PAGE NO.
DATE :

23/02/24
FRI.

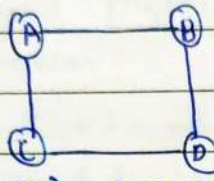
UNIT:NO: 3:(THREE):-

GIRAPH.

Terminology:-

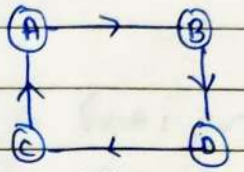
① Defination: A set of points or vertices / Edges.

Un-Directed Graph



V(G) = { A, B, C, D }
E(G) = { (A,B), (B,A), (B,D), (D,B), (C,D), (D,C), (A,C), (C,A) }

Directed Graph.



V(G) = { A, B, C, D }
EGE = { (A,B), (B,D), (D,C), (C,A) }

o A complete graph with n vertices has n(n-1)/2 edges.

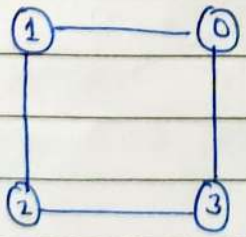
2 Weighted Graph:-

→ It has some values assigned to its edges.

Representation of Graphs:-

→ (1) Adjacency Matrix

Create and display a graph using adjacency matrix



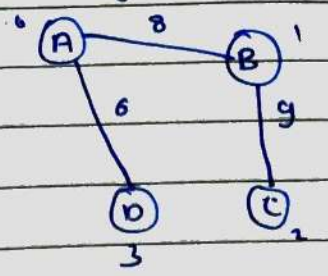
	0	1	2	3
0	0	1	0	1
1	1	0	1	0
2	0	1	0	1
3	1	0	1	0

PAGE NO.
DATE :

1. Intialize g[i][j] = 0
2. Enter the no. of nodes required.
3. for (i= 0; i<n ; i++)
   {
      for(j=0 ; j<n ; j++)
      {
         cin >> graph[i][j];
      }
   }
4. for (i= 0 ; i<n ; i++)
   {
      for (j=0; j<n ; j++)
      {
         cout << graph[i][j];
      }
   }

PAGE NO.
DATE :

→ (a.) : A Weighted Undirected graph.



(b.) Adjacency list.

0 A → [B|8] → [D|6] /
1 B → [A|8] → [C|9] /
2 C → [B|9] /
3 D → [A|6] /

→ Prim's Algorithm:-

STEPS:-

① ~~Choose~~ → Let the graph G has n vertices and assume that $V = \{1, 2, \ldots, n\}$ $G = (V, E)$.

1.) ② - Intialize T as the empty set.

$T = \{\phi\}$

③ $U = \{v_1\}$

④ $U - V = \{\quad\}$. Deduce the given difference

2.) while ($U != V$)

(begin)

① let (u, v) be the lowest cost edge

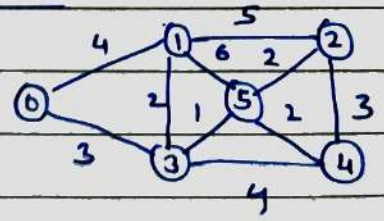
$u_1 \in U$ and $v_1 \in V - U$.

② $T = T \cup \{(u_1, v_1)\}$

$U = U \cup \{v\}$

(end)

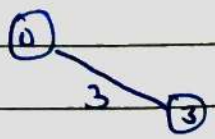
Example:-



① $U = \{0\}$ $V - U = \{1, 2, 3, 4, 5\}$

u	v-u	cost
0	1	4
0	2	∞
0	3	3
0	4	∞
0	5	∞

$T = \{(0, 3)\}$



PAGE NO.
DATE :

