Unit

:- Graph :-

Graph is a non linear data structure it is a set of vertex (V) and edges (E) so graph is defined as $G = (V, E)$.

Graph may be cyclic or non cyclic.

Diff^n b/w tree and graph :-

Tree	Graph
① Tree is a Non-linear data structure.	① Graph is also a Non-linear data structure.
② Tree is represent the Hierarchical model	② Graph represent the Network model.
③ Tree must be non cyclic	③ Graph may be cyclic or non-cyclic.
④ Tree always follow some rule and regulations	④ Graph may not be follow any rule and regulation.
⑤ In Tree If n no. of nodes then (n-1) no. of edges are required.	⑤ In graph edge is not independ on no of Vertex.
⑥ In Tree first element is treated as Root.	⑥ There is no Root node in Graph.

Types of Graph :-

① Directed Graph :-

* directed order pair of Vertex is known as directed Graph. where

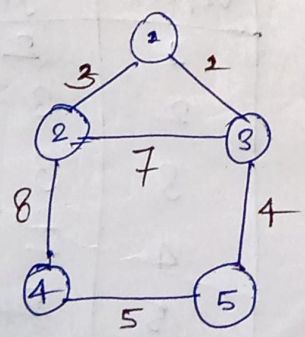
set of Vertex(V) (A,B)
set of Edges E = (A,B) (B,A)

② Undirected Graph :-

Unordered pair of Vertex is known as Undirect Graph.

set of Vertex (V) = (A,B)
set of Edges (E) = (A,B), (B,A)

③ weighted Graph :-



A weighted Graph is a graph in which edge are assigned with some value

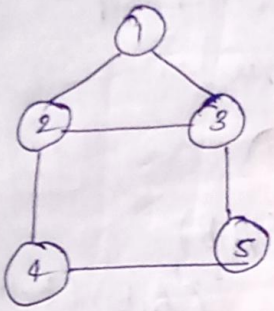
Path from 1 to 5

1-2-4-5 = 3+8+5 = 16km
1-2-3-5 = 3+7+4 = 14km
1-3-5 = 1+4 = 5km

shortest path = ~~1-3-5~~ 1-3-5

④ Cyclic graph:- A cycle is a simple path that start and end at the same vertex.

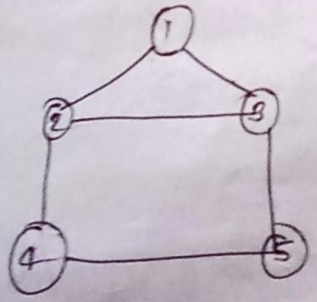
⑤



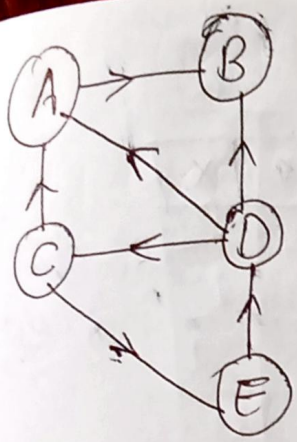
⑤ Degree of Vertex:- The total no. of edges link to a vertex is called it's degree

ⓐ In degree:- In degree of a vertex is the total no of edges comming to that node.

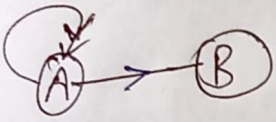
ⓑ out degree:- Out degree of a vertex is the total no of edges going out from that node.



Vertex	Indegree	outdegree
1	2	2
2	3	3
3	3	3
4	2	2
5	2	2



Vertex	Indegree	outdegree
A	2	1
B	2	0
C	1	2
D	1	3
E	1	1

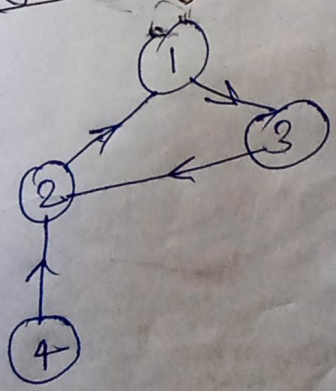


	Indegree	outdegree
A	1	2

Representation Graph in memory:- There are three way to represent Graph in memory.

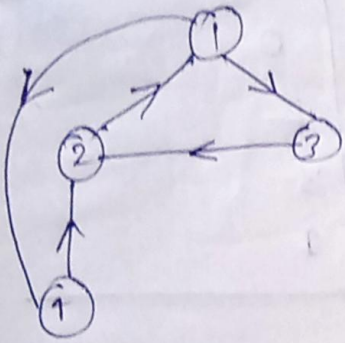
1. Adjancency Matrices:-
2. (i) Adjancency list
   (ii) Adjancency multilist

