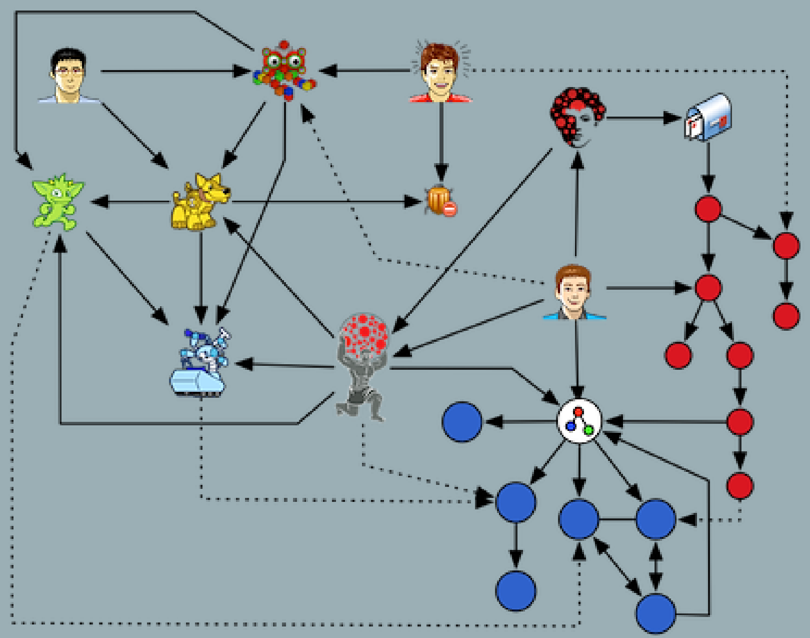
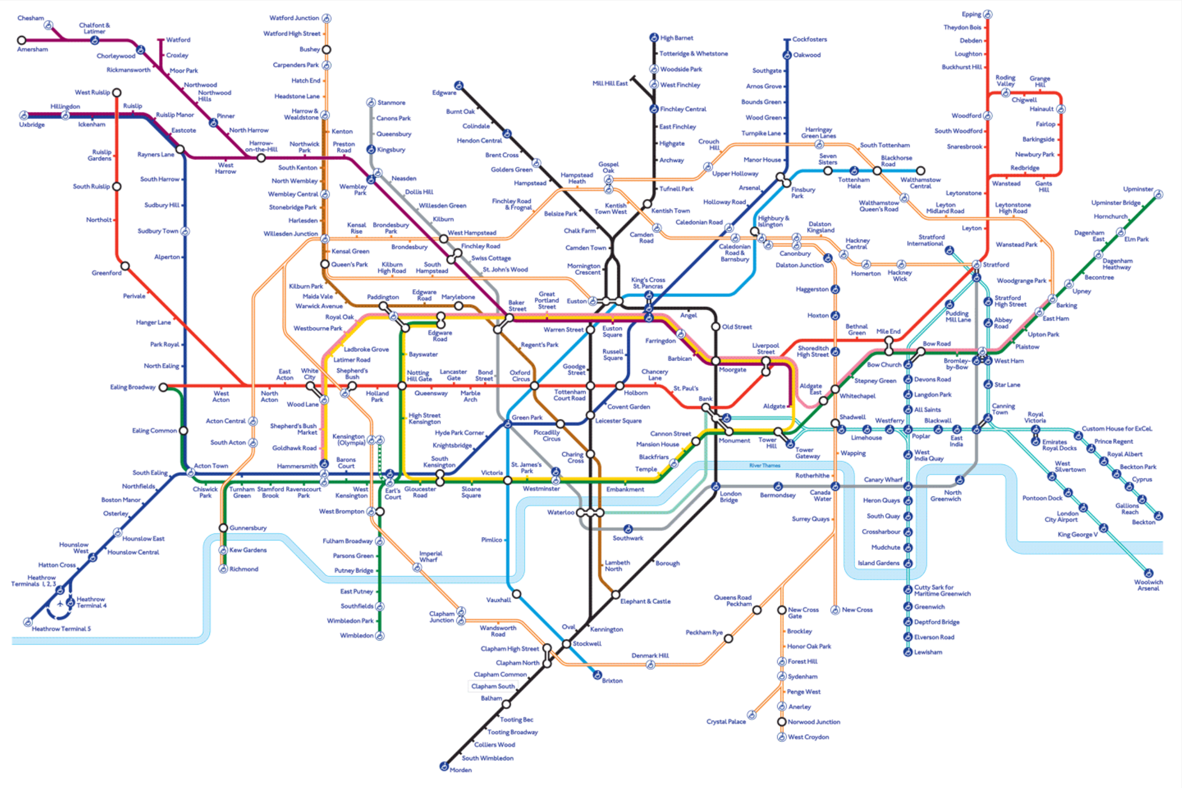
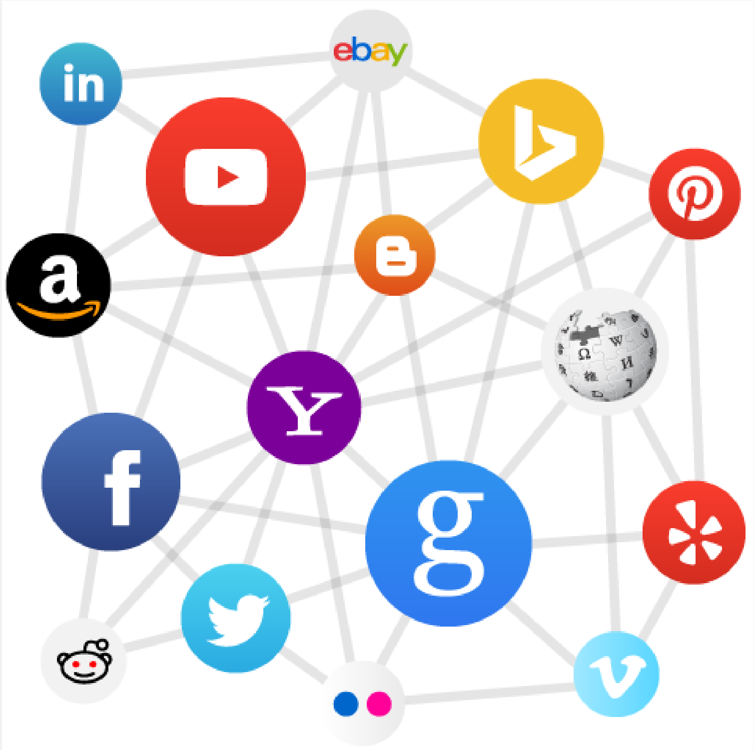
LECTURE 6:
GRAPH THEORY



GRAPH THEORY IN EVERYDAY LIFE





- Graph theory is used in **cybersecurity** to identify hacked or criminal servers and generally for network security.

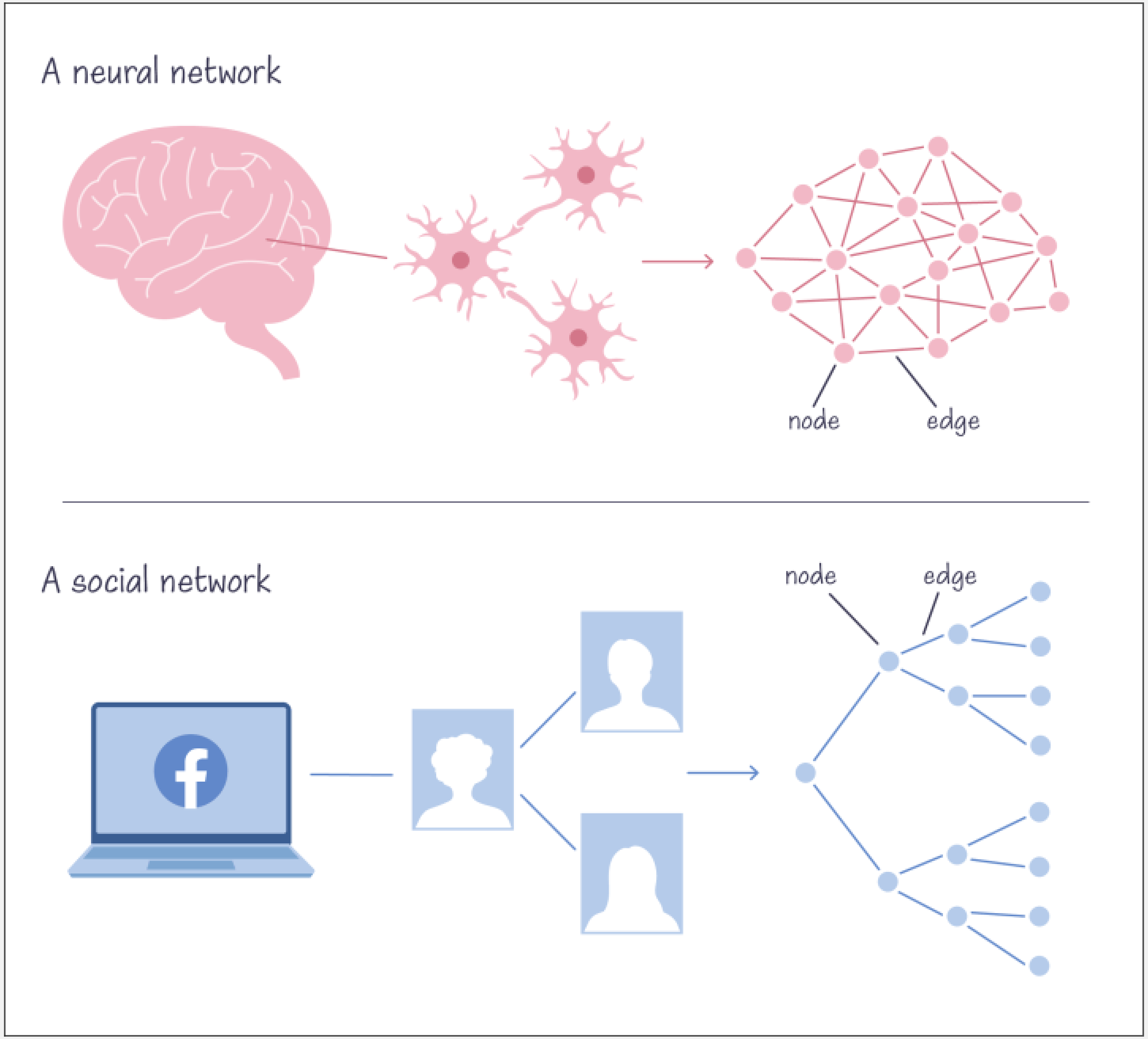
- Graph theory, and in particular rooted tree diagrams of a genome, is used in **the evolution of SARS-CoV-2**.

- Determining how best to add streets to congested areas of cities uses graph theory.

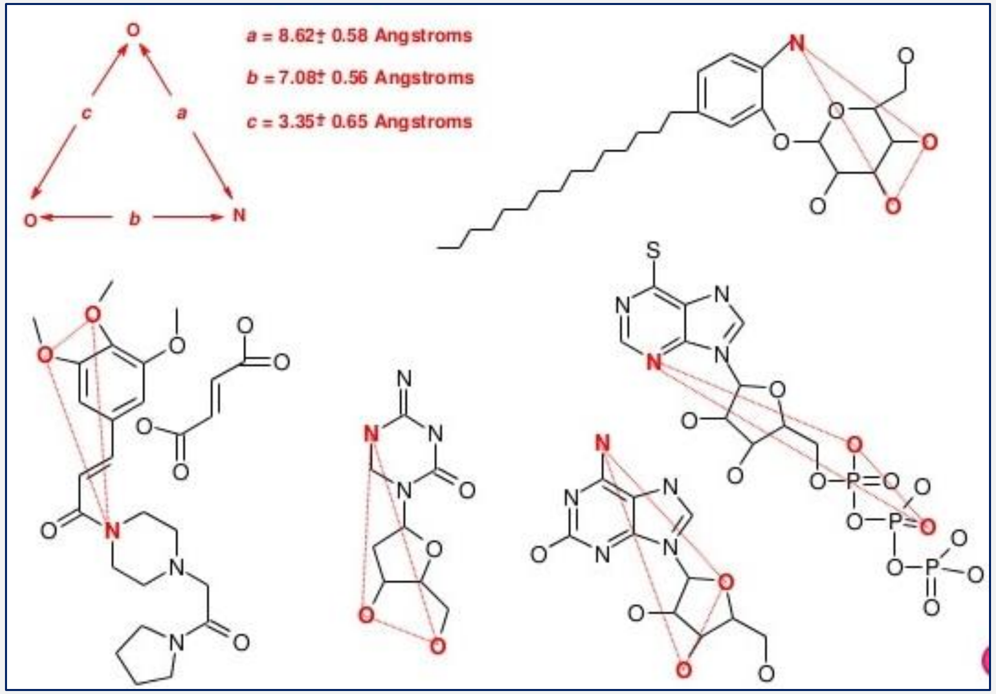
- Graph theory is used in **neuroscience** to study brain network organization

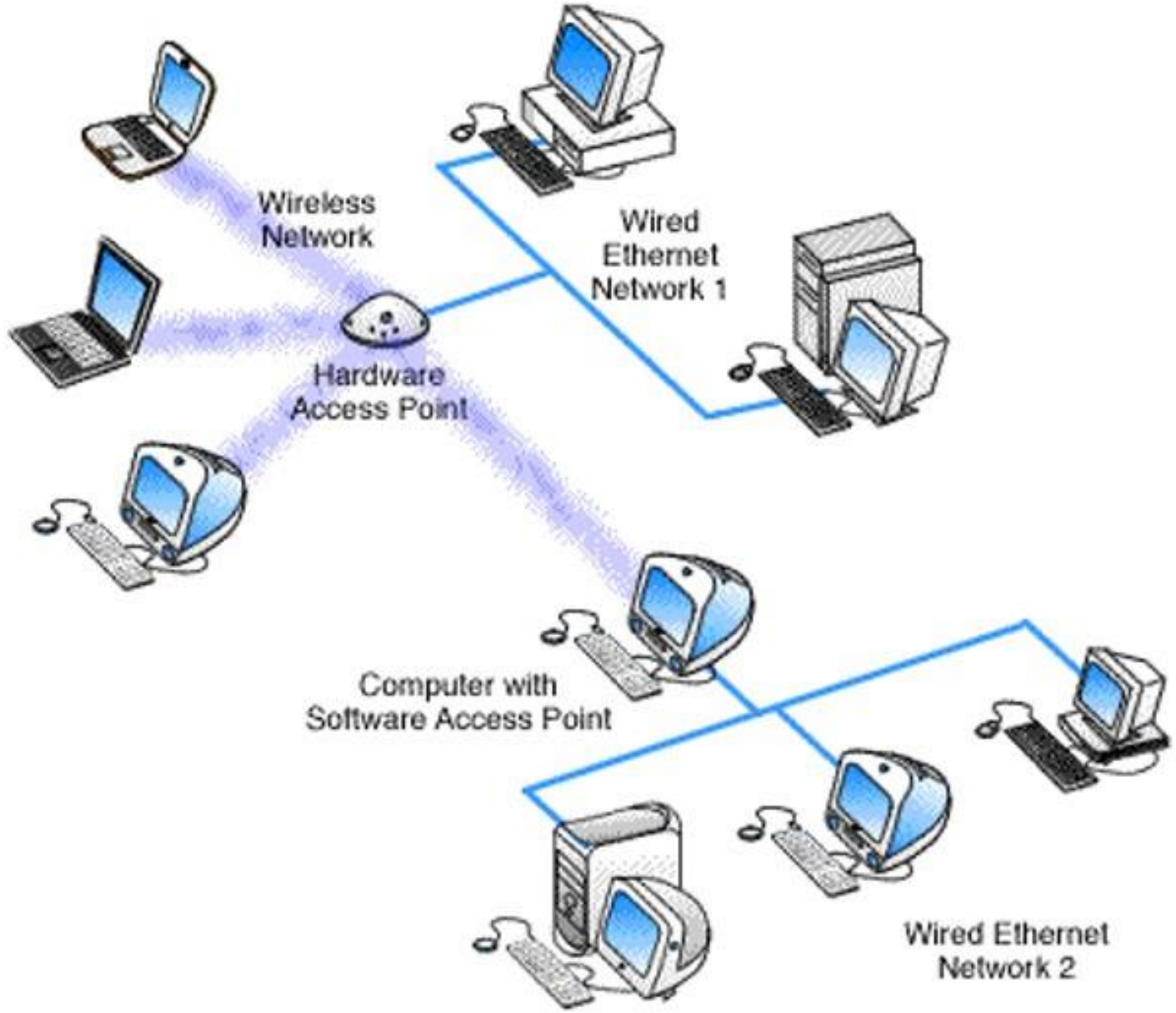
- Graph theory is used in **DNA sequencing**.

- **Design of radar and sonar systems** uses graph theory via Golomb rulers.

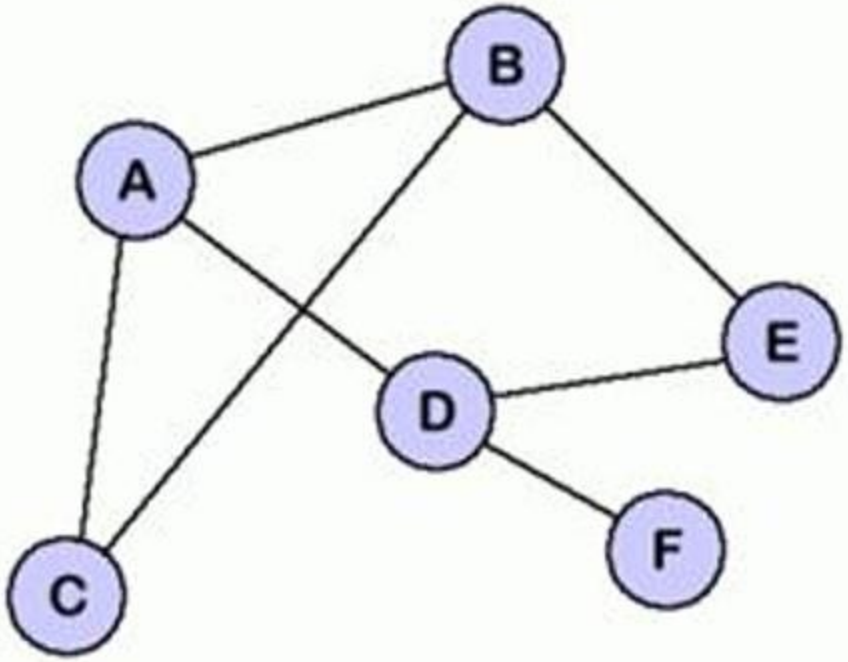


Graph Theory in Drug Design





Physical system: computer network

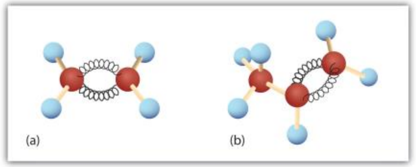


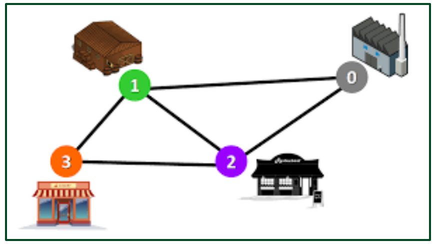
Model: graph structure

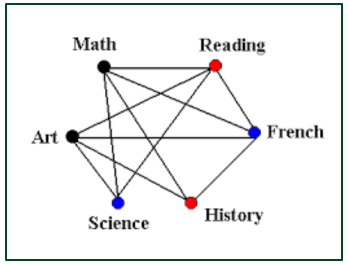
Vertices represent devices
Edges represent links

VARYING APPLICATIONS (EXAMPLES)

- Computer networks
- Distinguish between two chemical compounds with the same molecular formula but different structures
- Pages are linked by hyperlinks on the internet
- Components of electric circuit
- Solve shortest path problems between cities
- Scheduling exams and assign channels to television stations



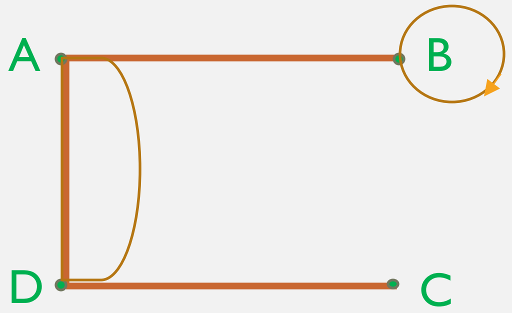




GRAPHS

- A graph $G = (V, E)$ consists of V, a non-empty set of vertices and E a set of edges. Edges connect either one or two vertices, called endpoints.

not a simple graph



Simple graph:
- each edge connects two different vertices
- no two edges connect the same two vertices

A simple railway tracks connecting different cities

Set of vertices and edges

Vertices cannot be empty set, but edges

SOME TERMINOLOGY

Multigraph

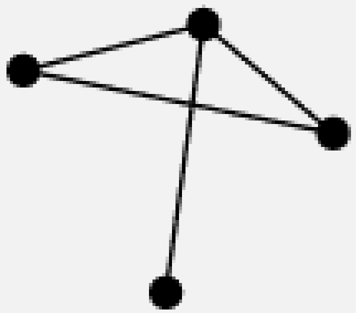
multiple edges connecting the same two vertices

Loop

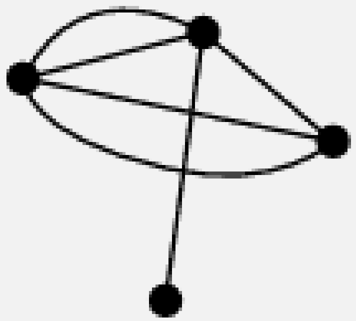
an edge that connects a vertex to itself

Pseudograph

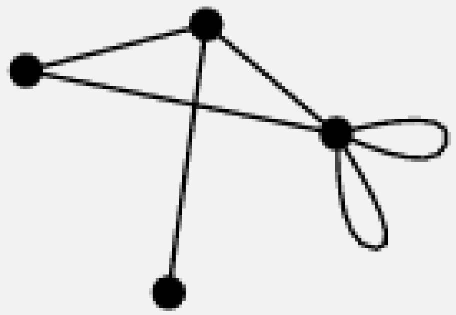
may include loops as well as multiple edges



simple graph



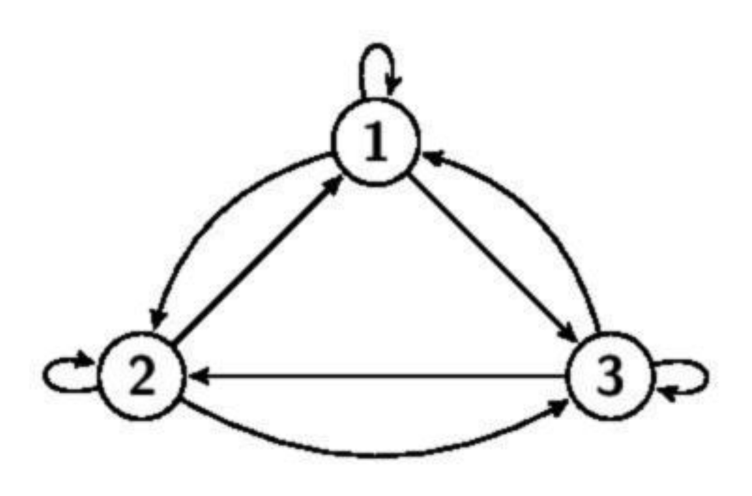
multigraph



pseudograph

DIRECTED GRAPH - DIGRAPH

- A graph $G = (V, E)$ where each edge is associated with pair of vertices. The directed edge associated with (a, b) begins at a and ends at b.



SUMMARY

Type	Directed edges	Multiple edges	Loops
Simple Graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple Directed Graph	Directed	No	No
Directed Multigraph	Directed	Yes	Yes
Mixed Graph	Both	Yes	Yes

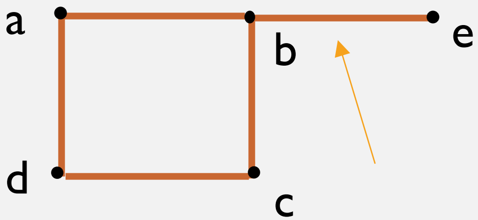
MORE TERMINOLOGY

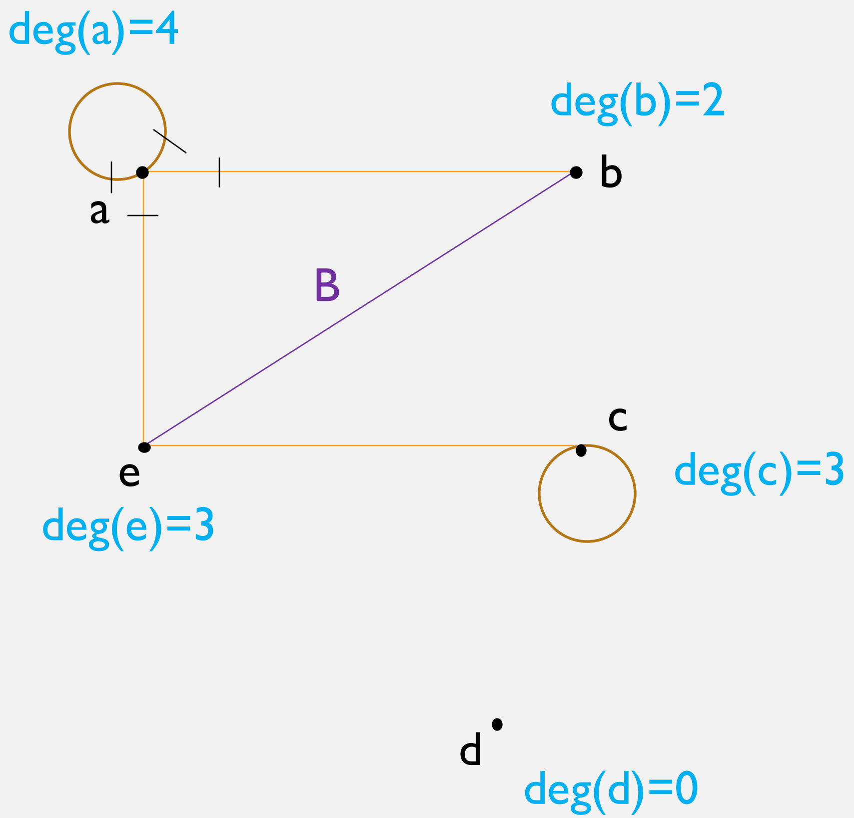
FOR UNDIRECTED GRAPHS

- **Adjacent vertices** (neighbors) - (a, b) but (b, c)
- **Incident** – edge B is incident with vertices e and b
- **Neighborhood**: $N(A) = V_{v \in A} N(v)$

N(e)={a, b, c}

- **Degree**: sum of ins and outs
- **Isolated** (d)
- **Pendant** (connected only once to another vertex) (e)





THE HAND-SHAKING THEOREM

Suppose there are 6 people in a room, and each must shake hands with every other person. How many handshakes?

$$\left.\begin{aligned} a &= 5 \\ b &= 4 \\ c &= 3 \\ d &= 2 \\ e &= 1 \end{aligned}\right\} = 15$$ total handshakes

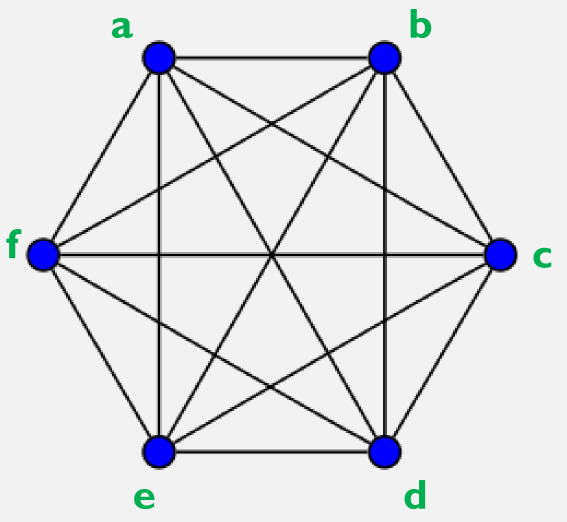
Why not 30?

$G(V, E)$ *with* m *edges*

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

$$2m = 30$$

$$m = 15$$



How many edges are there if you have 10 vertices, each of degree 6? $2m = 60,\ \ m = 30$

IMPORTANT

- The following conclusions may be drawn from the Handshaking Theorem.

In any graph,

✓ The sum of degree of all the vertices is always **even**.
✓ The sum of degree of all the vertices with odd degree is always **even**.
✓ The number of vertices with odd degree are always **even**.

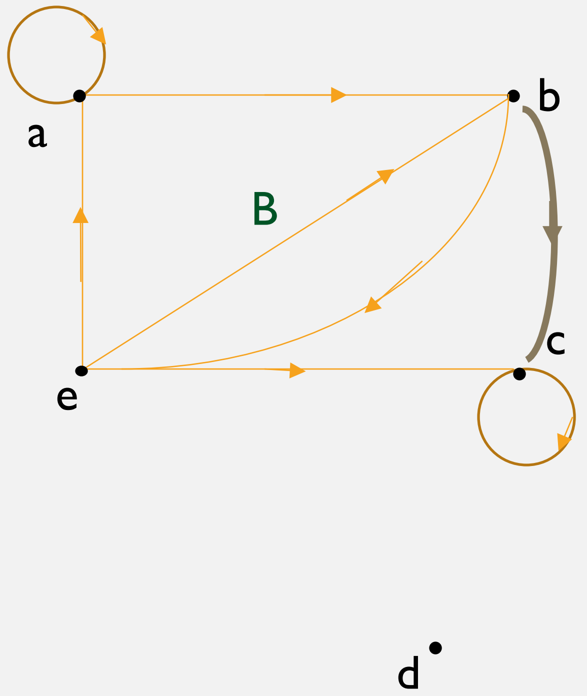
FOR DIRECTED GRAPHS

$\boxed{(a, b)}$

- Adjacent to a is adjacent to b
- Adjacent from b is adjacent from a
- Initial vertex a
- Terminal / end vertex b
- In-degree of vertex $v = deg^{-}(v)$
  b: $v = deg^{-}(b) = 2$
- Out-degree of vertex $v = deg^{+}(v)$
  b: $v = deg^{+}(b) = 1$
- $\sum_{v\in v} deg^{-}(v) = \sum_{v\in v} deg^{+}(v) = |E|$

$+1$ $+1$ $+1$

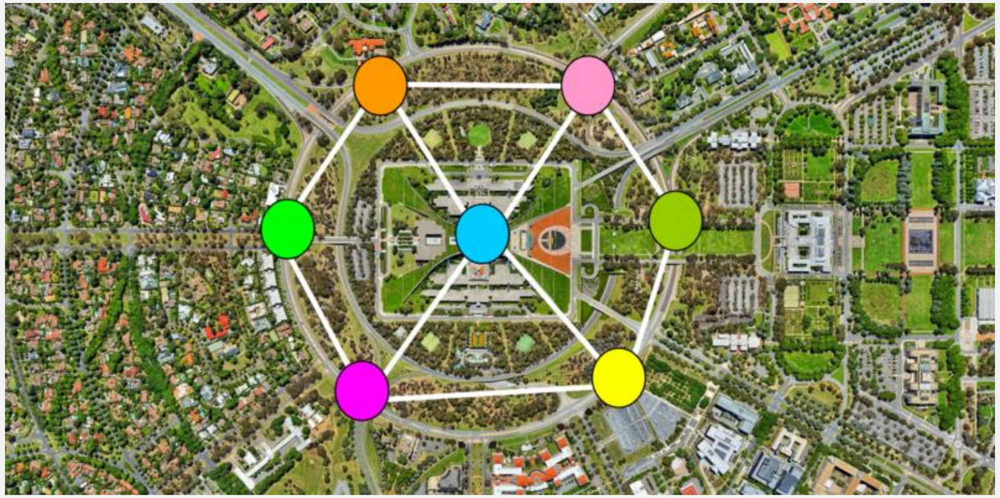
Number of edges



REPRESENTING GRAPHS WITH MATRICES

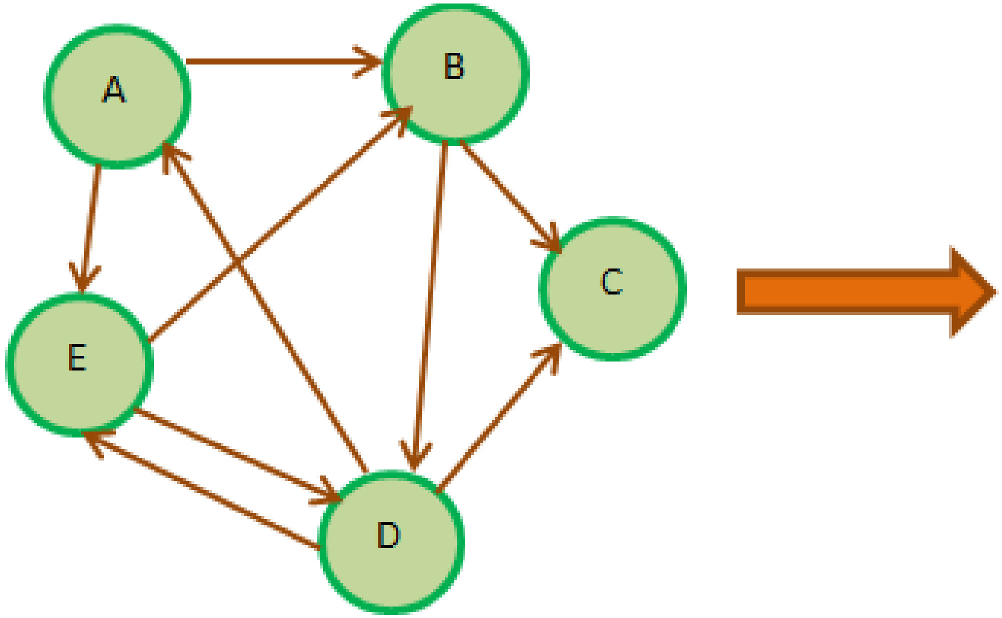
- Reference:

https://www.simplilearn.com/tutorials/data-structure-tutorial/graphs-in-data-structure



Adjacency matrix (vertex matrix)

DIRECTED GRAPH

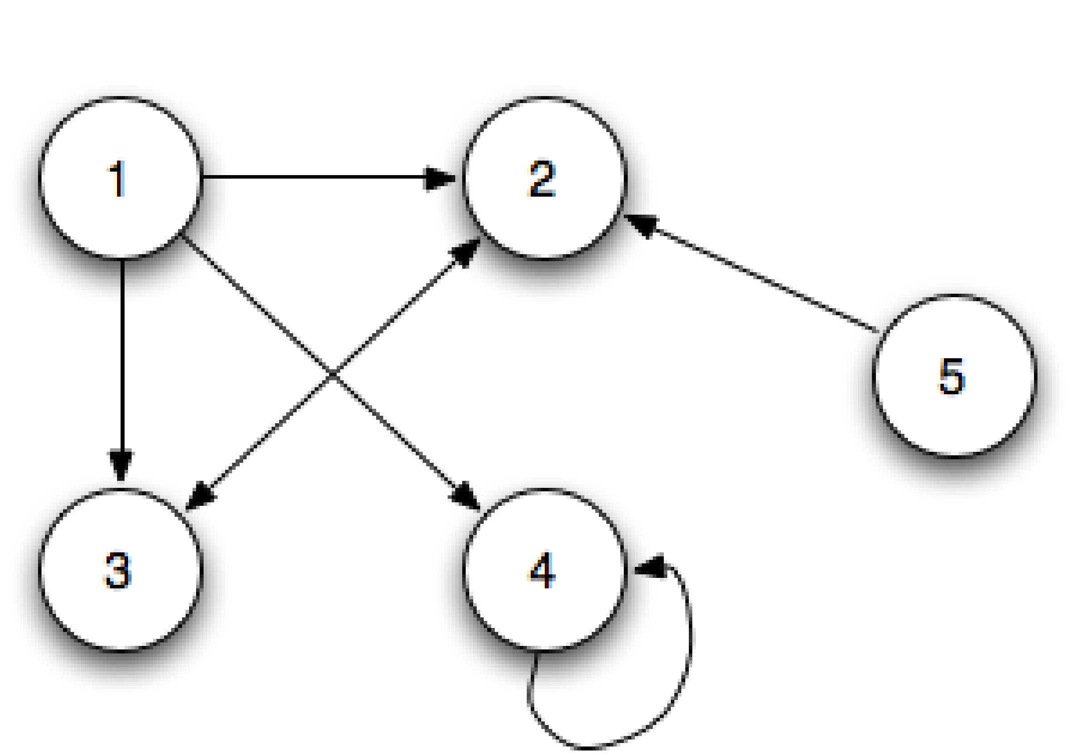


Directed graph

$$
\begin{array}{c|c}
 & \begin{array}{ccccc} A & B & C & D & E \end{array} \\
\begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0
\end{bmatrix}
\end{array}
$$

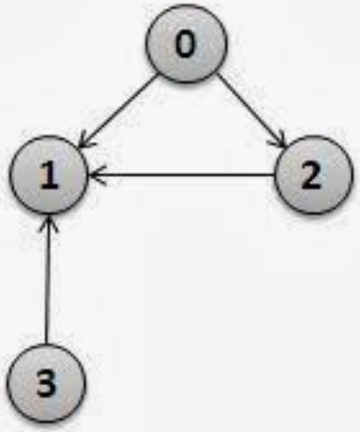
Adjacency Matrix

Adjacency matrix (vertex matrix)



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

Adjacency matrix (vertex matrix)

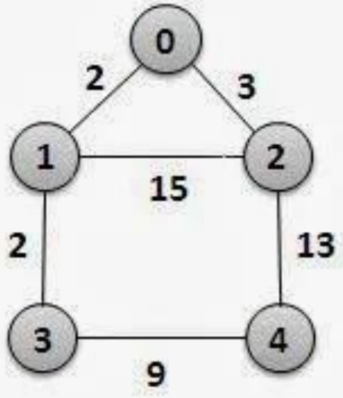


	0	1	2	3
0	0	1	1	0
1	0	0	0	0
2	0	1	0	0
3	0	1	0	0

Adjacency Matrix Representation of Directed Graph

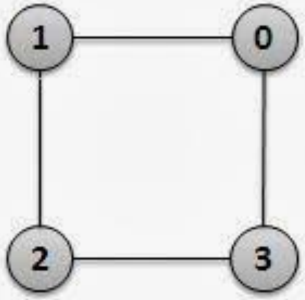
Adjacency matrix (vertex matrix)