② $U = \{0, 3\}$ $V-U = \{1, 2, 4, 5\}$

U	V-U	cost.
3	1	2
0	2	∞
3	4	4
3	5	1

$T = \{(0,3), (3,5)\}$

③ $U = \{0, 3, 5\}$ $V-U = \{1, 2, 4\}$

→

U	V-U	cost
3	1	2
5	2	2
5	4	2

$T = \{(0,3), (3,5), (3,1)\}$

④ $U = \{0, 3, 5, 1\}$ $V-U = \{2, 4\}$

→

U	V-U	cost
5	2	2
5	4	2

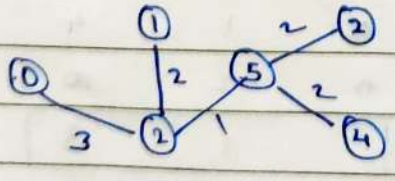
$T = \{(0,3), (3,5), (3,1), (5,2)\}$

⑤ $U = \{0, 3, 5, 1, 2\}$ $V-U = \{4\}$

U	V-U	cost
5	4	2

$T = \{(0,3), (3,5), (3,1), (5,2), (5,4)\}$

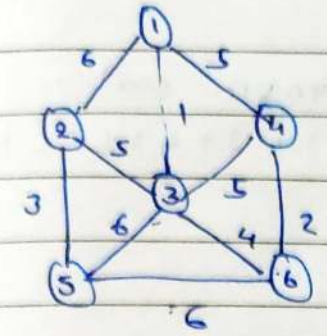
- Minimum Spanning Tree :-



Minimum cost = 3 + 2 + 1 + 2 + 2 = 10

PAGE NO.
DATE :

→ Example:- (2)



① U = {1} V-U = {2, 3, 4, 5, 6}

	U	V-U	cost
	1	2	6
→	1	3	1
	1	4	5
	1	5	∞
	1	6	∞

T = {(1,3)}

② U = {1,3} V-U = {2, 4, 5, 6}

	U	V-U	cost
	3	2	5
	1	4	5
	3	5	6
→	3	6	4

T = {(1,3), (3,6)}

③ U = {1, 3, 6} V-U = {2, 4, 5}

	U	V-U	cost
	3	2	5
	6	4	2
	3	5	6

T = {(1,3), (3,6), (6,4)}

④ U = {1, 3, 6, 4} V-U = {2, 5}

	U	V-U	cost
	3	2	5
	3	5	6

T = {(1,3), (3,6), (6,4), (3,2)}

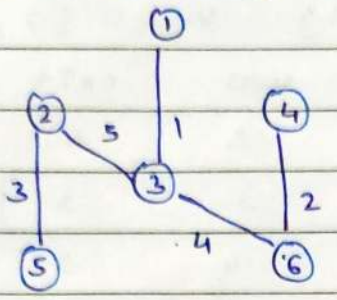
⑤ U = {1, 3, 6, 4, 2} V-U = {5}

	U	V-U	cost
	2	5	3

T = {(1,3), (3,6), (6,4), (3,2), (2,5)}

PAGE NO.
DATE :

