# Algorithm

Lis

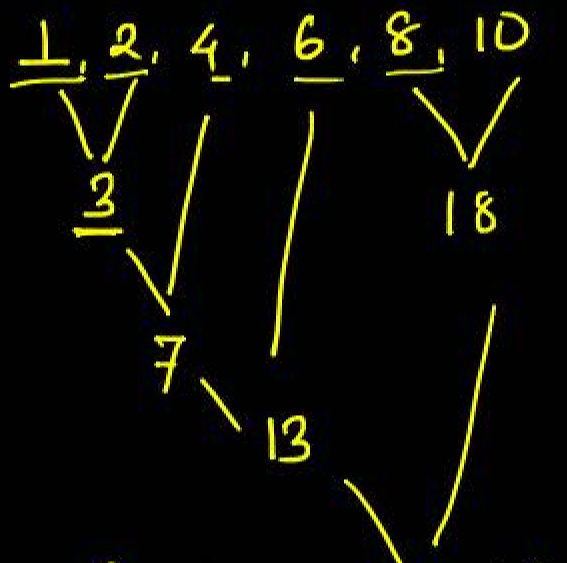
List of

Single tree node

```
procedure TREE(L, n)
for I = 1 to n - 1 do
    call GETNODE(T)
    LCHILD(T) = LEAST(L)
    RCHILD(T) = LEAST(L)
    WEIGHT(T)= WEIGHT(LCHILD(T)) + WEIGHT(RCHILD(T))
    call INSERTS( T, L)
end for
return (LEAST(L))
end TREE
```

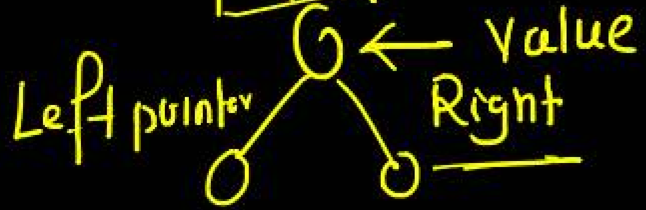
New tree Nucle

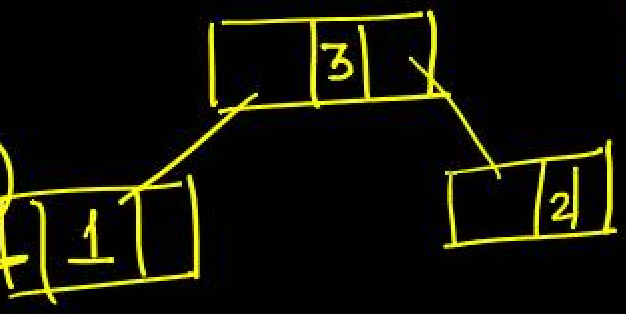
Delmin




3 4, 6, 8, 10

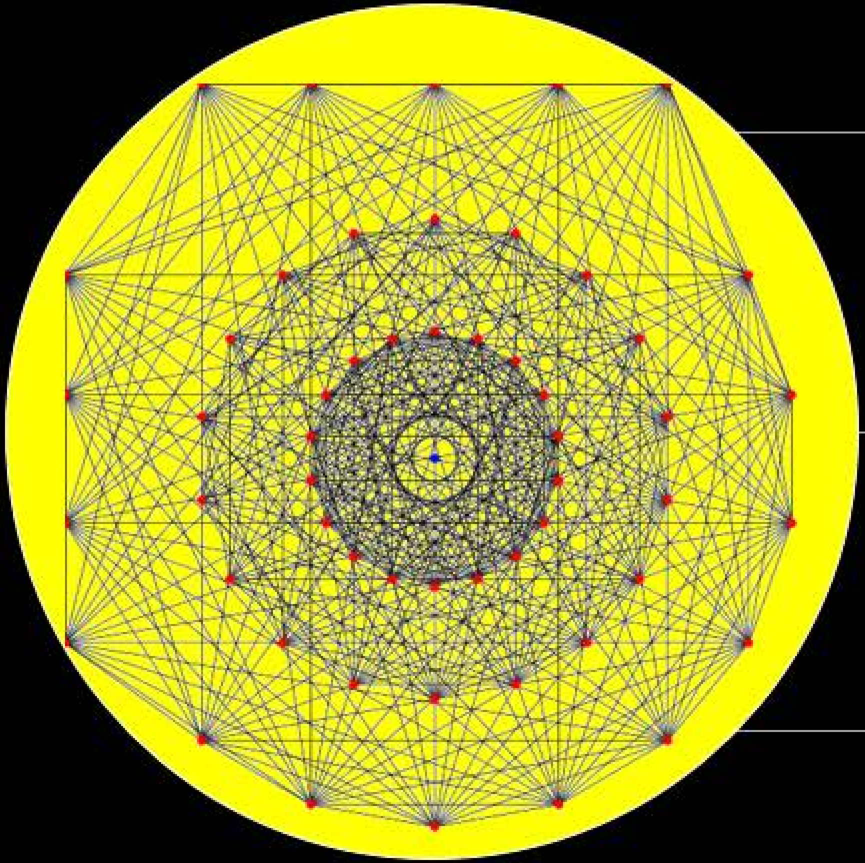
7, 6, 8, 10




$\Theta(n \log n)$

Least (L) returns the minimum weight in List and delete the the tree Node



# GRAPH THEORY

Completion of syllabus

\+

Revision & Test

Textbook

work book

bits - byte

practice

Remember concept

C - 3 Question Algorithm

$\boxed{40} \Leftarrow$ test More

$\Rightarrow$ Database + CD + TOC + OS + CN + CO

Regular $\boxed{DL}$ - 1

6 - 9

4+2 -

DM + Algorithm.

# Definitions - Graph

$G = (V, E)$

- V is set of vertices
- E is set of edges connecting pair of vertices

Graph . Data structure

Discrete Mathatics : Mathematical aspect

Algorithm : Minimum spanning tree

shortest path

Logical way is

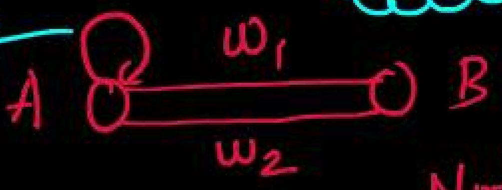
Various Real Life problem is ~~u~~ modeled by Graph.

- ~~Connect~~ Road connectivity between cities
- Build power Network;
- Network for communication
- pipe for waterflow.
- Social Networking

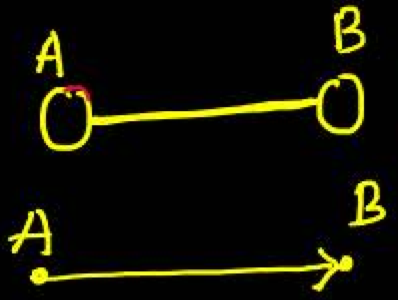
# Simple Graph

Simple Graph is graph in which No multiple ~~ect~~ edge between pair of vertex and No self loop

self loop ←




Not allowed

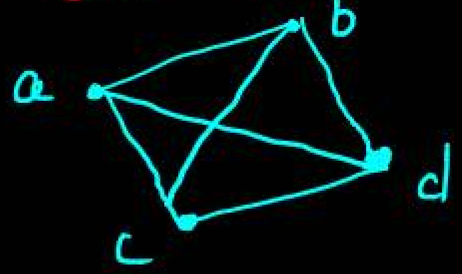



undirected

B is reachable from a

Directed graph

Q. How many maximum No. of edges possible for a Simple graph with n vertices?




4 edges

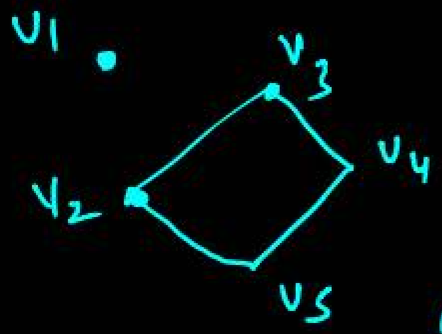
n = 4. the answer is 6

# *Definitions - Graph*

A generalization of the simple concept of a set of dots, links, edges or arcs.

*Definiton:* Graph $G = (V, E)$ consists set of vertices denoted by $V$, or *by* $V(G)$ and set of edges $E$, or $E(G)$

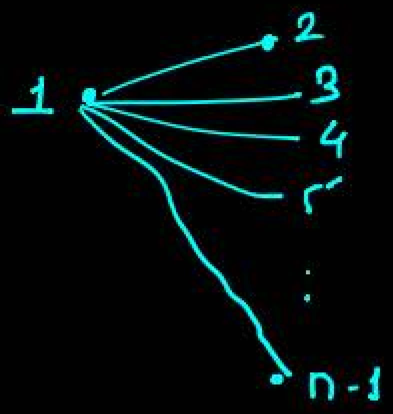
# Varying Applications (examples)

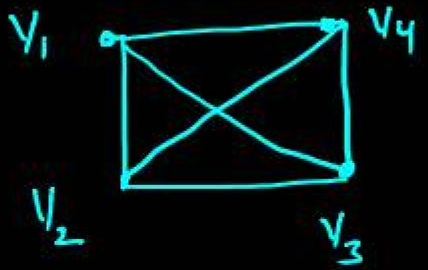



Isolated vertex is present

disconnected graph

Each vertex is reachable then we call is connected


$3+2+1$

$n-1+n-2+\cdots+1$

$$n-1+n-2+\cdots+1 = \frac{n(n-1)}{2}$$

- Edge between every pair of vertices. $^nC_2 = \frac{n(n-1)}{2}$

# Varying Applications (examples)

- Computer networks
- Distinguish between two chemical compounds with the same molecular formula but different structures.
- Solve shortest path problems between cities
- Scheduling exams and assign channels to television stations
- Register Allocation of Compiler Design

# Graph Representation :-

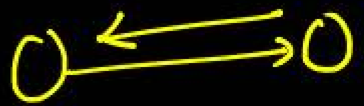
Simple graph :- yes connected graph
can have multiple edges

- Adjacency list
- Adjacency Matrix
- Incident Matrix

10 vertices

Maximum No. of edges possible so that graph will be disconnected

$^{9}C_2 = \frac{9 \times \cancel{8}4}{\cancel{2}} = 36$



Not multiple edge

15

Kolkata 1400 DelHi

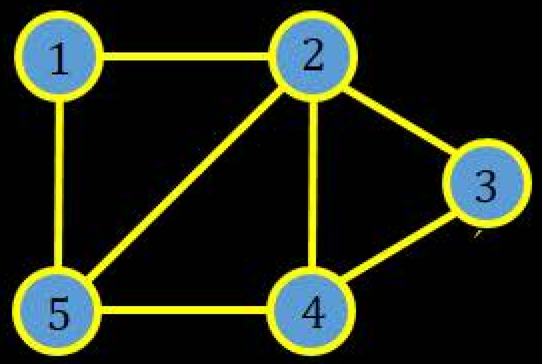
Rajdhani express

today 1 mark question

# *Adjacency List*

- The adjacency-list representation of a graph $G = (V, E)$ consists of an array *Adj* of $|V|$ lists, one for each vertex in $V$.
- For each $u \in V$, the adjacency list `Adj[`$u$`]` contains all the vertices such that there is an edge $(u, v) \in E$.
- That is, `Adj[`$u$`]` consists of all the vertices adjacent to $u$ in $G$. (Alternatively, it may contain pointers to these vertices.)

# Adjacency List




multigraph undirected

directed

Array of pointer

Node

Linked List

1 → 2 → 5

2 → 1 → 4 → 3 → 5

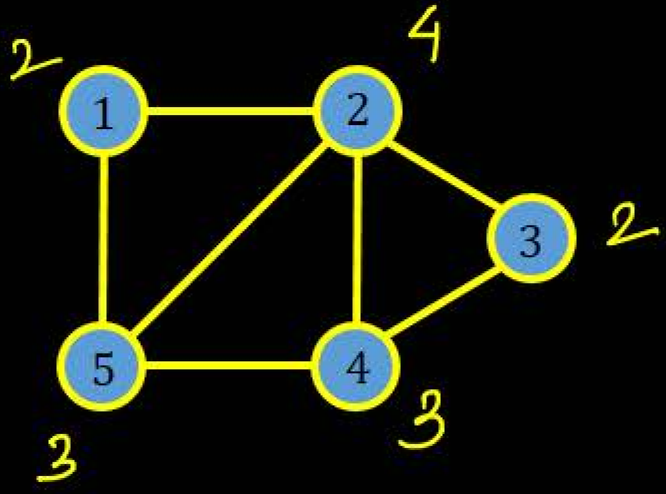
3 → 2 → 4

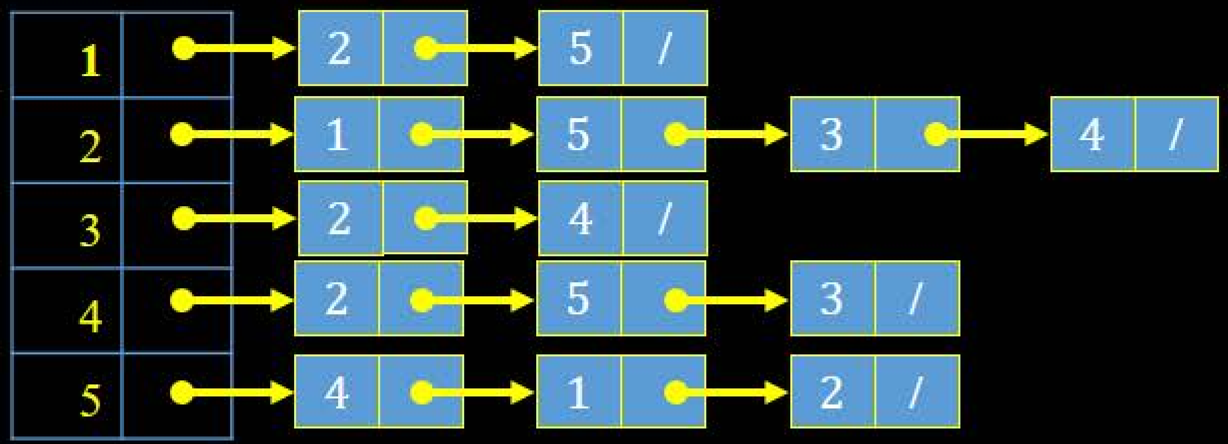
4 → 2 → 5 — 3

5 — 1 → 2 → 4

Adjacency List Representation of Simple Graph

# Adjacency List




Adjacent vertices

Adjacent to 5 is 1, 2 4

total Space Required

V vertices

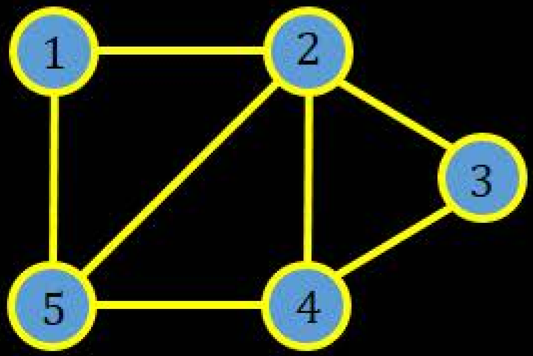
$$V + \sum_{v \in V} \deg(v)$$

$$V + 2E = \Theta(V+E)$$

Space Complexity

# Adjacency Matrix

- $a_{ij} \begin{cases} 1 & if\ (i,j) \in E \\ 0 & otherwise \end{cases}$




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

Adjacency List / matrix

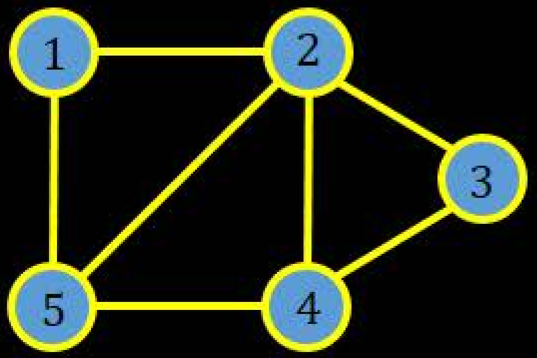
A Space for $V \times V$

$\boxed{\Theta(V^2)}$ Space Complexity

Space Complexity for this Representation

# *Adjacency Matrix*

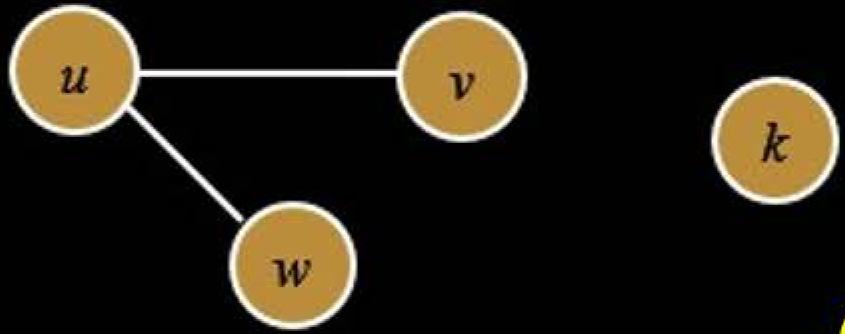
- $a_{ij} \begin{cases} 1 & if\ (i,j) \in E \\ 0 & otherwise \end{cases}$




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

# Degree of a Vertex

- Degree of Vertex




Undirected graph

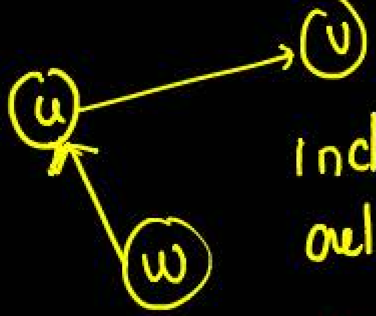
- No. of edges incident on a vertex.

$\deg(u) = 2$

$\deg(v) = 1$

$\deg(w) = 1$

$\deg(k) = 0$



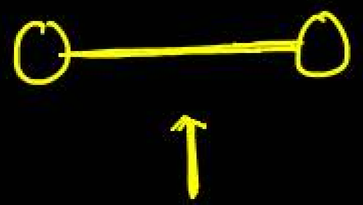

indegree(u) : 1

out degree(u) : 1

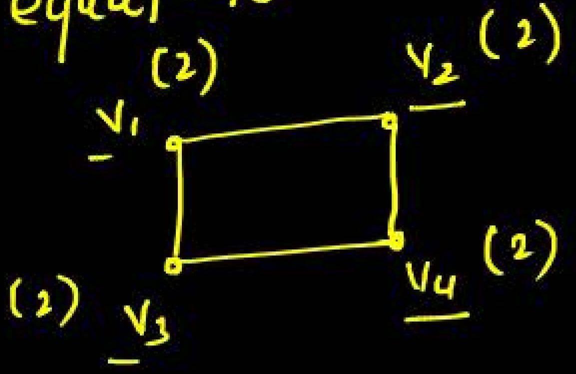
Directed graph

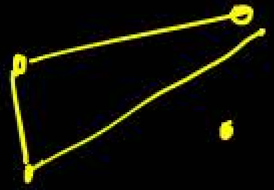
# The Handshaking Theorem

$$2e = \sum_{v \in V} \deg(v)$$



- Sum of the degree of all vertices is equal to twice the No. of edges



$2E = 6$

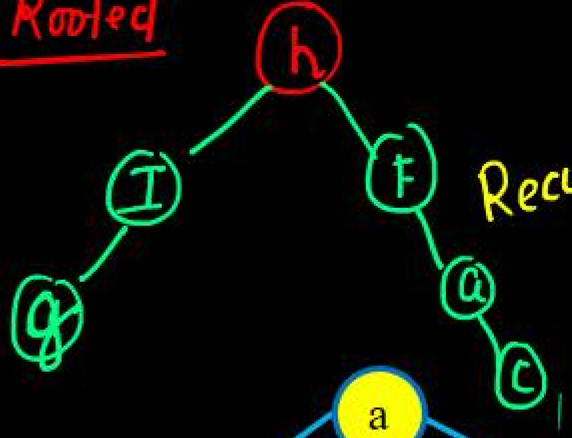
$\underline{E = 3}$

Sum of degree = 8

$2E = 8$

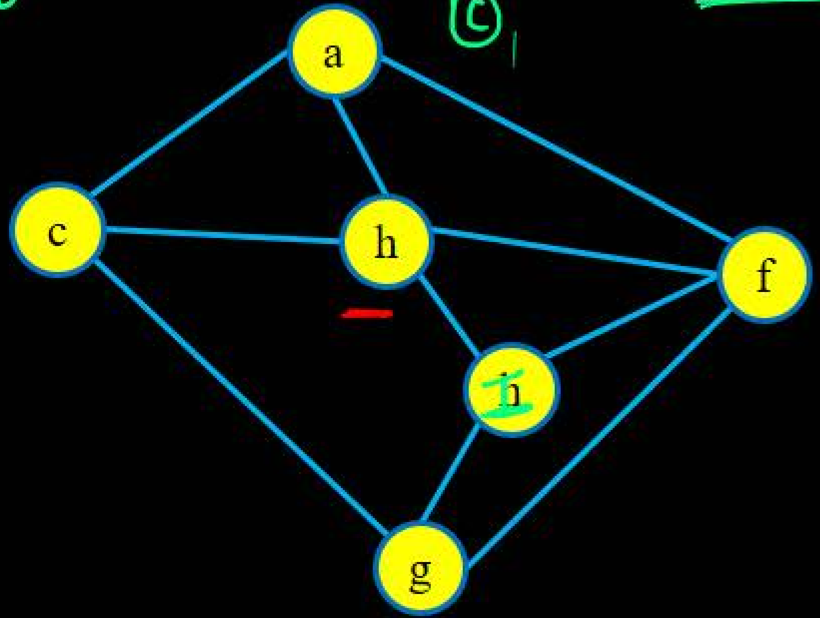
$E = \underline{4}$

Not a Rooted




Recursion tree

Root

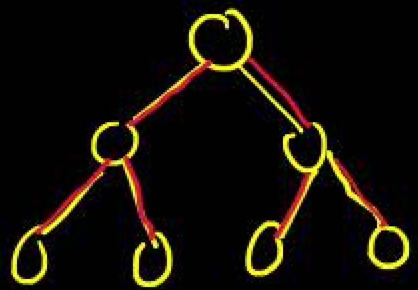



# Spanning Tree

Tree

Spanning tree

CBT (Complete Binary tree)



- A tree is Never disconnected.
- A tree Never contain cyle.
- tree is simple ~~ay~~ acyclic. connected graph.

A tree having 7 (n) vertices havin No. of edge 6 (n-1)

Subgraph of Graph

$G' = (V, E')$ $E' \subset E$

# Spanning Tree

**Spanning tree:** Spanning tree of a graph is Subgraph $G'(V, E')$, $E' \subseteq E$ ~~and~~ such that the given Subgraph is a Tree.

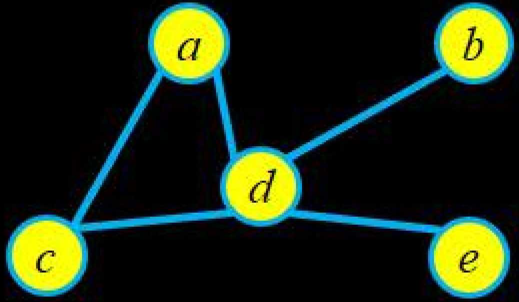
n vertice

( n-1 . edges, acyclic, connected)

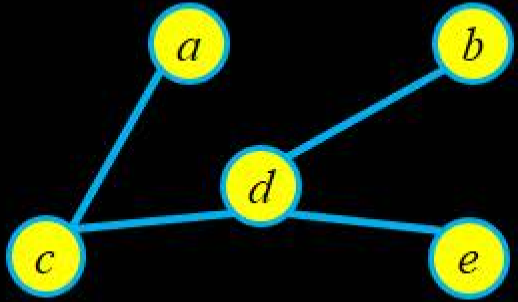
# Spanning Tree

**Spanning tree:** A sub graph $T$ of a undirected graph $G = (V, E)$ is a spanning tree of $G$ if it is a *tree* and contains every vertex of $G$.

