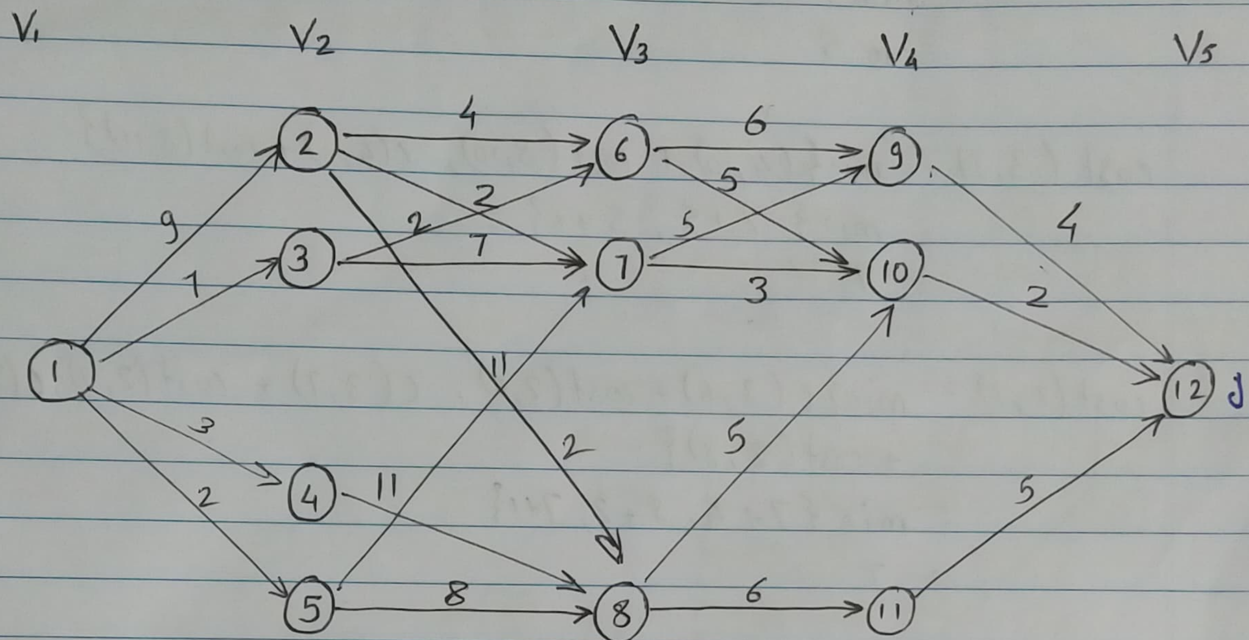
Meet Raut.
S21 84

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

# Assignment 2

Q.1)

Ans.




Ans. Formula : $cost(i,j) = \min\{c(j,l) + cost(i+1,l)\}$

| V | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| cost | 16 | 7 | 9 | 18 | 15 | 7 | 5 | 7 | 4 | 2 | 5 | 0 |
| d | 2/3 | 7 | 6 | 8 | 8 | 10 | 10 | 10 | 12 | 12 | 12 | 12 |

$$cost(3,6) = \min\{(4,9) + cost(6,9),\ c(4,10) + cost(6,10)\}$$
$$= \min\{4+6,\ 2+5)$$
$$= \min\{10,\ 7\}$$
$$= 7$$

$$\text{cost}(3,7) = \min\{c(4,9) + \text{cost}(7,9),\ c(4,10) + \text{cost}(7,10)\}$$
$$= \min\{4+4,\ 2+3\}$$
$$= 5$$

$$\text{cost}(3,8) = \min\{c(4,10) + \text{cost}(8,10),\ c(4,11) + \text{cost}(8,11)\}$$
$$= \min\{2+5,\ 5+6\}$$
$$= 7$$

$$\text{cost}(2,2) = \min\{c(3,6) + \text{cost}(2,6),\ c(3,7) + \text{cost}(2,7),\ c(3,8) + \text{cost}(2,8)\}$$
$$= \min\{7+4,\ 5+2,\ 7+1\}$$
$$= 7$$