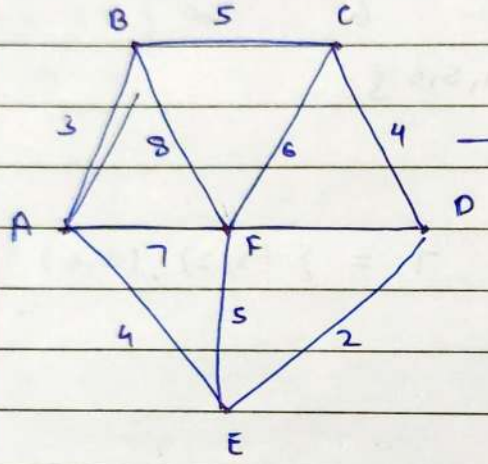
Minimum Spanning Tree:-



- Minimum cost:-

= 1+2+3+4+5 = 15

⟹



① U = {A} V-U = {B, C, D, E, F}

U	V-U	cost
→ A	B	3 ✓
A	C	∞
A	D	∞
A	E	4
A	F	7

T = {(A,B)} ✓

② U = {A,B} V-U = {C, D, E, F}

U	V-U	cost
B	C	5
A	D	∞
→ A	E	4 ✓
A	F	7

T = {(A,B), (B,E)}
A

③ U = {A,B,E} V-U = {B, C, D, F}

U	V-U	cost
B	C	5
→ E	D	2 ✓
E	F	5

T = {A, B, E, D}
(A,B), (B,E), (E,D)
A

④ U = {A,B,D,E} V-U = {C, F}

U	V-U	cost
→ D	C	4 ✓
E	F	5

T = {(A,B), (B,E), (E,D), (D,C)}
A

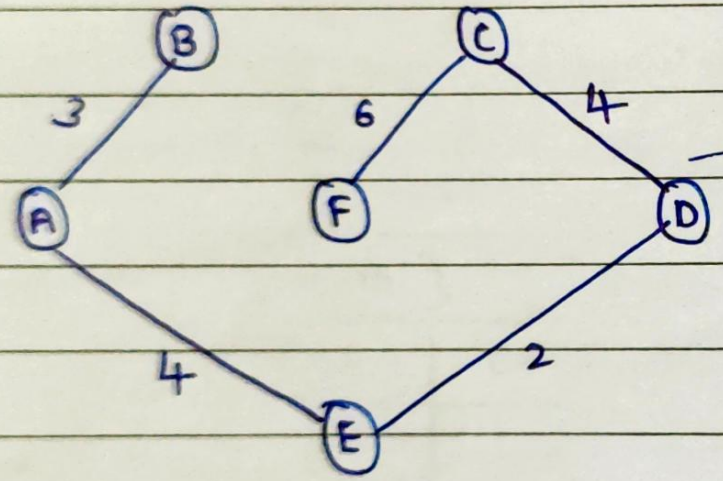
(3) $U = \{A, B, E, D, C\}$ $\quad V - U = \{F\}$

	U	V	cost.
→	E	F	5

$T = \{(A,B), (B,E), (E,D), (D,C), (E,F)\}$

A

- Minimum spanning Tree:-



- Minimum Cost:-

$= 3 + 4 + 2 + 4 + 6$

$= 19$

 DATE :

Kruskal's Algorithm:-

→ T = ∅

For each v ∈ V

  makeset(v)

Sort all E acc to cost E{}

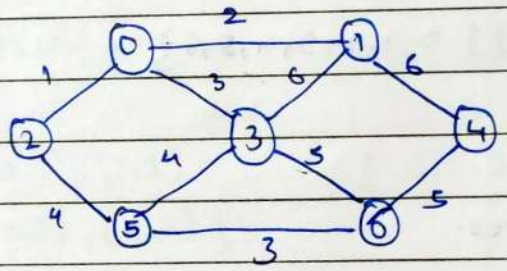
For each E (u,v) ∈ E

  if findset(u) ≠ findset(v)

    T = T∪(u,v)

    union(finset(u), findset(v))

} return T



E = {(0,2), (0,1), (0,3), (0,5), (2,5), (3,5), (3,6), (4,6), (1,3), (1,4)}

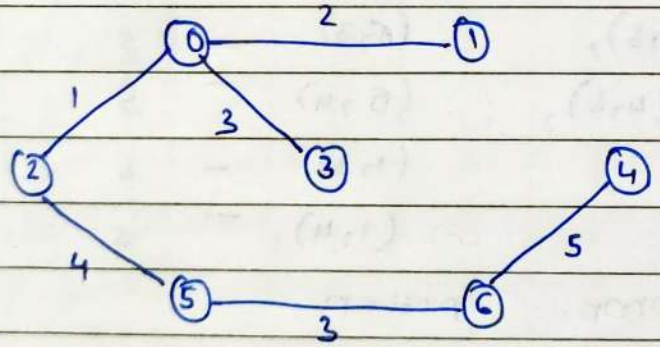
Edge		Weight
(0,2)	—	1
(0,1)	—	2
(0,3)	—	3
(5,6)	—	3
(2,5)	—	4
(5,3)	—	4
(6,3)	—	5
(6,4)	—	5
(1,3)	—	6
(1,4)	—	6

	T	connected comp.	Action
①	∅	{0}{1}{2}{3}{4}{5}{6}	—
②	{(0,2)}	{0,2}{1}{3}{4}{5}{6}	Added.
③	{(0,2), (0,1)}	{0,1,2}{3}{4}{5}{6}	Added.
④	{(0,2),(0,1),(0,3)}	{0,1,2,3}{4}{5}{6}	—