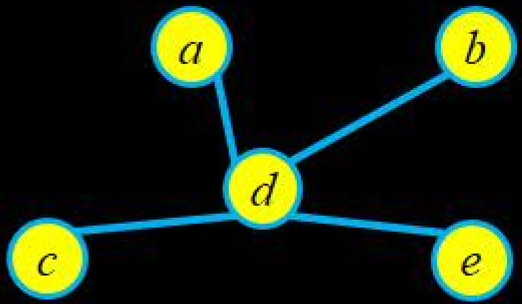
(n-1)



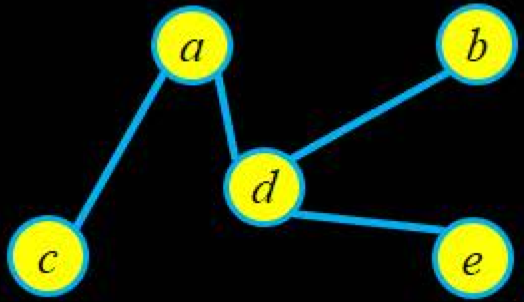

Graph




Spanning Tree-1




Spanning Tree-2




Spanning Tree-3

# Weighted Graph

## Weighted Graphs:

depends what model graph Represent we can assign some weight to it.

edges are assign weight

distance

a — b

City 1 City 2

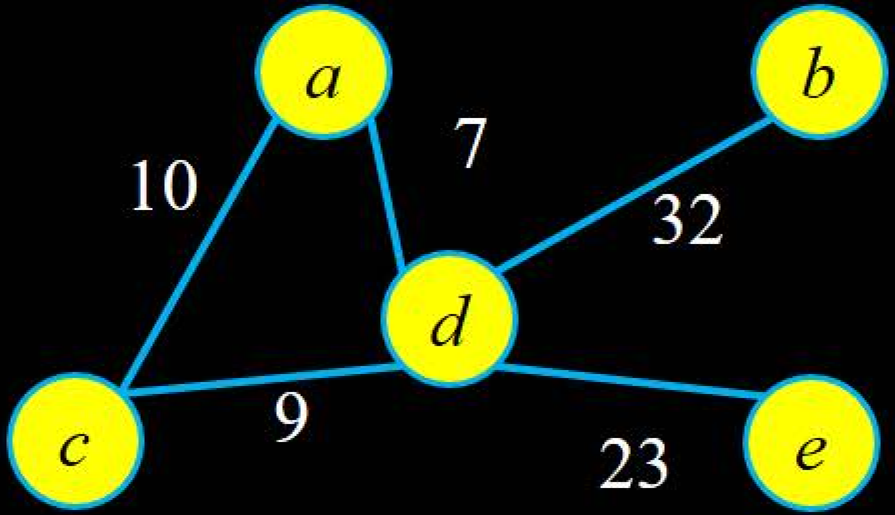
time Required

Bandwidth

$R_1$ — $R_2$

Link

delay - ~~prog~~ propagation delay



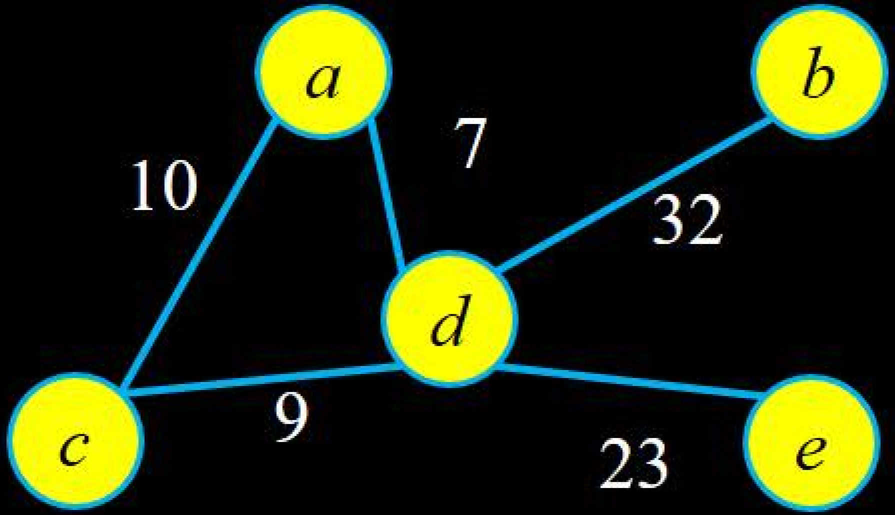

Graph

81

weight of the graph is Sum of weights of the edges.

# Weighted Graph

Weighted Graphs: A weighted graph is a graph, in which each edge has a weight (some real number). Weight of the graph may represent distance, cost, and bandwidth. The weight of the graph is sum of the weights of all edges.




Graph

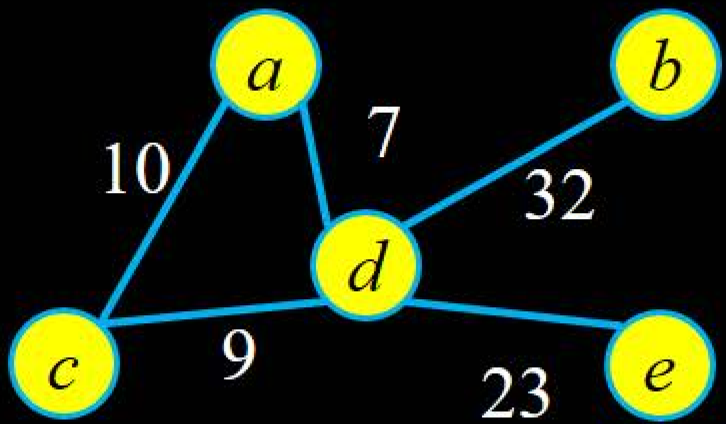
# Weighted Graphs Sum

The weight of the graph is sum of the weights of all edges. The weight of the tree is 74

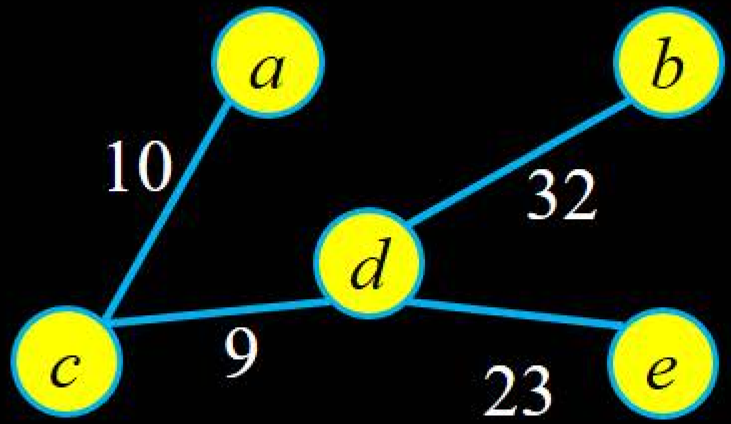
Spanning tree

Connected with minimum cost

81 - 7 = 74

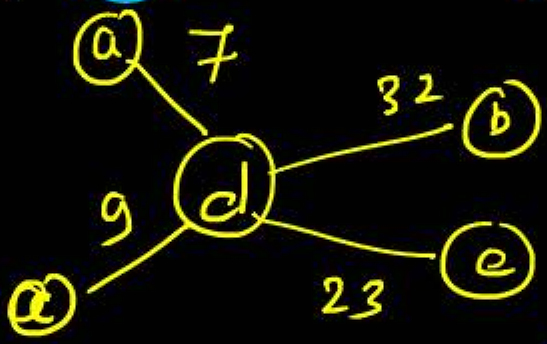



Graph




Graph

MST →




71

spanning

# Minimum Spanning Tree

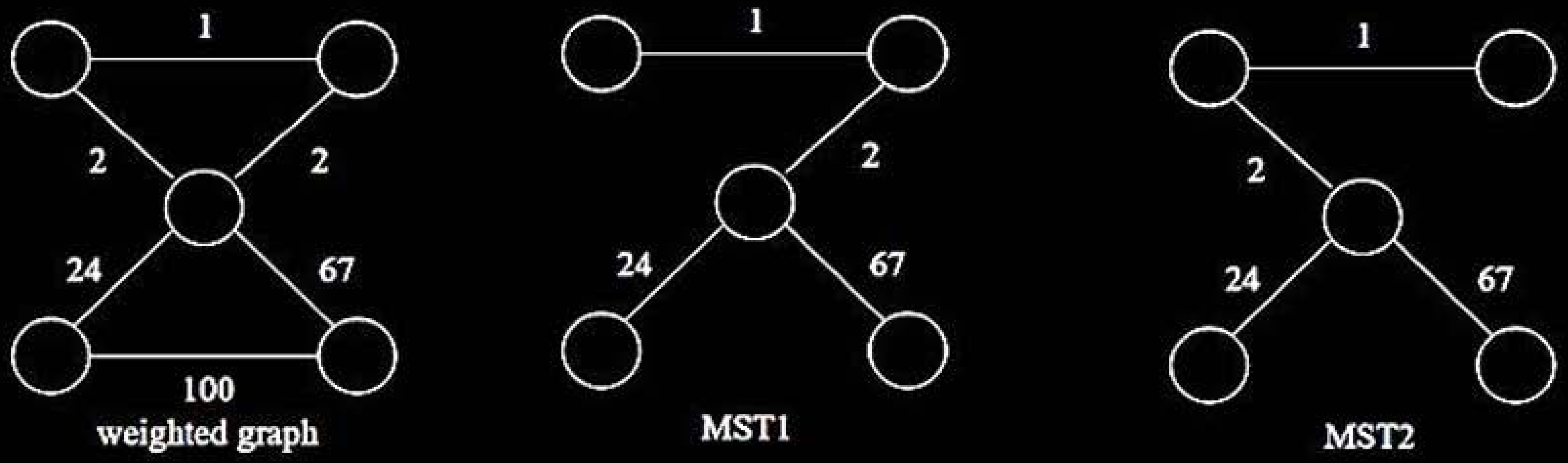
Minimum spanning tree of a graph is a Subgraph $G'(V, E')$, $E' \subseteq E$. is a tree and the weight of the tree. is Minimum.

# Minimum Spanning Tree

- A Minimum Spanning Tree in an undirected connected weighted graph is a spanning tree of minimum weight (among all spanning trees).

# Minimum Spanning Tree

- *Note*: *The minimum spanning tree may not be unique.* However, if the weights of all the edges are pairwise distinct, it is indeed unique (we won't prove this now).



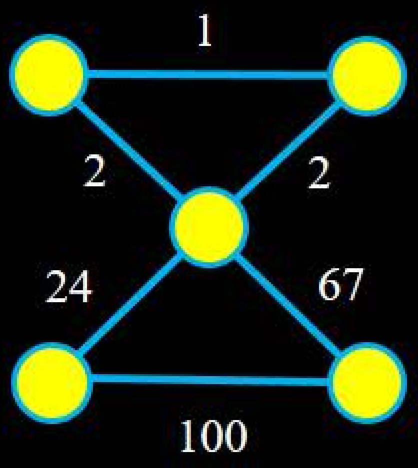

Weighted graph

# Minimum Spanning Tree

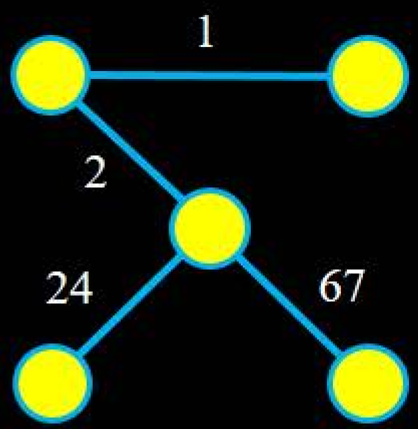
- *Note*: *The minimum spanning tree may not be unique.*

MST will be unique iff edge weights are distinct.

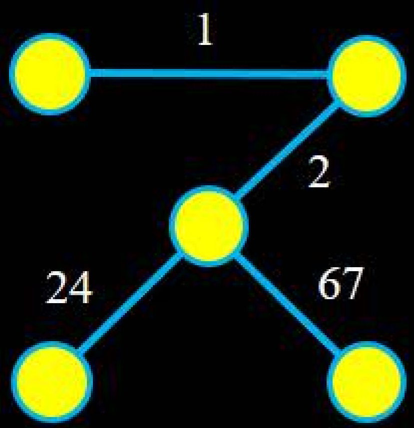
MST




Weighted graph




Tree-1




Tree-2

# Application

In order to minimize the cost of

- 
  - power Network
  - ~~ear~~ Network connectivity
  - Speech Recognition system.
  - piping Layout.

# Application

In order to minimize the cost of

- power networks,
- wiring connections,
- piping,
- automatic speech recognition, etc.,

# Counting Spanning Trees

- If G is itself a tree, then number of spanning tree is 1.

# Counting Spanning Trees

- When G is the cycle graph $\underline{C_n}$ with n vertices, then number of spanning tree is $\underline{n}$.

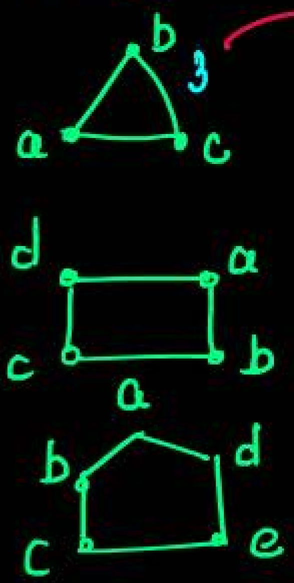
n vertices
n - edge

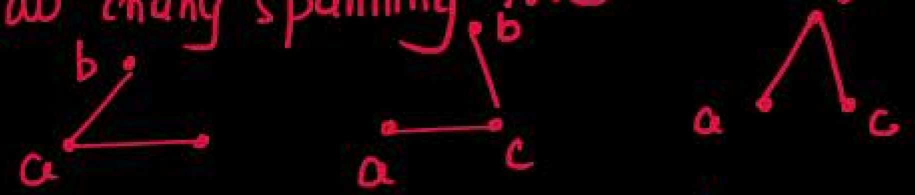
How many spanning tree

$C_3$ :

$C_4$ :

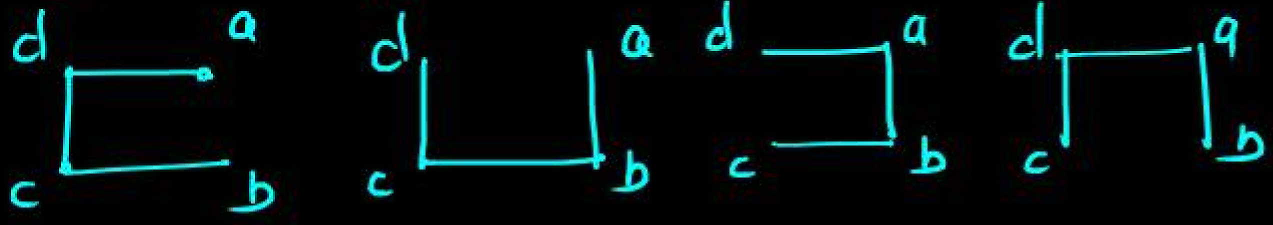
$C_5$




— No. of spanning tree

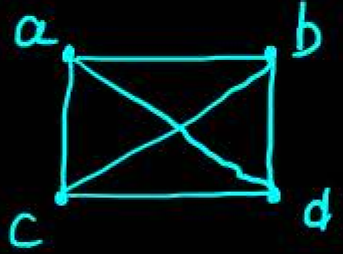
$\underline{3}$

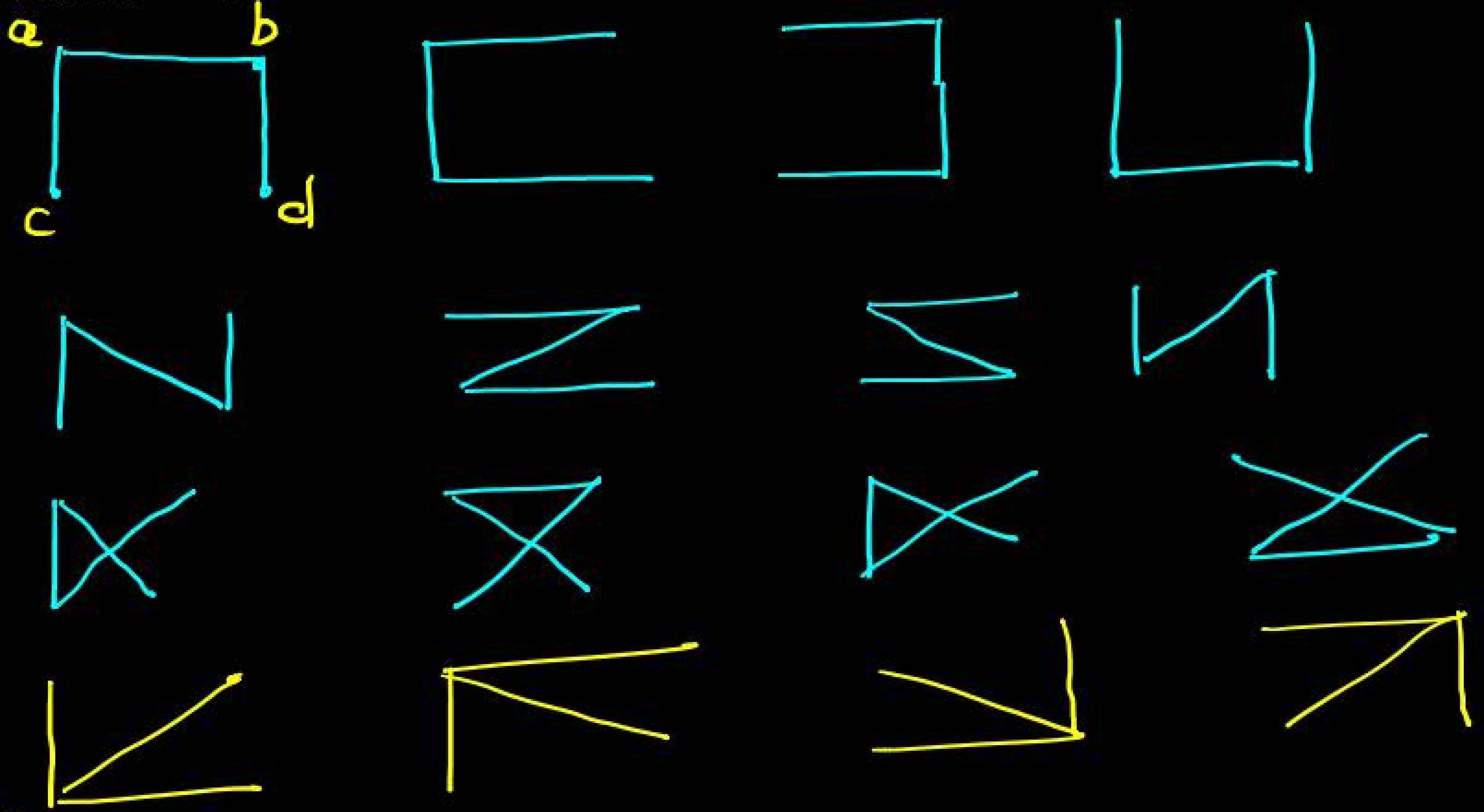



$C_n$ No. of spanning tree is $\underline{n}$

# Counting Spanning Trees

- For a complete graph with n vertices, Cayley's formula gives the number of spanning trees as $n^{n-2}$.




16 spanning tree

$4^{4-2} = 4^2 = \underline{16}$

Complete graph n vertice - $\underline{n^{n-2}}$

$n = 5 = 5^3 = \underline{125}$

# Counting Spanning Trees

- If G is the complete bipartite graph $K_{p,q}$ the number of spanning tree is $p^{q-1}q^{p-1}$

# Problem Statement

- Given a graph $G(V,E)$, the problem is to find a Subgraph $G'(V,E')$, $E' \subseteq E$ such that $G'$ is a tree with minimum cost.

# Problem Statement

- Given a connected weighted undirected graph $G$, design an algorithm that outputs a minimum spanning tree (MST) of $G$.

# Algorithm

- Prims Algorithm
- Kruskal Algorithm
- Which data structure support deletion of ~~mini~~ min element ?

Simple

Idea

Array – ordered
unorder

Min – Heap

$\Theta(\log_2 n)$ time

# Prim's Algorithm

- Prims algorithm grows like a tree.
- we have have a source from where construction of spanning tree begin.
- At each iteration, the edge with minimum cost is picked and added to the tree.
- This process is repeated till No more edge to be considered

# Prim's Algorithm

- The *Prim's* algorithm- it grows a single tree and adds a light edge in each iteration.
  - Start by picking any vertex r to be the root of the tree.
  - While the tree does not contain all vertices in the graph find shortest edge leaving the tree and add it to the tree.

Heap

Spanning tree

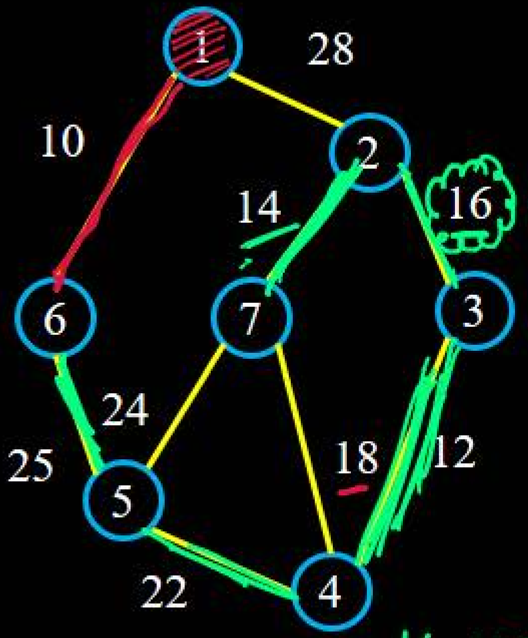
~~treee~~ tree

# Example

$7 \text{ time } \overline{\log(v)} \circ \sum_{v \in V} (1 + \deg v)$

$(V + 2E)(\log v)$

$\sum_{v \in V} \left( \log(v) + \deg(v) * \log(v) \right)$




(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| — | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| — | 28 | ∞ | ∞ | 25 | — | ∞ |
| — | 28 | ∞ | 22 | — | — | 24 |
| — | 28 | 12 | — | — | — | 18 |
| | 16 | — | — | — | — | 18 |
| — | — | — | — | — | — | 14 |
| — | — | — | — | — | — | — |

← Initialization

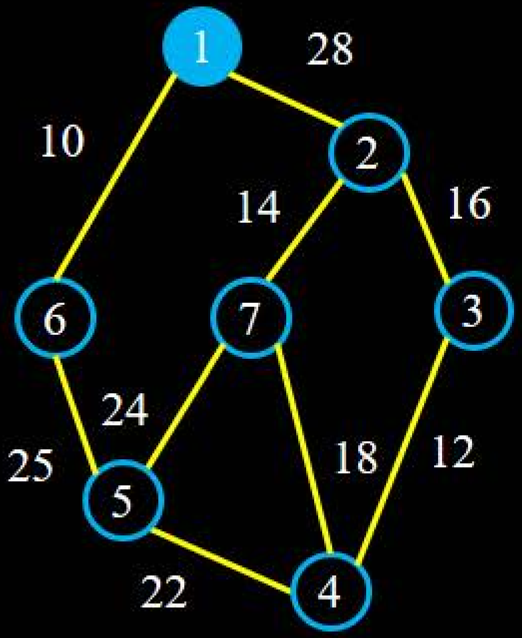
16<28

18<24

14<18

Empty

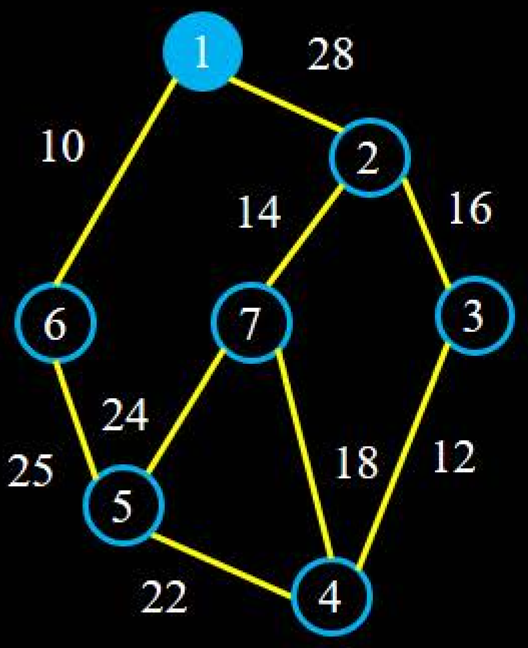
# Example



(a)

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

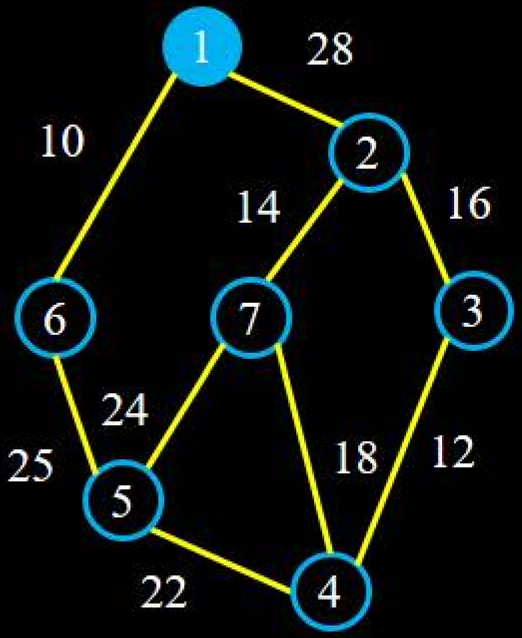
# Example




(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

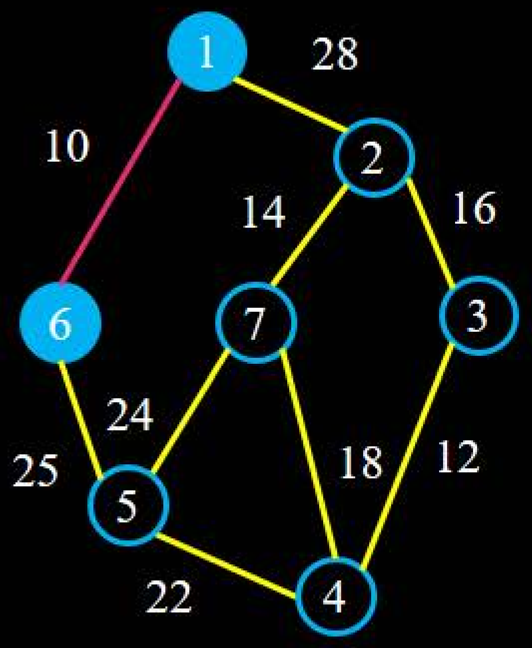
# Example



(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

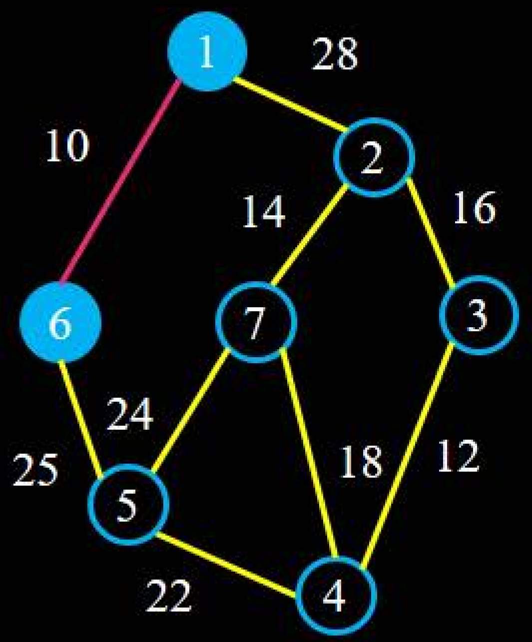
# Example



(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

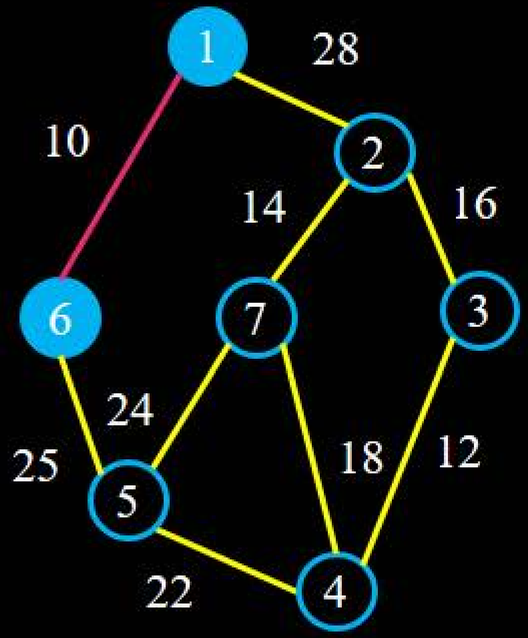
# Example




(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | 25 | - | ∞ |
| | | | | | | |
| | | | | | | |
| | | | | | | |