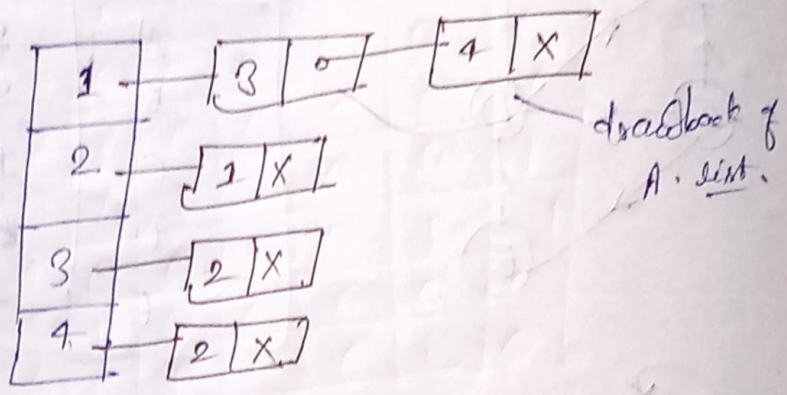
1. Adjancency Matrices:-



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

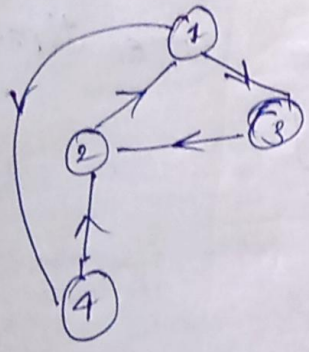
(ii) Adjacency list :-

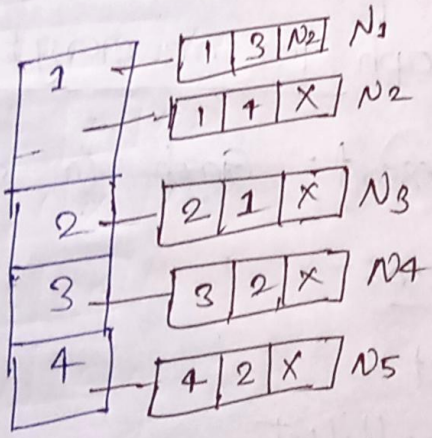




(iii) Adjacency

(iv) Adjacency Multilist :-





Q. Find out the following

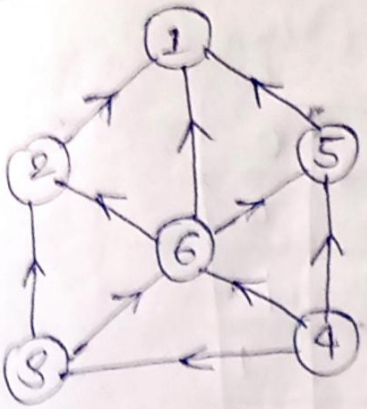
(i) Indegree and outdegree

(ii) Adjacency matrices

(iii) Adjacency list

(iv) Adjacency multilist

of a given graph

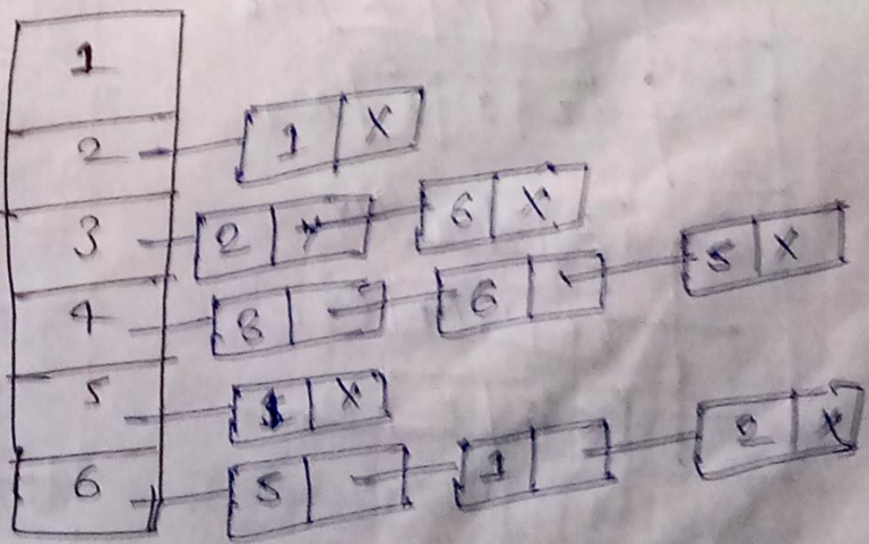


Vertex	Indegree	outdegree
①	3	0
②	2	1
⑤	1	2
④	0	3
⑤	2	1
⑥	2	3

(1) Adjacency matrices :-

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

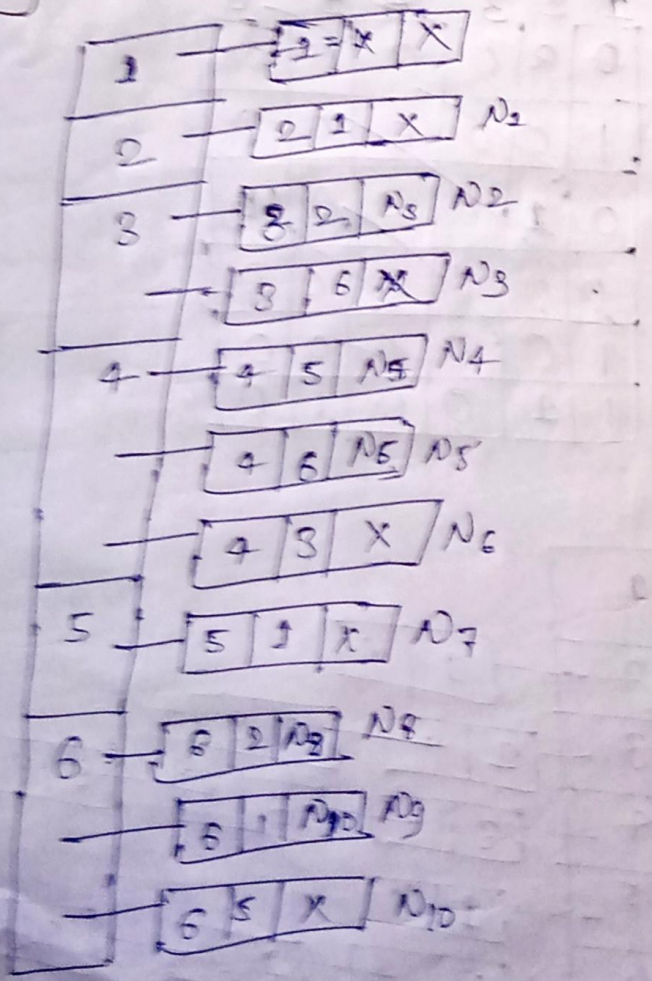
(N) Adjacency list :-



(3) Adjacency Matrices:-

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

(4) Adjacency Multilist



