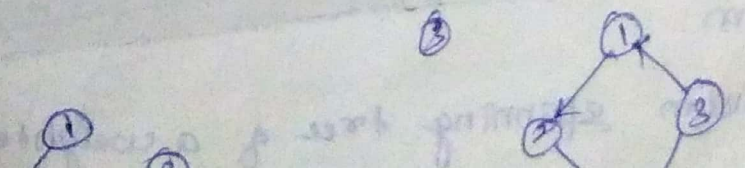
Difference between Trees and Graphs

	Trees	Graphs
Path	Tree is special form of graph i.e. **minimally connected graph** and having only one path between any two vertices.	In graph there can be more than one path i.e. graph can have uni-directional or bi-directional paths (edges) between nodes
Loops	Tree is a special case of graph having no loops, no circuits and no self-loops.	Graph can have loops, circuits as well as can have self-loops.
Root Node	In tree there is exactly one root node and every child have only one parent.	In graph there is no such concept of root node.
Parent Child relationship	In trees, there is parent child relationship so flow can be there with direction top to bottom or vice versa.	In Graph there is no such parent child relationship.
Complexity	Trees are less complex then graphs as having no cycles, no self-loops and still connected.	Graphs are more complex in compare to trees as it can have cycles, loops etc
Types of Traversal	Tree traversal is a kind of special case of traversal of graph. Tree is traversed in **Pre-Order, In-Order** and **Post-Order** (all three in DFS or in BFS algorithm)	Graph is traversed by DFS: Depth First Search and in BFS : Breadth First Search algorithm
Connection Rules	In trees, there are many rules / restrictions for making connections between nodes through edges.	In graphs no such rules/ restrictions are there for connecting the nodes through edges.
DAG	Trees come in the category of DAG : **Directed Acyclic Graphs** is a kind of directed graph that have no cycles.	Graph can be **Cyclic or A cyclic.**
Different Types	Different types of trees are : **Binary Tree , Binary Search Tree, AVL tree, Heaps.**	There are mainly two types of Graphs : **Directed and Undirected graphs.**
Applications	Tree applications : sorting and searching like Tree Traversal & Binary Search.	Graph applications : Coloring of maps, in OR (PERT & CPM), algorithms, Graph coloring, job scheduling, etc.

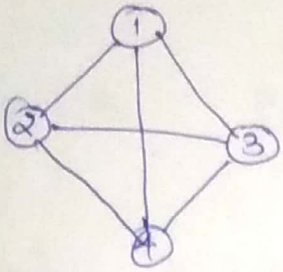


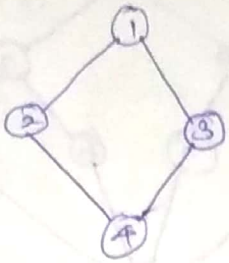
and spanning trees.

Connected component:

A connected component H of an undirected graph is a maximal connected subgraph.

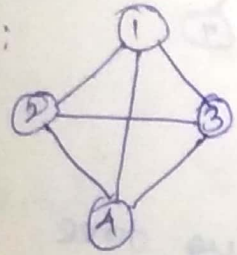
Eg:

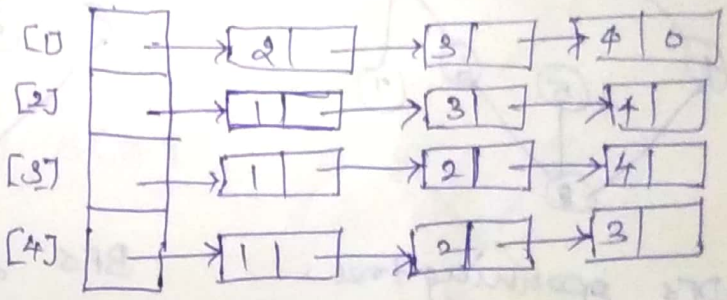




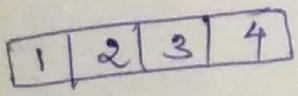
Finding connected components of graph using BFS

