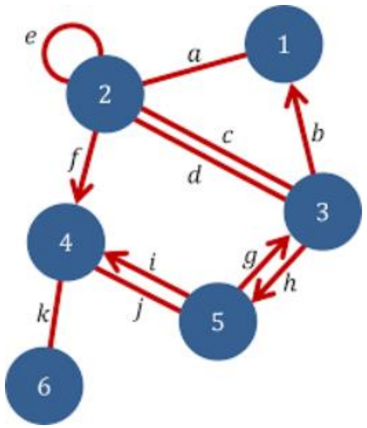
Data Structures and Algorithms

Graph



Prepared by:

Eng. Malek Al-Louzi

School of Computing and Informatics– Al Hussein Technical University

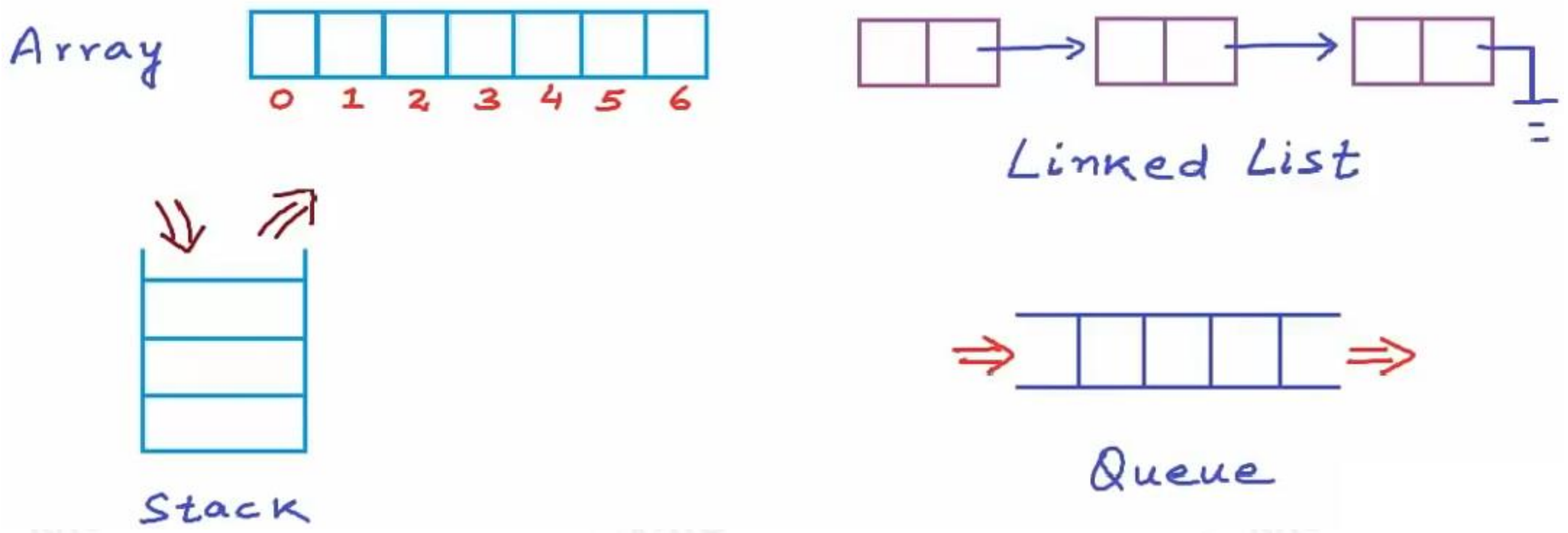
Spring 2021/2022

Outlines

- Introduction to Graph
- Graph Properties
- Applications of Graph
- Weighted Graph
- Graph Representation

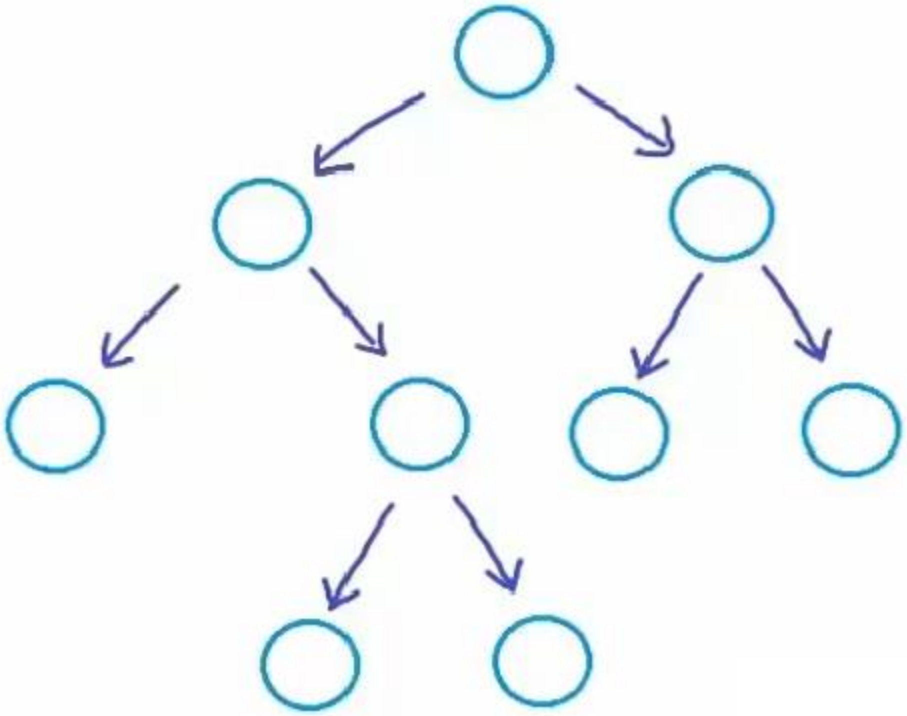
Introduction to Graph

- So far, we learned about Linear data structures:



Introduction to Graph

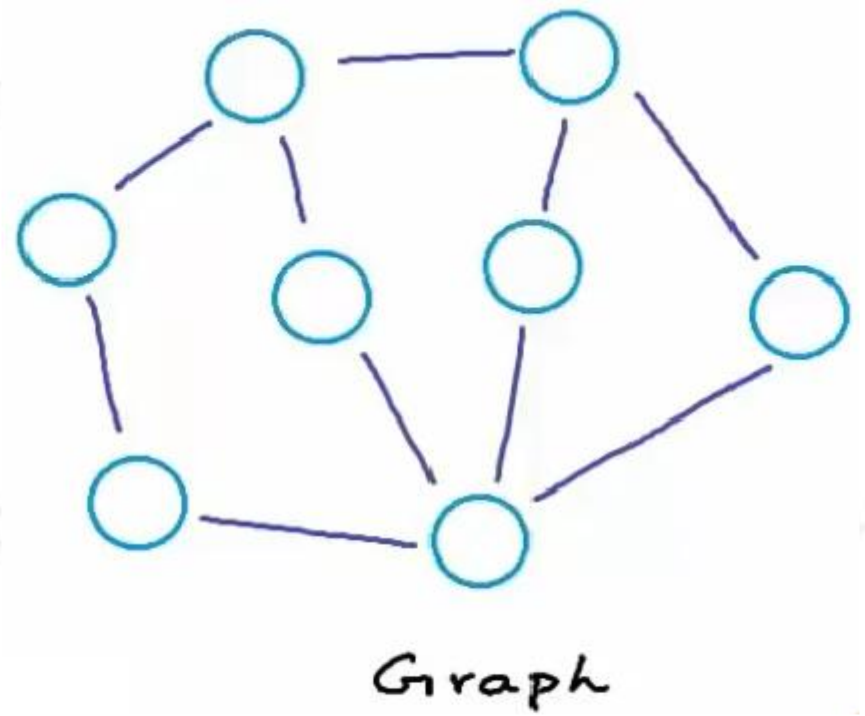
- Also, we learned about Tree which is a nonlinear data structure:



Tree

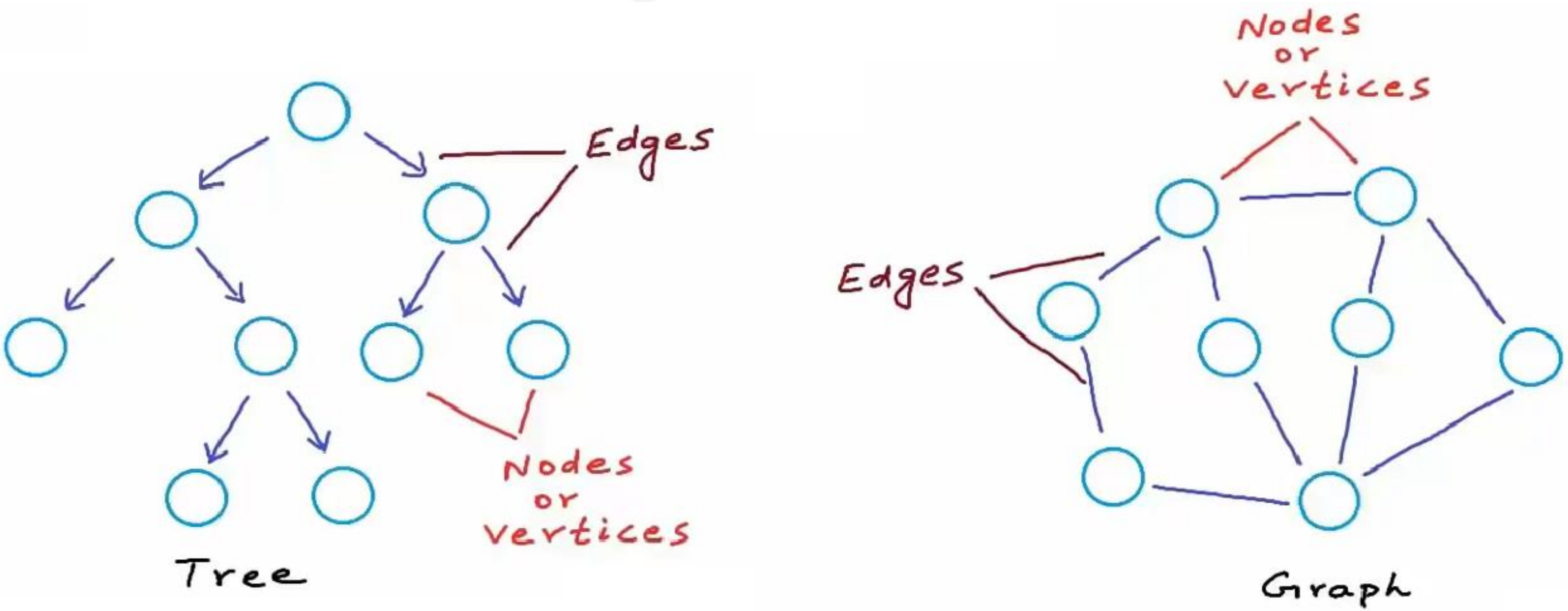
Introduction to Graph

- Now, we will study another nonlinear data structure that has got its application in a wide number of scenarios in computer science.
- This data structure is used to model and represent a variety of systems.
- This data structure is Graph.



Introduction to Graph

- A graph just like a tree, is a collection of objects that we call nodes or vertices, connected to each other through a set of edges.



