16/09/19

# Module- 3

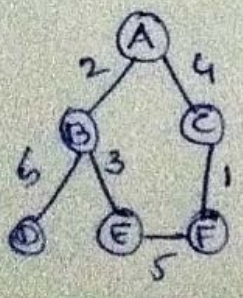
## Greedy Method

All problem have n inputs and obtains subset of output that satisfies some constraints (conditions). Any subset that satisfies these constraints is called feasible sol. We need to find a feasible sol. that maximizes or minimizes the given objective fun.

At each step in Greedy problem solving approach, making locally selection of elements with the help of finding a global optimum. Making locally choosing value cannot get optimum global value. Two properties followed by Greedy method:

(i) Greedy choice :- A global optimum that can be arranged by choosing a local optimum

(ii) optimal sub structure :-




An optimum sol. to the problem contains an optimum subset of elements.

Our objective here is minimisation.

Aim : to find f.

Two subset of sols : (A-B-E-F), (A-C-F)

(A-B-E-F) → minimum cost for getting F is 2, 3, 5

A-C-F → cost is 4, -1

A-C-F is global one and greedy chooses A-C-F.

Three different problems based on Greedy Method.

(i) Knapsack problem

a) Fractional knapsack

b) zero-one knapsack

(ii) Job scheduling problem

(iii) Finding shortest path using Minimum spanning Tree (MST)

a) prim's algorithm

b) Kruskal's algorithm

## Control Abstraction of Greedy Method

```
Sol Type ( Type a[] , int n )
// a[1:n] contains the n inputs
{
    Sol Type solution = EMPTY; // Initialize the solution
    for (int i = 1, i <= n; i++ ) {
        Type x = select (a);
        if feasible (solution , x )
            solution = Union (solution , x );
    }
    return solution;
}
```

17/9/19

## I. Knapsack problem

In this problem, we have a knapsack that have a weight limit 'w'. There are items $x_1, x_2, \ldots, x_n$ each having weight $w_1, w_2, \ldots, w_n$ and have the profit/value/benefit $p_1, p_2, \ldots, p_n$. In knapsack problem the objective function is to maximize the benefit such that the total weight inside the knapsack is atmost w.

Maximize $\sum_{1 \le i \le n} p_i x_i$ ——— (1)

subject to,

$$\sum_{1 \le i \le n} w_i x_i \le m$$

Two types of knapsack problems

(i) Zero-one knapsack : ie, each items is taken or not taken.

(ii) Fractional knapsack: can take fraction of items.

Steps

(i) calculate density, density = $\frac{\text{value}}{\text{weight}}$ for each items.

(ii) sort the element or items, as per value density in descending order.

(iii) Take as much item as possible not already taken in the knapsack.

1. consider the weights and benefits of following elements.

| item | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| weight | 10 | 15 | 3 | 5 |
| profit | 200 | 210 | 150 | 180 |

Sol: step 1 : density of each item $\Rightarrow \frac{p_i}{w_i}$

Item 1 : $\frac{200}{10} = 20$

2 : $\frac{210}{15} = 14$

3 : $\frac{150}{3} = 50$

4 : $\frac{180}{5} = 36$

**Step 2:** Arrange the density in descending order.

Density : 50 36 20 14

Draw the table.

| Item | 3 | 4 | 1 | 2 |
|---|---|---|---|---|
| weight | 3 | 5 | 10 | 15 |
| Density | 50 | 36 | 20 | 14 |
| Profit | 150 | 180 | 200 | 210 |

Step 3: Maximum weight can be holded = 25

i.e, M = 25.

M - W

1st object : 25 - 3 = 22

2nd obj : 22 - 5 = 17

3rd obj : 17 - 10 = 7

4th obj : $7 - \frac{7}{15} \times 15 = 0$. [Fractional].

Profit

P = 0, initially

: 0 + 150 = 150

150 + 180 = 330

330 + 200 = 530

$530 + \frac{7}{15} \times 210 = 530 + 98 = 628$

Benefit = $\frac{\text{weight taken}}{\text{Total weight}} \times \text{profit}$

## Control Abstraction for knapsack problem

```
void GreedyKnapsack (float m, int n)
// p[1:n] and w[1:n] contain the profit
                    and weights
// respectively of the n objs.
// ordered such that p[i]/w[i] >= p[i+1]/
                                  w[i+1]
// m is the knapsack size and x[1:n]
          is the solution vector.
{
  for (int i=1; i<=n; i++) x[i] = 0.0; // initialize x
  float U = m;
  for (i=1; i<=n; i++) {
  if (w[i] > U) break;
    x[i] = 1.0;
    U -= w[i];
  }
  if (i<=n) x[i] = U/w[i];
}
```

eg: $(x_1, x_2, x_3)$ $(w_1, w_2, w_3) = (5, 10, 15)$

$n = 3$, $W = 20$.

$u = m$

$u = 20$.

1st iteration:

$i = 1$, $1 \leq 3$, yes

$w[i] > U$, $w[1] > 20 \Rightarrow 5 > 20$, NO

$x[1] = 1$

$U = U - w[i] \Rightarrow 20 - 5 = 15$

$\therefore U = 15$

IInd iteration:

$U = 15$

$i = 2$, $2 <= 3$ yes

$w[2] > U \Rightarrow 10 > 15$, ~~yes~~ NO,

$x[2] = 1$

$U = 15 - 10 = 5$

$\therefore U = 5$

IIIrd iteration:

$U = 5$

$i = 3$, $3 \leq 3$, yes

$w[3] > 5 \Rightarrow 15 > 5$, yes, break.

~~$i \leq n$~~ $i <= n \Rightarrow 3 \leq 3$, yes.

$x[3] = 5 / w[3] \Rightarrow 5/15 = 1/3$

ie, $5 \times \frac{1}{3} = \underline{\underline{5}}$

19/9/19

* Consider the following problems.

| Item | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ |
|---|---|---|---|---|---|
| W | 5 | 7 | 2 | 3 | 4 |
| P | 6 | 14 | 6 | 9 | 5 |

Assume Maximum weight of knapsack W=15.

1 : $\frac{6}{5} = 1.2$

sol: Ratio: $\frac{P_i}{W_i} = \frac{14}{7} = 2$

3 : $\frac{6}{2} = 3$

6 : $\frac{9}{3} = 3$

5 : $\frac{5}{4} = 5/4 = 1.25$

$4\overline{)5}$ 1.25, 4, 10, 8, 20, 20

$5\overline{)6}$ 1.2, 5, 10

In descending order:

3 3 1.25 2 1.25 1.2

| Item | 3 | 4 | 2 | 5 | 1 |
|---|---|---|---|---|---|
| weight | 2 | 3 | 7 | 4 | 5 |
| Density | 3 | 3 | 2 | 1.25 | 1.2 |
| Profit | 6 | 9 | 14 | 5 | 6 |

M = 15

1st object = 15 - 2 = 13

IInd = 13 - 3 = 10

= 10 - 7 = 3

= 3 - 3/4 x 4 = 0

Profit,

P = 0,

0 + 6 = 6

6 + 9 = 15

15 + 14 = 29

$29 + \frac{3}{4} \times 5 = 29 + 3.75 = 32.75$

$29 + \frac{15}{4}$

$[I_3, I_4, I_2, I_5]$

* Find the optimal solution of knapsack instance.