Eg:

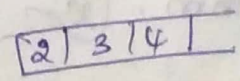


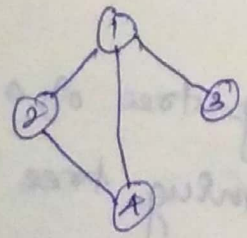


BFS:

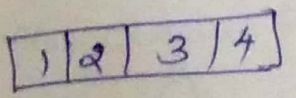


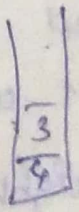
Queue



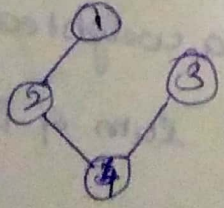


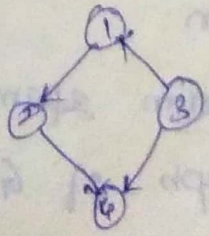
DFS:





Eg:

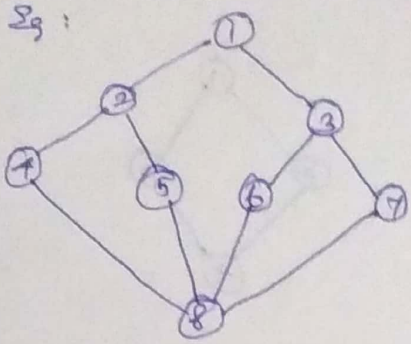


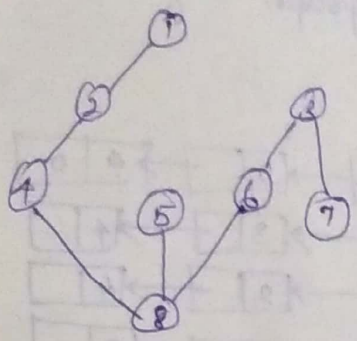


Spanning tree:

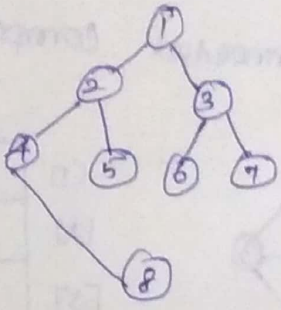
Spanning tree of an undirected graph G is a connected acyclic subgraph of G, that is a tree containing all the vertices of G.

Eg:





DFS spanning tree.



BFS spanning tree.

Prim's Algorithm:

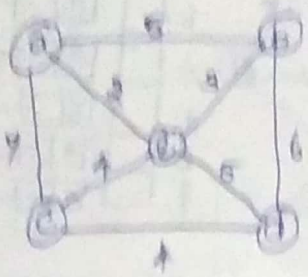
Minimum spanning tree:

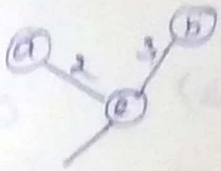
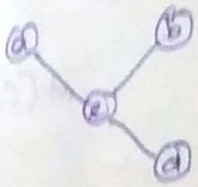
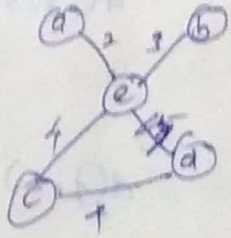
The minimum spanning tree of a weighted connected graph is its spanning tree with smallest weight where the weight is the sum of all weights of the edges.

prim's Algorithm:

A minimum spanning tree of a weighted connected graph of G(V, E) has the sum of the weight of the edges as smallest.

