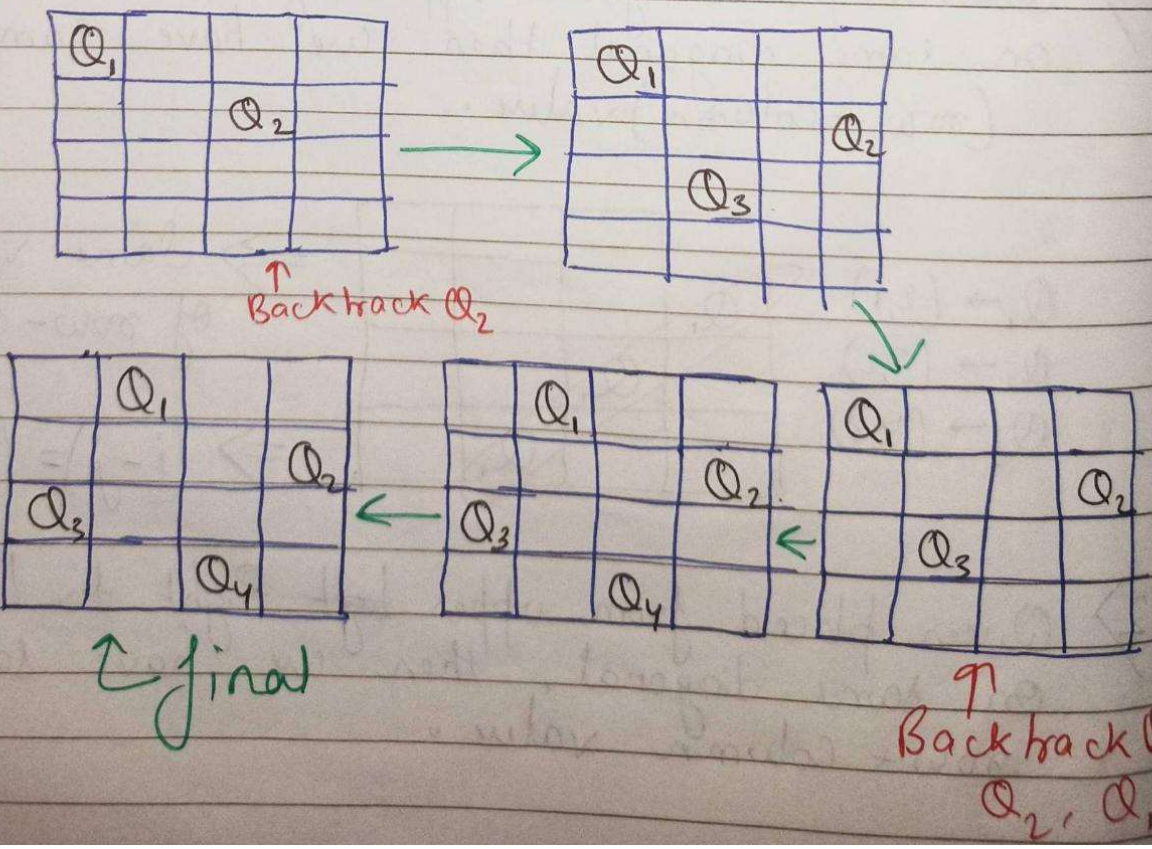
$Q_1 \rightarrow 2,4$
$Q_2 \rightarrow 3,3$
$Q_3 \rightarrow 4,2$

$$i+j = k+l$$

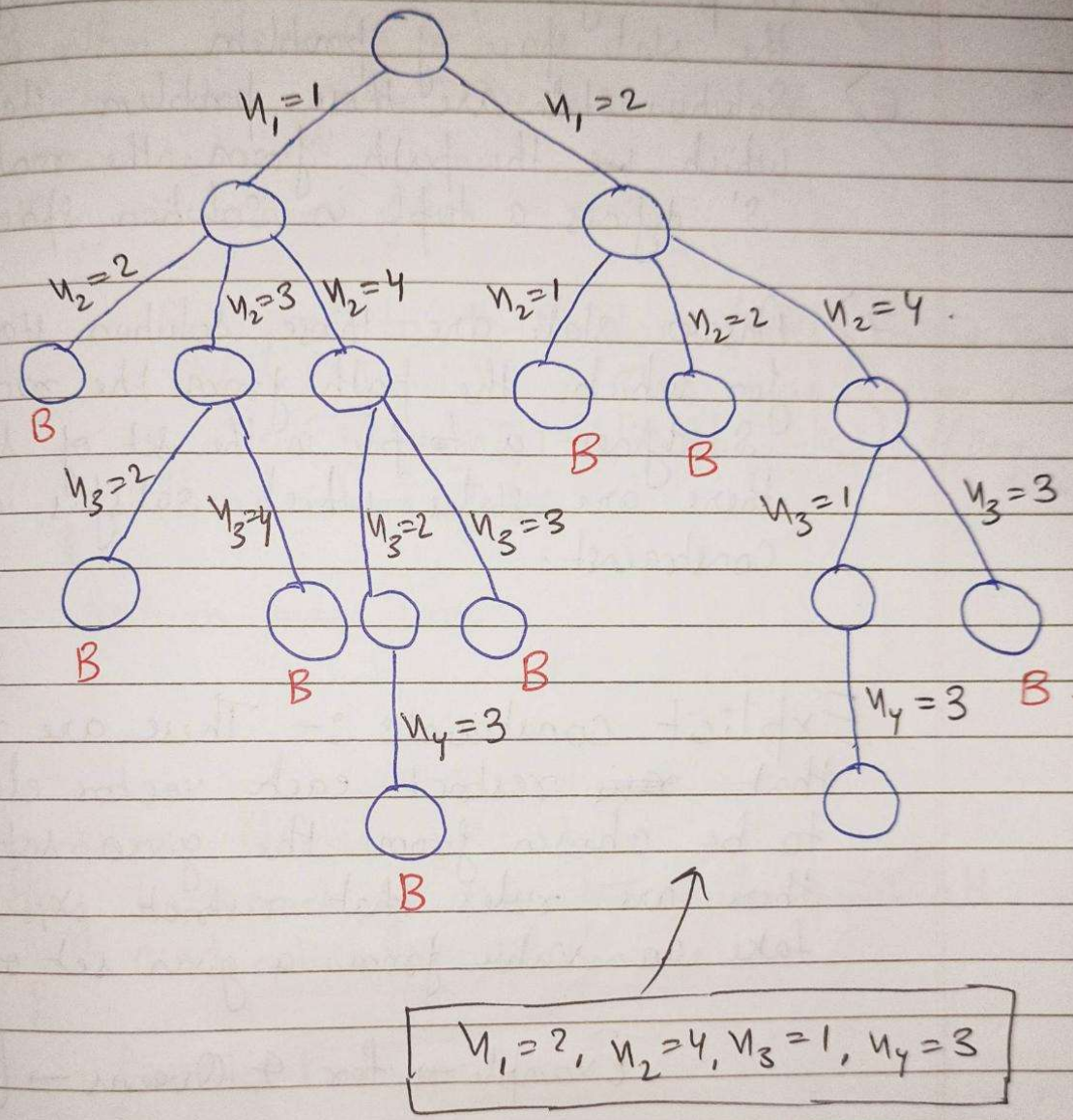
On Combining these two condition :-

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

4-Queens Problem :-



State Space Tree for 4 Queen's Problem



Terminology :-

1) Live Node :- A node which has been newly generated is called Live Node.
2) E-Node :- The live node whose children are currently being generated is called E-Node
3) Dead Node :- It is a generated node which are not to be expanded further.

4) Each node in the tree defines a problem state.