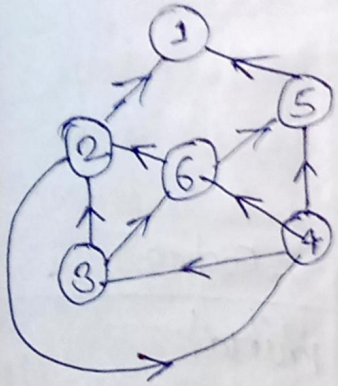
Traversal of Graph:-

There are two method to traverse a Graph.

(I) Depth first search (DFS) - it work on stack

(II) Breadth first search (BFS) - it work on queue.

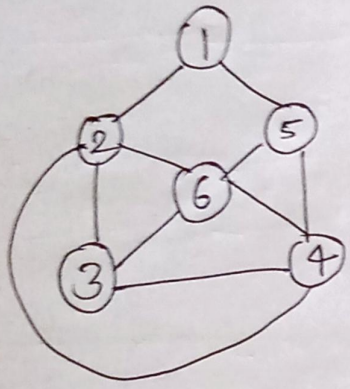
Prob 1 Traverse a given graph with the help of depth first search Algorithm.



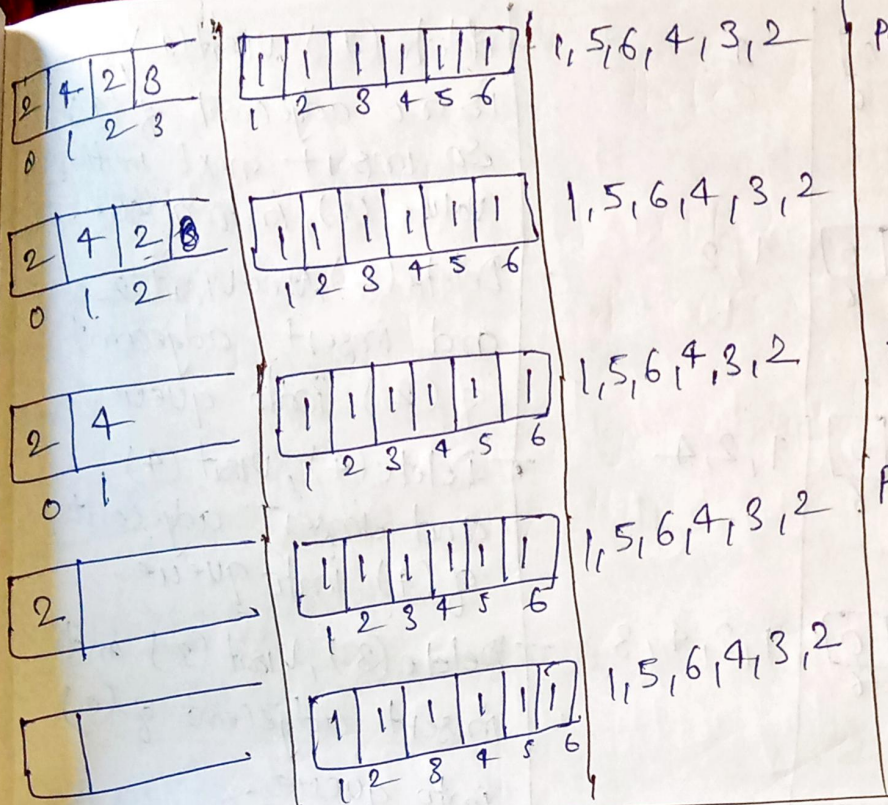
Stack	vertex	Vertex Visited	action
	0 0 0 0 0 0	—	initial
1	0 0 0 0 0 0	—	PUSH 1 into stack.
	0 0 0 0 0 0	1	POP(1) visit (1), there is already of (1) so push second initial Vertex into stack
2 4	1 0 0 0 0 0	1, 2	POP(2) visit (2), PUSH the adjecent of (2) into stack.
4	1 1 0 0 0 0	1, 2, 4	POP(4) visit (4) push the adjecent of (4) into stack)
3 5 6	1 1 0 1 0 0	1, 2, 4, 6	POP(6), visit (6), push the adjecent of (6) into stack
3 5 5	1 1 0 1 0 1		