Introduction to Graph

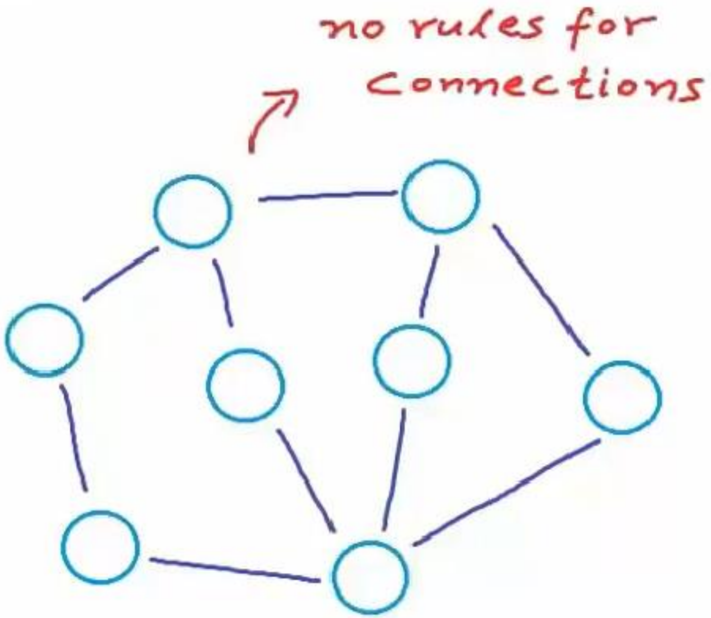
- But in a tree, connections are bound to be in a certain way.
  - There are rules dictating the connections among the nodes.

if N nodes
then (N-1) edges
one edge for each
parent-child relationship

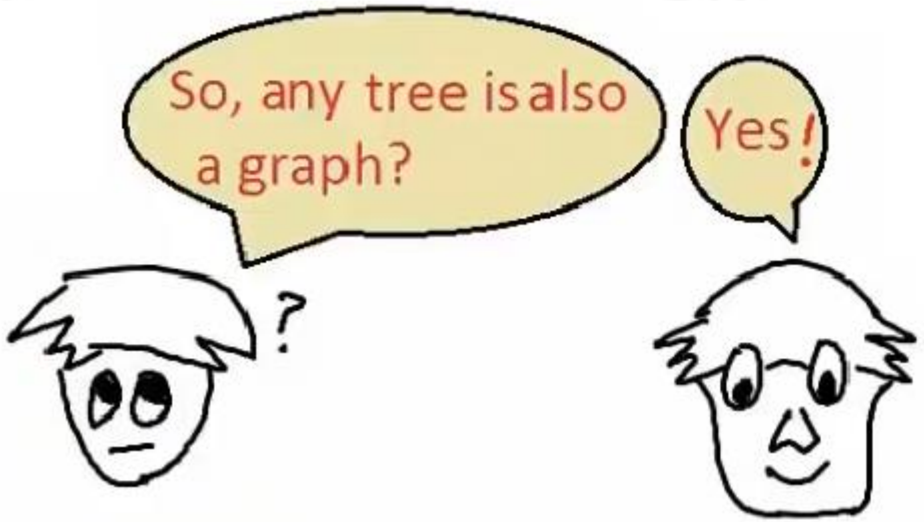
- All nodes must be reachable from the root and there must be exactly one possible path from root to a node.

Introduction to Graph

- Now, in a graph there are no rules dictating the connections among the nodes.
- A graph contains a set of nodes and a set of edges.
- Edges can be connecting nodes in any possible way.



Introduction to Graph



- Tree is only a special kind of graph.
- The study of graphs is often referred to as **graph theory**.

Introduction to Graph

- In Mathematics:

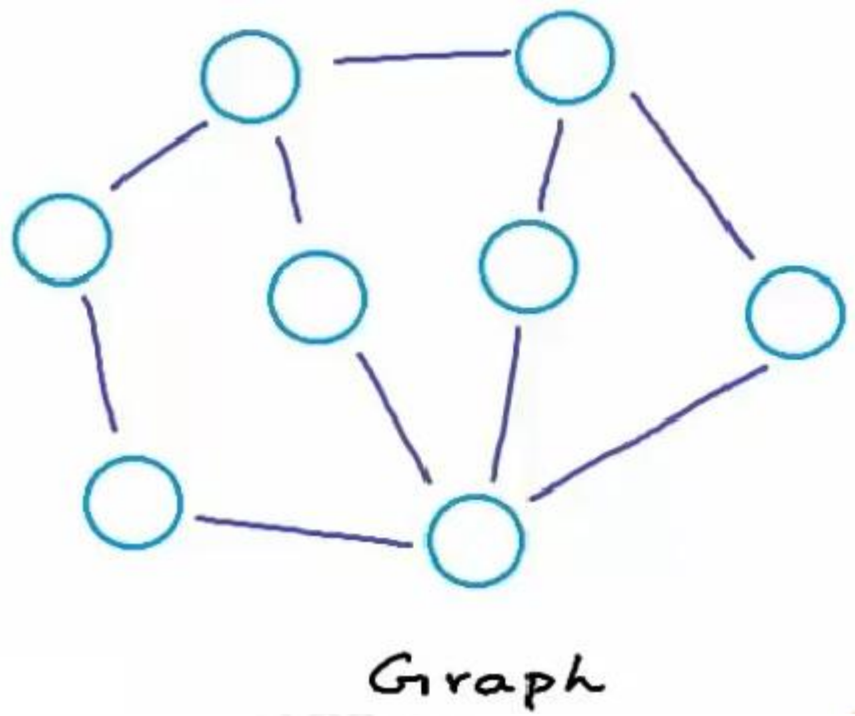
Graph:

A graph G is an ordered pair of a set V of vertices and a set E of edges.

$$G = (V, E)$$

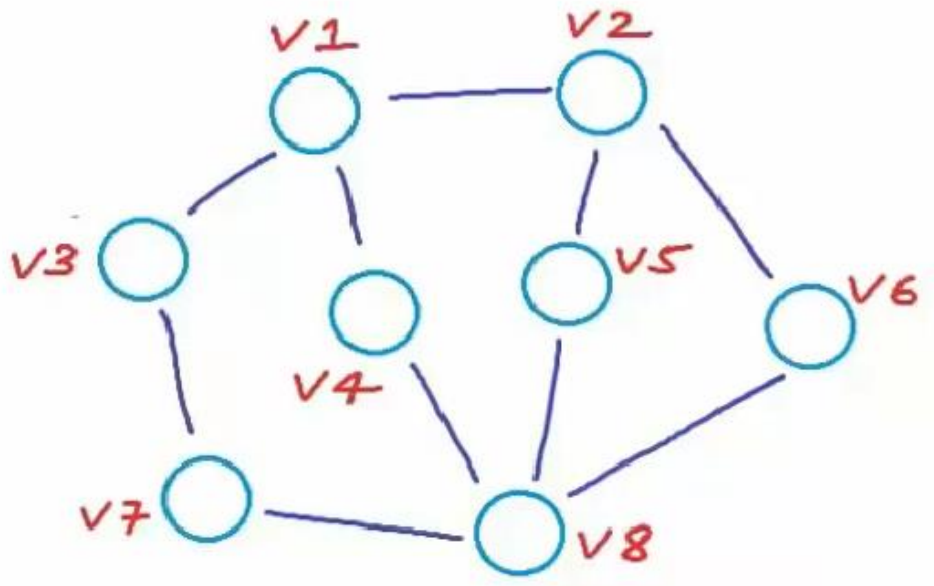
Introduction to Graph

- We have the following graph which contains 8 vertices and 10 edges.



Introduction to Graph

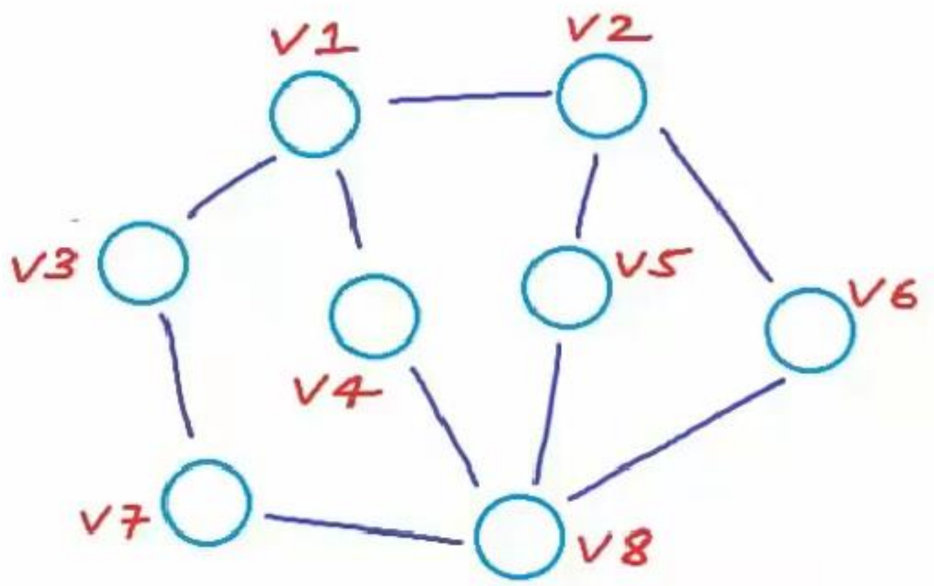
- Let us give some names to these vertices.
- This naming is not indicative of any order, there is no 1st, 2nd and 3rd node here.
- We could give any name to any node.



Introduction to Graph

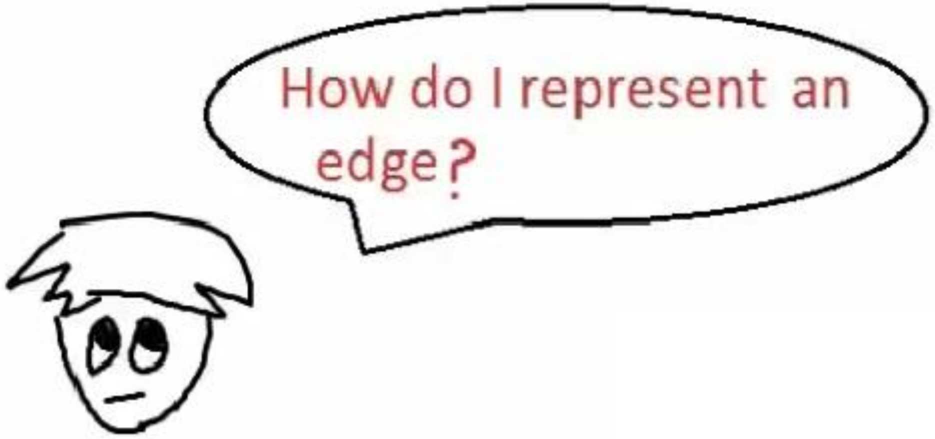
- So, the set of vertices for this graph is:

$$V = \{v1, v2, v3, v4, v5, v6, v7, v8\}$$



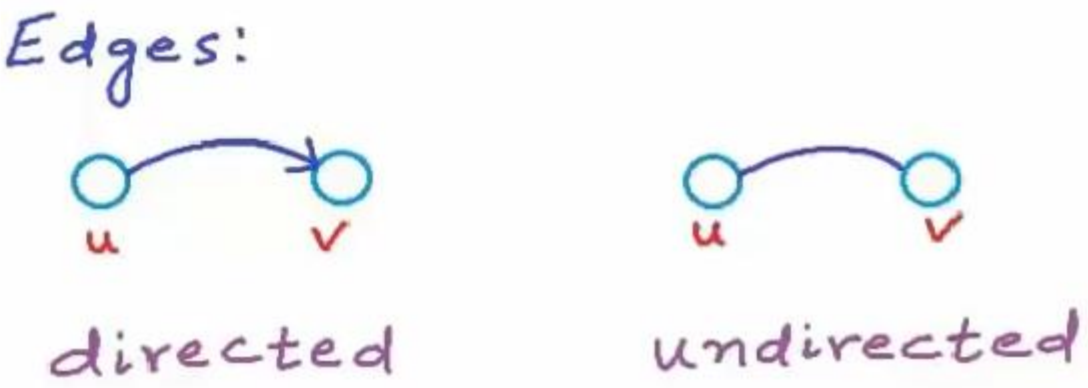
Introduction to Graph

- Now, what is the set of edges for this graph?
- To answer this, we first need to know how to represent an edge.