DATE :

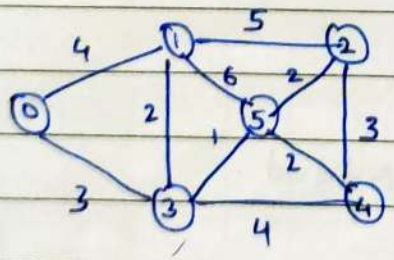
(5)	{(0,2),(0,1),(0,3), (5,6).}	{0,1,2,3} {5,6} {4}	Added.
(6)	{(0,2);(0,1),(0,3) (5,6),(2,5)}	{0,1,2,3,5,6} {4}	Added.
(7)	{(0,2),(0,1),(0,3) (5,6),(2,5)}	{0,1,2,3,4,5,6} {4}	(3,5) is rejected. (3,6) is rejected
(8)	{(0,2),(0,1),(0,3),(5,6) (2,5),(6,4)}	{0,1,2,3,4,5,6}	Added.

→ Minimum spanning Tree.



Minimum cost
= 2+1+4+3+3+5
= 18

(2)

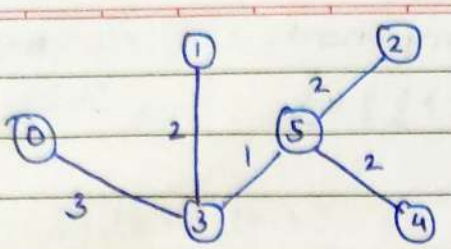


Edge	weight
(3,5)	1
(1,5)	2
(2,5)	2
(4,5)	2
(0,3)	3
(2,4)	3
(0,1)	4
(3,4)	4
(1,2)	5
(1,5)	6

PAGE NO.
DATE :

	T	Connected component	Action :-
①	ϕ	{0} {1} {2} {3} {4} {5} ~~{6}~~	—
②	{(3,5)}	{3,5} {0} {1} {2} {4} ~~{6}~~	Added.
③	{(3,5), {1,3}}	{1,3,5} {0} {2} {4} ~~{6}~~	Added.
④	{(3,5), (1,3), (2,5)}	{1,2,3,5} {0} {4} ~~{6}~~	Added
⑤	{(3,5), (1,3), (2,5), (4,5)}	{1,2,3,4,5} {0} ~~{6}~~	Added.
⑥	{(0,3), (1,3), (3,5), (2,5), (4,5)}	{0,1,2,3,4,5} ~~{6}~~	Added
⑦	"	{0,1,2,3,4,5} ~~{6}~~	(2,4) is rejected.
⑧	"	"	(0,1) is rejected
⑨	"	"	(3,4) is rejected
⑩	"	"	(1,2) is rejected
⑪	"	"	(1,5) is rejected

PAGE NO.
DATE :



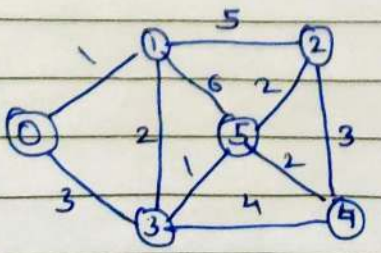
- Minimum cost :-
  = 3 + 2 + 2 + 2 + 1
  = 10

Kruskal's Algo.	Prim's Algo.
- Select the shortest Edge in a network	- Select any vertex.
- Select the shortest edge which does not create a cycle.	- Select the shortest Edge connected to that vertex
- Repeat step 2 until all vertices have been connected.	- Select the shortest edge connected to any vertex already connected.
	Repeat step 3 until all vertices have been connected.