n = 7, m = 15, w = {2, 3, 5, 7, 1, 4, 1}

sol: P = {10, 5, 15, 7, 6, 18, 3}

Density: $\frac{10}{2} = 5$

$2 : \frac{5}{3} = 1.66$

$3 : \frac{15}{5} = 3$

$4 : 7/7 = 1$

$5 : \frac{6}{1} = 6$

$6 : \frac{18}{4} = 4.5$

$7 : \frac{3}{1} = 3$

| Density : | 6 | 5 | 4.5 | 3 | 3 | 1.66 | 1 |
|---|---|---|---|---|---|---|---|
| | $I_5$ | 1 | 6 | 3 | 7 | 2 | 4 |

| Item | 5 | 1 | 6 | 3 | 7 | 2 | 4 |
|---|---|---|---|---|---|---|---|
| weight | 1 | 2 | 4 | 5 | 1 | 3 | 7 |
| Density | 6 | 5 | 4.5 | 3 | 3 | 1.66 | 1 |
| Profit | 6 | 10 | 18 | 15 | 3 | 5 | 7 |

M = 15

1st : 15 − 1 = 14

2nd 14 − 2 = 12

12 − 4 = 8

8 − 5 = 3

3 − 1 = 2

$2 - \frac{2}{3} \times 3 = 0$

P = 0

0 + 6 = 6

6 + 10 = 16

16 + 18 = 34

34 + 15 = 49

49 + 3 = 52

$52 + \frac{2}{3} \times 5 =$

$52 + \frac{10}{3} =$

55.33

Items: $[I_5, I_1, I_6, I_3, I_7, I_2]$

* Consider the following instance of knapsack problem,

$n = 3$, $M = 20$, $P_1, P_2, P_3 = \{25, 24, 15\}$

and $w_1, w_2, w_3 = \{18, 15, 10\}$. Find the feasible sol:

$D1 = \frac{25}{18} = 1.3$

$2 = \frac{24}{15} = 1.6$

$3 = \frac{15}{10} = 1.5$

Descending : 1.6 1.5 1.3

| | | | |
|---|---|---|---|
| I | 2 | 3 | 1 |
| w | 15 | 10 | 18 |
| D | 1.6 | 1.5 | 1.3 |
| P | 24 | 15 | 25 |

$M = 20$ $P = 0$

I st : $20 - 15 = 5$ $0 + 24 = 24$

$5 - \frac{5}{10} \times 10 = 0$ $24 + \frac{5}{10} \times 15 = 31.5$

subset = $\{2, 3\}$

* Consider the following instance of knapsack problem,

$P = \{5, 10, 15, 7, 8, 9, 4\}$

$W = \{1, 3, 5, 4, 1, 3, 2\}$

$M = 15$

$D_1 = \frac{5}{1} = 5$

$D_2 = \frac{10}{3} = 3.33$

$D_3 = \frac{15}{5} = 3$

$D_4 = \frac{7}{4} = 1.7$

$5 = \frac{8}{1} = 8$

$6 = \frac{9}{3} = 3$

$7 = \frac{4}{2} = 2$

Density : 8 5 3.33 3 3 2 1.7

(5) (1) (2) (3) (6) (7) (4)

| | | | | | | |
|---|---|---|---|---|---|---|
| I | 5 | 1 | 2 | 3 | 6 | 7 | 4 |
| W | 1 | 1 | 3 | 5 | 3 | 2 | 4 |
| D | 8 | 5 | 3.3 | 3 | 3 | 2 | 1.7 |
| P | 8 | 5 | 10 | 15 | 9 | 4 | 4 |

M = 15

15 - 1 = 14

14 - 1 = 13

13 - 3 = 10

10 - 5 = 5

5 - 3 = 2

2 - $\frac{2}{2}$ = 0

P = 0

0 + 8 = 8

8 + 5 = 13

13 + 10 = 23

23 + 15 = 38

38 + 9 = 47

47 + $\frac{2}{2}$ x 4 = 51

---

* Consider the following instance of knapsack problems:

(i) M = 25

w = {3, 5, 8, 2, 13, 14, 11, 10}

p = {10, 28, 56, 32, 15, 23, 19, 10}

(ii) w = {14, 3, 2, 10}

p = {16, 7, 6, 5}

M = 25

Solution

$D_1 = \frac{10}{3} = 3.3$

$2 = \frac{28}{5} = 5.6$

3. $\frac{56}{8} = 7$

4. $\frac{32}{2} = 16$

5. $\frac{15}{13} = 1.15$

6. $\frac{23}{14} = 1.6$

7. $\frac{19}{11} = 1.7$

8 = $\frac{10}{10} = 1$

Density: 16 7 5.6 3.3 1.7 1.6 1.15 1
(4) (3) (2) (1) (7) (6) (5) (8)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| I | 4 | 3 | 2 | 1 | 7 | 6 | 5 | 8 |
| w | 2 | 8 | 5 | 3 | 11 | 14 | 13 | 10 |
| D | 16 | 7 | 5.6 | 3.3 | 1.7 | 1.6 | 1.15 | 1 |
| p | 32 | 56 | 28 | 10 | 19 | 23 | 15 | 10 |

M = 25

P = O = M

1 : 25 - 2 = 23

23 - 8 = 15

15 - 5 = 10

0 + 32 = 32

32 + 56 = 88

88 + 28 = 116

$10 - 3 = 7$

$7 - \frac{7}{11} \times 11 = 0$

$116 + 10 = 126$

$126 + \frac{7}{11} \times 19 = 138.09$

Items : $I_4, I_3, I_2, I_1, I_7$.

(ii) $D_1 : \frac{\overset{8}{\cancel{16}}}{\cancel{7}\,14} = 1.1$

$2 : \frac{7}{3} = 2.3$

$3 : \frac{6}{2} = 3$

$4 : \frac{5}{10} = 0.5$

Density : 3 (3) | 2.3 (2) | 1.1 (1) | 0.5 (4)

| | | | | |
|---|---|---|---|---|
| I | 3 | 2 | 1 | 4 |
| W | 2 | 3 | 14 | 10 |
| D | 3 | 2.3 | 1.1 | 0.5 |
| P | 6 | 7 | 16 | 5 |

$M = 25$

$1 : 25 - 2 = 23$

$23 - 3 = 20$

$20 - 14 = 6$

$6 - \frac{6}{10} \times 10 = 0$

P = 0

0 + 6 = 6

6 + 7 = 13

13 + 16 = 29

$29 + \frac{8^3}{2^{10}} \times 8 = 32$

Items: $\{I_3, I_2, I_1, I_4\}$

20/9/19

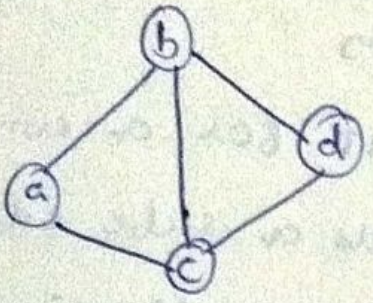
## Spanning Tree

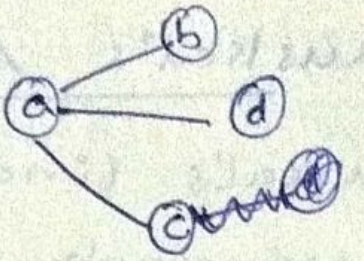
Given $G_1 = \{V, E\}$ where $V = \{V_1, V_2, V_3, \ldots V_n\}$

$E = \{E_1, E_2, \ldots E_n\}$.

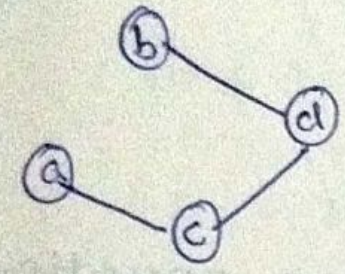
$E_{ij} = (V_i, V_j)$ where $V_i$ and $V_j$ are adjacent vertex.

Let $G_1 = (V, E)$ be an undirected connected graph. A subgraph $T = \{V', E'\}$ of $G_1$ is a spanning tree of $G_1$, if and only if T is a tree. We can construct any no. of spanning tree from a graph. A spanning tree should have n vertex and some $(n-1)$ edges without any cycle.