UNDIRECTED GRAPH



	0	1	2	3	4
0	0	2	3	0	0
1	2	0	15	2	0
2	3	15	0	0	13
3	0	2	0	0	9
4	0	0	13	9	0

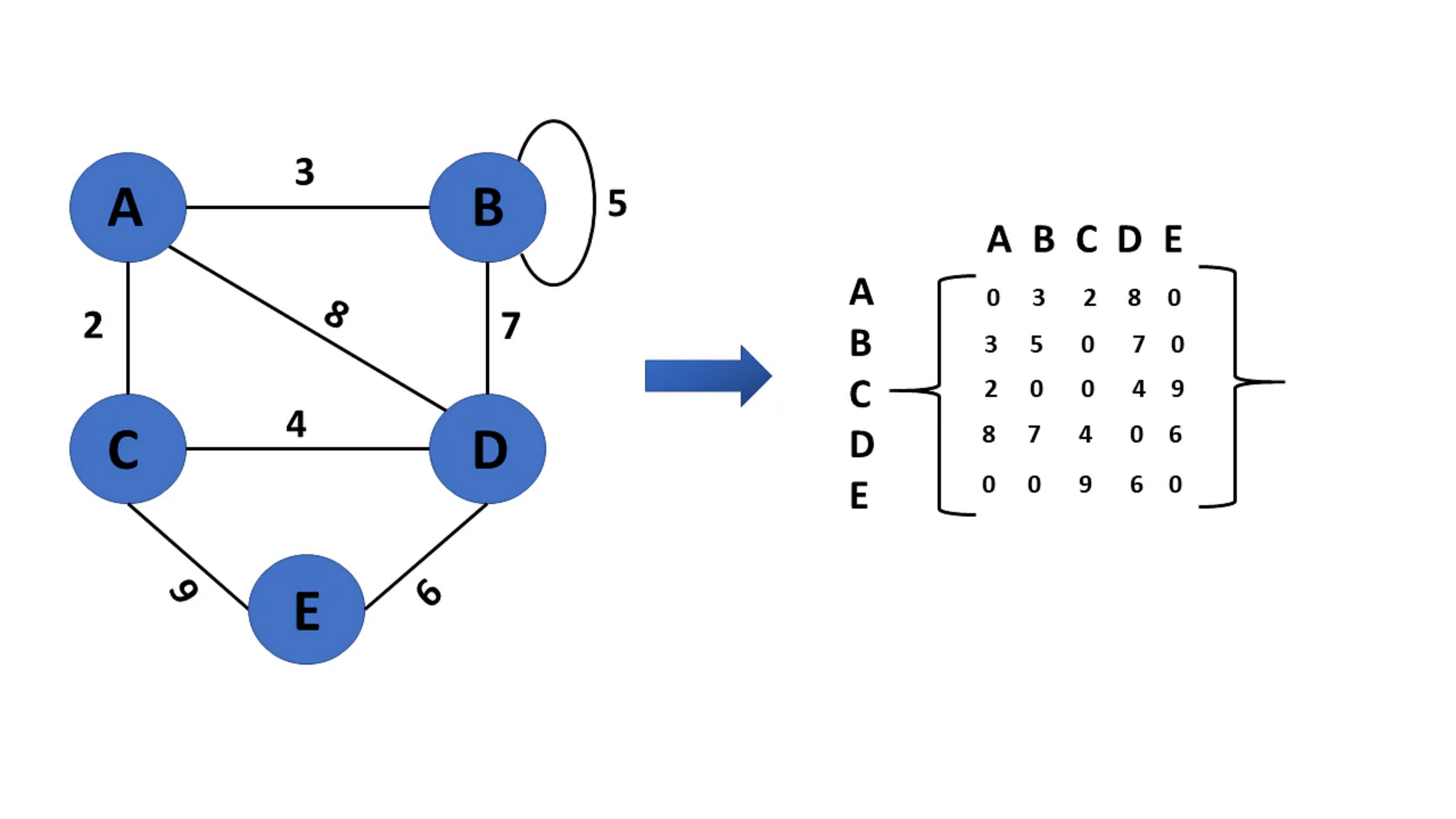
Adjacency Matrix Representation of Weighted Graph



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

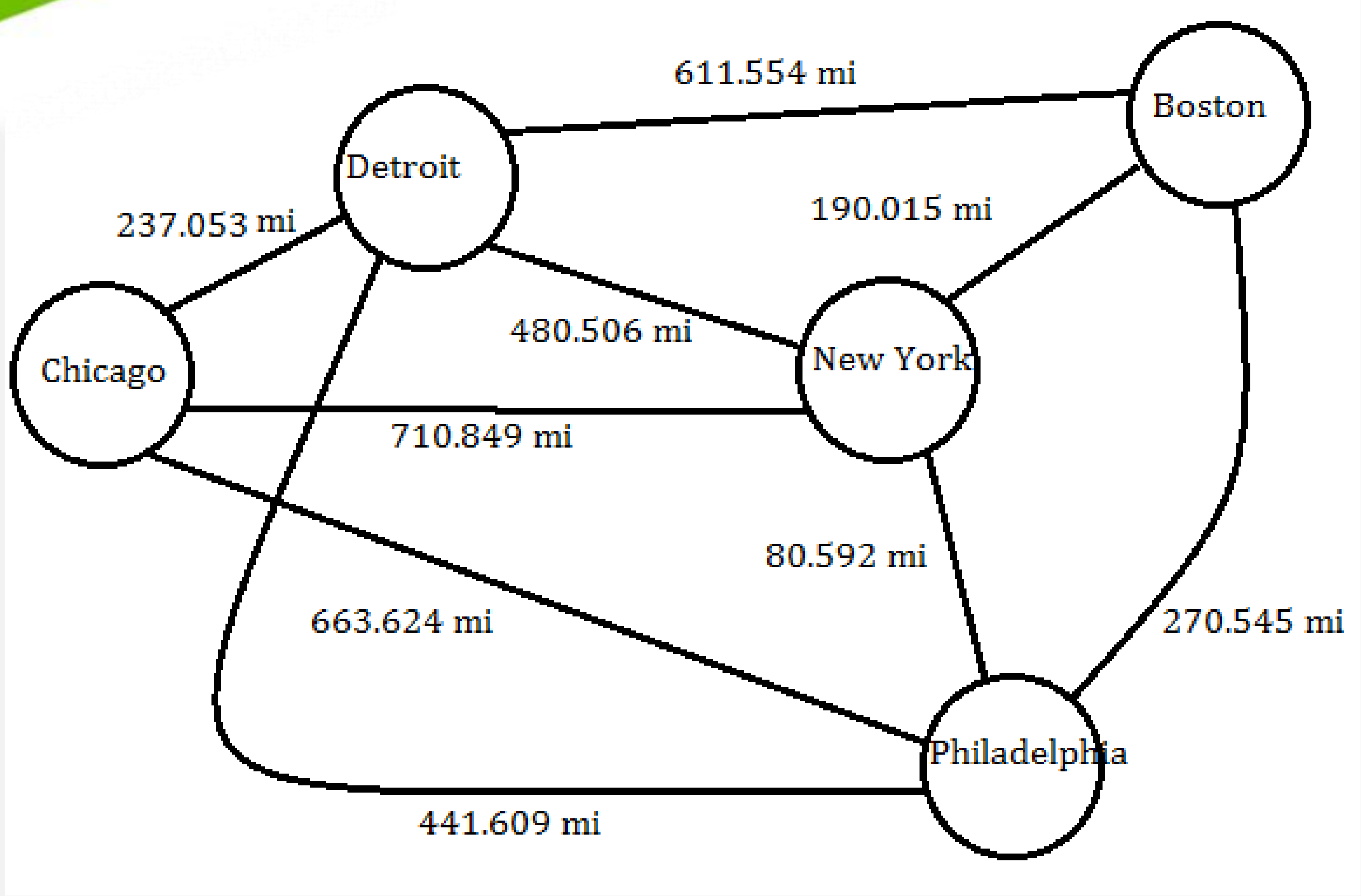
Adjacency Matrix Representation of Undirected Graph

Adjacency matrix (vertex matrix)



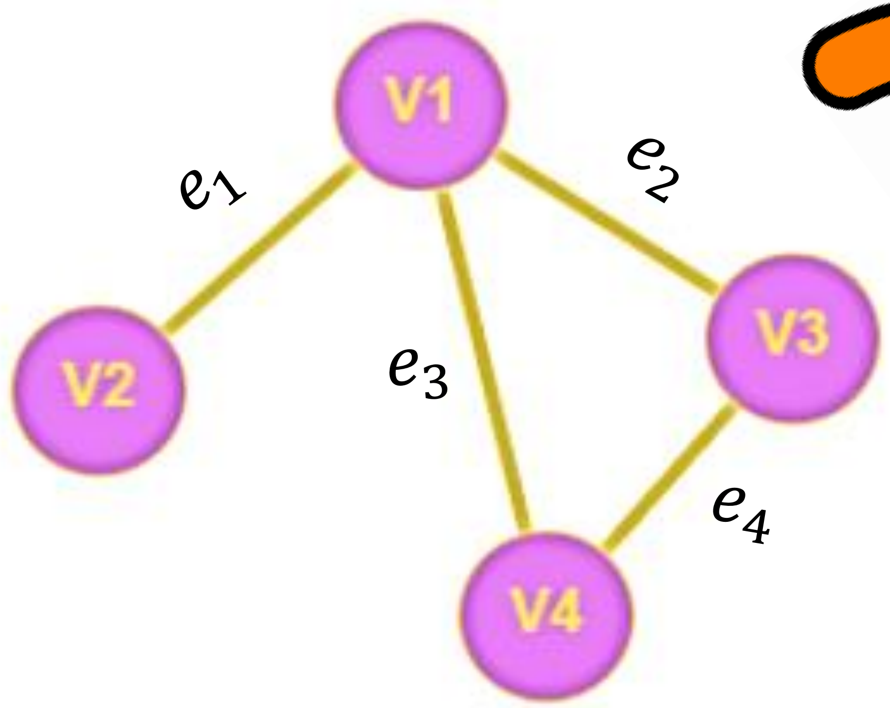
Weighted Graph

TRY IT NOW

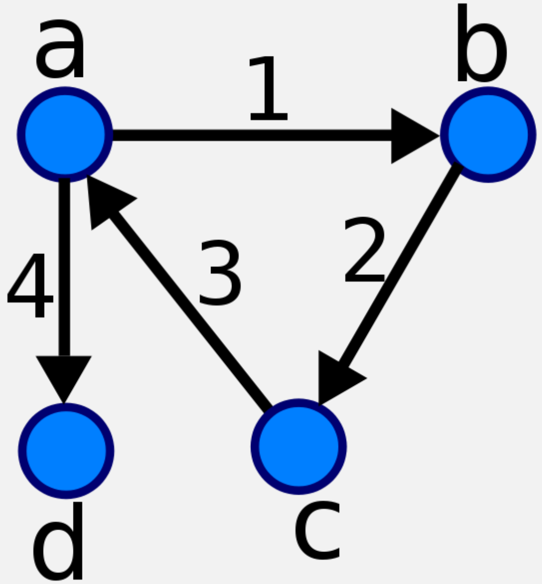


Detroit
Boston
Chicago
New York
Philadelphia
611.554 mi
237.053 mi
190.015 mi
480.506 mi
710.849 mi
80.592 mi
663.624 mi
270.545 mi
441.609 mi

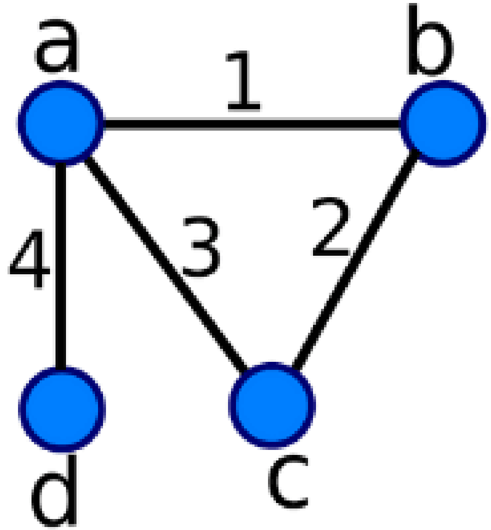
The Vertex-Edge Incidence Matrix



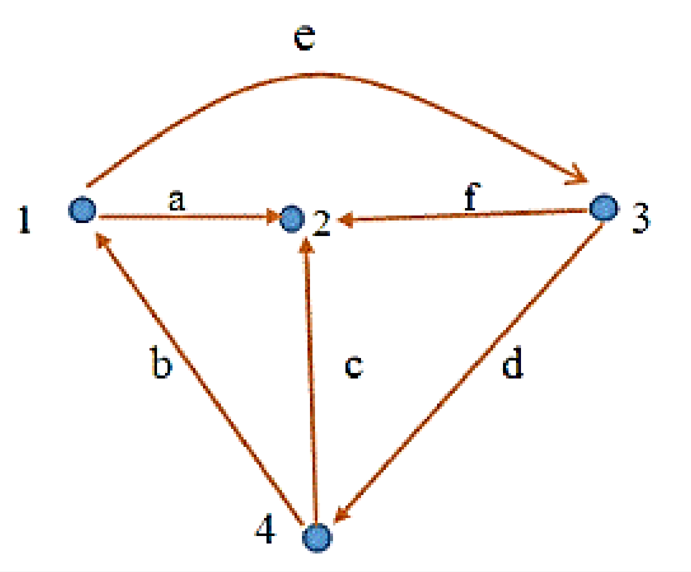
	e_1	e_2	e_3	e_4
V_1	1	1	1	0
V_2	1	0	0	0
V_3	0	1	0	1
V_4	0	0	1	1



	1	2	3	4
a	1	0	-1	1
b	-1	1	0	0
c	0	-1	1	0
d	0	0	0	-1

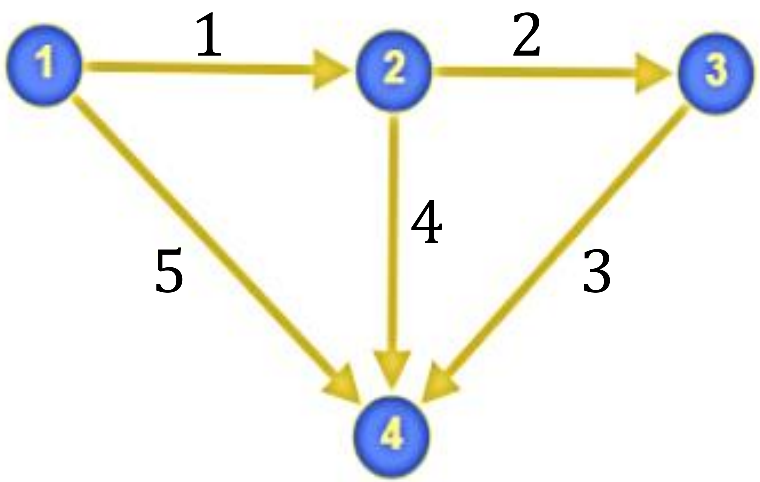


	1	2	3	4
a	1	0	1	1
b	1	1	0	0
c	0	1	1	0
d	0	0	0	1



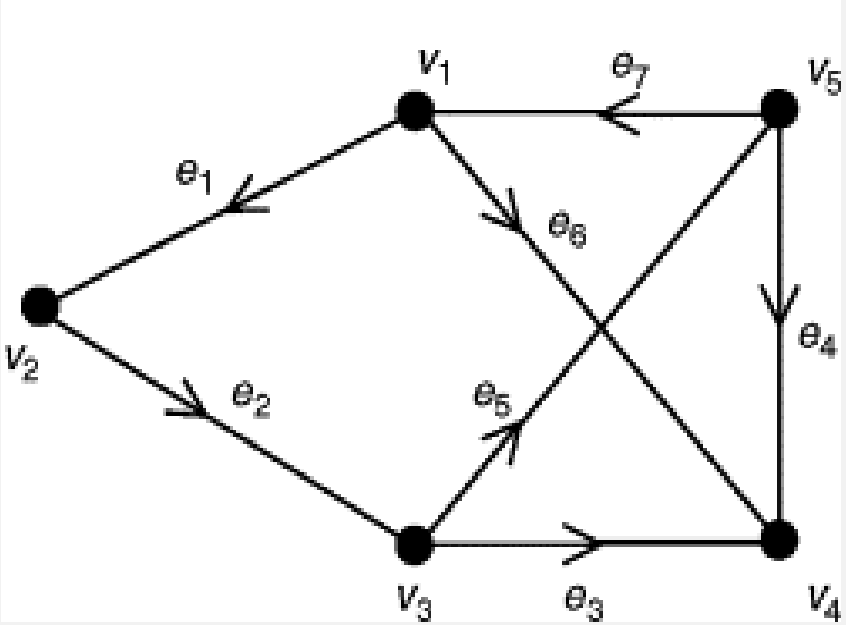
a) Create a matrix considering arrows.
b) Create a matrix without arrows.

TRY IT NOW



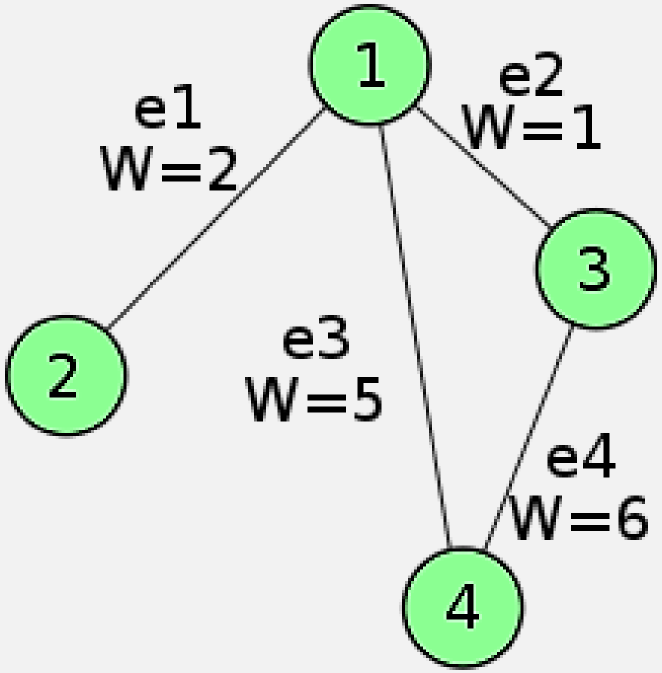
Branches

$$
\begin{array}{cc}
\text{Nodes} & \begin{array}{ccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \end{bmatrix}
\end{array}
$$



$$
\begin{array}{c}
 \\
V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} \boldsymbol{e_1} & \boldsymbol{e_2} & \boldsymbol{e_3} & \boldsymbol{e_4} & \boldsymbol{e_5} & \boldsymbol{e_6} & \boldsymbol{e_7} \end{array} \\
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & -1 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & -1 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 1
\end{bmatrix}
\end{array}
$$

Incidence Matrix

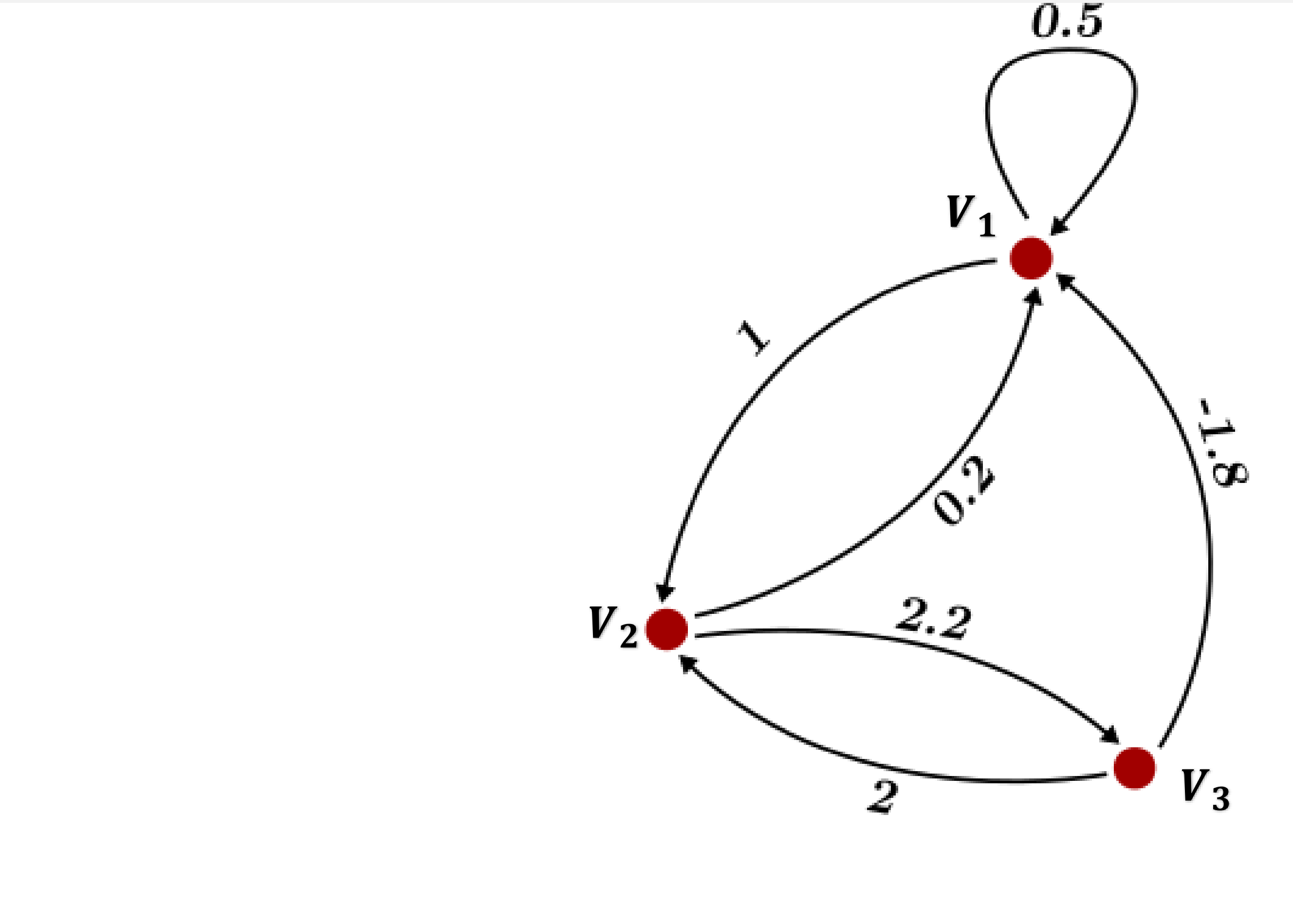


Weighted Graph

$$\begin{array}{c|cccc} & e_1 & e_2 & e_3 & e_4 \\ \hline 1 & & & & \\ 2 & & & & \\ 3 & & & & \\ 4 & & & & \end{array}$$

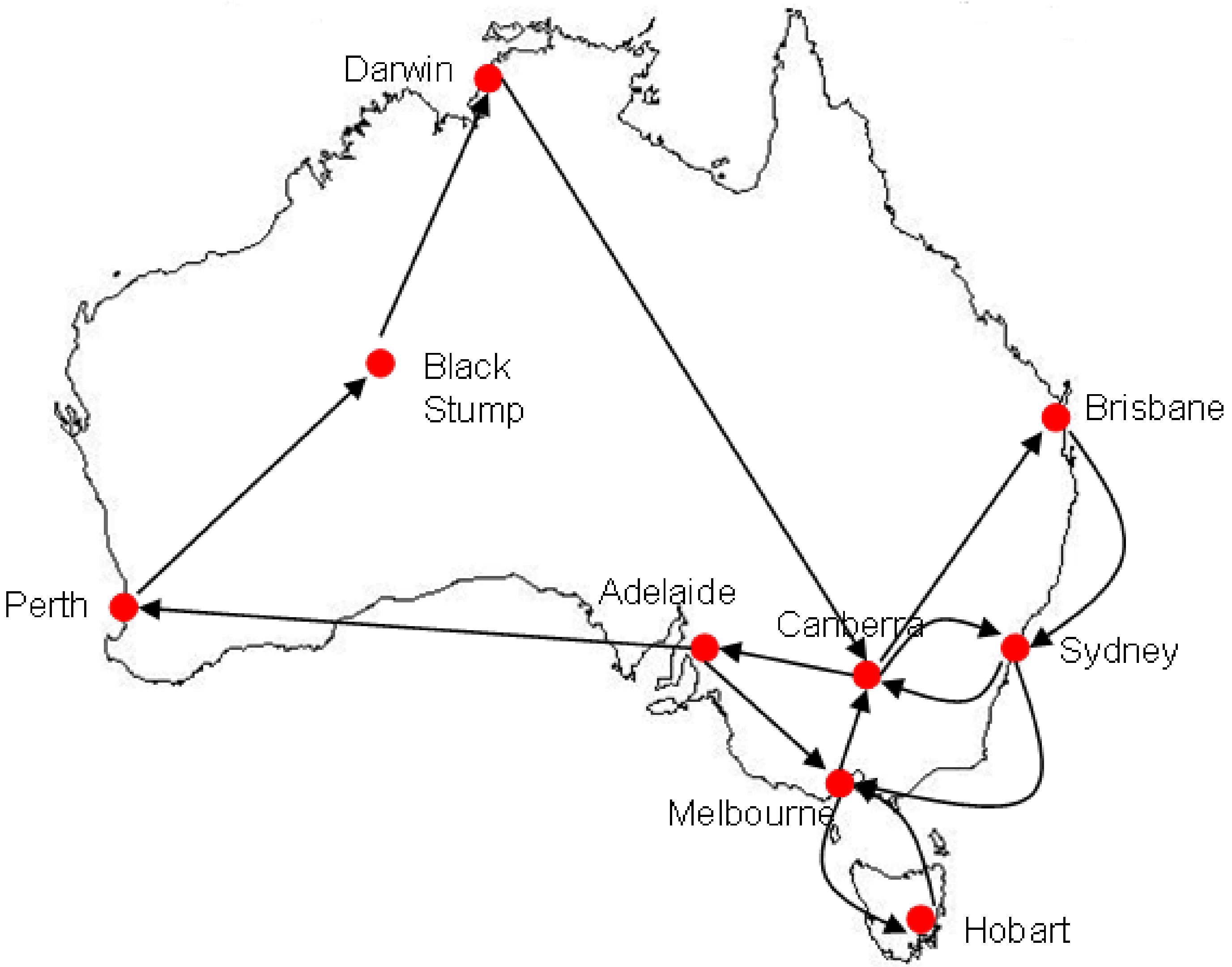
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} e_1 & e_2 & e_3 & e_4 \end{array} \\ \begin{pmatrix} 2 & 1 & 5 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 5 & 6 \end{pmatrix} \end{array} = \begin{bmatrix} 2 & 1 & 5 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 5 & 6 \end{bmatrix}.$$

Adjacency Matrix



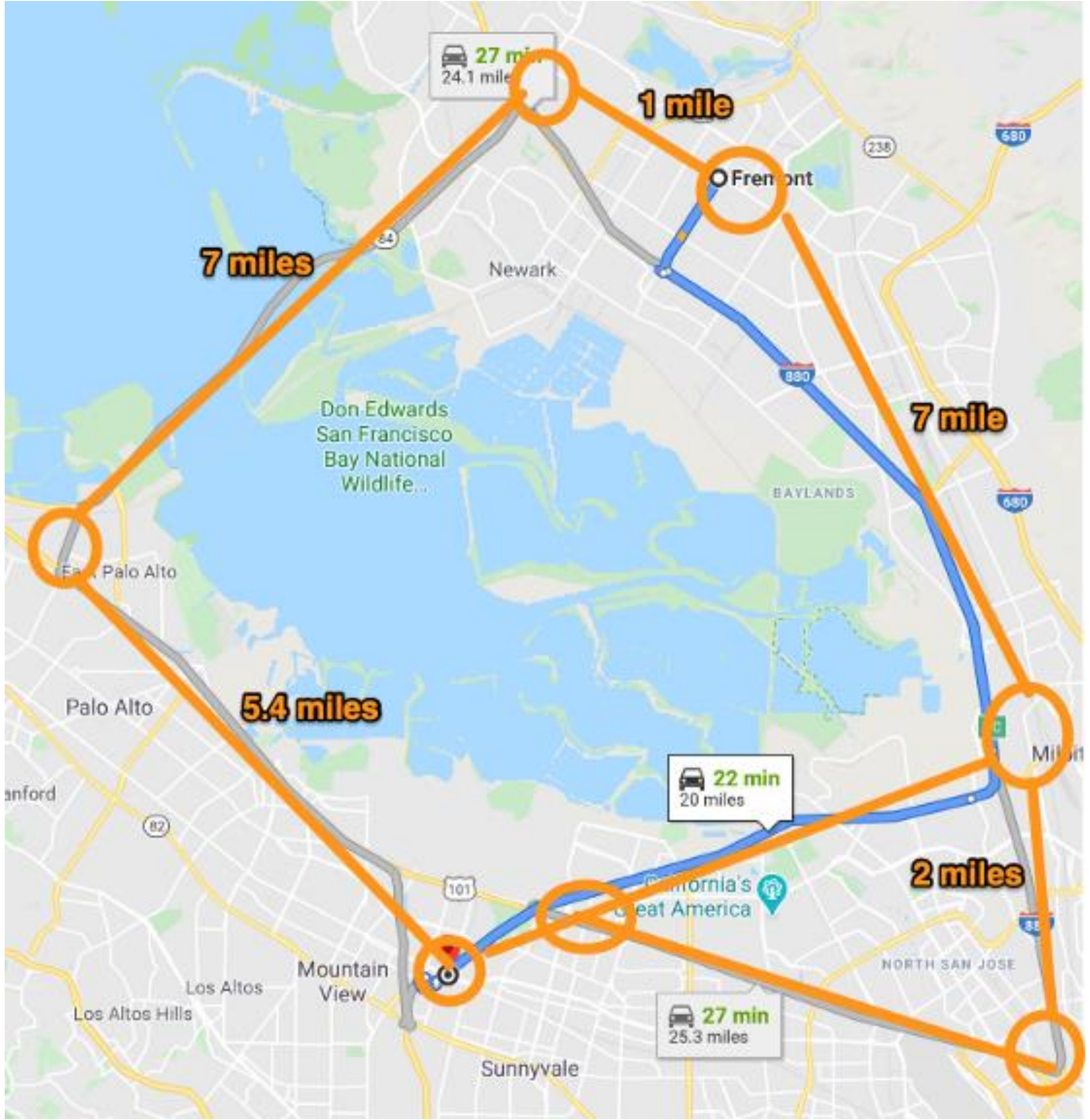
Weighted Graph

HOMEWORK



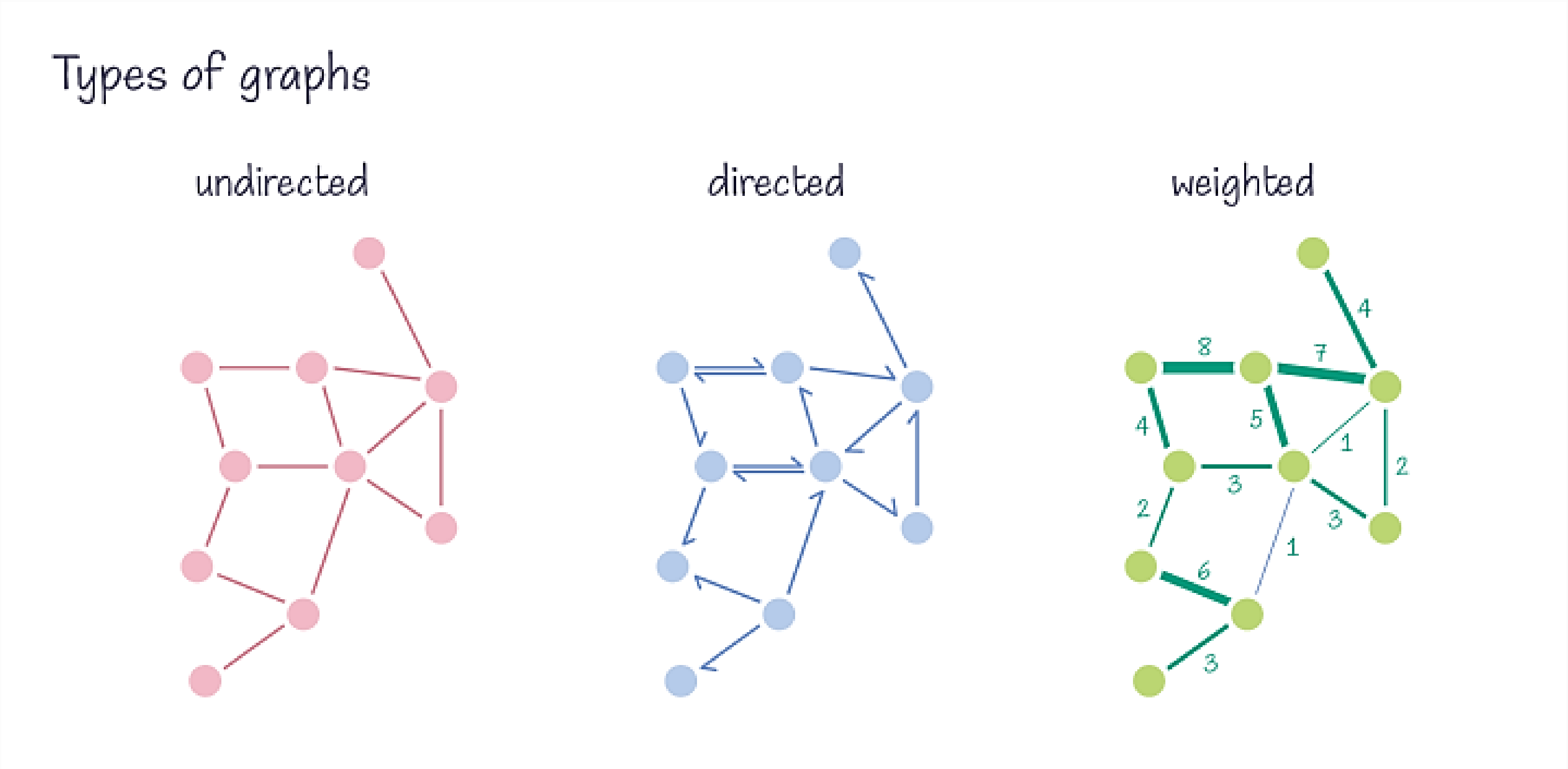
Darwin
Black
Stump
Brisbane
Perth
Adelaide
Canberra
Sydney
Melbourne
Hobart

HOMEWORK



27 min
24.1 miles
1 mile
Fremont
7 miles
Newark
Don Edwards
San Francisco
Bay National
Wildlife...
7 mile
BAYLANDS
Palo Alto
5.4 miles
Palo Alto
22 min
20 miles
2 miles
Great America
NORTH SAN JOSE
Mountain
View
Los Altos
Los Altos Hills
27 min
25.3 miles
Sunnyvale

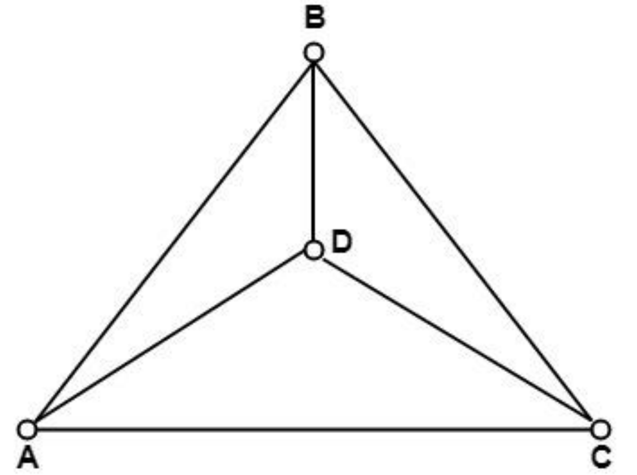
SUMMARY


Types of graphs
undirected
directed
weighted
8
7
4
4
5
1
2
3
2
3
1
6
3

SUMMARY

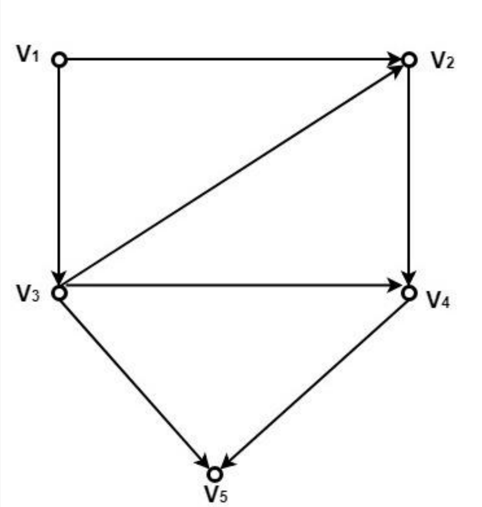
Adjacency Matrix: rows and columns represent vertices.

undirected graph



$$M_A = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 1 & 1 \\ C & 1 & 1 & 0 & 1 \\ D & 1 & 1 & 1 & 0 \end{array}$$

directed graph

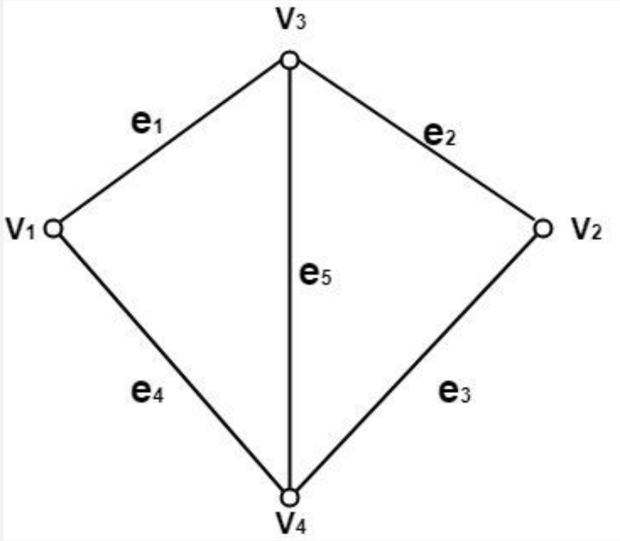


$$M_A = \begin{array}{c|ccccc} & V_1 & V_2 & V_3 & V_4 & V_5 \\ \hline V_1 & 0 & 1 & 1 & 0 & 0 \\ V_2 & 0 & 0 & 0 & 1 & 0 \\ V_3 & 0 & 1 & 0 & 1 & 1 \\ V_4 & 0 & 0 & 0 & 0 & 1 \\ V_5 & 0 & 0 & 0 & 0 & 0 \end{array}$$

SUMMARY

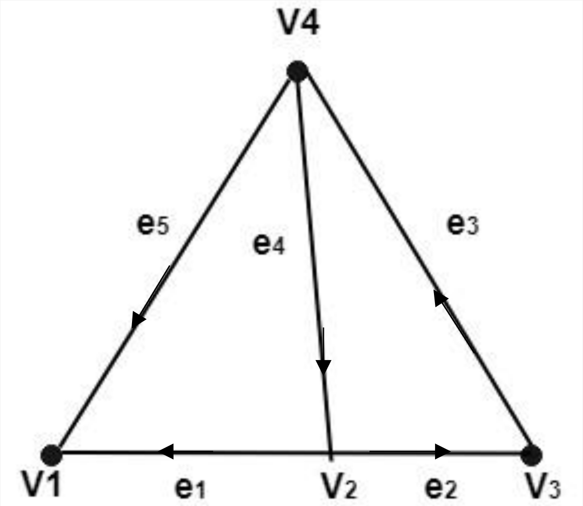
Incidence Matrix: rows represent vertices and columns represent edges.

undirected graph



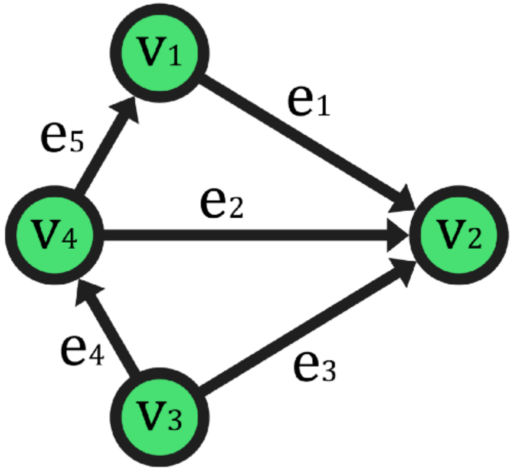
$$M_I = \begin{array}{c|ccccc} & e_1 & e_2 & e_3 & e_4 & e_5 \\ \hline V_1 & 1 & 0 & 0 & 1 & 0 \\ V_2 & 0 & 1 & 1 & 0 & 0 \\ V_3 & 1 & 1 & 0 & 0 & 1 \\ V_4 & 0 & 0 & 1 & 1 & 1 \end{array}$$

directed graph



$$M_I = \begin{array}{c|ccccc} & e_1 & e_2 & e_3 & e_4 & e_5 \\ \hline V_1 & -1 & 0 & 0 & 0 & -1 \\ V_2 & 1 & 1 & 0 & -1 & 0 \\ V_3 & 0 & -1 & 1 & 0 & 0 \\ V_4 & 0 & 0 & -1 & 1 & 1 \end{array}$$