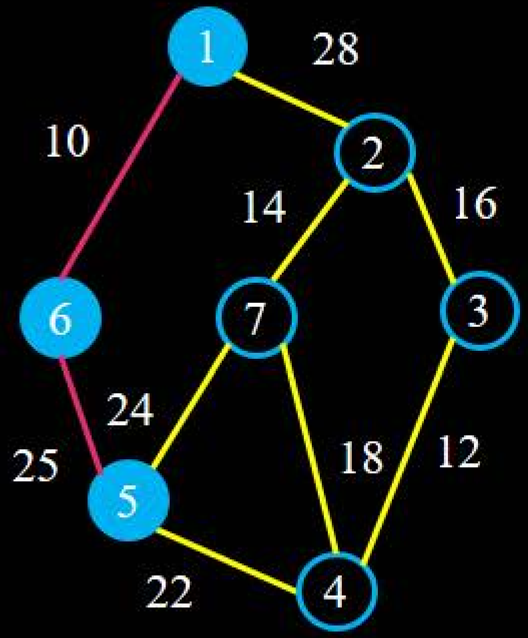
# Example




(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | 25 | - | ∞ |
| | | | | | | |
| | | | | | | |
| | | | | | | |

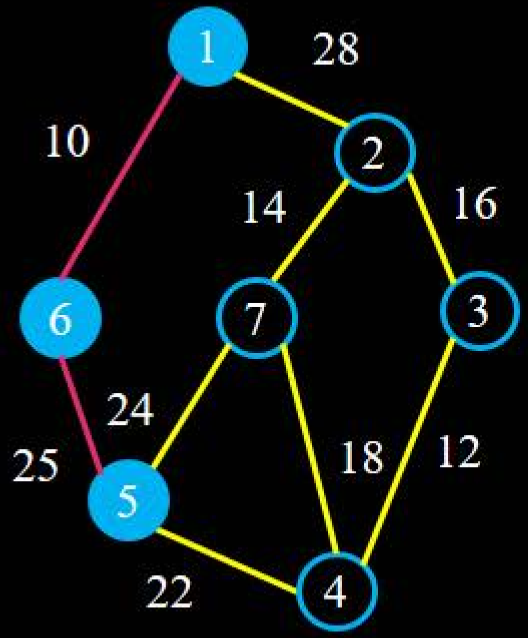
# Example



(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | - | - | ∞ |
| | | | | | | |
| | | | | | | |
| | | | | | | |

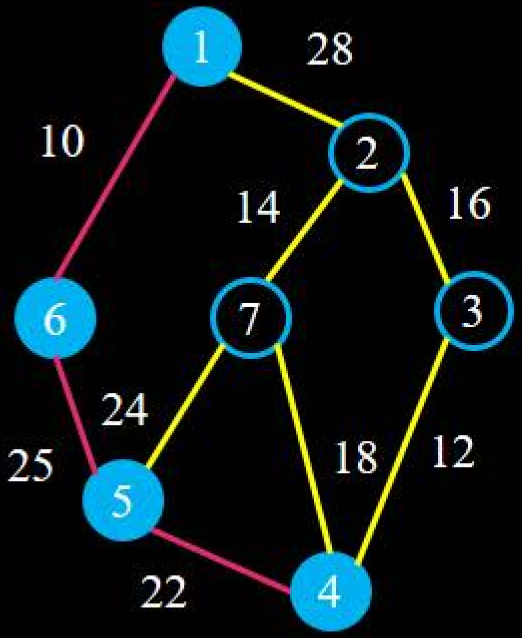
# Example




(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | - | - | ∞ |
| - | 28 | ∞ | 22 | - | - | 24 |
| | | | | | | |
| | | | | | | |

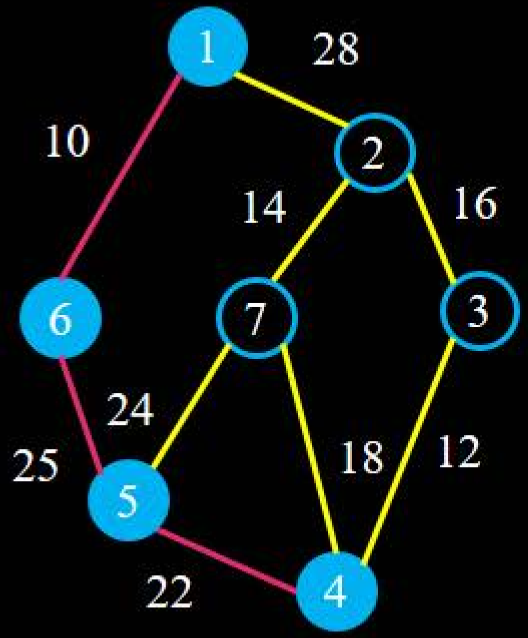
# Example




(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | - | - | ∞ |
| - | 28 | ∞ | - | - | - | 24 |
| | | | | | | |
| | | | | | | |

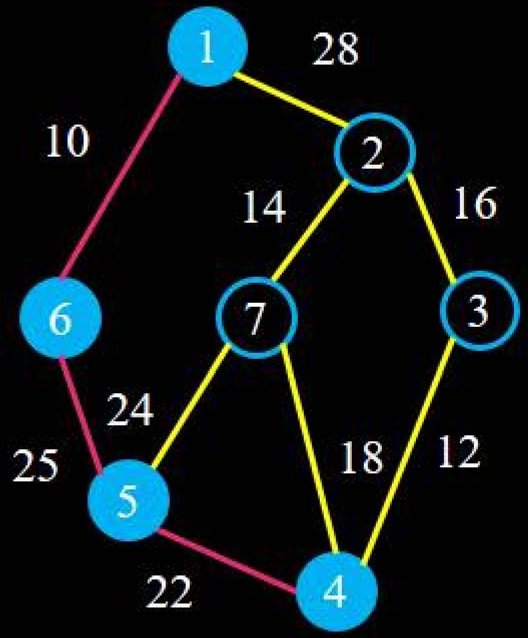
# Example




(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | - | - | ∞ |
| - | 28 | ∞ | - | - | - | 24 |
| - | 28 | 12 | - | - | - | 18 |
| | | | | | | |

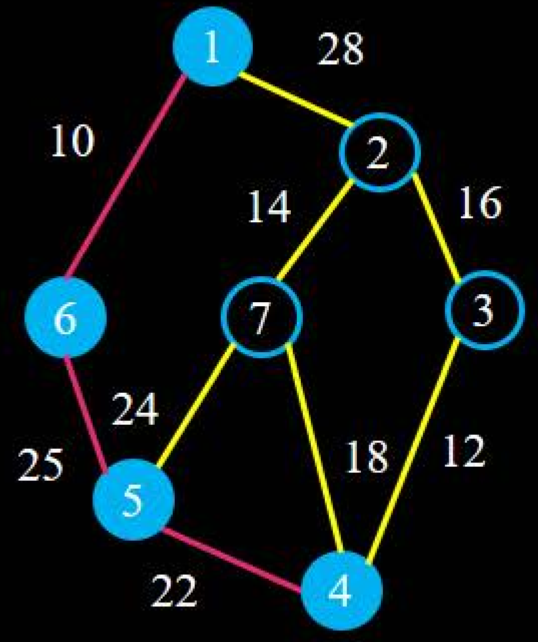
# Example




(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | - | - | ∞ |
| - | 28 | ∞ | - | - | - | 24 |
| - | 28 | 12 | - | - | - | 18 |
| | | | | | | |

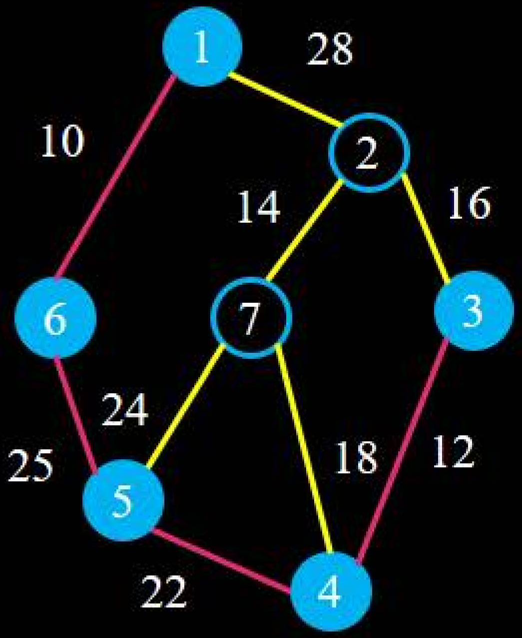
# Example



(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | - | - | ∞ |
| - | 28 | ∞ | - | - | - | 24 |
| - | 28 | 12 | - | - | - | 18 |
| | | | | | | |

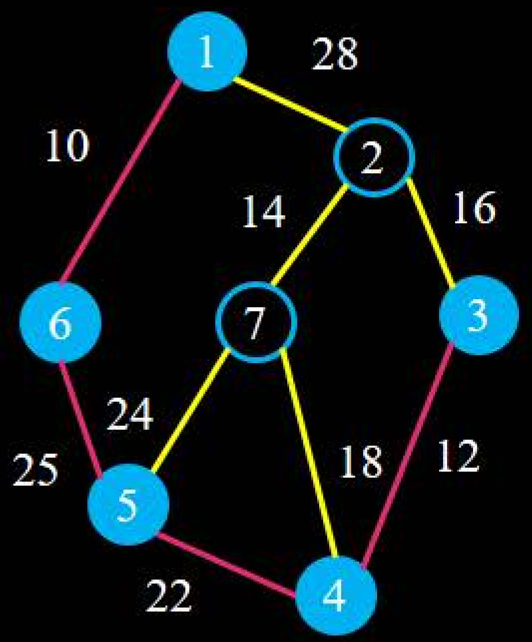
# Example



(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | - | - | ∞ |
| - | 28 | ∞ | - | - | - | 24 |
| - | 28 | - | - | - | - | 18 |
| | | | | | | |

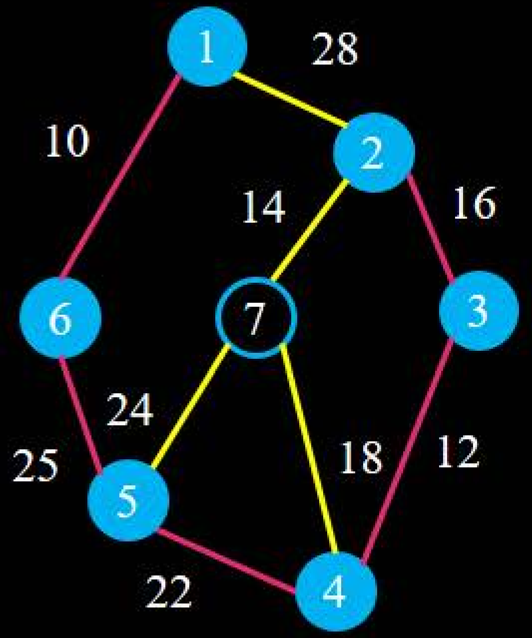
# Example



(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | - | - | ∞ |
| - | 28 | ∞ | - | - | - | 24 |
| - | 28 | - | - | - | - | 18 |
| - | 16 | - | - | - | - | 18 |

# Example

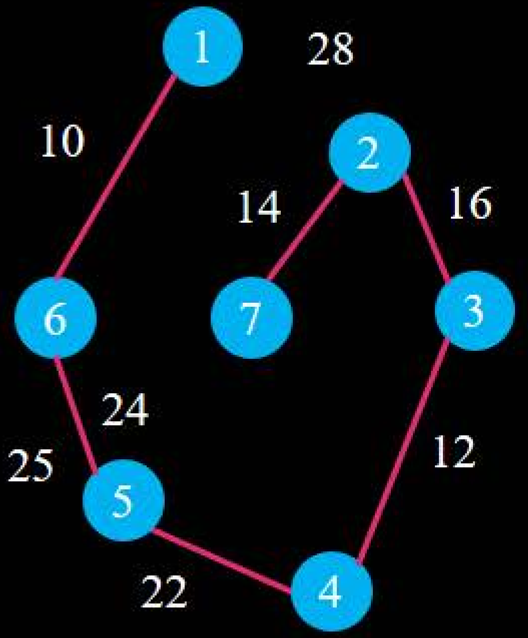



(a)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| - | 28 | ∞ | ∞ | ∞ | 10 | ∞ |
| - | 28 | ∞ | ∞ | ∞ | - | ∞ |
| - | 28 | ∞ | ∞ | - | - | ∞ |
| - | 28 | ∞ | - | - | - | 24 |
| - | 28 | - | - | - | - | 18 |
| - | 16 | - | - | - | - | 18 |
| - | - | - | - | - | - | 14 |

# The MST

Gllgonthmic approach




(a)

Algorithm

# Algorithm

```
Prims(G,w,s){
   for (each u∈V){
      d[u] = ∞;
      u.π= NIL;
   }
   d[s] = 0;
   s.π= NIL;
   Q = (queue with all vertices);
   while (Non-Empty(Q)){
     u = Extract-Min(Q);
     for (each v ∈ Adj[u])
        if (v ∈ Q and w(u,v)< d[v]){
           v.π= u;
           d.[v] = w(u,v)
        }
}
```

$\Theta(V)$ yes/NO $\frac{6}{10}$ $\frac{2}{26}$

deg(1) + deg(2)

deg(3) + . de(7)

$\boxed{2E}$

Vertex No. of times

Heap – $\log_2 V$ time

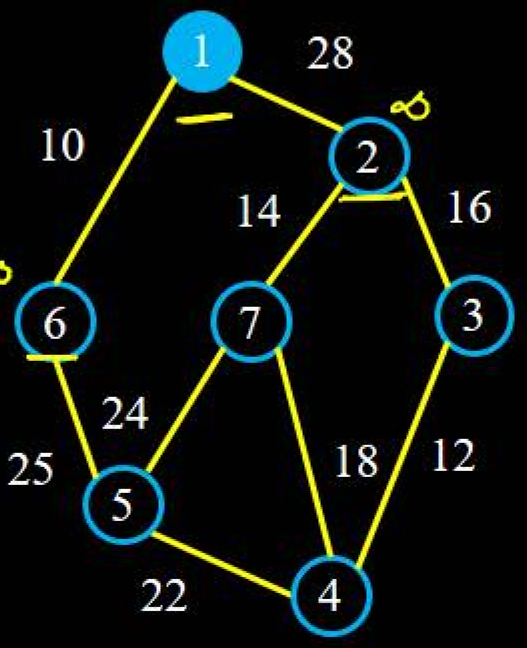
for loop will Run is for every vertex : size of Adjacency list

Cumulative $\sum_{v \in V} deg(v) = 2E$

∞ $\log_2 V$

decrease.key(Q, d[v])

~~decrea~~ decrease key $v \to 10$ $v \to 26$




(a)

# Cumulative Analysis

- Initialization takes $O(|V|)$ time
- Extract min will take $O(\log|V|)$ times
- The loop will run for adjacency list of every vertex, $\sum(\deg(u)) = 2E$
- Decrease key operation will take $O(\log|V|)$
- $\sum_{u\ in\ V}[O(\log|V|) + O(\deg(u)\log|V|)]$
- $= \log|V| \sum_{u\ in\ V}[1 + O(\deg(u))]$
- $= \log|V| O(|V| + 2|E|)$
- $= \boxed{O((|V| + |E|)\log|V|)}$

Heap

Heap data

# Kruskal's Minimum Spanning Tree Algorithm

- Kruskal's algorithm finds a safe edge to add to the growing forest by finding, of all the edges that connect any two trees in the forest, an edge (u, *v*) of least weight.

• Tree
MST cost

Kruskal

- kruskal Algorithm Sort the edge weight in decreasing order
- pick up an edge with minimum weight
  Add the edge iff it s Not forming a cycle
- If cycle is formed then edge is discarded and select the next edge.

# Kruskal’s Minimum Spanning Tree Algorithm

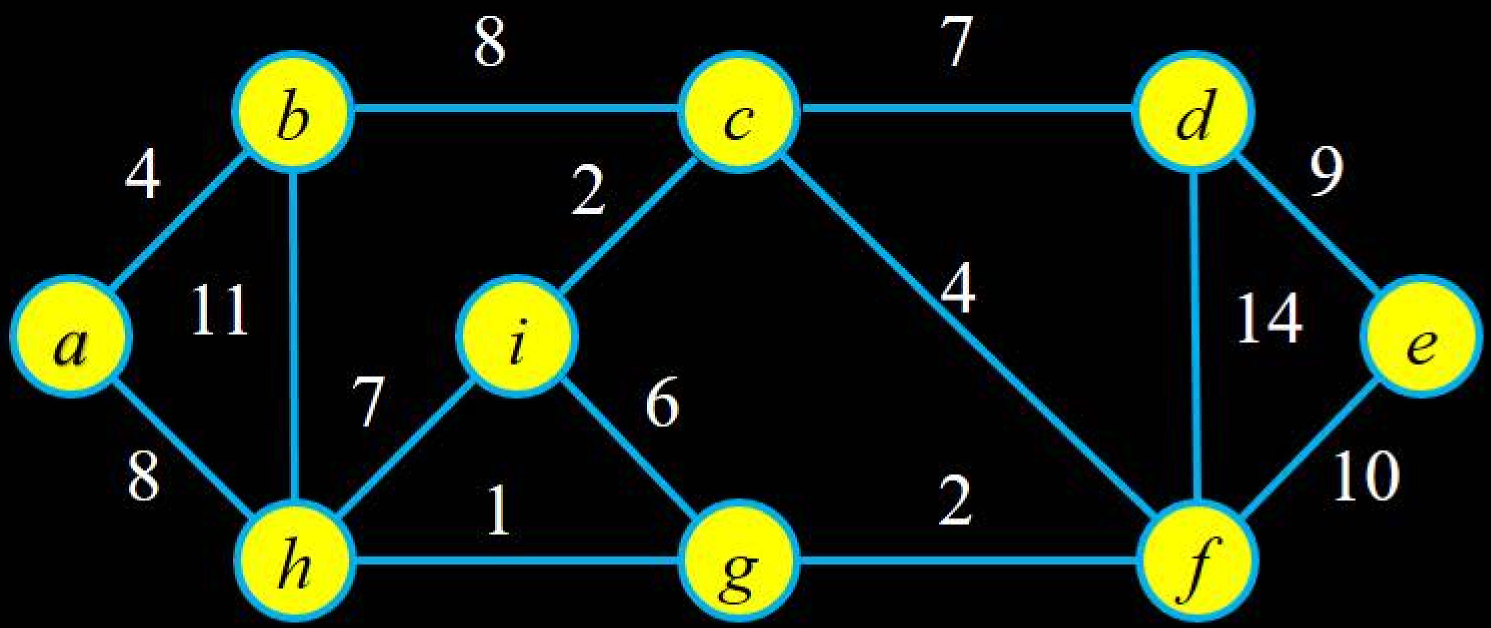
- Kruskal's algorithm qualifies as a greedy algorithm because at each step it adds to the forest an edge of least possible weight.
- It uses a disjoint-set data structure to maintain several disjoint sets of elements. Each set contains the vertices in one tree of the current forest.

# Kruskal’s Minimum Spanning Tree Algorithm

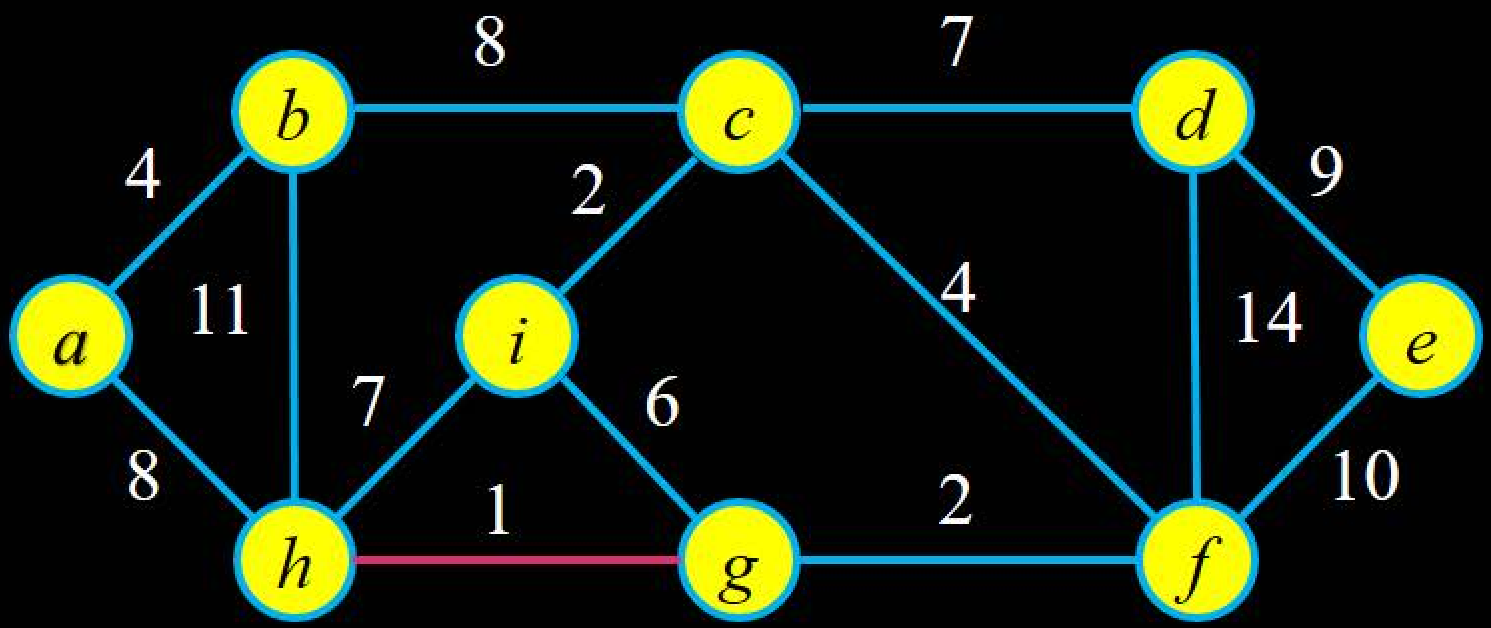
- It uses a disjoint-set data structure to maintain several disjoint sets of elements. Each set contains the vertices in one tree of the current forest.