3 5 (0 1)	1 1 0 1 1 1 (1 2 3 4 5 6)	1, 2, 4, 6, 5	Pop(5), visit(5), their adjecent already push
3 (0)	1 1 0 1 1 1 (1 2 3 4 5 6)	1, 2, 4, 6, 5	pop(5)
(0)	1 1 1 1 1 1 (1 2 3 4 5 6)	1, 2, 4, 6, 5, 3	pop(3) and visit(3)

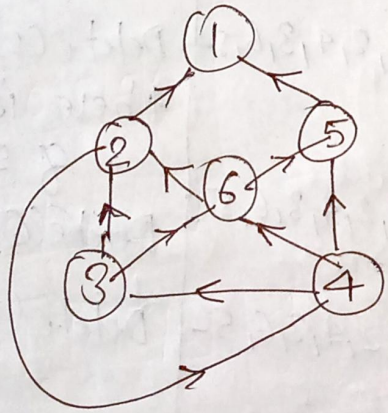
(2)



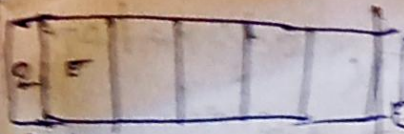
stack	vertex	vertex visited	action
(0 1)	0 0 0 0 0 0 (1 2 3 4)	—	initial
1 (0 1)	0 0 0 0 0 0 (1 2 3 4 5 6)	—	PUSH(1) into stack
2 5 (0 1)	1 0 0 0 0 0 (1 2 3 4 5 6)	1	pop(1), visit(1) and push adjecent of (1) into stack.
2 4 6 (0 1 2)	1 0 0 0 1 0 (1 2 3 4 5 6)	1, 5	pop(5), visit(5) and push adjecent of (5) into stack
2 4 2 3 4	1 0 0 0 1 1 (1 2 3 4 5 6)	1, 5, 6	pop(6), visit(6) and push adjecent of (6) into stack.
2 4 2 3 3	1 0 0 1 1 1 (1 2 3 4 5 6)	1, 5, 6, 4	pop(4), visit(4) and push adjecent of 4 into stack
2 4 2 3 2	1 0 1 1 1 1 (1 2 3 4 5 6)	1, 5, 6, 4, 3	pop(3), visit(3) and push adjecent of 3 into stack



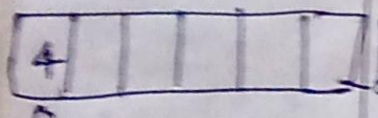
Q. Traverse a given graph with the help of Breath first search algorithm.

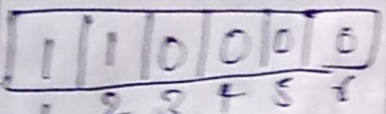


~~Stack~~ Queue	Vertex	Vertex visited	Action
0 1 2 3	0 0 0 0 0 0 / 1 2 3 4 5 6	—	initial
1 / 0 1 2 3 4 5	0 0 0 0 0 0 / 1 2 3 4 5 6	—	insert(1)

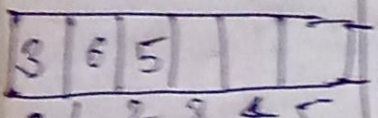


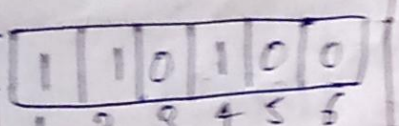
1 — Delete (1), visit (1), there is no adjecent of (1) & insert next initial value (2) into queue.



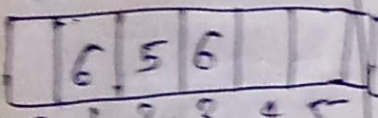


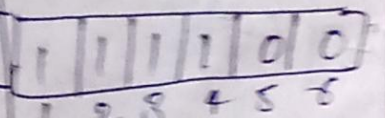
1,2 — Delete (2), visit 2 and insert adjecent of (2) into queue.



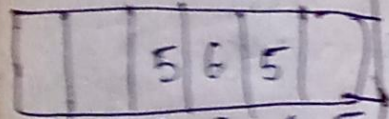


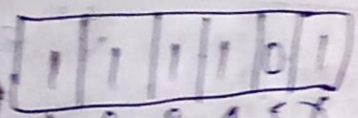
1,2,4 — Delete (4), visit (4), and insert adjecent of (4) into queue



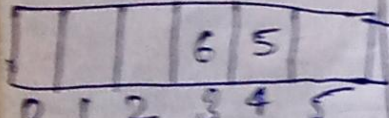


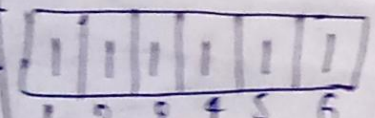
1,2,4,3 — Delete (3), visit (3) and insert adjecent of (3) into queue.