$$\text{cost}(2,3) = \min\{c(3,6) + \text{cost}(3,6),\ c(3,7) + \text{cost}(3,7)\}$$
$$= \min\{7+2,\ 5+1\}$$
$$= 9$$

$$\text{cost}(2,4) = \min\{c(3,8) + \text{cost}(4,8)\}$$
$$= \min\{7+11\}$$
$$= 18$$

$$\text{cost}(2,5) = \min\{c(3,7) + \text{cost}(5,7),\ c(3,8) + \text{cost}(5,8)\}$$
$$= \min\{5+11,\ 7+8\}$$
$$= 15$$

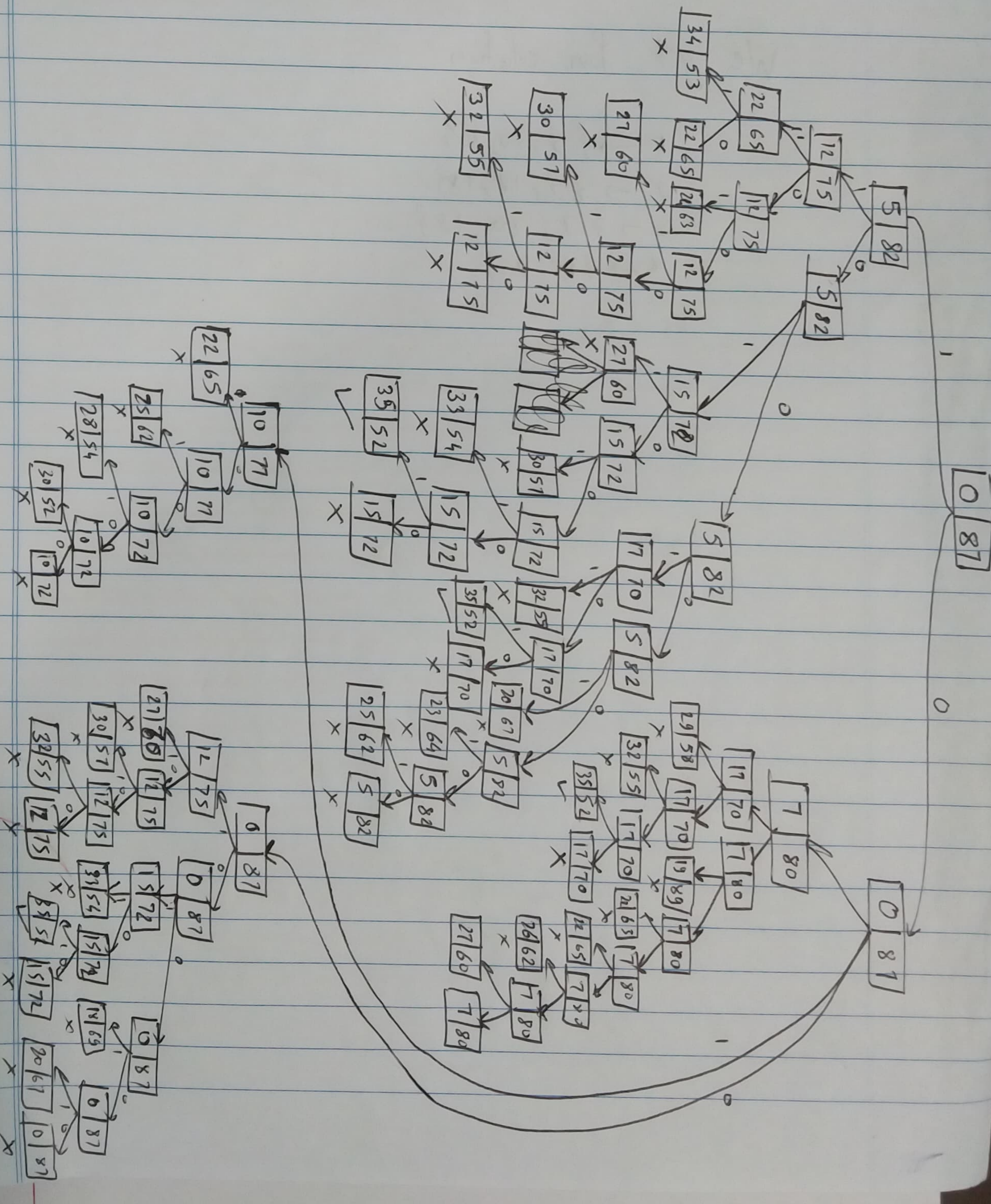
$$\text{cost}(1,1) = \min\{c(2,2) + \text{cost}(1,2),\ c(2,3) + \text{cost}(1,3),\ c(2,4) + \text{cost}(1,4),\ c(2,4) + \text{cost}(1,4),\ c(2,5) + \text{cost}(1,5)\}$$
$$= \min\{7+9,\ 9+7,\ 18+3,\ 15+2\}$$
$$= 16$$