COMPARE



Incidence matrix

$$
\begin{array}{c c}
 & \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\
\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 & -1 \\ -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 1 \end{pmatrix}
\end{array}
$$

Adjacency matrix

$$
\begin{array}{c c}
 & \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\
\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}
\end{array}
$$

READING

- https://sitn.hms.harvard.edu/flash/2021/graph-theory-101/
- https://towardsdatascience.com/what-is-graph-theory-and-why-should-you-care-28d6a715a5c2
- https://www.xomnia.com/post/graph-theory-and-its-uses-with-examples-of-real-life-problems/

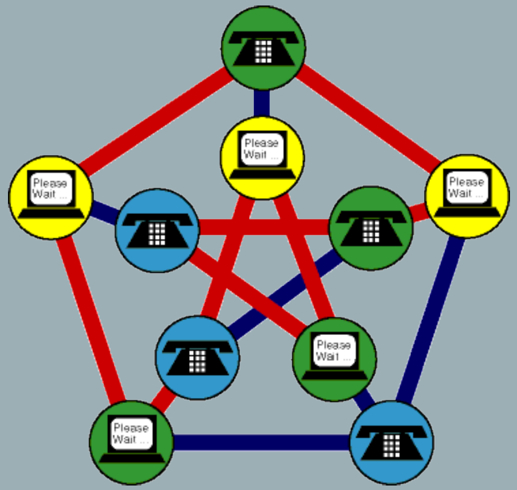
SPECIAL TYPES OF GRAPHS

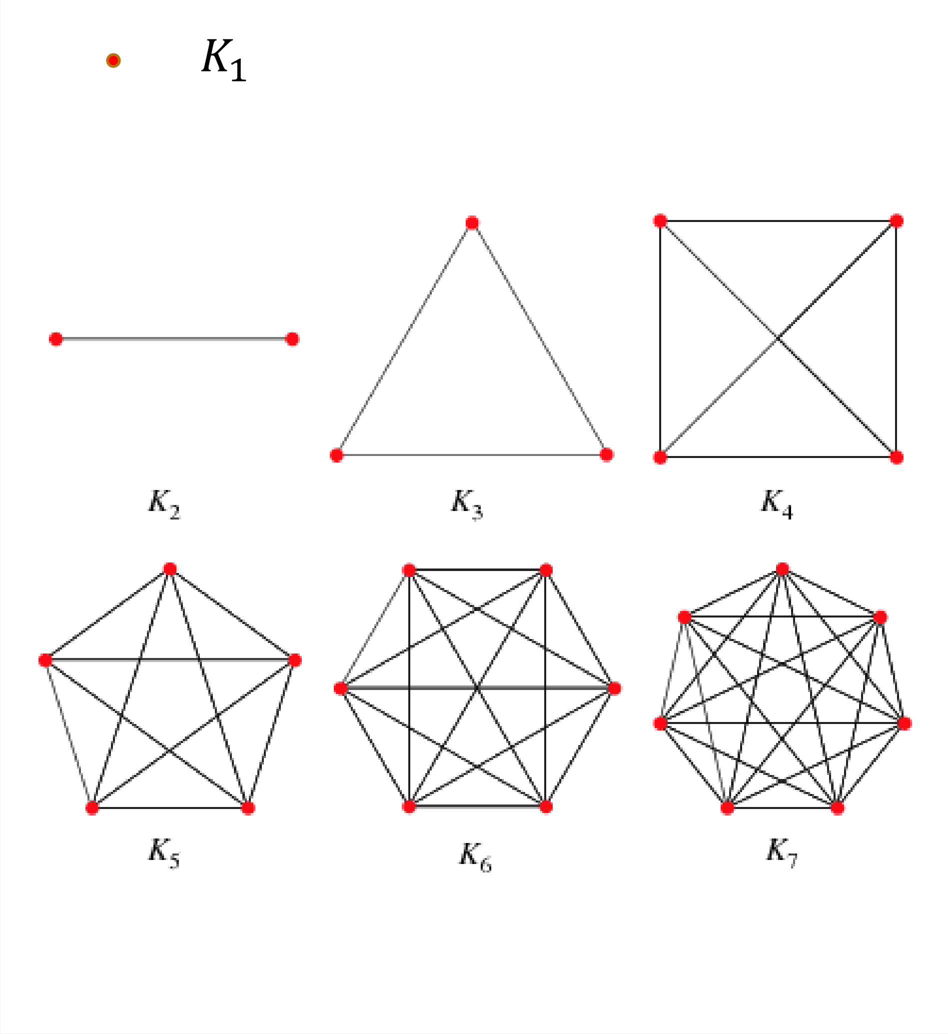
Importance of different graphs in our life:

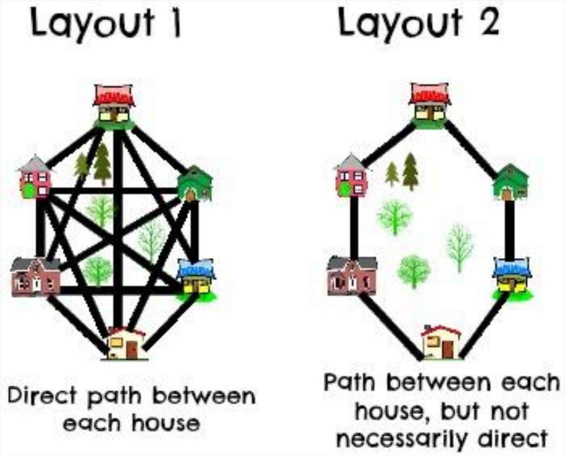
- **Visual presentation of data makes it easier to understand large amounts of data, trends, and relationships**.
- The use of graphs in daily life also helps in **making an analysis**. For example, it provides structure in assessing performances, sales, and even deadlines.
- Also, it helps making **calculations easier**.

COMPLETE GRAPHS

A complete graph, denoted K_n, is a simple graph that contains exactly one edge between each pair of n distinct vertices.



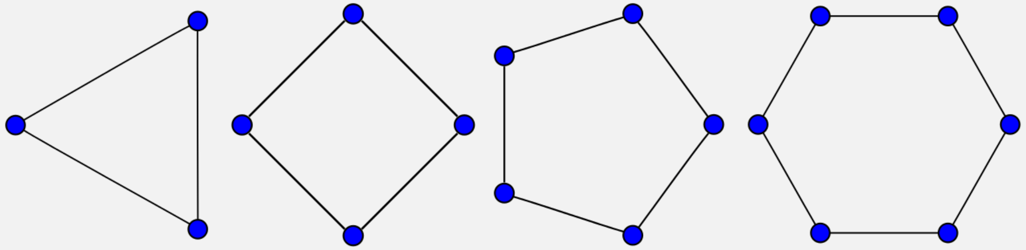


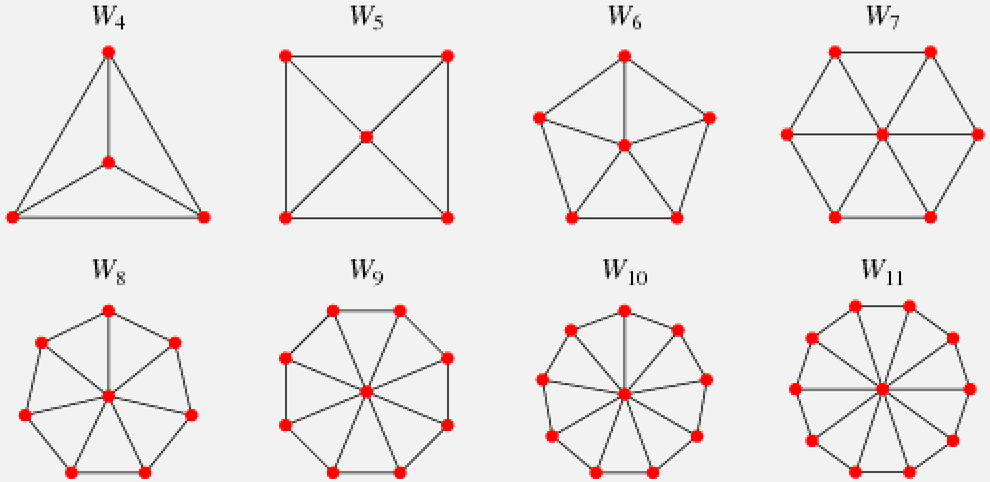


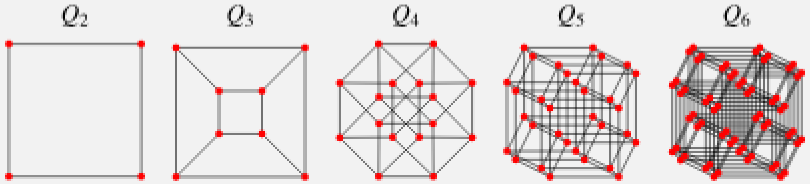
CYCLES AND WHEELS

- A cycle $C_n, n \geq 3$, consists of vertices $v_1, v_2, .., v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, .., \{v_{n-1}, v_n\}, \{v_n, v_1\}$.

- A wheel W_n is obtained when we add an additional vertex to C_n, and connect that vertex to each existing vertex.







N-CUBES OR HYPERCUBES

- An n-dimensional hypercube, or n-cube, denoted Q_n, is a graph with vertices representing the 2^n bit strings of length n. Adjacent vertices differ by exactly one bit-position.

$$Q_0 = 2^0 = 1$$

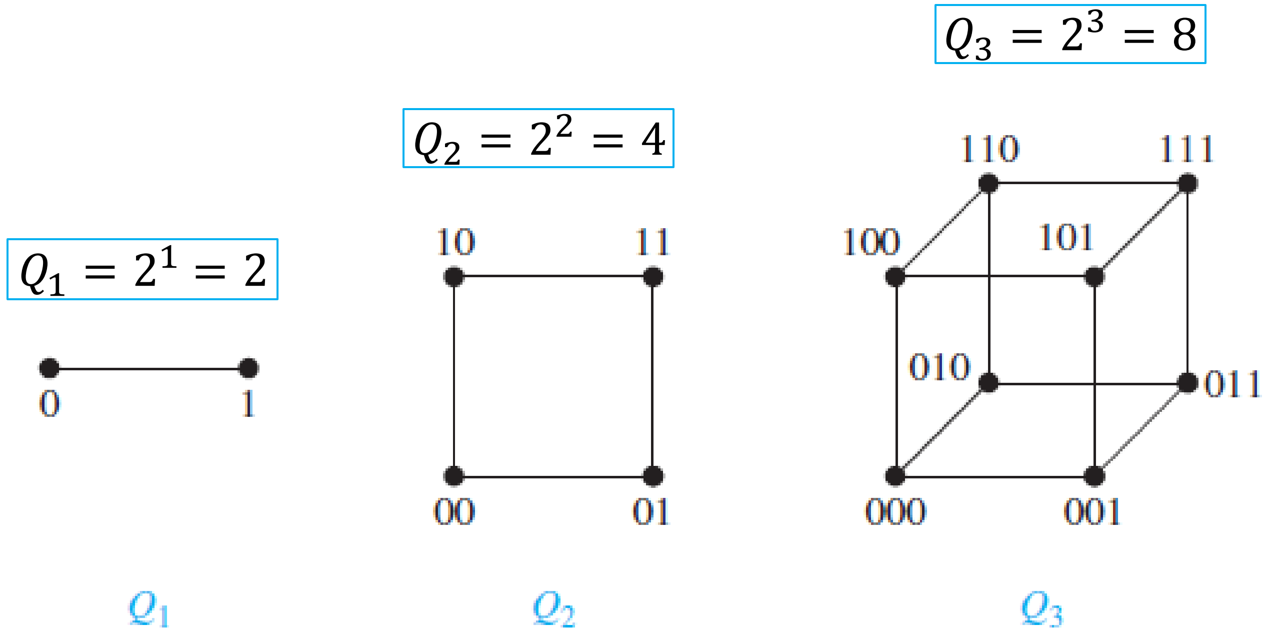
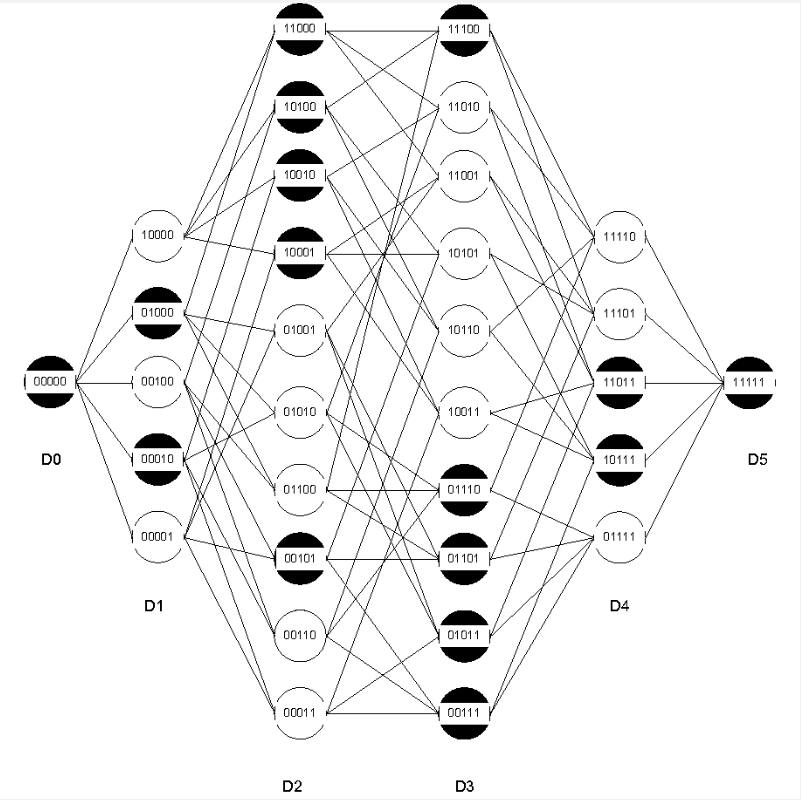


FIGURE 6 **The n-cube Q_n, $n = 1, 2, 3$.**

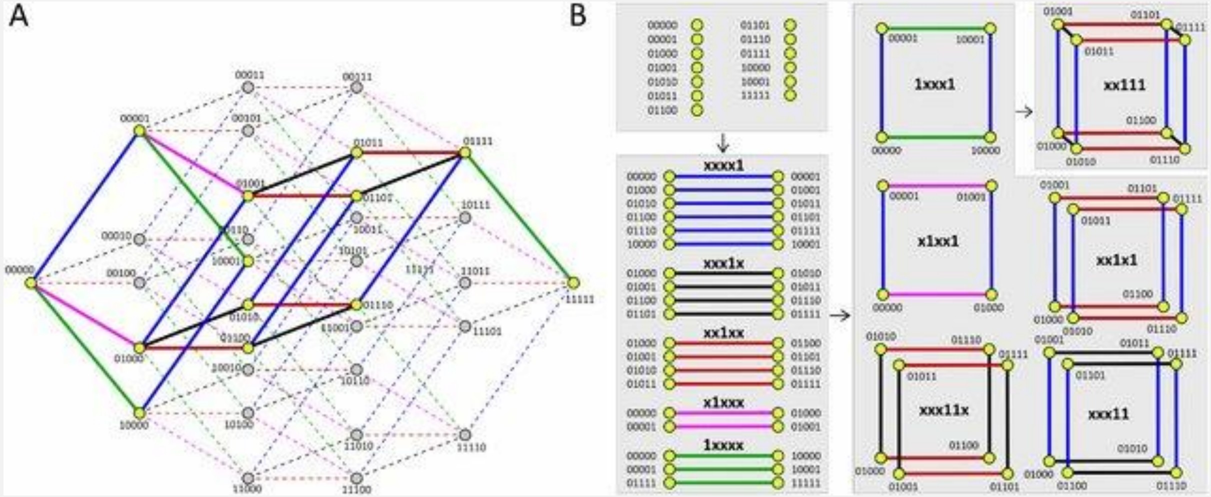
Vertices differ in exactly 1 bit. They get joined by edge.

HYPERCUBES APPLICATIONS

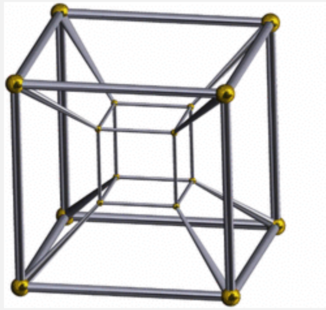
Hypercube Network

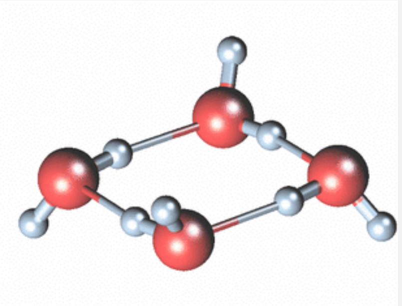


Mutations in Biology



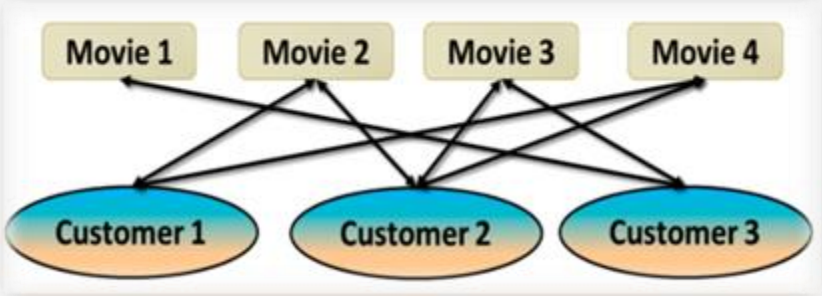
Science

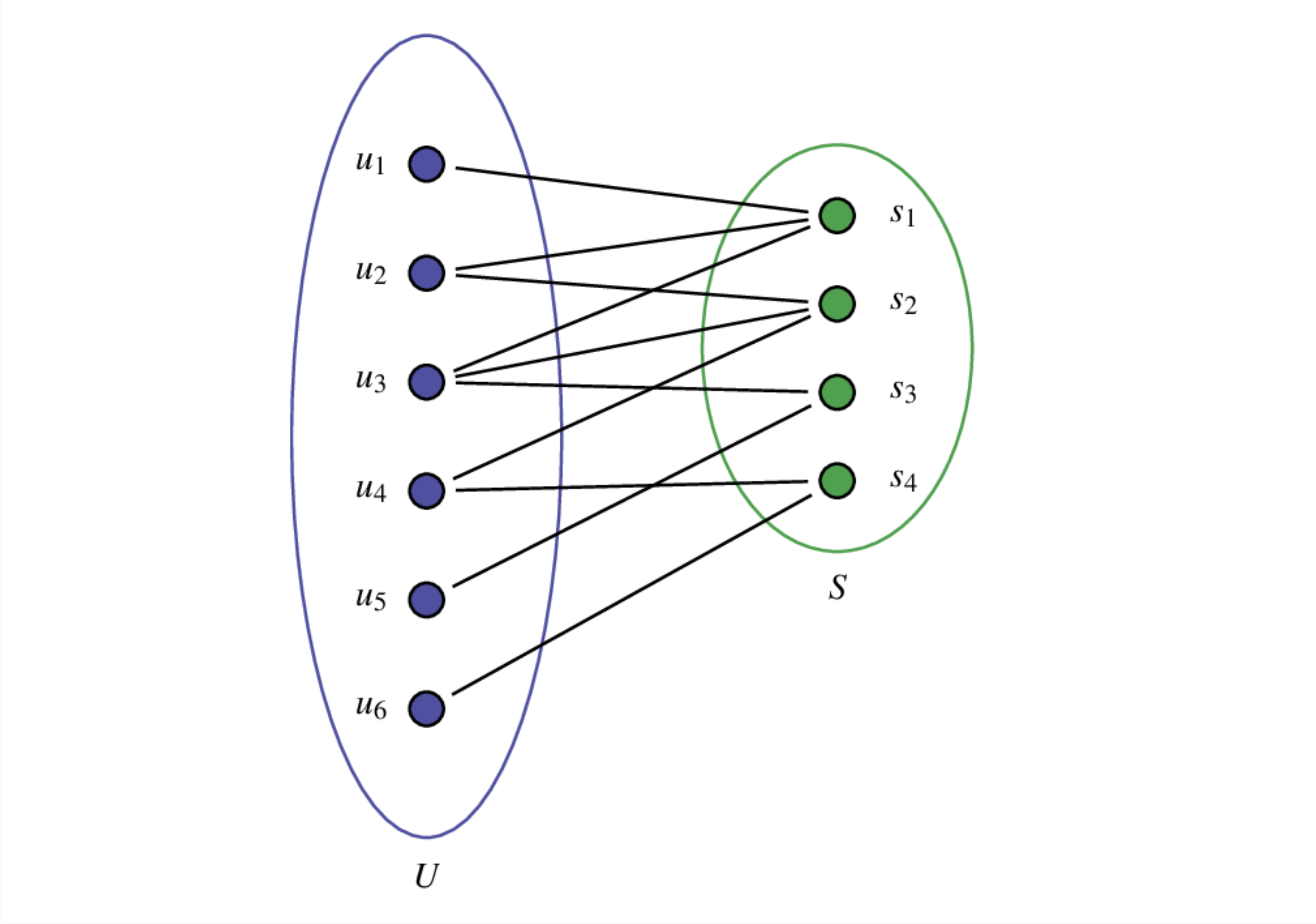




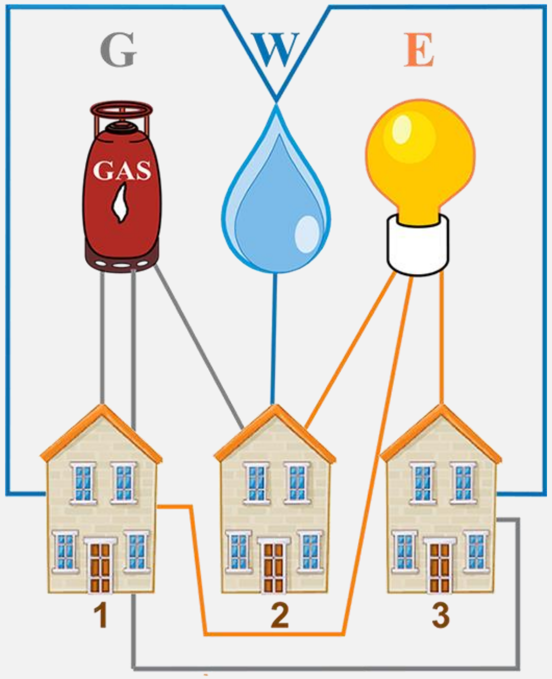
BIPARTITE GRAPHS

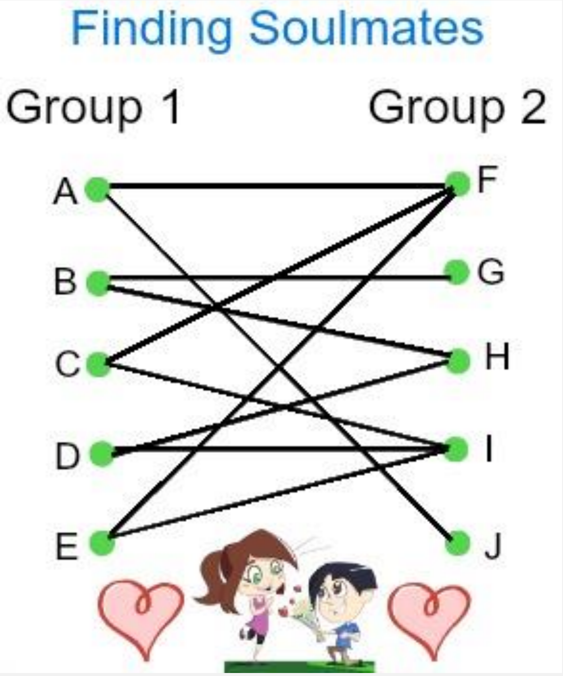
- A simple graph is a called bipartite if its vertex set, V, can be partitioned into two disjoint subsets such that each edge connects a vertex from one subset to a vertex of the other.

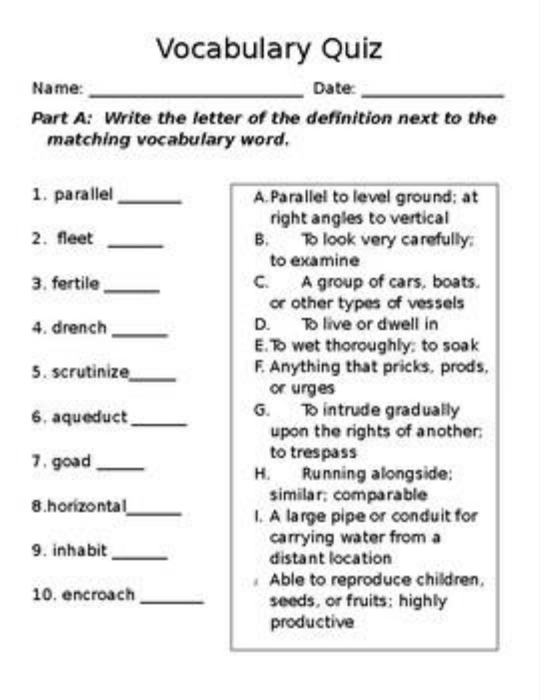




BIPARTITE GRAPH APPLICATIONS





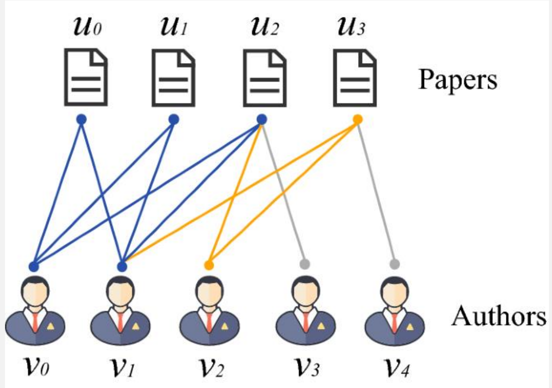

Vocabulary Quiz

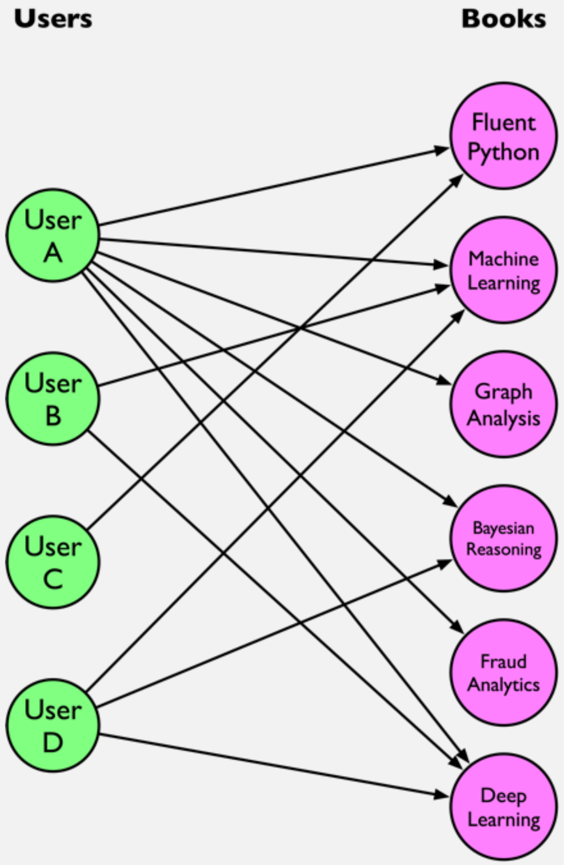
Name: ____________________ Date: ______________

Part A: Write the letter of the definition next to the matching vocabulary word.

1. parallel ______
2. fleet ______
3. fertile ______
4. drench ______
5. scrutinize______
6. aqueduct ______
7. goad ______
8. horizontal______
9. inhabit ______
10. encroach ______

A. Parallel to level ground; at right angles to vertical
B. To look very carefully; to examine
C. A group of cars, boats, or other types of vessels
D. To live or dwell in
E. To wet thoroughly; to soak
F. Anything that pricks, prods, or urges
G. To intrude gradually upon the rights of another; to trespass
H. Running alongside; similar; comparable
I. A large pipe or conduit for carrying water from a distant location
Able to reproduce children, seeds, or fruits; highly productive





GRAPH COLORING

- Determine if graphs G or H are bipartite. Try the graph coloring technique!

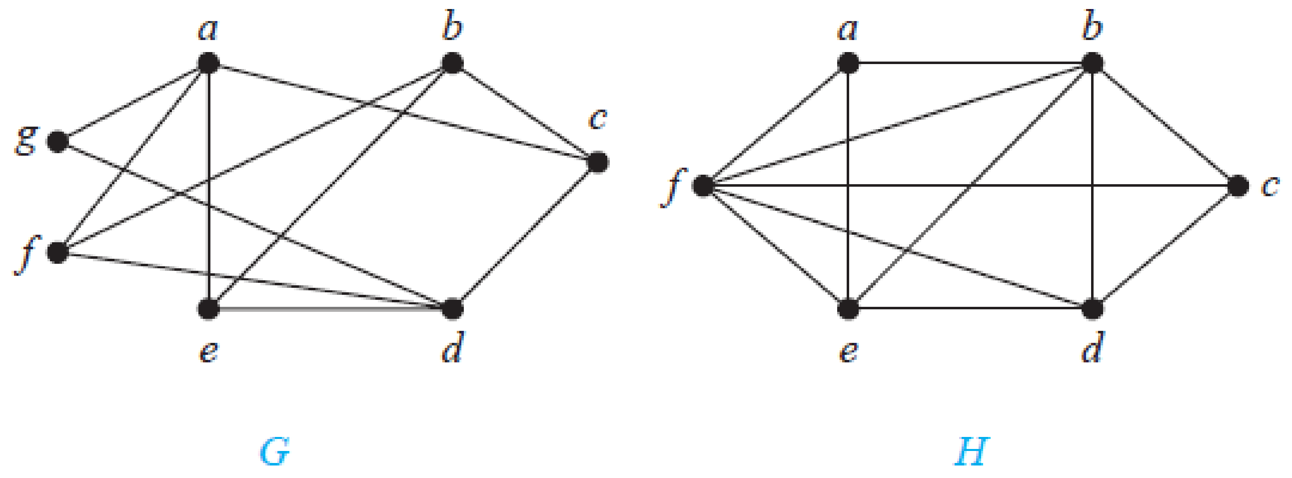
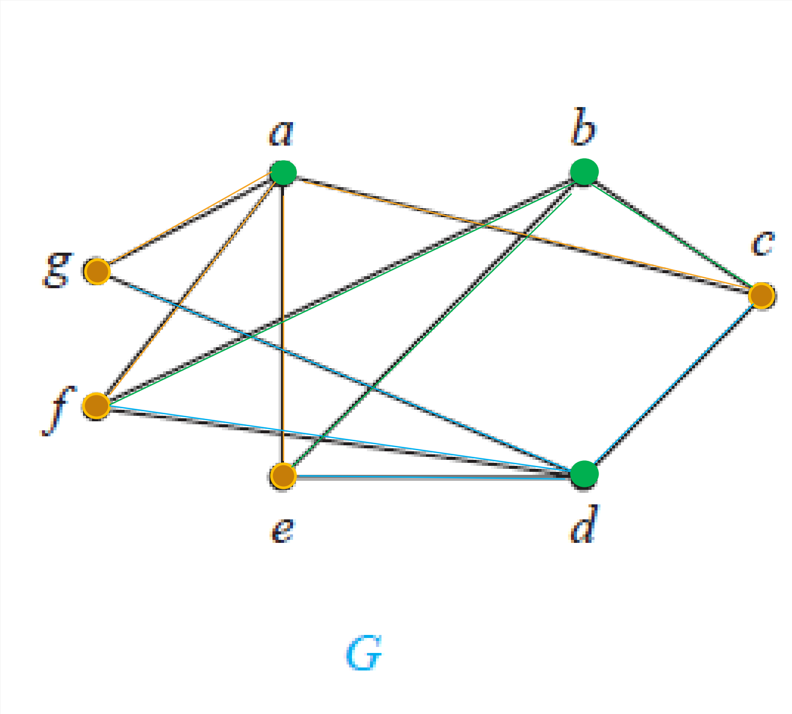
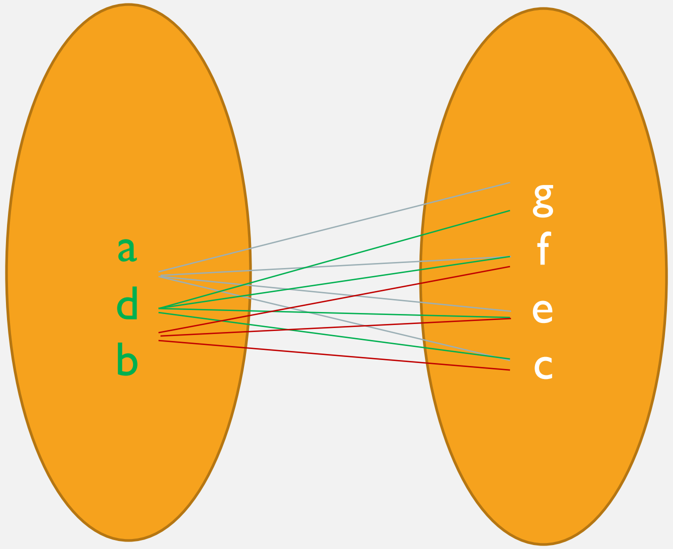


FIGURE 8 **The Undirected Graphs G and H.**

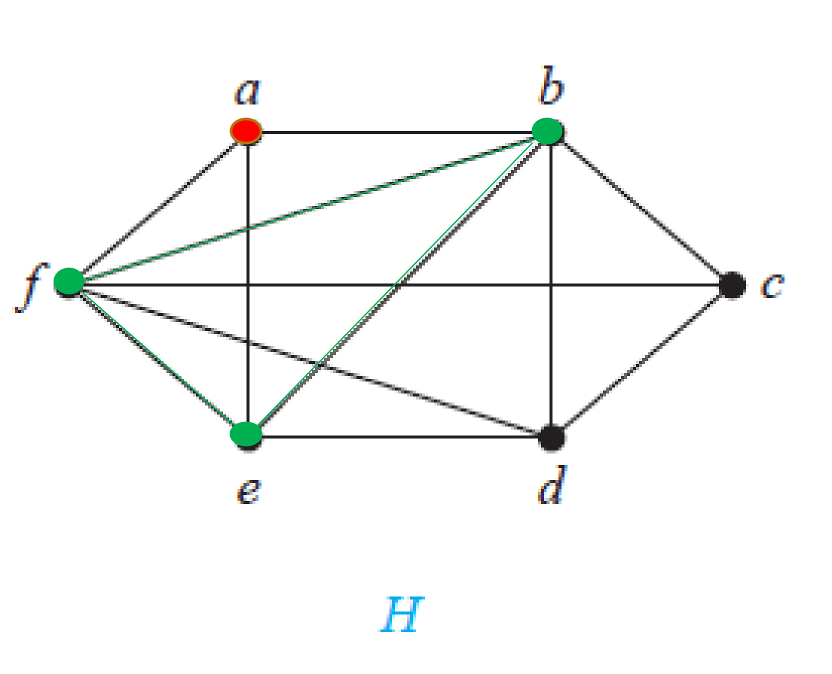
Isomorphism



⟺



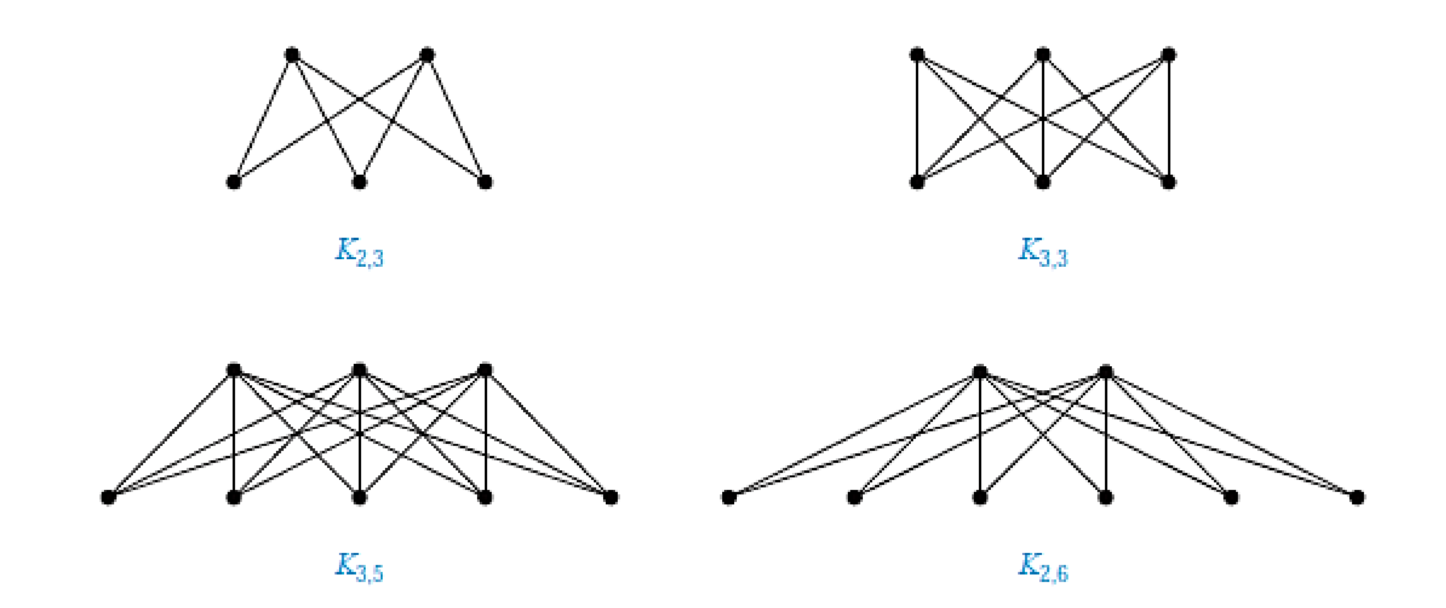
Yes, it is a bipartite!



No need to continue, it is not a bipartite graph!

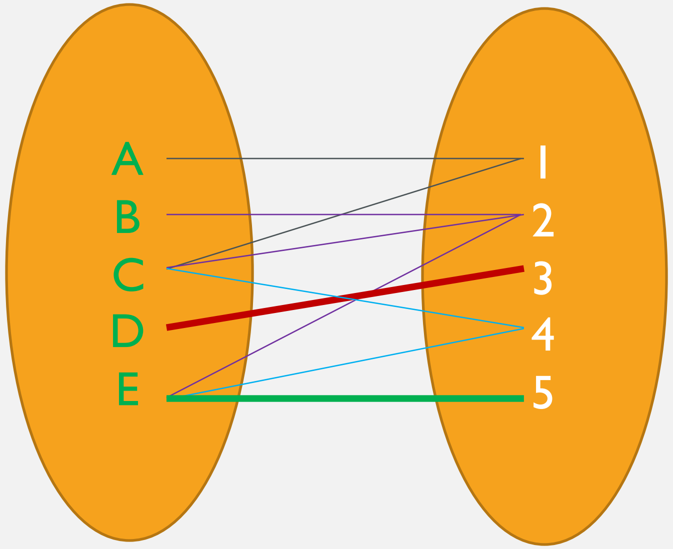
COMPLETE BIPARTITE GRAPHS

- A complete bipartite graph $K_{m,n}$ is a bipartite graph with subsets of m and n vertices, respectively with an edge between each pair of vertices from opposite subsets.