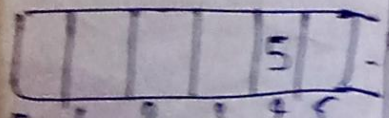


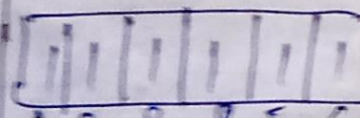
1,2,4,3,6 — Delete (6), visit (6) and insert adjecent of (6) into queue.



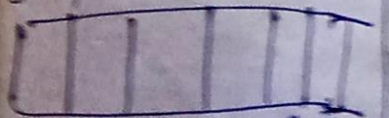


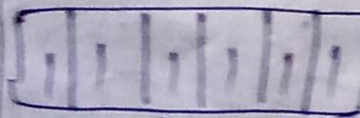
1,2,4,3,6,5 — Delete (5), visit (5) there is no adjecent of 5





1,2,4,3,6,5 — Delete (6)





1,2,4,3,6,5 — Delete (5)

Q Write an algorithm for depth first search (DFS).

Ans:-

step-1:- Initialize all vertex to the Ready state.

step-2:- PUSH the starting vertex onto stack and change its status to the waiting state.

step-3:- Repeat step 4 and 5 until stack is empty.

step-4:- Pop the Top vertex 'n' and process n and change its status to the processed state.

step-5:- PUSH onto stack all the neighbours of vertex 'n'. that are still in ready state and change their status to the waiting state.

step-6:- Exit

Q Write An algorithm for breath first search (BFS).

Ans

step-1:- Initialize all vertex to the Ready state.

step-2:- Insert the starting vertex into queue and change its status to the waiting state.

step-3:- Repeat step 4 and 5 until queue is empty.

step-4:- Delete the front vertex 'n' and process n and change its status to the processed state.

step-5:- Insert into queue all the neighbours of vertex 'n'. that are still in ready state and Change their status to the waiting state.

Step 6:- Exit.

Applications of Graph:-

① Graph is non linear data structure and is used to present various operations and algorithms.

② Graph is used to find shortest path

③ Graph is used for topological sorting.

Q. Illustrate the importance of various traversing technique in graph along with its application

AKTU 2016-17

Ans:- There are two type of traversal-

(I) Depth first search (DFS)

(II) Breath first search (BFS)

Importance of DFS:-

(I) Testing whether graph is Connected.

(II) Computing the Connected Component of G.

(III) Computing a path between two vertices of G or reporting that no such path exist.

(IV) Computing a cycle in G or reporting that no such cycle exist.

Application of DFS:-

Algorithms that use depth first search as a building block include.

① Finding Connected Components.

② Topological sorting.
③ Finding bridge of a graph.
④ Finding strongly Connected Components.

Importance of BFS:-

① It is one of the single source shortest path algorithm, so it is used to compute the shortest path.

⑪ It is also used to solve puzzles such as Rubik's cube.

⑪ BFS is not only the quickest way to solving the Rubik's cube, but also most optimal way to solve it.

Application of BFS:-

BFS can be used to solve many Problems in graph theory, for example-

① Copying garbage Collection.

② Finding the shortest path b/w two nodes u and v with path length measured by no of edges.

③

---

for spanning tree:- A Connected subgraph 'S' of Graph 'G' is said to be spanning tree

① 'S' should contain all verteces of 'G'

② 'S' should contain (|V|-1) edges

V- no of verteces

Spanning tree and minimum cost of spanning tree:-

With the application of weighted graph it is necessary to find the spanning tree for which total weight of the edge in the tree is small as possible such a spanning tree is called the minimum cost of spanning tree.

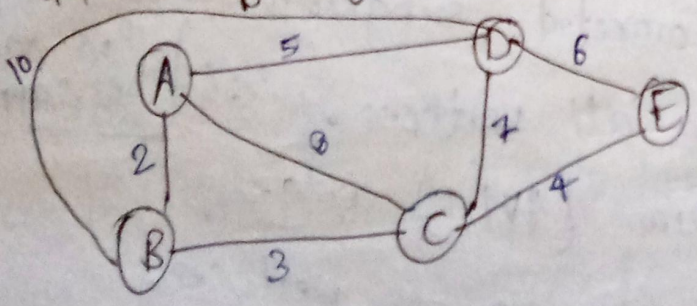
There are two popular algorithm to construct a spanning tree -

① Kruskal's Algorithm – it work on edge
② Prim's Algorithm – it work on vertex.

Note:- ① In the minimum cost of spanning tree the graph is always undirected,

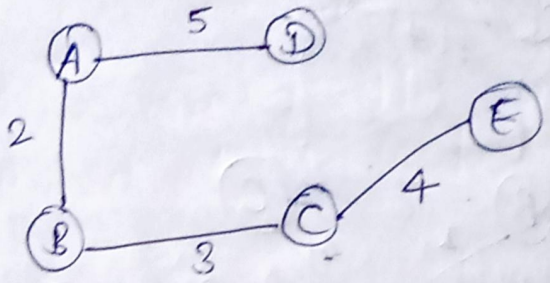
② All branches or edges along with non negative weight, there is no cycle generated after constructing a spanning tree.