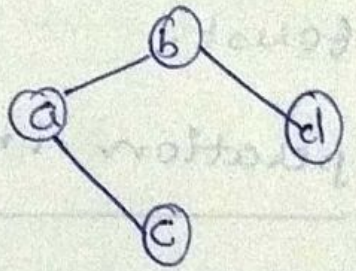



$sp_1$




$sp_2$




$sp_3$

If the graph is weighted, then a spanning tree with minimum cost is called

conditions of spanning tree

(i) $V' = V$, ie, no. of vertices is equal in main and sub graphs.

(ii) $E' \subset E-1$ ie, E' is the subset of E if n edges in E, then (n-1) edges in E'.

(iii) without any cycle or closed path.

S is the subset of G, where $S = \{V', E'\}$

$S \subseteq h$.

# Kruskal's Algorithm

Kruskal's find a MST for a connected weighted graph, it finds a safe edge, to add the growing forest by finding all the edges that connect any two trees in the forest.

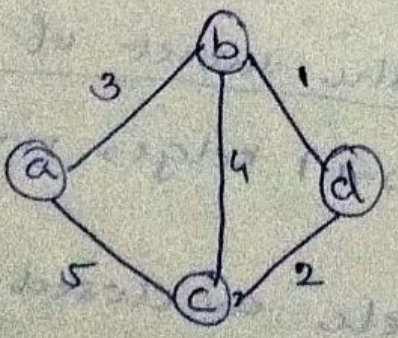
## 3 operation in kruskal

(i) MAKE-SET : The MAKE-SET operation determine or arrange, each tree in non-decreasing order.

(ii) FIND-SET : Find-set operation determine two vertices U and V belong to same tree by testing whether,

Find-set U = Find-set V.

(iii) To combine trees Kruskal's call a fun. known as UNION.



step 1: Use MAKE-SET operation and write in non-decreasing order.

(b, d) = 1

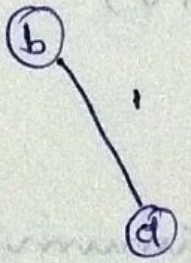
(c, d) = 2

(a, b) = 3

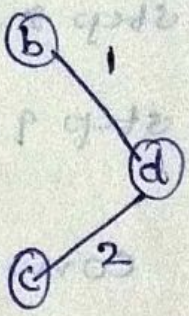
(b, c) = 4

(a, c) = 5

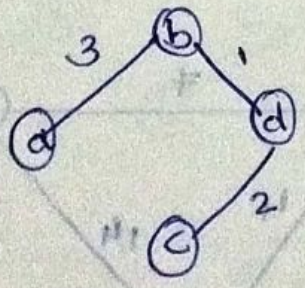
steps: (i)




step: ii




(iii)




∴ Minimum cost = 1 + 2 + 3 = 6

## Algorithm

MST - Kruskal (G, w)

step 1: A = ϕ

step 2 : for each vertex $V \in V[G]$

step 3 : MAKE-SET(v)

step 4 : sort the edges into non-decreasing order by weight.

step 5 : for each edge (u, v) = G, taken in non-decreasing order.

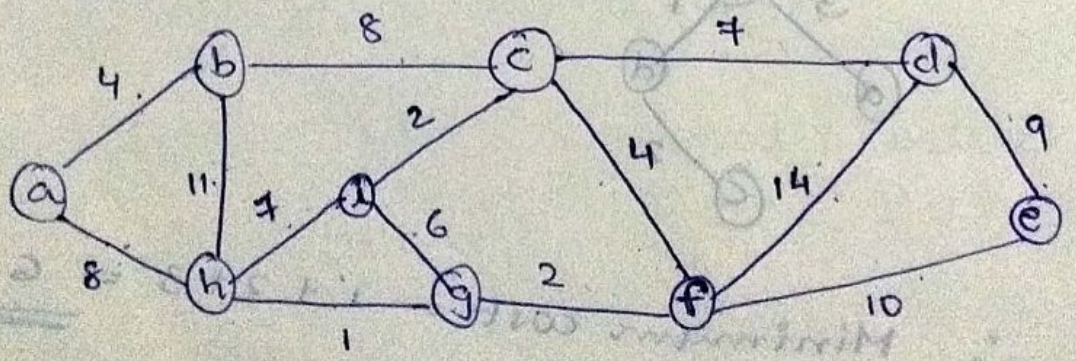
step 6 : If FIND-SET(u) ≠ FIND-SET(v)

step 7 : A = A ∪ {u, v}

step 8 : union (u, v)

step 9 : return A

? Construct minimum cost spanning tree for the following graph using kruskal's method.




MAKE-SET

(h, g) - 1

(g, f) - 2

(i, c) - 2

(a, b) - 4

(c, f) - 4

(i, g) - 6

(h, i) - 7

(c, d) - 7

(a, h) - 8

(b, c) - 8

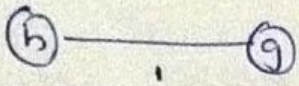
(d, e) - 9

(f, e) - 10

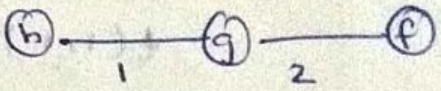
(b, b) - 11

(f, d) - 14

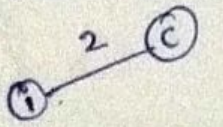
step:(i)

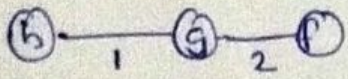



step:ii

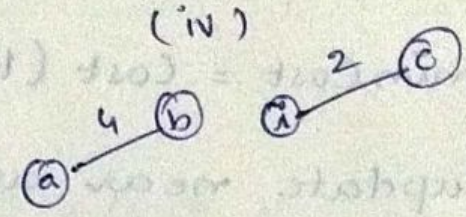



step:(iii)



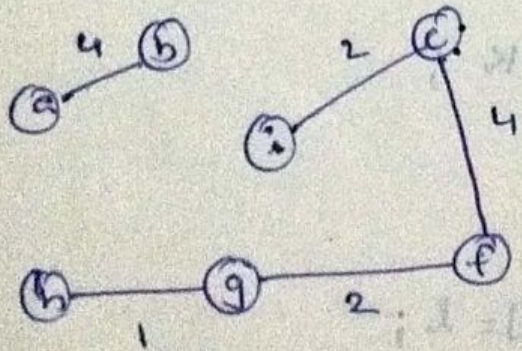




(iv)

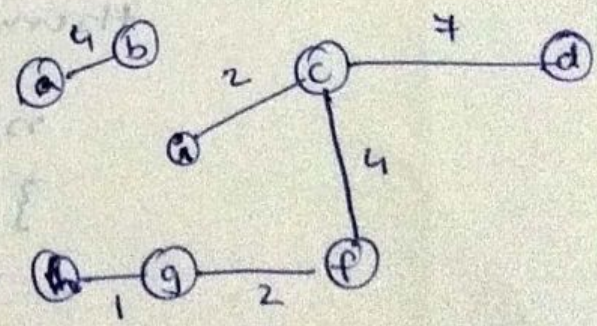


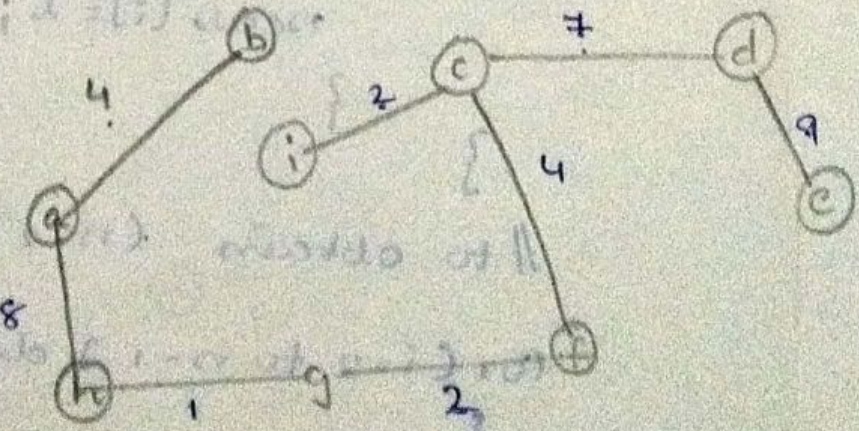


(v)