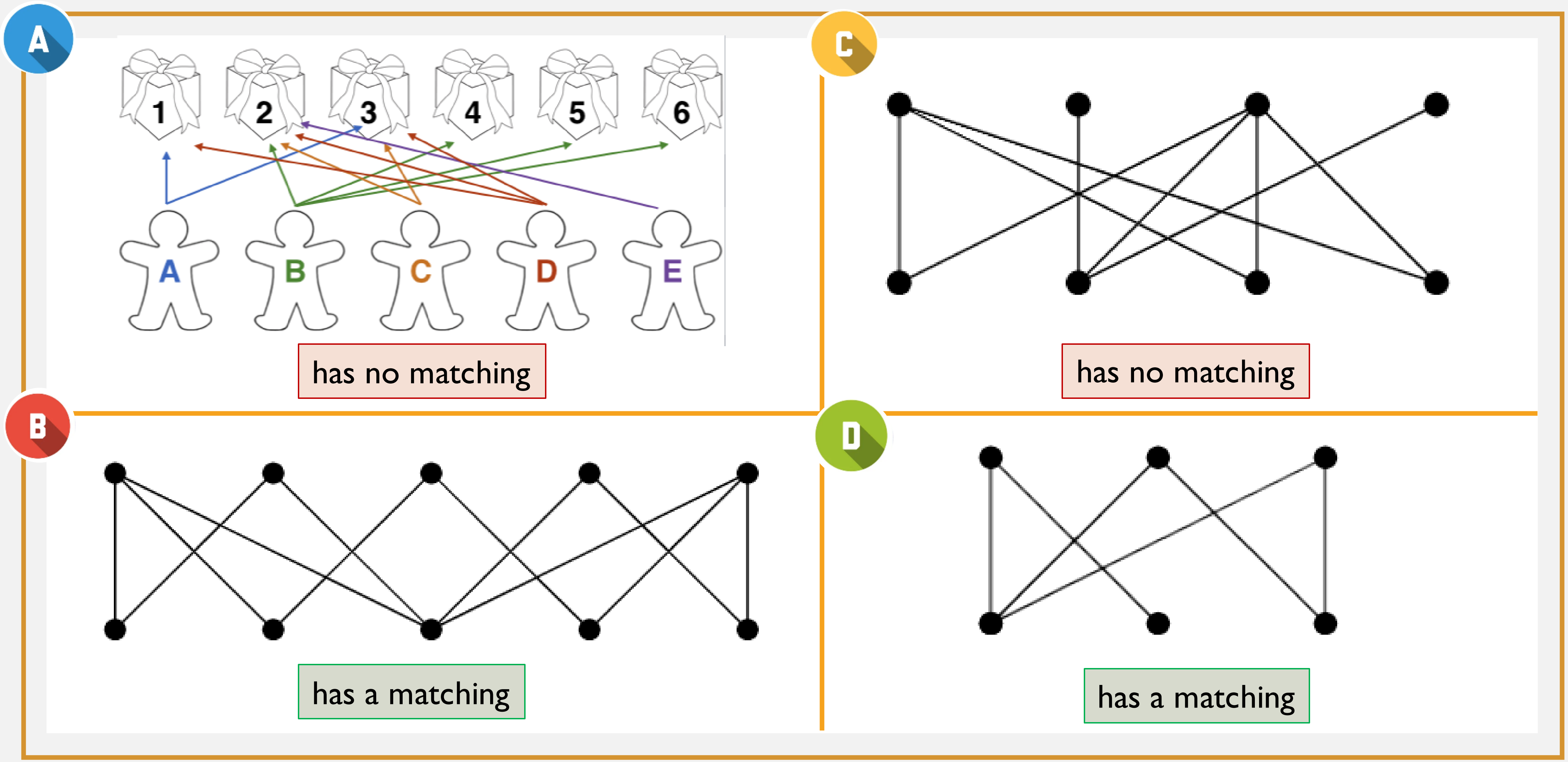
Suppose Adam, Ben, Chris, David and Eric are training for tasks at work. Adam and Chris are training for task 1, Ben, Chris and Eric are training for Task 2, David is training for task 3, Chris and Eric are training for task 4 and Eric is training for task 5. Create a graph to model this, then determine if a **matching** is possible.

graph



matching

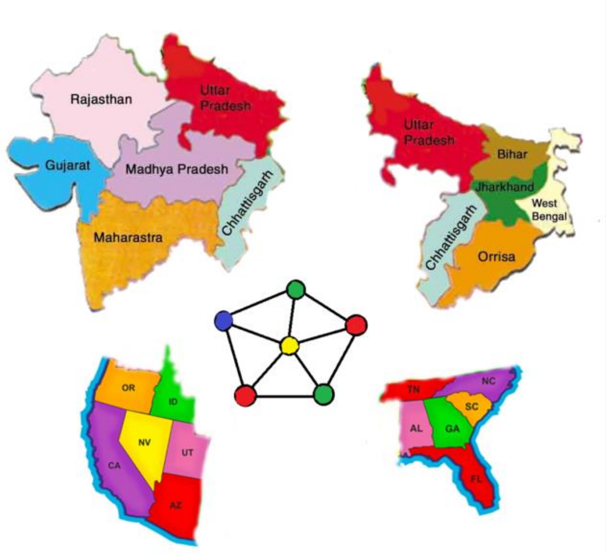
$D - 3$
$E - 5$
$C - 4$
$A - 1$
$B - 2$

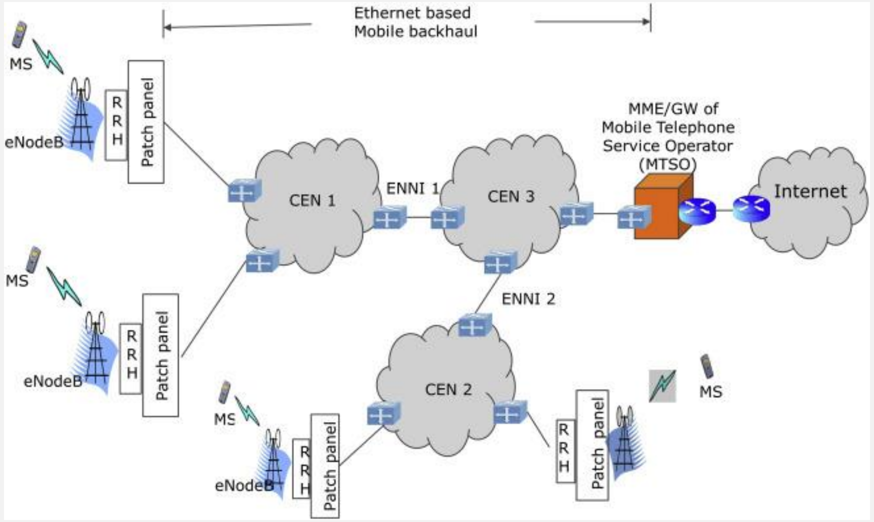

A
1
2
3
4
5
6
A
B
C
D
E
has no matching
C
has no matching
B
has a matching
D
has a matching

Matching

APPLICATIONS OF GRAPH COLORING

- Scheduling problems in management science
- Allocating transmission frequencies to TV and radio stations
- Study of Cell Phone traffic
- Coloring maps so that no two regions that share a boundary are the same color

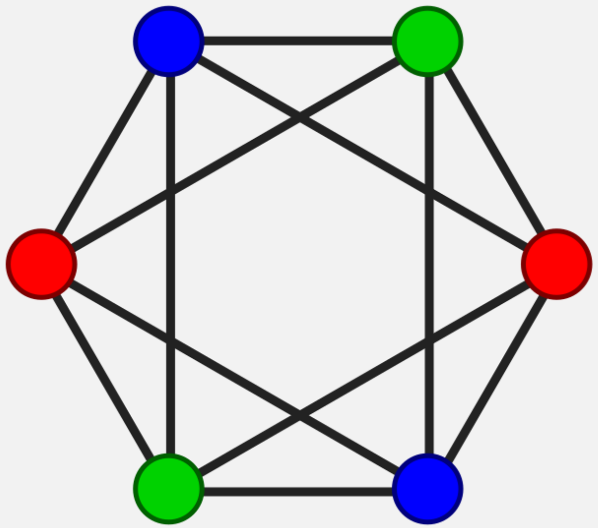




GRAPH COLORING

A coloring for a graph is a coloring of the vertices in such a way that the vertices joined by an edge have different colors.

The chromatic number of a graph is the least number of colors needed to make a coloring.



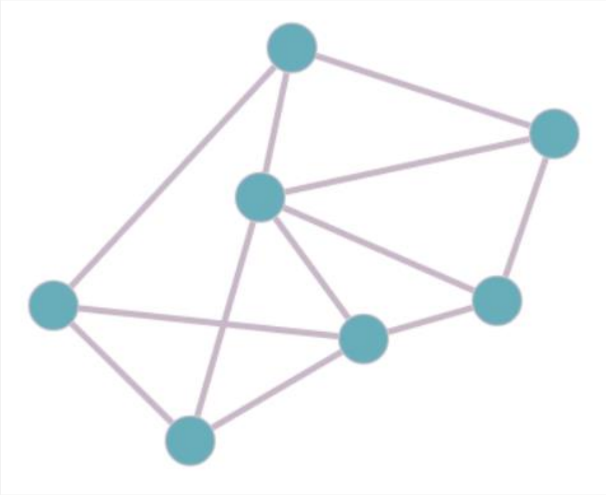
COLORING A GRAPH

Step 1: Choose a vertex with highest degree, and color it. Use the same color to color as many vertices as you can without coloring vertices joined by an edge of the same color.

Step 2: Choose a new color and repeat what you did in Step 1 for vertices not already colored.

Step 3: Repeat Step 1 until all vertices are colored.

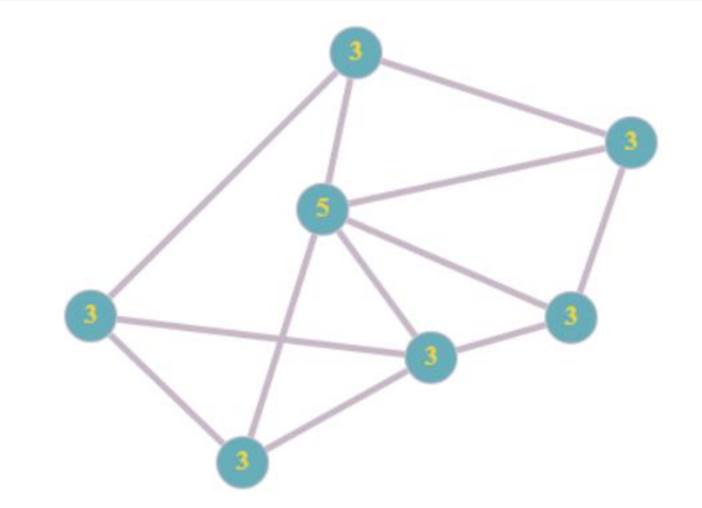
Color the graph and give its chromatic number.

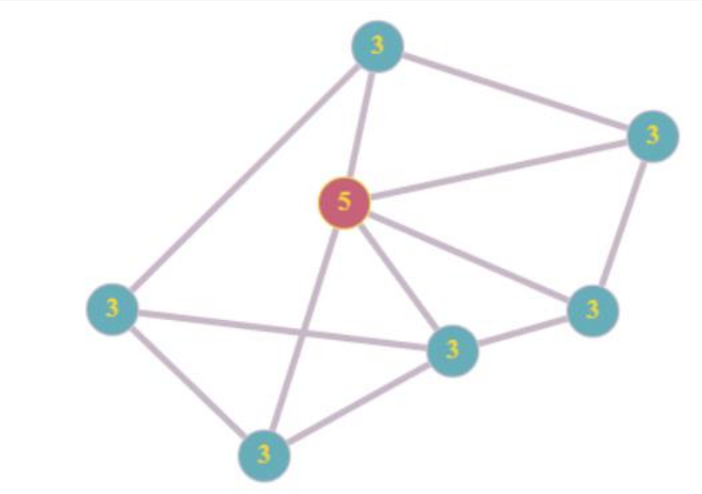


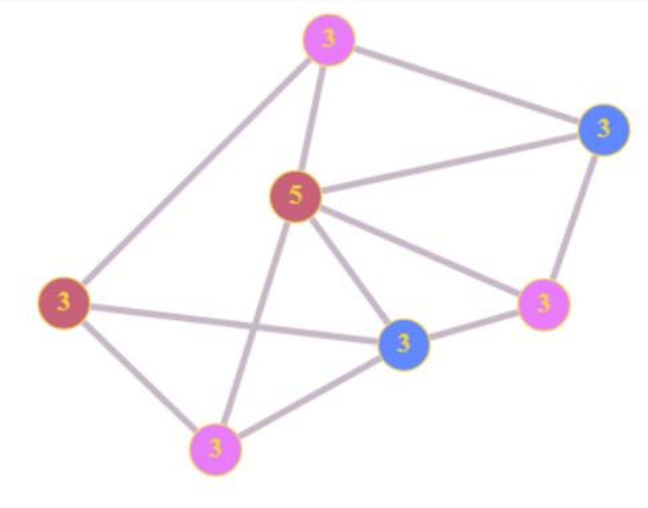
Count degrees

Color the highest degree

Color vertices





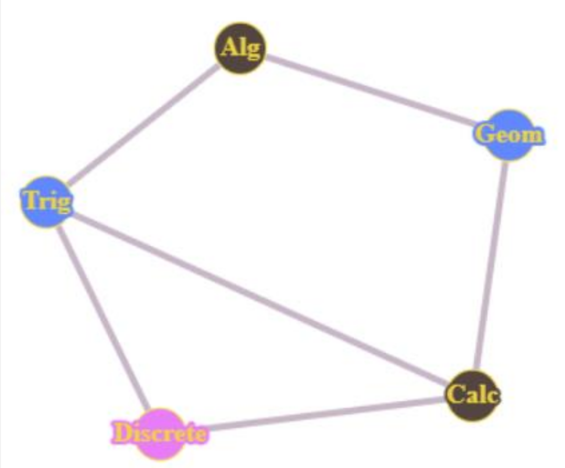


Its chromatic number is **3**

SCHEDULING

- Suppose we have several classes to offer but a limited number of class times. Some classes cannot be offered at the same time because the same instructor teaches the classes. What is the minimum number of classrooms needed for the following?

Course	Conflicts with
Algebra	Geom, Trig
Geometry	Calc, Alg
Calculus	Geom, Tri, Discret
Discrete	Calc, Trig
Trigonometry	Discrete, Calc, Alg



Chromatic number is 3. Which means we need 3 class times to hold these classes.

THREE GOALS OF SCHEDULING PROBLEMS

- Optimization issues – Try to maximize profit, minimize cost.

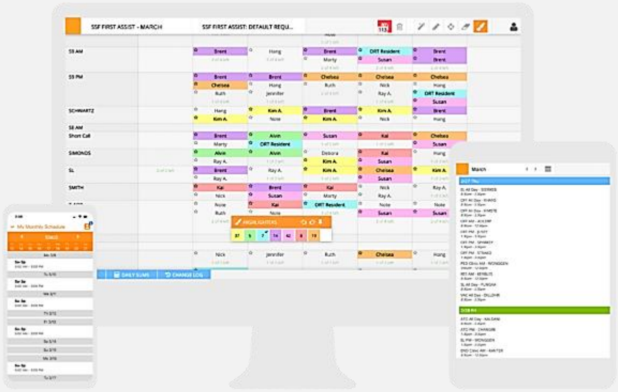
  Example: Scheduling machine time for earliest completion time

- Equity – Try to make things fair for all participants.

  Example: Schedule baseball games (same number home and away games)

- Conflict Resolution – Try to prevent conflicts from happening.

  Example: Scheduling college final examinations for end of term

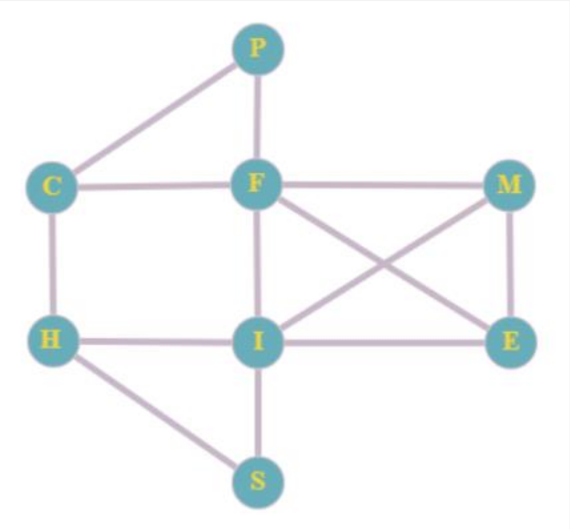


SCHEDULING

- An education center is offering eight courses during its summer session. The table shows with an X which pairs of courses have one or more students in common. Only two air-conditioned lecture halls are available for use at any one time.
- To design an efficient way to schedule the final examinations, represent the information in this table by using a graph. In the graph, represent courses by vertices and join two courses by an edge if there is any student enrolled in both courses.

	F	M	H	P	E	I	S	C
French (F)		X		X	X	X		X
Math (M)	X				X	X		
History (H)						X	X	X
Philosophy (P)	X							X
English (E)	X	X				X		
Italian (I)	X	X	X		X		X	
Spanish (S)			X			X		
Chemistry (C)	X		X	X				

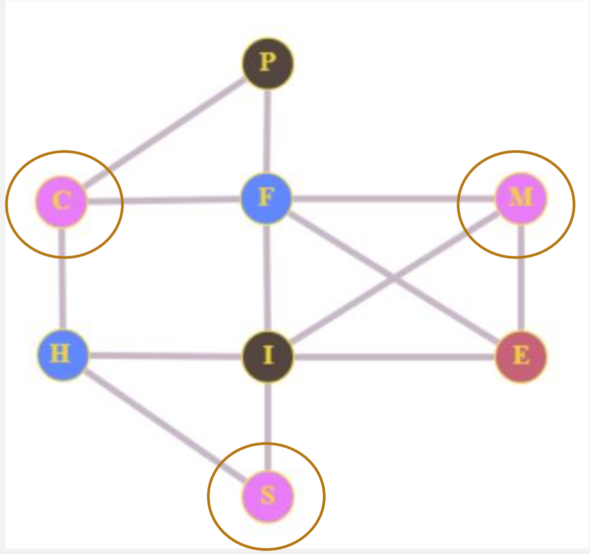
	F	M	H	P	E	I	S	C
French (F)		X		X	X	X		X
Math (M)	X				X	X		
History (H)						X	X	X
Philosophy (P)	X							X
English (E)	X	X				X		
Italian (I)	X	X	X		X		X	
Spanish (S)			X			X		
Chemistry (C)	X		X	X				



We see that $x = 4$.
This means that four times are needed.
Time Slot 1: F, H
Time Slot 2: P, I
Time Slot 3: C, M, S
Time Slot 4: E

What if only two rooms have air-conditioning?

Time Slot 1: F, H
Time Slot 2: P, I
Time Slot 3: C, M
Time Slot 4: E, **S**



EXAMPLE

Corncob College elects 10 students to serve as officers on 8 committees. The list of the members of each of the committees is:

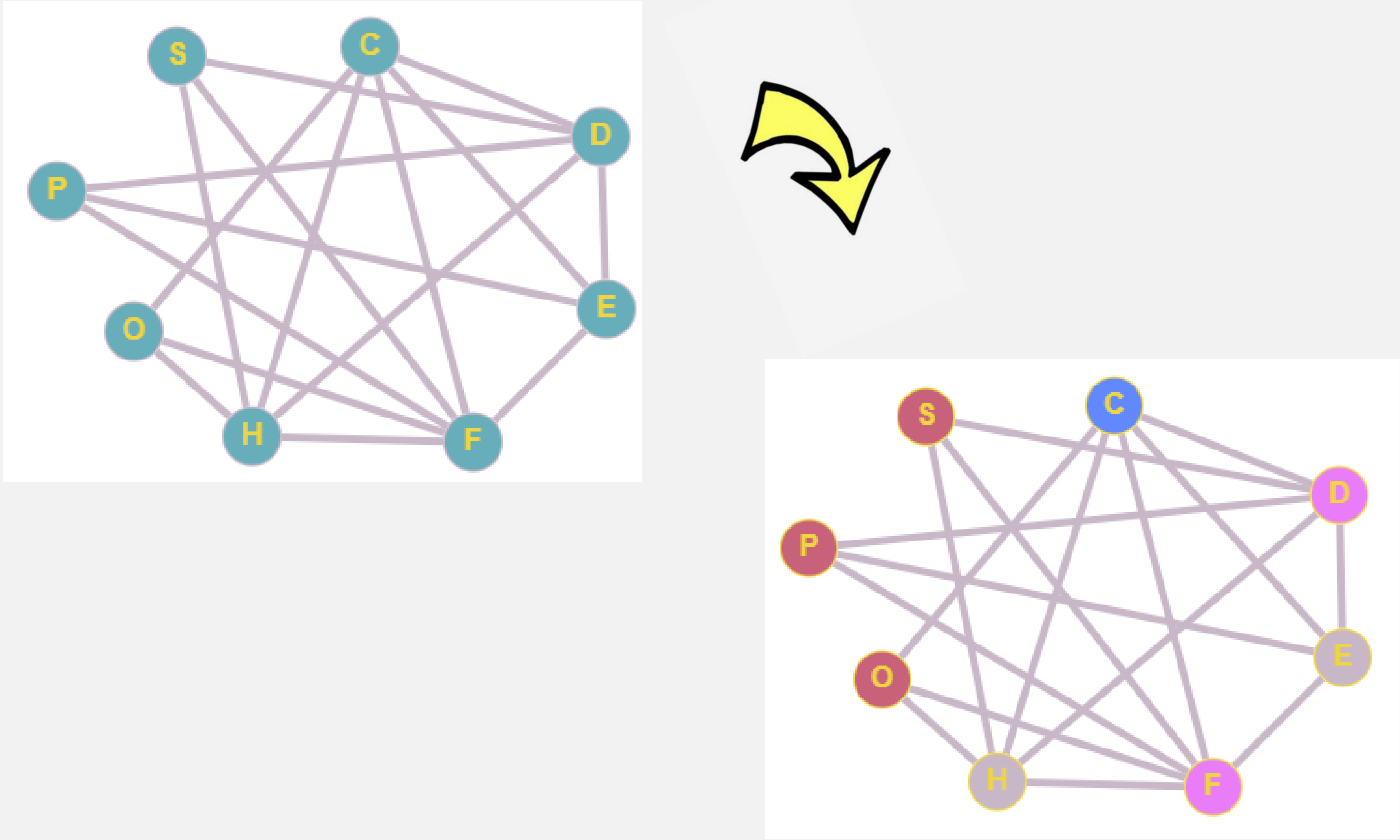
- Corn Feed Committee: *Darcie, Barb, Kyler*
- Dorm Policy Committee: *Barb, Jack, Anya, Kaz*
- Extracurricular Committee: *Darcie, Jack, Miranda*
- Family Weekend Committee: *Kyler, Miranda, Jenna, Natalie*
- Homecoming Committee: *Barb, Jenna, Natalie, Skye*
- Off Campus Committee: *Kyler, Jenna, Skye*
- Parking Committee: *Jack, Anya, Miranda*
- Student Fees Committee: *Kaz, Natalie*

They need to schedule meetings for each of these committees, but two committees cannot meet at the same time if they have any members in common.

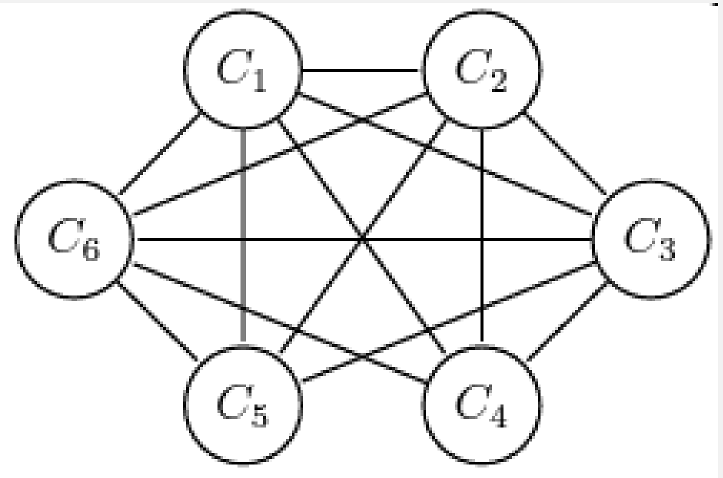
a) Draw a graph representing this situation. (Hint: let the vertices represent the committees.)

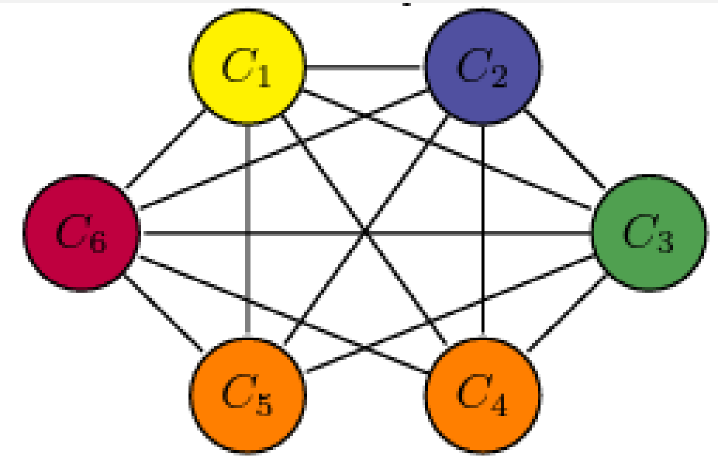
b) How many different meeting times will we need?


Solution


S
C
D
P
E
O
H
F
S
C
D
P
E
O
H
F

- The mathematics department has six committees, each meeting once a month. How many different meeting times must be used to ensure that no member is scheduled to attend two meetings at the same time if the committees are
- C1 = {Arlinghaus, Brand, Zaslavsky},
- C2 = {Brand, Lee, Rosen},
- C3 ={Arlinghaus, Rosen, Zaslavsky},
- C4 = {Lee, Rosen, Zaslavsky},
- C5 = {Arlinghaus, Brand},
- C6 = {Brand, Rosen, Zaslavsky}?



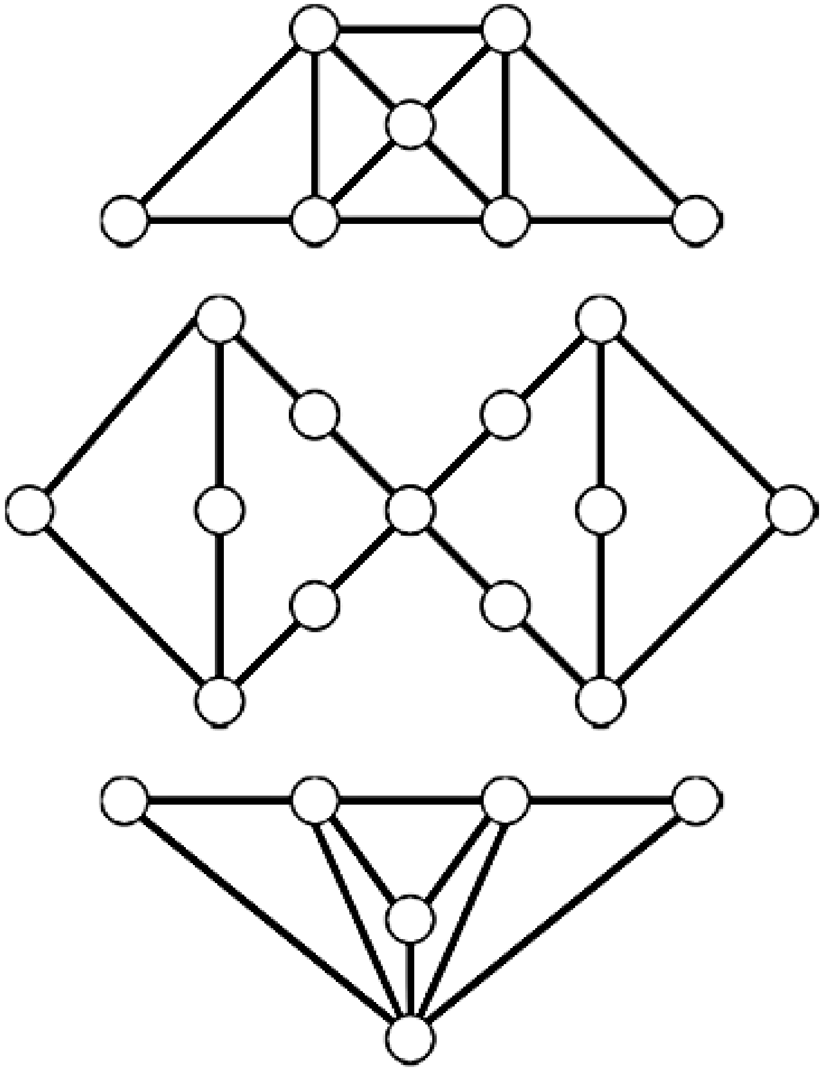


5 meeting times are needed

GRAPHING ONLINE TOOL

- https://graphonline.ru/en/#
- https://mathigon.org/course/graph-theory/introduction

HOMEWORK



HOMEWORK

The mathematics department at Cornucopia College will offer seven courses next semester: Math 105 (M), Numerical Analysis (N), Linear Operators (O), Probability (P), Differential Equations (Q), Real Analysis (R), Statistics (S). The department has twelve students, who will take the following classes:

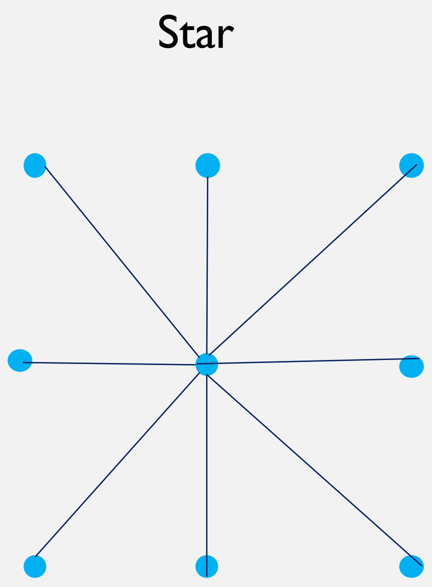
Alice: N, O, Q
Bob: N, R, S
Greg: M,P
Homer: R,O
Inez: N, Q
Fonz: N, R

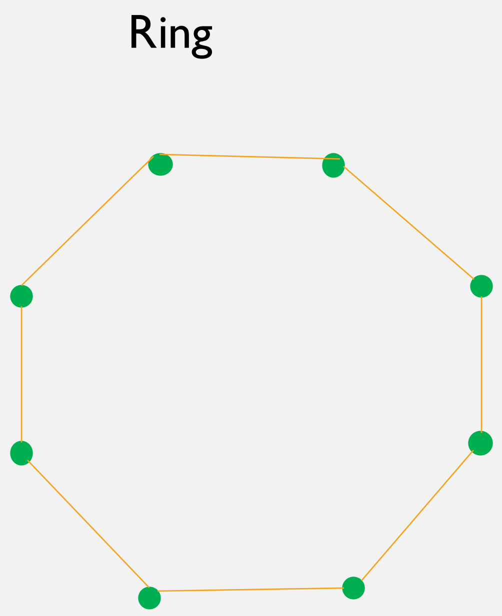
Chaz : R, M
Dan: N,O
Emma: O, M
Kate: P,S
Lara: P,Q
Jill: N,S,Q

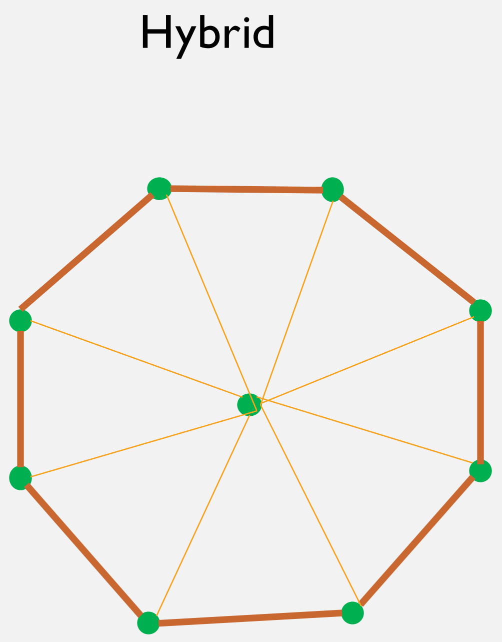
The department needs to schedule class times for each of these courses, but two courses cannot meet at the same time if they have any students in common. How many different class times will they need?

APPLICATIONS OF GRAPHS

Suppose 8 devices (computers, printers, etc.) must be connected through a local area network. Let's explore how this might look.



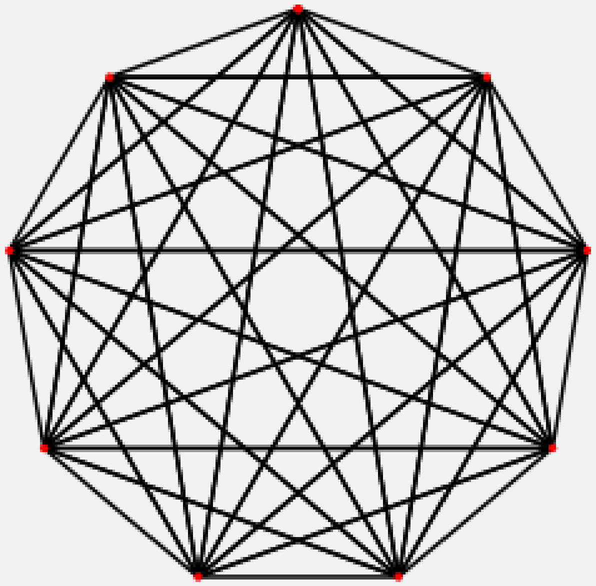




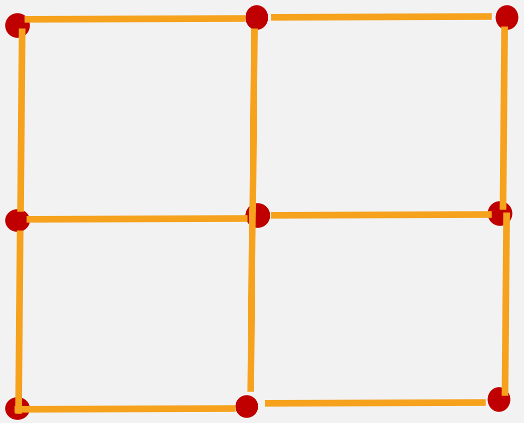
Suppose you have 9 processors to carry out an algorithm. Describe three different ways to arrange the processors and the advantage or disadvantage of each.

Linear Array

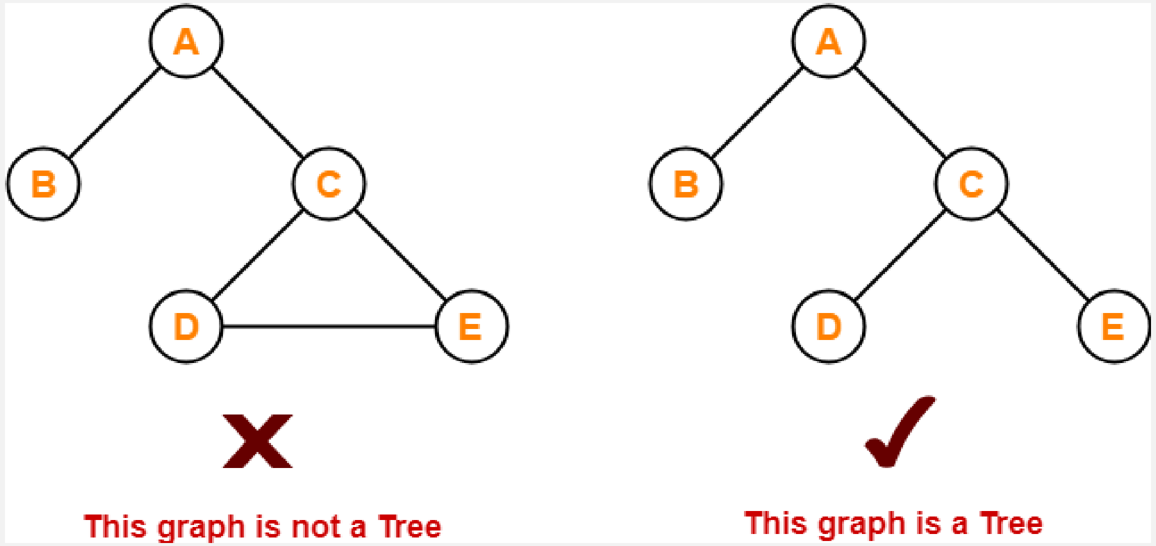
Complete

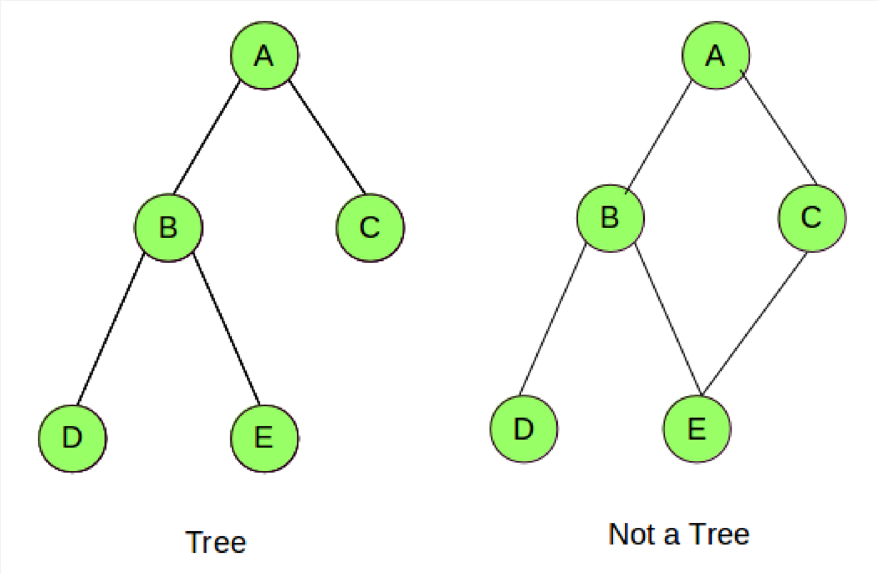


Mesh Network



TREES

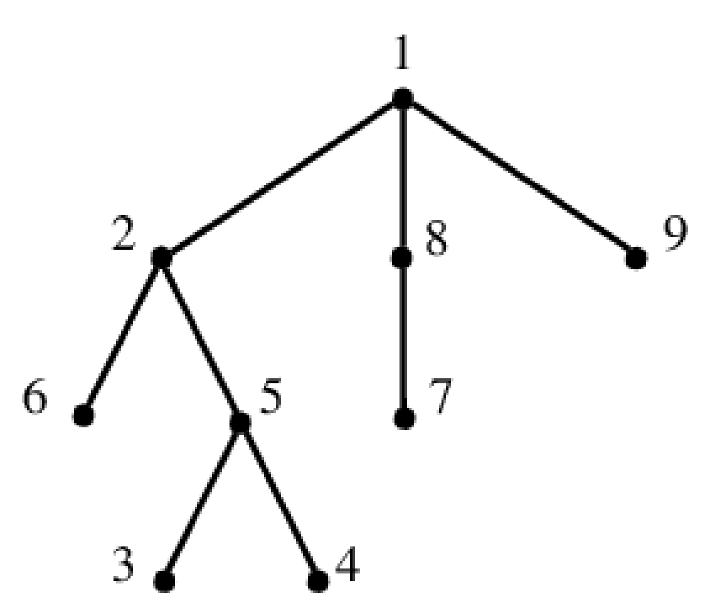




- A tree is a simple, connected, undirected graph with no simple circuits. This means there is a unique simple path between any two of its vertices.