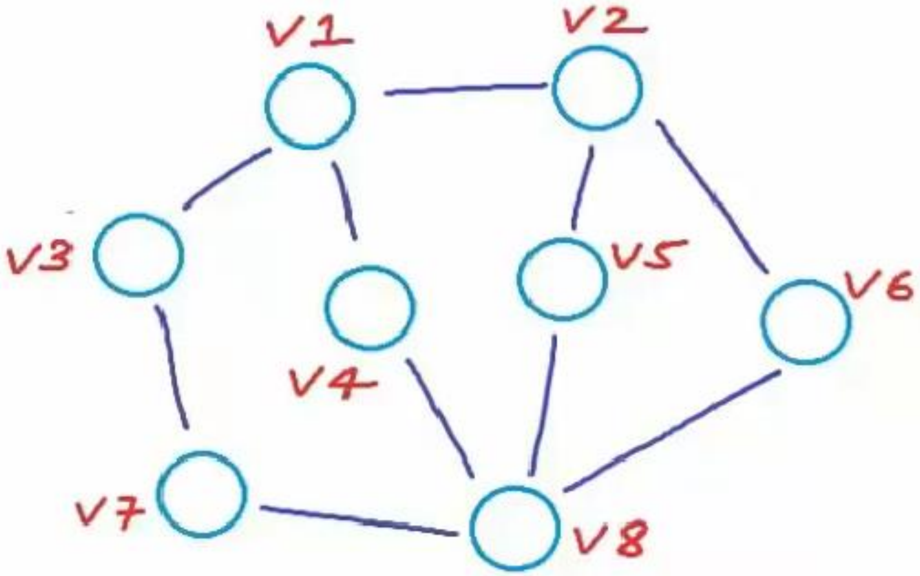
Introduction to Graph

- An edge is uniquely identified by its two endpoints.
- So, we can just write the names of the two endpoints of an edge as a pair.
- But edges can be of two types.



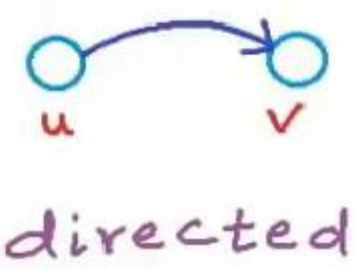
Introduction to Graph

- In directed edge, the connection is one-way.
- In undirected edge, the connections is two-way.
- In our example the edges of the graph are undirected.



Introduction to Graph

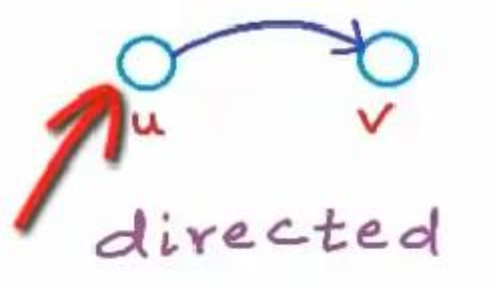
- In tree, the edges are directed.
- In the following directed edge, we are saying that there is link or path from vertex U to V, but we cannot assume a path from V to U.



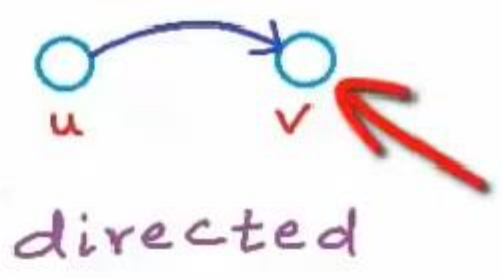
- This connection is one-way.
- For a directed edge, one of the endpoints would be the origin and other endpoint would be the destination.

Introduction to Graph

- We draw the directed edge with an arrowhead pointing towards the destination.
- For our edge here, Origin is **u**.

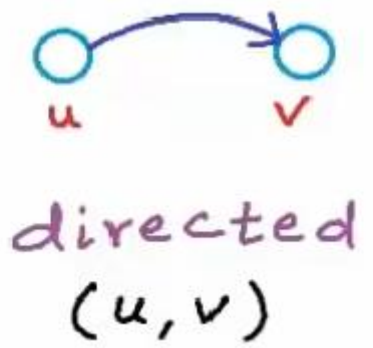


- And destination is **v**.



Introduction to Graph

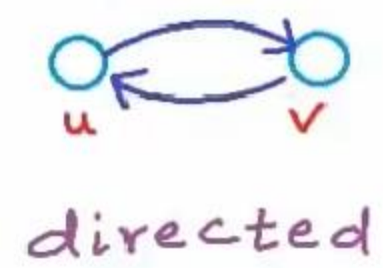
- A directed edge can be represented as an ordered pair.



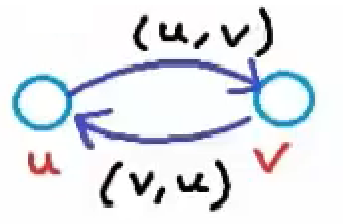
- First element in the pair is the origin and second element is the destination.

Introduction to Graph

- If we want a path from V to U, we need to draw another directed edge with V as origin and U as destination.



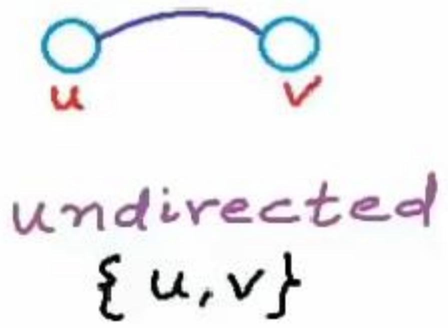
- This new edge can be represented also as ordered pair.



- These two edges are not same.

Introduction to Graph

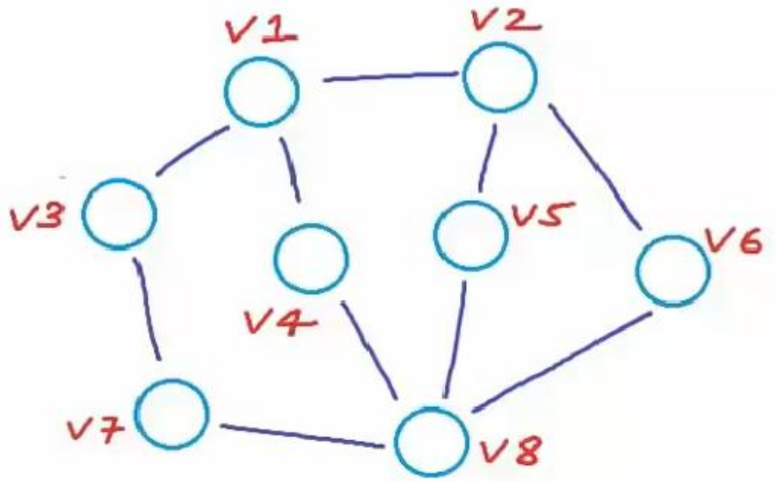
- If the edge is undirected, the connection is two-way.
- Undirected edge can be represented as unordered pair.



- Because the edge here is bidirectional, origin and destination are not fixed.
- We only need to know what two endpoints have been connected by the edge.

Introduction to Graph

- Now we know how to represent edges, we can write the set of edges for our example graph.



$$V = \{v1, v2, v3, v4, v5, v6, v7, v8\}$$

$$E = \{\{v1, v2\}, \{v1, v3\}, \{v1, v4\}, \{v2, v5\}, \{v2, v6\}, \{v3, v7\}, \{v4, v8\}, \{v7, v8\}, \{v5, v8\}, \{v6, v8\}\}$$

Graph Properties

- To denote number of elements in a set (cardinality of a set), we use the same notation that we used for absolute value.

$|V| \rightarrow$ number of vertices

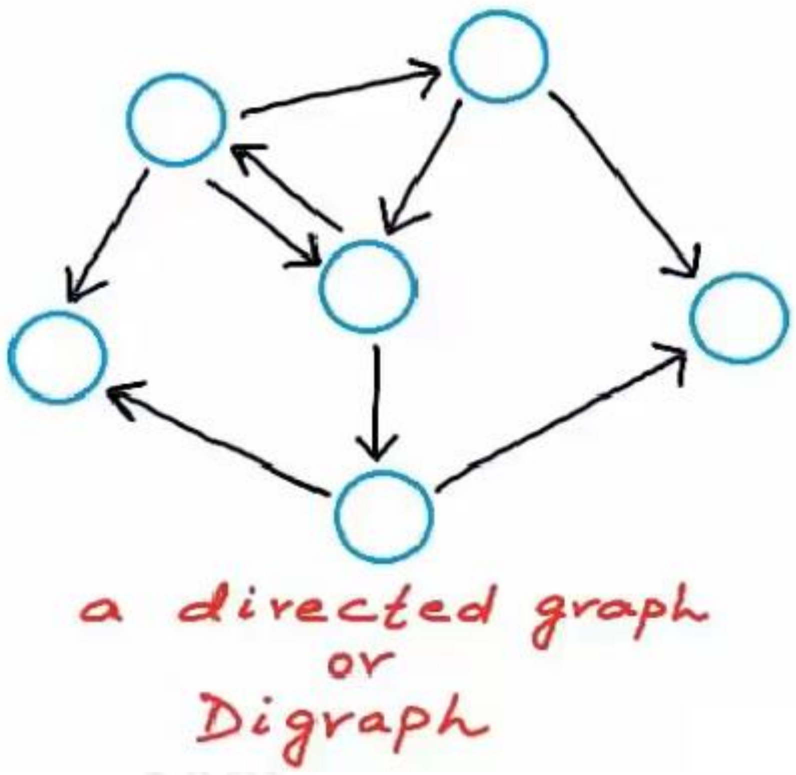
$|E| \rightarrow$ number of edges

Graph Properties

- Typically, in a graph all edges would either be directed or undirected.
- It is possible for a graph to have both directed and undirected edges.
- We will focus only on graphs in which all edges would either be directed or undirected.