# Example

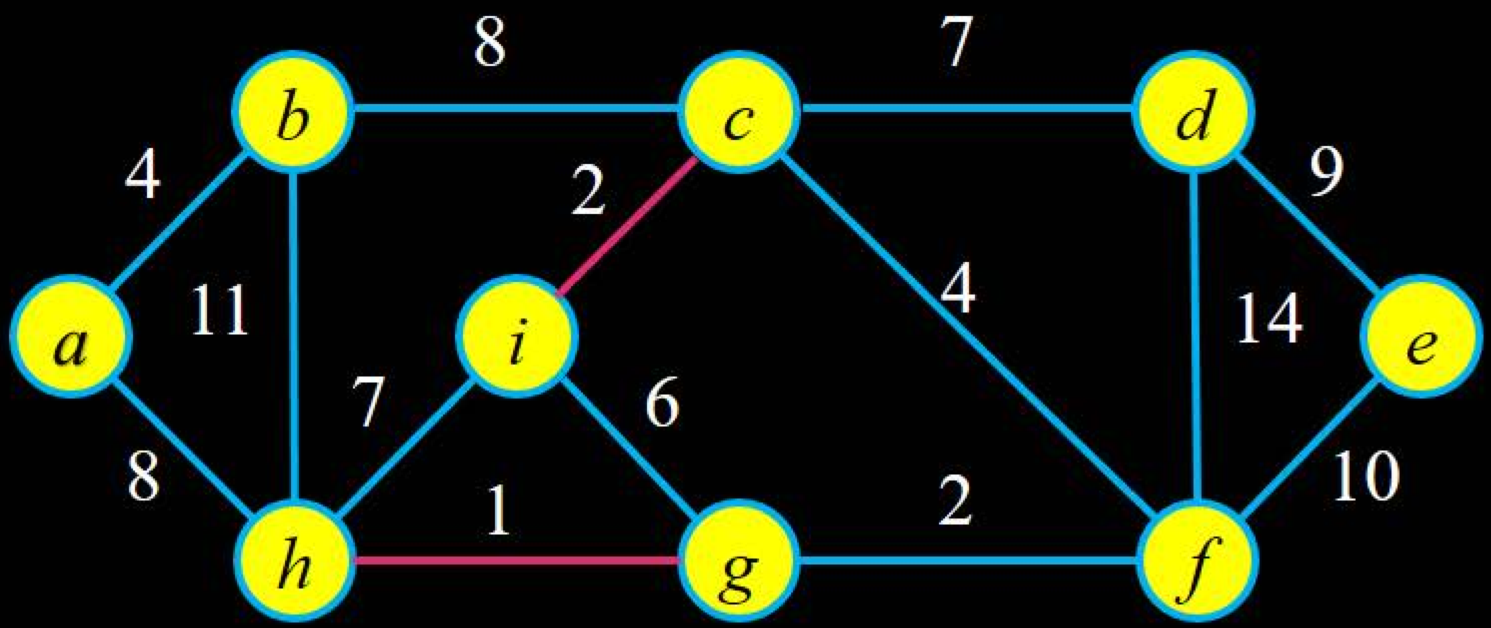
Select the edge with minimum weight



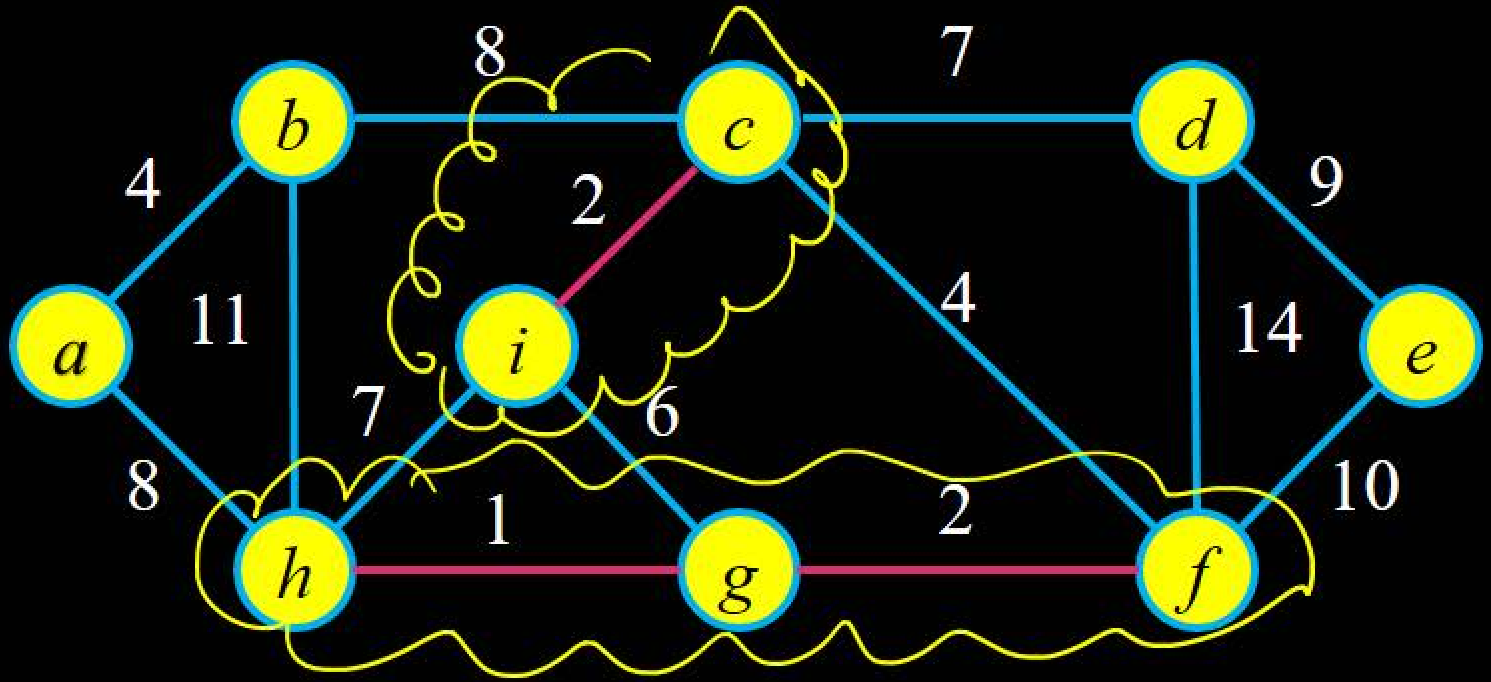
# Example



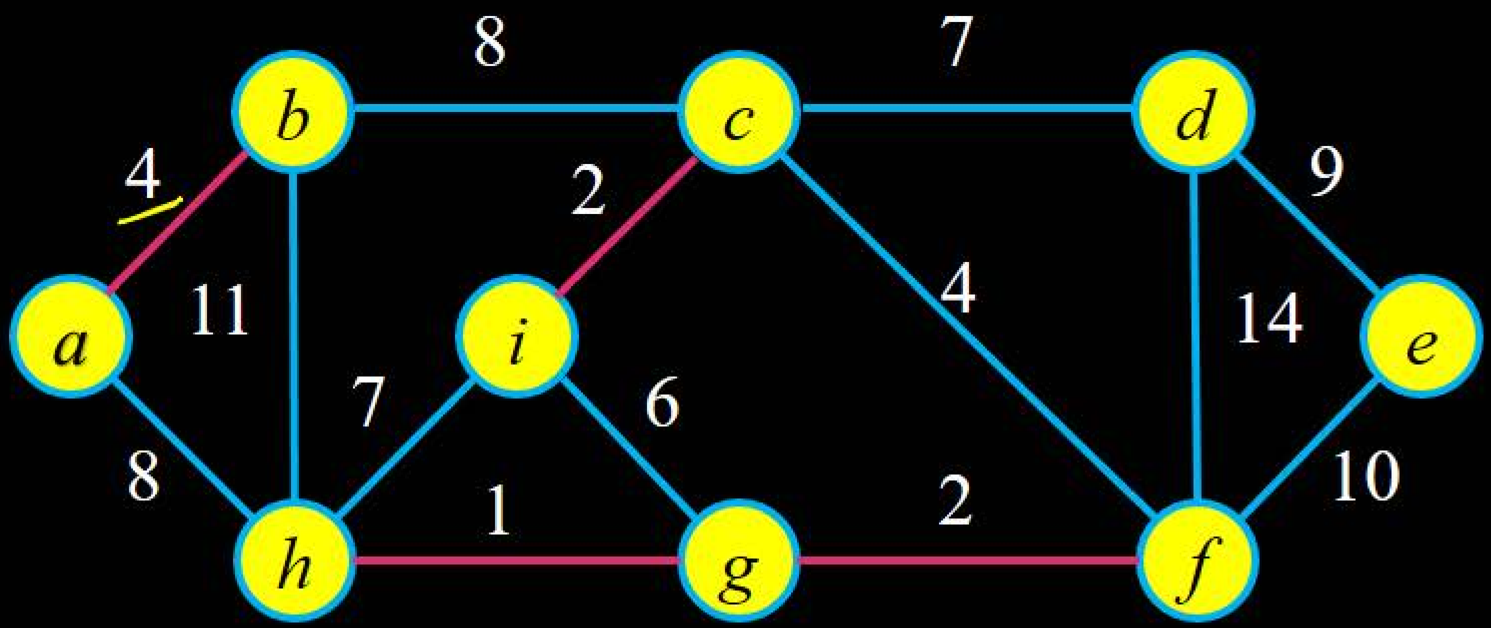
# Example



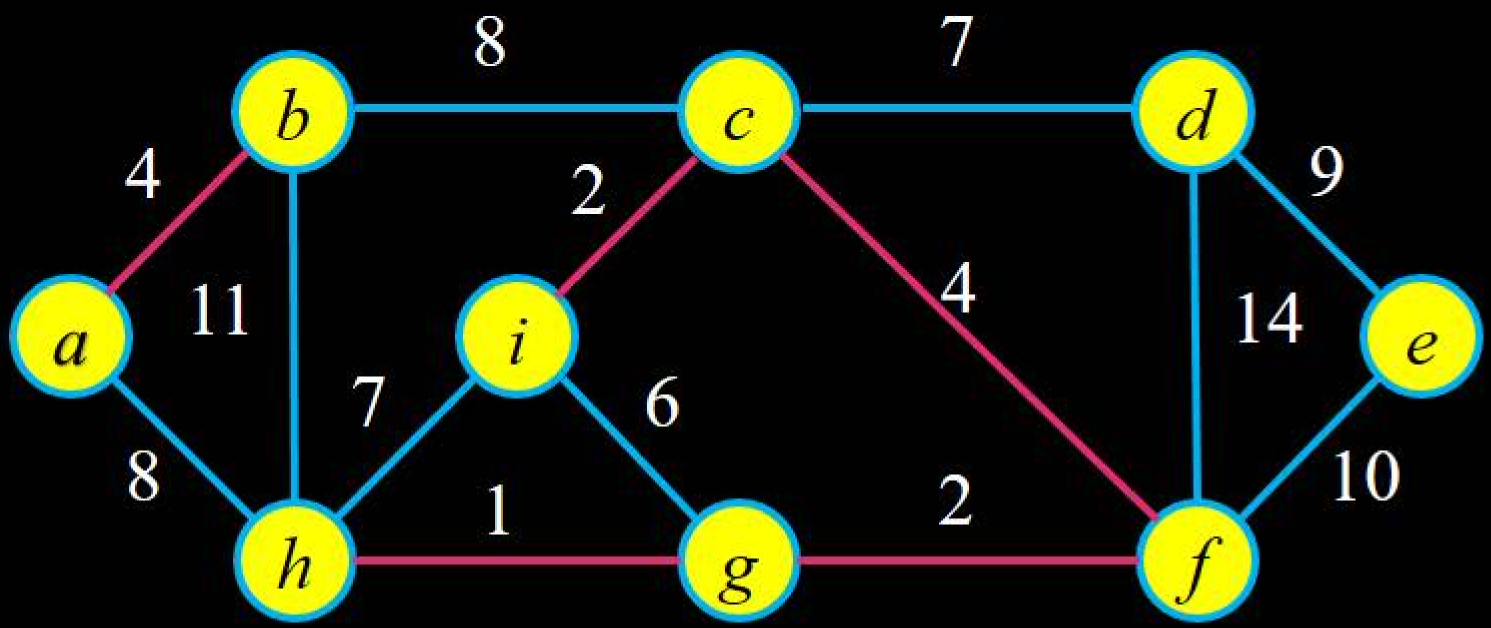
# Example



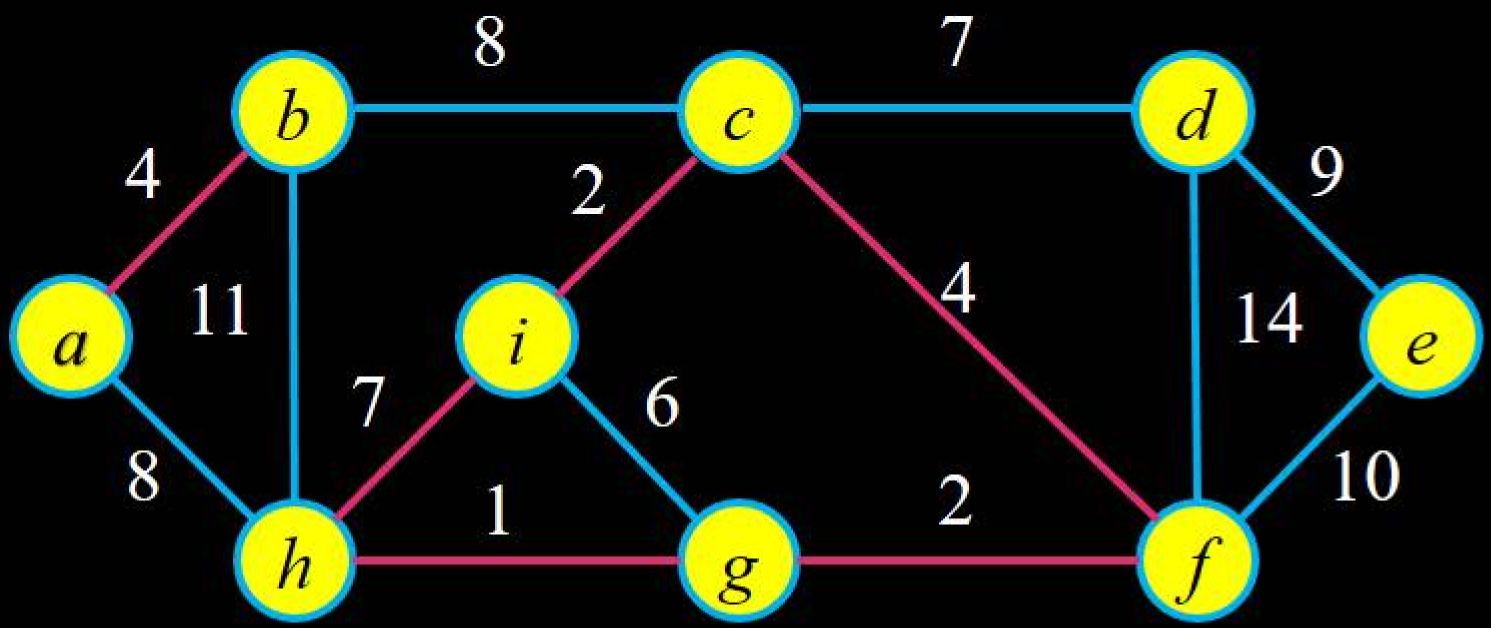
# Example



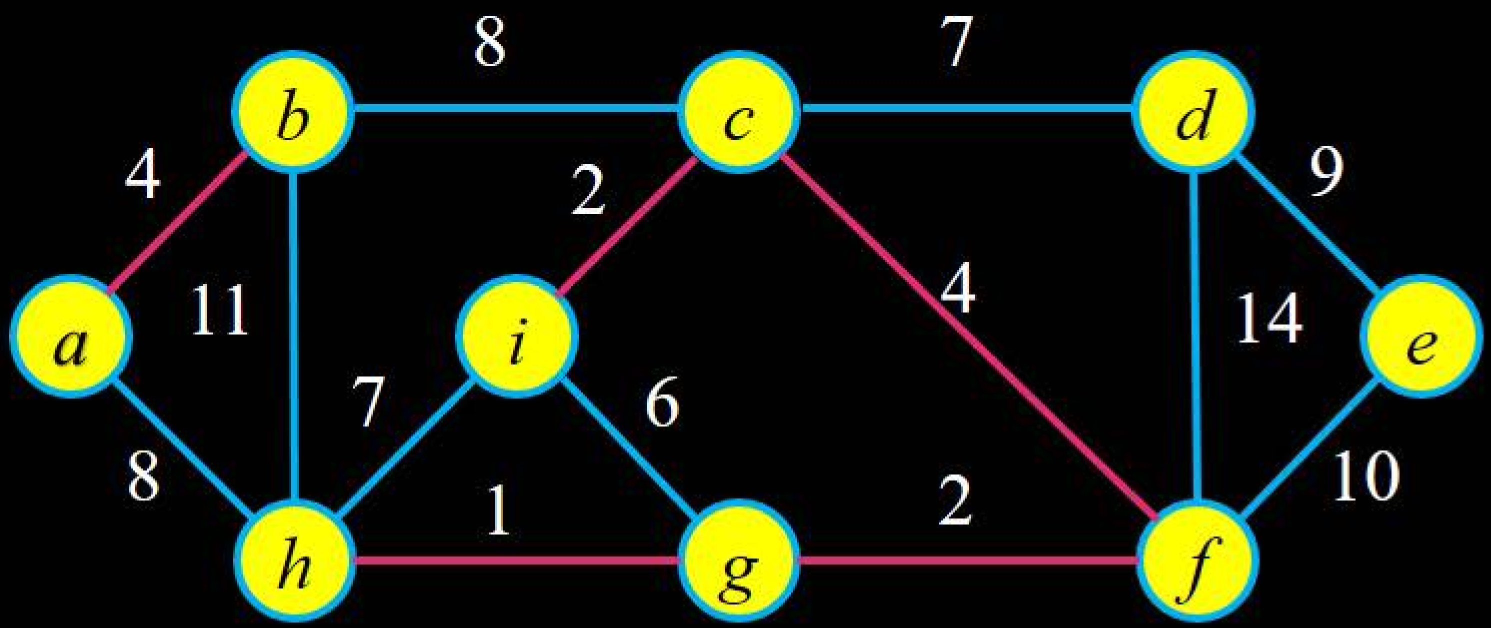
# Example



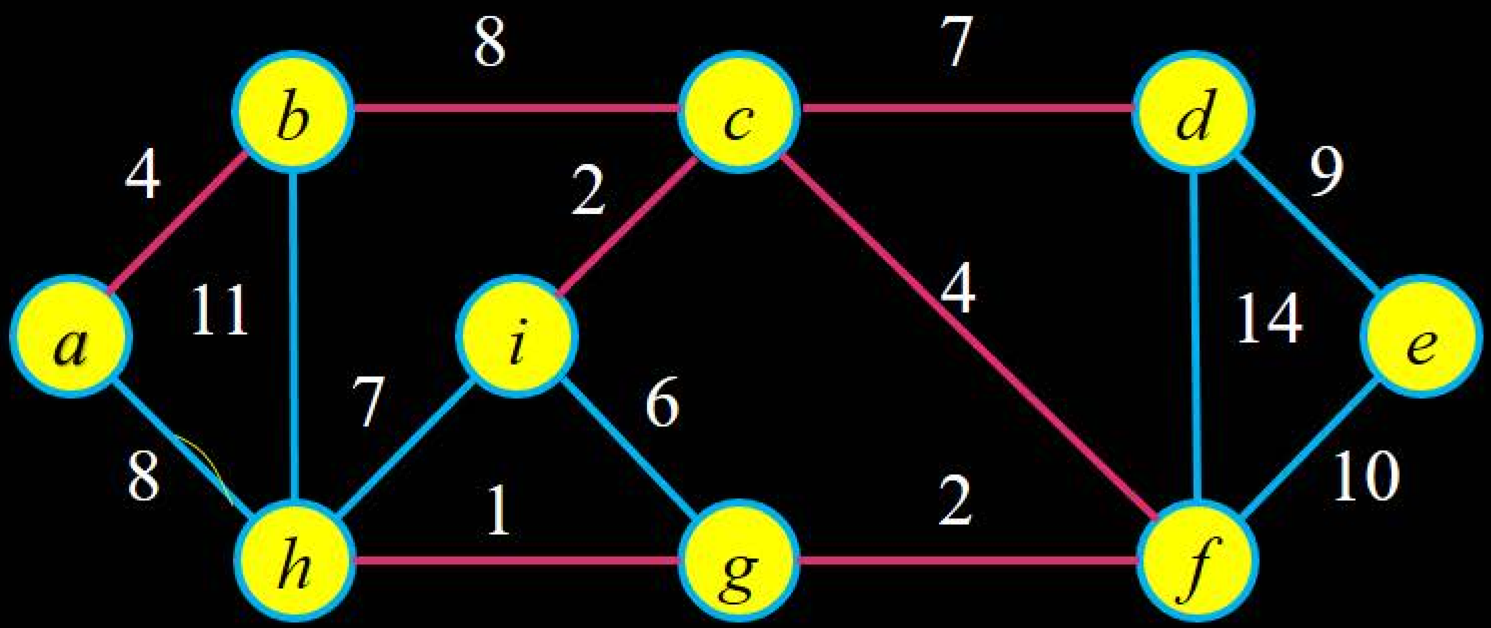
# Example



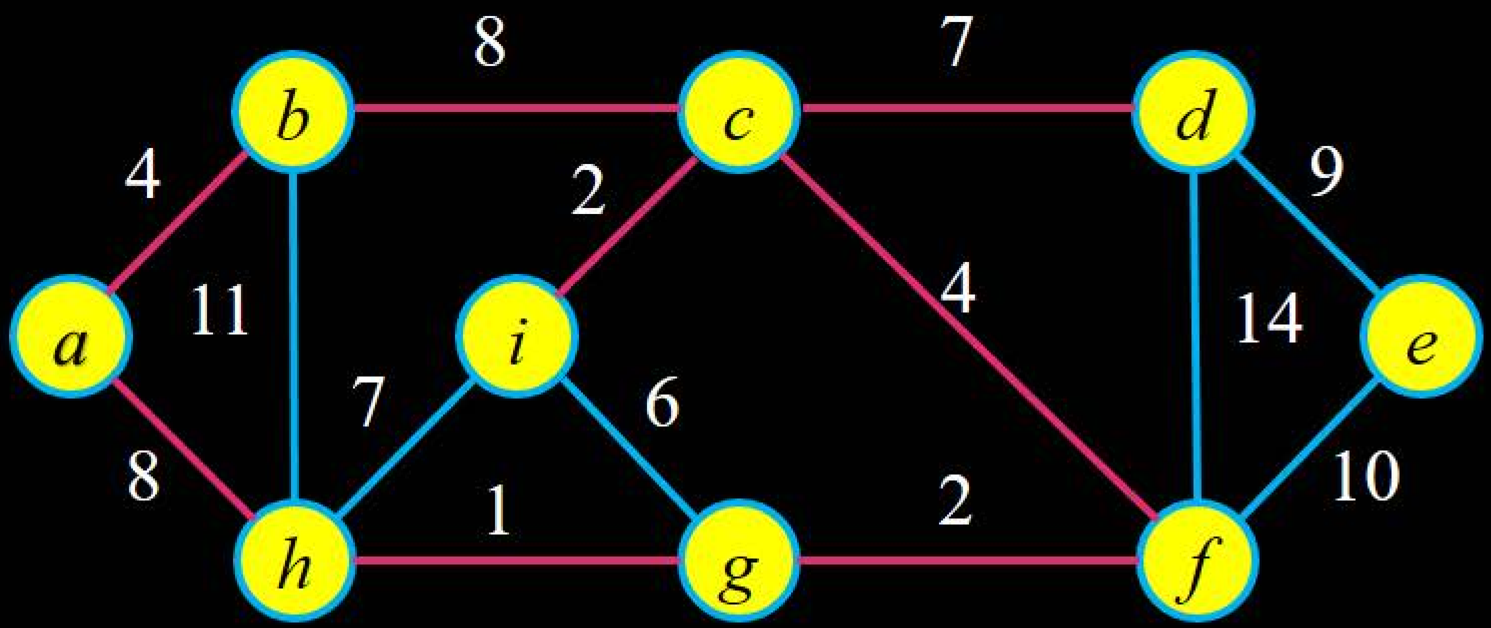
# Example



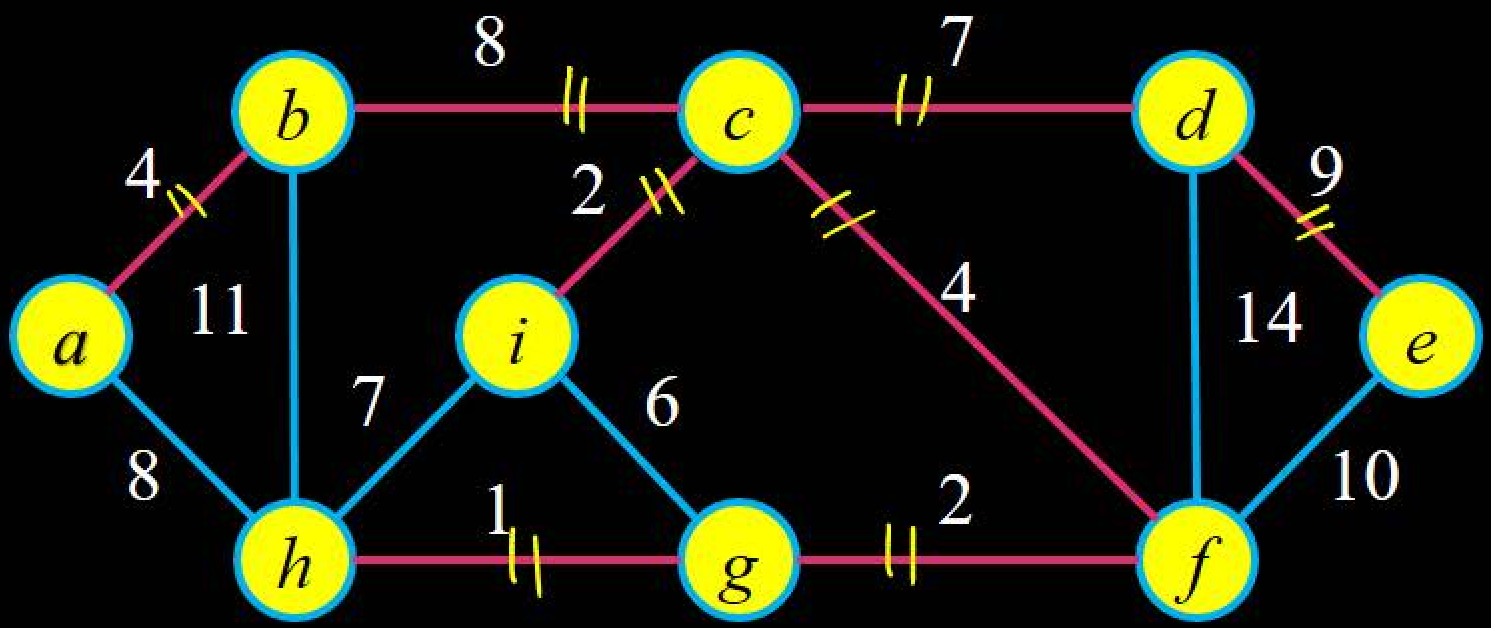
# Example



# Example



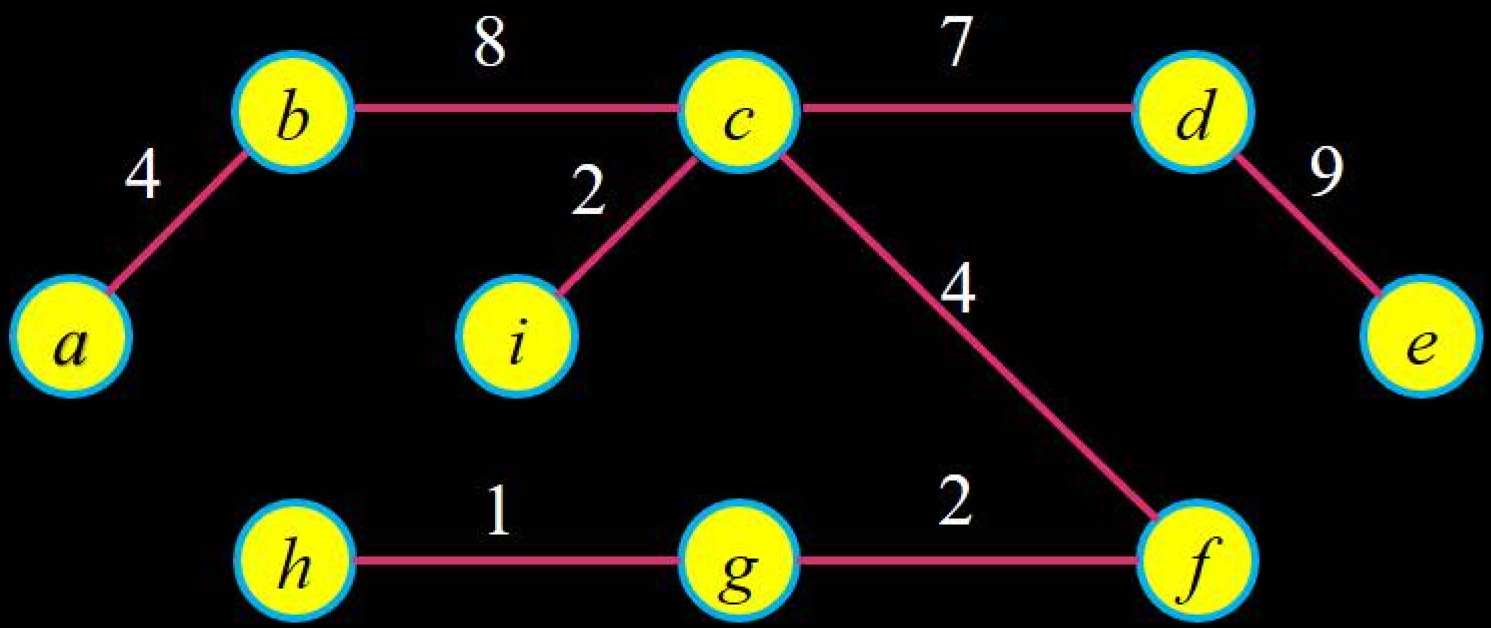
# Example




9
~~10~~ vertices
8 edge

# The MST

Spanning tree
- Prims
- kruskal



# Single Source Shortest path

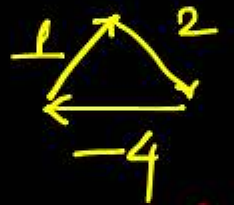
## Dijkstra’s Algorithm

# Single Source Shortest path

- Let G be an directed weighted graph G(V,E). The single source shortest path find shortest path between source vertex to all other vertex.