ROOTED TREES

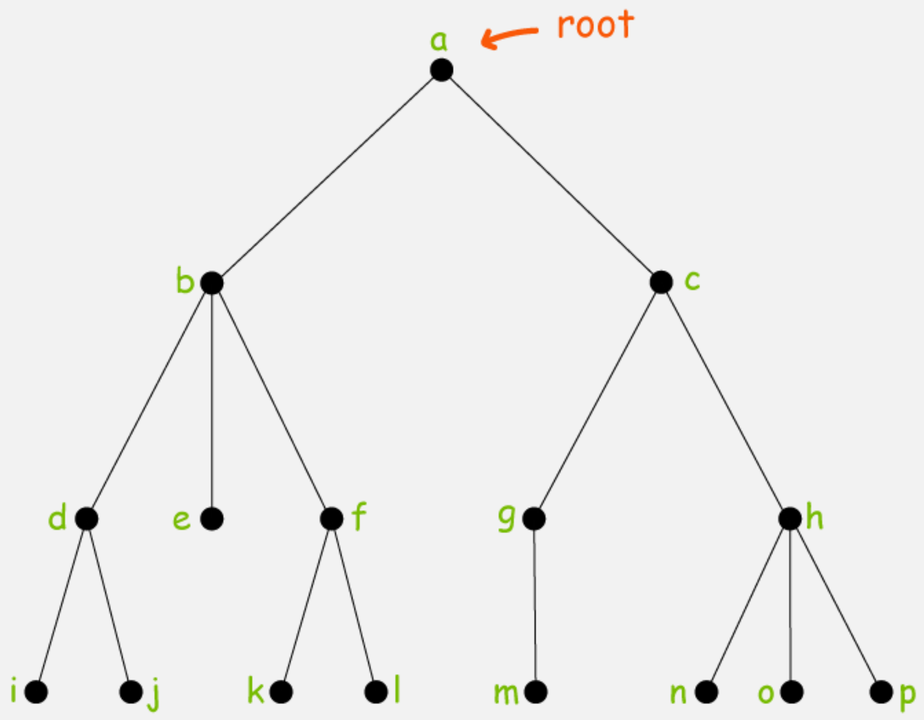
- A tree in which one vertex has been designated as the root and every edge is directed away from the root. We typically place the root at the top of the tree.



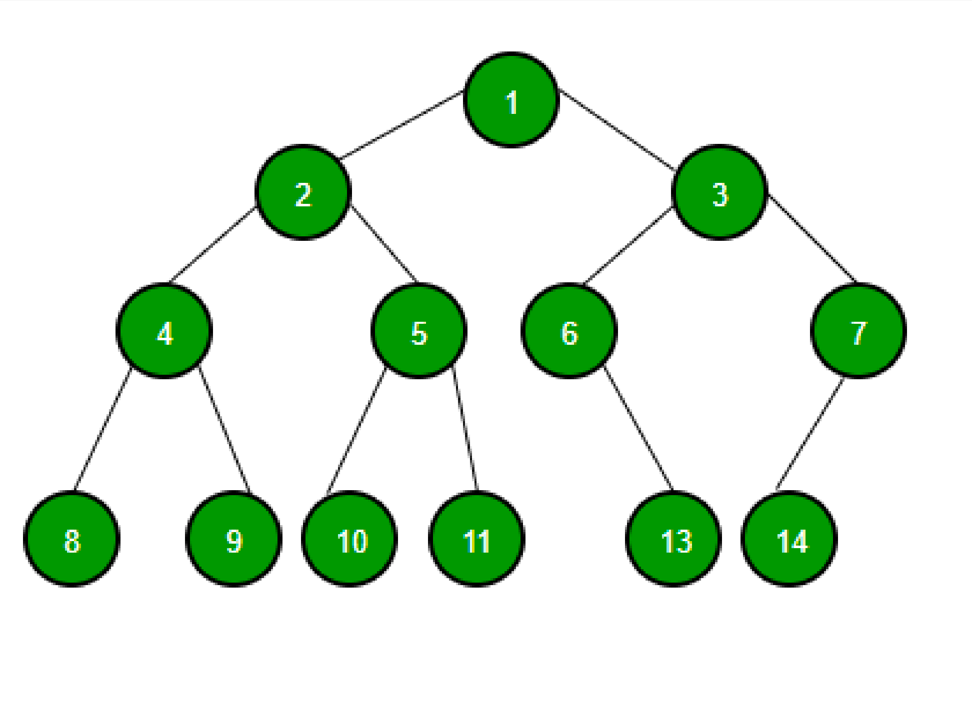
Create a rooted tree from **5**.

MORE TERMINOLOGY FOR ROOTED TREES

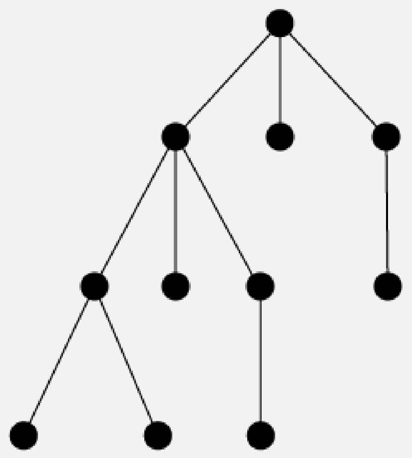
- m-ary tree
- binary tree:
- parent: a
- child: b, c
- sibling: b and c
- ancestor: a
- descendants: b - p
- leaf: no children
- internal vertices: has children



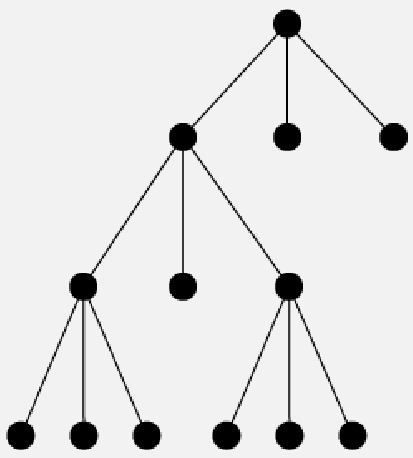
BINARY TREE



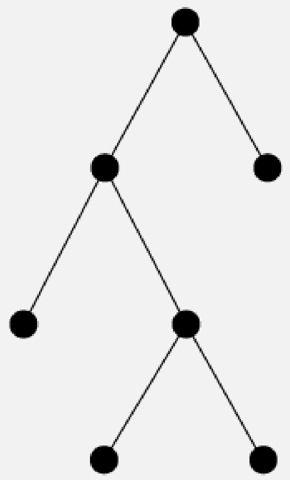
M-ARY TREE



3-ary tree
(each internal vertex has
no more than 3 children)



Full 3-ary tree
(each internal vertex
has exactly 3 children)

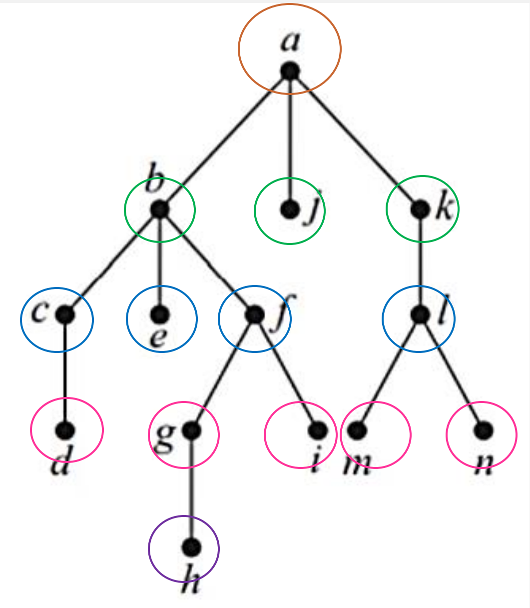


Full Binary tree
(each internal node
has exactly 2 children)

BALANCED M-ARY TREES

- The **height** of a rooted tree is the maximum of the levels of vertices.
- A rooted m-ary tree of height h is **balanced** if all leaves are at levels h or $h-1$.

Example: Find the level of each vertex in the following rooted tree. What is the height of this tree?



Level:0

Level:1

Level:2

Level:3

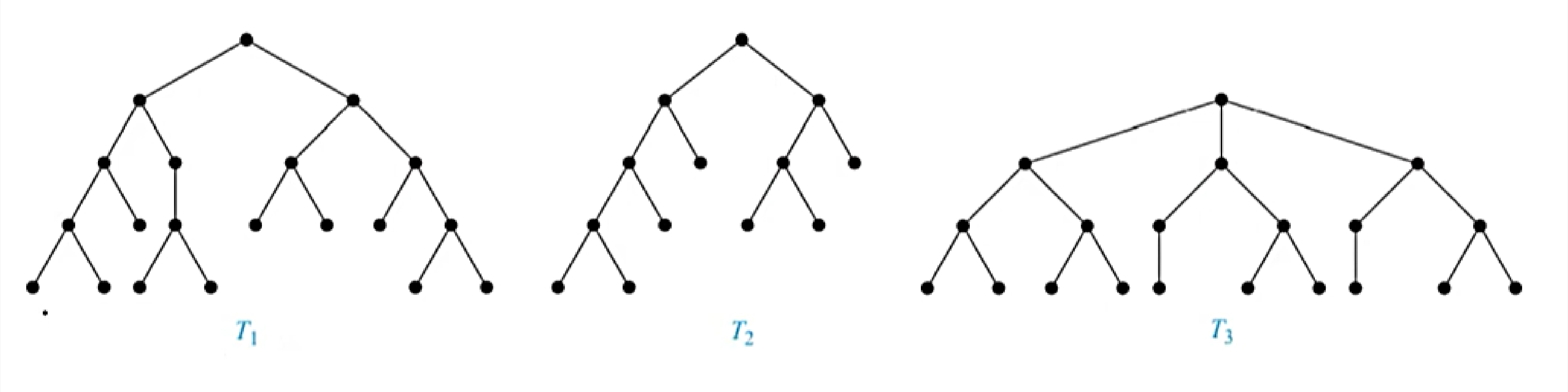
Level:4

$h = 4$

Not balanced: leaves at $h-2$ also exist

BALANCED M-ARY TREES

Balanced | Not Balanced | Balanced



<u>Balancing</u> the tree makes **for better search times** $O(\log(n))$ as opposed to $O(n)$.

An <u>unbalanced</u> tree is, in its worst case, just a linked list and the worst case to find an element becomes $O(n)$, instead of $O(logn)$.

A balanced tree avoids this worst case, and ones that are nearly as bad.

PROPERTIES OF TREES

- A tree with n vertices has $n-1$ edges
- A full $m-ary$ tree with i internal vertices contains $n = mi + 1$ vertices
- A full $m-ary$ tree with:

i. n vertices has $i = \frac{n-1}{m}$ internal vertices and $l = \frac{(m-1)n+1}{m}$ leaves

ii. i internal vertices has $n = mi + 1$ vertices and $l = (m-1)i + 1$ leaves

iii. l leaves has $n = \frac{(ml-1)}{m-1}$ vertices and $i = \frac{l-1}{m-1}$ internal vertices

Suppose someone starts a chain letter. Each person is asked to send the letter to four other people. Everyone does it. How many people have seen the letter, including the original sender, if no one receives more than one letter, and the chain ends after 100 people read it but did not send it on. **How many people sent out the letter?**

It is a **tree**, because no one receives more than one letter.

leaves =100

$m = 4$ (full m-ary tree)

How many people have seen the letter: $n = ?$

$$n = \frac{ml - 1}{m - 1}$$

$$n = \frac{4(100) - 1}{4 - 1} = \frac{399}{3} = 133$$

How many people sent out the letter? $i = ?$

internal vertices

$$i = \frac{l - 1}{m - 1}$$

$$i = \frac{100 - 1}{4 - 1} = \frac{99}{3} = 33$$

A chain letter starts when a person sends a letter to five others. Each person who receives the letter either sends it to five other people who have never received it or does not send it to anyone. Suppose that 10,000 people send out the letter before the chain ends and that no one receives more than one letter. **How many people receive the letter, and how many do not send it out?**

i internal vertices has $n = mi + 1$ vertices and $l = (m-1)i + 1$ leaves

$m = 5$ (full 5-ary tree)

$i = 10000$

How many people receive the letter: $n = ?$

$$n = 5 \cdot 10000 + 1 = 50001$$

How many do not send it out: $l = ?$

$$l = (5-1) \cdot 10000 + 1 = 40001$$

❖ How many matches are played in a tennis tournament of 27 players?

$\boxed{26 \ matches}$

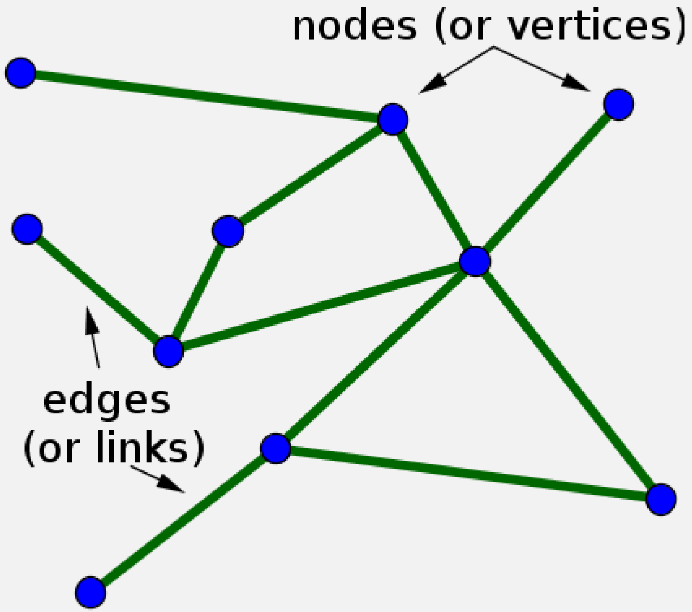
❖ There are 256 players in a chess tournament (singles). Two players play a match. Matches are played on a knockout basis, the loser is eliminated after each match. How many matches need to be played to declare a winner? There is no draw.

$\boxed{255 \ matches}$

In both questions, idea is the same:

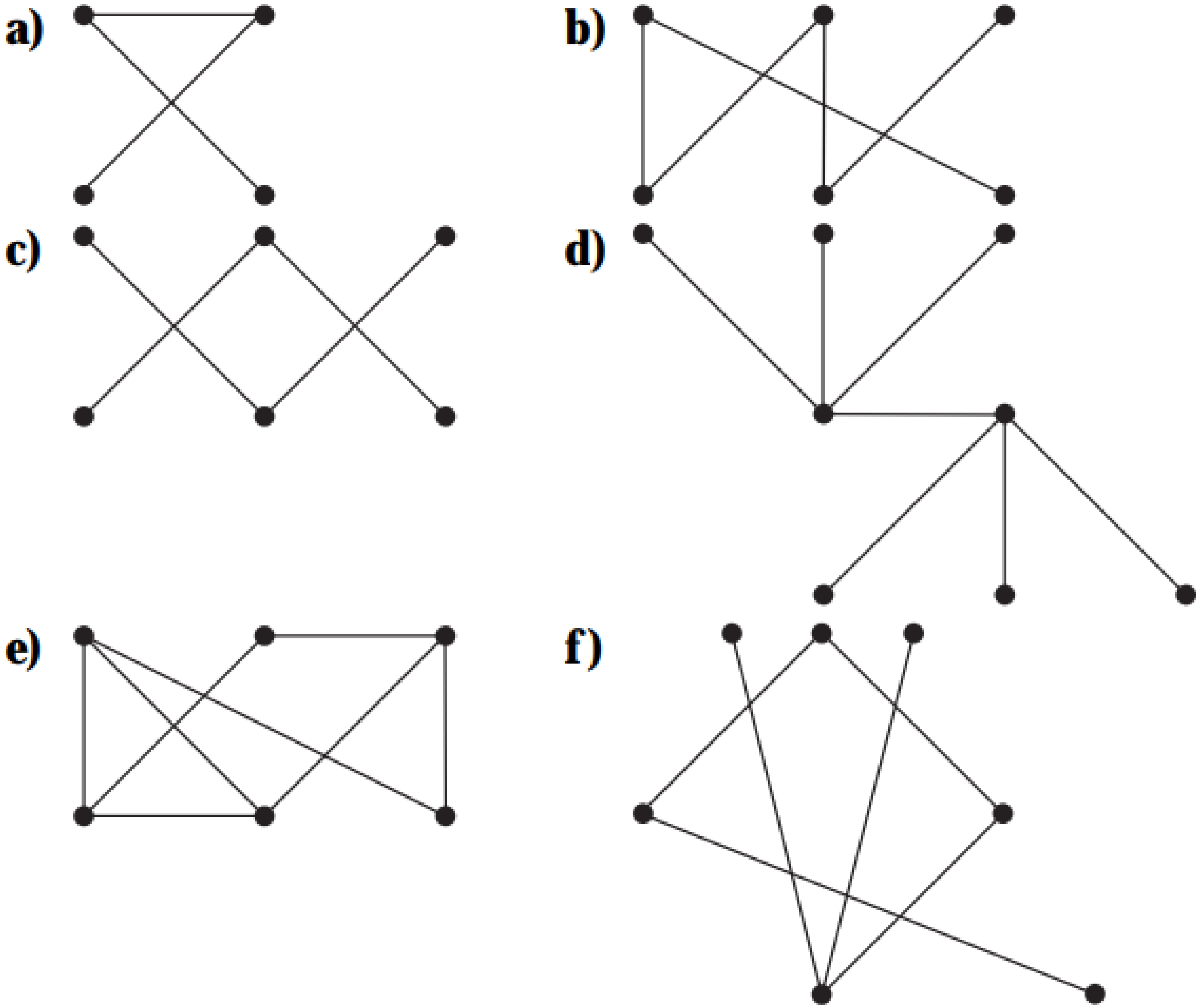
$n = (number \ of \ players) = number \ of \ vertices$

Find $number \ of \ edges$?



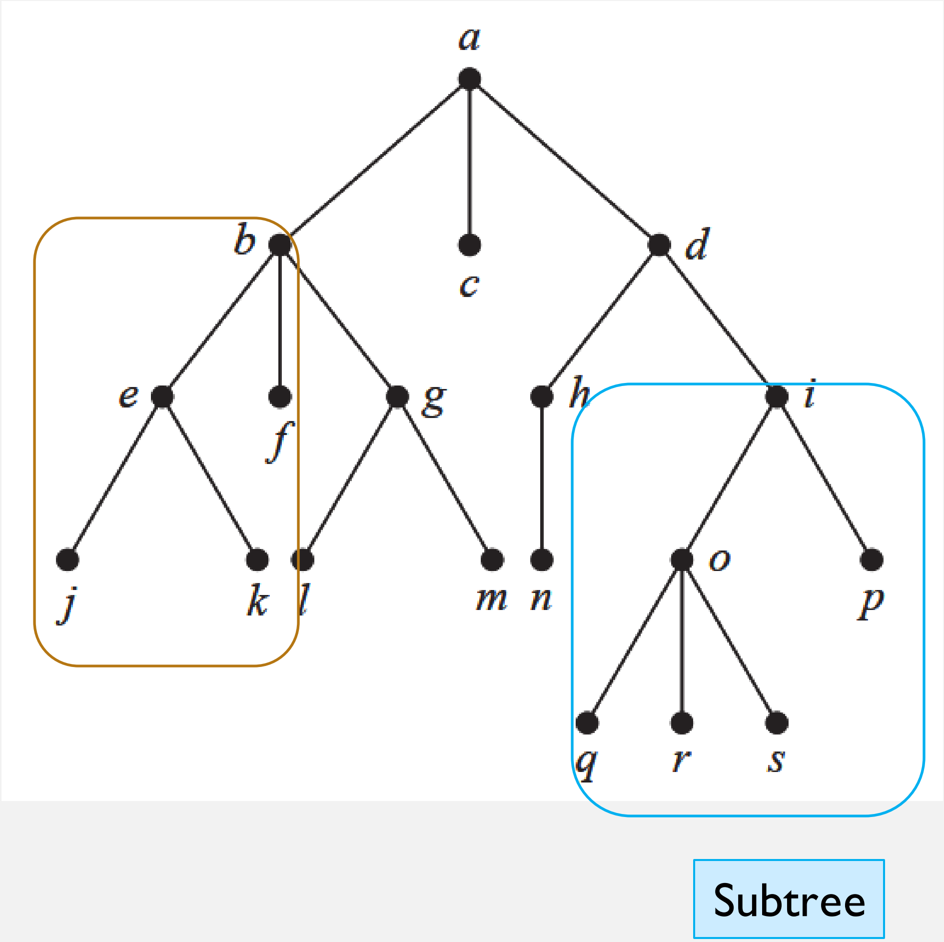
A tree with n vertices has $n - 1$ edges

WHICH OF THESE GRAPHS ARE TREES?



Answer these questions about the rooted tree illustrated.

a) Which vertex is the root?
b) Which vertices are internal?
c) Which vertices are leaves?
d) Which vertices are children of j?
e) Which vertex is the parent of h?
f) Which vertices are siblings of o?
g) Which vertices are ancestors of m?
h) Which vertices are descendants of b?

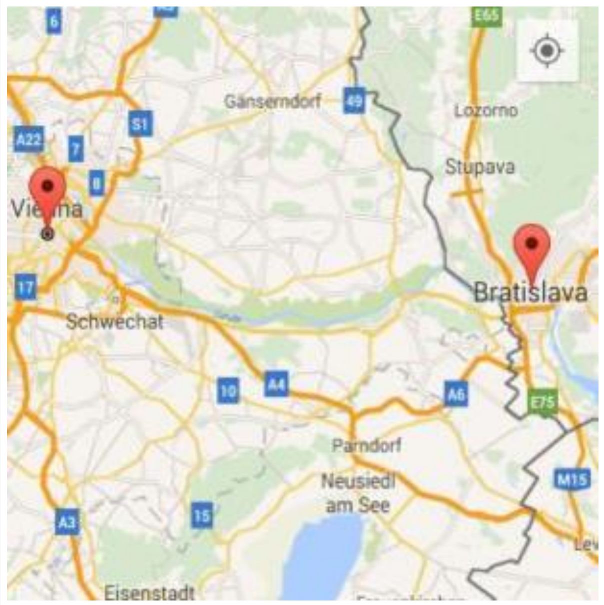


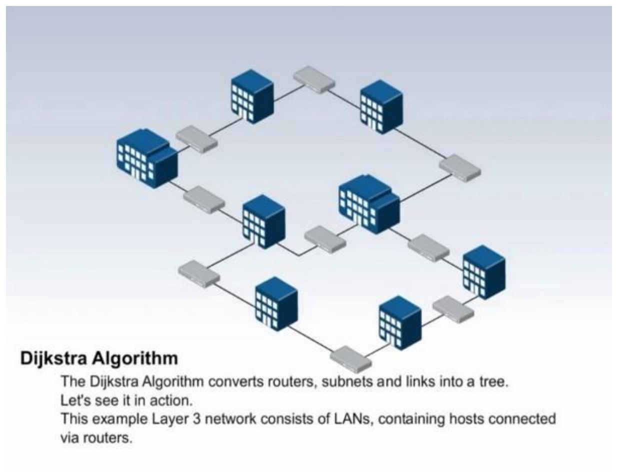
DIJKSTRA'S SHORTEST PATH ALGORITHM

- **Objective**: to find the shortest path between any two vertices in a graph.

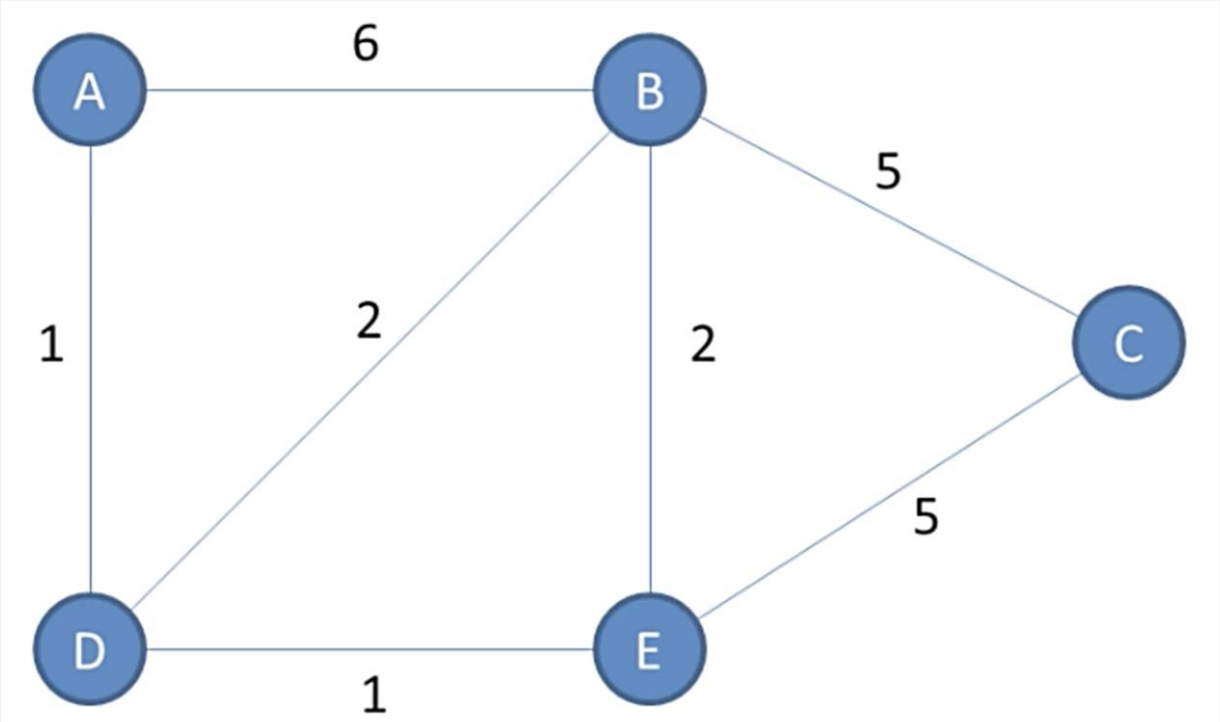
USAGE OF DJIKSTRA'S ALGORITHM

- Digital Mapping Services in Google Maps
- Social Networking Applications
- Telephone Network
- IP routing to find Open shortest Path First
- Flighting Agenda
- Designate file server
- Robotic Path





Find the shortest path from vertex A to every other vertex

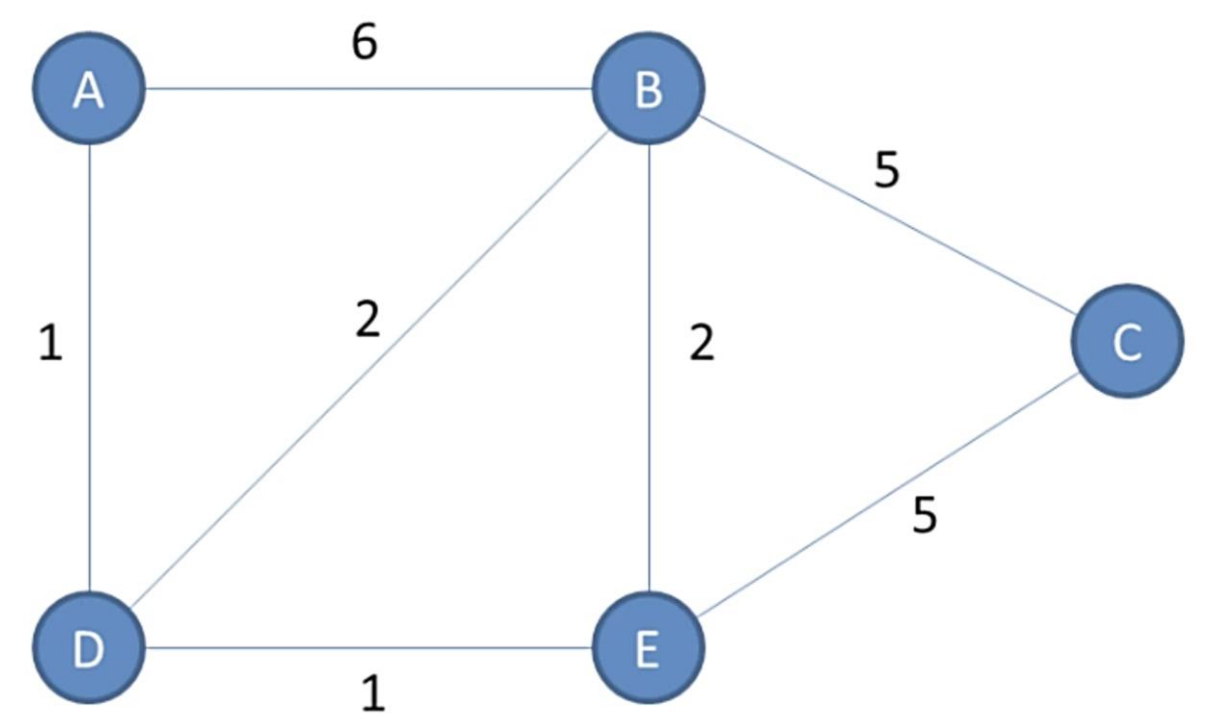


Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

Consider the start vertex, A

Distance from A to A = 0

Distances to all other vertices from A are unknown, therefore ∞ (infinity)

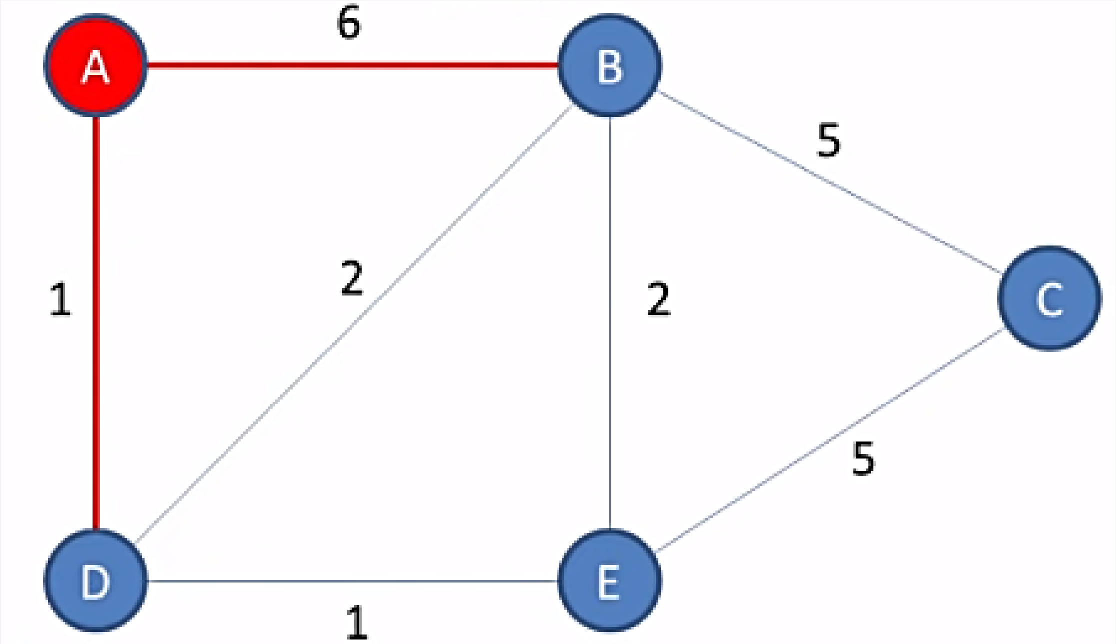


Vertex	Shortest distance from A	Previous vertex
A	0	
B	∞	
C	∞	
D	∞	
E	∞	

Visited [ ]	Unvisited [A, B, C, D, E]

Visit the unvisited vertex with the smallest known distance from the start vertex

This time around, it is vertex D



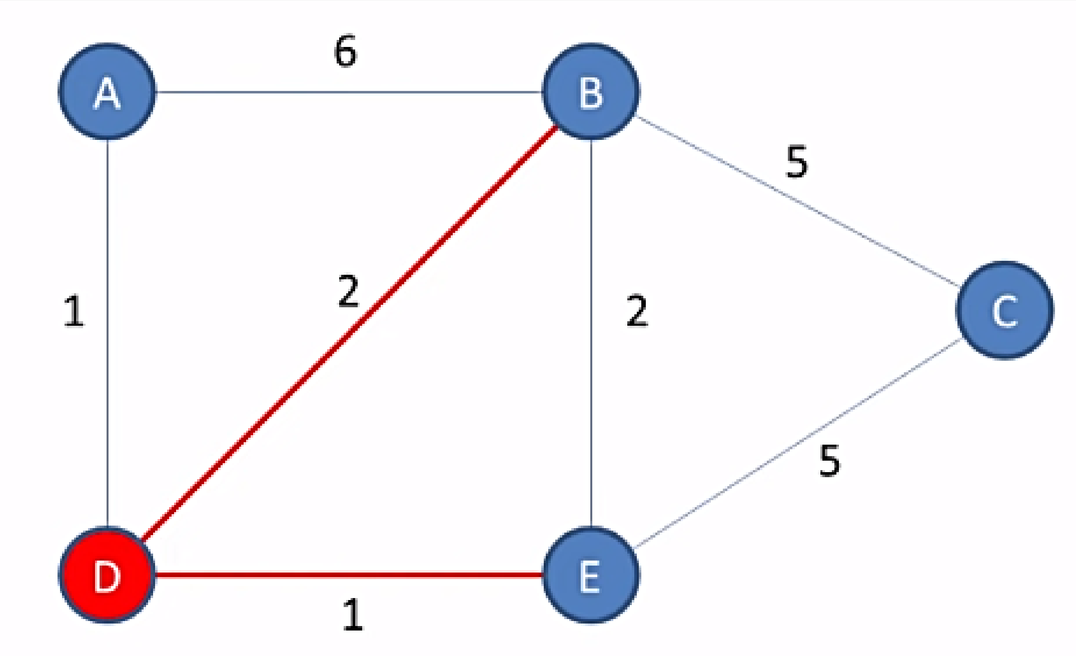
Vertex	Shortest distance from A	Previous vertex
A	0	
B	6	A
C	∞	
D	$\boxed{1}$	A
E	∞	

Visited [A]	Unvisited [B, C, D, E]

For the current vertex, examine its unvisited neighbors

We are currently visiting D and its unvisited neighbors are B and E

For the current vertex, calculate the distance of each neighbor from the start vertex



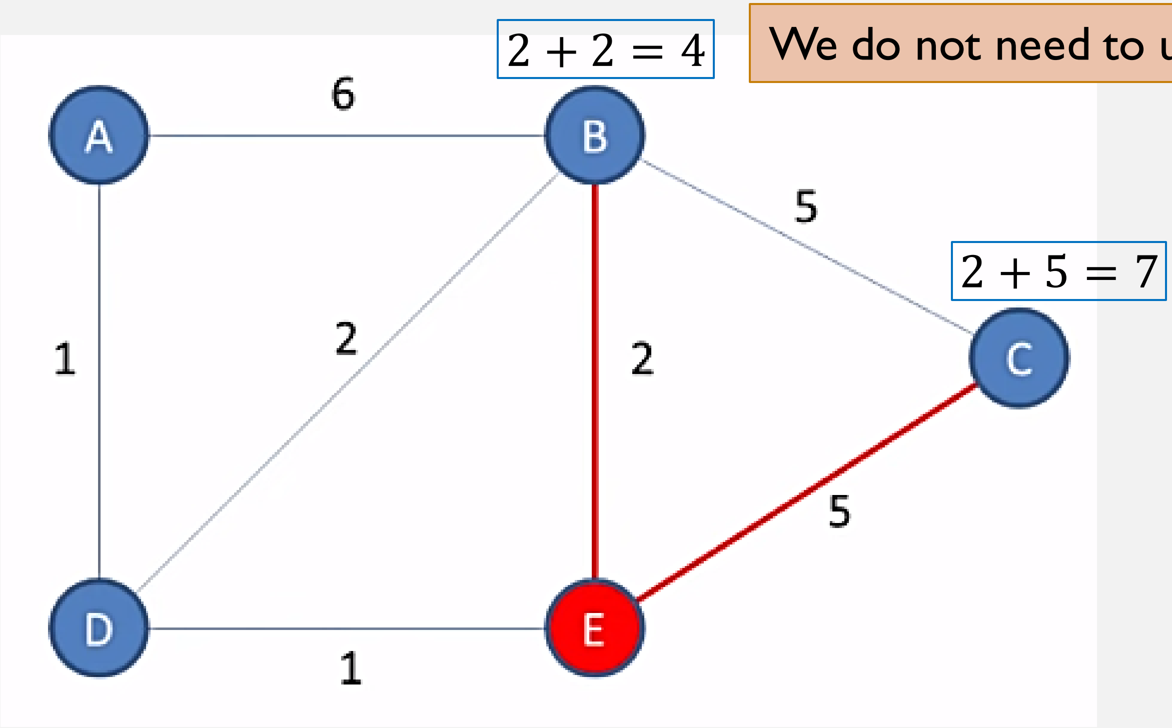
Vertex	Shortest distance from A	Previous vertex
A	0	
B	6 ≫ 3	A ≫ *D*
C	∞	
D	1	A
E	2	D

Visited [A, D]	Unvisited [B, C, E]

Visit the unvisited vertex with the smallest known distance from the start vertex

This time around, it is vertex E

We do not need to update the distance to B

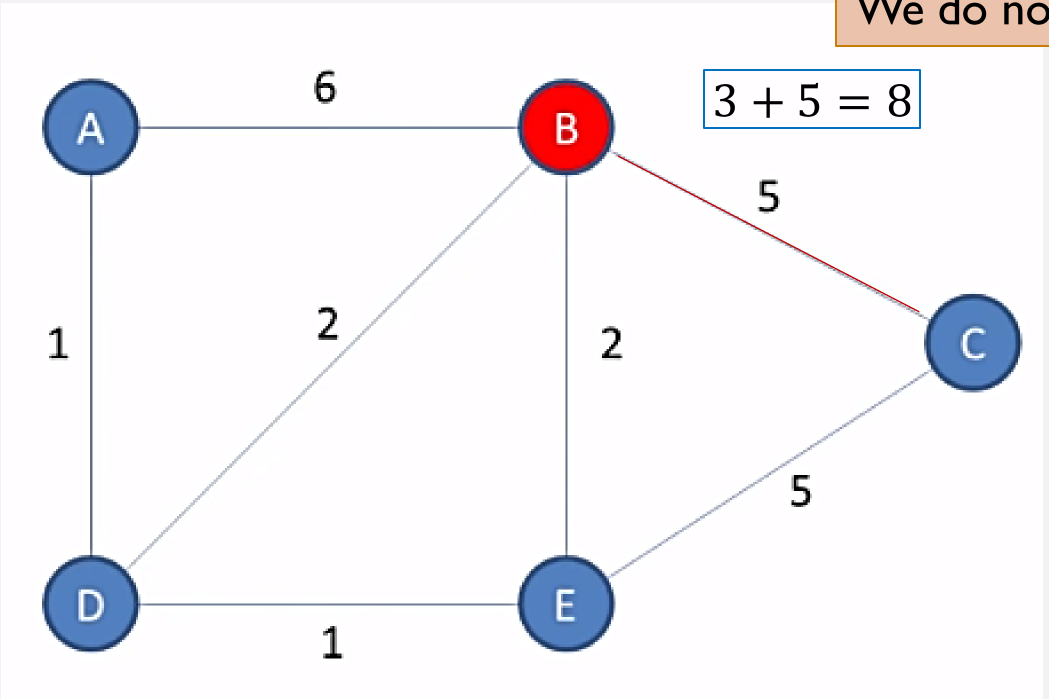


Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	7	E
D	1	A
E	$\boxed{2}$	D

Visited [A, D, E]	Unvisited [B, C]