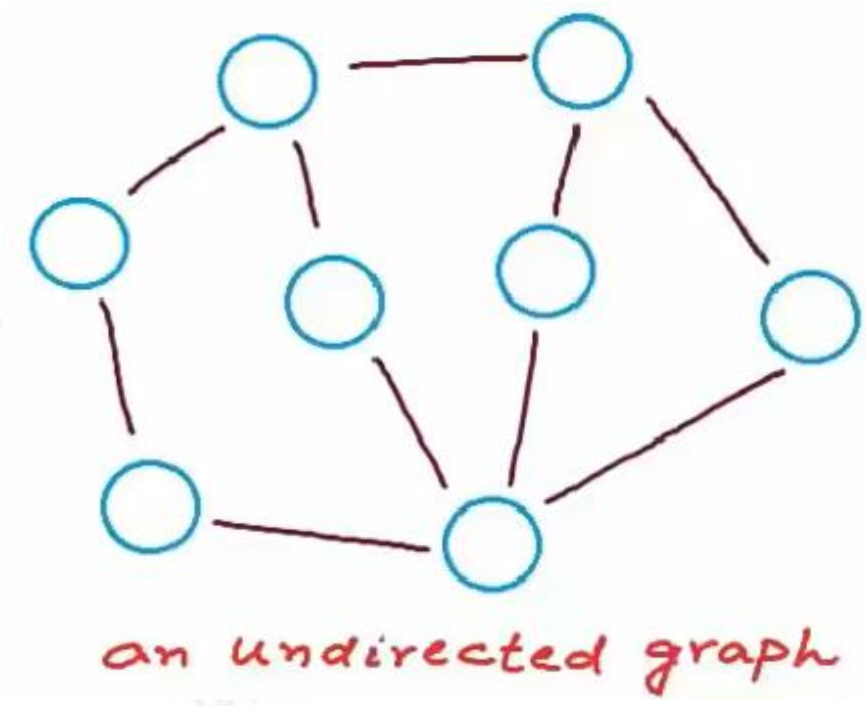
Directed vs Undirected Graph

- A graph with all directed edges is called a directed graph or Digraph.



Graph Properties

- A graph with all undirected edges is called an undirected graph.
- There is no special name for an undirected graph.

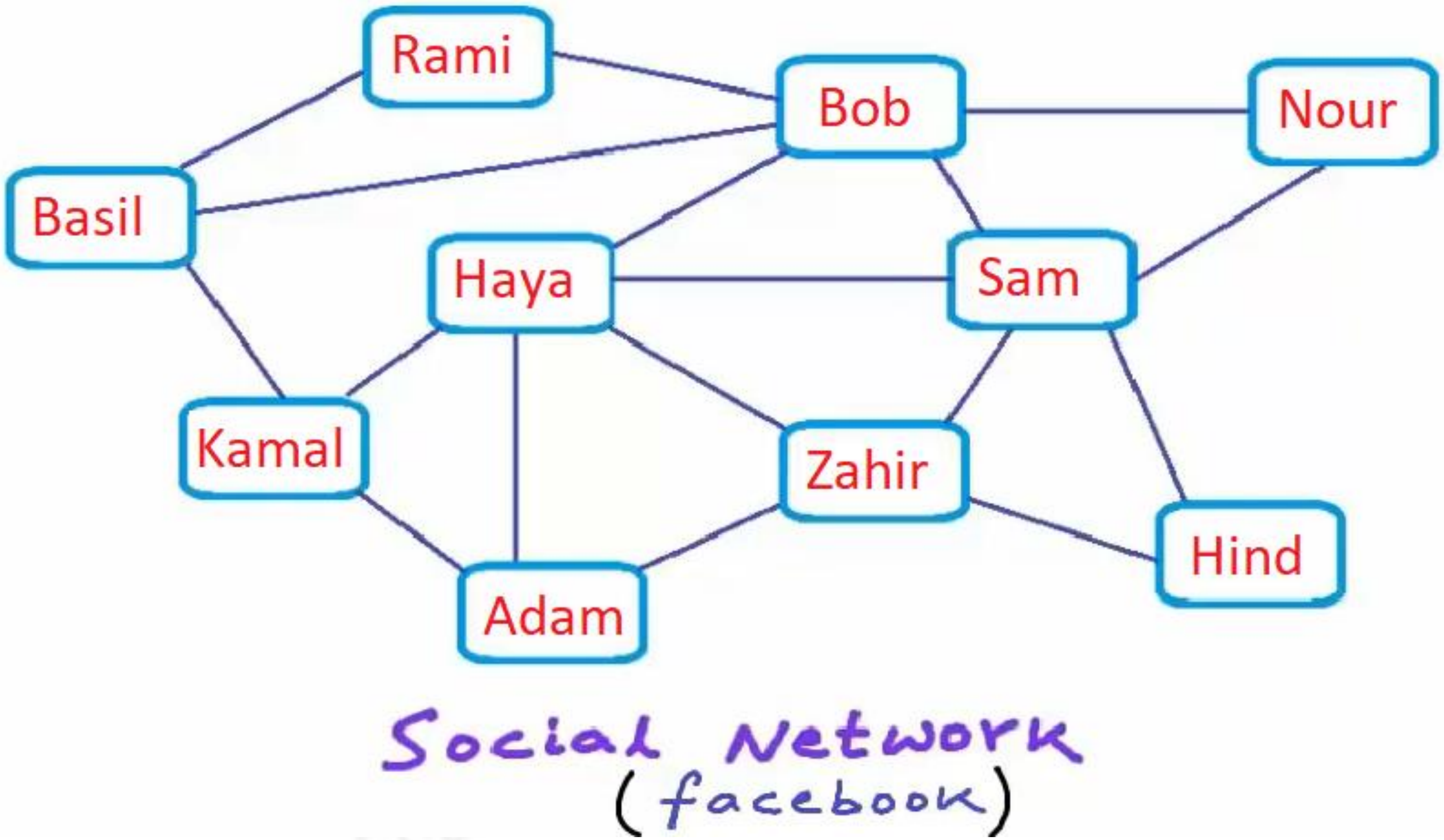


Applications of Graph

- Many real-world systems and problems can be modeled using a graph.
- Graphs can be used to represent any collection of objects that having pairwise relationship.
- Let us have a look at some of the interesting examples.

Applications of Graph
Social Network

- A social network like Facebook can be represented as an undirected graph.

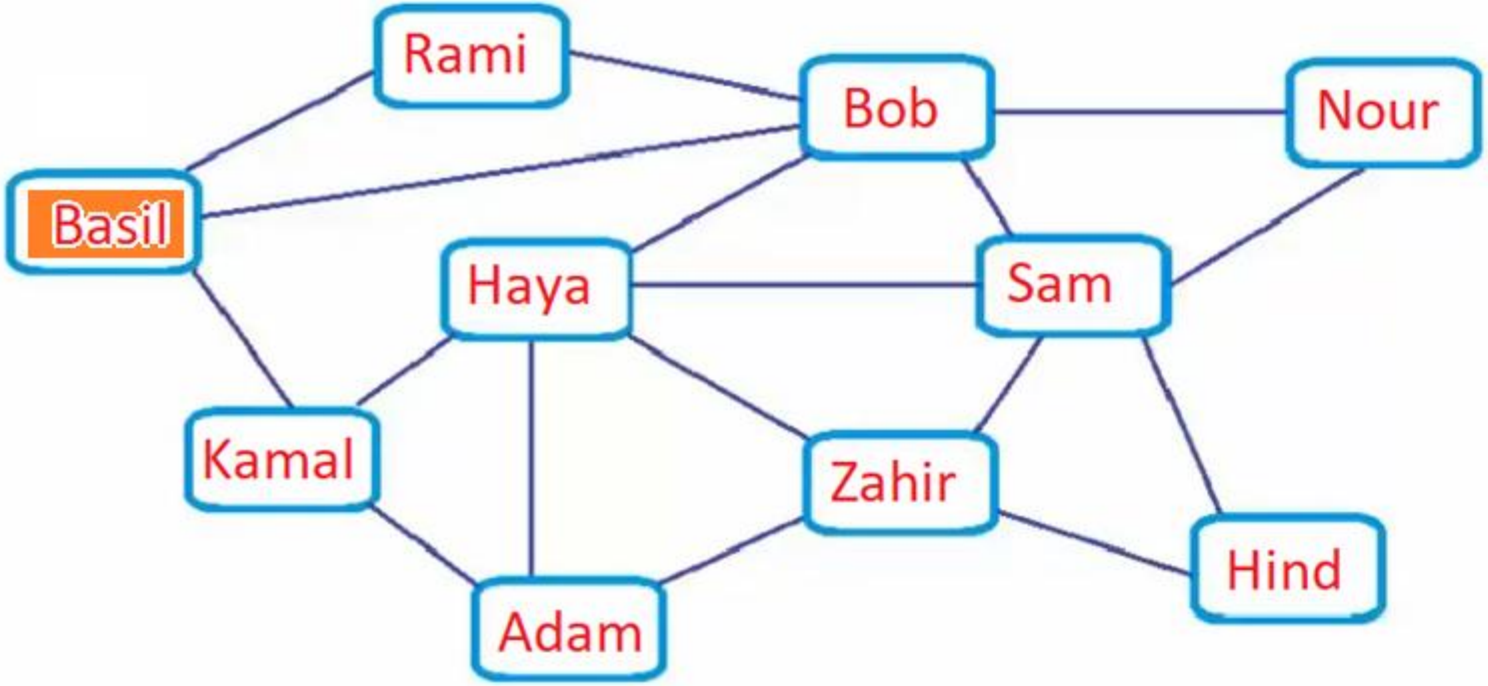


Applications of Graph
Social Network

- A user would a node in the graph.
- If two users are friends, there would be an edge connecting them.
- Social network is an undirected graph because friendship is a mutual relationship.
- If I'm your friend, then you are my friend too, so connections have to be two-way.

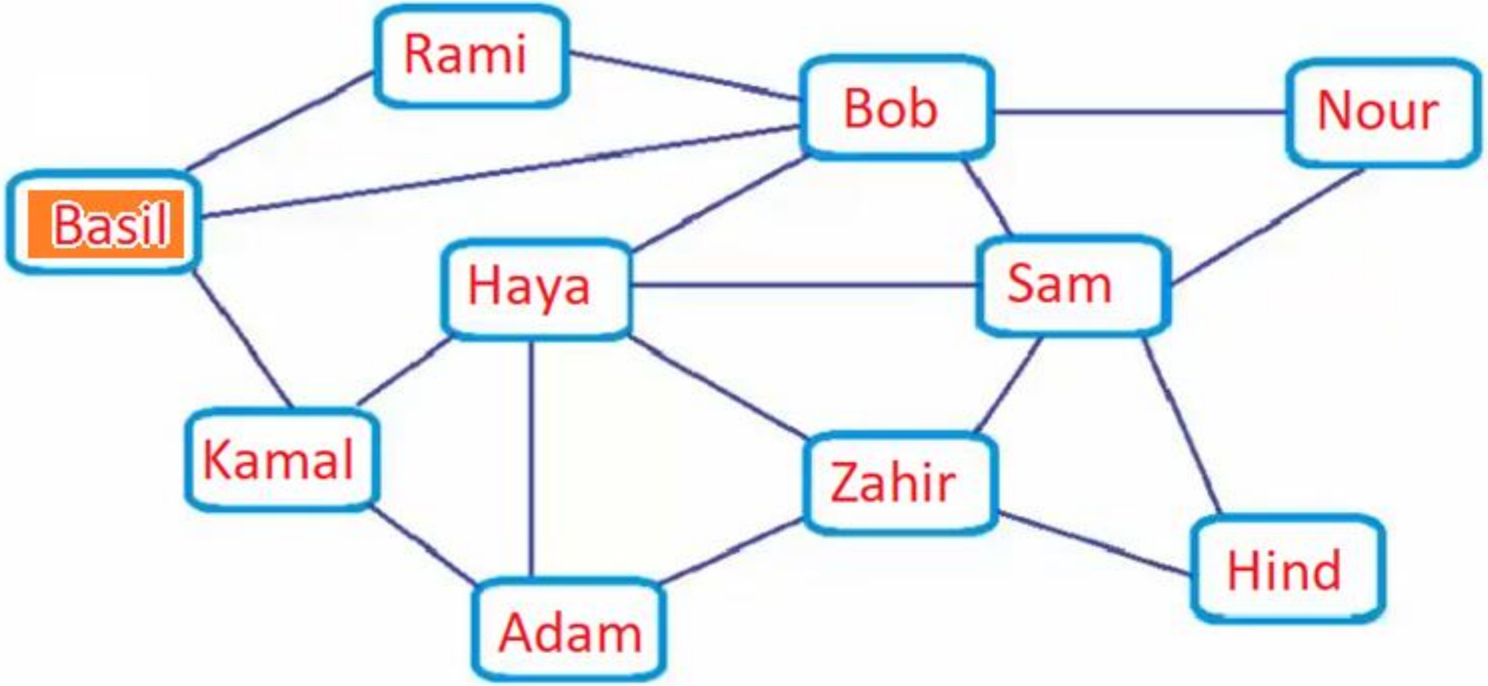
Applications of Graph
Social Network

- Once a system modeled as a graph, a lot of problems can be easily solved by applying standard algorithms in graph theory.
- For example, in this social network we want to suggest friends to Basil.