- The distance between source & other vertex is summation of the ~~path~~ edge weight in the shortest path.

Dijkstra's algorithm does Not work with Negative edge weight cycle.




Negative edge weight cycle

Bellmanford Algorithm
– Single source shortest path
Reports Negative edge weight cycle

# Single Source Shortest path

- Let G = (V, E) be a weighted digraph, with weight function $w: E \rightarrow R$ mapping edges to real-valued weights. If e = $(u, v)$, we write w$(u, v)$ for weight of the edge.

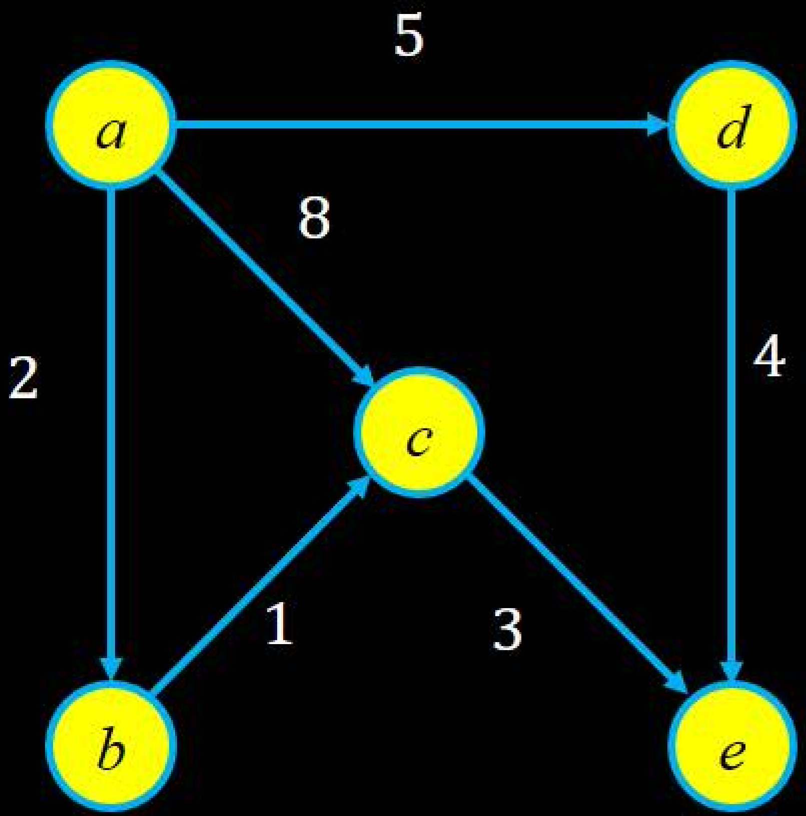
# Single Source Shortest path

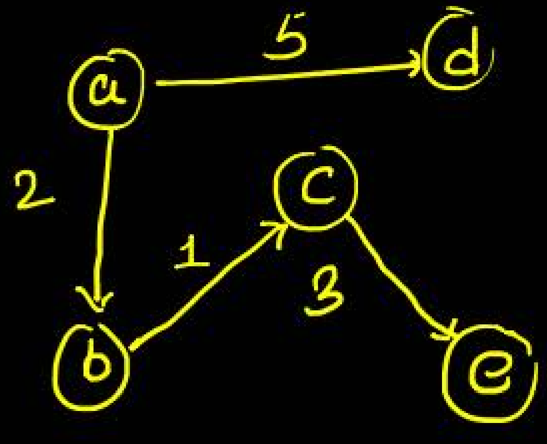
- The length of a path $p = (v_0, v_1, \ldots v_k)$ is the ***sum of the weight*s** of its constituent edges:

# Example

Smaller instance is an aptitude

- Example:
- length({a, b, c, e}) = 6 from a
- distance from a to e is 6




$C.\pi = b$

$B.\pi = a$

a—b + b—c

2 + 1 = $\boxed{3}$

objective:

minimize the cost

distance from source

to destination

(all vertices)

Constrains:

-Availability of direct path from source

# Problem Statement

Objective function:- It is required to compute shortest path from source to all other vertex

minimize distance from source to all other vertex.

# Problem Statement

- **Problem statement:** Given a digraph with non-negative edge weights $G = (V, E)$ and a distinguished source vertex, e. g. V, determine the distance and a shortest path from the source vertex to every vertex in the digraph.

# Problem Statement:

- **Intuition:** Report the vertices in increasing order of their distance from the source vertex. Construct the shortest path tree edge by edge; at each step adding one new edge, corresponding to construction of shortest path to the current new vertex.

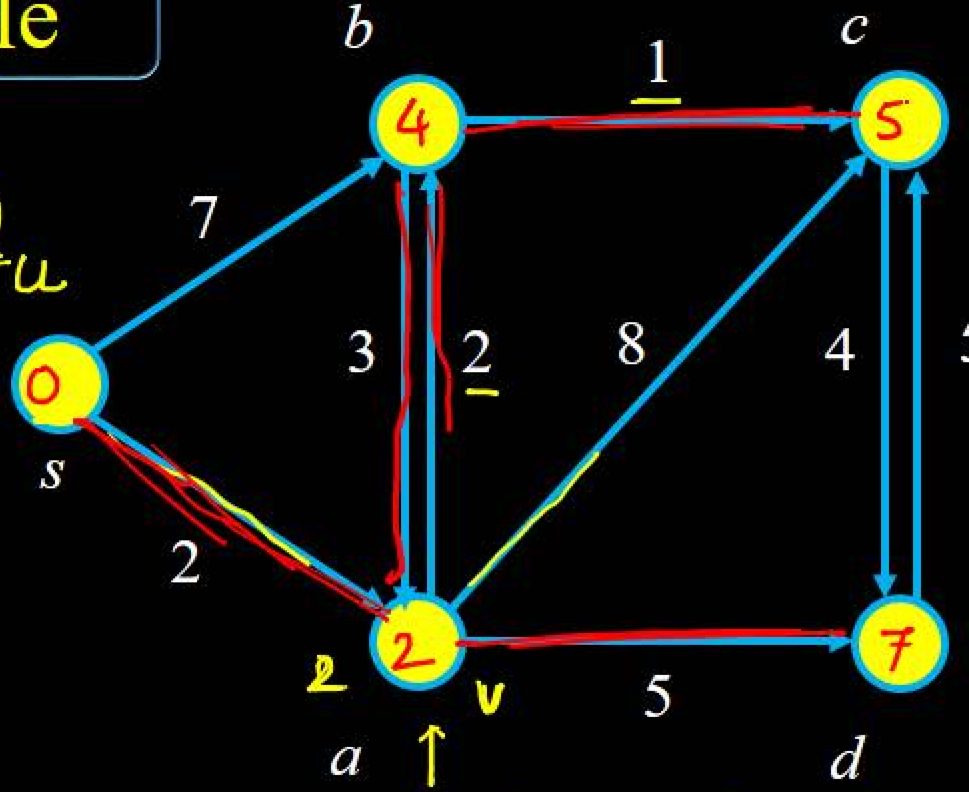
# Example

if $\underset{\infty}{d(v)} > \underset{0}{d[u]} + \underset{2}{w(u,v)}$ $\quad 7 > 2+4$

$d[b] = 7$

$d[b] > \underset{2}{d(a)} + w(a,b)$

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| | 0 | ∞ | ∞ | ∞ | ∞ |
| | — | 2 | 7 | ∞ | ∞ |
| | — | — | 4<7<br>4 | 10<br>2+8 | 7 |
| | — | — | — | 4+1<10<br>5 | 7 |
| | — | — | — | — | 7 |
| | — | — | — | — | |




2 will taken (Extract min)

prims Algo update . B D C

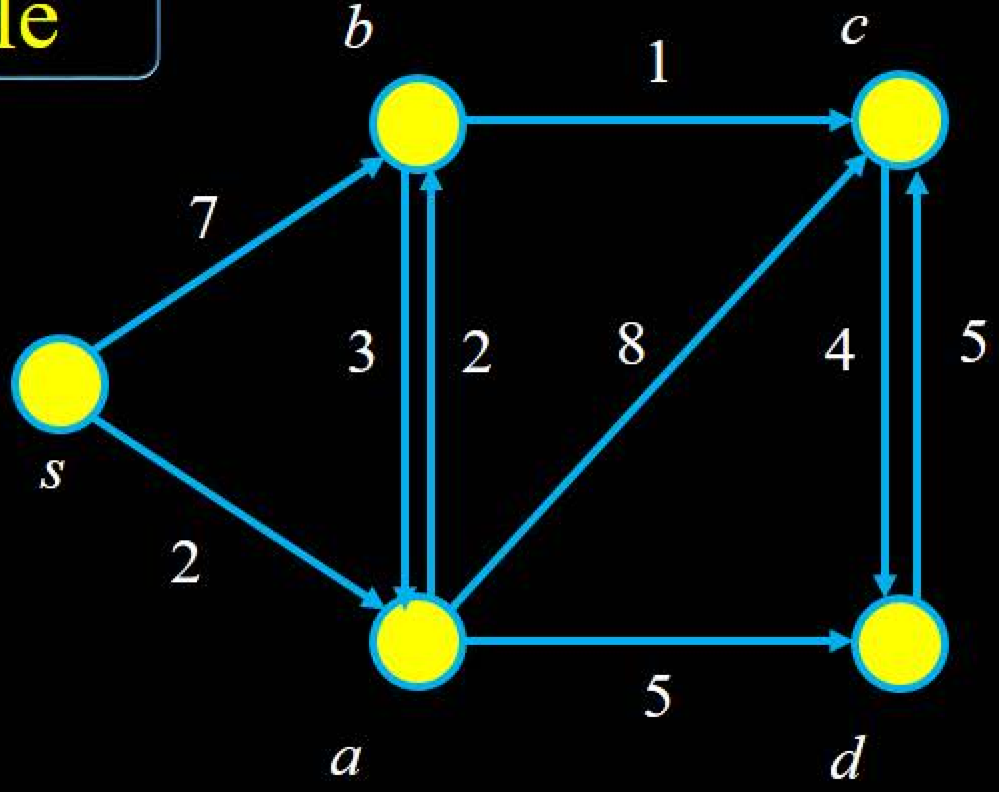
2 5 8

Dijkhas Algo B D C

2+2 2+5 2+8

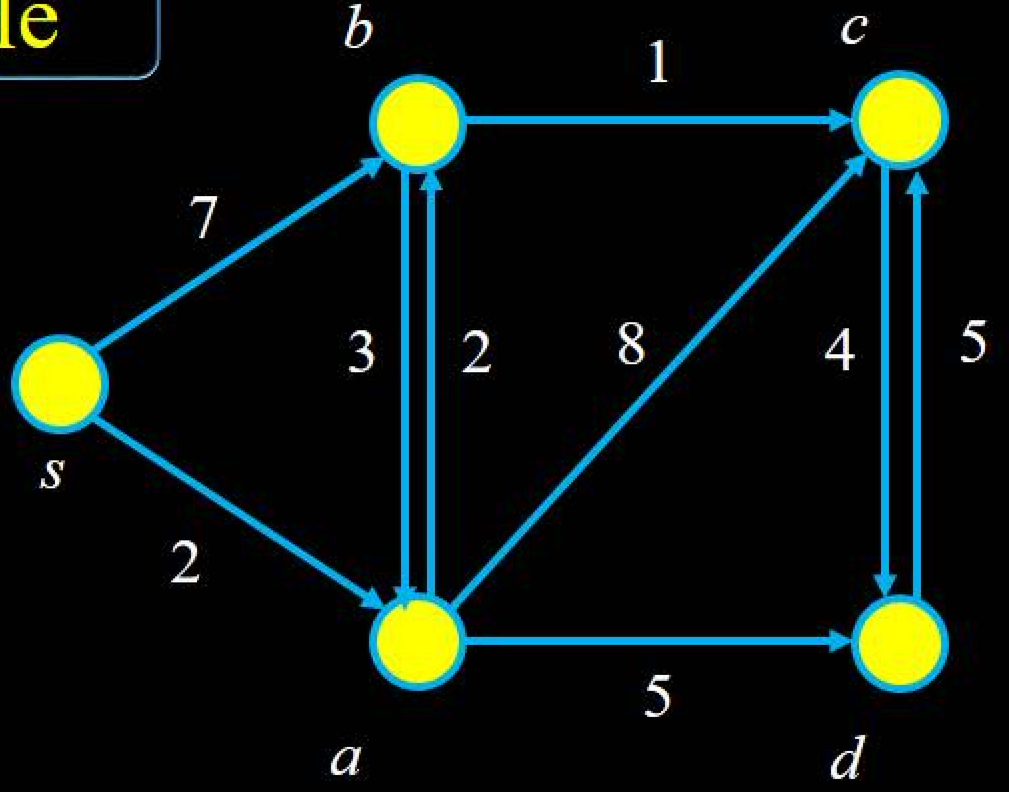
# Example

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | | | | | |
| v.π | | | | | |



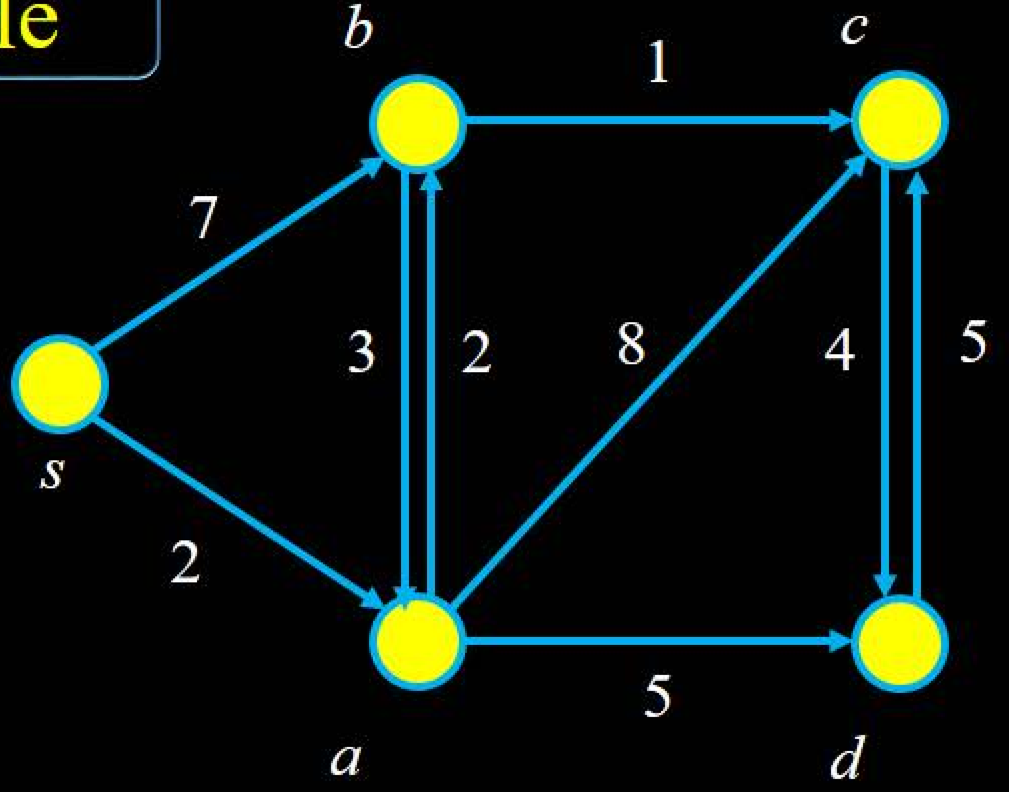
# Example

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | | | | | |
| v.π | | | | | |



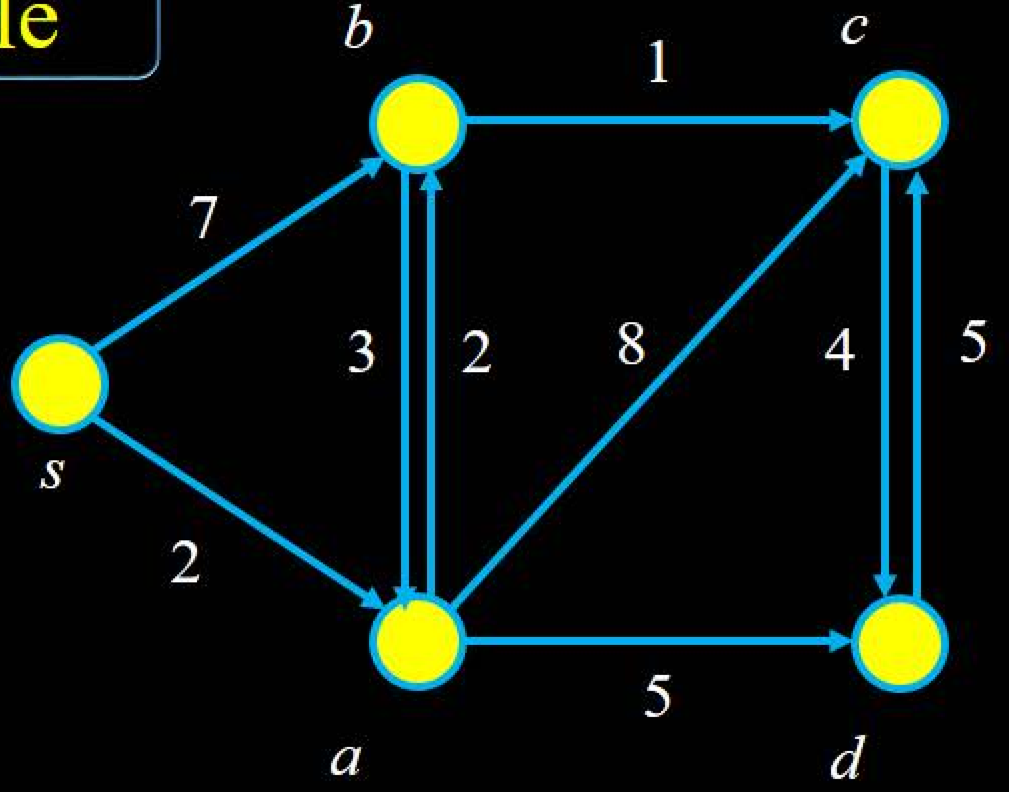
# Example

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | | | | | |
| v.π | | | | | |

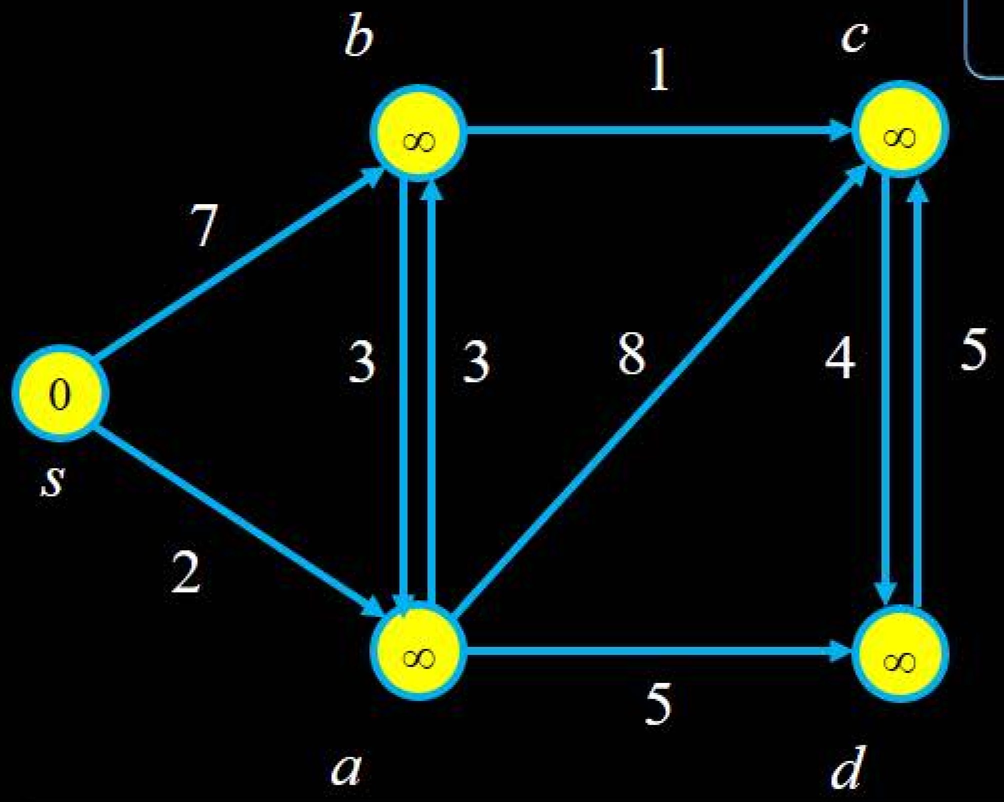


# Example

| V | s | a | b | c | d |
| --- | --- | --- | --- | --- | --- |
| d[v] | | | | | |
| v.π | | | | | |