For the current vertex, examine its unvisited neighbors

We do not need to update the distance to C

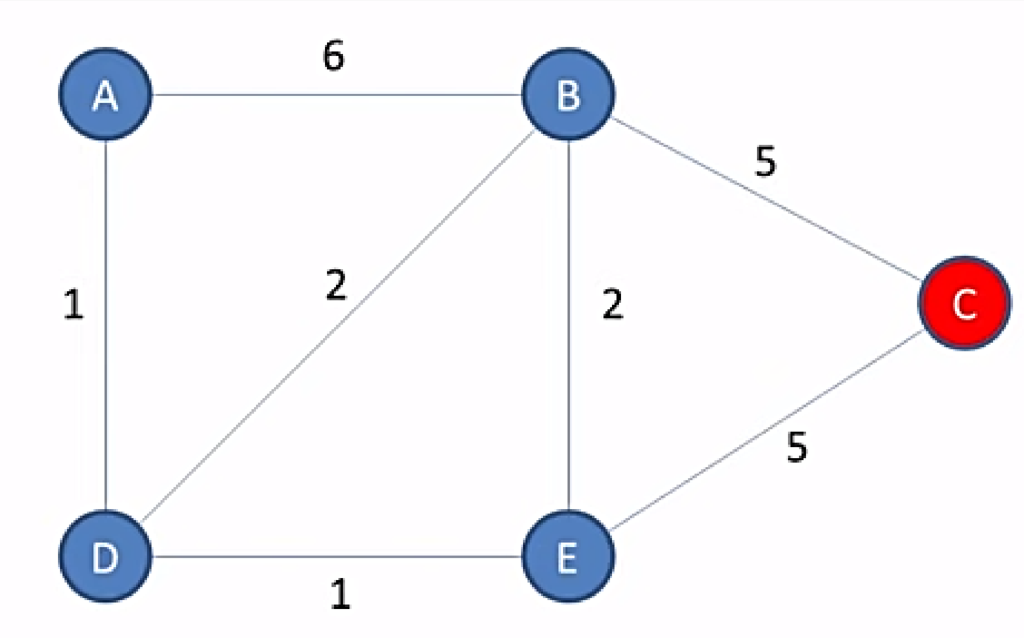


$\boxed{3 + 5 = 8}$

Vertex	Shortest distance from A	Previous vertex
A	0	
B	$\boxed{3}$	*D*
C	7	E
D	1	A
E	2	D

Visited [A, D, E, B]	Unvisited [C]

We are currently visiting C and it has no unvisited neighbors



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	*D*
C	$\boxed{7}$	E
D	1	A
E	2	D

Visited [A, D, E, B, C]	Unvisited [ ]

ALGORITHM

- Let distance of start vertex from start vertex = 0
- Let distance of all other vertices from start = ∞ (infinity)

Repeat

- Visit the unvisited vertex with the smallest known distance from the start vertex
- For the current vertex, examine its unvisited neighbors
- For the current vertex, calculate distance of each neighbor from start vertex
- If the calculated distance of a vertex is less than the known distance, update the shortest distance
- Update the previous vertex for each of the updated distances
- Add the current vertex to the list of visited vertices

Until all vertices visited

ALGORITHM

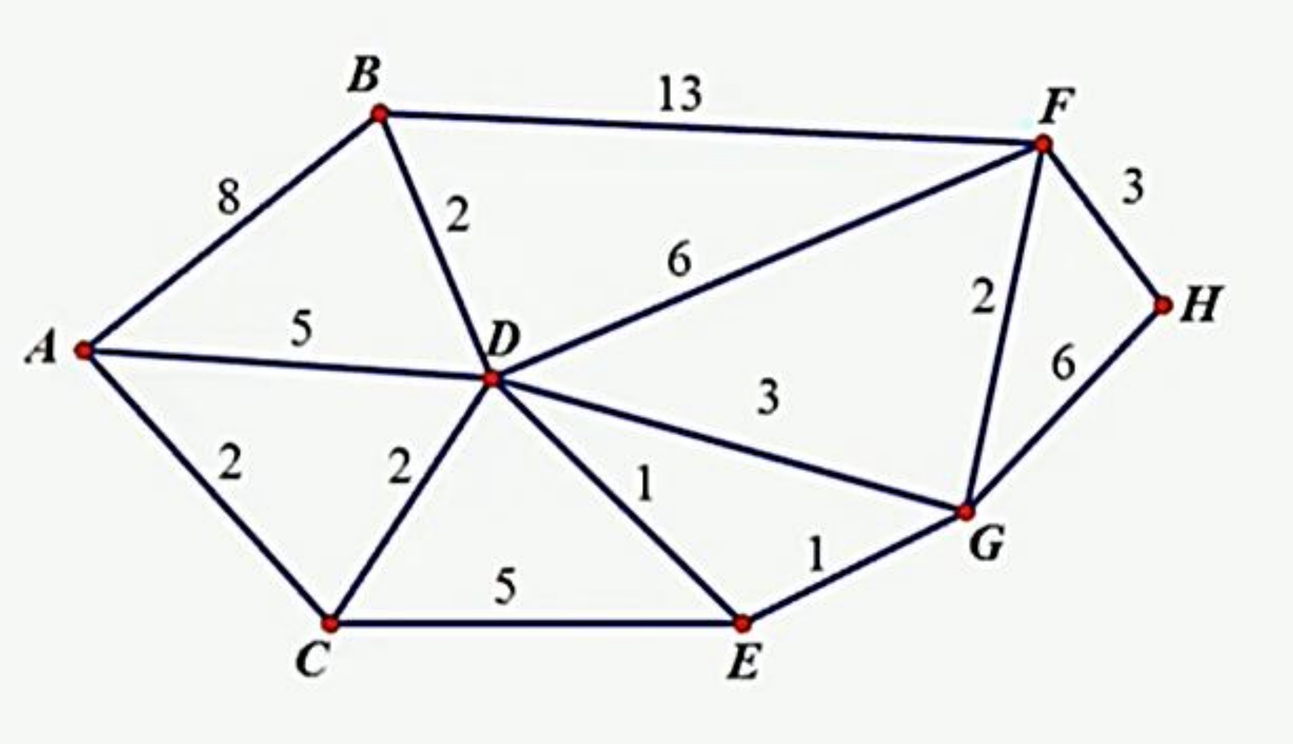
- Let distance of start vertex from start vertex = 0
- Let distance of all other vertices from start = ∞ (infinity)

WHILE vertices remain unvisited

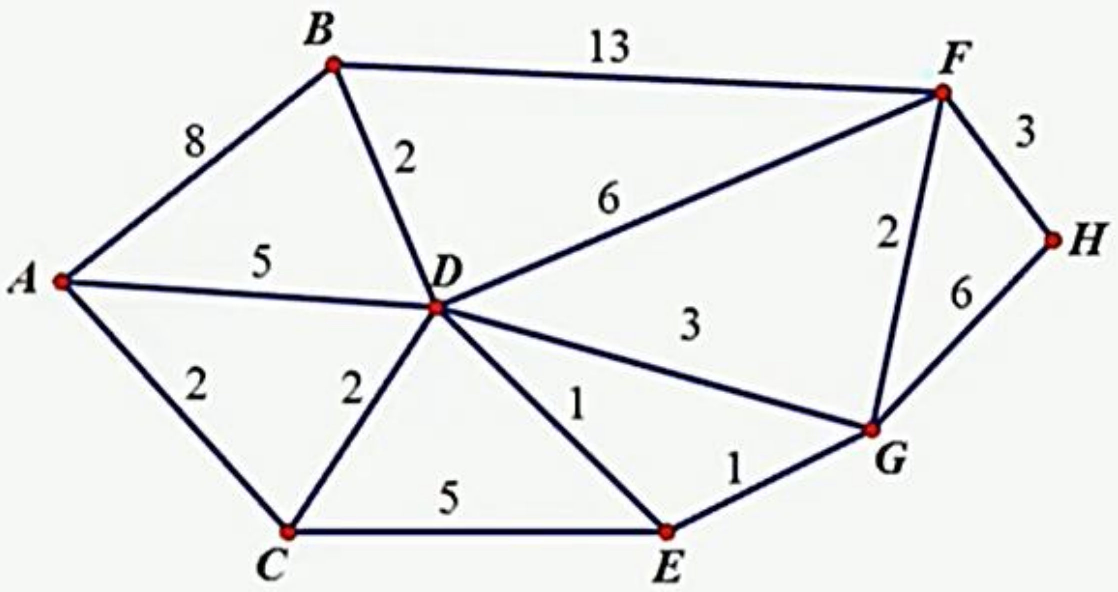
- Visit unvisited vertex with the smallest known distance from the start vertex (call this current vertex)
- FOR each unvisited neighbor of the current vertex
- Calculate distance from start vertex
- If the calculated distance of this vertex is less than the known distance
- Update the shortest distance to this vertex
- Update the previous vertex with the current vertex
- end if
- NEXT unvisited neighbor
- Add the current vertex to the list of visited vertices

END WHILE

PRACTICE



Find the shortest path from vertex A to all other vertices.



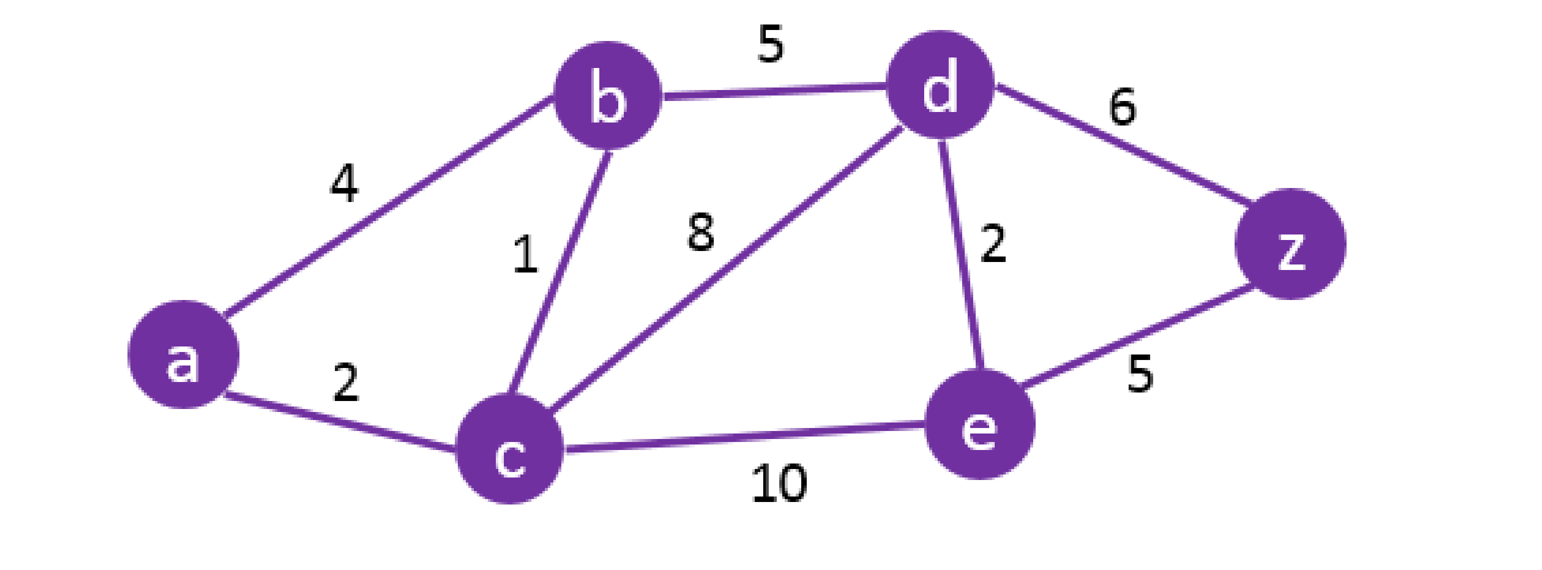
V	A	B	C	D	E	F	G	H
A	$\boxed{0_A}$	8_A	2_A	5_A	∞	∞	∞	∞
C		8_A	$\boxed{2_A}$	4_C	7_C	∞	∞	∞
D		6_D		$\boxed{4_C}$	5_D	10_D	7_D	∞
E		6_D			$\boxed{5_D}$	10_D	6_E	∞
B		$\boxed{6_D}$				10_D	6_E	∞
G						8_G	$\boxed{6_E}$	12_G
F						$\boxed{8_G}$		11_F
H								$\boxed{11_F}$

TRY IT

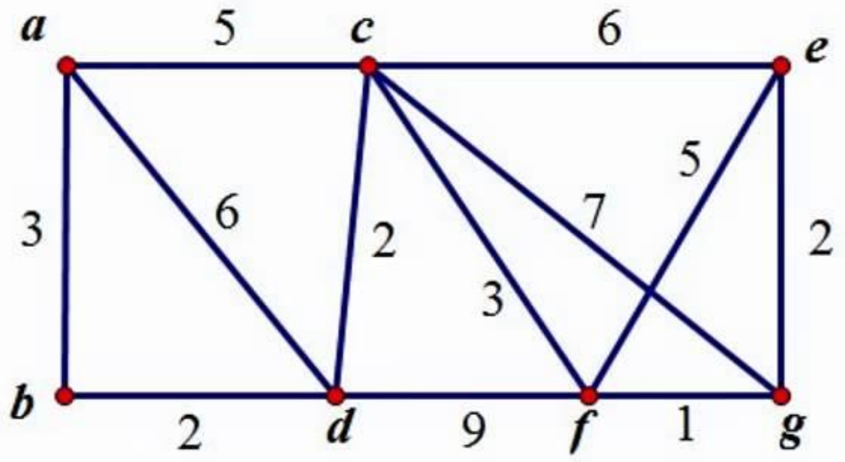
Show your solution in a table format as shown below.

Vertex	Shortest distance from A	Previous vertex
a	0	
b		
c		
d		
e		
z		

Find the shortest path from vertex A to all other vertices.



CONTINUE THE SOLUTION



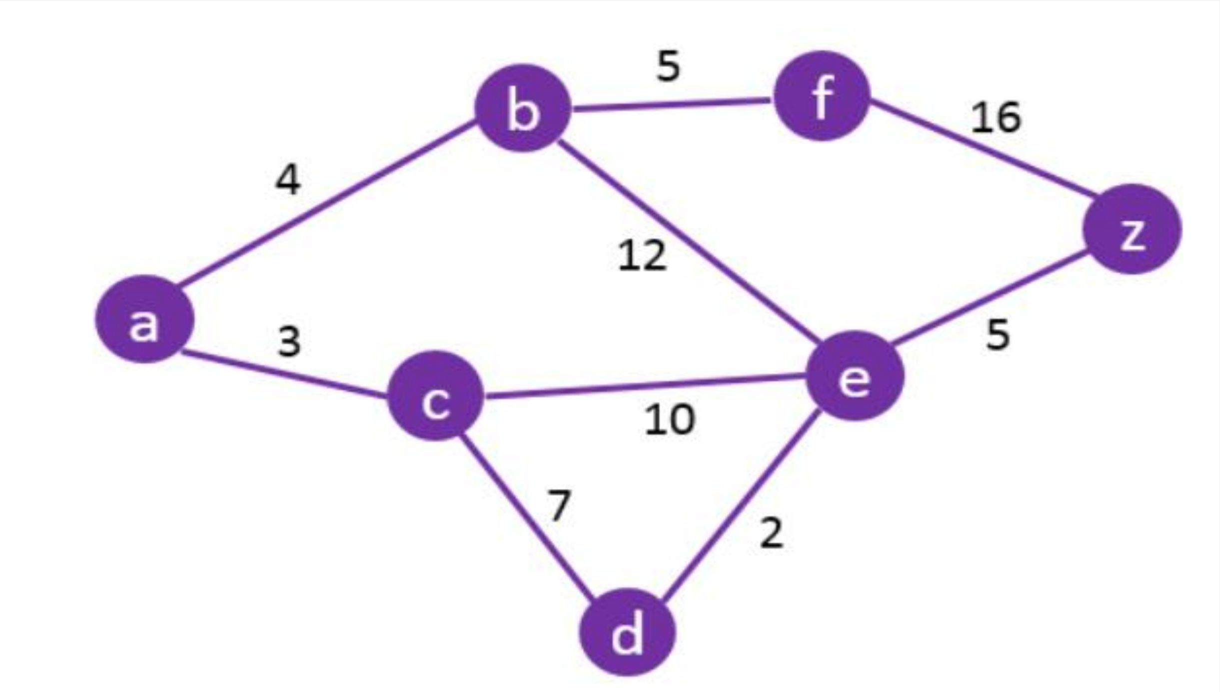
v	a	b	c	d	e	f	g
a	0_a	3_a	5_a	6_a	∞_a	∞_a	∞_a
b	0_a	3_a	5_a	5_b	∞_a	∞_a	∞_a
c	0_a	3_a	5_a	5_b	11_c	8_c	12_c
d	0_a	3_a	5_a	5_b	11_c	8_c	12_c
f	0_a	3_a	5_a	5_b	11_c	8_c	9_f

TRY IT

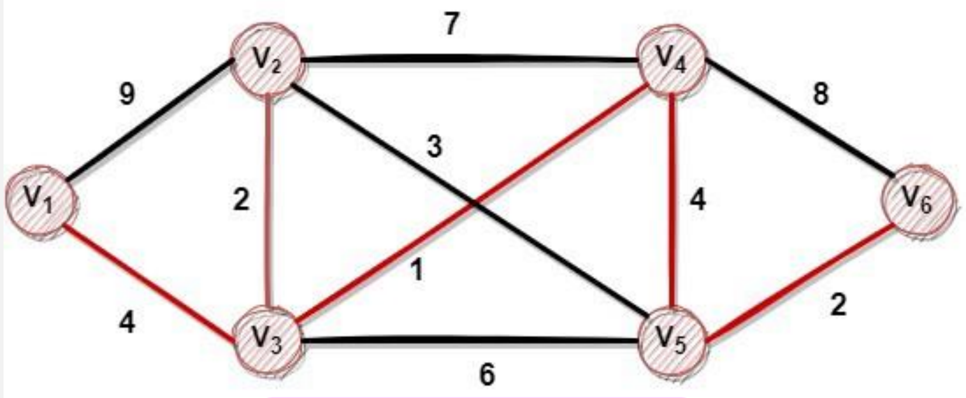
Find the shortest path from vertex A to all other vertices.

Show your solution in a table format as shown below (for each vertex).

	a	b	c	d	e	f	z
a							



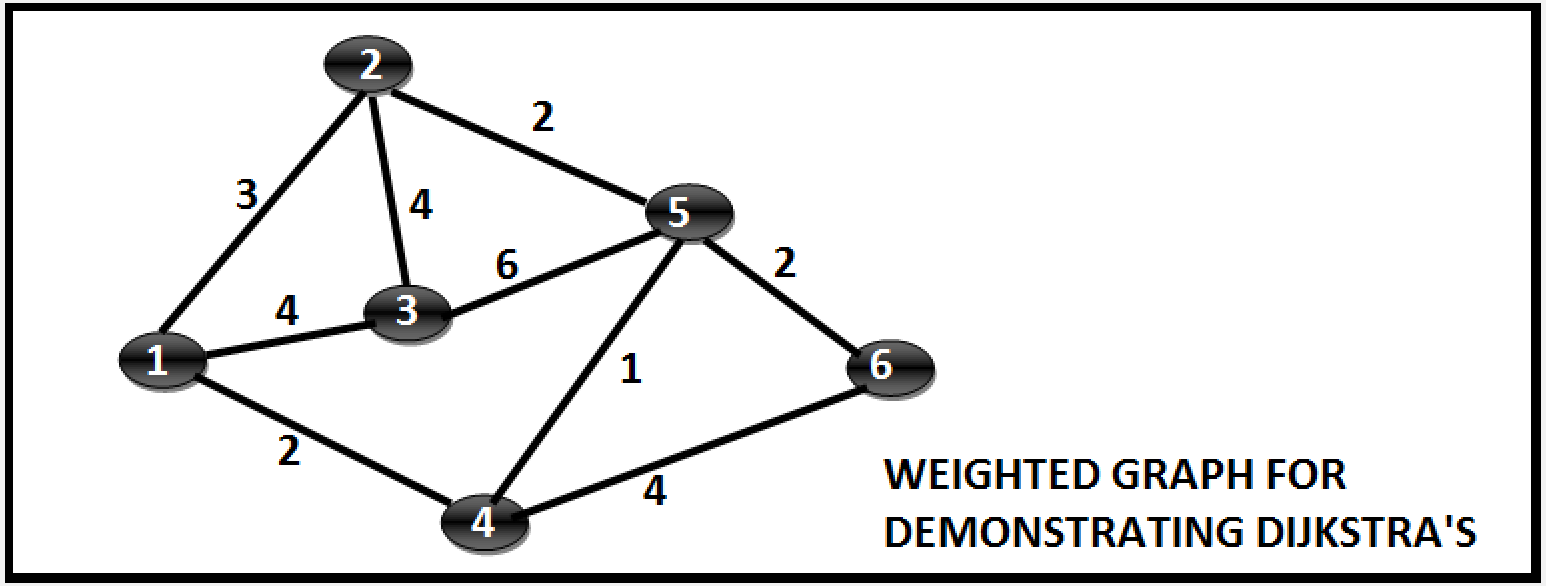
Dijkstra's Algorithm



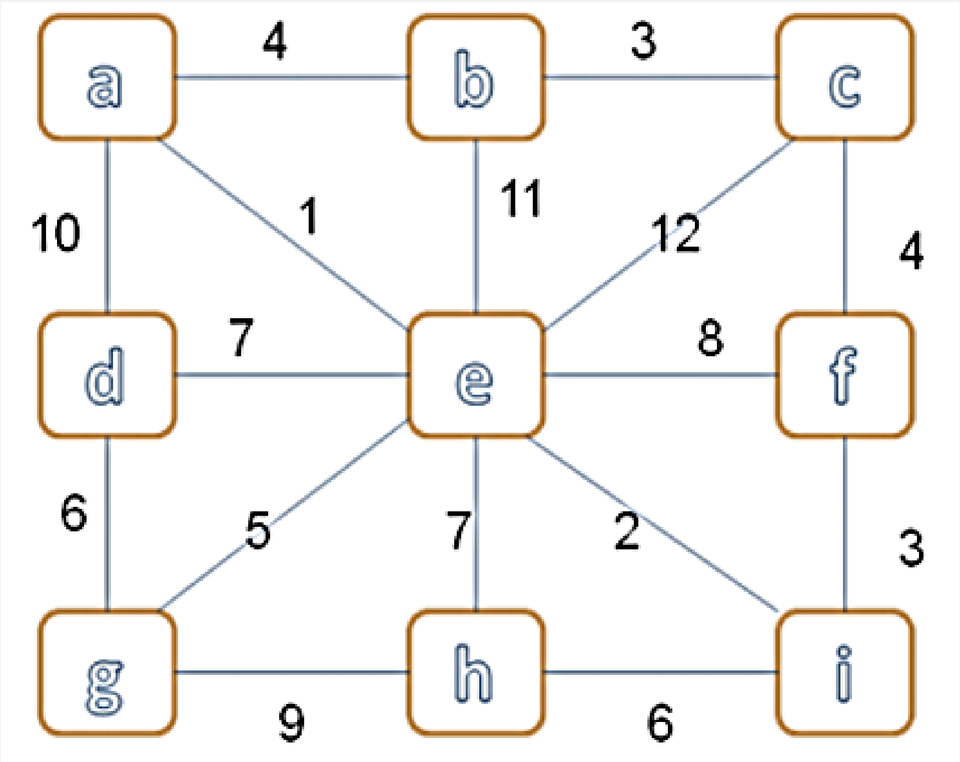
Find the shortest path from V_1 to V_6.

Find the shortest path from vertex 1 to any other vertex.

Show your solution in a table format (for each vertex).



Find the shortest path from vertex a to each vertex using Dijkstra’s algorithm.

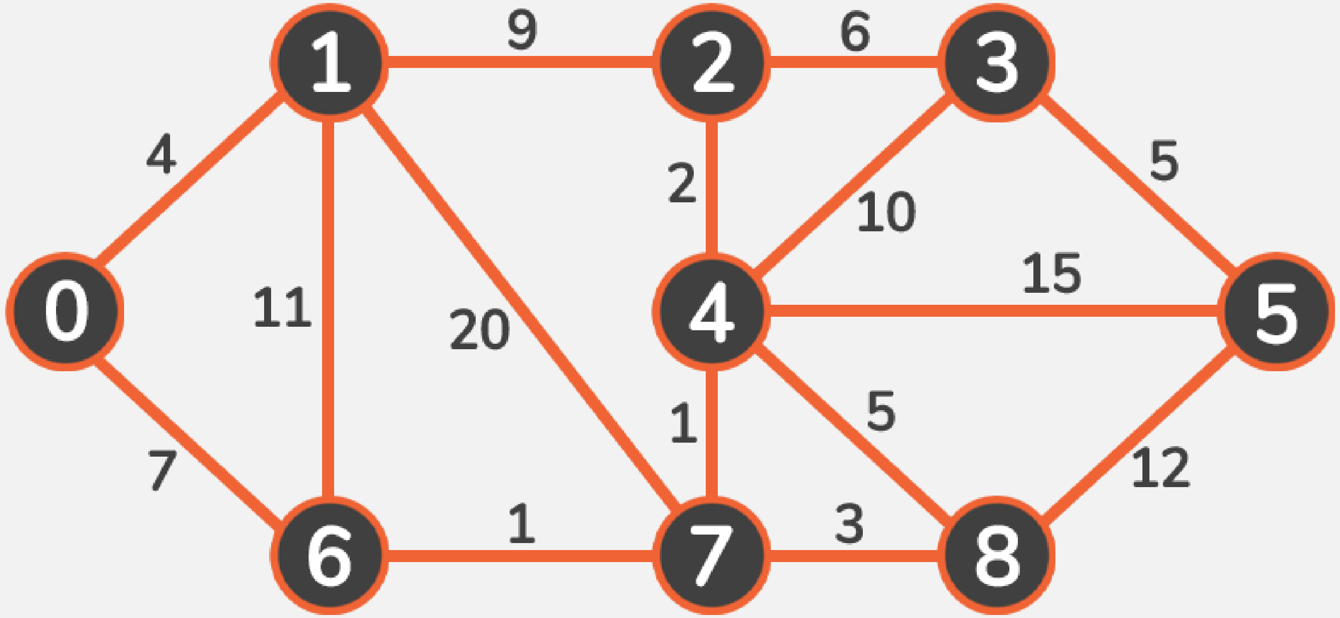


Vertex	Known	Distance	Path
A	T	0	-
B	T	4	A
C	T	7	B
D	T	8	E
E	T	1	A
F	T	6	I
G	T	6	E
H	T	8	E
I	T	3	E

PRACTICE on your OWN

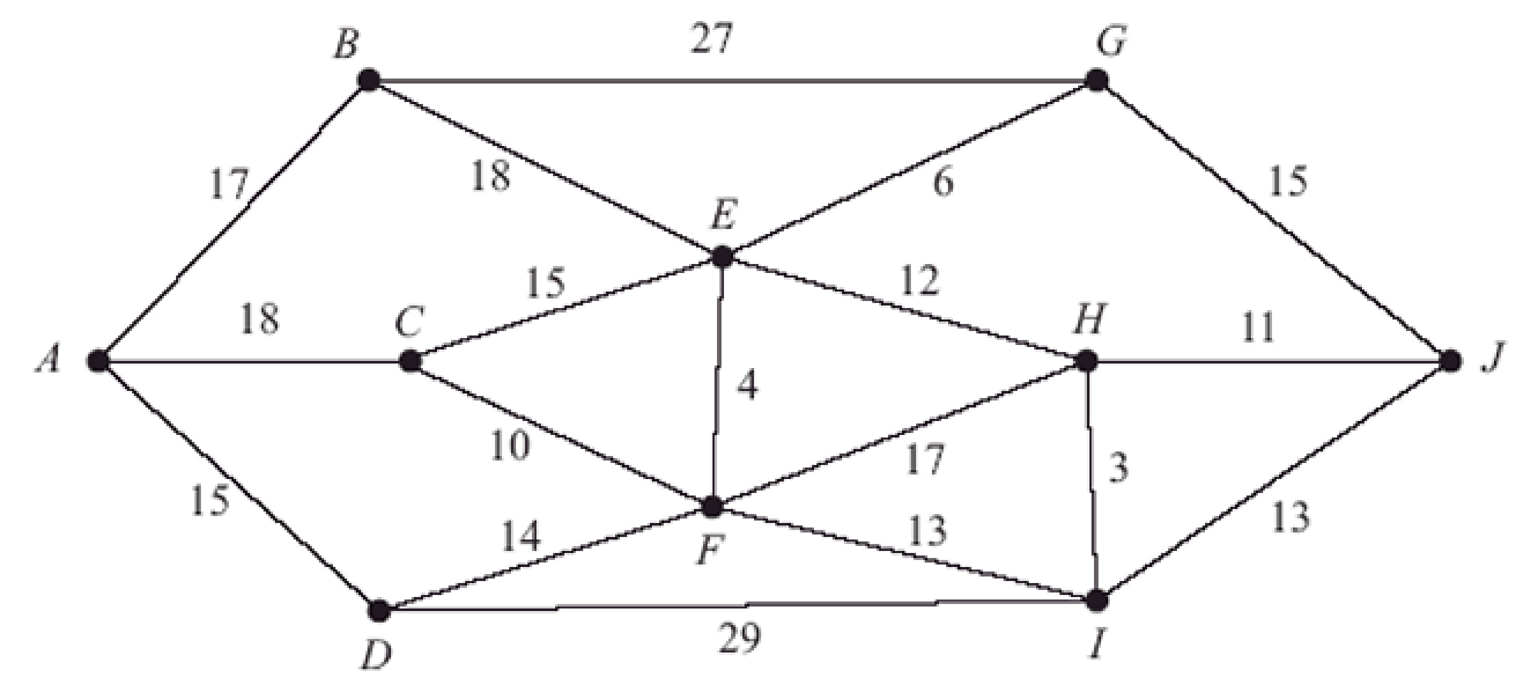
Dijkstra's algorithm

Finding the shortest path from a vertex to any other vertices.



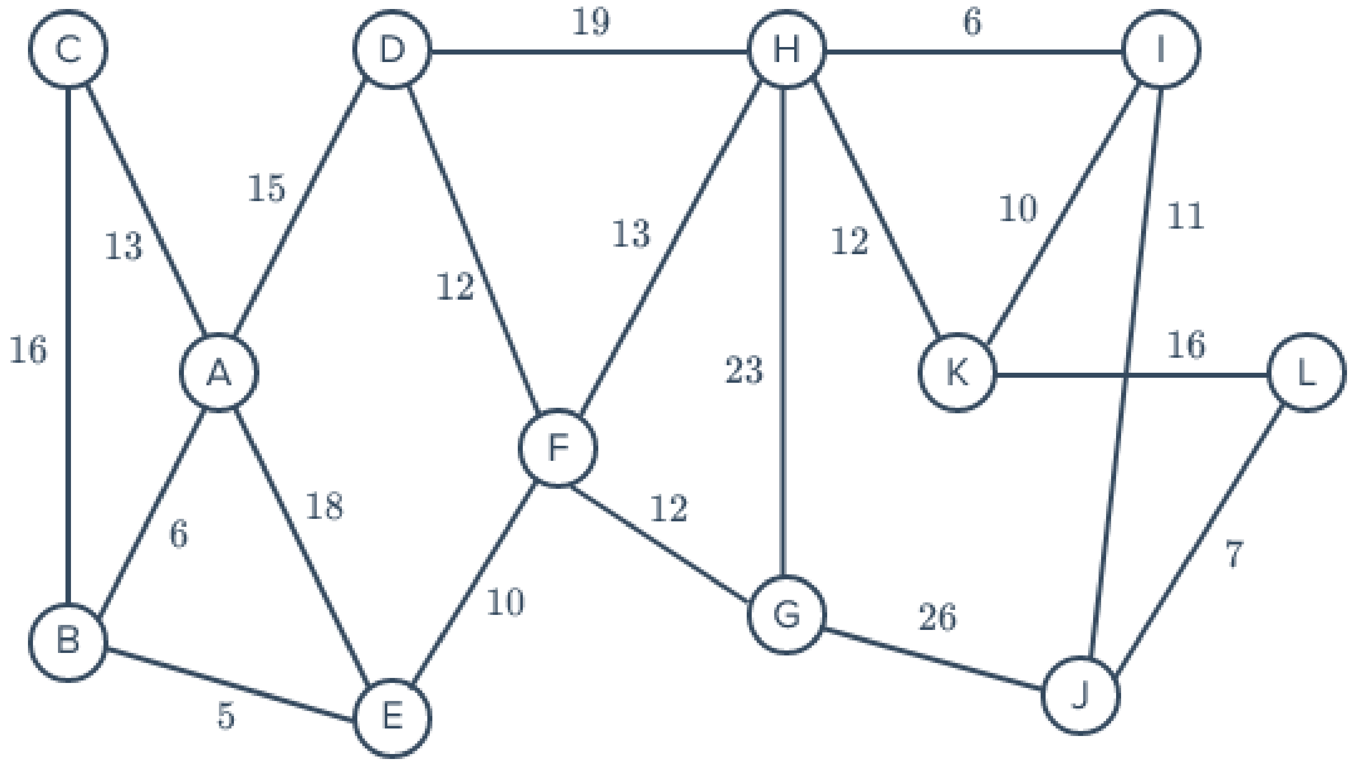
Show your solution in a table format (both).

Use Dijkstra's algorithm to find the shortest route from **A to J**. State your shortest route and its length.



Route: A C F E G J
Length: 53 km

HOME WORK



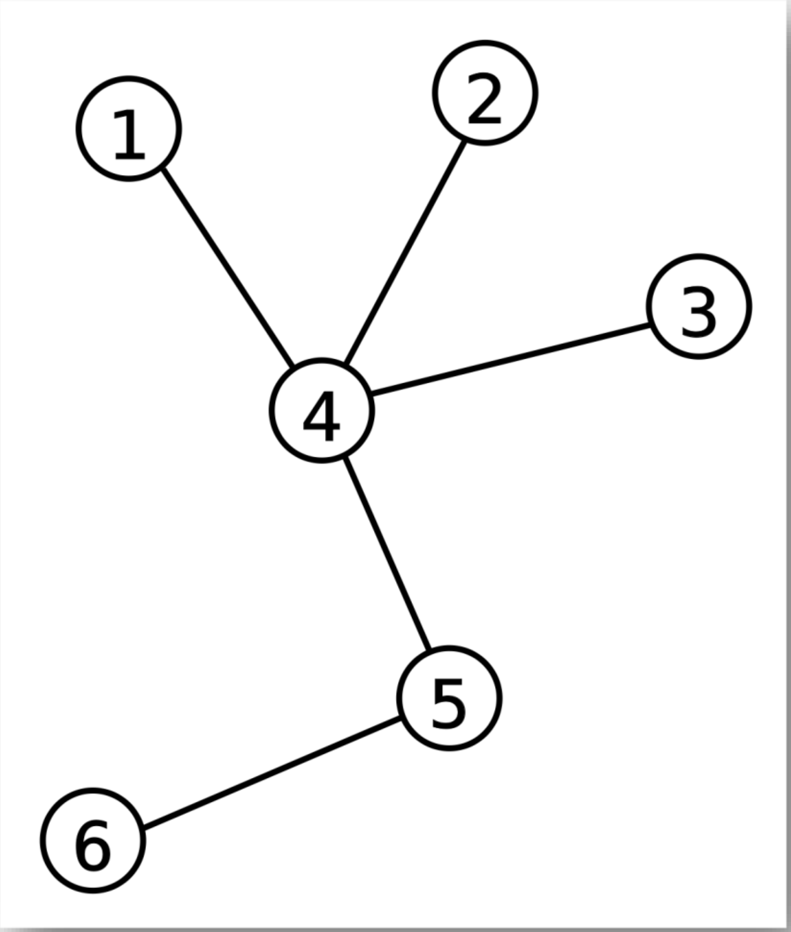
Use Dijkstra's algorithm to find the shortest route from A to any other vertices. State your shortest route and its length.

HOME WORK

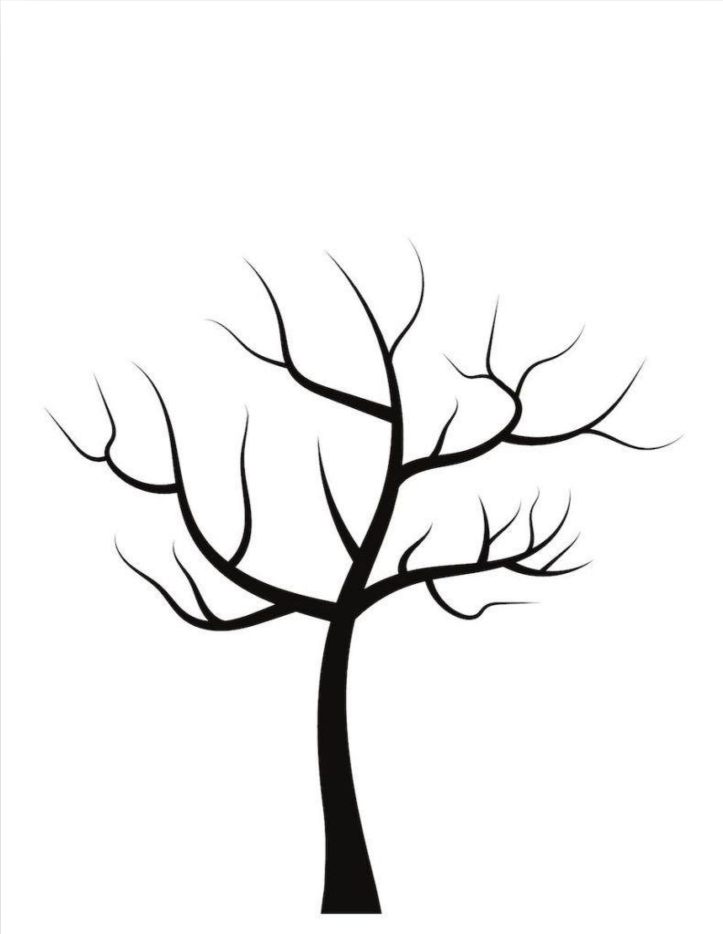
Kütahya is a city famous for its ceramics. A company located in Kütahya will distribute the ceramic dinner sets it produced to other provinces in the Aegean Region. The distribution truck of each province is separate. A route for each truck is required. What are the shortest distance routes to the seven provinces in the region starting from Kütahya? Show your answer by creating a model. To solve the problem, you may use the map below and the distances between the provinces.

Kütahya-Afyon: 110 km
Kütahya-Uşak: 140 km
Kütahya-Manisa: 350 km
Afyon-Uşak: 130 km
Afyon-Denizli: 240 km
Uşak-Denizli: 140 km
Uşak-İzmir: 220 km
Uşak-Manisa: 190 km
Denizli-Manisa: 200 km
Denizli-Aydın: 120 km
Denizli-Muğla: 150 km
Muğla-Aydın: 110 km
Aydın-İzmir: 110 km
İzmir-Manisa: 40 km

Recall: trees are graphs that have *no circuits*.