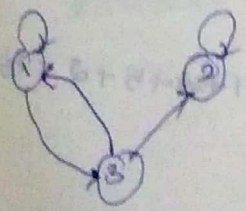
Apply Prim's algorithm and find the minimum spanning tree for the given graph

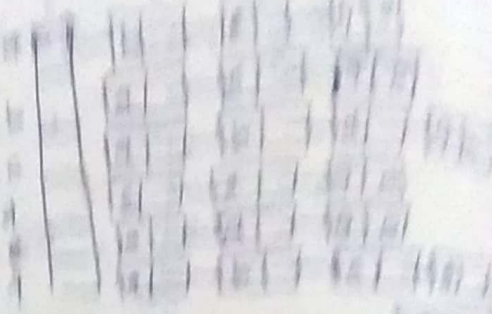


Tree vertices	Remaining vertices	Illustration
a(-,-)	b(a,5), c(a,7) d(a,∞), e(a,2)	
e(a,2)	b(e,3), c(e,4) d(e,5)	
b(e,3)	c(b,4), d(b,5)	
d(e,5)	c(d,4)	

Minimum cost = 2+3+4+4 = 13.

Transitive Closure Example





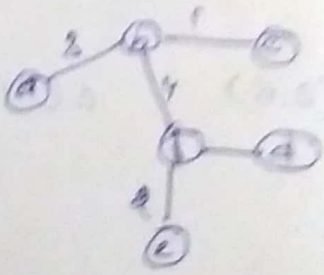
Tree vertex — Remaining vertices — Illustration

a(-,-) — b(a,3), c(-,∞), d(-,∞), e(a,1), f(a,4)

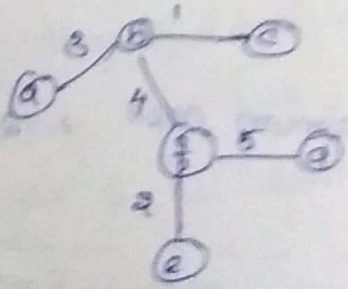
e(a,1) — c(b,1), d(-,∞), e(a,6), f(b,4)

c(b,1) — d(c,6), e(a,6), f(b,4)

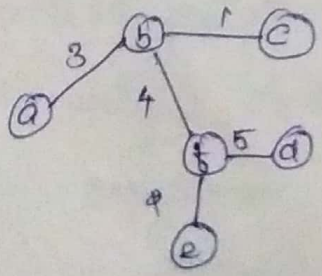
f(b,4) — d(f,5), e(f,2)



e(f,2) — d(f,5)

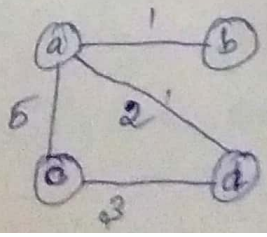


Minimum spanning tree is



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

(2)



Algorithm:

prims (G)

$V_T \leftarrow \{v_0\}$

$E_T \leftarrow \phi$

for ($l = 1$ to $v-1$) do

find a minimum weight edge $e^* = (v^*, u^*)$ among all edges (v, u) such that v is in V_T & u is in $V - V_T$

$V_T \leftarrow V_T \cup \{u^*\}$

$E_T \leftarrow E_T \cup \{e^*\}$

return E_T

Algorithm Analysis:

Time Complexity:

$T(n) = O(|E| \log |V|)$ if adjacency matrix is used

$T(n) = O(|V|^2)$

space complexity:

$S(p) = O(|V| + |V|)$

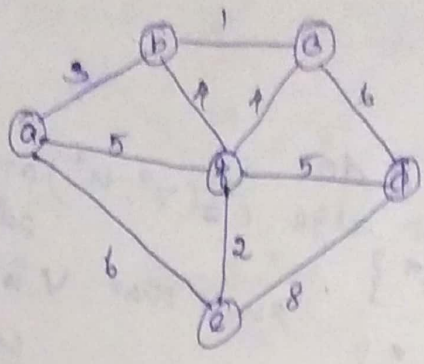
$S(p) = O(|V|^2)$

$S(p) = O(|E| + |V|)$

Kruskals Algorithm:

A minimum spanning tree for a weighted connected graph $G(V, E)$ as a acyclic subgraph with $(V-1)$ edges for which sum of the edge weights is smallest.