(vi)




Total cost = 37

23

27/9/19

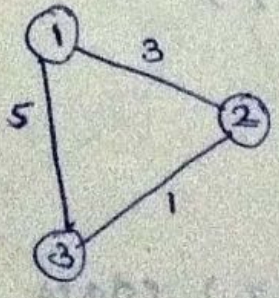
# Prim's Algorithm

```
Algo prims (cost, E, T, n)
{
  Let (k, l) is an edge with minimum
   cost in set of edges 'E'.
   t(1,1) = k;
   t(1,2) = l;
   minCost = Cost (k, l) // min cost
  //update near array
   for i = 1 to n do
   {
    if Cost (i, k) < cost (i, l)
      then {
        near [i] = k;
      }
      else {
        near [i] = l;
      }
   }
  // to obtain (n-2) edges
   for (i = 2 to n-1) do
   {
     Let j be an index such that
```

near[j] ≠ 0 and cost[j, near[j]] is minimum.

```
t[i, 1] = j;
t[i, 2] = near[j];
mincost = mincost + cost[j, near[j]);
near[j] = 0;
// update near array loop
for k = 1 to n
 {
   do
     if near[k] ≠ 0 && cost[k, near[k]) > cost[k, j]
                                          then
     {
       near[k] = j;
     }
   }
 }
 return mincost;
}
```

eg:



Adjacency matrix:

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0 | 3 | 5 |
| 2 | 3 | 0 | 1 |
| 3 | 5 | 1 | 0 |

Let (1,2) be the edge with min cost 3 from vertex 1.

tree:

| $V_1$ | $V_2$ |
|---|---|
| | |

$t(1,1) = K \Rightarrow 0$

$t(1,2) = 1 \Rightarrow 2$

$[0 < 2 , \text{True}]$

min cost = 3

update near array:

near[i] ; i = 1

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

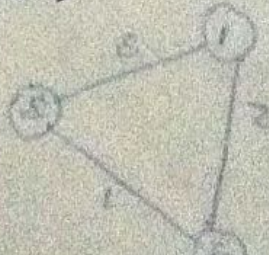
i = 1 to 3

near[1] = 1

c(2,1) < c(2,2)

3 < 0 , false

near = 2




// to obtain (n-2) edges

10/10/19

j = 2 to n - 1

findout j , 2 to 2

near[j] ≠ 0

cost[j], near[j] is minimum

t[i,1] = j

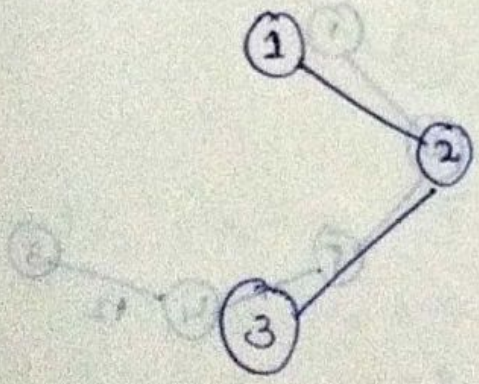
t[i,2] = near[j].

(1,5) => 1, minimum.

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 3 | 2 |
| 5 | 1 | 3 |

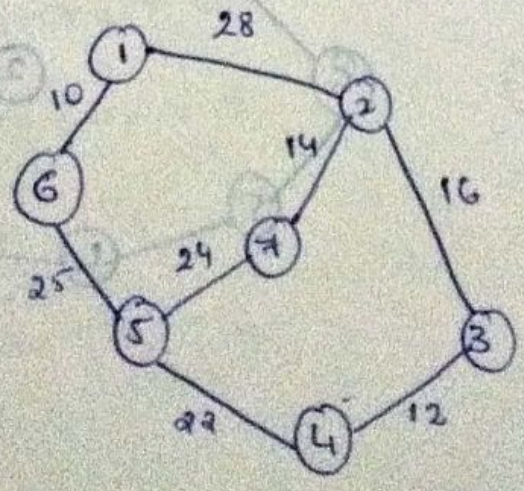
| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 0 | 2 |
| 5 | 1 | 3 |