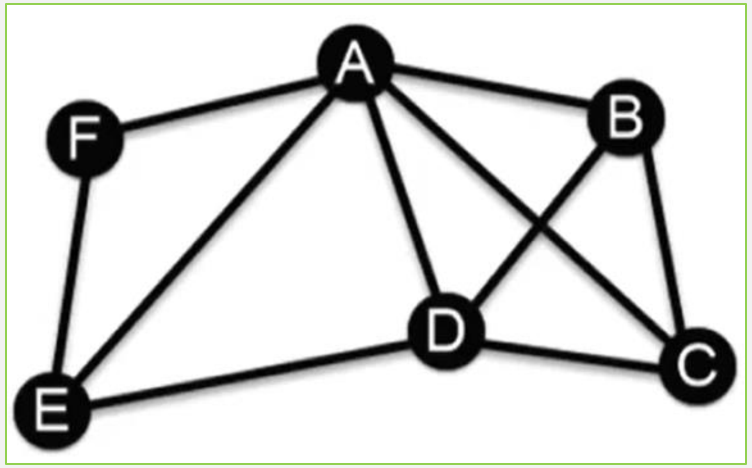


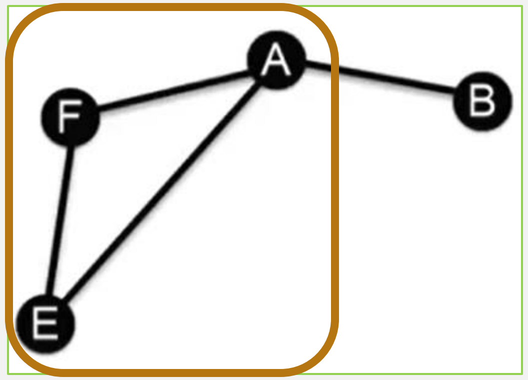
≈



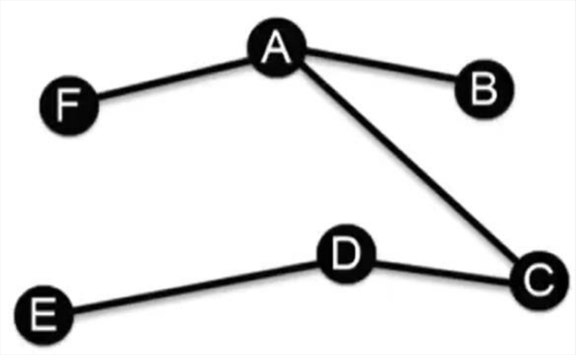
Subgraph

A subgraph for a graph is a graph whose vertices and edges are subsets of the original graph.





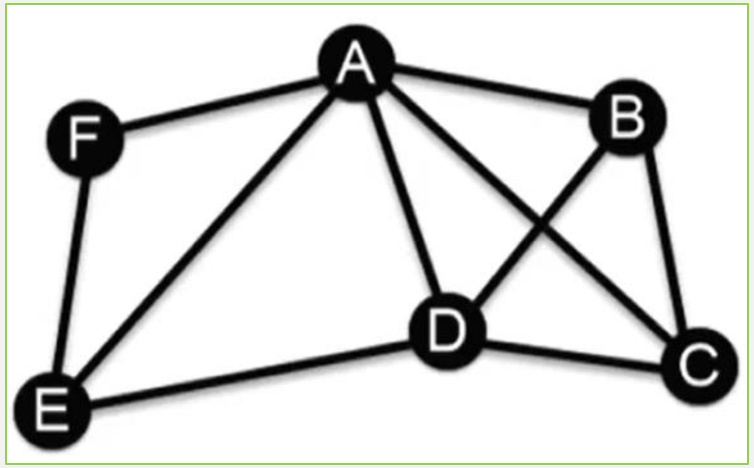
Not a tree

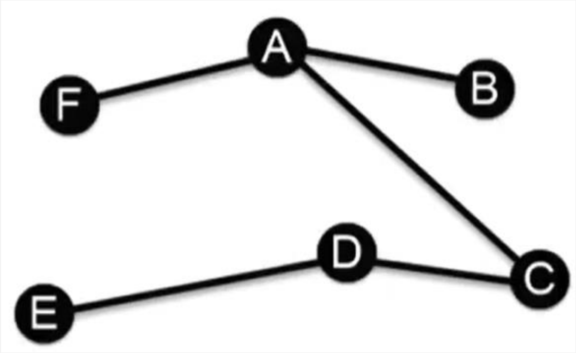


Tree

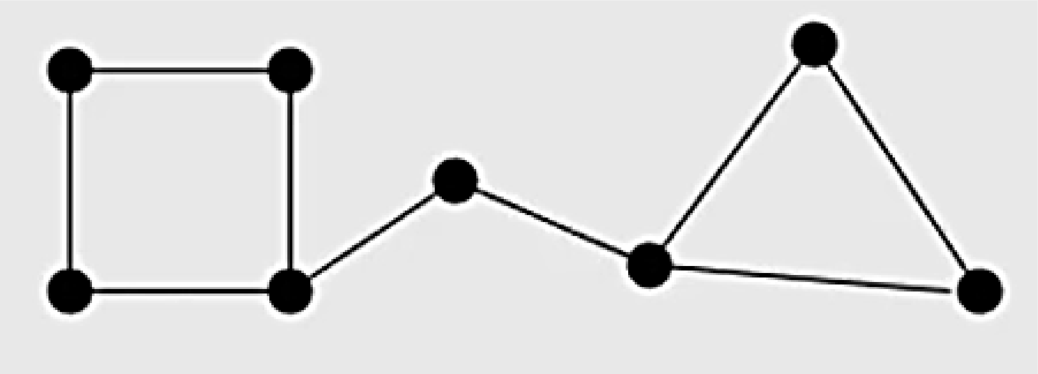
SPANNING TREES

- A spanning tree for a graph is a subgraph that includes every vertex of the original and is a simple tree with no cycles and loops.





Find the spanning tree for the graph below.

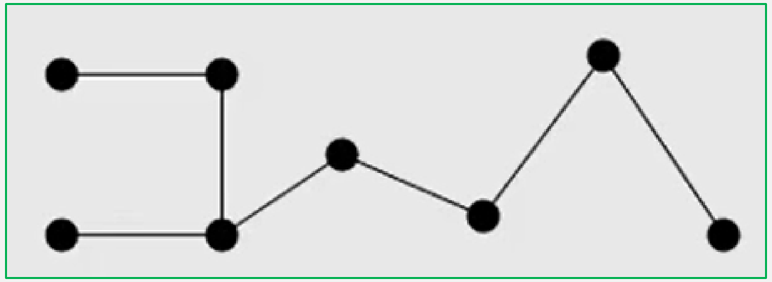


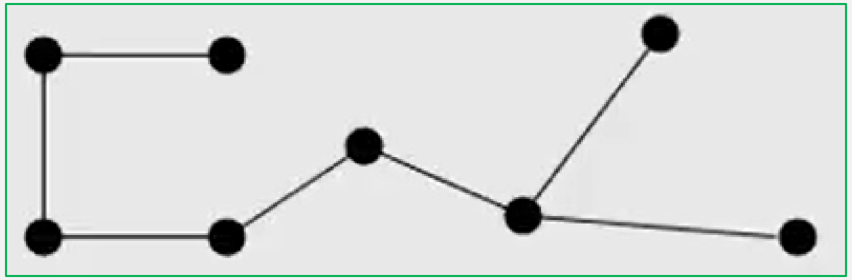
This graph has 8 vertices, 9 edges and 2 circuits.

A spanning tree for 8 vertices <u>must have</u> 7 edges.

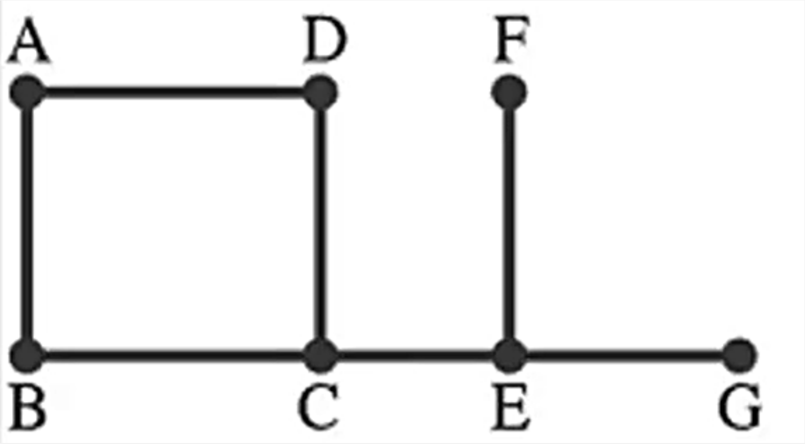
We must remove 2 edges and break 2 circuits.

We break the two circuits by removing a single edge from each. Two possibilities of many:





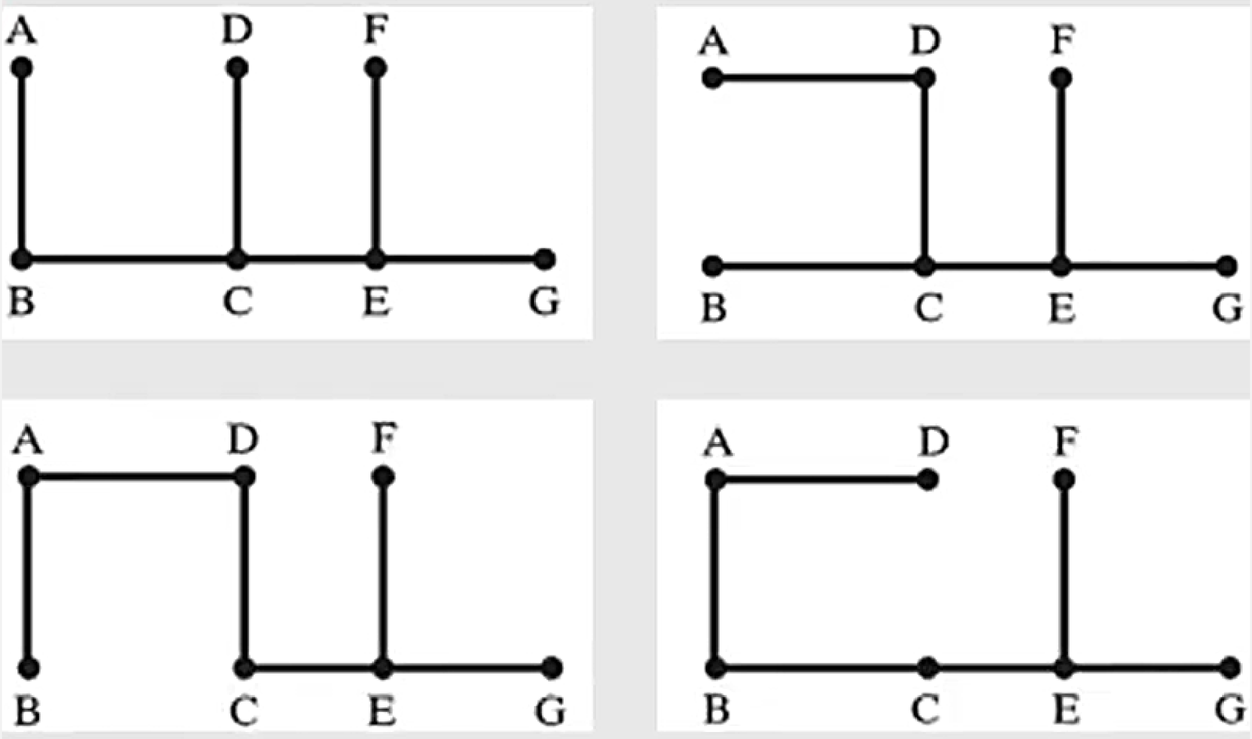
Find all spanning tree for the graph below.



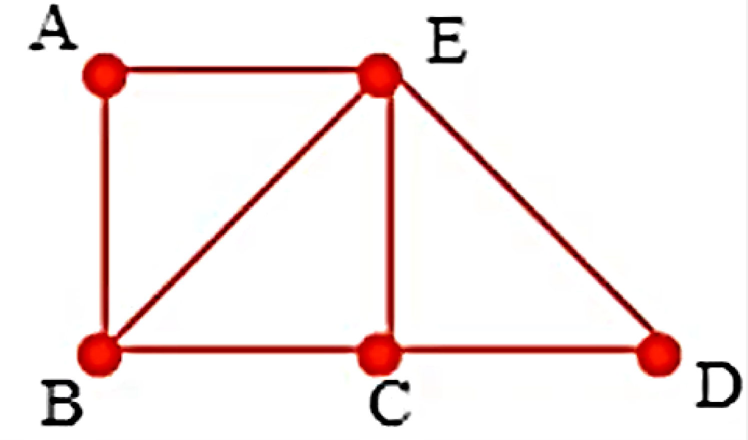
This graph has 7 vertices, 7 edges and 1 circuit.

A spanning tree for 7 vertices must have 6 edges.

We must remove 1 edge and break the circuit.



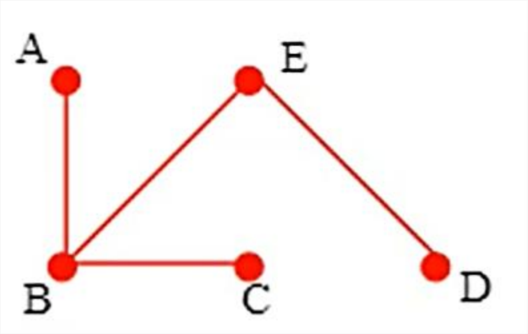
Find 3 different spanning trees for the graph below.

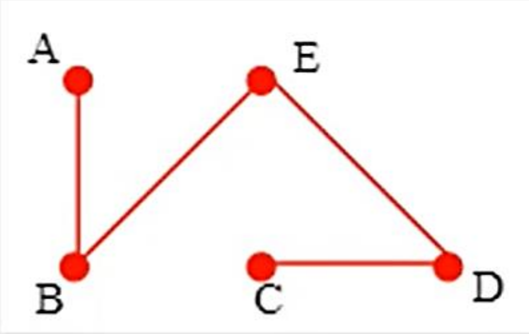


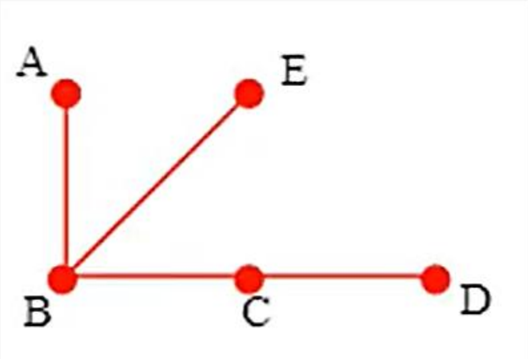
This graph has 5 vertices, 7 edges, multiple overlapping circuits.

A spanning tree for 5 vertices must have 4 edges.

We must remove 3 edges and break the circuits.





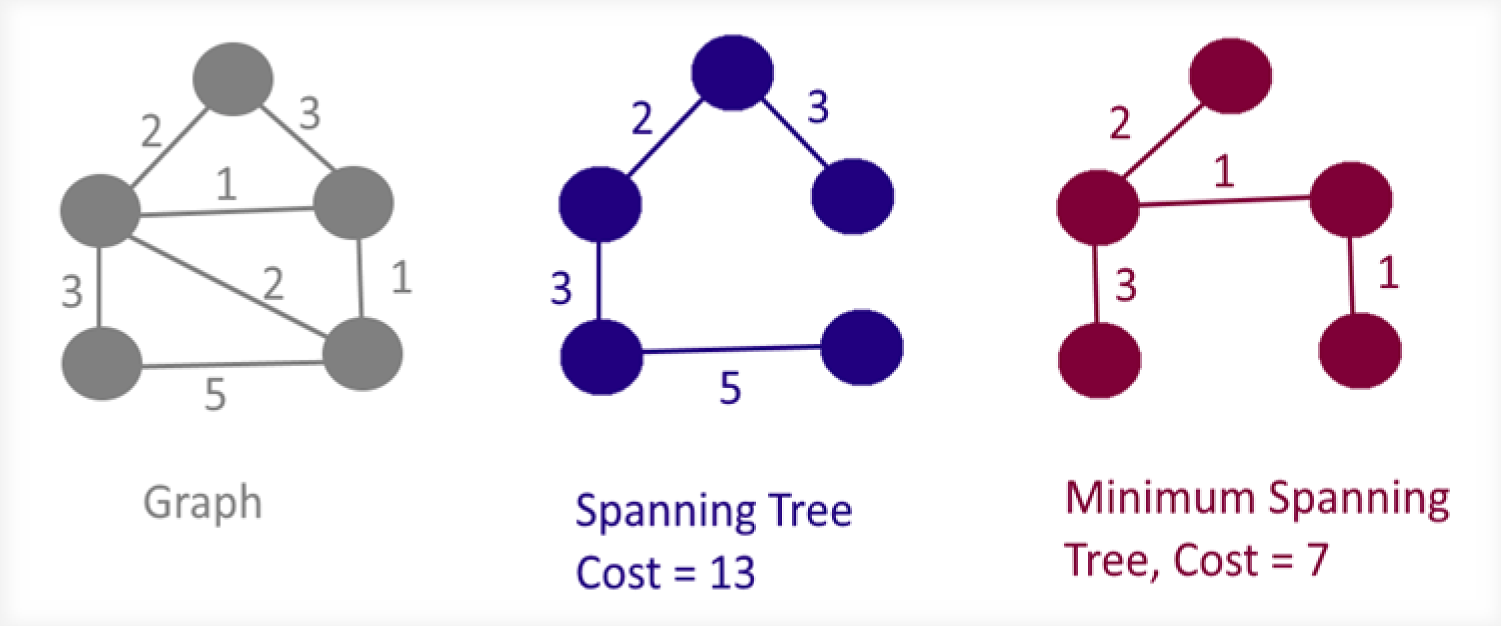


All of these are subgraphs of the original graph, have no circuits, and include all the vertices. So, they are **spanning trees.**

A weighted graph is called a **network**.

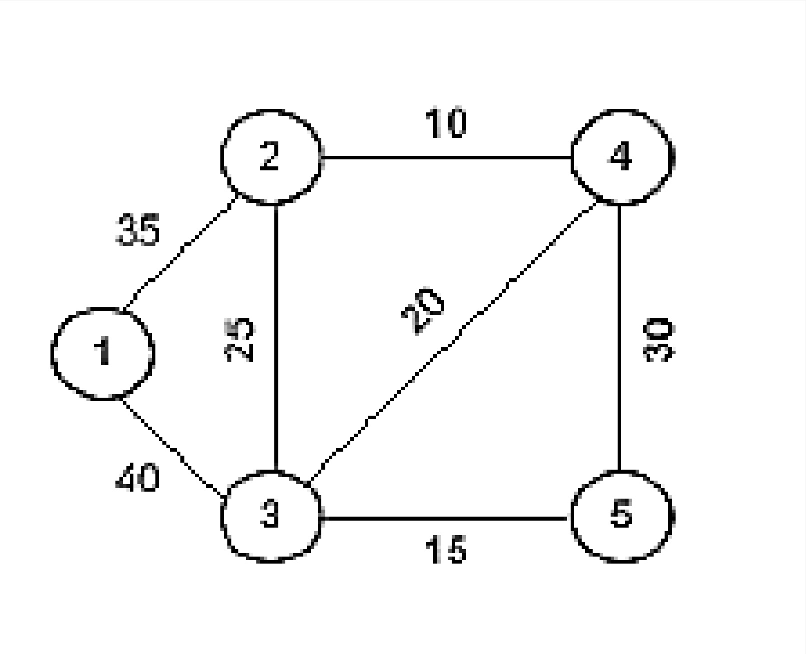
MINIMUM SPANNING TREE (MST)

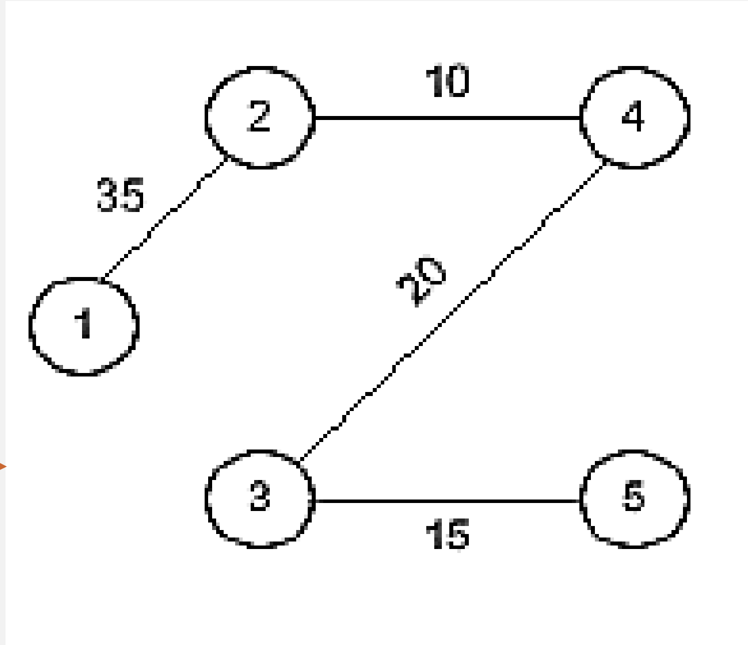
- total / sum of weights of edges must be minimum.



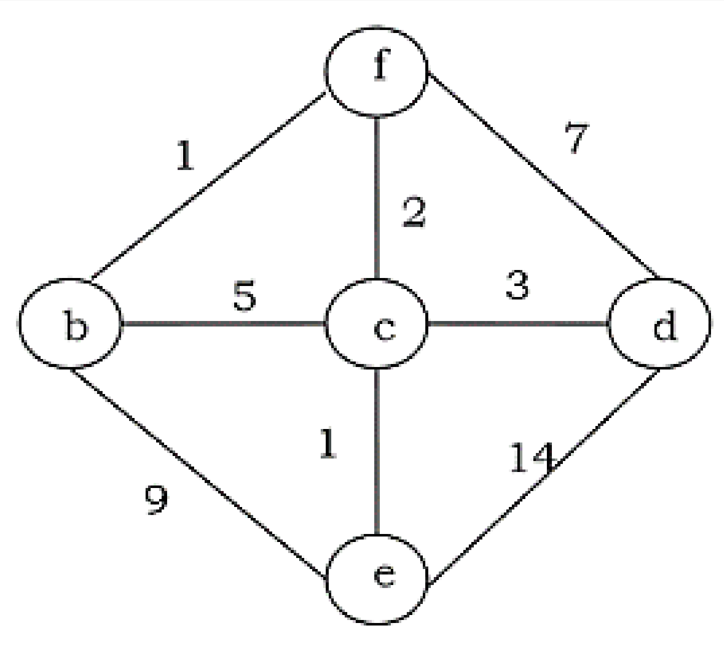
Recall: trees <u>cannot</u> form a cycle

Define MST



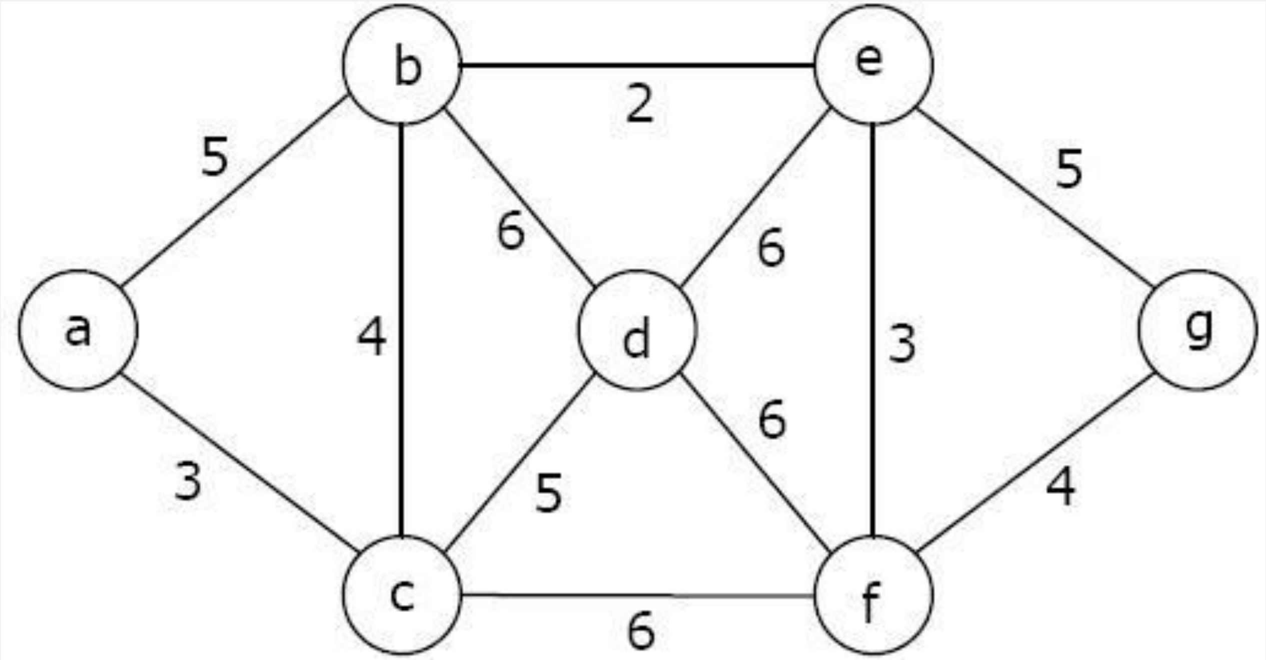


Define MST



A spanning tree for 5 vertices must have 4 edges.

TRY IT

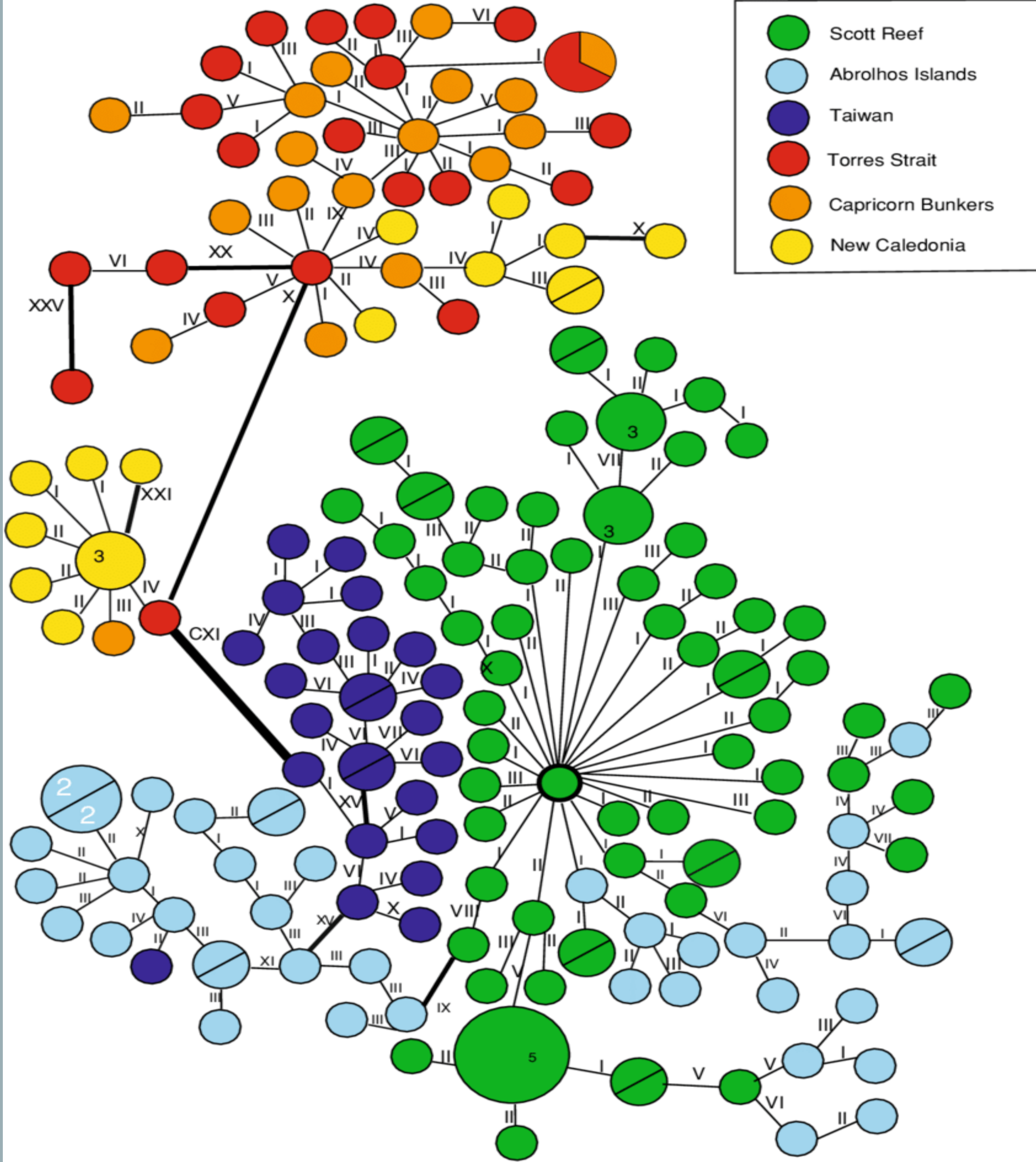


IF THERE ARE THOUSANDS OF VERTICES AND EDGES, HOW WILL IT BE SOLVED?

Algorithms:

Prim's Algorithm

Kruskal's Algorithm



PRIM'S ALGORITHM

To find the MST we can use Prim’s Algorithm (Greedy Algorithm)

- Basically, Prim’s algorithm is a modified version of Dijkstra’s algorithm. First, we choose a node to start from and add all its neighbors to a priority queue.

- Step 1:

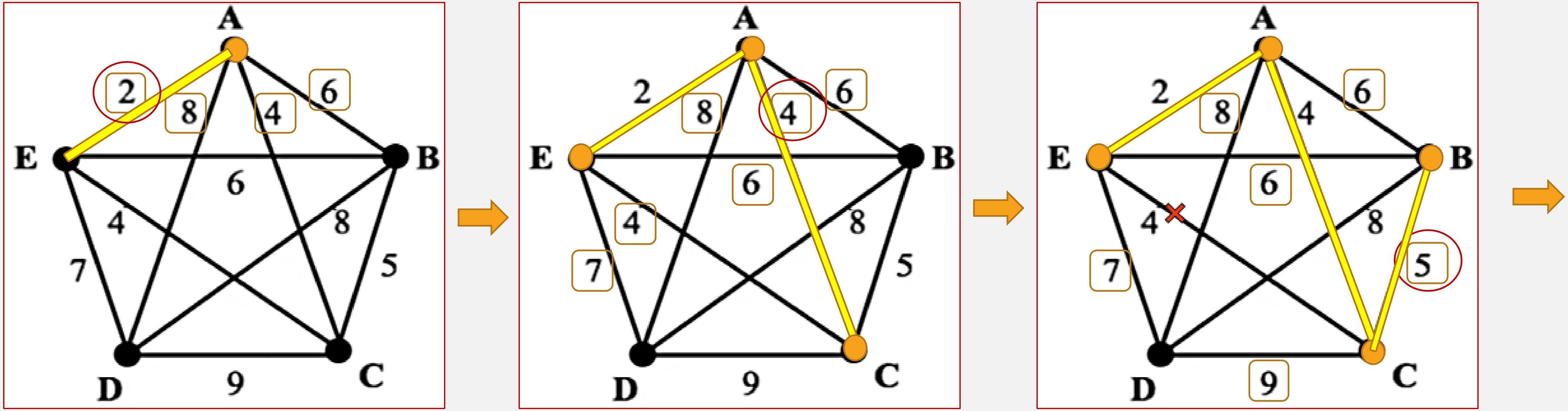
Select any node to be the first of T.

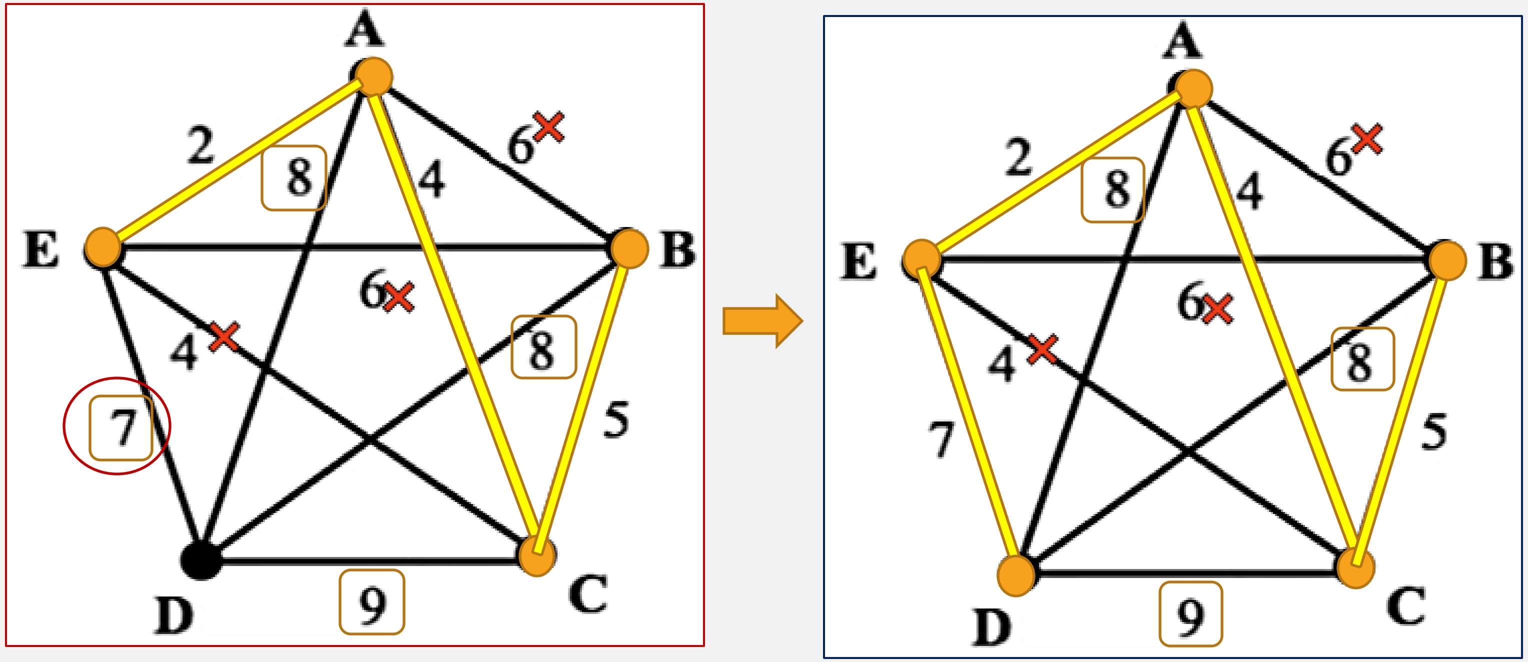
- Step 2:

Consider which arcs connects nodes in T to nodes outside T. Pick 1 with minimum weight (if more than 1, choose any). Add this arc and node to T.

- Step 3:

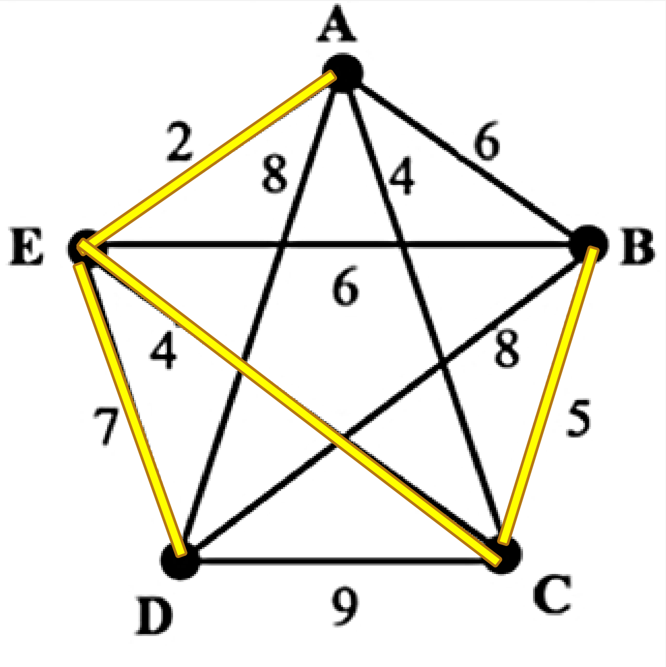
Repeat Step 2 until T contains every node of the graph.





Weight:

$7 + 2 + 4 + 5 = 18$



Converting a network to **matrix form**

- Step 1:

Select any node to be the first of T.

- Step 2:

Circle the new node of T in the top row and cross out the row corresponding to this new node.

- Step 3:

Find the smallest weight left in the columns of the nodes of T. Circle this weight. Then choose the node whose row the weight is in to join T. If several choose any.

- Step 4

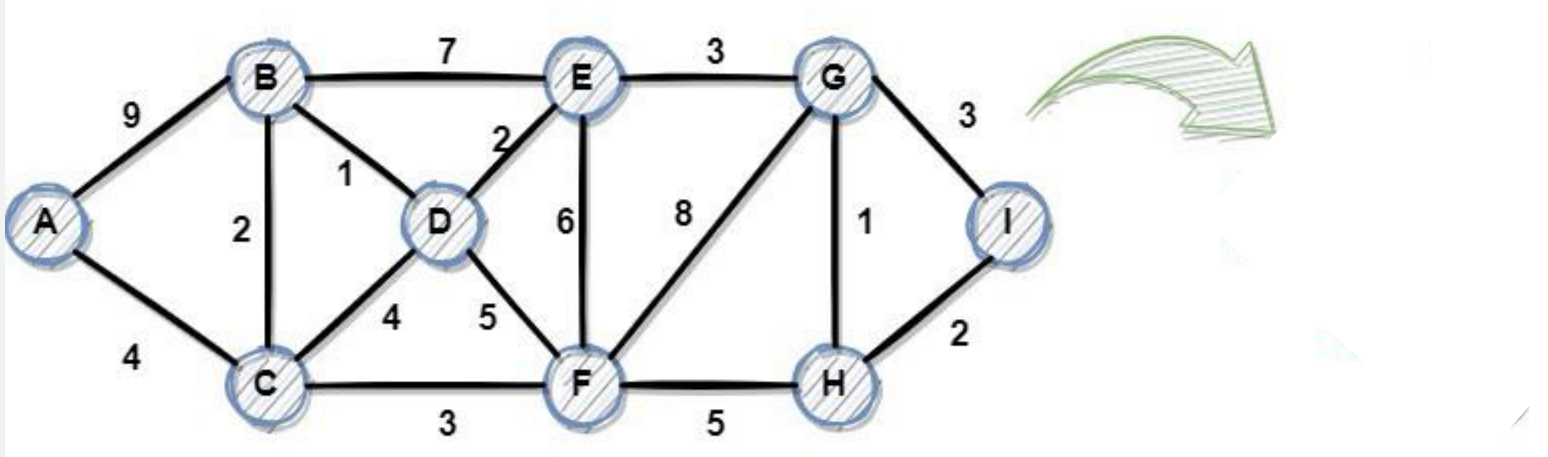
Repeat steps 2 and 3 until T contains every node.