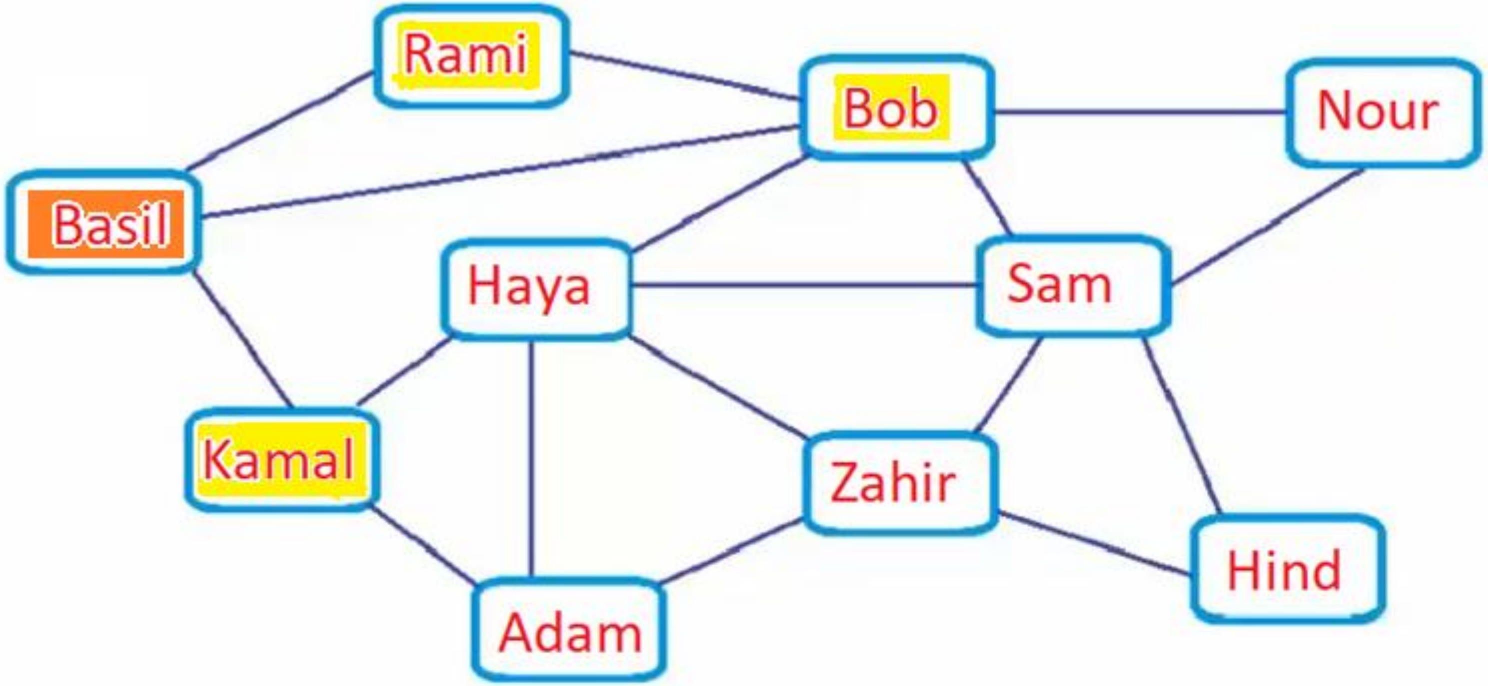
Applications of Graph
Social Network

- One possible approach to do is suggesting friends of friends who are not connected already.



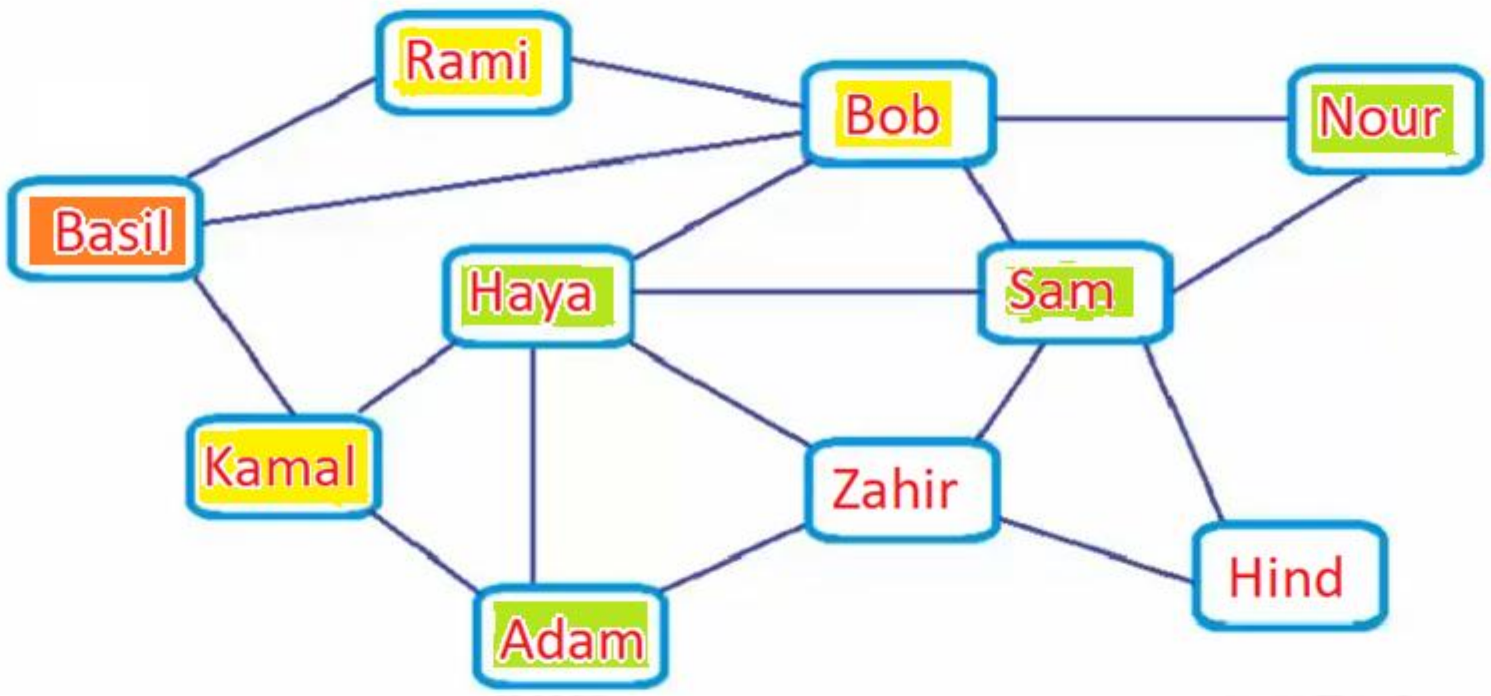
Applications of Graph
Social Network

- Basil has three friends, Rami, Bob, and Kamal, and friends of these three that are not connected to Basil already can be suggested.



Applications of Graph
Social Network

- We can suggest these four users to Basil.



Applications of Graph
Social Network

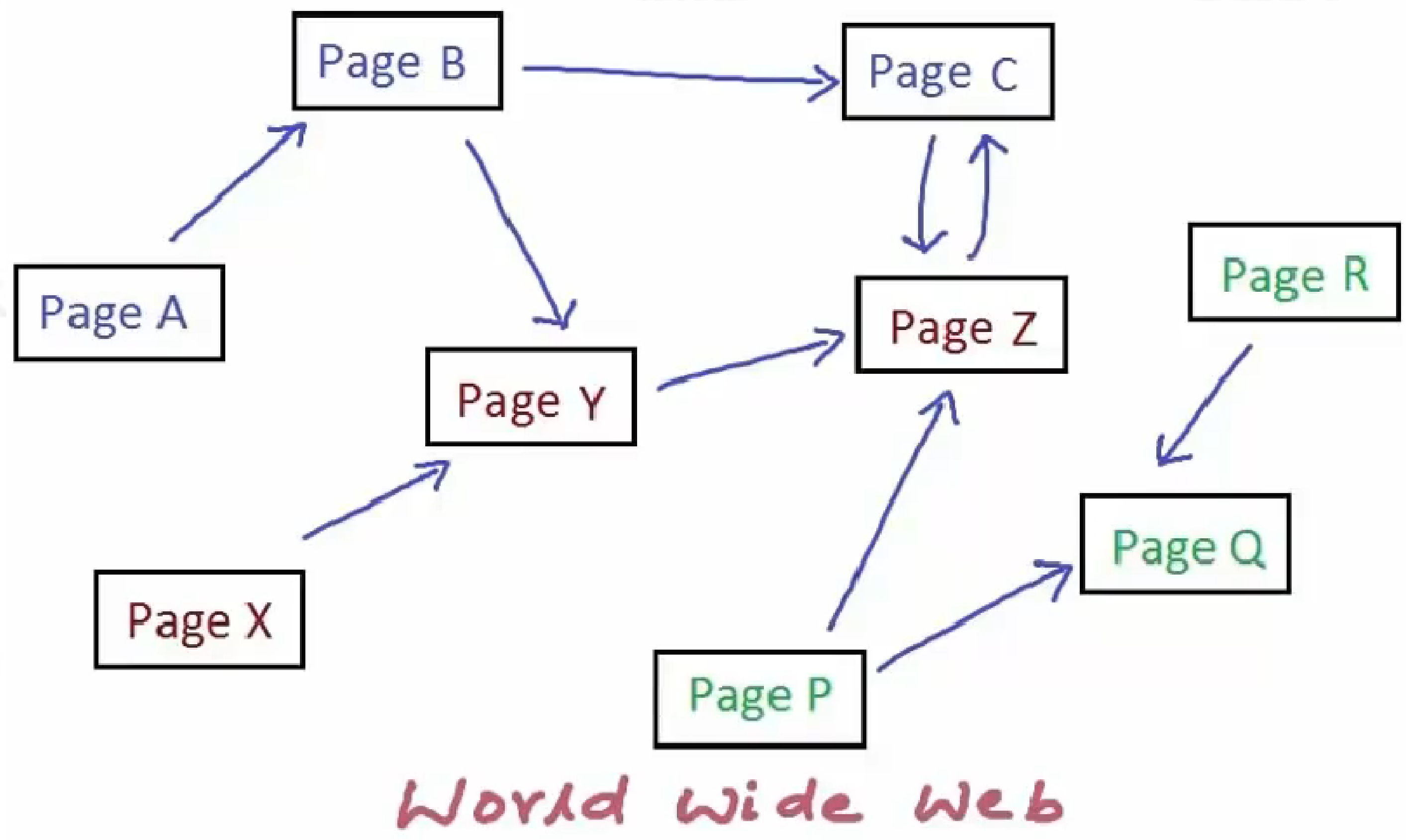
- Even if we described this problem in context of a social network, this is a standard graph problem.
- The problem here in pure graph terms is:

Find all nodes having
length of shortest path
from Basil equal to 2

- Standard algorithms can be applied to solve this problem.