| $V_1$ | $V_2$ |
|---|---|
| 1 | 2 |
| 2 | 3 |



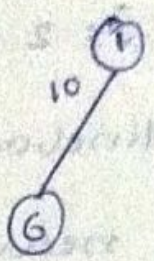

11/10/19

→



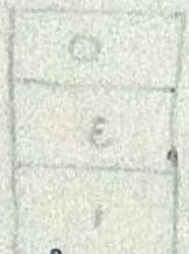
step 1: $1 \to 6 = 10$ , Min

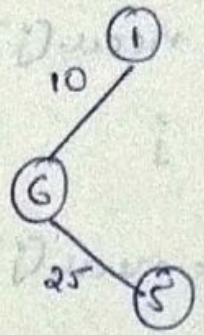
$1 \to 2 = 28$




2: $1 \longrightarrow 2 = 28$

$6 \to 5 = 25$

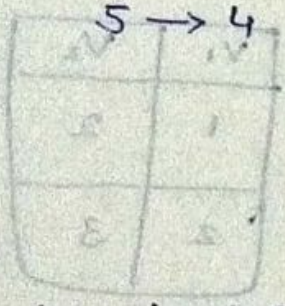


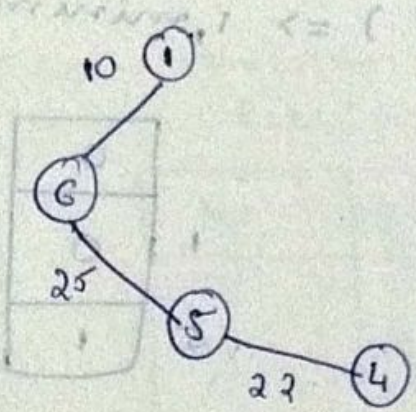



3: $1 \longrightarrow 2 = 28$

$5 \to 7 = 24$

$5 \to 4 = 22$

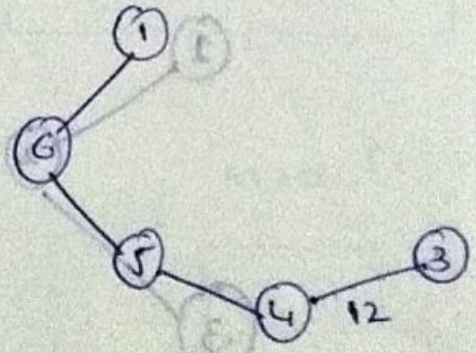





4: $1 \longrightarrow 2 = 28$

$5 \to 7 = 24$

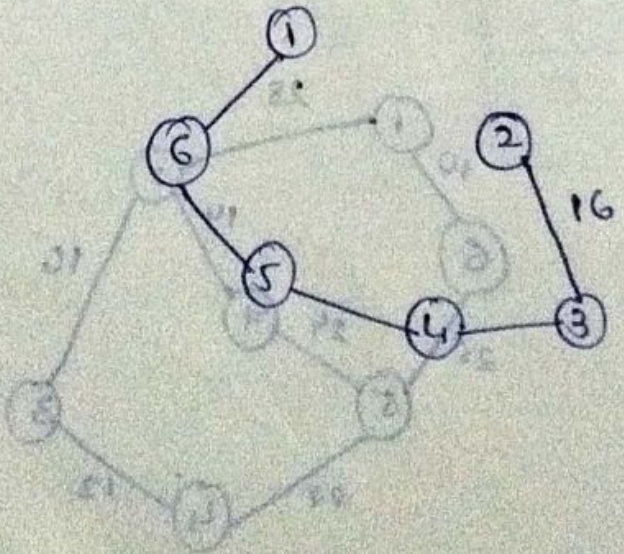
$6 \to 6$ $4 \to 3 = 12$




5: $1 - 2 = 28$

$5 - 7 = 24$

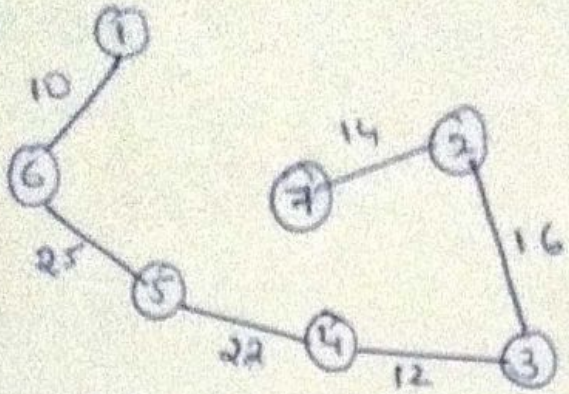
$3 - 2 = 16$




6: $1 - 2 = 28$

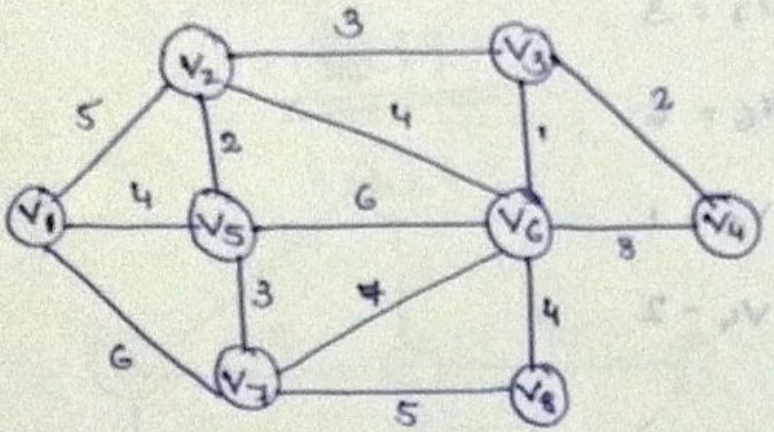
$5 - 7 = 24$

$2 - 7 = 14$




Total cost = 99

→ Compute minimum cost using prim's Method.