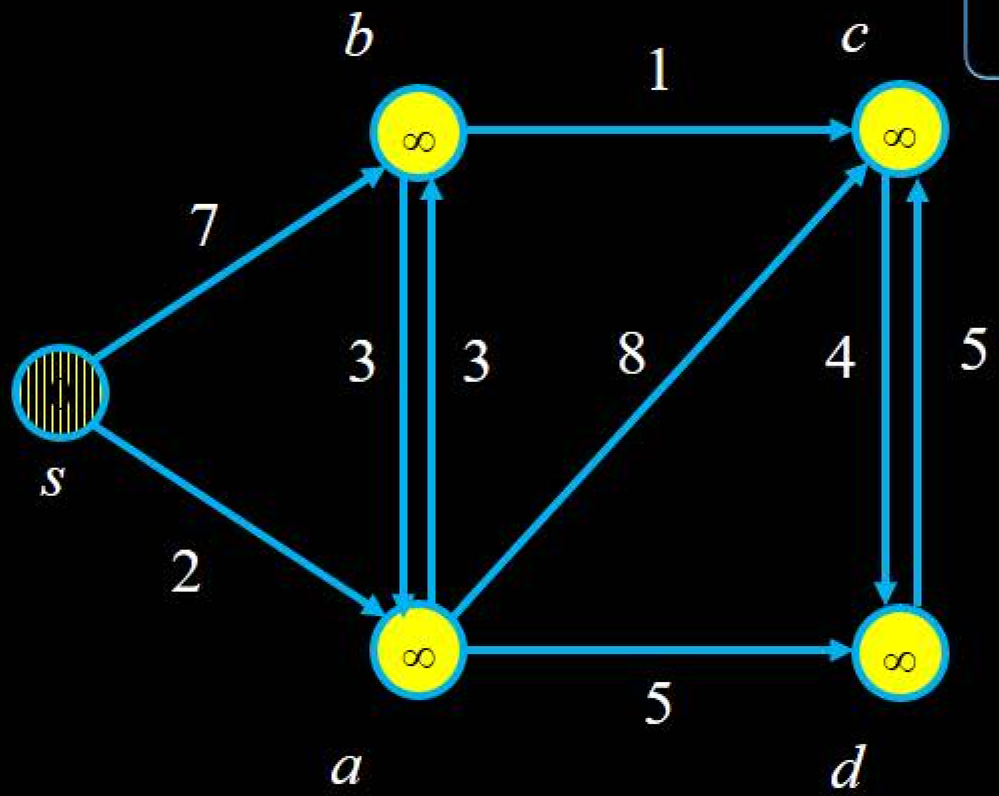


# Example



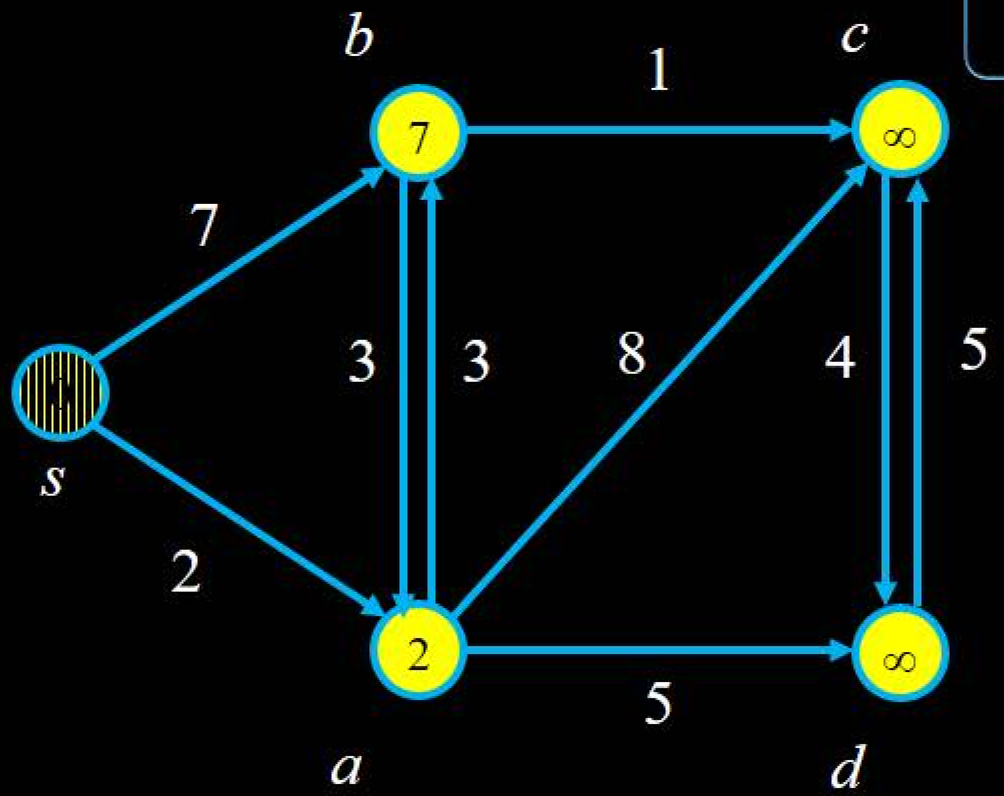

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | 0 | ∞ | ∞ | ∞ | ∞ |
| v.π | nil | nil | nil | nil | nil |

# Example



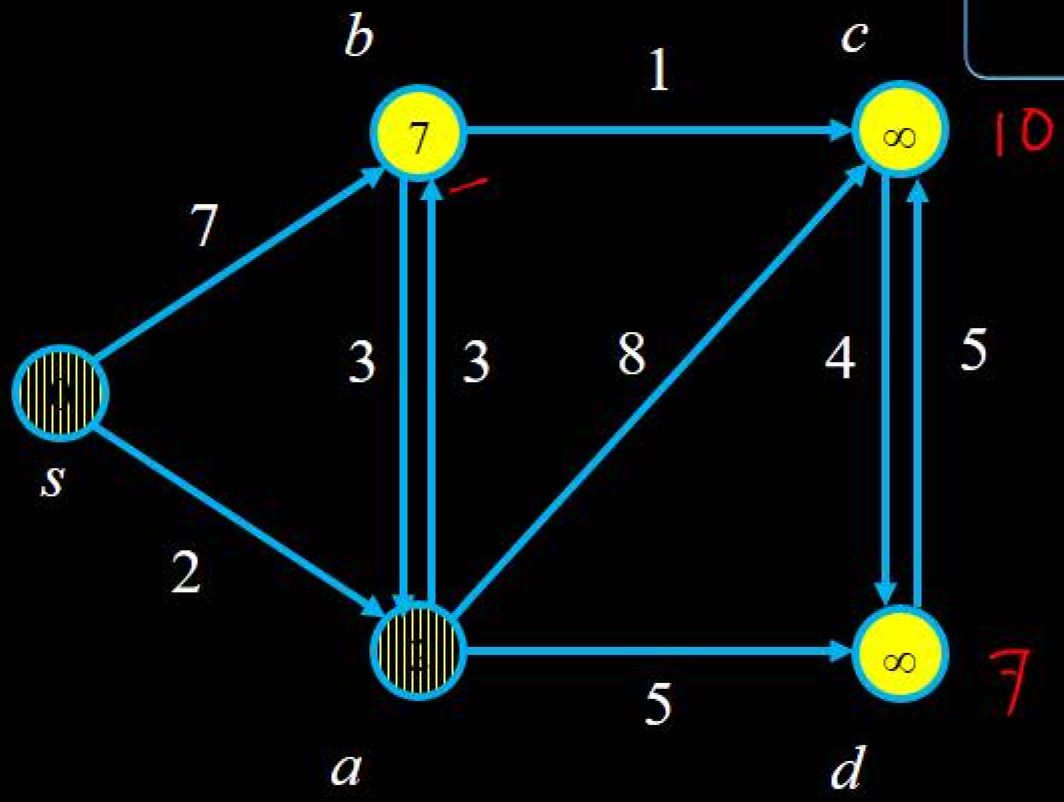

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | 2 | 7 | ∞ | ∞ |
| v.π | nil | s | s | nil | nil |

# Example



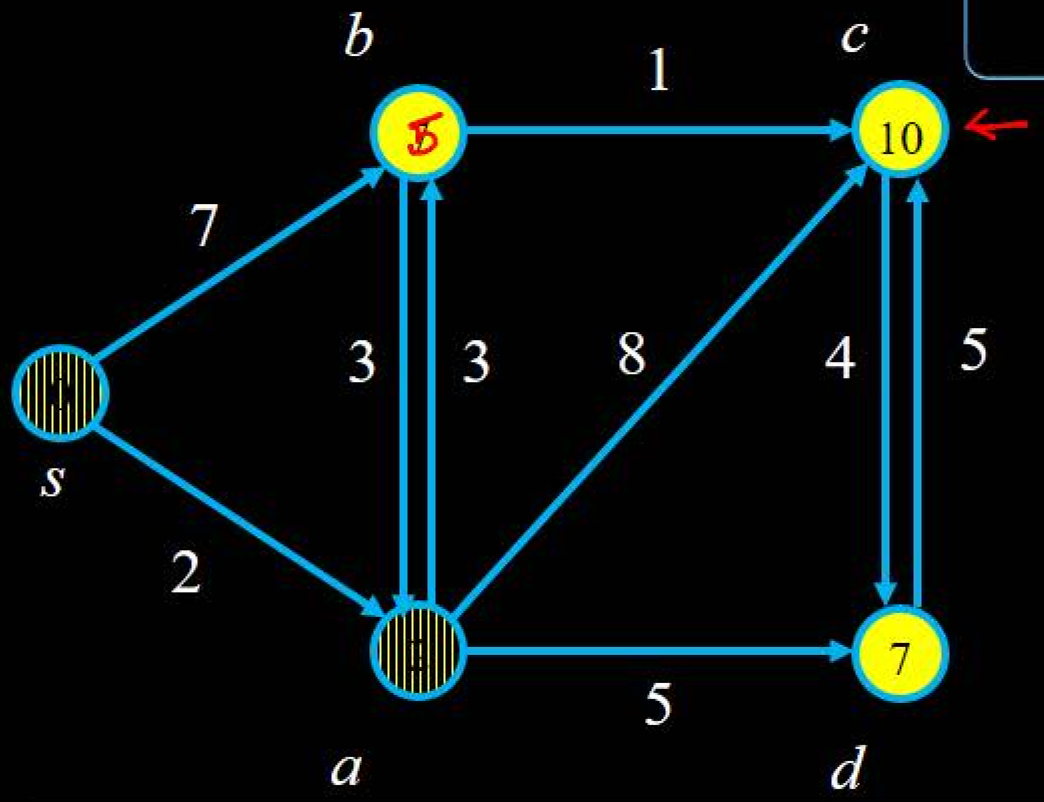

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | 2 | 7 | ∞ | ∞ |
| v.π | nil | s | s | nil | nil |

# Example



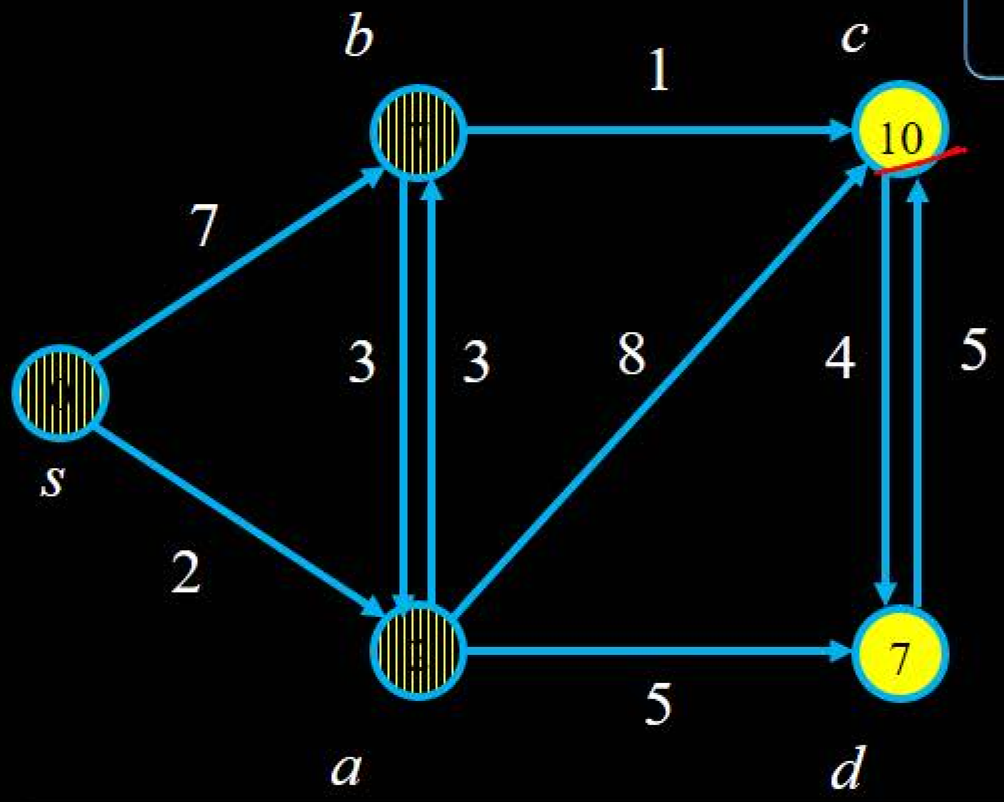

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | - | 7 | ∞ | ∞ |
| v.π | nil | s | s | nil | nil |

# Example




| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | - | 5 | 10 | 7 |
| v.π | nil | s | a | a | a |

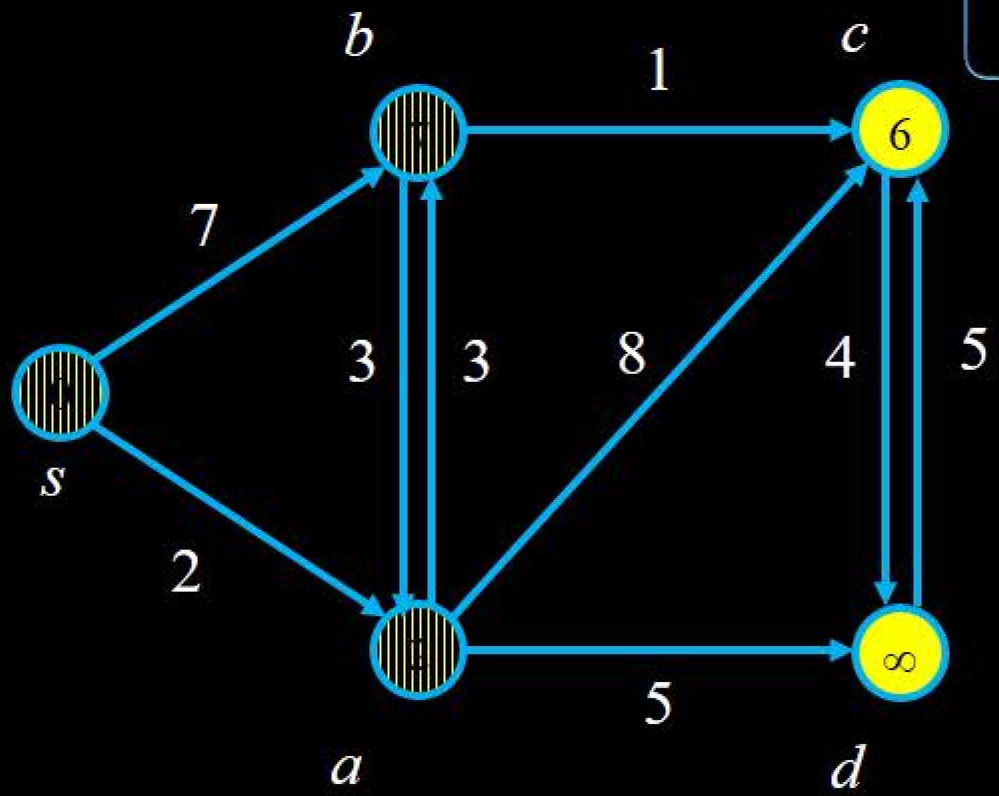
# Example




~~10<~~ 6<10

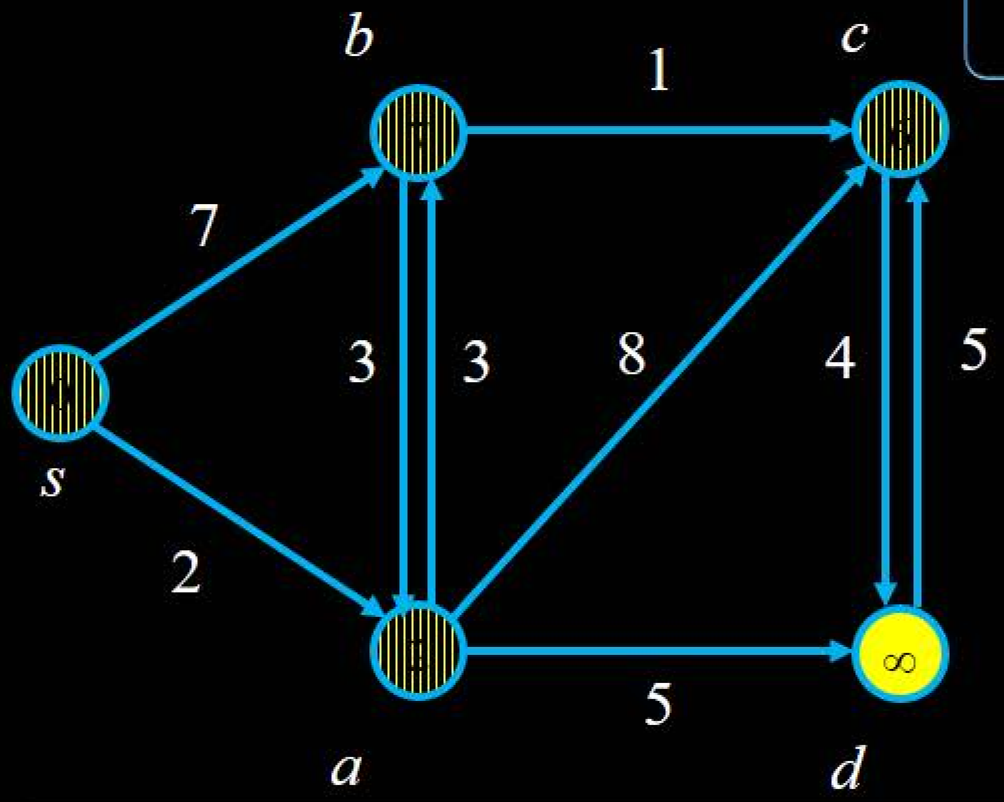
| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | - | - | 10 | 7 |
| v.π | nil | s | a | a | a |

# Example



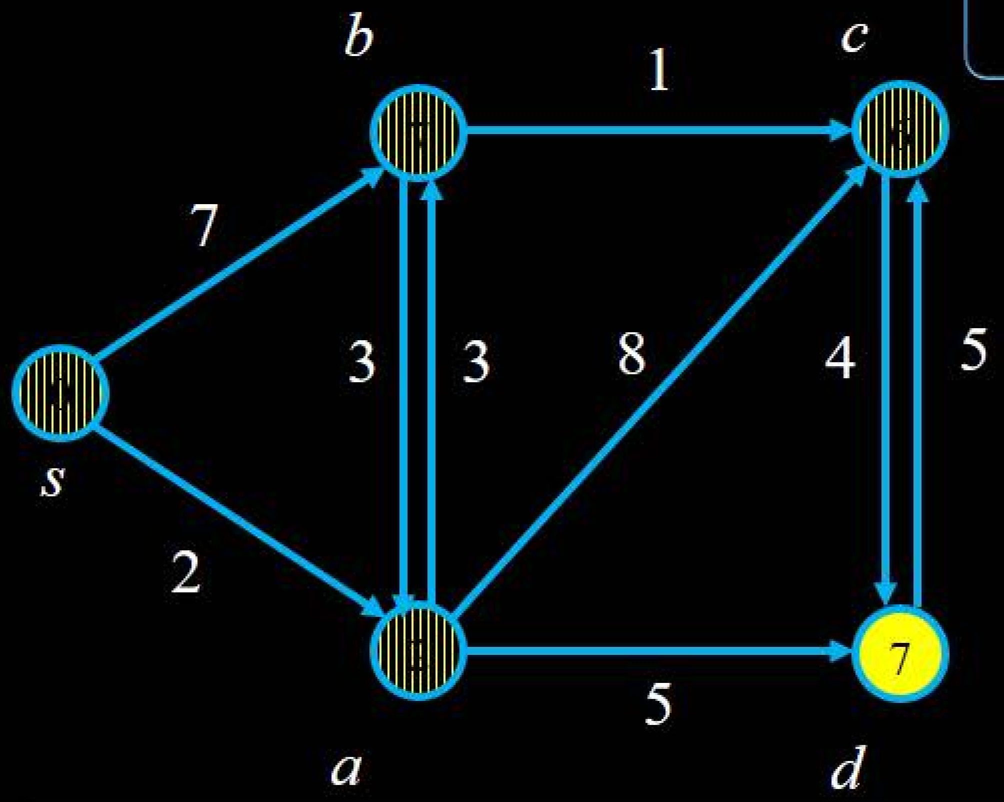

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | - | - | 6 | 7 |
| v.π | nil | s | a | b | a |

## Example



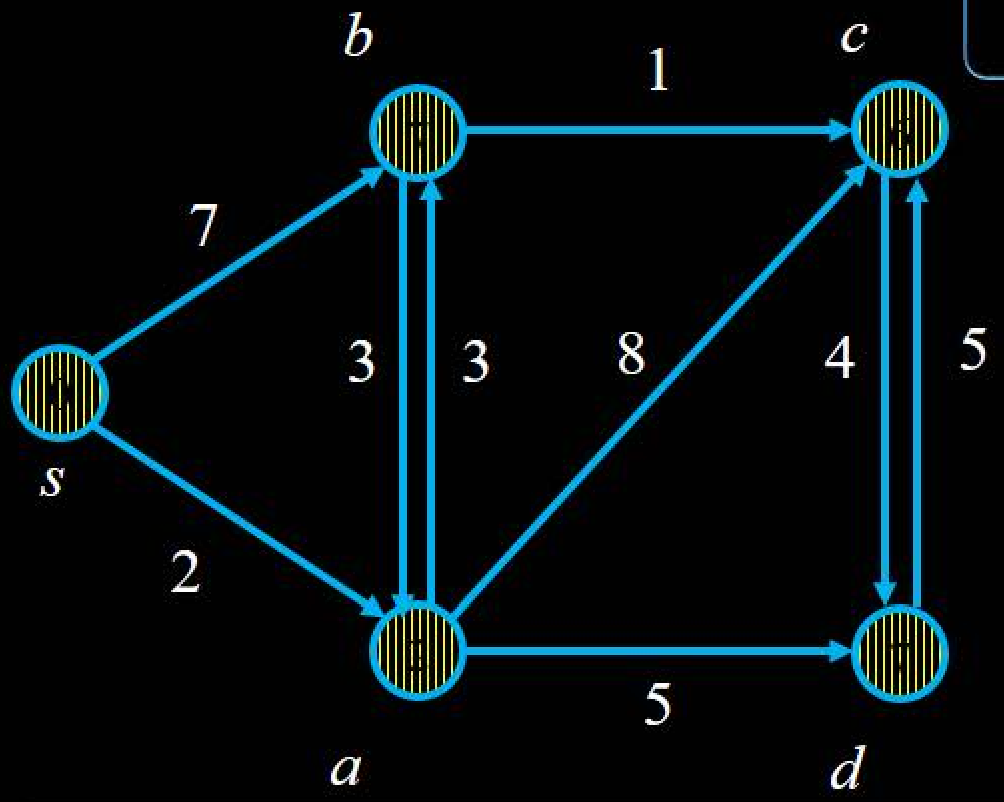

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | - | - | 6 | 7 |
| v.π | nil | s | a | b | a |

# Example



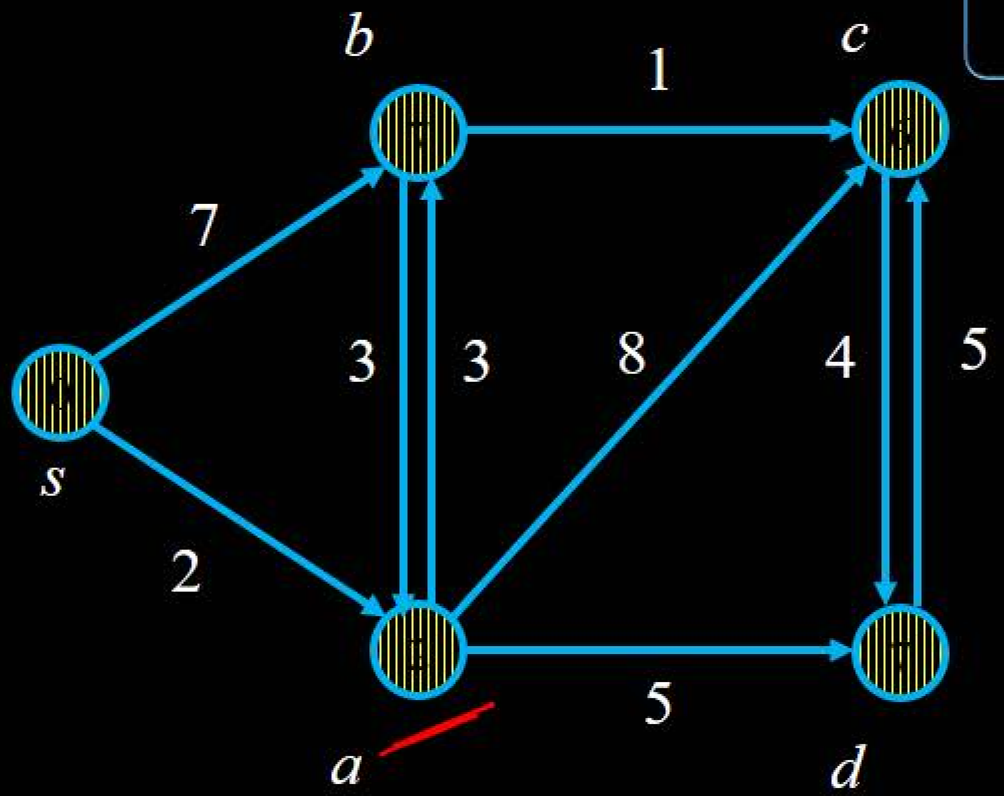

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | - | - | - | 7 |
| v.π | nil | s | a | b | a |

# Example



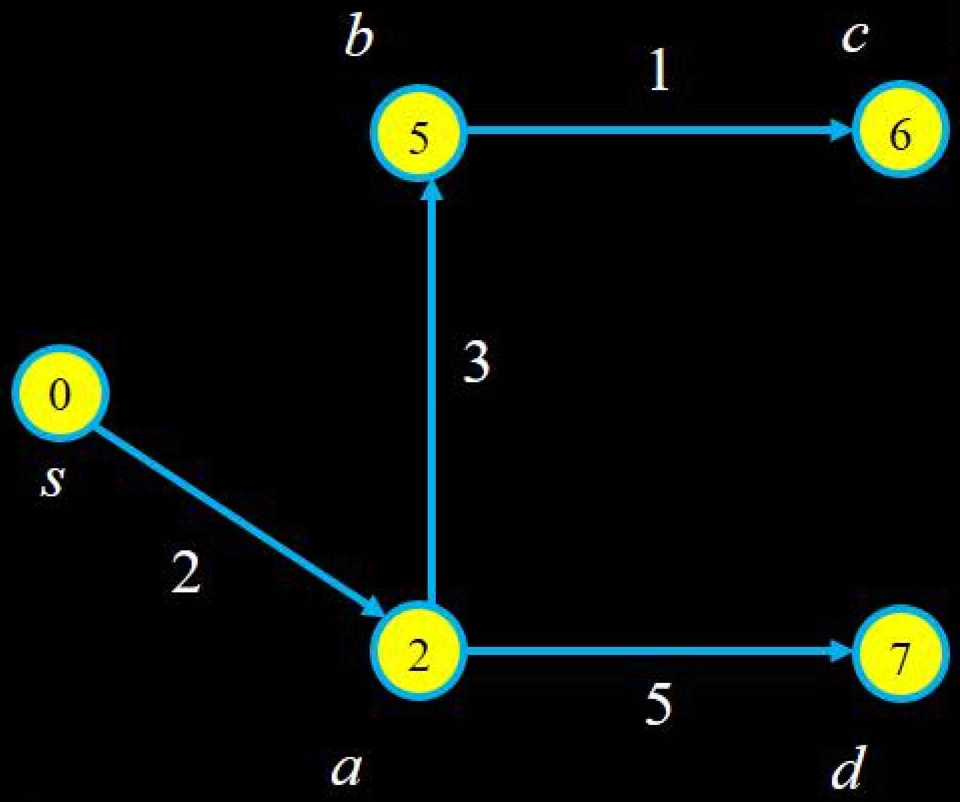

| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | - | - | - | 7 |
| v.π | nil | s | a | b | a |