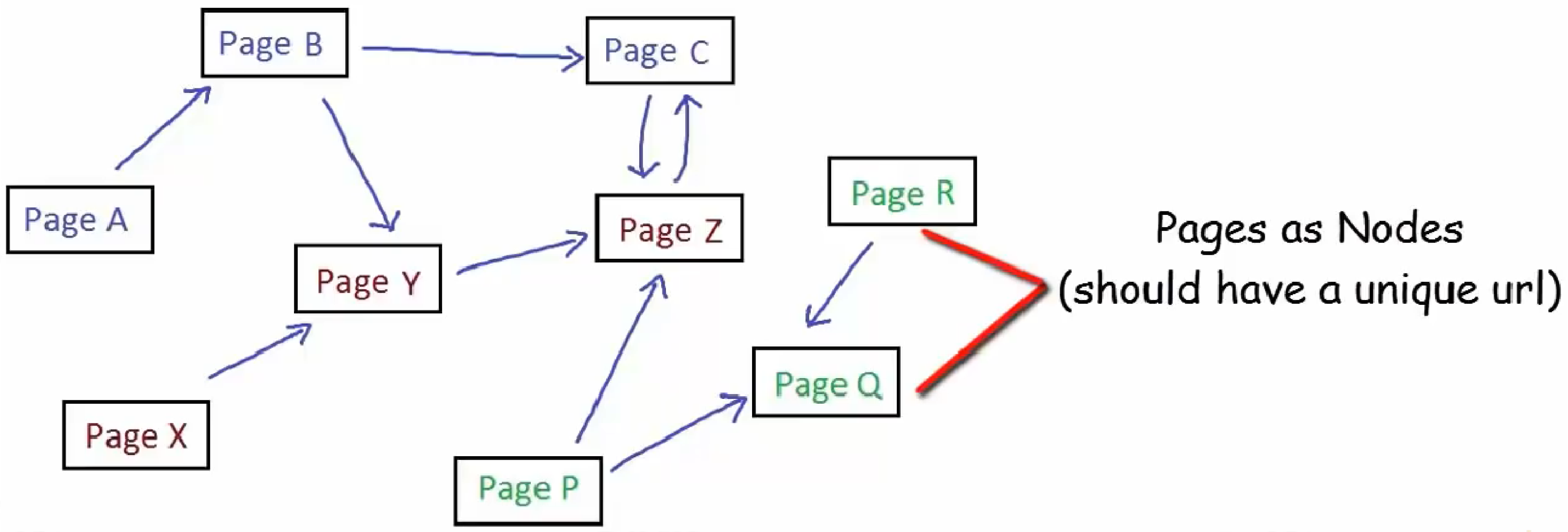
Applications of Graph
World Wide Web

- The World Wide Web can be represented as a directed graph.



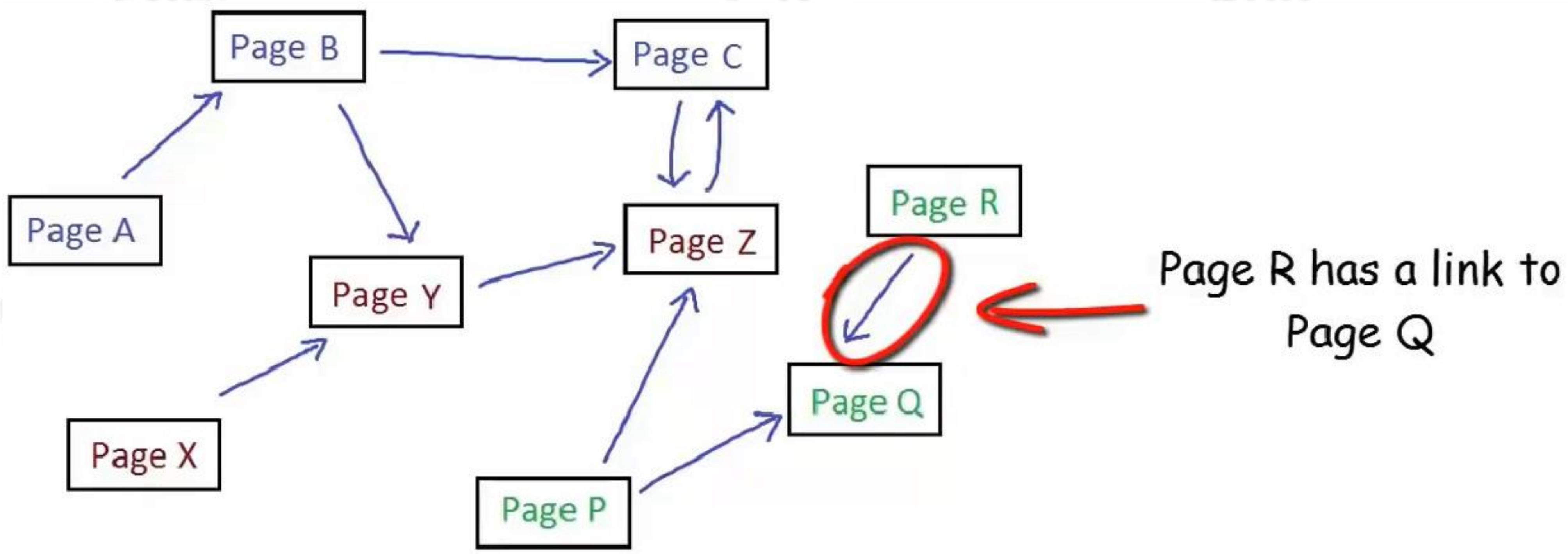
Applications of Graph
World Wide Web

- A Web page would have a unique address or URL can be a node in the graph.



Applications of Graph
World Wide Web

- We can have a directed if the page contains link to another page.



Applications of Graph
World Wide Web

- If we can represent web as a directed graph, we can apply standard graph theory algorithms to solve problems and perform tasks.
- One of the tasks that search engines like Google perform very regularly is web crawling.

Web-crawling

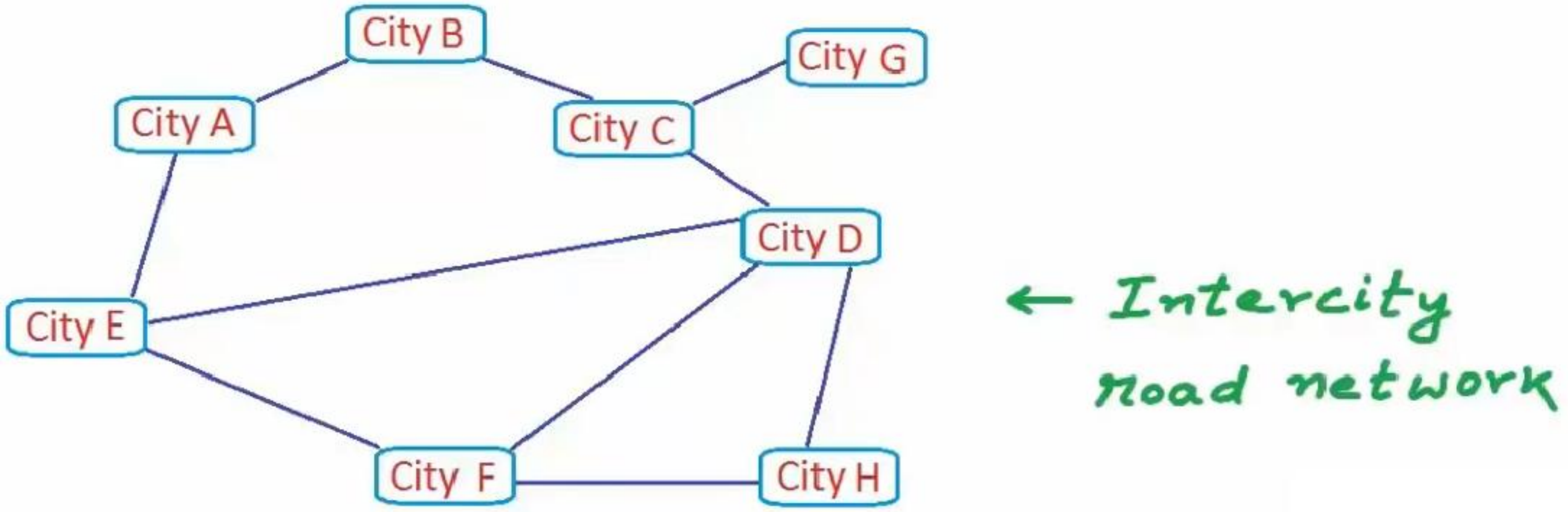
is

Graph Traversal

- In simpler words, act of visiting all nodes in a graph.
- There are standard algorithms for graph traversal.

Weighted Graph

- Sometimes in a graph, all connections cannot be treated as equal, Some connections can be preferable to others.
- For example, we can represent intercity road network as undirected graph (we are assuming all highways are bi-directional).

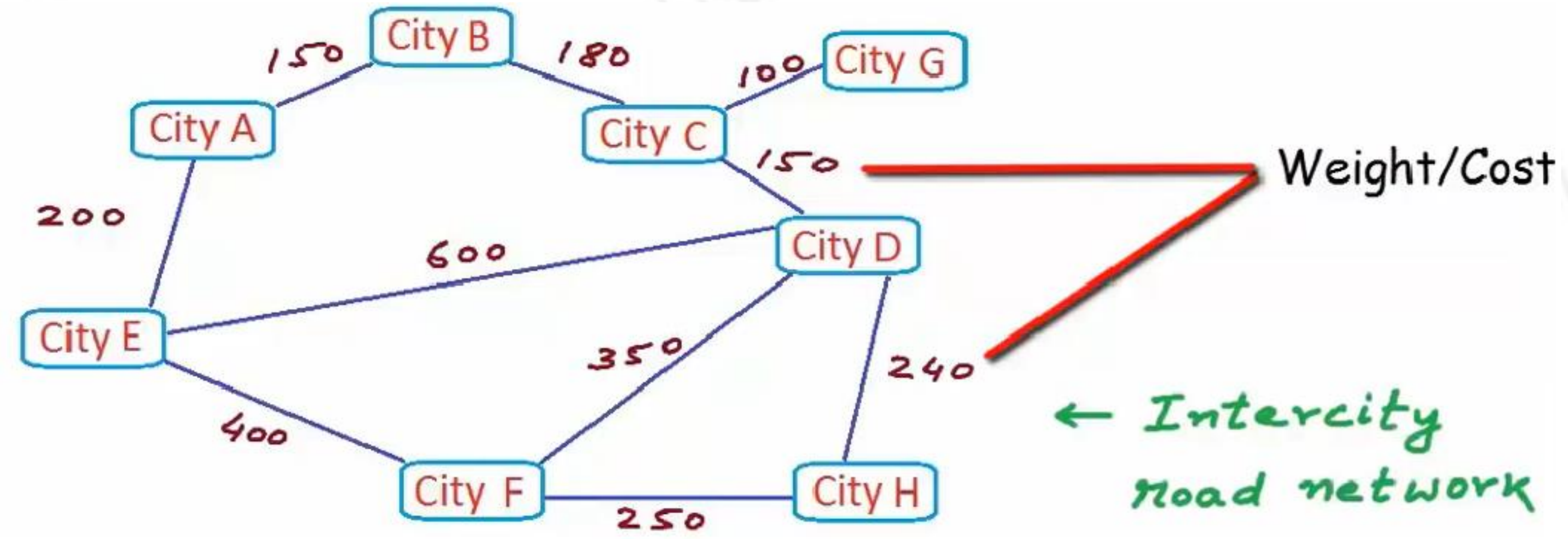


Weighted Graph

- Intracity road network (road network within a city) would definitely have one-way roads.
- So intracity road network must be represented as a directed graph.
- Clearly, we cannot treat all connections as equal in both intercity and intracity road network.
- Roads would be of different lengths; we need to take length of roads into account.
- In such cases, we associate some weight or cost with every edge.