Q. Construct a spanning tree and find out the minimum cost of spanning tree of a given graph with the help of Kruskal's Algorithm.



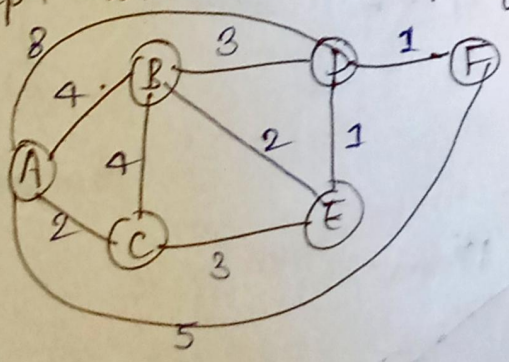
Edges are sorted in assending order of their weights.

AB = 2
BC = 3
CE = 4
AD = 5
DE = 6
CD = 7
AC = 8
BD = 10

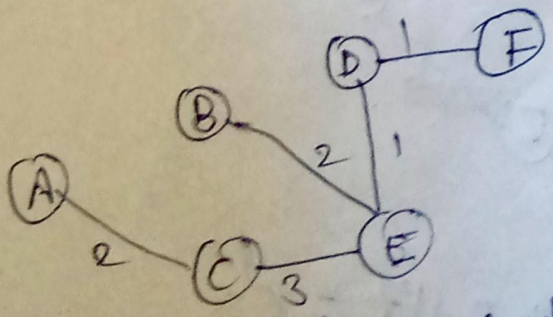


Total minimum cost of spanning tree
2+3+4+5 = 14

Find out minimum Cost of spanning tree of a given graph with the help of kruskal's algorithm.

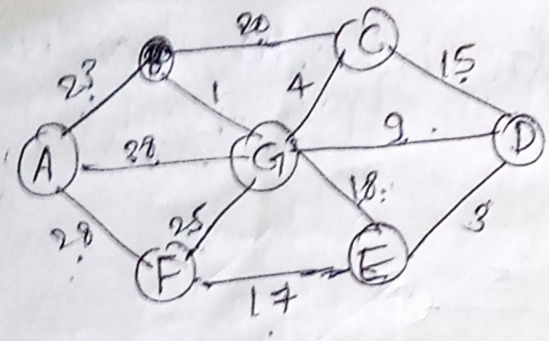


✓ DE = 1
✓ DF = 1
✓ BE = 2
✓ AC = 2
✗ BD = 3
✗ BC = 4
✗ AB = 4
✗ BC = 4
✗ AF = 5
✗ AD = 7
✓ CE = 3

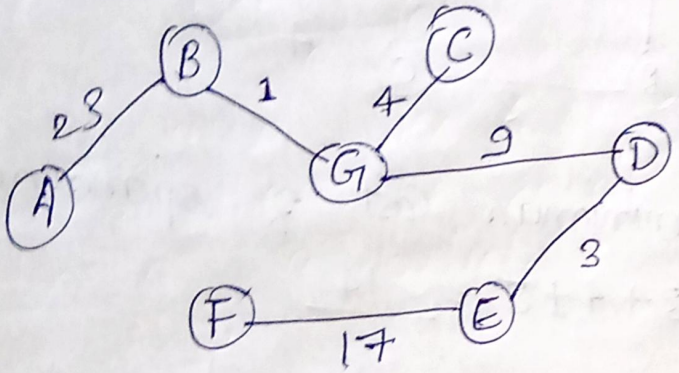


Total minimum cost —
2+3+2+1+1 = 9

Q.