# Example




| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | - | - | - | - |
| v.π | nil | s | a | b | a |

# Example




| V | s | a | b | c | d |
|---|---|---|---|---|---|
| d[v] | - | - | - | - | - |
| v.π | nil | s | a | b | a |

# Relaxation

```
Relax(u, v){
    if (d[u] + w(u, v)  < d[v]){
        d[v] = d[u] + w(u, v);
        v.π  = u
    }
}
```

# Initialization

```
Init(G,w,S)
{
      for (each u ∈ V)
      {
           d[u] = ∞;
           u.π = NIL;
      }
```

# Algorithm

```
Dijkstra(G,w,s){
Init(G,w,S)
d[s] = 0;
s.π = NIL;
Q = (queue with all vertices);
     while (Non-Empty(Q)){
          u = Extract-Min(Q);
          for (each v ∈ Adj [u])
            if (d[u] + w(u, v) < d[v]){
             d[v] = d[u] + w(u, v);
             Decrease-Key (Q, v, d[v]);
             v.π = u;
            }
      }
}
```

# Initialization

```
Init(G,w,S)
{
      for (each u ∈ V)
      {
            d[u] = ∞;
            u.π = NIL;
      }
```

- Initialization takes $O(|V|)$ time

# Algorithm

```
Dijkstra(G,w,s){
Init(G,w,S)
d[s] = 0;
s.π = NIL;
Q = (queue with all vertices);
      while (Non-Empty(Q))    {
            u = Extract-Min(Q);
            for (each v ∈ Adj [u])
              if (d[u] + w(u, v) < d[v]){
               d[v] = d[u] + w(u, v);
               Decrease-Key (Q, v, d[v]);
               v.π = u;
              }
       }
}
```

Decrease key

- Initialization takes $O(|V|)$ time
- Extract min will take O(log|$V$|) times
- The loop will run for adjacency list of every vertex, $\sum(\deg(u)) = 2E$
- Decrease key operation will take O(log|$V$|)
- $\sum_{u\ in\ V}[O(\log|V|) + O(\deg(u)\log|V|)]$
- $= \log|V| \sum_{u\ in\ V}[1 + O(\deg(u))]$
- $= \log|V| O(|V| + 2|E|)$
- $= O((|V| + |E|)\log|V|)$

# GATE 2003

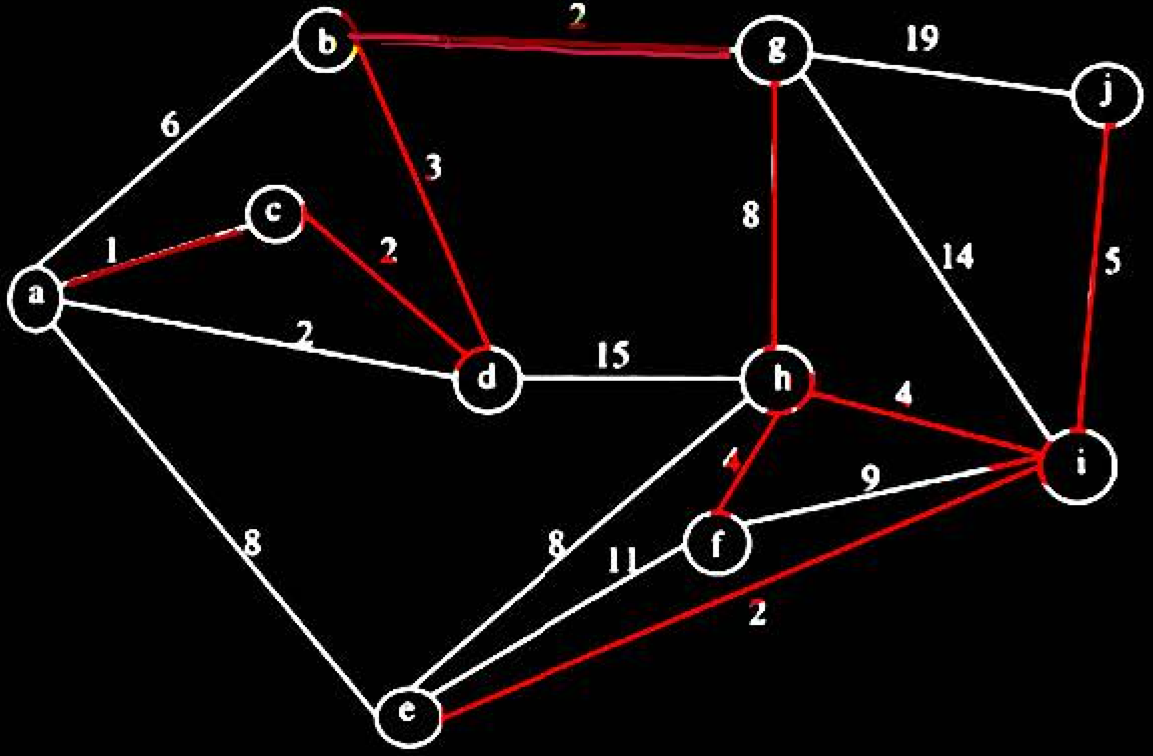
Q. What is the weight of a minimum spanning tree of the following graph?

(a)29

(b) 31

(c) 38

(d) 41




kruskal

7am

31 prims/kruskal

How many spanning tree possible?

# GATE 2014

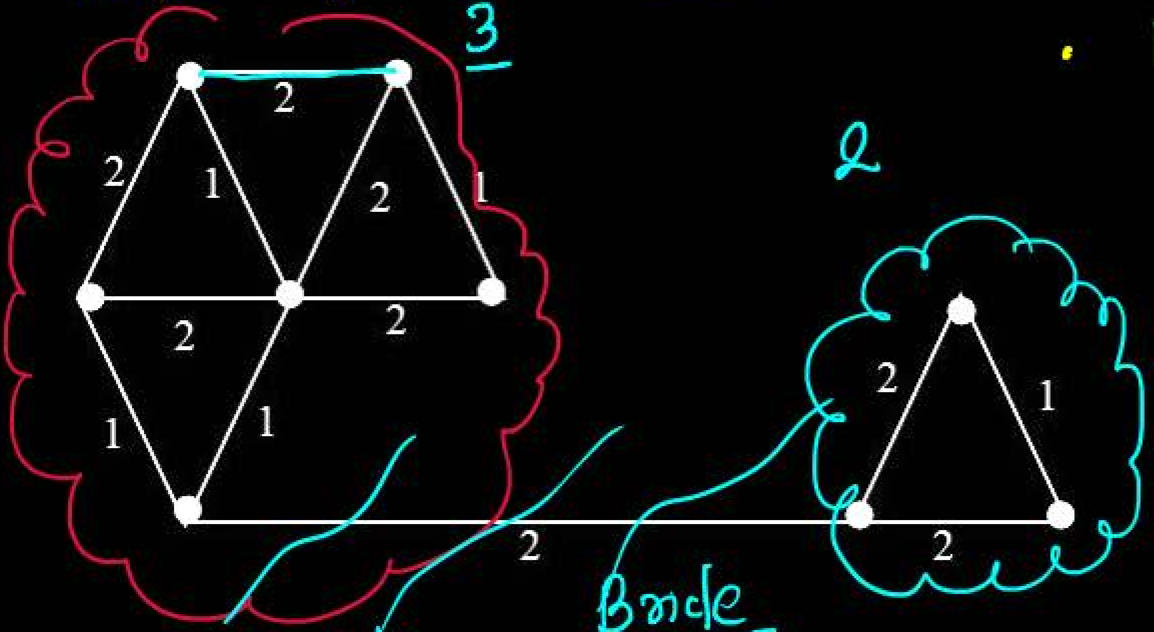
(08) text book

Q. The number of distinct minimum spanning trees for the weighted graph below is

Edge weight are Not distinct

· Prims & kruskal

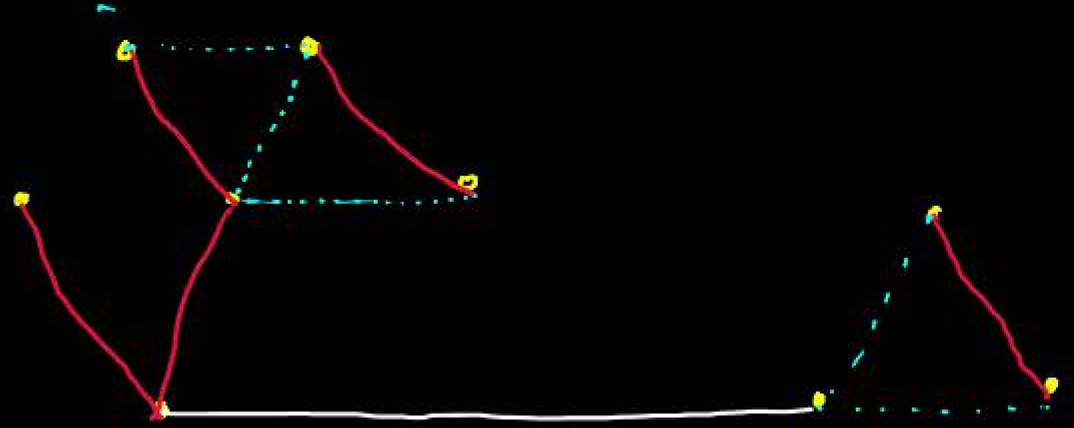
· Extention of prims Algorithm
Single Source shortest path




3 possible Spanning trees

Bridge
⇑
cut edge

Graph Theory



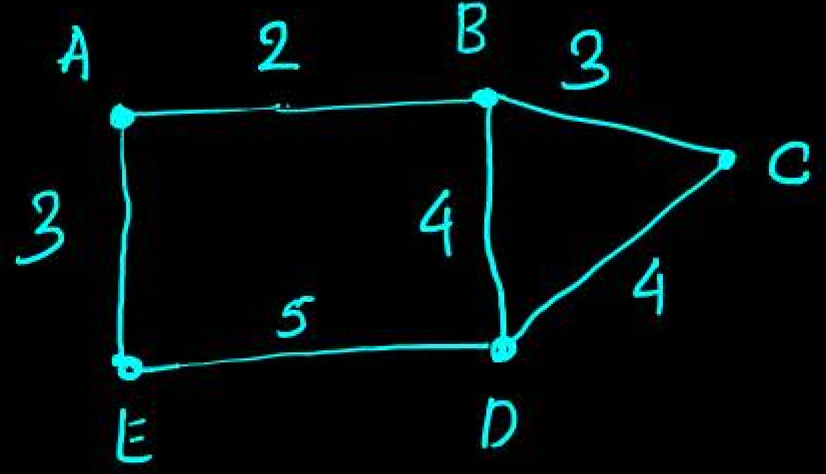

Question No. 1 Text book

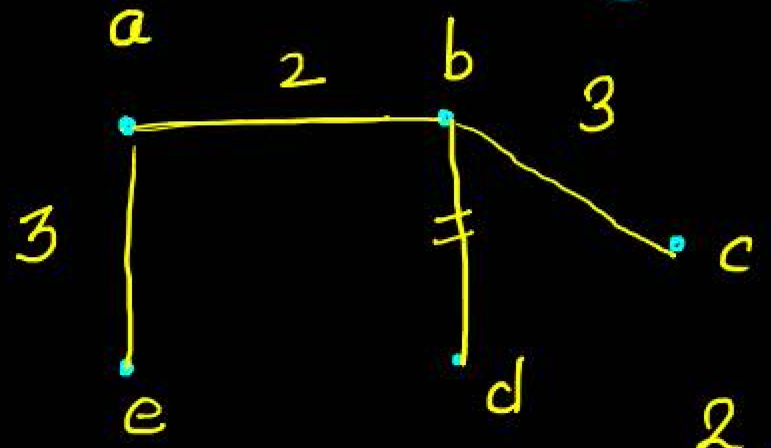
Complete - complete grap

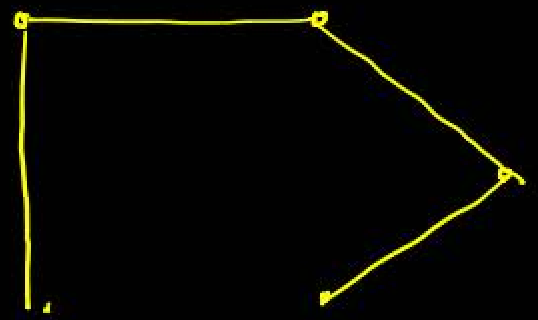
$\boxed{n^{n-2}}$

How many Spanning tree exists?

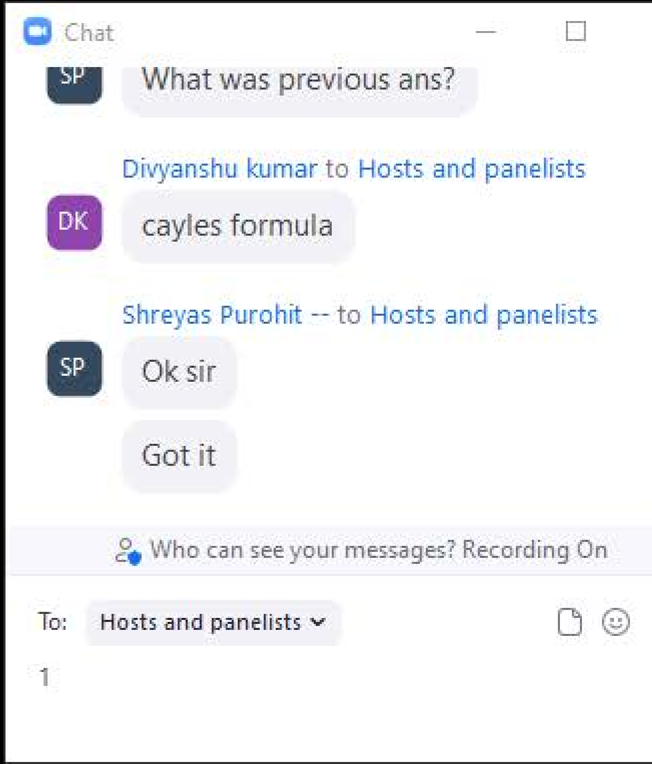
cost



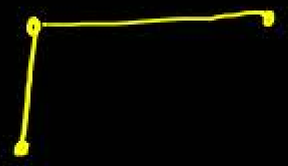



2 Spanning



## Krukal Method

- Edge weights are distinct.
    - edge with minimum weight will be part of spanning tree.
    - ~~Edged~~ Edge with second mininum weight part of it.
    - Edge with 3rd minimum weight will be part MST



Its required 3 edges to form a cycle.

Not part of MST — forming a cycle

Not forming a cycle — MST

# GATE 2015 SET – III

Q. Let G be a connected undirected graph of 100 vertices and 300 edges. The weight of a minimum spanning tree of G is 500. When the weight of each edge of G is increased by five, the weight of a minimum spanning tree becomes $\boxed{995}$.

100 vertices, 300 edges

Spanning tree

A subgraph G'(V, E)

100 vertices

minimum spanning tree – 500

The No. of edges in spanning tree with n vertices – n-1

100 vertices – 100-1 = 99 edges

500+99*5 = 500+495 = 995

## GATE 2015 SET - II

Q. The graph shown below has 8 edges with distinct integer edge weights. The minimum spanning tree (MST) is of weight 36 and contains the edges: {(A, C), (B, C), (B, E), (E, F), (D, F)} . The edge weights of only those edges which are in the MST are given in the figure shown below. The minimum possible sum of weights of all 8 edges of this graph is_______.

# GATE 2015 SET - III

Test ~~ur~~ your

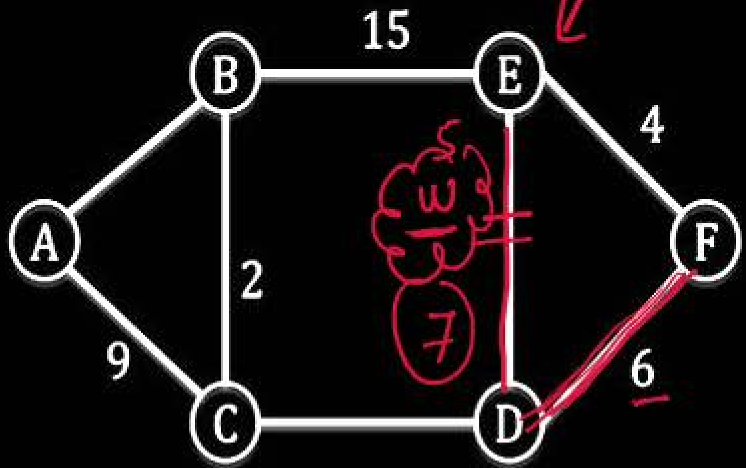
Q. The graph shown below has 8 edges with distinct integer edge weights. The minimum spanning tree (MST) is of weight 36 and contains the edges: {(A, C), (B, C), (B, E), (E, F), (D, F)} . The edge weights of only those edges which are in the MST are given in the figure shown below. The minimum possible sum of weights of all 8 edges of this graph is________.

36 total weight of MST

2 and w(c,d)

(3)

15 , 4 , 6




36
10
16
17
69 Answer

3 - 15

# GATE 2016 SET - I

Q. Let G be a complete undirected graph on 4 vertices, having 6 edges with weights being 1, 2, 3, 4, 5, and 6. The maximum possible weight that a maximum minimum weight spanning tree of G can have is

1 minimum – 1

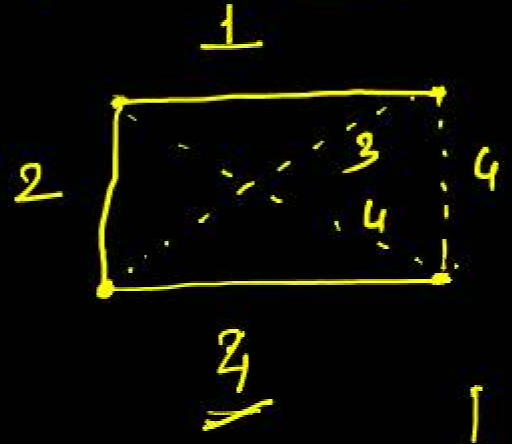
2nd minimum - 2

3rd minimum.

may or not

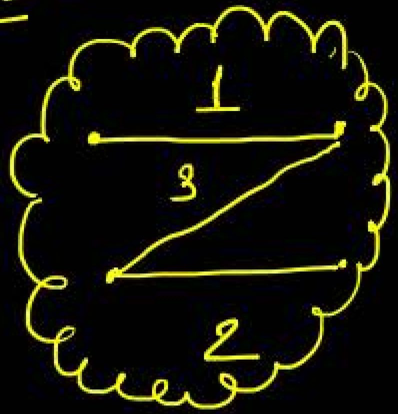
7

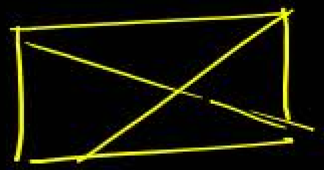
Complete graph




4+ +1 = 7

1+2+3 =

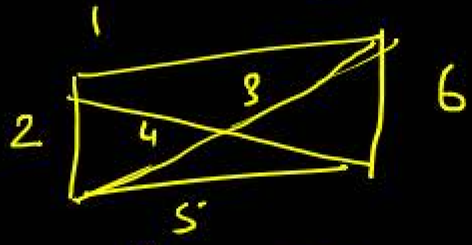


15

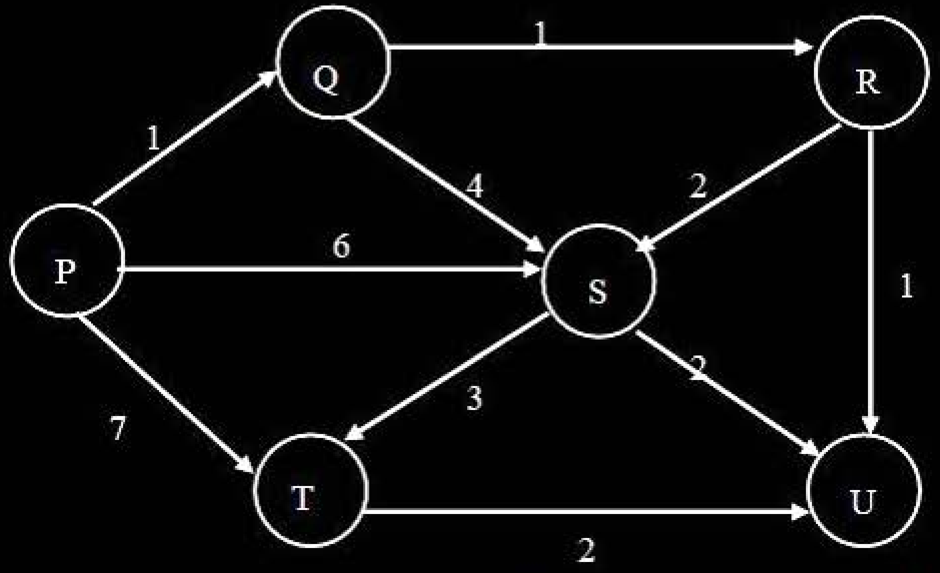
6

1+2+3 = $\boxed{16}$




# GATE 2004

Q. Suppose we run Dijkstra's single source shortest-path algorithm on the following edge-weighted directed graph with vertex P as the source.




In what order do the nodes get included into the set of vertices for which the shortest path distances are finalized?

(a) P,Q,R,S, T, U

(b) P,Q,R,U,S,T

(c) P,Q,R,U,T,S

(d) P,Q, T,R,U,S

## GATE 2012

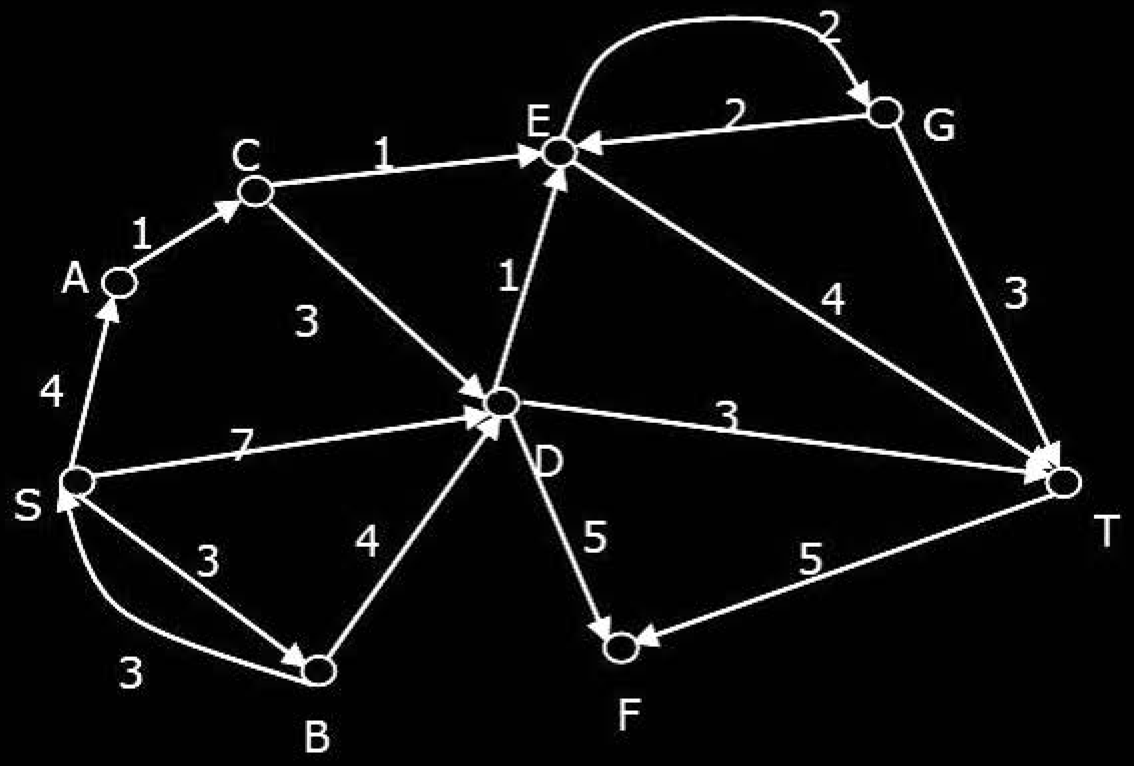
Q. Consider the directed graph shown in the figure below. There are multiple shortest paths between vertices S and T. Which one will be reported by Dijkstra's shortest path algorithm? Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is

(a) SDT

(b) SBDT

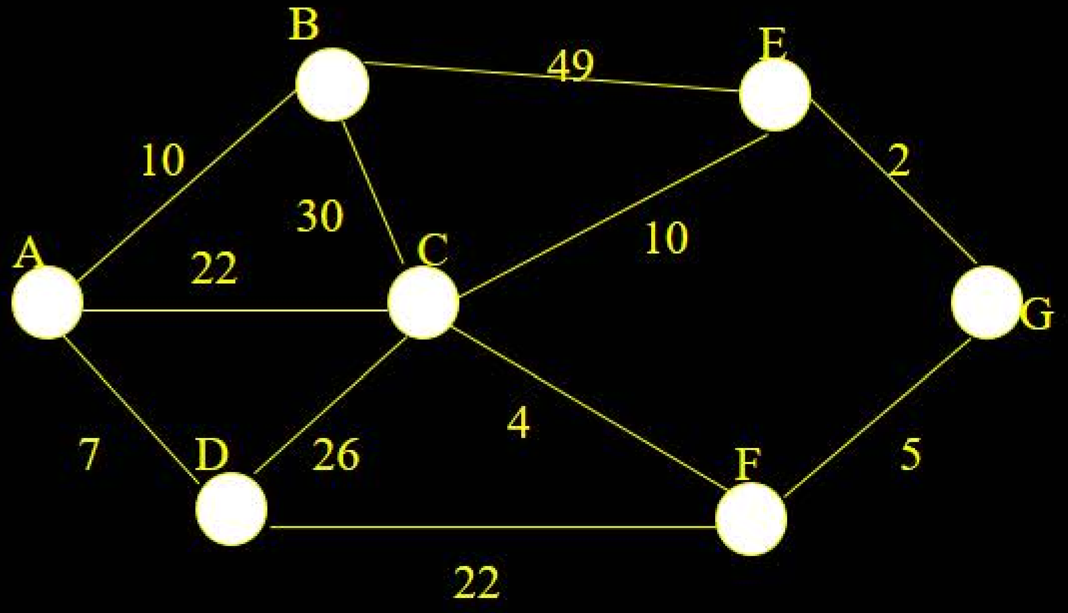
(c) SACDT

(d) SACET



# GATE 2004 (IT)

Consider the undirected graph below:




(A) (E, G), (C, F), (F, G), (A, D), (A, B), (A, C)

(B) (A, D), (A, B), (A, C), (C, F), (G, E), (F, G)

(C) (A, B), (A, D), (D, F), (F, G), (G, E), (F, C)

(D) (A, D), (A, B), (D, F), (F, C), (F, G), (G, E)

# GATE 2004 (IT)

Using Prim's algorithm to construct a minimum spanning tree starting with node A, which one of the following sequences of edges represents a possible order in which the edges would be added to construct the minimum spanning tree?