Shortest Path: $1 \to 2 \to 7 \to 10 \to 12$ & $1 \to 3 \to 6 \to 10 \to 12$

Q. 2)

Ans. $M = 35$

$W = (5, 7, 10, 12, 15, 18, 20)$



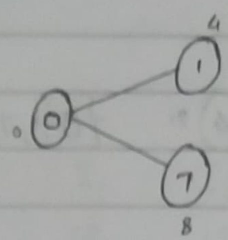
At each node end, 'x' means it cannot accommodate any of next value, so we cut-sort them for further expansion and 'v' means it is the solution states.

We get four solutions

1 → {5, 10, 20}
2 → {5, 12, 18}
3 → {7, 10, 18}
4 → {15, 20}

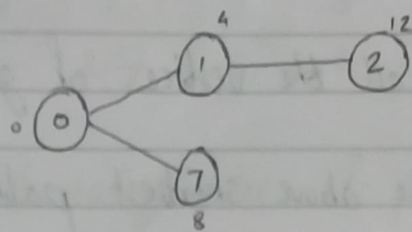
Q.3)

Ans. Step 1:



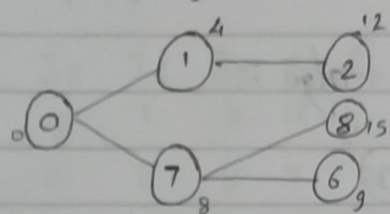

{0}

Step 2:



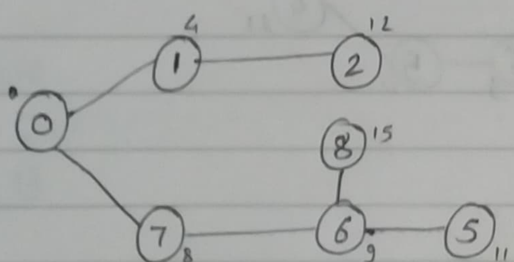

{0, 1}

Step 3:



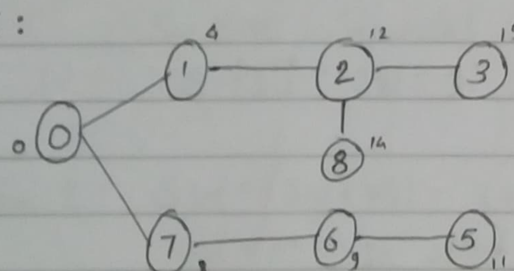

{0, 1, 7}

Step 4:



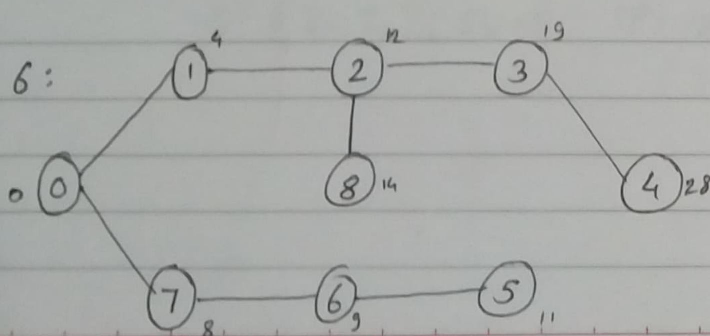

{0, 1, 7, 6}

Step 5:



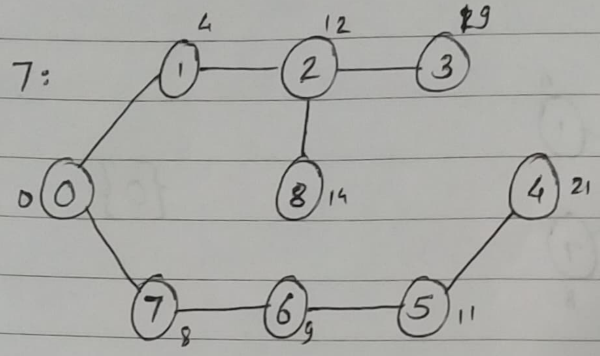

{0, 1, 7, 6, 2}

Step 6:




{0, 1, 7, 6, 2, 8, 3}

Step 7:

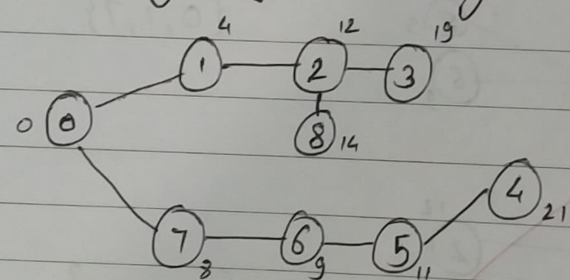



{0, 1, 7, 6, 8, 3, 5, 4}

Spt set includes all the vertices of given graph.

∴ Finally we get the above shortest path tree.

SPT using Djikstra's algo:



Q.4)
Ans. Initially :

| Item ($x_i$) | $I_1$ | $I_5$ | $I_4$ | $I_3$ | $I_6$ | $I_2$ |
|---|---|---|---|---|---|---|
| Profit ($V_i$) | 18 | 12 | 10 | 9 | 7 | 5 |
| Weight ($W_i$) | 7 | 3 | 5 | 3 | 2 | 2 |
| $P_i = V_i/W_i$ | 2.57 | 4 | 2 | 3 | 3.5 | 2.5 |

$\therefore$ decreasing order of profit / Weight value table.

| $x_i$ | $I_5$ | $I_6$ | $I_3$ | $I_1$ | $I_2$ | $I_4$ |
|---|---|---|---|---|---|---|
| $V_i$ | 12 | 7 | 9 | 18 | 5 | 10 |
| $W_i$ | 3 | 2 | 3 | 7 | 2 | 5 |
| $P_i$ | 4 | 3.5 | 3 | 2.57 | 2.5 | 2 |

$SW = 10$
$SP = 0$
$M = 13$

Iteration 1 : $SW = (SW + W_5) = 0 + 3 = 3$
$SW \leq M$, So select $I_5$
$\therefore S = \{I_5\}$ ; $SW = 3$; $SP = 0 + 12 = 19$

Iteration 2 : $SW = (SW + W_6) = 3 + 2 = 5$
$SW \leq M$ ; So select $I_6$
$S = \{I_5, I_6\}$ ; $SW = 5$ ; $SP = 12 + 7 = 19$

Iteration 3 : $SW = (SW + W_3) = 5 + 3 = 8$

$SW < M$, so select $I_3$

$S = \{I_5, I_6, I_3\}$; $SW = 8$; $SP = 19 + 9 = 28$

Iteration 4 : $SW + W_1 > M$, so break down item $I_1$

The remaining capacity of knapsack is 5, so select only 5 items of $I_1$.

$$frac = ((M - SW) / W[i]) = \frac{13 - 8}{7} = \frac{5}{7}$$

$$S = \{I_5, I_6, I_3, I_1 * 5/7\}$$

$$SP = SP + V_i * 5/7 = 28 + (18 * 5/7) = 28 + 12.857$$

$$SP = 40.85$$

$$SW = SW + W_1 * 5/7 = 8 + (7 * 5/7) = 8 + 5 = 13$$

∴ ~~Keep~~ knapsack is full.