5) All path from root to other nodes defines the state space of problem

6) Solution state are those problem state for which ~~wt~~ the path from the root to 'S' defines a tuple in solution space.

7) Answer state are those solution state 'S' for which the path from the root to S defines a tuple in the set of solutions, there are states which satisfies implicit constraint.

Explicit constraints :- These are rules that ~~this~~ restrict each vector element to be chosen from the given set or there are rules that restrict x_i to take an value from a given set only.

Example → for 4 Queens → [1, 2, 3, 4]

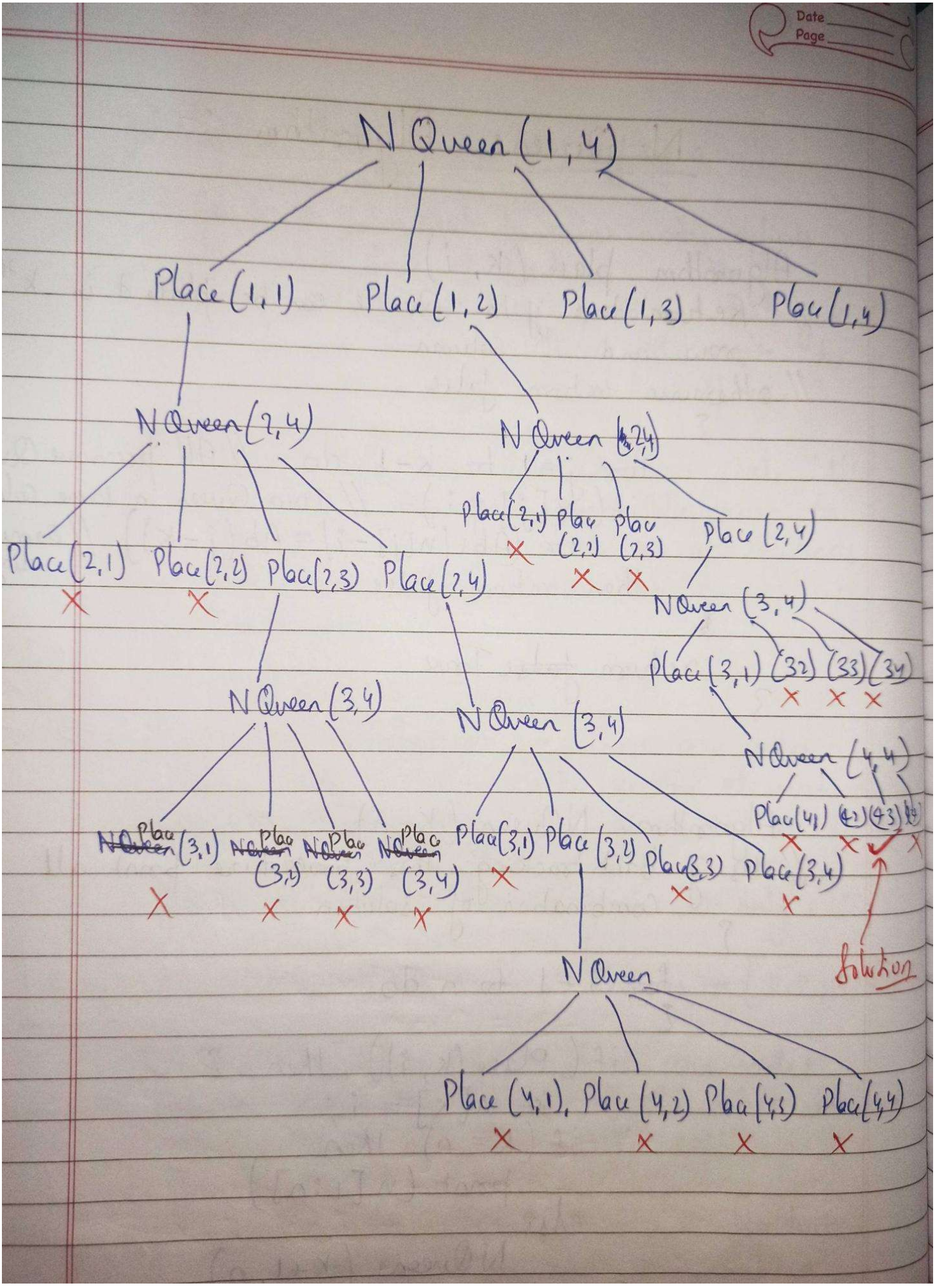
Implicit Constraints :- These are rules which determines which of the tuple in the solution actually satisfy criterion function.

Example → For N-Queens :- No two Queen should be in same row, Column and diagonal.

N-Queen's Algorithm :-

```
Algorithm place (k, i)
// Return true if the Queue can be placed in k^th
   row and i^th column
// otherwise return false
   {
      for j = 1 to k-1 do  // All previous Queen
      if (x[j] = i)  // Two Queen in same column
         or (Abs(x[j] - i) = Ab(j - k))  // Same diagonal
      then return false
   }
   return ~~false~~ True
}
```

```
Algorithm NQueen (k, n)
// Using backtracking, this procedure print all
   Combination of solution
   {
      for i = 1 to n do
      {
         if ( Place (k, i)) , then {
            x[k] = i;
         if (k = n) then
            print (x[1:n])
         else
            NQueens (k+1, n)
      }
   }
}
```



Place (2, 1) :-

j = 1 to 1

j = 1 :-

x[i] = 1

2 = 1 false

or

abs(x[i] - 1) = abs(1 - 2)

|2 - 1| = |1 - 2| (True)

∴ Return false.

Place (2, 2) :-

j = 1 to 1

j = 1 :- x[i] = ~~2~~

2 = ~~2~~ (~~false~~) (True)

or

abs(x[i] - 2) = abs(1 - 2)

2 - 2 = abs(1 - 2) (false)

∴ return false.

Place (2, 3) :-

j = 1 :- x[i] = 3

2 = 3 (false)

or abs(x[i] - 3) = abs(1 - 2)

|2 - 3| = |1 - 2| (True)

∴ return false.

Place (2, 4) :-

j = 1 :- x[i] = 4 (false)

or abs(x[i] - 4) = 1 - 2 (False)

Both false ∴ return True.

Place(3,1) :-

j=1 to 2 :-

j=1 : x[1] = 1

2 > 1 (false)

or x[1] - 1 = 1 - 3

2 - 1 ≠ 1 - 3 (False)

j = 2 : x[2] = 1

4 = 1 (false)

x[2] - 1 = 2 - 3

4 - 1 = 2 - 1 (false)

∴ return True.

Place(4,1) :-

j = 1 to 3 :-

j = 1 : x[1] = 1

2 = 1 (false)

or x[1] - 1 = 1 - 4

2 - 1 = 1 - 4 (false)

j = 2 : x[2] = 1

4 = 1 (false)

or x[2] - 1 = 2 - 4

4 - 1 = 2 - 4 (false)

j = 3 :- x[3] = 1

1 = 1 (true)

or x[3] - 1 = 3 - 4

1 - 1 = 3 - 4 (false)

∴ Return false.

Place (4,2) :-

j = 1 :- x[1] = 2

2 = 2 (True)

∴ Return false

Place (4,3) :-

j = 1 :- x[1] = 3

2 = 3 (false)

or x[1] - 3 = 1 - 4

2 - 3 = 1 - 3 (false)

j = 2 :- x[2] = 3

4 = 3 (false)

or x[2] - 3 = 2 - 4 (false)

j = 3 :- x[3] = 3

1 = 3 (false)

or x[3] - 3 = 3 - 4

1 - 3 = 3 - 4 (false)

∵ All are false

∴ Return ~~false~~ True.