(A) (E, G), (C, F), (F, G), (A, D), (A, B), (A, C)

(B) (A, D), (A, B), (A, C), (C, F), (G, E), (F, G)

(C) (A, B), (A, D), (D, F), (F, G), (G, E), (F, C)

(D) (A, D), (A, B), (D, F), (F, C), (F, G), (G, E)

**GATE 2016 SET-I**

$G = (V,E)$ is an undirected simple graph in which each edge has a distinct weight, and $e$ is a particular edge of $G$. Which of the following statements about the minimum spanning trees (MSTs) of $G$ is/are TRUE?

I. If $e$ is the lightest edge of some cycle in G, then every MST of $G$ includes $e$

II. If $e$ is the heaviest edge of some cycle in G, then every MST of G excludes $e$

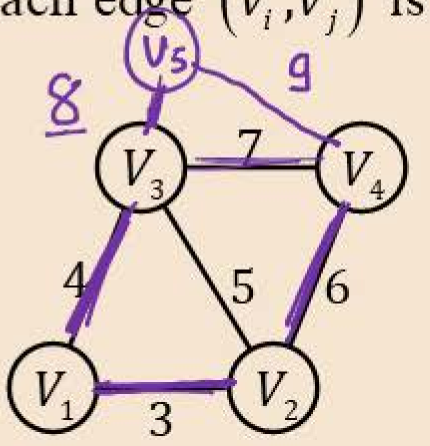
(A) I only (B) II only (C) both I and II (D) neither I nor II

# GATE 2011, Question Number 54, 2-Marks

An undirected graph $G(V,E)$ contains $n(\, n > 2\,)$ nodes named $v_1, v_2, \ldots, v_n$. Two nodes $v_i, v_j$ are connected if and only if $0 < |i-j| \le 2$. Each edge $(v_i, v_j)$ is assigned a weight $i + j$. A sample graph with $n = 4$ is shown below

Logical question

- maximum
- min will be always pml




Series will be produced (06 - 07)

n=4 n=5

13 is the answer 21

What will be the cost of the minimum spanning tree (MST) of such a graph with $n$ nodes?

(A) $\frac{1}{12}\left(11n^2 - 5n\right)$ ✓ $\frac{1}{12}(176 - 20$ $\frac{1}{12} \times 156 = 13$

(B) $n^2 - n + 1$ $25 - 5 + 1 = 21$ $16 - 4 + 1 = 13$

$\frac{1}{12}(11 \times 25 - 25)$ $250/12 = 21$

$30 - 11 = 19$

(C) $6n - 11$ 13

(D) ~~$2n+1$~~ 9

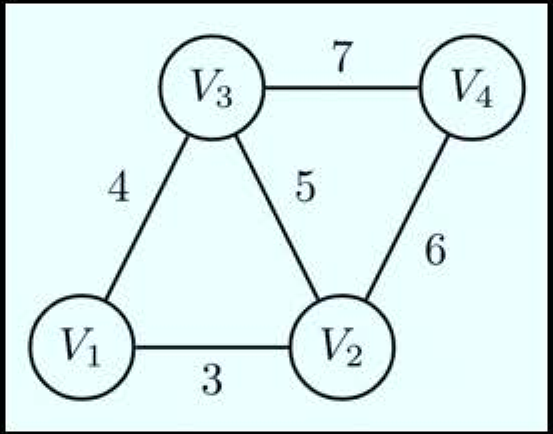
# GATE 2011, Question Number 54, 2-Marks

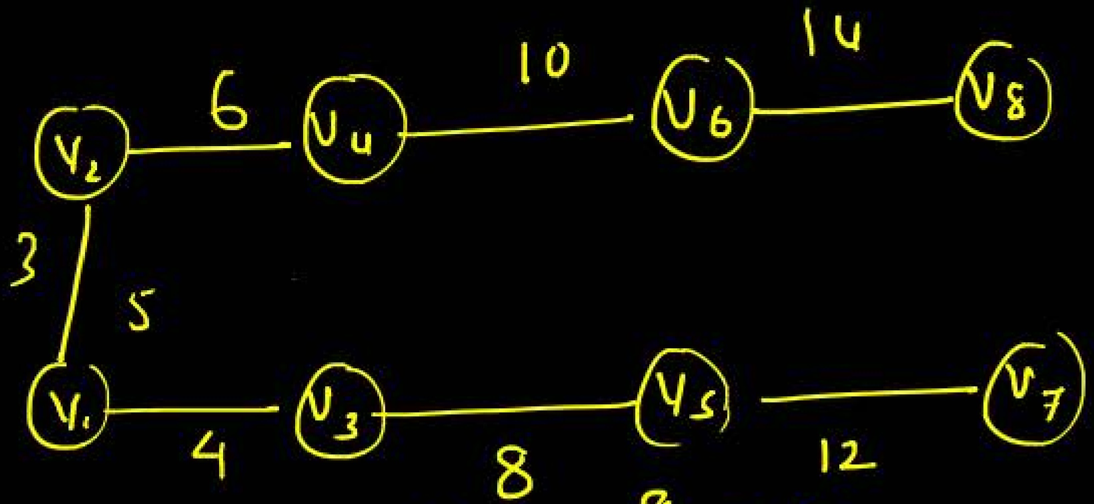
An undirected graph G(V,E) contains n(n>2) nodes named v1,v2,…,vn. Two nodes vi,vj are connected if and only if 0< |i−j| ≤ 2. Each edge (vi,vj) is assigned a weight i+j. A sample graph with n=4 is shown below.

$v_1 - v_5$

$0 < |1-5| \leq 2$

$\underline{4 \leq 2}$




$\underline{2(n-1)}$ $\quad \underline{8}$ $\quad n=8$

$3+4+6+8+10+12+\underline{14}+\cdots+\underline{2(n-1)}$

$1+2+4+6+\cdots+2(n-1) = 1+2(1+2+3+\cdots(n-1)) = 1+\frac{n(n-1)}{2}\cdot 2 = n^2-n+1$

# GATE 2011, Question Number 54, 2-Marks

What will be the cost of the minimum spanning tree (MST) of such a graph with n nodes?

(A) $\frac{1}{12}(11n^2 - 5n)$

(C) $n^2 - n + 1$

(C) $6n - 11$

(D) $2n + 1$

## GATE 2011, Question Number 55

The length of the path from $v_5$ to $v_6$ in the MST of previous question with $n = 10$ is

(A) 11 (B) 25

(C) 31 (D) 41

## GATE 2011, Question Number 55

Consider a complete undirected graph with vertex set {0, 1, 2, 3, 4}.

Entry $W_{ij}$ in the matrix W below is the weight of the edge {i, j}.

$$W = \begin{pmatrix} 0 & 1 & 8 & 1 & 4 \\ 1 & 0 & 12 & 4 & 9 \\ 8 & 12 & 0 & 7 & 3 \\ 1 & 4 & 7 & 0 & 2 \\ 4 & 9 & 3 & 2 & 0 \end{pmatrix}$$

What is the minimum possible weight of a spanning tree T in this graph such that vertex 0 is a leaf node in the tree T?

(A) 7 (B) 8

(C) 9 (D) 10

# GATE 2010

Consider a complete undirected graph with vertex set {0, 1, 2, 3, 4}.

Entry $W_{ij}$ in the matrix W below is the weight of the edge {i, j}.

$$W = \begin{pmatrix} 0 & 1 & 8 & 1 & 4 \\ 1 & 0 & 12 & 4 & 9 \\ 8 & 12 & 0 & 7 & 3 \\ 1 & 4 & 7 & 0 & 2 \\ 4 & 9 & 3 & 2 & 0 \end{pmatrix}$$

What is the minimum possible weight of a spanning tree T in this graph such that vertex 0 is a leaf node in the tree T?

(A) 7 (B) 8

(C) 9 (D) 10

# GATE 2010

Consider a complete undirected graph with vertex set {0, 1, 2, 3, 4}. Entry $W_{ij}$ in the matrix W below is the weight of the edge {i, j}.

$$W = \begin{pmatrix} 0 & 1 & 8 & 1 & 4 \\ 1 & 0 & 12 & 4 & 9 \\ 8 & 12 & 0 & 7 & 3 \\ 1 & 4 & 7 & 0 & 2 \\ 4 & 9 & 3 & 2 & 0 \end{pmatrix}$$

What is the minimum possible weight of a path P from vertex 1 to vertex 2 in this graph such that P contains at most 3 edges?

(A) 7 (B) 8

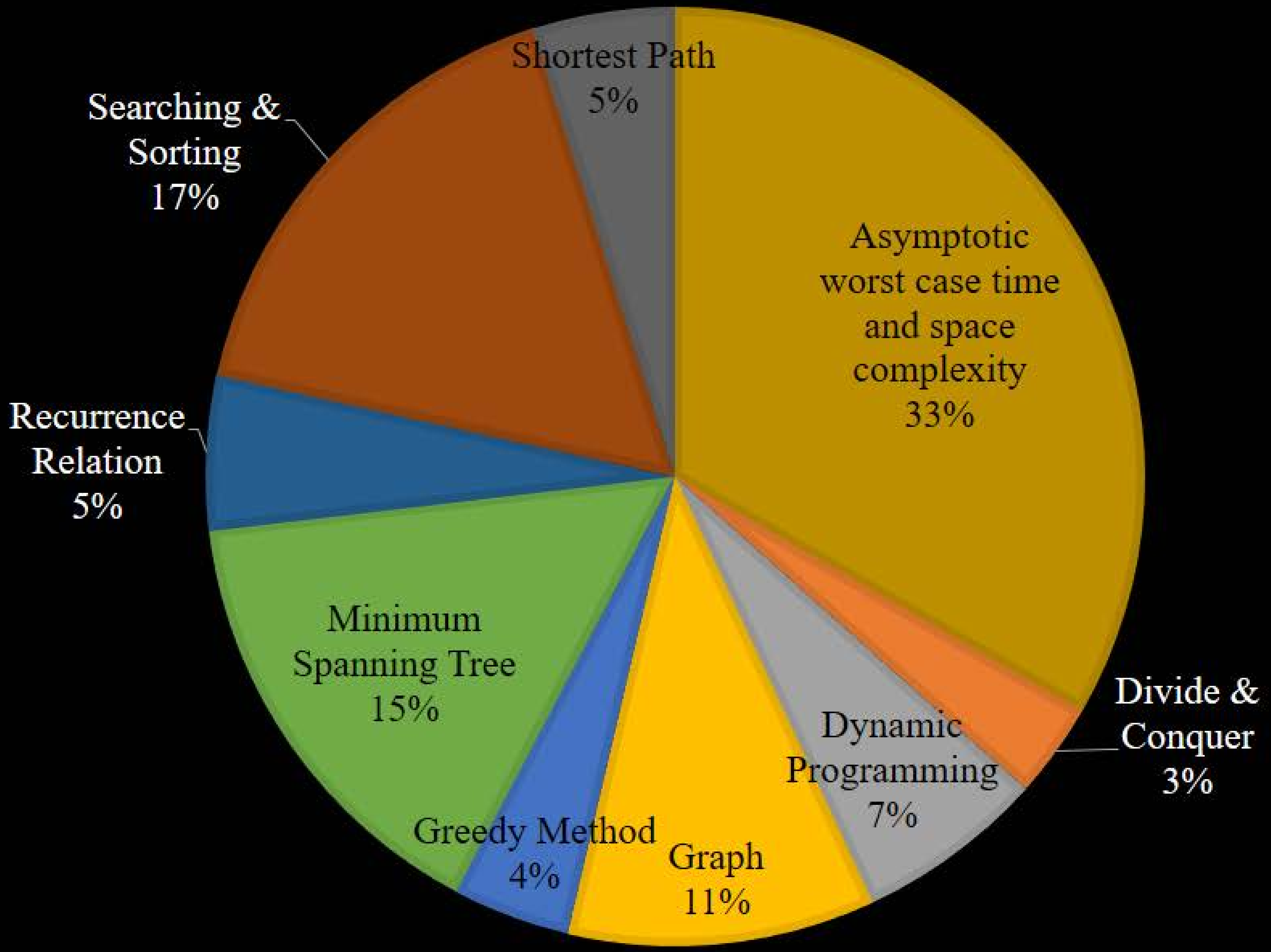
(C) 9 (D) 10

## GATE 2007-IT

What is the largest integer m such that every simple connected graph with n vertices and n edges contains at least m different spanning trees?

(A) 1 (B) 2 (C) 3 (D) n

# *Single Source Shortest Path*

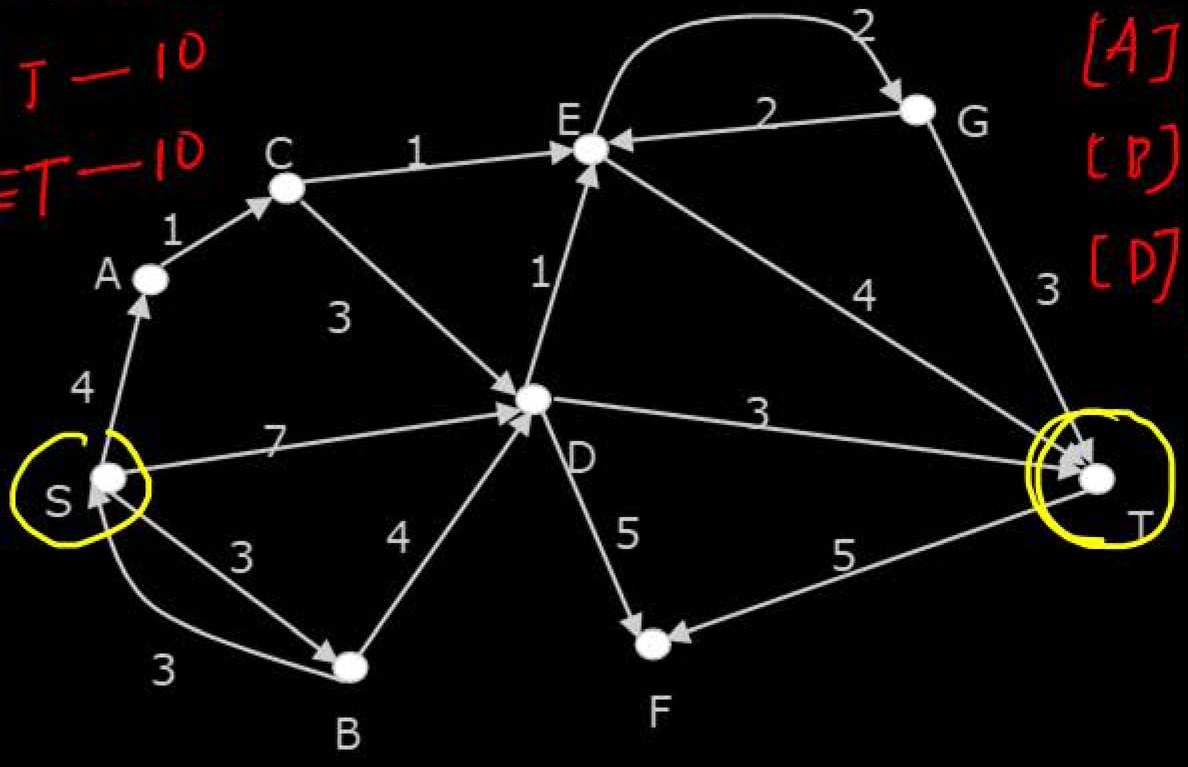


# GATE 2003, Question Number 67

Not which order vertex

Consider the directed graph shown in the figure below. There are multiple shortest paths between vertices S and T. Which one will be reported by Dijkstra's shortest path algorithm? (Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered.)

SDT – 10
SBDT – 10
SACET – 10




[A]
[B]
[D]

strictly shorter path than update

$\underline{7}$ another cost of 7

to reach from s to t which path should be followed

(A) SDT

(B) SBDT

(C) SACDT

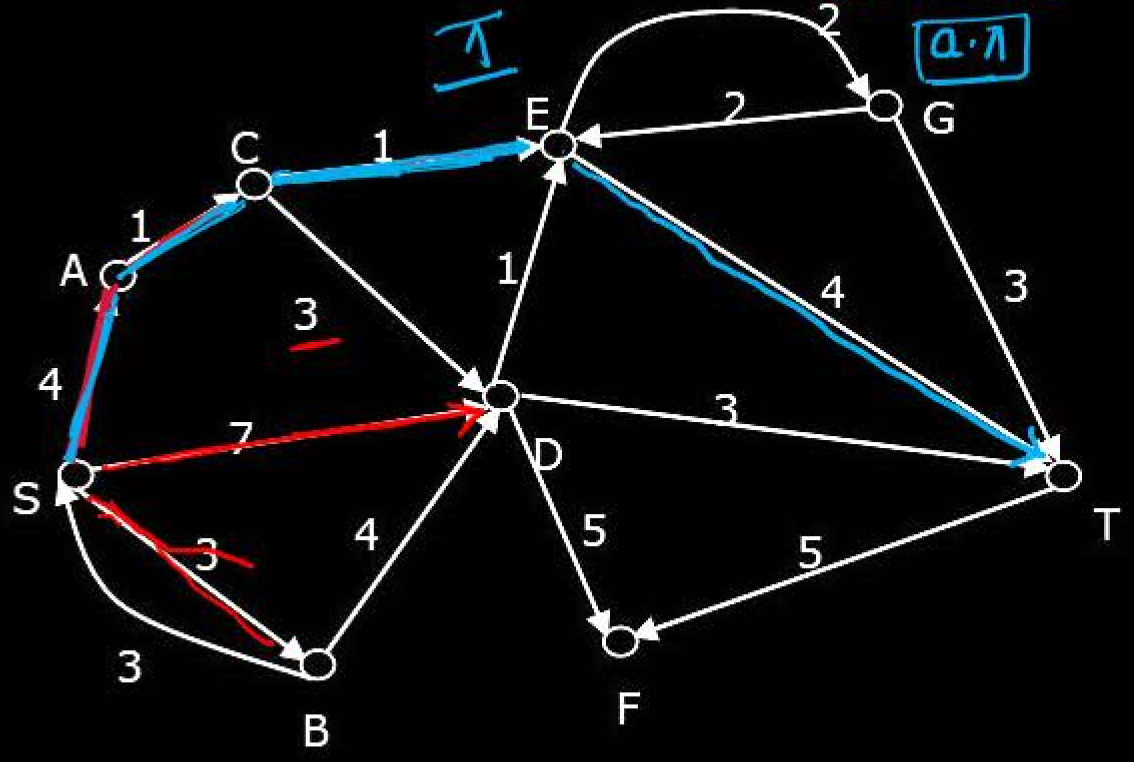
(D) SACET

S,

(S,T)

SACET

d[a]
a[b] d(c)
$\boxed{a \cdot 1}$




| s | a | b | c | d | e | f | g | t |
|---|---|---|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| — | 4(s) | 3(s) | ∞ | 7(s) | ∞ | ∞ | ∞ | ∞ |
| — | 4 | — | ∞ | 7(s) | ∞ | ∞ | ∞ | ∞ |
| — | — | — | 5(a) | 7(s) | ∞ | ∞ | ∞ | ∞ |
| — | — | — | — | 7(s) | 6(c) | ∞ | ∞ | ∞ |
| — | — | — | — | 7(s) | 6 — | ∞ | 8(E) | 10(E) |
| — | — | — | — | — | — | 12(D) | 8 | 10(E) |
| — | — | — | — | — | — | 12 | — | 10(G) |
| — | — | — | — | — | — | 12 | — | — |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

# GATE 2004, Question Number 44

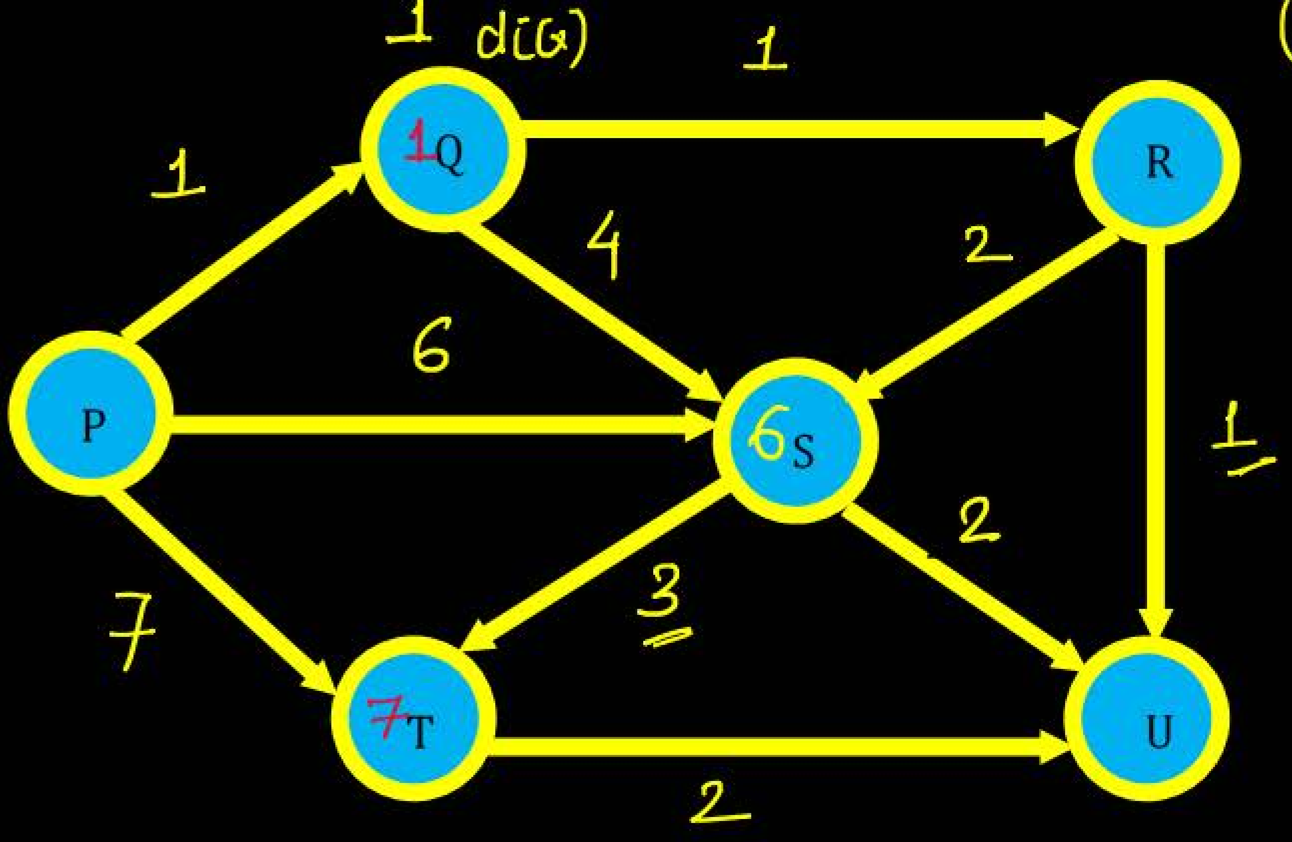
Suppose we run Dijkstra's single source shortest-path algorithm on the following edge-weighted directed graph with vertex P as the source.

Question 12 Text book

Krusal

Execute

P, Q R U S t




| P | Q | R | S | T | U |
|---|---|---|---|---|---|
| 0 | ∞ | ∞ | ∞ | ∞ | ∞ |
| – | 1 | ∞ | 6 | 7 | ∞ |
| – | – | 2 | 5 | 7 | ∞ |
| – | – | – | 4 | 7 | 3 |
| – | – | – | 4 | 7 | – |
| – | – | – | – | 7 | – |
| – | – | – | – | – | – |