Sol^{n}

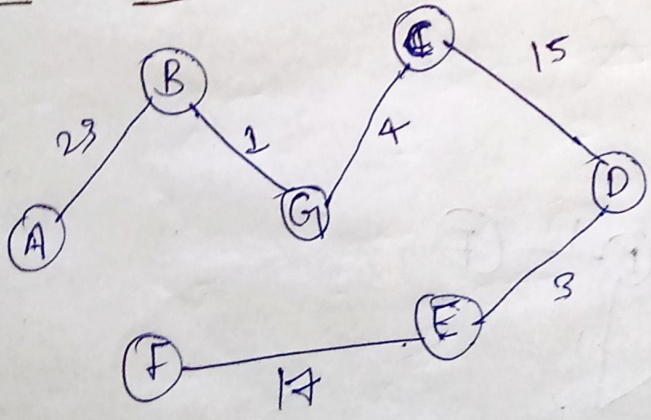


Kruskal-1

The minimum Cost -

$1+4+9+3+17+28 = 57$

Prim's Algorithm :-

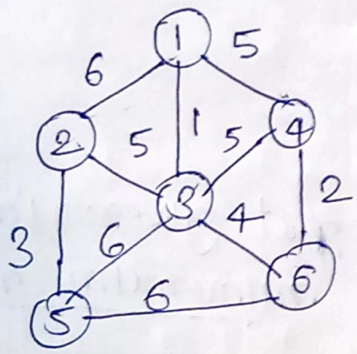


Minimum Cost :-

$23+1+9+15+3+17 = 63$

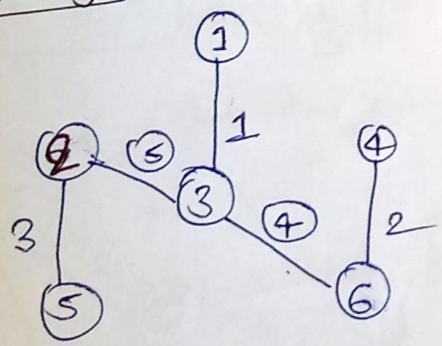
Q Construct a spanning tree and find out the minimum cost of a spanning tree of a given graph with the help of Kruskal's Algorithm

Soln:-



Kruskal's Algo:-

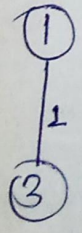
⇒



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

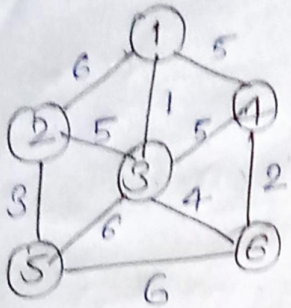
= 15

Prim's Algo.



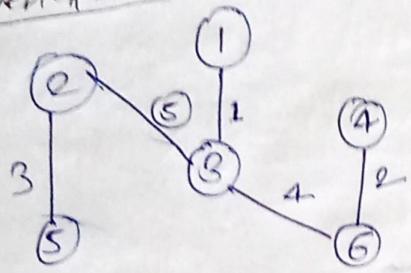
Q) Find out Minimum Cost of spanning tree of a given graph with the help of Prim's Algo.

Sol^{n}



Sol^{n}:- if initial vertex 'x' is not given in the problem then we select minimum no. of vertex as a initial.

Initial Vertex = 1



Minimum Cost = 15

Vertex 1 = 1-2 (6), (1-3 (1)), 1-4 (5)

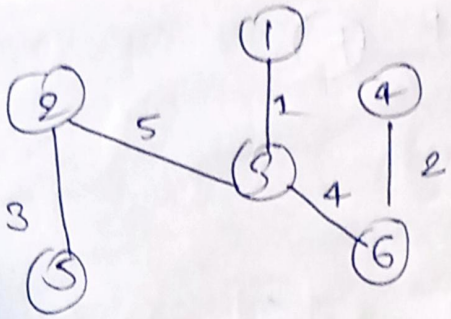
Vertex 3 = (3-2 (5)), 3-4 (5), (3-6 (4)), 3-5 (6)

Vertex 6 = (6-4 (2)), 6-5 (6)

Vertex 4 = ——

Vertex 2 = (2-5 (3))

Vertex 5 = ——

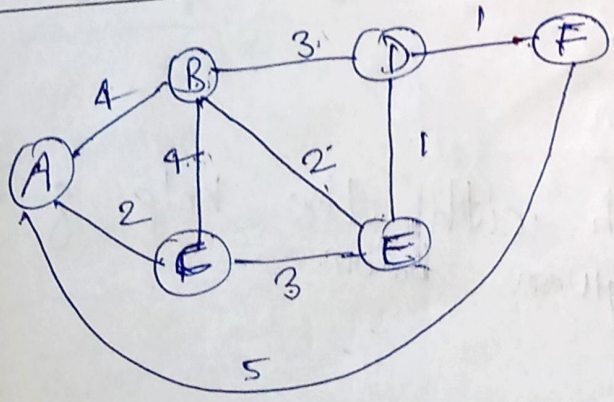


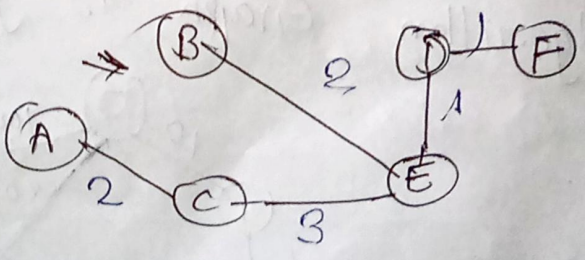
spanning tree

Total minimum cost of vertex = 5+3+1+4+2

= 15

Q. Prim's Algo :- where initial vertex X = F





Vertex F :- (F-D 1) F-A 5

Vertex D :- (D-E 1) D-B 3

Vertex E :- (E-C 3) (E-B 2)

Vertex B :- B-A 4 B-C 4

Vertex C :- (C-A 2)

Vertex A :- —

Kruskal's Algorithm:-

① Sort all the edges in non-decreasing order of their weight.