<2, 4, 1, 3> Solution

Sum of Subset :-

Sum of Subset problem is to find subset of elements that are selected from a given set whose sum adds up to a given number k.

We are considering set as non-negative set. Also assuming that there is no duplicates.

Algorithm :-

```
Sum of Subset (s, k, r) {
    x[k] = 1;
    if (s + w[k] = m) then
        write (x[1 : k])
    else if (s + w[k] + w[k+1] ≤ m)
        then sumofSubset (s+w[k], k+1, r-w[k])
    if (1 + r - w[k] > m) and (s + w[k+1] ≤ m)
    then {
        x[k] = 0;
        sumofSubset (s, k+1, r-w[k]);
    }
}
```

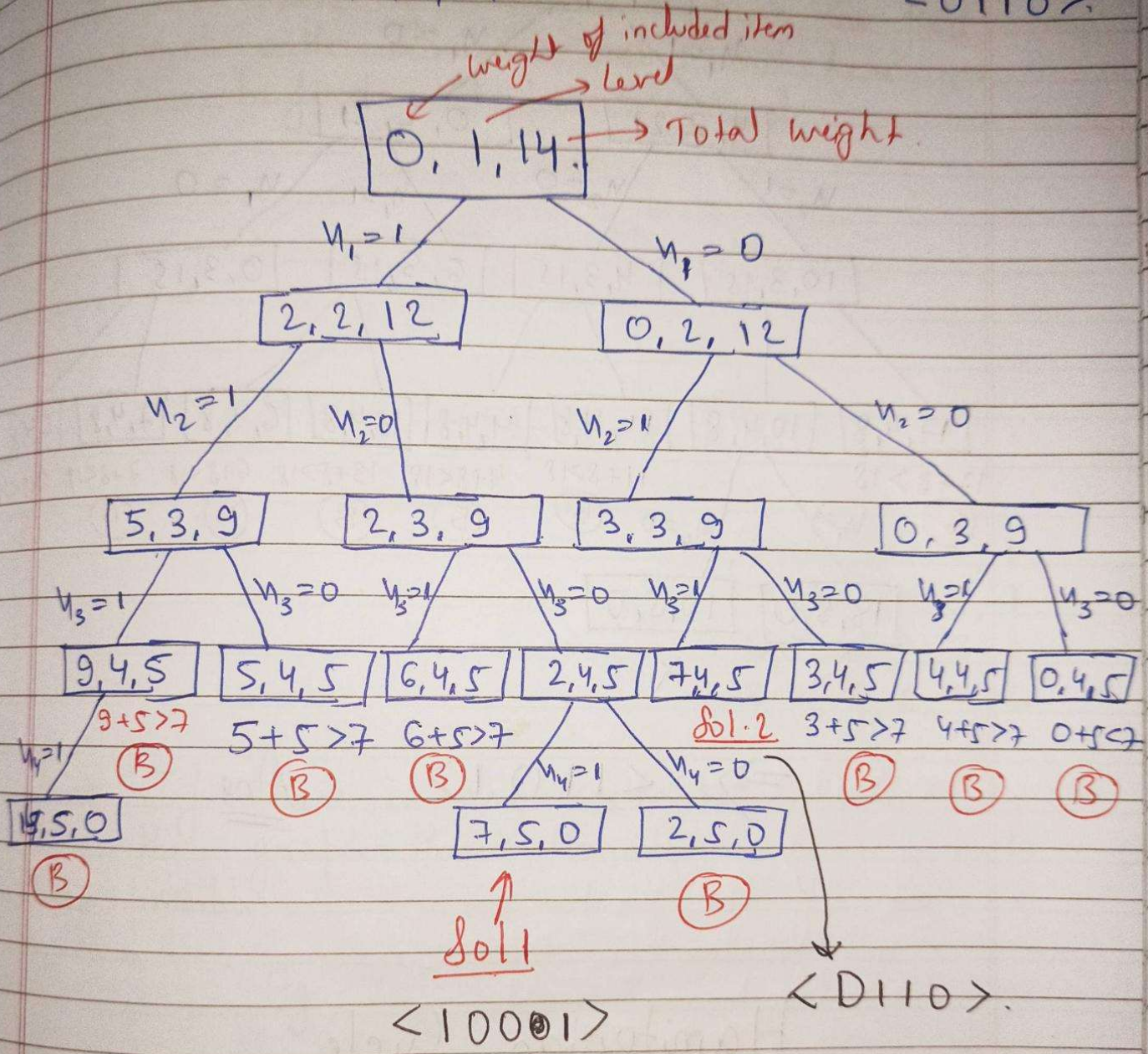
Q Given a set S having following entity $S = \{2, 3, 4, 5\}$ and $m = 7$, find sum of subset.

$m = 7$

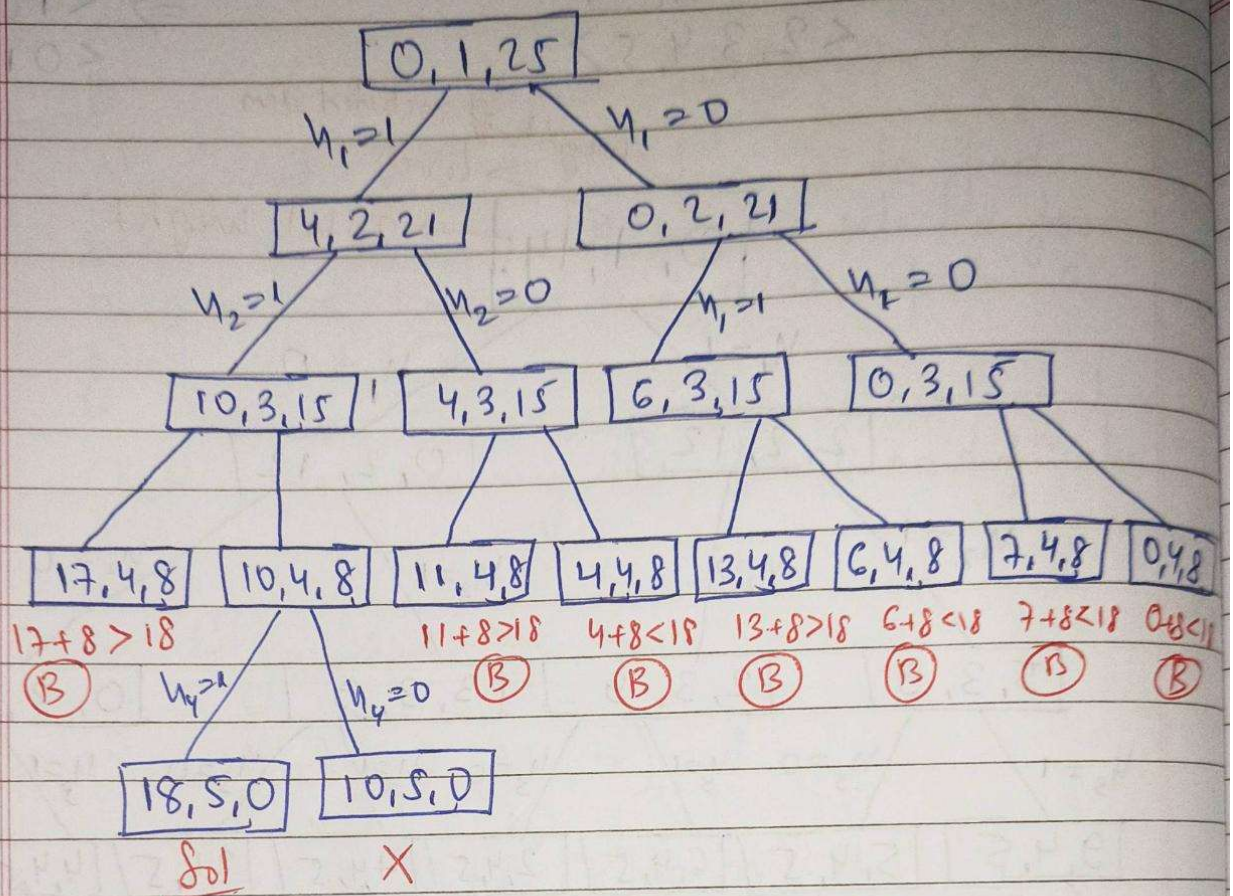
$\langle 2, 3, 4, 5 \rangle$

$\Rightarrow \langle 1001 \rangle$

$\langle 0110 \rangle$.