Weighted Graph

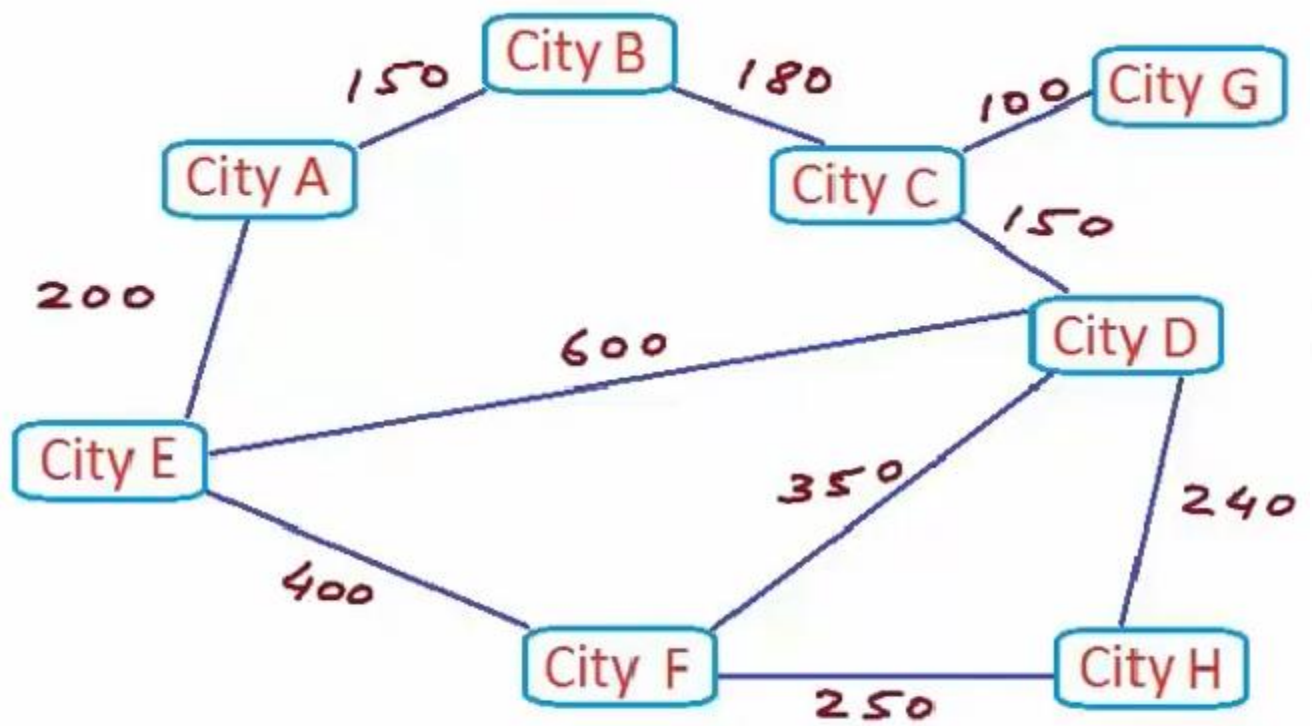
- Simply we label the edges with their weights.
- In roads network, weight can be length of the roads.



- Now edges are weighted, so the graph can be called a **weighted graph**.

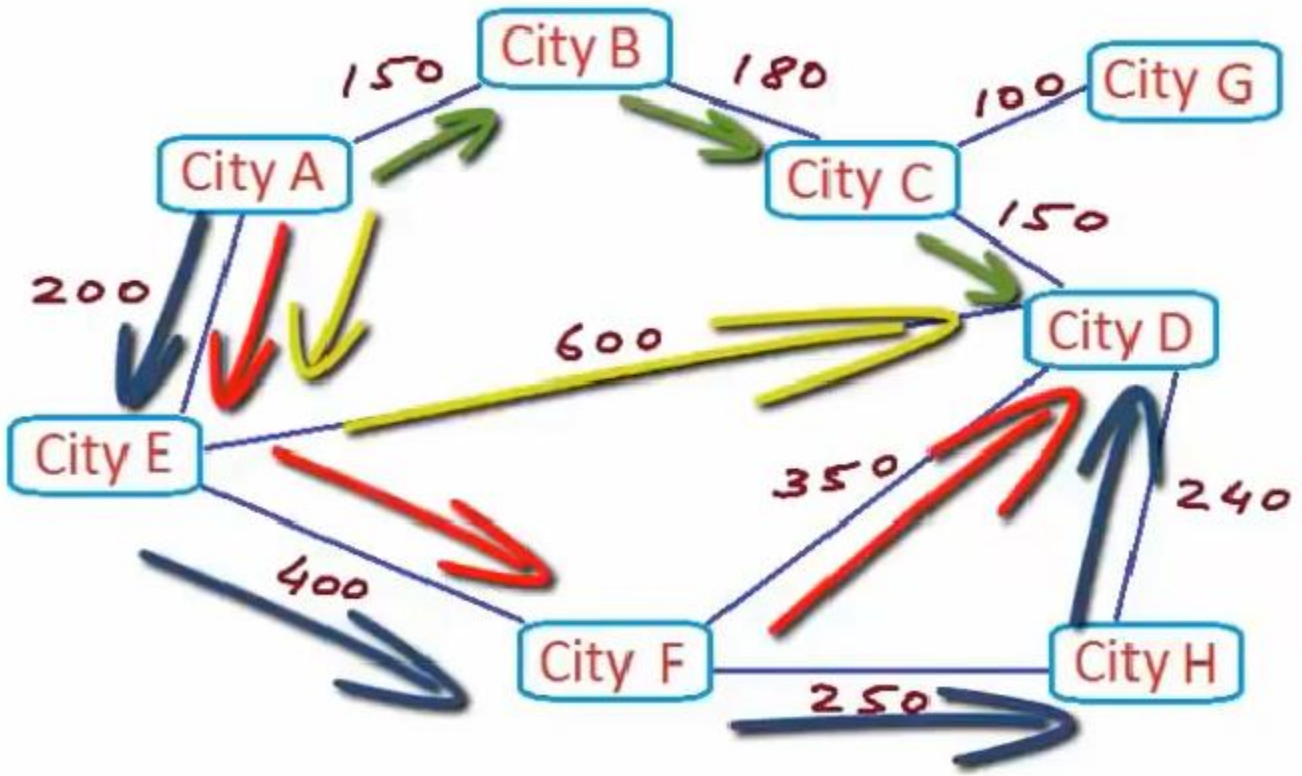
Weighted Graph

- let’s say we want to pick the best route from city A to city D.



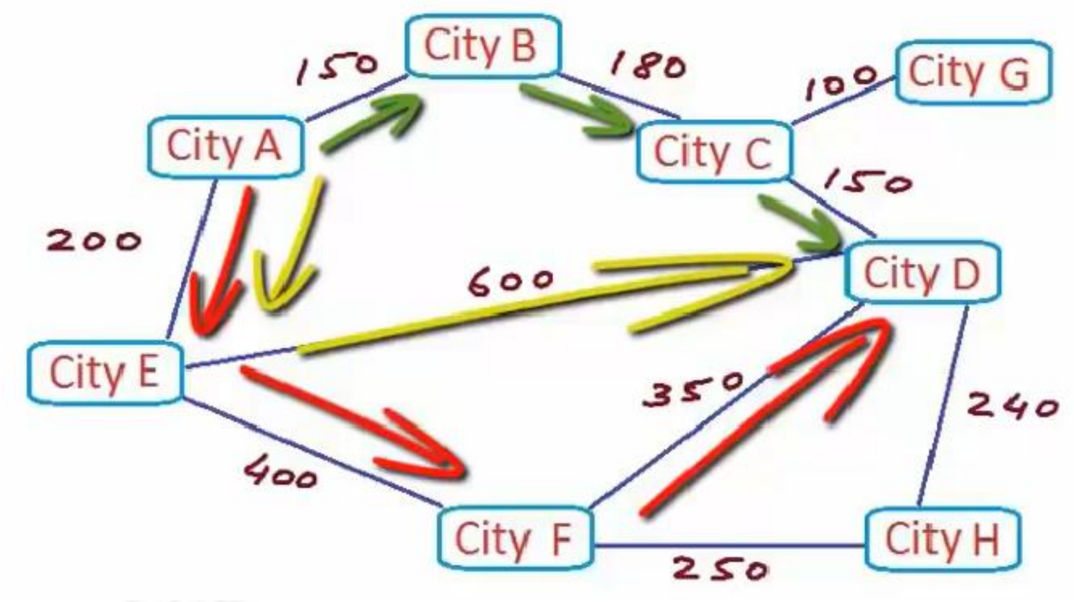
Weighted Graph

- There are four possible routes as shown in different colors.