	A	B	C	D	E
A	$-$	6	4	8	2
B	6	$-$	5	8	6
C	4	5	$-$	9	4
D	8	8	9	$-$	7
E	2	6	4	7	$-$

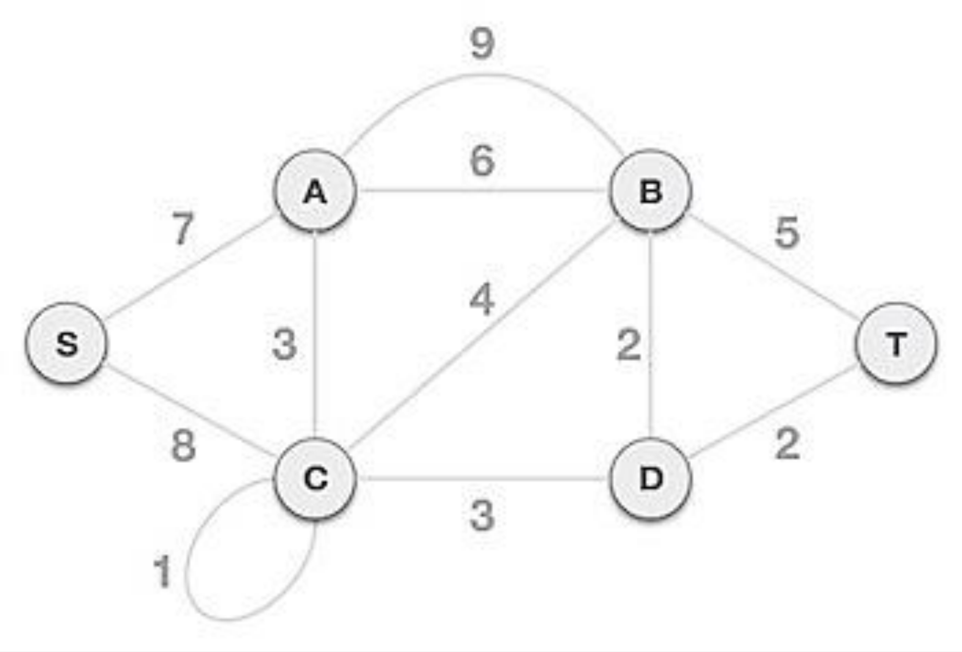
Weight:

$$7+2+4+5=18$$

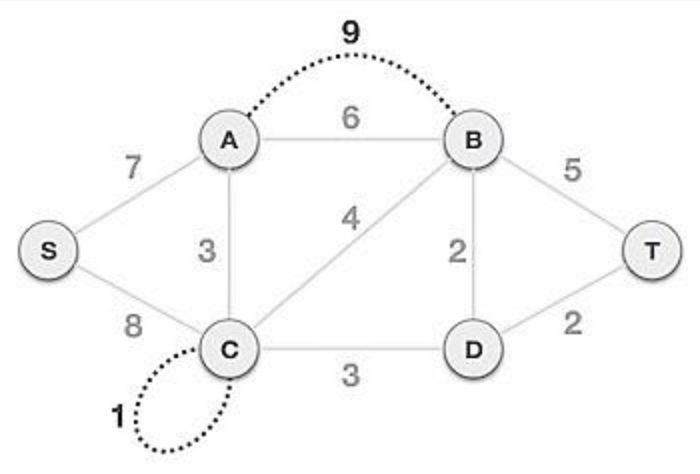
Prim's Algorithm



TRY THIS!

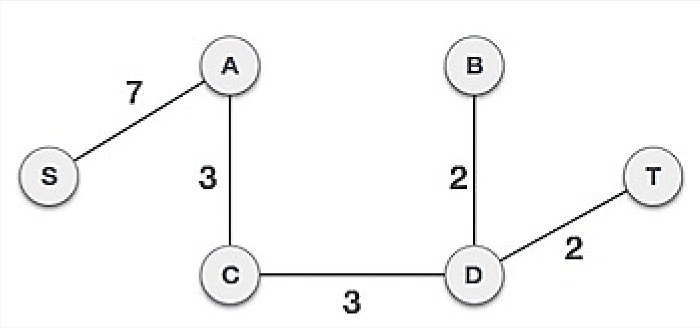


Step 1 - Remove all loops and parallel edges

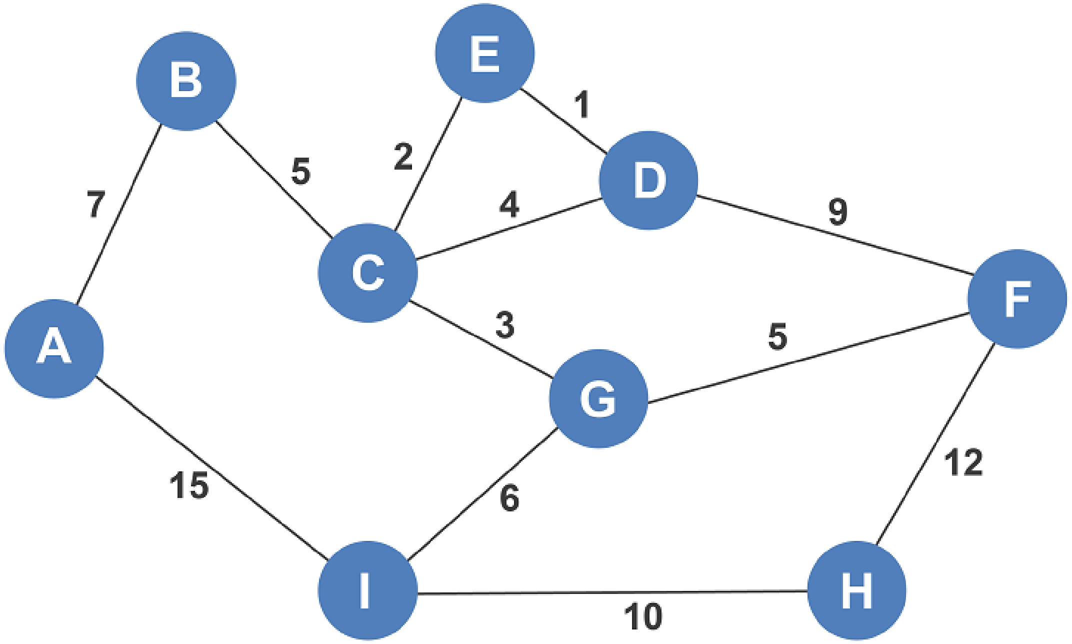


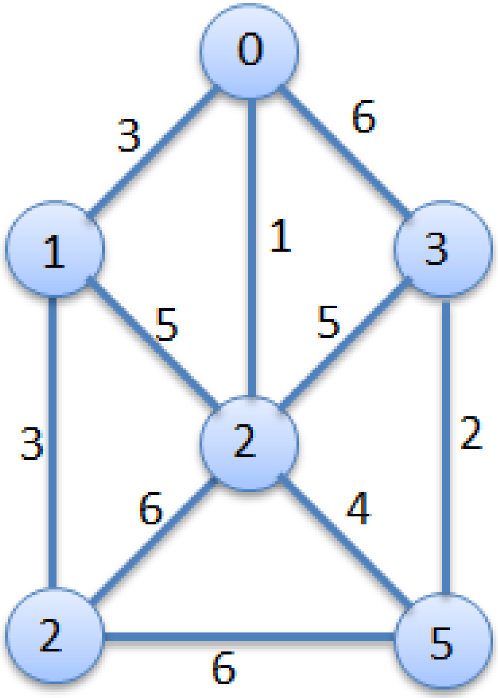
Step 2 - Choose any arbitrary node as root node

Step 3 - Check outgoing edges and select the one with less cost



Prim’s Algorithm





Consider the following pseudocode for Prim's algorithm.

```
Algorithm 2: Prim's Algorithm
Data: G: The given graph
      source: The node to start from
Result: Returns the cost of the MST
totalCost ← 0;
included ← {false };
Q.addOrUpdate(source, 0, Φ);
while ¬ Q.empty() do
    u ← Q.getNodeWithLowestWeight();
    totalCost ← totalCost + u.weight;
    if u.edge ≠ Φ then
        mst.add(u.edge);
    end
    included[u.node] ← true;
    for v ∈ G.neighbors(u.node) do
        if ¬included[v.node] then
            Q.addOrUpdate(v.node, weight(u.node, v.node),
             v.edge);
        end
    end
end
return totalCost, mst;
```

KRUSKAL'S ALGORITHM

To find the MST of a graph with n nodes

- Step 1:

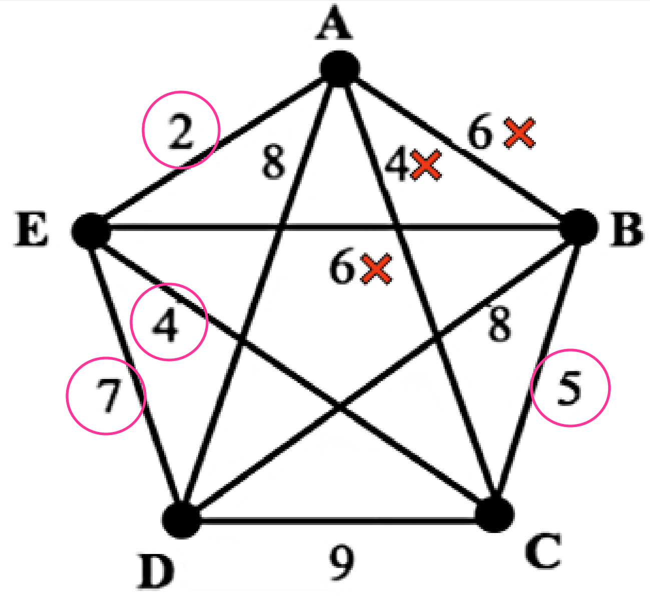
Choose the arc of least weight.

- Step 2:

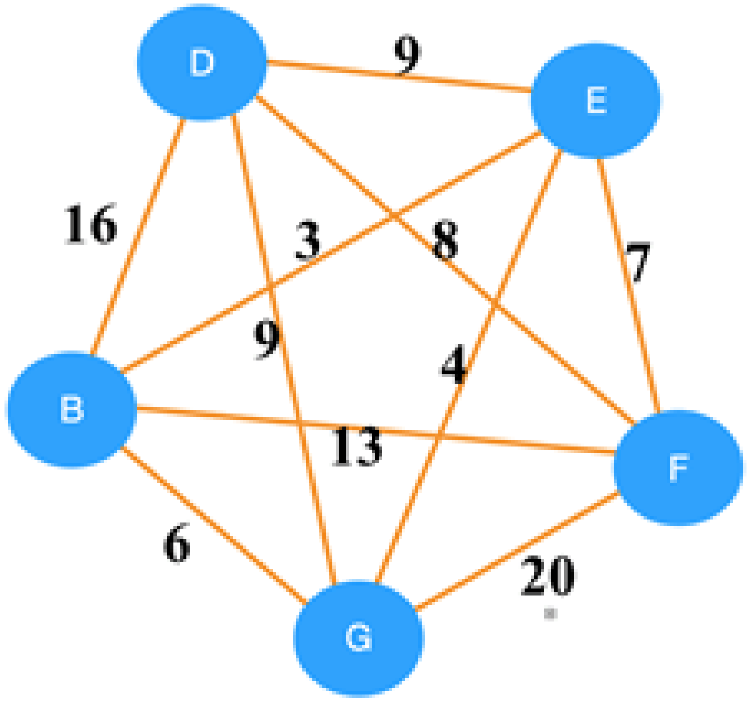
Choose from those arcs remaining the arc of least weight which do not form a cycle. (if more than 1, choose any)

- Step 3:

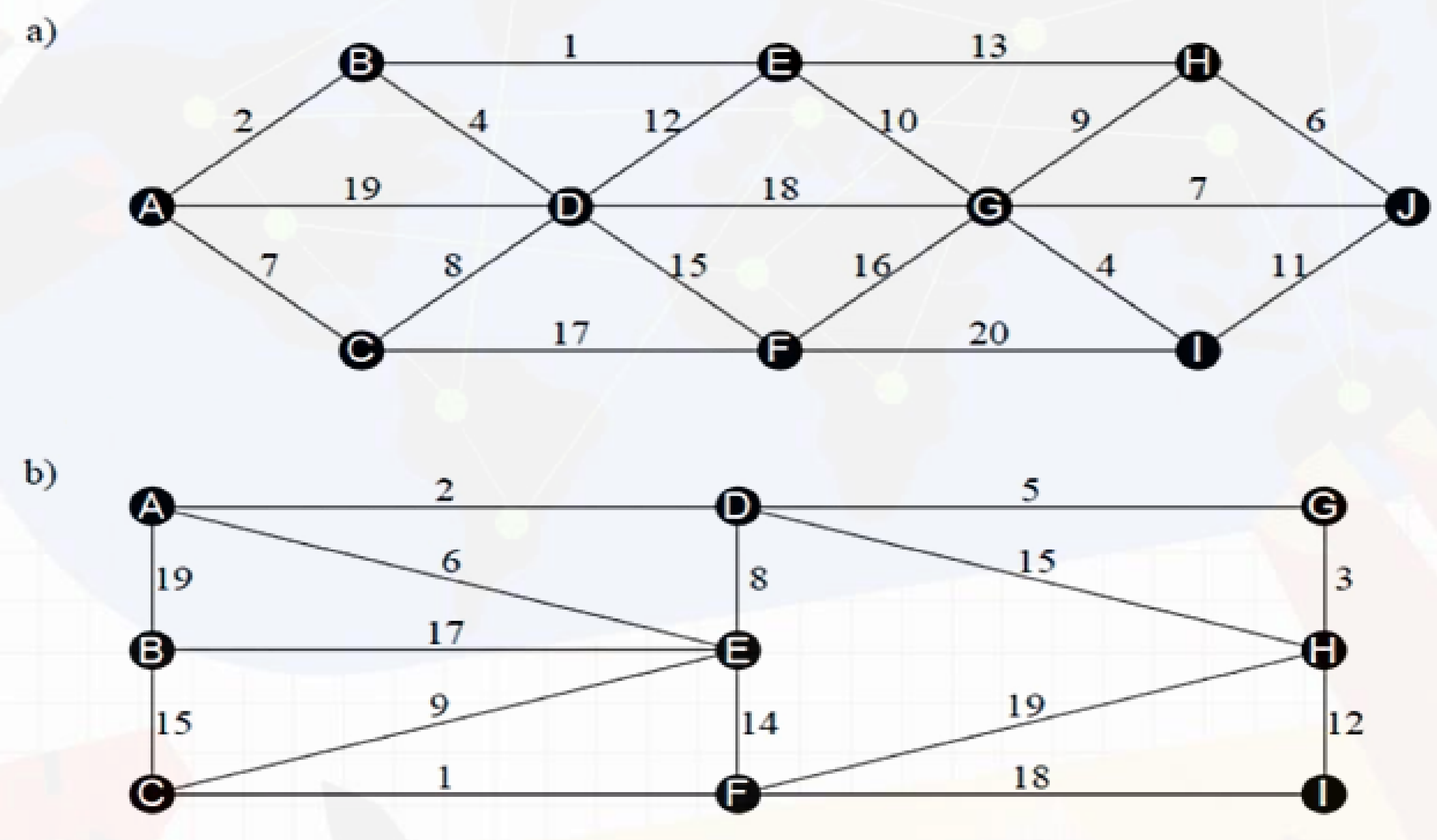
Repeat Step 2 until (n-1) arcs have been chosen.



$$\left.\begin{aligned} AE &= 2 \\ EC &= 4 \\ ED &= 7 \\ BC &= 5 \end{aligned}\right\} = 18$$


D
E
B
F
G
9
16
3
8
7
9
4
13
6
20

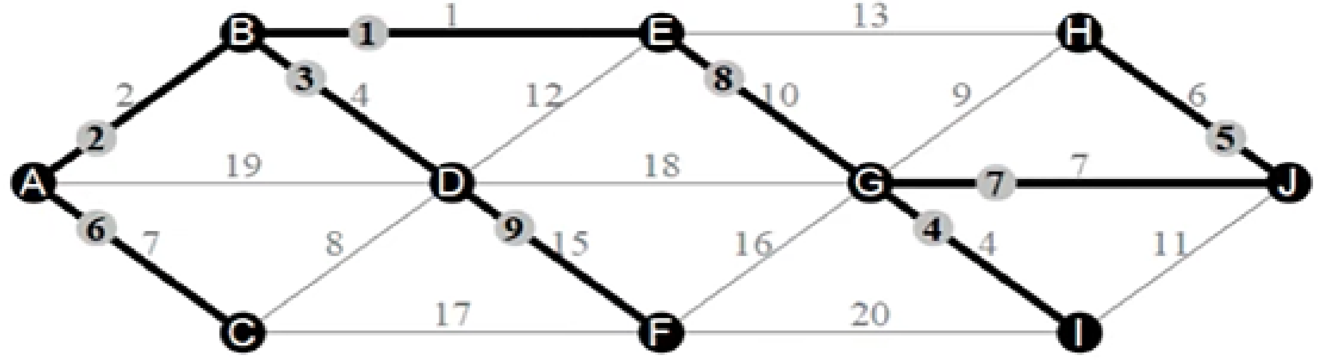
Find the Minimum Spanning Tree using Kruskal's Algorithm:



Answer

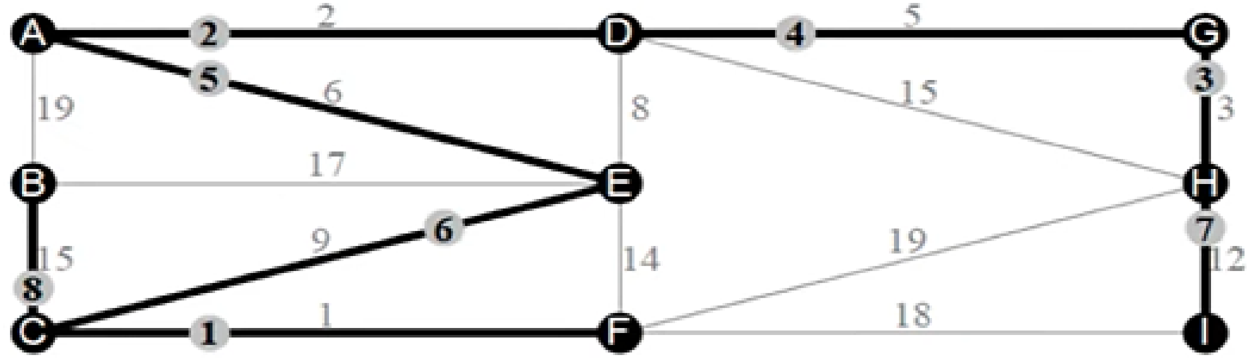
Find the Minimum Spanning Tree using Kruskal's Algorithm:

a)



Arcs: BE, AB, BD, GI, JH, AC, GJ, EG, DF. Total length=56

b)



Arcs: CF, AD, GH, DG, AE, EC, HI, CB. Total length=53

Take a look at the pseudocode for Kruskal's algorithm.

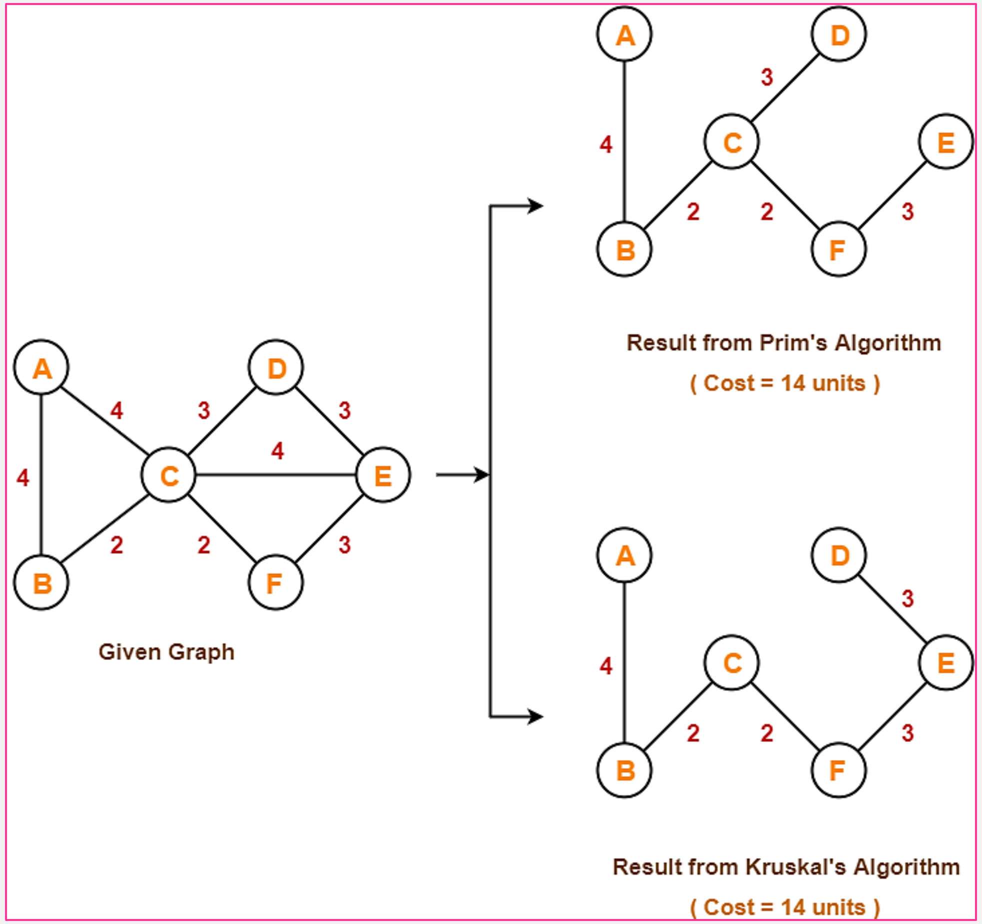
Algorithm 1: Kruskal Algorithm

```
Data: edges: List of edges of the graph
Result: Returns the cost and the edges of the MST
sort(edges);
totalCost ← 0;
for edge ∈ edges do
    if ¬ dsu.isMerged(edge.u, edge.v) then
        totalCost ← totalCost + edge.weight;
        mst.add(edge);
        dsu.merge(edge.u, edge.v);
    end
end
return totalCost, mst;
```

What is difference between Prims and Kruskal algorithm?

- Prim's Algorithm grows a solution **from a random vertex** by adding the next cheapest vertex to the existing tree.

- Kruskal's Algorithm grows a solution **from the cheapest** edge by adding the next cheapest edge to the existing tree / forest.

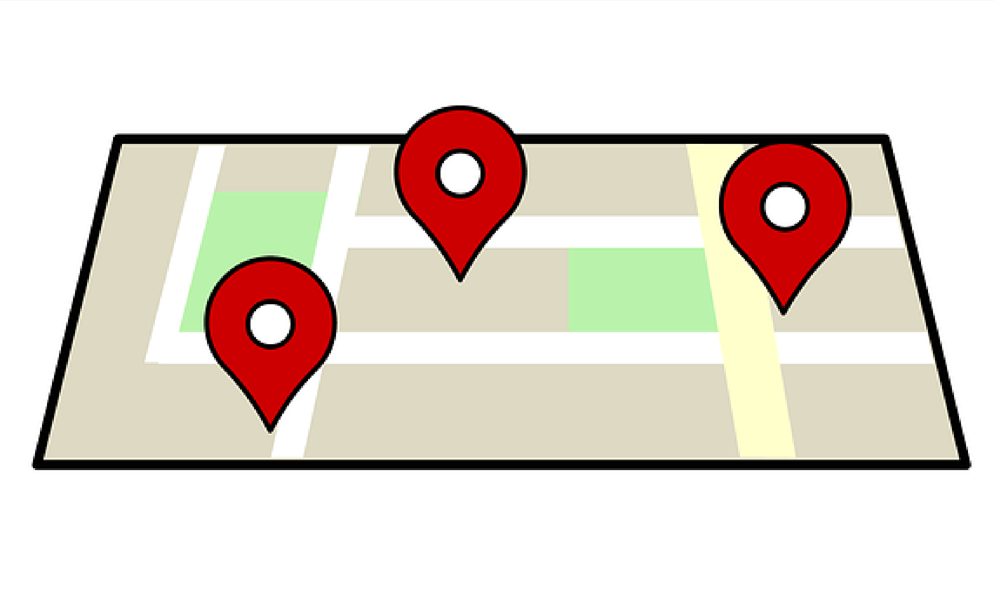


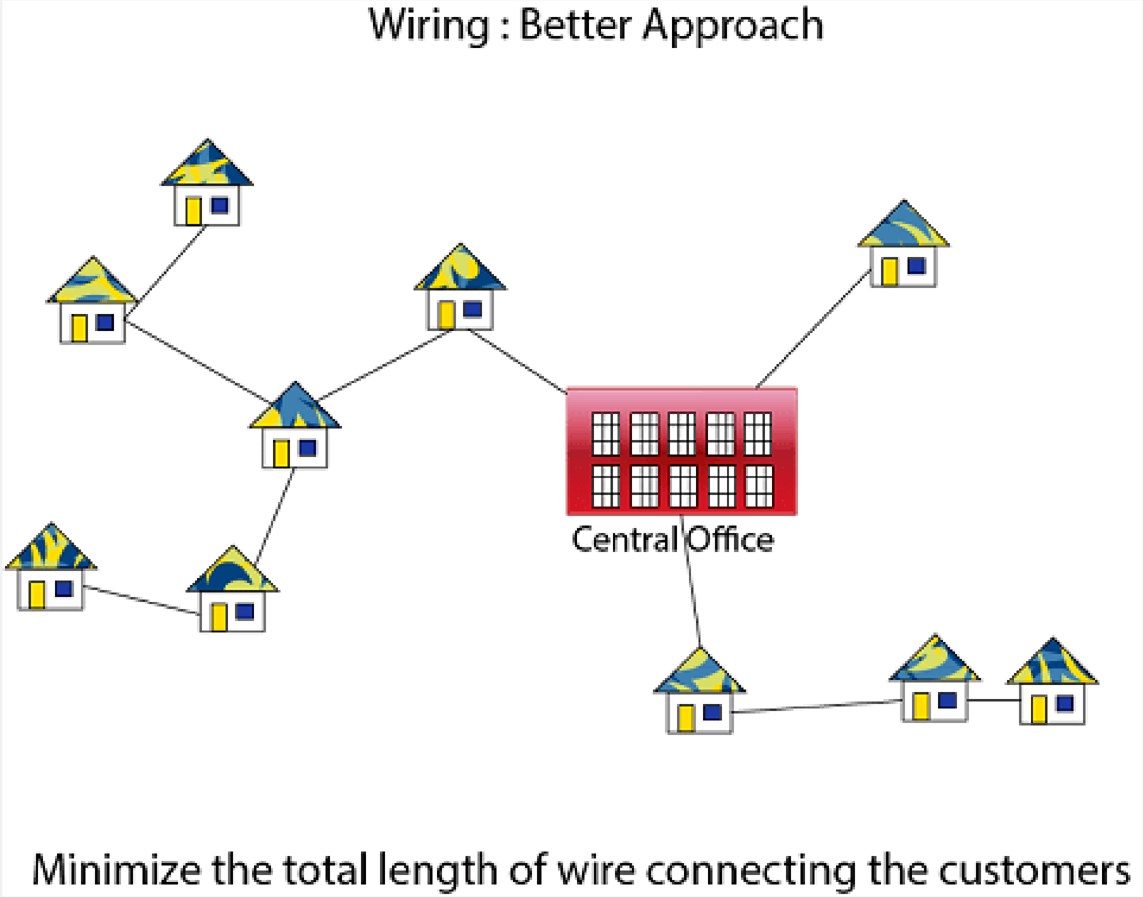
Let’s highlight some key differences between the two algorithms.

	Kruskal	**Prim**
Multiple MSTs	Offers a good control over the resulting MST	Controlling the MST might be a little harder
Implementation	Easier to implement	Harder to implement
Requirements	Disjoint set	Priority queue
Time Complexity	$O(E \cdot log(V))$	$O(E + V \cdot log(V))$

As we can see, **the Kruskal algorithm is better to use regarding the easier implementation and the best control over the resulting MST**. However, **Prim’s algorithm offers better complexity.**

PRIM'S & KRUSKAL'S ALGORITHMS IN REAL LIFE





APPLICATIONS

APPLICATIONS WHERE KRUSKAL'S ALGORITHM IS GENERALLY USED:

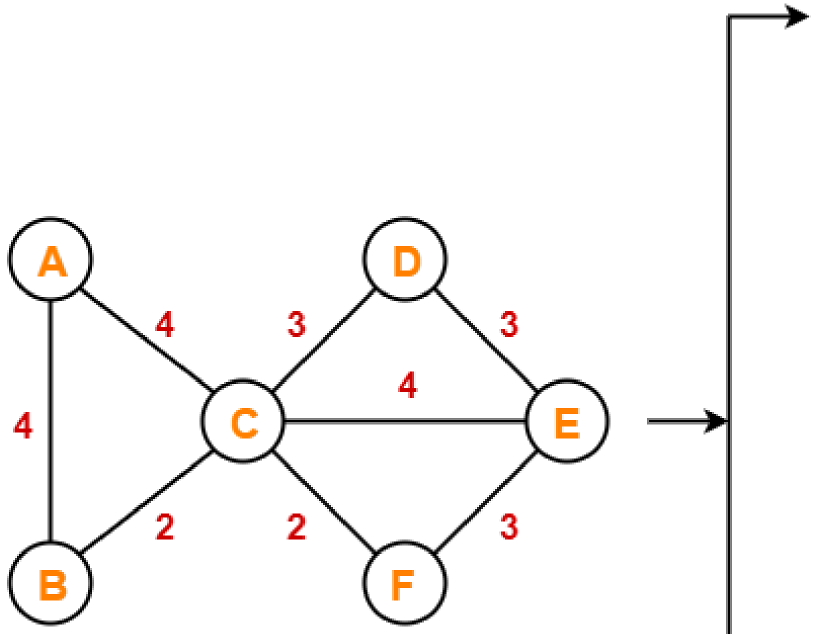
- 1. Landing cables
- 2.TV Network
- 3.Tour Operations
- 4. LAN Networks
- 5.A network of pipes for drinking water or natural gas.
- 6.An electric grid
- 7. Single-link Cluster

APPLICATIONS WHERE PRIM'S ALGORITHM IS GENERALLY USED:

- 1.All the applications stated in the Kruskal's algorithm's applications can be resolved using Prim's algorithm (use in case of a dense graph).
- 2. Network for roads and Rail tracks connecting all the cities.
- 3. Irrigation channels and placing microwave towers
- 4. Designing a fiber-optic grid or ICs.
- 5.Travelling Salesman Problem.
- 6. Cluster analysis.
- 7. Pathfinding algorithms used in AI(Artificial Intelligence).
- 8. Game Development
- 9. Cognitive Science

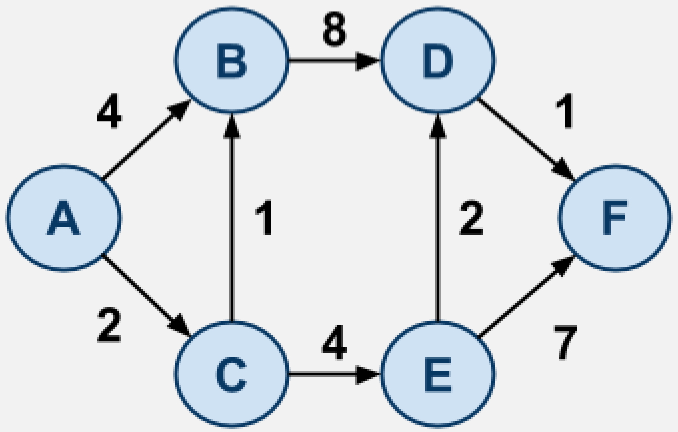
Time for Practice

Find MST using:
a) Prim's Algorithm
b) Kruskal's Algorithm

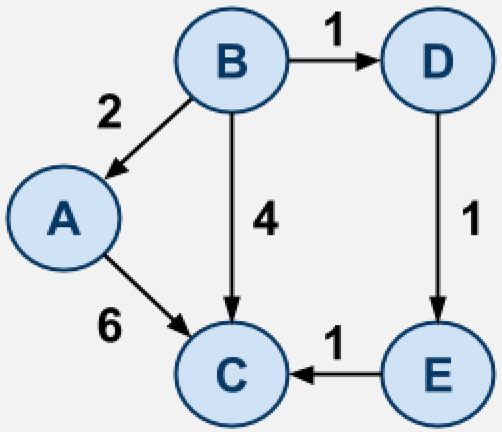


Given Graph

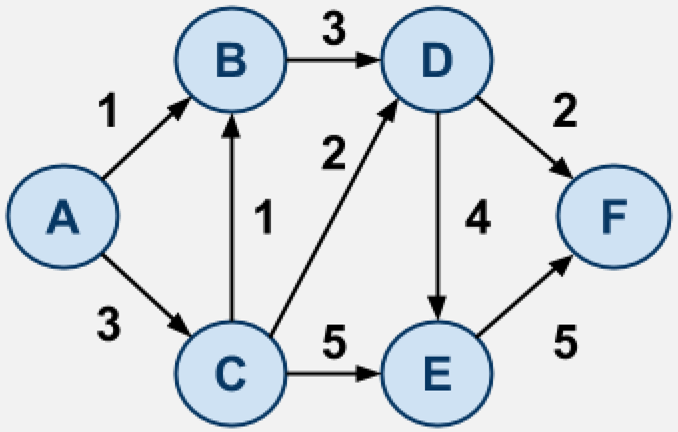
Graph 0



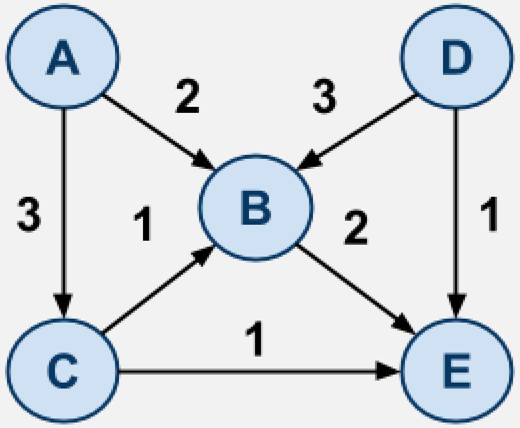
Graph 1

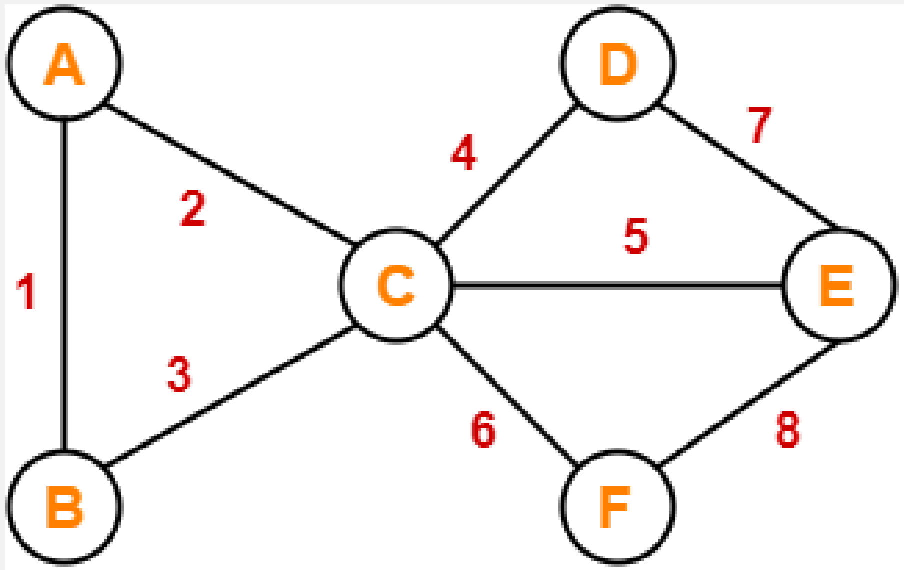


Graph 2



Graph 3

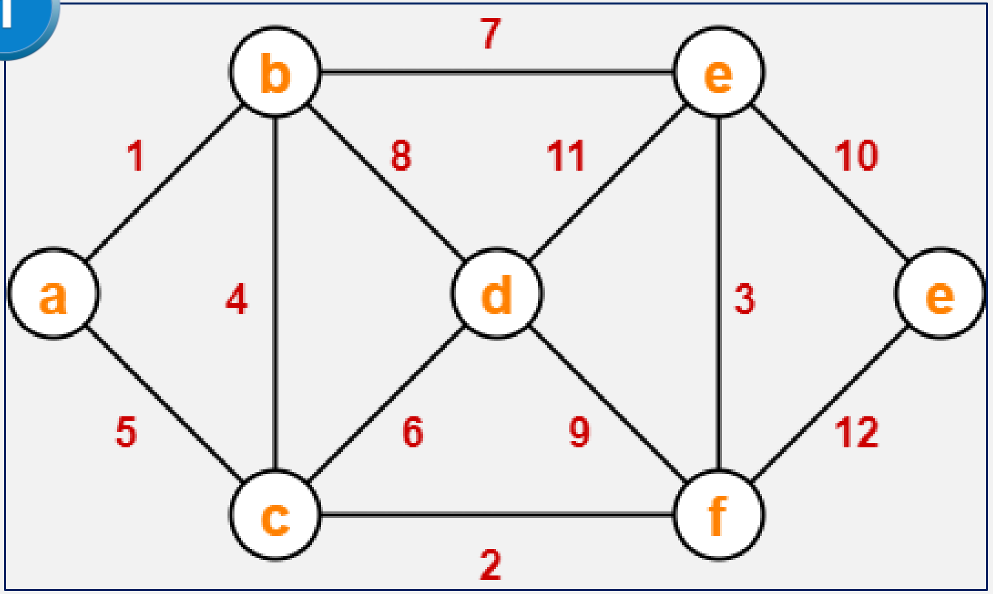




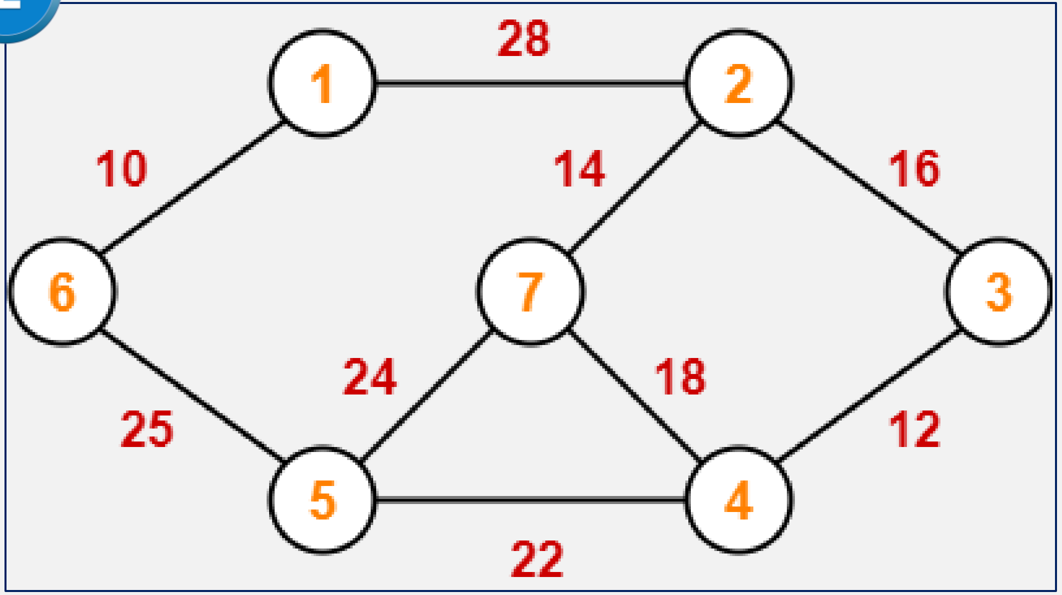
Given Graph

Homework

1



2



3