Question 2:- Given a set S having following entity $S = \langle 4, 6, 7, 8 \rangle$ and $m = 18$. Find Sum of Subset.

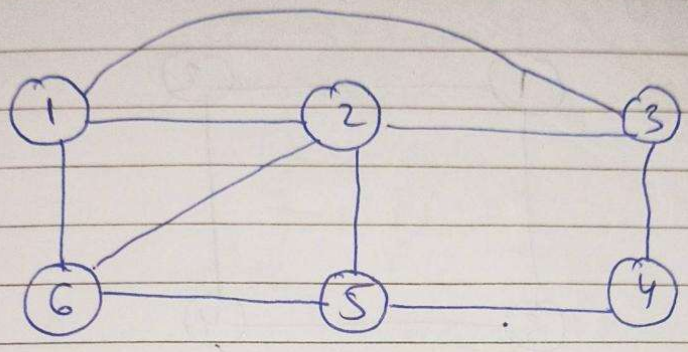


$\Rightarrow$ <1101>. Ans

Hamiltonian Cycle

If a graph is given then we have to start from some starting vertex and visit all the vertex exactly once and return back to starting vertex. This forms a cycle. We have to check if there is any Hamiltonian cycle possible in a graph or not. If there is more than one such cycle, find all those cycle.

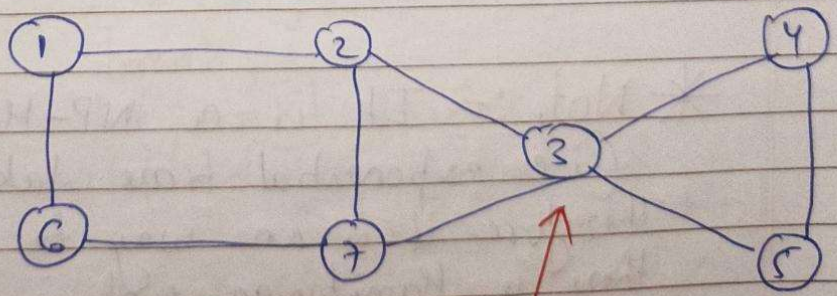
* In Travelling Salesman Problem, we find cycle with minimum cost, but here, we have to find all possible cycle.



1, 2, 3, 4, 5, 6, 1
1, 2, 6, 5, 4, 3, 1
1, 6, 2, 5, 4, 3, 1
2, 3, 4, 5, 6, 1, 2

* If there is any articulation point in the graph then Hamiltonian cycle is not possible.

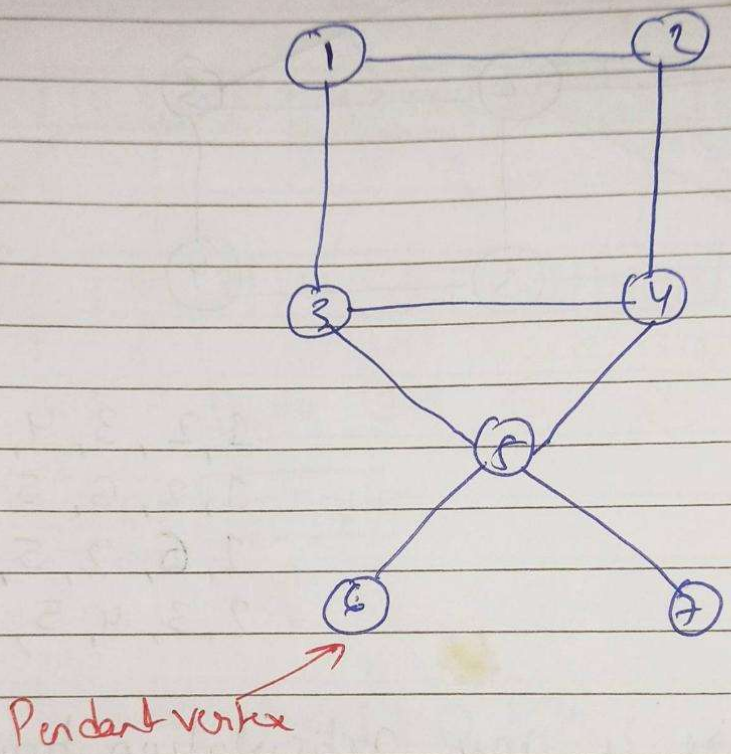
Eg.



Articulation Point / Connection Point / Junction Point.

* If there is any Pendant vertex in graph, Hamiltonian cycle is not possible.

Eg.



* If the graph is not connected (i.e any vertex is left isolated), then Hamiltonian cycle is not possible.

* Not :- It is a NP-Hard problem (i.e exponential time taking problem). Therefore it is no way to find out if there is Hamiltonian cycle in graph or not.

$$\text{Complexity} = O(n!)$$

Algorithm :-

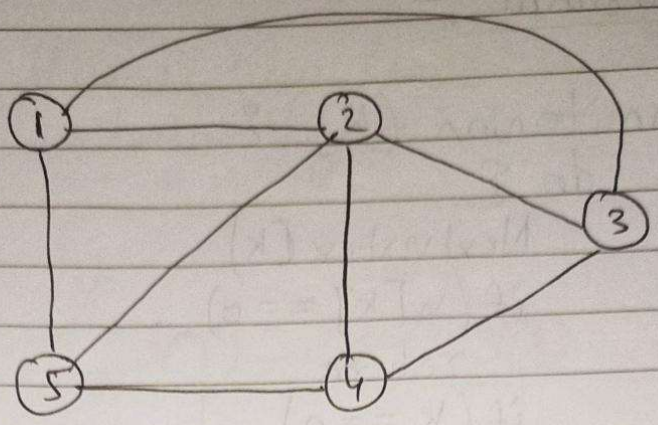
```
Hamiltonian (k) {
    do {
        Nextvertex [k]
        if (x[k] == 0)
            return
        if (k == n)
            print [1 : n]
        else
            Hamiltonian (k+1);
    } while (true);
}

Nextvertex (k) {
    do
        x[k] = (x[k] + 1) mod (n+1)
        if (x[k] == 0)
            return;
        if ( G[x[k-1], x[k]] ≠ 0) {
            for j = 1 to k-1
            do
                if (x[j] == x[k])
                    break;
            if (j == k) then
                if (k < n or k == n) &&
                    G(x[n], x[1]) ≠ 0)
                    return;
    } while (true);
}
```

Check if there is edge b/w vertex

No duplicates

Check edge back to vertex 1

Q



Sol →

$$G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

1 means edge is there. 0 means No edge

Initially.