1: $V_1 - V_2 = 5$

$V_1 - V_3 = 6$

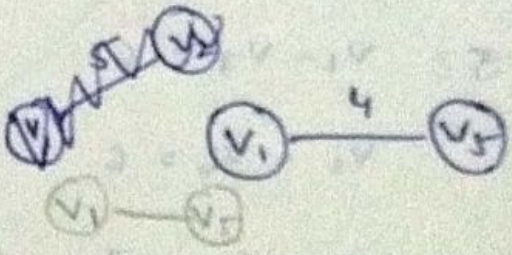
$V_1 - V_5 = 4$

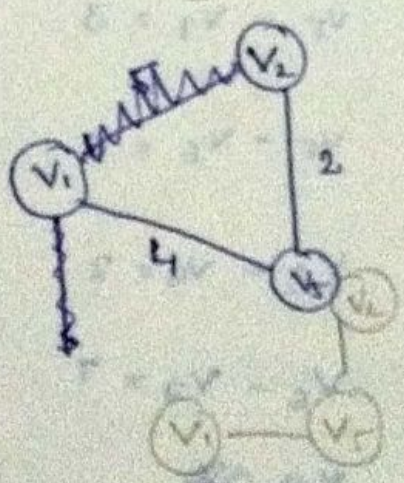
2: $V_1 - V_7 = 6$

$V_2 - V_5 = 2$

$V_2 - V_3 = 3$

$V_2 - V_6 = 6$



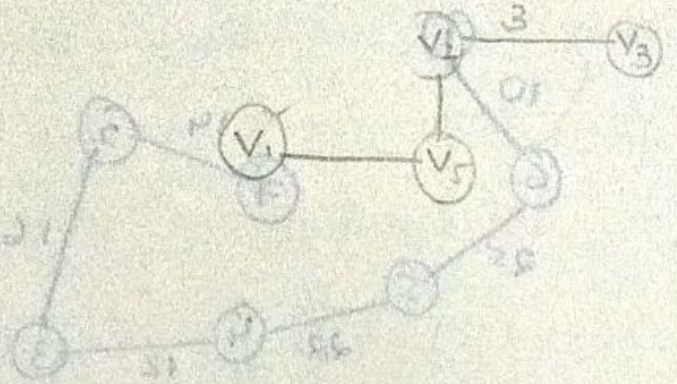
3: $V_1 - V_7 = 6$

$V_2 - V_3 = 3$

$V_2 - V_6 = 6$

$V_5 - V_7 = 3$

$V_5 - V_6 = 6$




4: $V_1 - V_7 = 6$

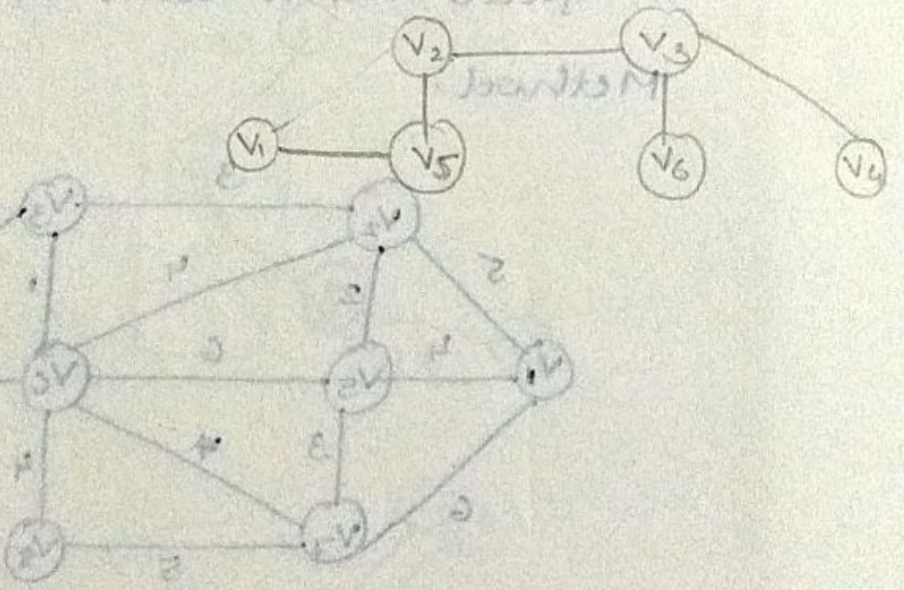
$V_2 - V_6 = 6$

$V_5 - V_7 = 3$

$V_5 - V_6 = 6$

$V_3 - V_6 = 1$

$V_3 - V_4 = 2$




5: $V_1 - V_7 = 6$

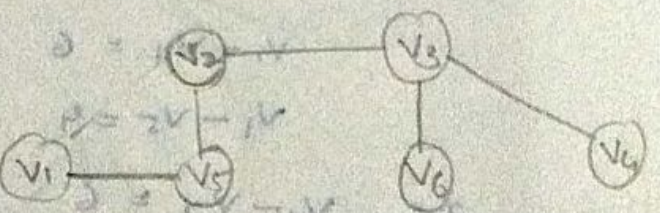
$V_2 - V_6 = 6$

$V_5 - V_7 = 3$

$V_5 - V_6 = 6$

$V_3 - V_4 = 2$

$V_6 - V_7 = 7$




$V_6 - V_4 = 8$

$V_6 - V_8 = 4$

6: $V_1 - V_7 = 6$

$V_2 - V_6 = 6$

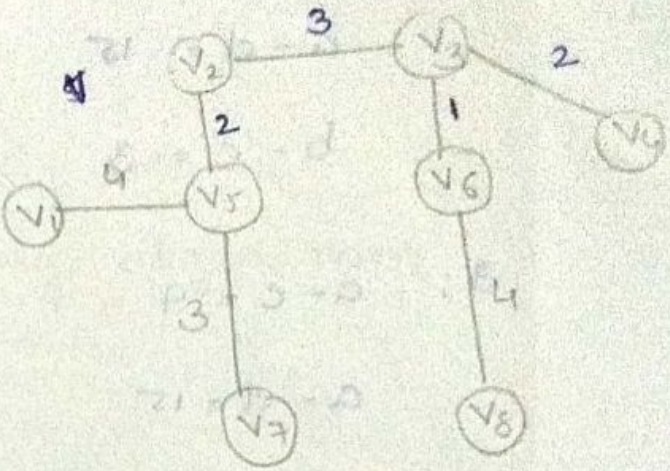
$V_5 - V_7 = 3$

$V_5 - V_6 = 6$

$V_6 - V_7 = 7$

$V_6 - V_4 = 8$

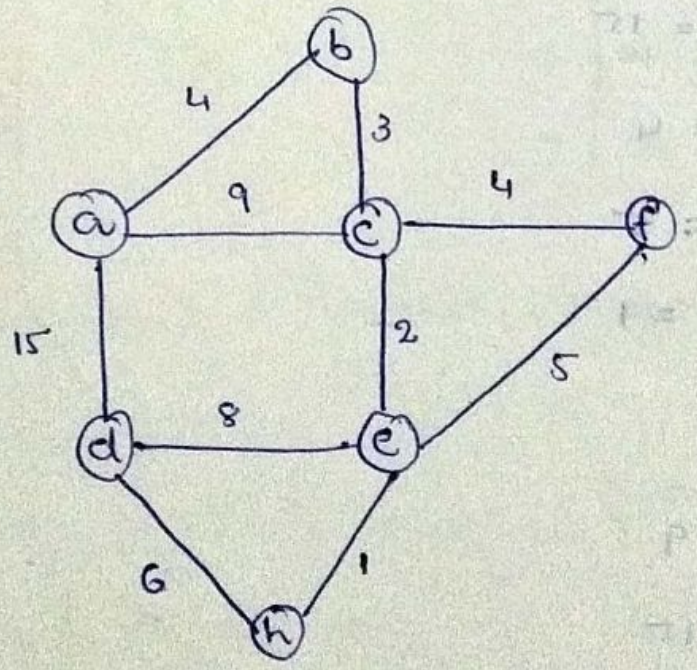
$V_6 - V_8 = 4$




Total cost = 4 + 3 + 2 + 3 + 1 + 2 + 4

= 19

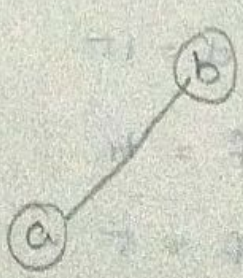
→




1: a — b = 4

a - c = 9

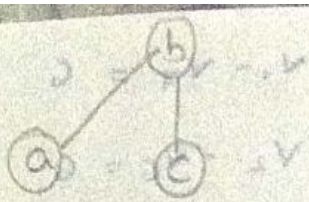
a - d = 15



2: $a - c = 9$

$a - d = 15$

$b - c = 3$

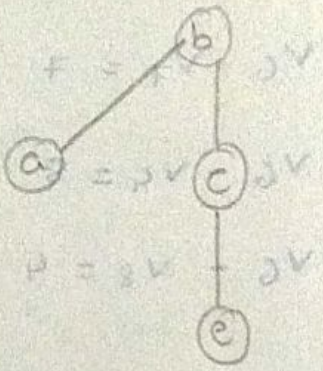



3: $a - c = 9$

$a - d = 15$

$c - e = 2$

$c - f = 4$



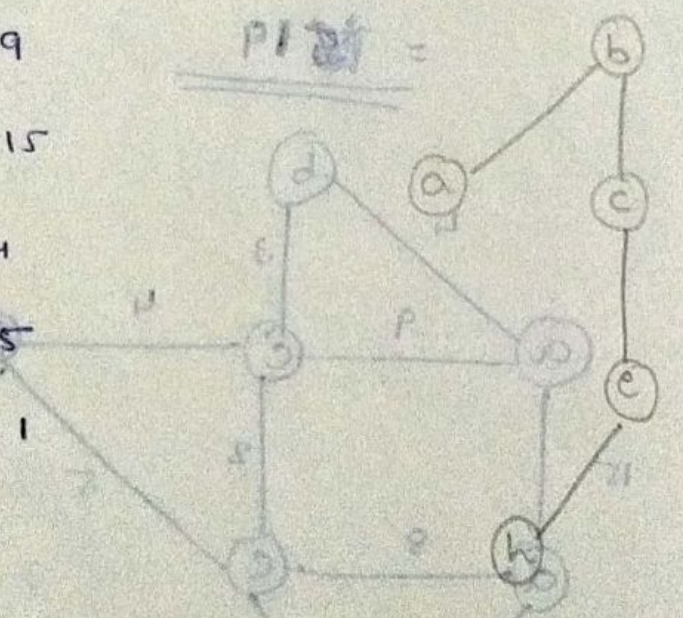

4: $a - c = 9$

$a - d = 15$

$c - f = 4$

$e - f = 5$

$e - h = 1$



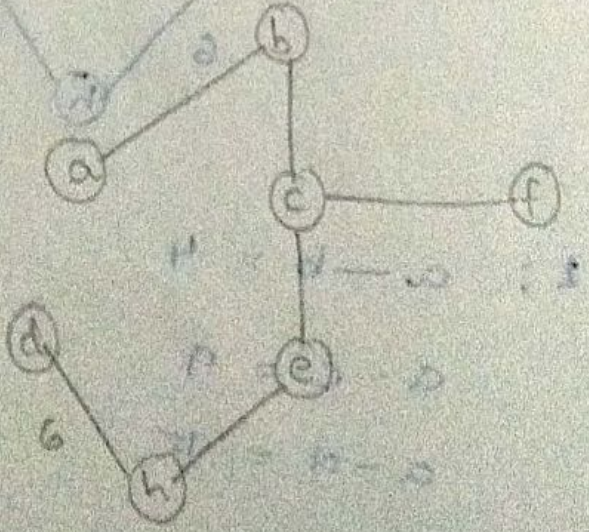

5: $a - c = 9$

$a - d = 15$

$c - f = 4$

$e - f = 5$

$h - d = 6$

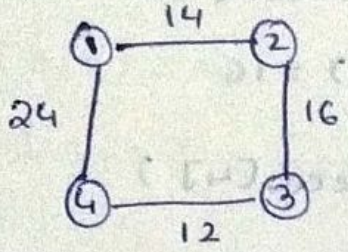


Total cost = 4 + 3 + 4 + 2 + .1 + .6

= 20

14/10/19

→ prim's working using algorithm




Sol:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 14 | ∞ | 24 |
| 2 | 14 | 0 | 16 | ∞ |
| 3 | ∞ | 16 | 0 | 12 |
| 4 | 24 | ∞ | 12 | 0 |

k = 1

l = 2

T

| 1 | 2 |
|---|---|

Min cost = 14

i = 1 to 4

cost (1,1) < cost(1,2) => 0 < 14

near[i] = 1

cost (2,1) ≠ cost(2,2)