- If the graph is created using matrix the time complexity of both is $O(n^2)$ for prim's and kruskals Algorithm. (n = no. of nodes)
- If the graph is created using ~~matrix~~ list the time complexity of prim's : $O(E \cdot \log n)$
  Time complexity of Kruskal's : $O(n^2) + O(E \log n) = O(n^2)$

Adjacency Matrix :-

→

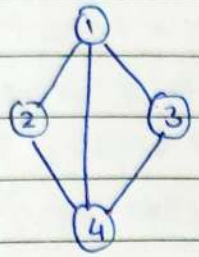


	0	1	2	3	4	5
0	∞	4	∞	3	∞	∞
1	4	∞	5	2	∞	6
2	∞	5	∞	∞	3	2
3	3	2	∞	∞	4	1
4	∞	∞	3	4	∞	2
5	∞	6	2	1	2	∞

PAGE NO.
DATE :

Adjacency Multilist:-

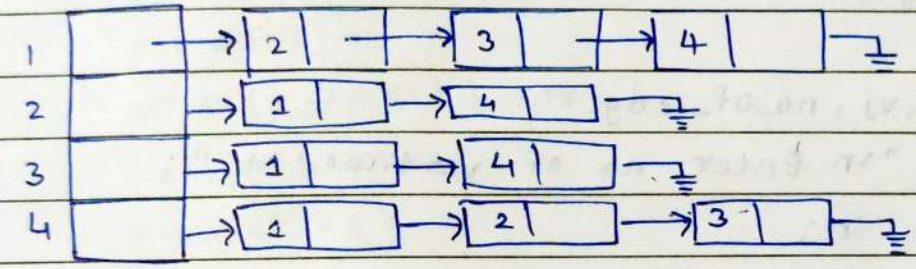
→ Multilist representation is a Edge based representation.

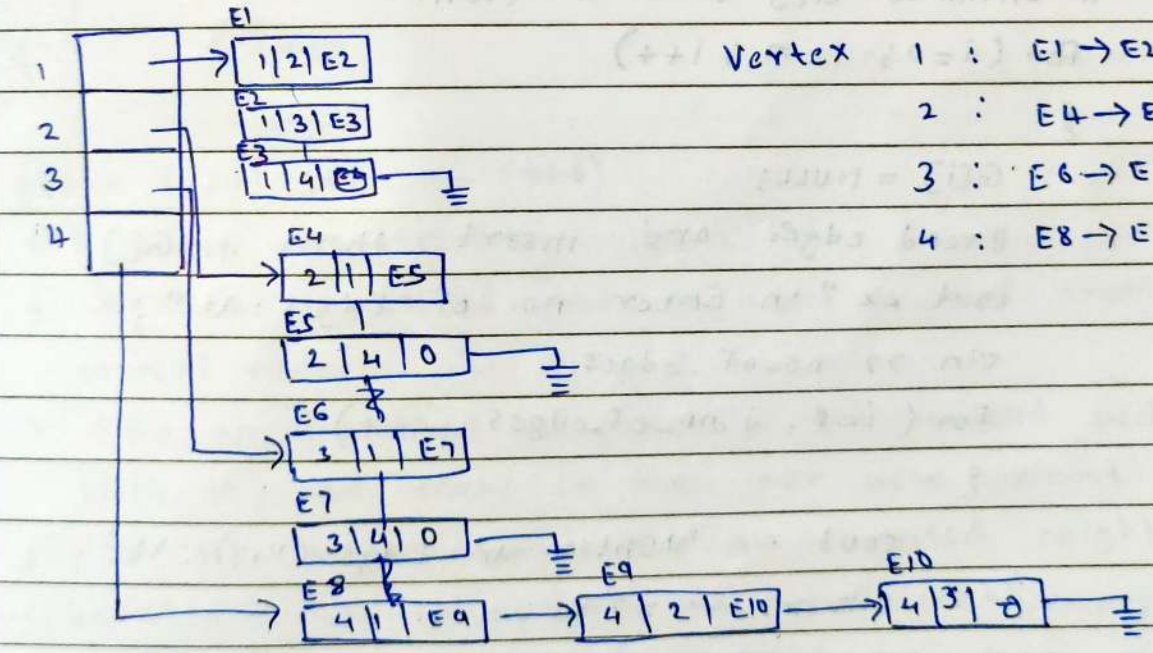


• Adjacency Matrix representation.

	1	2	3	4
1	0	1	1	1
2	1	0	0	1
3	1	0	0	1
4	1	1	1	0

• Adjacency Multilist:-





Vertex 1 : E1 → E2 → E3

2 : E4 → E5

3 : E6 → E7

4 : E8 → E9 → E10.