$x_i \rightarrow$ | 0 | 0 | 0 | 0 | 0 |

Hamiltonian (1) ∴ Next vertex (1)

k = 1 :-

$x[1] = (x[1] + 1) \bmod (5+1)$

$x[1] = (0 + 1) \bmod 6$

$x[1] = 1$

∴ $x[1] = 1$

Hamiltonian (k+1) i.e. Hamiltonian (2)

∴ Next vertex (2).

k = 2 :-

$x[2] = (x[2] + 1) \bmod 6$

$x[2] = 1$

G(n[1], n[2] ≠ 0) True
∴ there is an edge b/w ① and ②
j = 1 to k-1 i.e. 1 to 2-1

j = 1 :- n[j] == n[k]
n[1] == n[2]
⇒ 1 == 1 (True)
break;

∴ n[2] = (1+1) mode 6
= 2

j = 1 :- n[1] == n[2]
= 1 == 2 (false)
∴ n[2] = 2.

Hamiltonian (3)
Next vertex (3).

k = 3 :-
n[3] = (n[3]+1) mod 6
n[3] = 1

G(n[2], n[3] ≠ 0)
∴ there is edge b/w vertex 2 & 3.
j = 1 to 2 :-
j = 1 :- n[1] == n[3] i.e 1 == 1 True;
break;

n[3] = (1+1) mod [6] = 2
n[1] == n[3] ⇒ 1 == 2 False
j = 2 :- n[2] == n[3] ⇒ 2 == 2 True; break
n[3] = (2+1) mod 6 = 3
n[2] == n[3] ⇒ 2 == 3 (false)
∴ n[3] = 3

Hamiltonian (4) :-

K=4, NextVertex(4).

$x[4] = (x[4]+1) \bmod 6$.

$x[4] = (0+1) \bmod 6 = 1$

$G[x[3], x[4]] \neq 0$

∴ There is edge b/w ③ & ④

j = 1 to 3 :-

j = 1 :-

$x[1] == x[4]$ i.e. 1 == 1 (True).

Break;

$x[4] = (x[4]+1) \bmod 6 = 2$

$x[1] == x[4]$ i.e. 1 == 2 (false)

j = 2 :-

$x[2] == x[4]$ i.e. 2 = 2 (True)

Break;

$x[4] = (x[4]+1) \bmod 6 = 3$

$x[2] == x[4] \Rightarrow 2 == 3$ (false)

j = 3 :-

$x[3] == x[4]$ i.e. 3 = 3 (True)

Break;

$x[4] = (x[4]+1) \bmod 6$.

$x[4] = 4$

$x[3] == x[4] \Rightarrow 3 == 4$ (false)

∴ Set $x[4] = 4$.

Hamiltonian (5)

K=5, NextVertex (5) :-

$x[5] = (x[5]+1) \bmod 6 = 1$

$G(x[5], x[4]) \neq 0$

∴ There is edge between 4 & 5.

j = 1 to 4 :-

j = 1 :- n[1] == n[5] ⇒ 1 == 2 (True)

Break;

n[5] = (1+1) mod 6 = 2

n[1] == n[5] ⇒ 1 == 2 (false)

j = 2 :- n[2] == n[5] ⇒ 2 == 2 (True)

Break;

n[5] = (2+1) mod 6 = 3

n[2] == n[5] ⇒ 2 == 3 (false)

j = 3 :- n[3] == n[5]

⇒ 3 == 3 (True), break

n[5] = (3+1) mod 6 = 4

n[3] == n[5] ⇒ 3 == 4 (false)

j = 4 :-

n[4] == n[5]

⇒ 4 == 4 (True)

Break

n[5] = (4+1) mod 6

n[5] = 5

n[4] == n[5] (false)

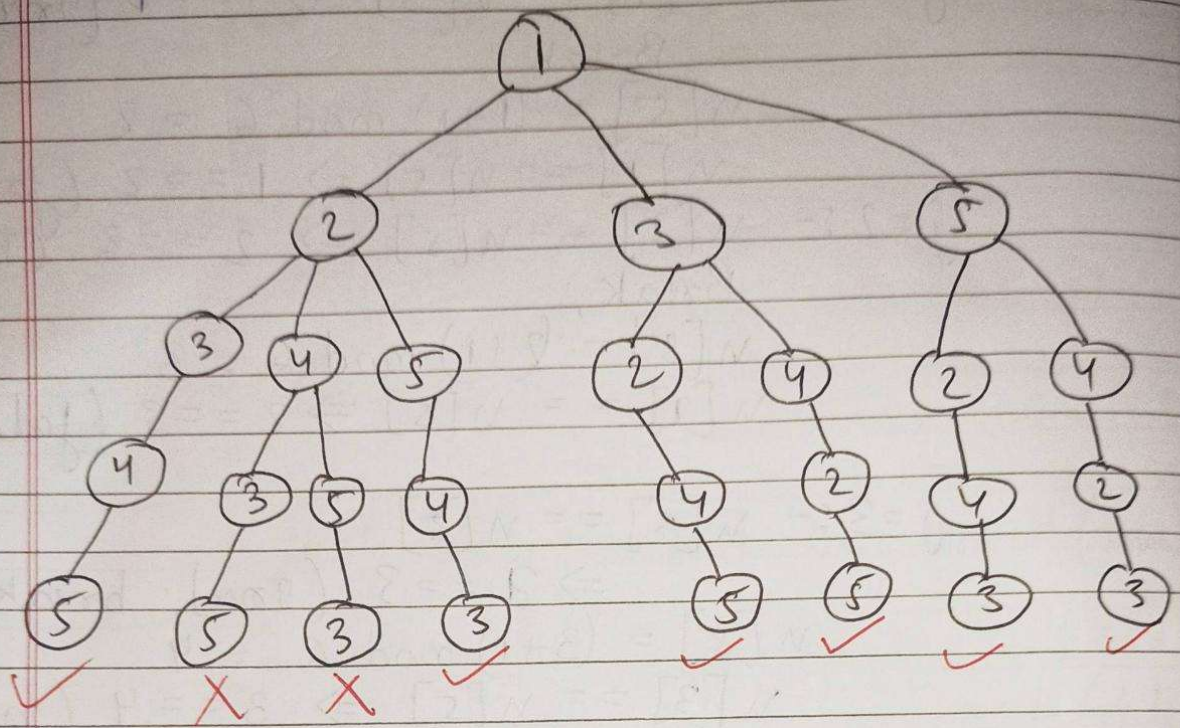
∴ Set n[5] = 5

K == n i.e 5 == 5 && G[n[5], n[i] ≠ 0]

True.

∴ There is edge between 5 & 1

⇒ | 1 | 2 | 3 | 4 | 5 | Ans



Time Complexity = $O(n!)$ or $O(n^n)$.

Branch & Bound :-

Generates Branches and we apply bound on the branches.

Based on BFS. We find optimal solution to problem.

Similarity :- We generate state space tree in this too.