14 > 0

near[2] = 2

cost (3,1) ≠ cost(3,2)

∞ > 16

near[3] = 2

cost (4,1) < cost (4,2)

24 < ∞

near[4] = 1

near[i]

| |
|---|
| 1 |
| 2 |
| 2 |
| 1 |

near[1] ≠ 0 & cost[1, near[1]]

cost(1,1) = 0

cost(2, near[2])

cost(2,2) = 0

cost(3, near[3])

cost(3,2) = 16

cost(4, near[4])

cost(4,1) = 24

j = 3

t(2,1) = 3

t(2,2), near[3] = 2

T

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

mincost = 14 + cost(3, near[3])

= 14 + cost(3,2)

= 14 + 16 = 30

near[3] = 0

| |
|---|
| 0 |
| 0 |
| 0 |
| 1 |

k = i to 4

near[1] ≠ 0 & cost(1, near[1]) => cost(k,i)

cost(1,1) = 0 < cost(1,3) = ∞

cost(2, near[2]) cost(2,3)

cost(2,2) = 0 < 16

cost(4, near[4]) cost(4,3)

cost (4,1) = 24 > 12

near [4] = 3

| 0 |
|---|
| 0 |
| 0 |
| 3 |

near [4] ≠ 0 & cost (4, near[4]) = 24

j = 4

T(3,1) = 4

T(3,2) = near [4] = 3

| 1 | 2 |
|---|---|
| 3 | 2 |
| 4 | 3 |

- mincost = 26 + cost (4, near[4])

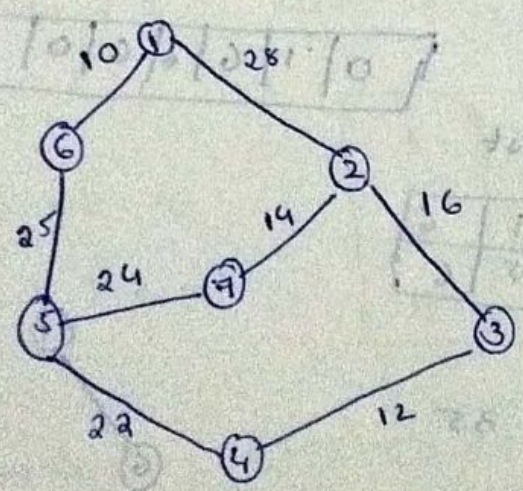
= 26 + cost (4,3)

= 30 + 12 = 42

near [4] = 0

| 0 |
|---|
| 0 |
| 0 |
| 0 |

Ans




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| 2 | 28 | 0 | 16 | ∞ | ∞ | ∞ | 14 |
| 3 | ∞ | 16 | 0 | 12 | ∞ | ∞ | ∞ |
| 4 | ∞ | ∞ | 12 | 0 | 22 | ∞ | ∞ |
| 5 | ∞ | ∞ | ∞ | 22 | 0 | 25 | 24 |
| 6 | 10 | ∞ | ∞ | ∞ | 25 | 0 | ∞ |
| 7 | ∞ | 14 | ∞ | ∞ | 24 | ∞ | 0 |

Sol k = 1

d = 2

| 1 | 6 |
|---|---|

mincost = 10

$i = 1$ to $7$

| near |
|---|
| 1 |
| 1 |
| 6 |
| 6 |
| 6 |
| 6 |
| 6 |

cost(1,1) cost(1,6)

0 < 10

cost(2,1) cost(2,6)

28 < ∞

cost(3,1) cost(3,6)

∞ = ∞

cost(4,1) cost(4,6)

∞ ∞

cost(5,1) cost(5,6)

∞ > 25

cost(6,1) cost(6,6)

10 > 0

$i = 2$

near[j] ≠ cost(j, near[j]) is minimum

| 0 | 1 | 6 | 6 | 6 | 0 | 6 |
|---|---|---|---|---|---|---|

$\therefore j = 5$

cost

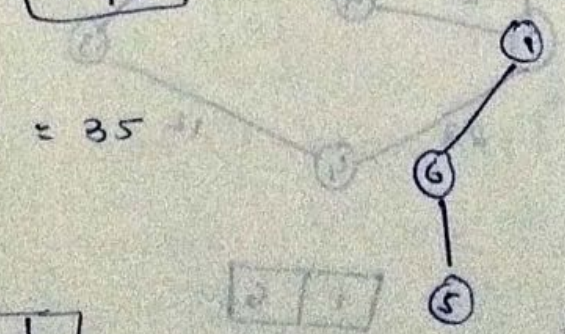
| 1 | 6 |
|---|---|
| 5 | 6 |

$t(2,1) = 5$

$t(2,2) = 6$

mincost = 10 + 25 = 35

near[5] = 0




| 0 | 1 | 6 | 6 | 0 | 0 | 6 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

$K = 2$

near [2] ≠ 0 & cost(2,1) cost(2,5)

28 < ∞

near [3] ≠ 0 cost(3,6) cost(3,5)

∞ = ∞

near [4] ≠ 0 cost(4,6) cost(4,5)

∞ > 22

near[4] = 5

| 0 | 1 | 6 | 5 | 0 | 0 | 6 |
|---|---|---|---|---|---|---|

near[7] ≠ 0 cost(7,6) cost(7,5)

∞ > 24

| 0 | 1 | 6 | 5 | 0 | 0 | 5 |
|---|---|---|---|---|---|---|

i = 3

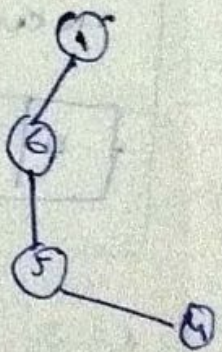
near[j] ≠ 0 cost(j, near(j)) is min.

∴ j = 4

t(3,1) = 4

t(3,2) = 5

| | |
|---|---|
| 1 | 6 |
| 5 | 6 |
| 4 | 5 |




min cost = 35 + 22 = 57

near [4] = 0

| 0 | 1 | 6 | 0 | 0 | 0 | 5 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

K = 1 to 7

near [2] ≠ 0 cost (2,1) < cost (2,4) => 28 < α

near [3] ≠ 0 cost (3,6) · cost (3,4) => α > 12

near [3] = 4

| 0 | 1 | 4 | 0 | 0 | 0 | 5 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

near [7] ≠ 0 cost (7,5) cost (7,4)

24 < α

i = 4

| 0 | 1 | 4 | 0 | 0 | 0 | 5 |
|---|---|---|---|---|---|---|

cost 28 12 24

near [j] ≠ 0 & cost (j, near [j]) is minimum.

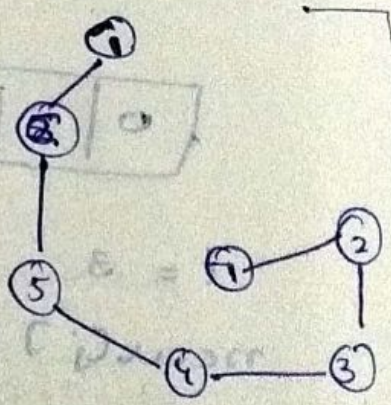
Ans:

mincost = 85 + 14

= 99

near [7] = 0

| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|

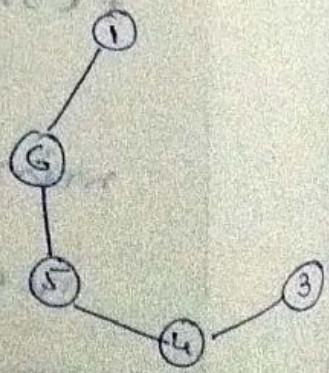



j = 3

t (4,1) = 3

t (4,2) = 4

| 1 | 6 |
|---|---|
| 5 | 6 |
| 4 | 5 |
| 3 | 4 |




mincost = 57 + 12 = 69

near [3] = 0

| 0 | 1 | 0 | 0 | 0 | 0 | 5 |
|---|---|---|---|---|---|---|

k = 1 to 7

near[2] ≠ 0  cost(2,1)  cost(2,3)