PAGE NO.
DATE :

Algorithm:-

```
class node
{
  int/char vertex;
  node * next;
};
node *G[10];

void read_graph()
{
  int i, vi, vj, no_of_edges;
  cout << "\n Enter no. of vertices : \t ";
  cin >> n;
  // Intialise G[] with a NULL.
  for (i=0; i<n ; i++)
  {
     G[i] = NULL;
     // read edges and insert them in G[]
     cout << "\n Enter no. of Edges : \t ";
     cin >> no_of_edges;
     for ( i=0; i<no_of_edges ; i++)
     {
        cout << "\nEnter an Edge (u,v) : \t";
        cin >> vi >> vj;
        insert (vi, vj);
     }
  }
}
void insert (int vi, int vj)
{
  node *p, *q;
  // Aquire memory for the new node.
```

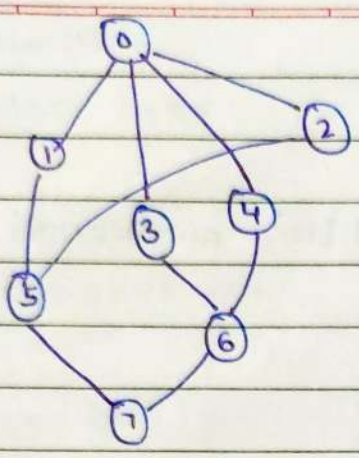
```
q = new node;
q -> vertex = vj;
q -> next = NULL;
// Insert the node in the Linked list number vi
if (G[vi] == = NULL)
        G[vi] = q;
else
{
  // go to end of linked list
   p = G[vi]
   while (p -> next ! = NULL)
        p = p -> next
   p -> next = q;
}
}
```

DEPTH FIRST SEARCH:- (DFS)

DFS follows the following rules :

1. Select an unvisited node x, visit it, and treat as the current node.
2. Find an unvisited neighbor of the current node, visit it, and make it the ~~cur~~ new current node
3. If the current node has no unvisited neighbours, backtract to its parent, and make that current node
4. Repeat steps 3 and 4 until no more nodes can be visited.
5. If there are still unvisited nodes, repeat from step.

PAGE NO.
DATE :



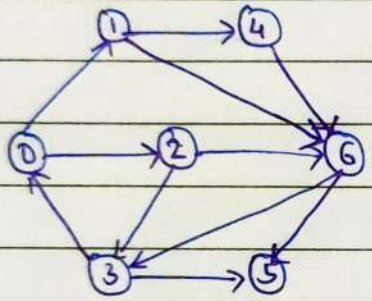
- Find the DFS path

start : 7

ANS: 7 5 1 0 2 3 6 4

backtracks

7 → 5 → 1 → 0 → 2 → 0
4 ← 6 ← 3 ←



- Find the DFS path

start : 0

→ 0 1 4 6 3 5 2

→ DFS (Pseudocode):-

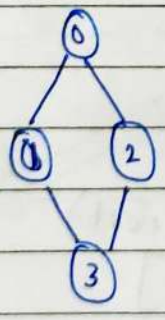
```
1. procedure DFS-iterative(G,v):
2. Let S be a attack
3. S.push(v)
4. while S is not empty.
5.     v = S.pop()
6.     if v is not visited :
7.         Mark v as visited
8.         AddAll adjacent vertices w to v in
               G.adjacentEdges(v)
9.     if w is not visited then push w on to
           the stack S.push(w)
```

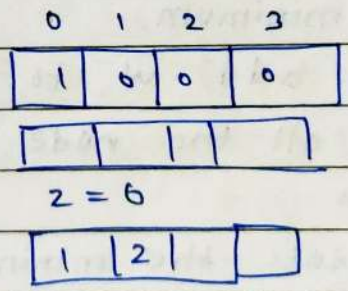
NOTE:- For BFS there is a Unique path.
For DFS there are always two paths