② Pick the smallest edge. check if it forms a cycle, with the spanning tree formed so far. if cycle is not formed, include this edge. Else discard it

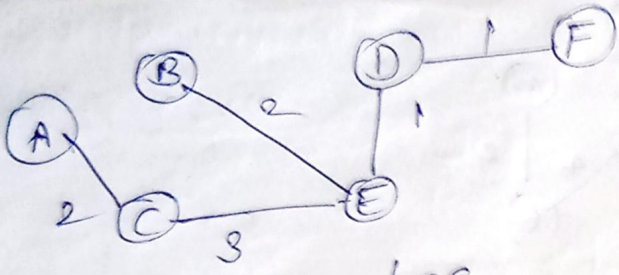
③ Repeat steps ② until there are (V-1) edges in the spanning tree.

Prim's Algo:-

① Initialize the minimum spanning tree with a vertex chose at random.

② Find all the edges that connect the tree to new vertices find the minimum and add it to tree.

③ Keep repeating steps 2 until we get a MST.



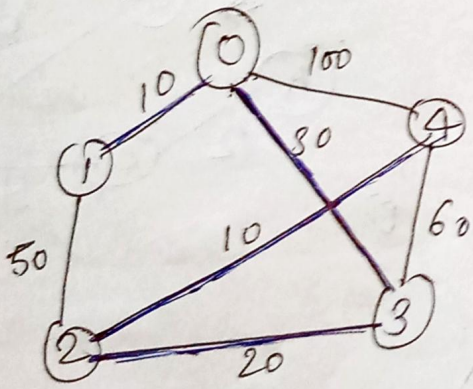
spanning tree

Total min cost of spanning tree = 1+1+2+2+3
= 9

Shortest Path :- Finding the shortest path from source to destination some Algorithm for shortest path are given below -.

① Dijiskstra Algorithm

Q. Find the shortest path with the help of Dijiskstra Algorithm Graph is given below.



Assuming vertex 0 as source and ④ as Terminal (destination)

At source vertex (0)

Adj. Vertex	Path	length
1	01	10
3	03	30
4	04	100

At source vertex (0,1)

2	012	60
3	03	30
4	04	100

At source vertex (0,1,3)

2	012	60
	032	50
4	04	100
	034	90

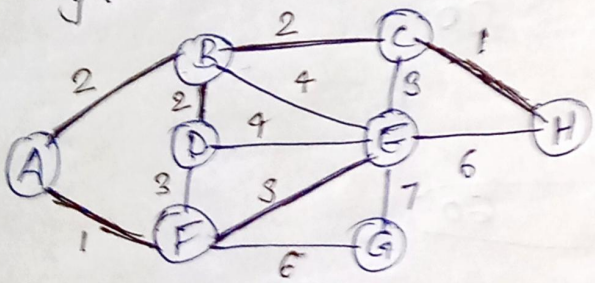
If Source Vertex (0,1,3,2)

4 — 0324 60
0324 90
04 100

So shortest path is 0824 which is 60.

= 0-3-2-4
= 30+20+10
= 60

Q. Find the shortest path of a given graph with the help of dijisktra Algorithm



A) Source (A)

Adj. vertex	Path	length
B	AB	2
F	AF	1

A) Source (A,F)

D	AFD	4
E	AFE	4
G	AFG	7
B	AB	2

A) Source (A,F,B)

C	ABC	4
E	ABE	6
D	ABD	4

D AFD 4

E AFE 4

G AFG 6

A source Vertex (A, F, B, C)

H ABCH 5

E ABCE 7

ABE 6

AFE 4

G AFG 7

D ABD 4

AFD 4

A source Vertex (A, F, B, C, D)

H ABCH 5

E ABE 6

ABCE 7

ABDE 8

AFE 4

G AFG 7

A source Vertex (A, F, B, C, D, E)

G AFEG 11

ABCE

H ABCH 5

AFEH 10

shortest path

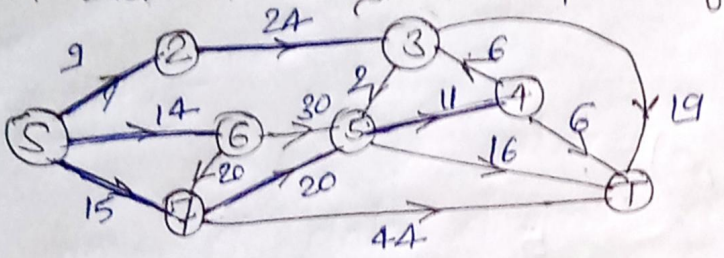
A-B-C-H

1 + 2 + 1

= 5

Q. Find out the shortest path of a given graph.

Assignment



Solⁿ

At vertex S

Adj. Vertex	Path	length.
2	S2	9
6	S6	14
7	S7	15

At vertex (S,2)

7	S7	15
6	S6	14
3	S23	33

At vertex (S,2,6)

7	S7	15
	S67	34
3	S23	33
5	S65	44

At vertex (S,2,6,7)

3	S23	33
5	S65	44
	S75	35
T	S7T	59

At vertex (S, 2, 6, 7, 3)

5 ← S235 85
5 ← 865 44
5 ← S75 35

T ← S7T 59
T ← S23T 52

At vertex (S, 2, 6, 7, 3, 5)

4 ← S754 46

T ← S75T 51
T ← S7T 59

At vertex (S, 2, 6, 7, 5, 4)

T ← S23T 52
T ← S754T 52
T ← S75T 51

Shortest path :-

5-7-5-T

= 51