Q. Find the minimum spanning tree using Kruskal's Algorithm for the following graph.



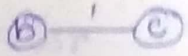
Tree Edges

sorted list of edges

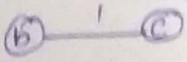
bc, ef, ab, bf, cf, df, ae, cd, ed,

1 2 3 4 4 5 6 6 8

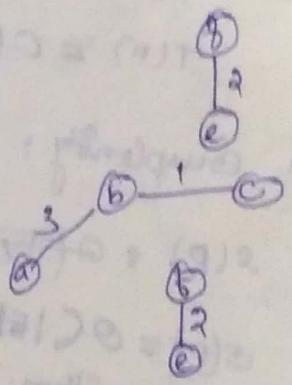
Illustration



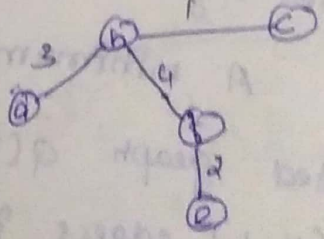
bc — ef, ab, bf, cf, df, ae, cd, ed.



ef — ab, bf, cf, df, af, ae, cd, ed

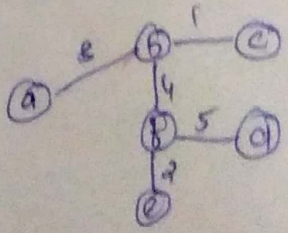


ab — bf, cf, df, af, ae, cd, ed.



bf — cf, df, af, ae, cd, ed.

cf. will create cycle so it cannot be taken



df — af, ae, cd, ed

ab, ae, ed, cd

ab will create a cycle.
so it cannot be taken.

ae, ed, cd

ae will create a cycle.

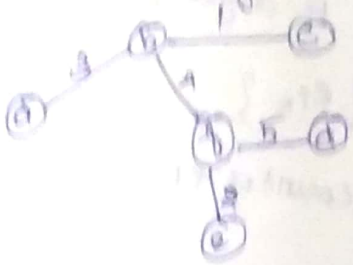
ed, cd

ed will create a cycle

cd

cd will create a cycle
so it cannot be taken

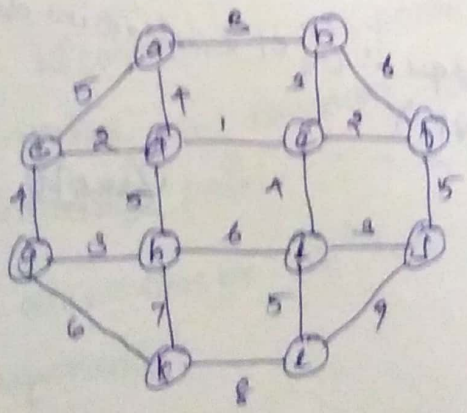
The minimum spanning tree is



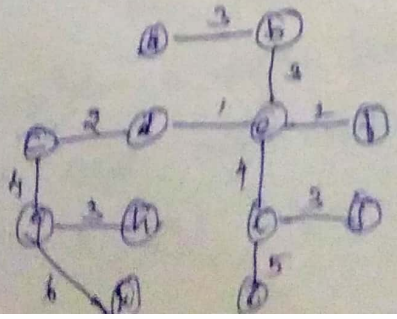
Minimum cost = 3+1+4+5+2 = 15

(3) Apply Kruskal's algorithm for the given graph and find the minimum spanning tree

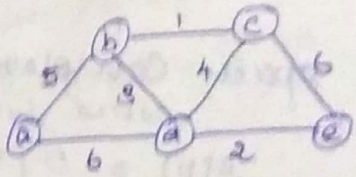
(a)



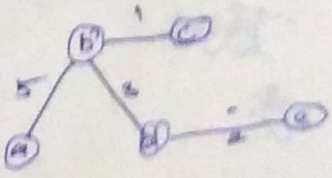
Minimum spanning tree:



(b)



MST:



Minimum cost = 11

~~1+2+3+4+3+2+1~~

= 1+2+2+3+3+3+4+4+3+5+6

= 36