Breadth First Search :-

→ BFS (input : graph G)

```
{
    Queue Q;
    integer x, y, z;
    while (G has an unvisited node i)
    {
     visit (i);
       Enqueue (i, Q);
     while (Q is not empty)
     {
       z := Dequeue (Q);
       for all (unvisited neighbour y of z)
                {
                visit (y);
                   Enqueue (y, Q);
            }
     }
}
```

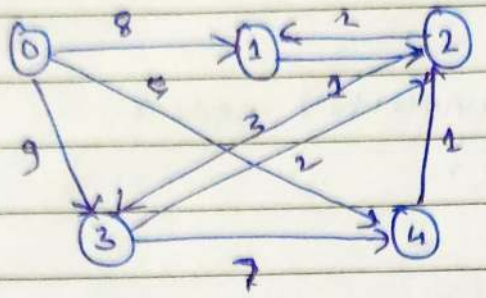




Shortest Path:-

→ shortest path between two vertices in a weighted graph has smallest edge-weight sum.

(a)



	0	1	2	3	4
0	∞	8	∞	9	4
1	∞	∞	1	∞	∞
2	∞	2	∞	3	∞
3	∞	∞	2	∞	7
4	∞	∞	1	∞	∞

→ Dijkstra's Algorithm for ~~shorte~~ shortest path is also called as single path algorithm'

→ Algorithm:-

(1.) • $N = \{s\}$ ← source node

$D_s = 0$ ....(Distance is zero itself.)

$D_j = c_{sj}$ Per all $j \neq s$, distance of directly connected neighbour. and origin all other node.

(2.) Choose a vertex $(w \in V-N)$ such that

$D(w)$ is a minimum.

$N = N \cup \{w\}$ add w to N

If N contains all the node stop

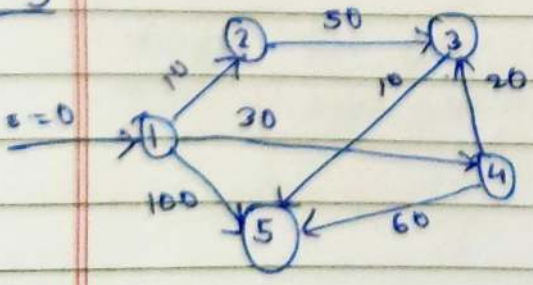
(3.) For Each vertex

$v \in V-N$ (Update the minimum cost)

$D[v] = \min [D[v], D(w) + ([w,v])]$

go to step 2

Ex.)



① Intially,

$N = \{1\}$

$D[102] = 10$ $D[4] = 30$

$D[3] = \infty$ $D[5] = 100$

② $N = \{1,2\}$

$D[3] = \min[\infty, 10+50] = \min[\infty, 60] = \underline{60}$

$D[4] = \min[30, 10+\infty] = \min[30, \infty] = \underline{30}$

$D[5] = \min[100, 10+\infty] = \min[100, \infty] = \underline{100}$

The minimim cost : $D[4] = 30$.

③ $N = \{1,2,4\}$ $\quad N-V = \{3,5\}$

$\therefore D[3] = \min\{60, 30+20\} = \min\{60, 50\} = \underline{50}$

$D[5] = \min\{100, 30+60\} = \min\{100, 90\} = \underline{90}$.

$\therefore$ The minimum cost :- $D[3] = 50$

④ $N = \{1,2,3,4\}$ $\quad N-V = \{5\}$

$\therefore D[5] = \{90, 50 + 10\} = \{90, 60\} = \underline{60}$

⑤ $W = 5$ $\quad N = \{1,2,4,3,5\}$

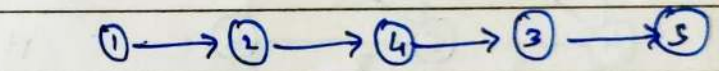
$D[2] = 10$

$D[3] = 50$

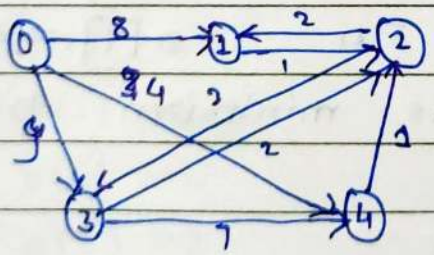
$D[4] = 30$

$D[5] = 60$

• Shortest path :-



Ex]