∴ Fractional knapsack select items $\{I_5, I_6, I_3, I_1 * 5/7\}$ and it gives total profit of 40.85 units.

Q.5)
Ans. Algorithm :

```
Algorith N-Queen (k, n)
// Input : n = no. of queen, k = No. of queen being processed currently
// Output : n x 1 solution tuple.

  for i <- 1 to n do
      if PLACE (k, i) then
         x[k] <- i
         if k == n then
            print x [1...N]
         else
            N-Queen (k+1, n)
         end
      end
  end

function Place (k, i)
   for j <- 1 to k-1 do
      if x[j] == i OR ((abs(x[j] - i) == abs(j-k)) then
         return false
      end
   end
return true
```

# 4-Queen problem:

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

4x4 chessboard

putting $Q_1$: It can be placed anywhere.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | $Q_1$ | X | X | X |
| 1 | X | X | | |
| 2 | X | | X | |
| 3 | X | | | X |

putting $Q_2$: Placing at the non-attacked places.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | $Q_1$ | X | X | X |
| 1 | X | X | X | $Q_2$ |
| 2 | X | | X | X |
| 3 | X | X | | X |

putting $Q_3$: placing at the non-attacked places.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | $Q_1$ | X | X | X |
| 1 | X | X | X | $Q_2$ |
| 2 | X | $Q_3$ | X | X |
| 3 | X | X | X | X |

There is no place for $Q_4$.

By making some adjustment we get,

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | X | $Q_1$ | X | X |
| 1 | X | X | X | $Q_2$ |
| 2 | $Q_3$ | X | X | X |
| 3 | X | X | $Q_4$ | X |

Therefore, through backtracking we reached a solution where 4 queens are put in each row & column so that no queen is attacking any other on a 4x4 chessboard.

The two solution for $n=4$ queens: $(1,3,0,2)$ & $(2,0,3,1)$

Q.6)

Ans. $X$ = ababcde

$Y$ = bacadb

If $x[i] == y[j]$

$c[i][j] = 1 + [i-1, j-1]$

else

$c[i][j] = Max\{c[i-1, j], c[i, j-1]\}$

$Y \rightarrow$

$X \downarrow$

| - | ∧ | b | a | c | a | d | b |
|---|---|---|---|---|---|---|---|
| ∧ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a | 0 | ←0 | ↖1 | ←1 | ↖1 | ←1 | ←1 |
| b | 0 | ↖1 | ←1 | ←1 | ←1 | ←1 | ↖2 |
| a | 0 | ↑1 | ↖2 | ←2 | ↖2 | ←2 | ←2 |
| b | 0 | ↖1 | ↑2 | ←2 | ←2 | ←2 | ↖3 |
| c | 0 | ↑1 | ↑2 | ↖3 | ←3 | ←3 | ←3 |
| d | 0 | ↑1 | ↑2 | ↑3 | ←3 | ↖4 | ←4 |
| e | 0 | ↑1 | ↑2 | ↑3 | ←3 | ↑4 | ←4 |

LCS = dcab

Page No.
Date / /

Q. 7)
Ans.

| Prim's algo | Kruskal's algo |
|---|---|
| 1. Vertex based algorithm | 1. Edge-based algorithm. |
| 2. Time complexity is $O(V^2)$ | 2. Time complexity is $O(E+V\log V)$ |
| 3. It tends to perform better on dense graphs | 3. It tends to perform better on sparse graphs. |
| 4. It commonly uses a priority queue data structure to effectively select the next edge to add. | 4. It typically uses disjoint set data structures to efficiently check for cycles and maintain connected components. |