0/1 Knapsack :-

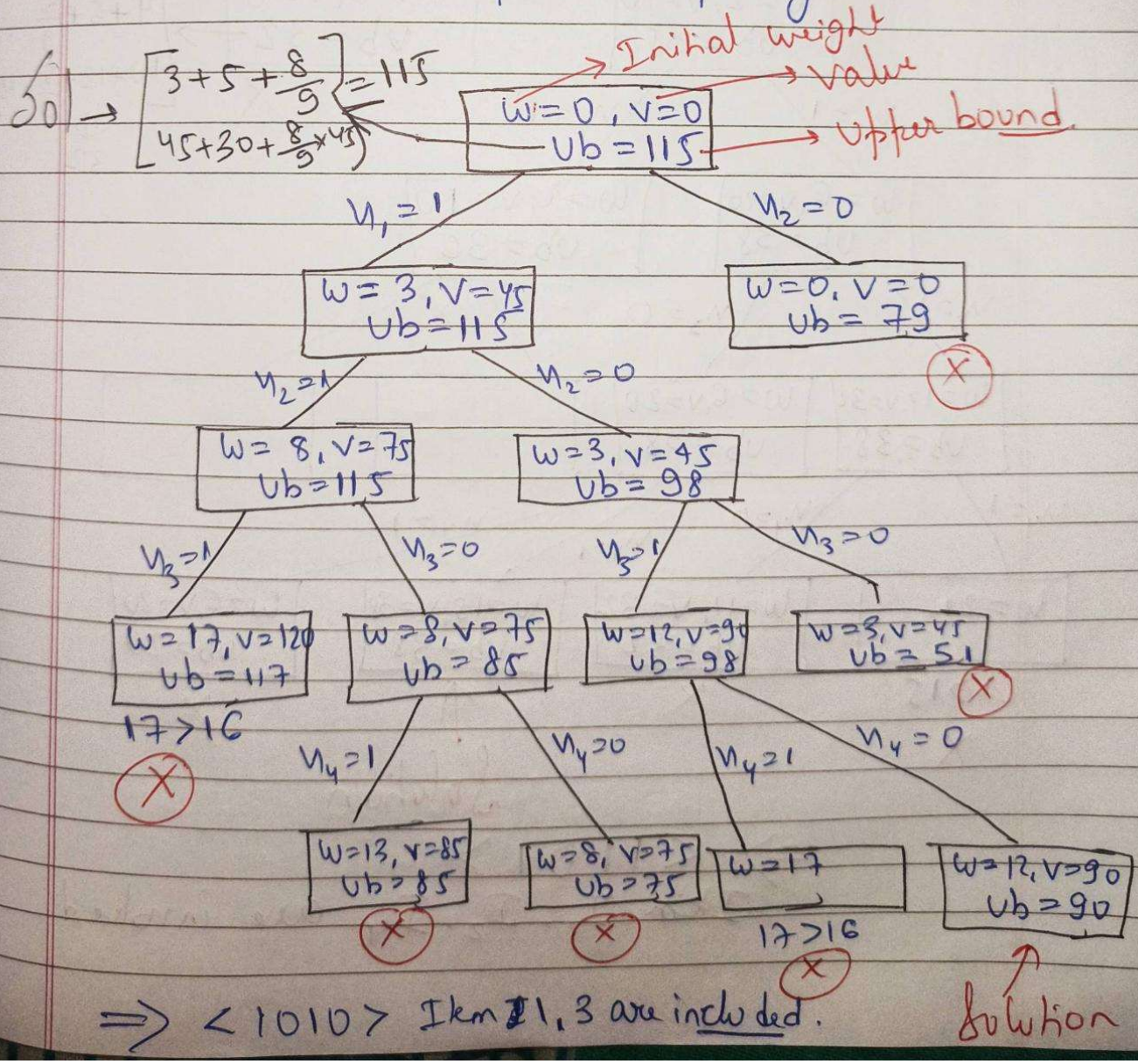
Q

Items	Weight	Value	$P_i = \frac{V_i}{w_i}$
x_1	3	45	15
x_2	5	30	6
x_3	9	45	5
x_4	5	10	2

Knapsack Capacity = 16

Sol → $\left[3+5+\frac{8}{9}\right] = 115$

$\left[45+30+\frac{8}{9}\times 45\right]$



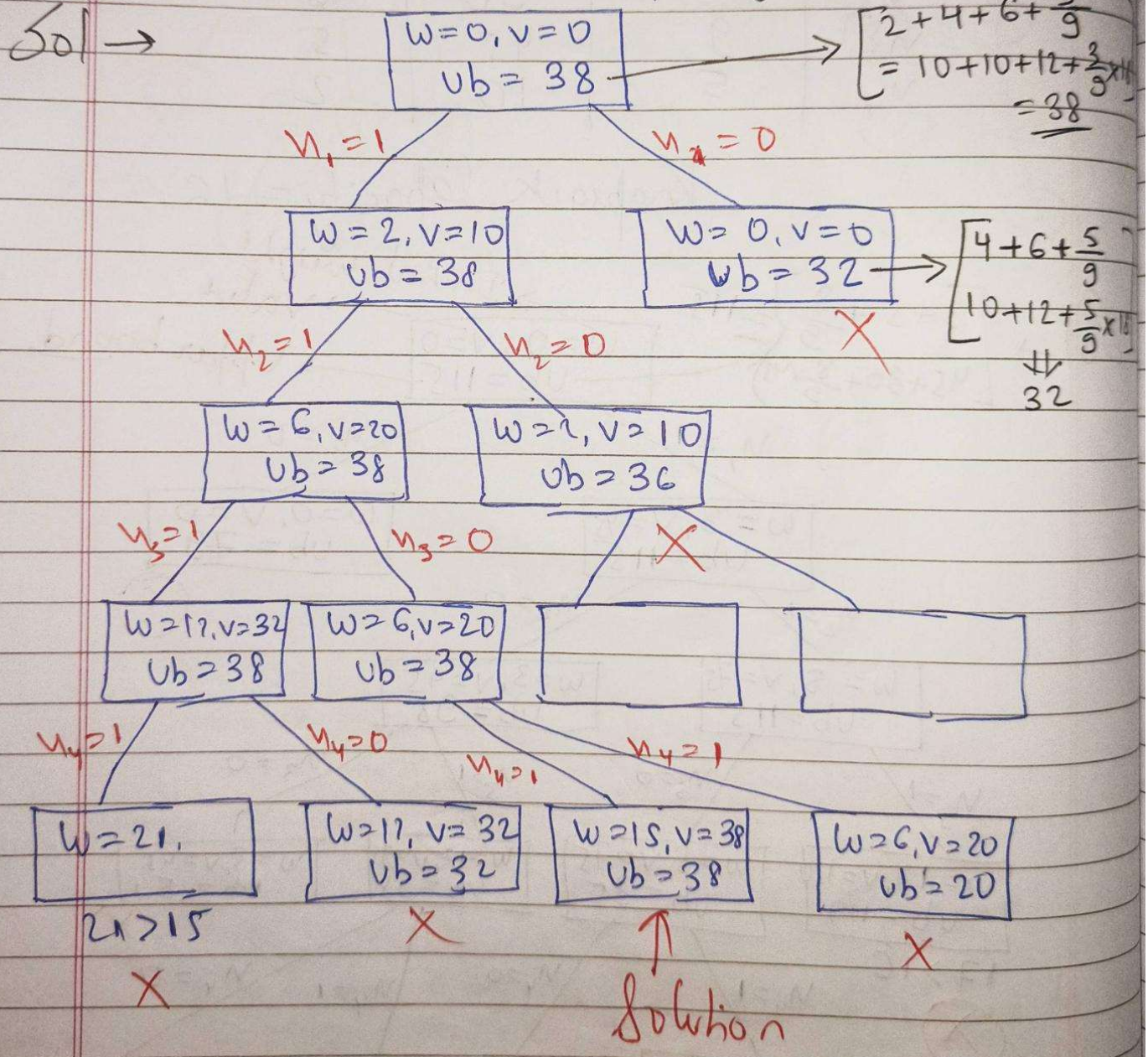
⇒ <10110> Item 1, 3 are included.

Q.

Item	Weight	Value	P_i
x_1	2	10	5
x_2	4	10	2.5
x_3	6	12	2
x_4	9	18	2

Capacity = 15

Sol →

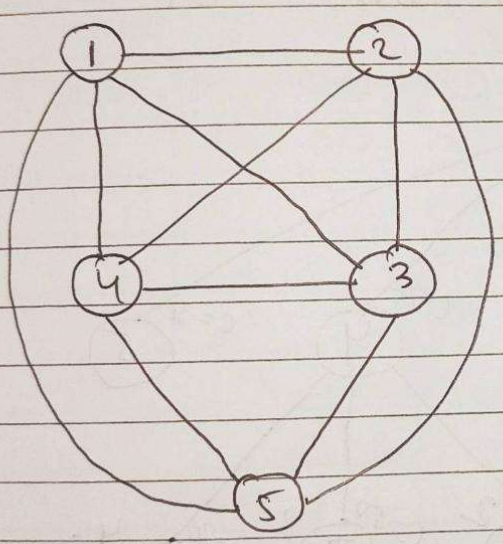


⇒ <1101>

Item I_1, I_2, I_4 are involved

Travelling Saleman Problem :-

Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visit every city exactly once and returns to the starting point.