① Intially,

$N = \{0\}$

$D[1] = 8 \quad D[2] = \infty$

$D[3] = 9 \quad D[4] = 4$

$\therefore D[4] = 4$ is minimum

② $N = \{0, 4\}$ $\quad W = \{4\}$

$\therefore D[1] = \min\{8, 4+\infty\} = \min\{8, \infty\} = 8$

$D[2] = \min\{\infty, 4+1\} = \min\{\infty, 5\} = 5$

$D[3] = \min\{9, 4+\infty\} = \min\{9, \infty\} = 9$

$D[2]$ is minimum $\quad D[2] = 5$

DATE :

③ $N = \{0, 4, 2\}$ $w = \{2\}$

$D[1] = \min\{8, 5+2\} = \min\{8, 7\} = 7$

$D[3] = \min\{9, 5+3\} = \min\{9, 8\} = 8$

$\therefore D[1]$ is minimum.

④ $N = \{0, 1, 2, 4\}$ $w = \{1\}$

$D[3] = \min\{8, 7+\infty\} = \min\{8, \infty\} = 8$

⑤ $N = \{0, 4, 2, 1, 3\}$ ~~$w = 3$~~

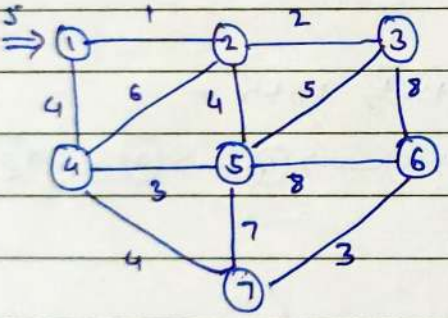
$D[4] = 4$

$D[2] = 5$

$D[1] = 7$

$D[3] = 8$

Q.]



Find shortest path.

① Intially;

$N = \{1\}$ ~~$V-N$~~ $= \{2, 3, 4, 5, 6, 7\}$

$D[2] = 1$ $D[5] = \infty$

$D[3] = \infty$ $D[6] = \infty$

$D[4] = 4$ $D[7] = \infty$

$\therefore D[2]$ is minimum $D[2] = 1$

② $N = \{1, 2\}$ $w = \{2\}$

$D[3] = \{\infty, 1+2\} = 3$

$D[4] = \{4, 1+6\} = 4$

$D[5] = \{\infty, 1+4\} = 5$

$D[6] = \{\infty, 1+\infty\} = \infty$

$D[7] = \{\infty, 1+\infty\} = \infty$

$D[3]$ is minimum, $D[3] = 3$

③ $N = \{1, 2, 3\}$ $w = \{3\}$ $V-N = \{4, 5, 6, 7\}$

$\therefore D[4] = \min\{4, 3+\infty\} = 4$

$\therefore D[5] = \min\{5, 3+5\} = 5$

PAGE NO.
DATE :

$D(6) = \min\{\infty, 3+8\} = 11$
$D(7) = \min\{\infty, 3+\infty\} = \infty$
$\therefore D(4)$ is minimum, $D(4) = 4$.

④ $N = \{1,2,3,4\}$ $w = \{4\}$
$D(5) = \min\{5, 4+3\} = 5$
$D(6) = \min\{11, 4+\infty\} = 11$
$D(7) = \min\{\infty, 4+4\} = 8$
$D(5)$ is minimum, $D(5) = 5$

⑤ $N = \{1,2,3,4,5\}$ $w = \{5\}$
$D(6) = \min\{11, 5+8\} = 11$
$D(7) = \min\{8, 5+7\} = 8$
$\therefore D(7)$ is minimum, $D(7) = 8$

⑥ $N = \{1,2,3,4,5,7\}$ $w = \{7\}$
$D(6) = \min\{11, 8+3\} = 11$

⑦ $N = \{1,2,3,4,5,7,6\}$
$D(2) = 1$
$D(3) = 3$
$D(4) = 4$
$D(5) = 5$
$D(7) = 8$
$D(6) = 11$

• The shortest path is given as:-