28 > 16

near[2] = 3

| 0 | 3 | 0 | 0 | 0 | 0 | 5 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

near[7] ≠ 0  cost(7,5)  cost(7,3)

24 < ∞

i = 5

| 0 | 3 | 0 | 0 | 0 | 0 | 5 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

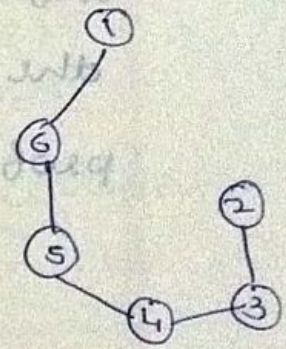
16  24

near[j] ≠ 0 cost(j, near[j]) is minimum.

j = 2 , t(5,1) = 2

t(5,2) = 3

| 1 | 6 |
|---|---|
| 5 | 6 |
| 4 | 5 |
| 3 | 4 |
| 2 | 3 |




min cost = 69 + 16

= 85

near[2] = 0

| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

k = 1 to 7

near[k] ≠ 0  cost(7,5)  cost(7,2)

24 > 14

near[7] = 2

| 0 | 0 | 0 | 0 | 0 | 0 | 14 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

i = 6

cost

near[j] ≠ 0 cost(j, near[j]) is min.

j = 7

t(6,1) = 7

t(6,2) = 2

| 1 | 6 |
|---|---|
| 5 | 6 |
| 4 | 5 |
| 3 | 4 |
| 2 | 3 |

15/10/19

# Job sequencing (or scheduling) with deadline using Greedy Method

It is a type of optimisation problem that uses greedy strategy. We have 'n' jobs, each with a deadline, $d_i \geq 0$ and profit, $p_i \geq 0$. Our objective is to earn maximum profit ~~within~~ deadline. We will earn profit only when a job is completed on or before the deadline. The optimal solution is the feasible solution with maximum profit.

$$\text{MAX} \sum_{i=1}^{n} p_i$$

Consider the following 5 jobs.

| Jobs | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ |
|---|---|---|---|---|---|
| profit | 20 | 15 | 10 | 5 | 1 |
| deadline | 2 | 2 | 1 | 3 | 3 |

Sol. Step 1: Sort the job according to their profit in descending order.

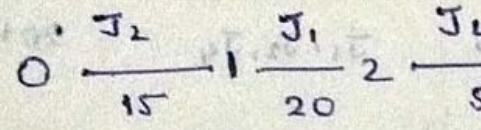
Assume, the machine should complete each job in one unit of time.

deadline: A job should wait until its completion time.

eg: If the job's deadline is 2, it has to wait for 2hrs.

Here, the maximum deadline is 3.

so, 0, 1, 2, 3




1: consider job 1,
p = 20,
d = 2

2: job 2,
p = 15,
d = 2,
∴ It completes within 1hr

3: job 3,
p = 10
d = 1, so, it cannot be completed within 1hr and also profit is less.
so, Job 3 is eliminated

4: job 4
p = 5
d = 3
so, the job can wait for 3hrs.

5: Job 5
p = 1
d = 3,
There is no time slot to complete the job

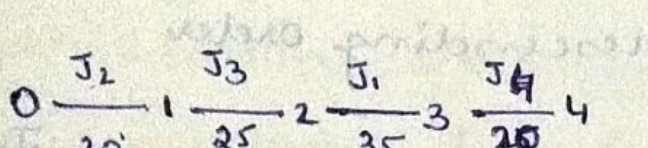
| Job considered | slot | solution | Profit |
|---|---|---|---|
| – | – | $\phi$ | 0 |
| $J_1$ | $\{0,1\}, \{1,2\}$ | $J_1$ | 20 |
| $J_2$ | $\{0,1\}, \{1,2\}$ | $J_1, J_2$ | 20+15 = 35 |
| $J_3$ | $\{0,1\}, \{1,2\}$ | $J_1, J_2$ | 20+15 = 35 |
| $J_4$ | $\{0,1\}, \{1,2\}, \{2,3\}$ | $J_1, J_2, J_4$ | 20+15+5 = 40 |
| $J_5$ | $\{0,1\}, \{1,2\}, \{2,3\}$ | $J_1, J_2, J_4$ | 20+15+5 = 40 |

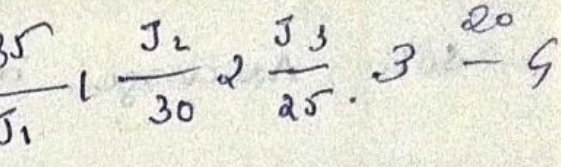
control abstraction.

```
GreedyJob(int d[], set J, int n)
//J is a set of jobs that can be completed
  by their deadlines.
{
  J = {1};
  for (int i = 2; i <= n; i++) {
  if (all jobs in J ∪ {i} can be completed
      by their deadlines)
    J = J ∪ {i};
  }
}
```

* 

| Jobs | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ |
|---|---|---|---|---|---|---|---|
| profit | 35 | 30 | 25 | 20 | 15 | 12 | 5 |
| deadline | 3 | 4 | 4 | 2 | 3 | 1 | 2 |




| Job considered | slot | solution | profit |
|---|---|---|---|
| - | - | $\phi$ | 0 |
| $J_1$ | $\{2,3\}$ | $J_1$ | 35 |
| $J_2$ | $\{0,1\},\{2,3\}$ | $J_1, J_2$ | 30+35 =65 |
| $J_3$ | $\{0,1\}\{1,2\}\{2,3\}$ | $J_1, J_2, J_3$ | 65+25 =90 |
| $J_4$ | $\{0,1\}\{1,2\}\{2,3\}$ $\{3,4\}$ | $J_1, J_2, J_3, J_4$ | 90+20 =110 |
| $J_5$ | $\{0,1\}\{1,2\}\{2,3\}$ $\{3,4\}$ | $J_1, J_2, J_3,$ $J_5$ | 110 |
| $J_6$ | $\{0,1\}\{1,2\}\{2,3\}$ $\{3,4\}$ | $J_1, J_2, J_3,$ $J_4$ | 110 |
| $J_7$ | $\{0,1\}\{1,2\}\{2,3\}$ $\{3,4\}$ | $J_1, J_2, J_3,$ $J_4$ | 110 |

| * J: | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ |
|---|---|---|---|---|---|
| D: | 2 | 1 | 3 | 2 | 1 |
| P: | 60 | 100 | 20 | 40 | 20 |

Sol: Arrange into descending order

| J | $J_2$ | $J_1$ | $J_4$ | $J_3$ | $J_5$ |
|---|---|---|---|---|---|
| D | 1 | 2 | 2 | 3 | 1 |
| P | 100 | 60 | 40 | 20 | 20 |

$$0 \frac{J_2}{100} 1 \frac{J_1}{60} 2 \frac{J_4}{40} 3$$

| Job considered | slot | solution | profit |
|---|---|---|---|
| - | - | $\phi$ | 0 |
| $J_2$ | $\{0,1\}$ | $J_2$ | 100 |
| $J_1$ | $\{0,1\}\{1,2\}$ | $J_2, J_1$ | 100+60 |
| $J_4$ | $\{0,1\}\{1,2\}\{2,3\}$ | $J_2, J_1, J_4$ | 100+60+40 =200 |
| $J_3$ | $\{0,1\},\{1,2\}\{2,3\}$ | $J_2, J_1, J_4$ | 200 |
| $J_5$ | $\{0,1\}\{1,2\}\{2,3\}$ | $J_2, J_1, J_4$ | 200 |