	1	2	3	4	5
1	∞	20	30	10	11
2	15	∞	16	7	2
3	3	5	∞	2	4
4	19	6	18	∞	3
5	16	4	7	16	∞

Sol→

∞	20	30	10	11	10
15	∞	16	7	2	2
3	5	∞	2	4	2
19	6	18	∞	3	3
16	4	7	16	∞	4

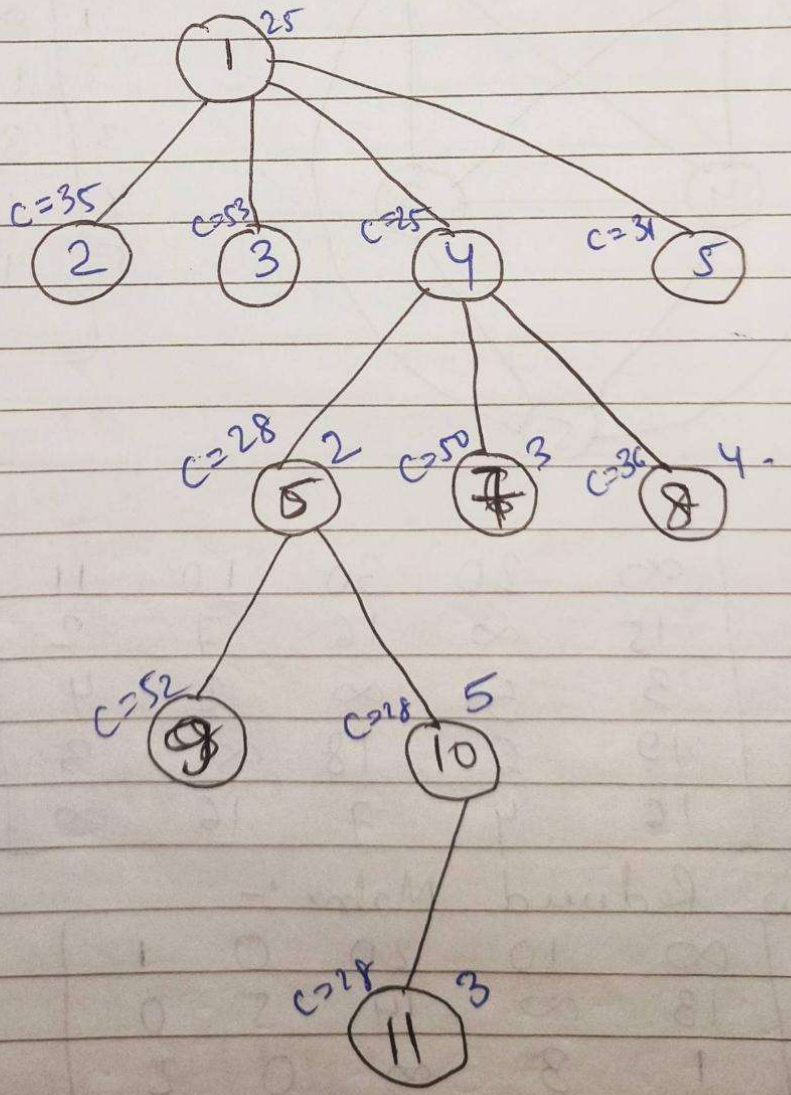
Row Reduced Matrix :-

∞	10	20	0	1
13	∞	14	5	0
1	3	∞	0	2
16	3	15	∞	0
12	0	3	12	∞
1	0	3	0	0

Column Reduced Matrix :-

∞	10	17	0	1	10
12	∞	11	2	0	2
0	3	∞	0	2	2
15	3	12	∞	0	3
11	0	0	12	∞	4
1	0	3	0	0	Total = 25

Cost of reduced matrix = 25



1-2 :-

∞	∞	∞	∞	∞	∞
∞	∞	11	2	0	0
0	∞	∞	0	2	0
15	∞	12	∞	0	0
11	∞	0	12	∞	0
0		0	0	0	$\hat{\gamma} = 0$

Cost of 1-2 :- $c[1,2] + \gamma + \hat{\gamma}$
$= 10 + 25 + 0$
$= 35$

1-3 :-

∞	∞	∞	∞	∞	
12	∞	∞	2	0	0
∞	3	∞	0	2	0
15	3	∞	∞	0	0
11	0	∞	12	∞	0
11	0		0	0	$\hat{\gamma} = 11$

Cost of 1-3 = $c[1,3] + \gamma + \hat{\gamma}$
$= 17 + 25 + 11$
$= 53$

1-4 :-

∞	∞	∞	∞	∞	
12	∞	11	∞	0	0
0	3	∞	∞	2	0
∞	3	2	∞	0	0
11	0	0	∞	∞	0
0	0	0		0	$\hat{\gamma} = 0$

Cost of 1-4 :- $c[1,4] + \gamma + \hat{\gamma}$
$= 0 + 25 + 0 = 25$

1-5 :-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 15 & 3 & 12 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \\ 0 & 0 & 0 & 0 & \end{bmatrix} \begin{matrix} 2 \\ 0 \\ 3 \\ 0 \\ \hat{\gamma} = 5 \end{matrix}$$

Cost of 1-5 :- $c[1,5] + \gamma + \hat{\gamma}$

$= 1 + 25 + 5$

$= 31$

Now, Minimum Cost route is (1-4) so we choose this and its matrix become the Base Matrix.

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

4-2 :-

(2-1) ←

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

Cost of [4-2] = $c[4,2] + c[4] + \hat{\gamma}$

$= 3 + 25 + 0$

$= 28$

4-3 :-

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

$$\begin{matrix} 11 & 0 & 0 \end{matrix} \qquad \hat{r} = 13$$

Cost of [4-3] = $c[4,3] + c[4] + \hat{r}$

= 12 + 25 + 13

= 50

4-5 :-

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

$$\begin{matrix} 0 & 0 & 0 \end{matrix} \qquad \hat{r} = 11$$

Cost of [4-5] = $c[4,5] + r[4] + \hat{r}$

= 0 + 25 + 11

= 36

Now, Minimum cost route is (4-2). So we Choose this and its matrix become base Matrix.

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

2-3 :-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix} \quad \hat{\gamma} = 13$$

(3-1) →

Cost of [2-3] = c[2,3] + c[6] + $\hat{\gamma}$

= 11 + 28 + 13

= 52

2-5 :-

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix} \quad \hat{\gamma} = 0$$

(5-1) →

Cost of [2-5] = c[2,5] + c[6] + $\hat{\gamma}$

= 0 + 28 + 0

= 28

Now, Minimum Cost route is (2-5), so, we select this and its matrix become base matrix.

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

5-3 :-

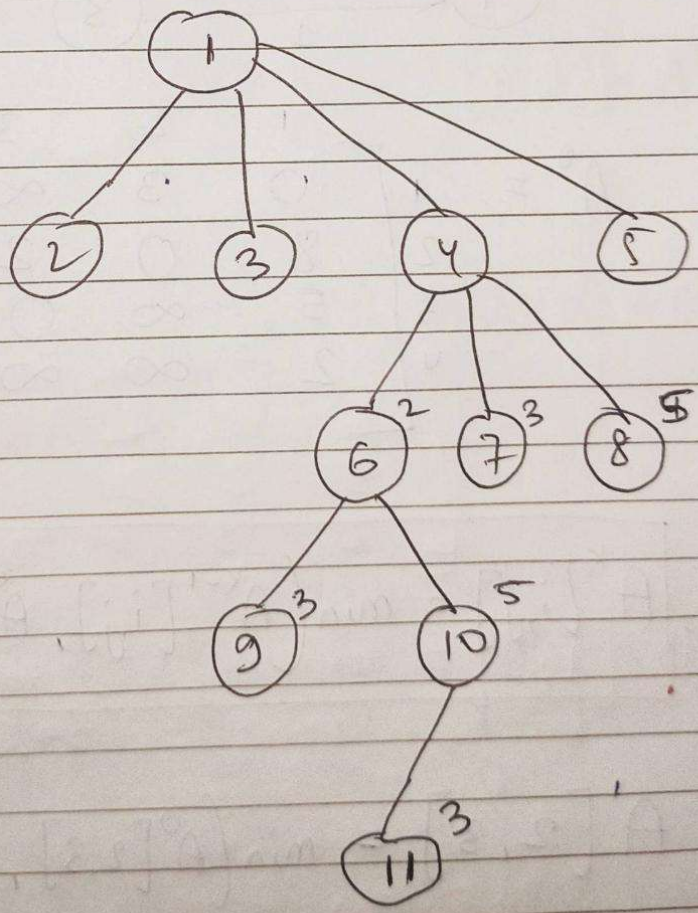
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix} \quad \hat{r} = 0$$

(3-1) → row 3

Cost of $c[5-3] = c[5,3] + c[10] + \hat{r}$

$= 0 + 28 + 0$

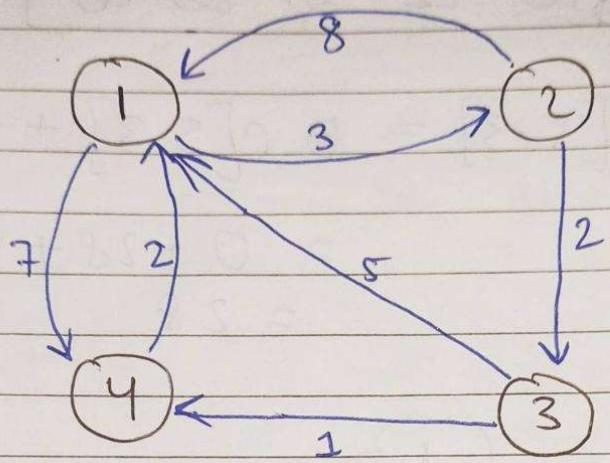
$= 28$



Route → 1-4-2-5-3 Ans

Floyd-Warshall Algorithm for all Pairs shortest Path

Q.



Sol →

$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & \infty \\ 5 & \infty & 0 & 1 \\ 2 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

formula ←

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

$$A^1[2,3] = \min\left(A^0[2,3],\ A^0[2,1] + A^0[1,3]\right)$$

$$= \min(2,\ 8+\infty)$$

$$= 2$$

$$A^1[2,4] = \min(A^0[2,4],\ A^0[2,1] + A^0[1,4])$$
$$= \min(\infty,\ 8+7)$$
$$= 15$$

$$A^1[3,2] = \min(A^0[3,2],\ A^0[3,1] + A^0[1,2])$$
$$= \min(\infty,\ 5+3)$$
$$= 8$$

$$A^1[3,4] = \min(A^0[3,4],\ A^0[3,1] + A^0[1,4])$$
$$= \min(1,\ 5+7)$$
$$= 1$$

$$A^1[4,2] = \min(A^0[4,2],\ A^0[4,1] + A^0[1,2])$$
$$= \min(\infty,\ 2+3)$$
$$= 5$$

$$A^1[4,3] = \min(A^0[4,3],\ A^0[4,1] + A^0[1,3])$$
$$= \min(\infty,\ \infty)$$
$$= \infty$$

$$\Rightarrow A^1 = \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & \infty & 0 \end{bmatrix}$$

$$A^2[1,3] = \min(A^1[1,3],\ A^1[1,2] + A^1[2,3])$$
$$= \min[\infty,\ 3+2]$$
$$= 5$$

$A^2[1,4] = \min[A^1[1,4], A^1[1,2] + A^1[2,4]]$
$= \min(7, 3+15)$
$= 7$

$A^2[3,1] = \min(A^1[3,1], A^1[3,2] + A^1[2,1])$
$= \min(5, 8+8)$
$= 5$

$A^2[3,4] = \min(A^1[3,4], A^1[3,2] + A^1[2,4])$
$= \min(1, 8+15)$
$= 1$

$A^2[4,1] = \min(A^1[4,1], A^1[4,2] + A^1[1,2])$
$= \min(2, 5+8)$
$= 2$

$A^2[4,3] = \min(A^1[4,3], A^1[4,2] + A^1[2,3])$
$= \min(\infty, 5+2)$
$= 7$

$$\Rightarrow A^2 = \begin{bmatrix} 0 & 3 & 5 & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix}$$

$A^3[1,2] = \min(A^2[1,2], A^2[1,3] + A^2[3,2])$
$= \min(3, 5+8)$
$= 3$

$A^3[1,4] = \min(A^2[1,4], A^2[1,3] + A^2[3,4])$
$= \min(7, 5+1)$
$= 6$

$$A^3[2,1] = \min(A^2[2,1], A^2[2,3] + A^2[3,1])$$
$$= \min(8, 2+5)$$
$$= 7$$

$$A^3[2,4] = \min(A^2[2,4], A^2[2,3] + A^2[3,4])$$
$$= \min(15, 2+1)$$
$$= 3$$

$$A^3[4,1] = \min(A^2[4,1], A^2[4,3] + A^2[3,1])$$
$$= \min(2, 7+5)$$
$$= 2$$

$$A^3[4,2] = \min(A^2[4,2], A^2[4,3] + A^2[3,2])$$
$$= \min(5, 7+8)$$
$$= 5$$

$$A^3 = \begin{bmatrix} 0 & 3 & 5 & 6 \\ 7 & 0 & 2 & 3 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix}$$

$$A^4[1,2] = \min(A^3[1,2], A^3[1,4] + A^3[4,2])$$
$$= \min(3, 6+5)$$
$$= 3$$

$$A^4[1,3] = \min(A^3[1,3], A^3[1,4] + A^3[4,3])$$
$$= 5$$

$$A^4[2,1] = \min[A^3[2,1], A^3[2,4] + A^3[4,1]]$$
$$= \min(7, 3+2)$$
$$= 5$$

$$A^4[2,3] = \min[A^3[2,3], A^3[2,4] + A^3[4,3]]$$
$$= \min(2, 3+0)$$
$$= 2$$

$$A^4[3,1] = \min[A^3[3,1], A^3[3,4] + A^3[4,1]]$$
$$= \min(5, 1+2)$$
$$= 3$$

$$A^4[3,2] = \min[A^3[3,2], A^3[3,4] + A^3[4,2]]$$
$$= \min(8, 1+5)$$
$$= 6$$

$$A^4 = \begin{bmatrix} 0 & 3 & 5 & 6 \\ 5 & 0 & 2 & 3 \\ 3 & 6 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix}$$

Algorithm of Floyd Warshall :-

```
for (k=1; k<=n; k++) {
    for(i=1; i<=n; i++) {
        for(j=1; j<=n; j++) {
            A^k[i,j] = min(A^(k-1)[i,j], A[i,k]+A[k,j])
```

3

3 3

floydWarshall Σ

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

Complexity = $O(n^3)$