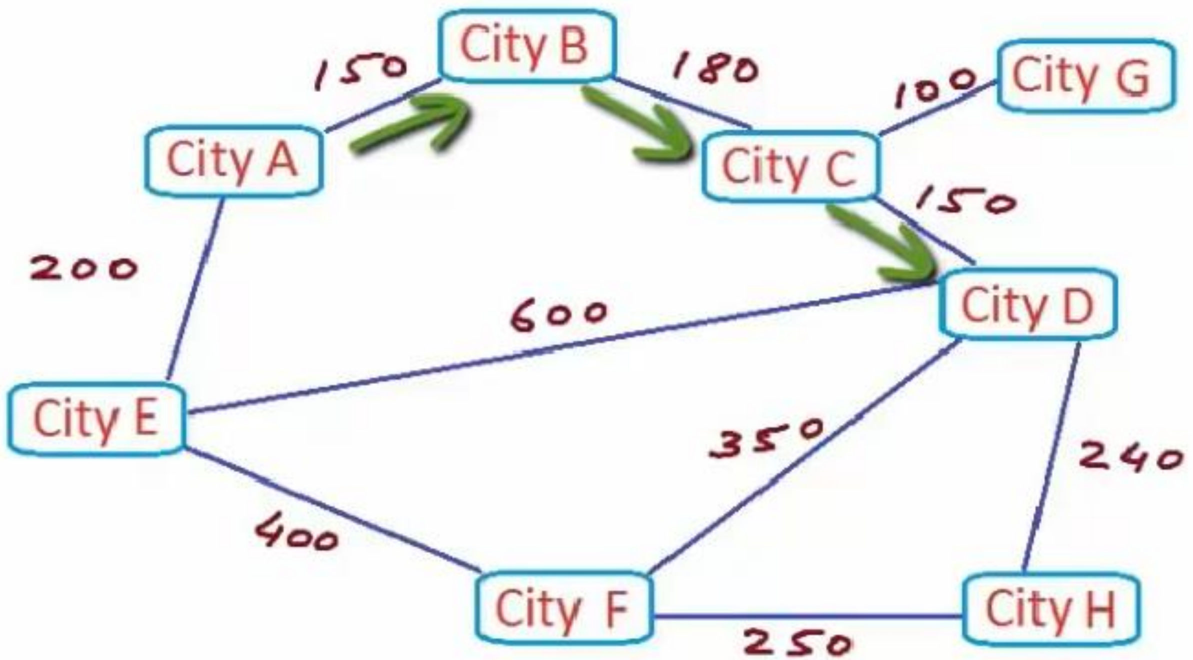
Weighted Graph

- If we want to treat all edges as equal, then:
  - The green route and the red route are equally good, both have three edges.
  - The yellow route is the best because we have only two edges in this path.



Weighted Graph

- If we want take weights into consideration, we need to add up weights of edges in a path to calculate the total cost.
- In this case, the shortest path is the green path.



Weighted Graph
Notes

- Connections have different weights.
- We can look at all the graphs as weighted graphs:
  - Unweighted graph can basically be seen as a weighted graph in which weight of all the edges is same.
  - Typically, we assume the weight as One.

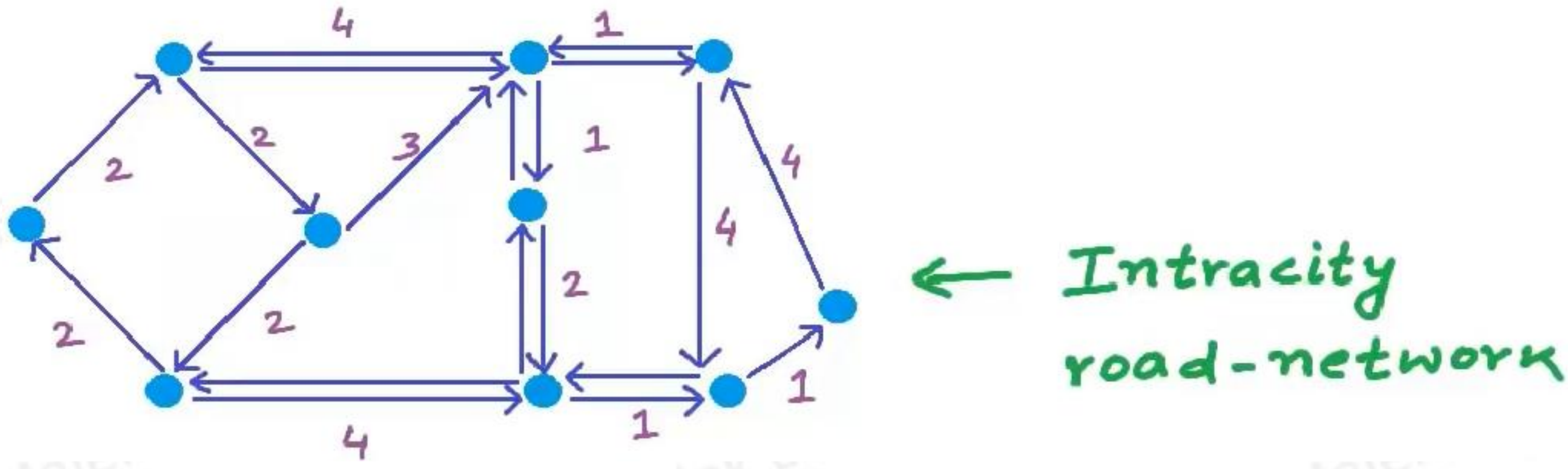
Unweighted graph
↳ a weighted graph
with all edges having
weight = 1 unit

Weighted Graph
Notes

- We have represented intercity road network as a **weighted undirected graph**.
- Social network was an **unweighted undirected graph**.
- World Wide Web was an **unweighted directed graph**.
- Intracity road network can be modeled as a **weighted directed graph** as shown in the next slide.

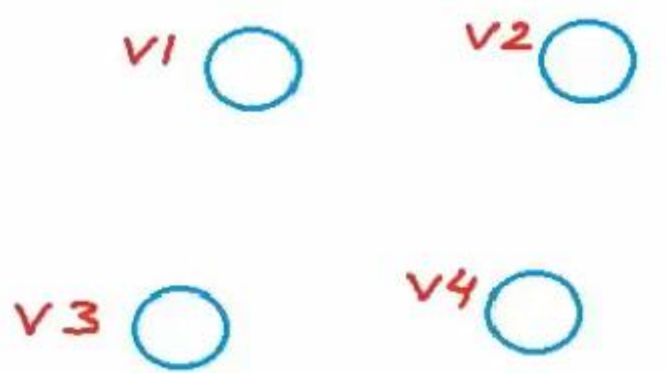
Weighted Graph

- We can represent intracity road network as the following graph.
- Intersections are nodes and road segments are edges.



Graph Properties
Number of Edges

- We want to draw a graph with four vertices.

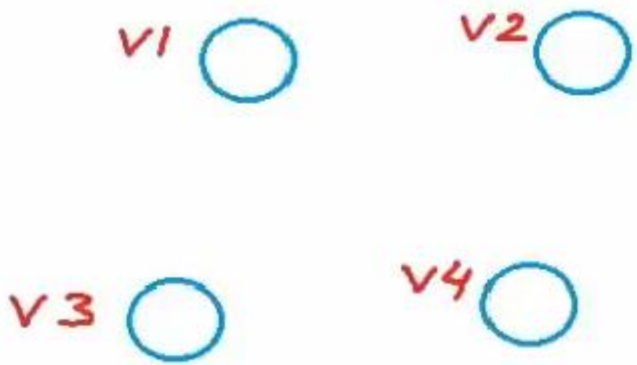


- The set of vertices and number of element are:

$$V = \{v1, v2, v3, v4\}$$
$$|V| = 4$$

Graph Properties
Number of Edges

- It is perfectly fine if we choose not to draw any edge here, this will still a graph.



- Set of edges can be empty, and nodes can be totally disconnected.

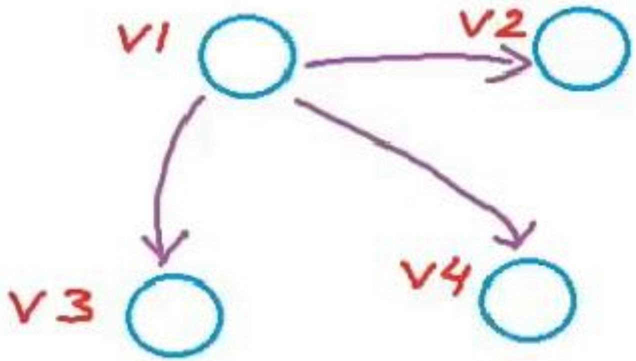
$$V = \{v1, v2, v3, v4\}$$

$$|V| = 4$$

$$E = \phi$$

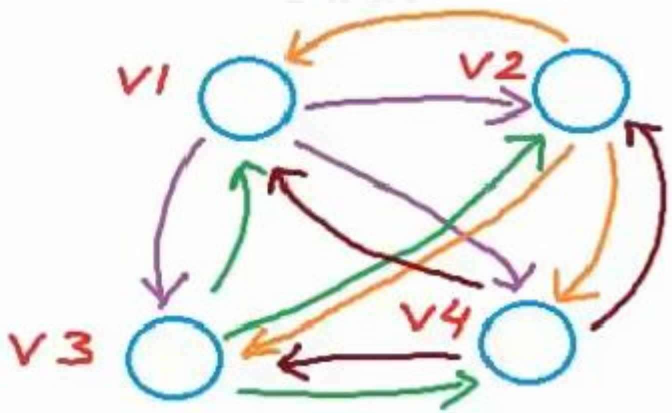
Graph Properties
Number of Edges (Directed Graph)

- So minimum number of edges in a graph is zero.
- What is the maximum number of edges?
- In a directed graph, each node can have directed edges to all other nodes.



Graph Properties
Number of Edges (Directed Graph)

- We have four nodes in total in our graph.



- So maximum number of edges is:

if $|V| = n$

then,

$0 \leq |E| \leq n(n-1)$, if directed

Properties
Number of Edges (Undirected Graph)

- In an undirected graph, we can have only one bidirectional edge between a pair of nodes.

if $|V| = n$

then,

$$0 \leq |E| \leq \frac{n(n-1)}{2}, \text{ if undirected}$$

Properties
Number of Edges

- As we can see, number of edges in a graph can be large compared to number of vertices.
- For example, in a directed graph:

if $|V| = 10, \quad |E| \leq 90$

if $|V| = 100, \quad |E| \leq 9900$

- Maximum number of edges is close to square of number of vertices.

Properties
Number of Edges

- A graph is called **Dense** if number of edges in the graph is close to maximum.
- A graph is called **Sparse** if the number of edges is close to number of vertices.

Dense → too many edges
Sparse → too few edges

- There is no defined boundary for what can be called dense and what can be called sparse, but this is an important classification.

Properties
Number of Edges

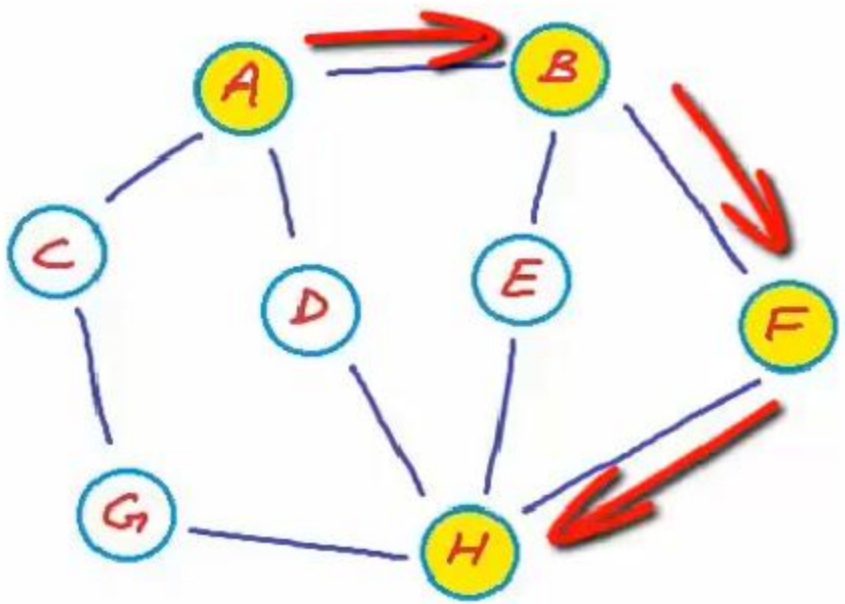
- While working with a graphs, a lot of decisions are made based on whether the graph is dense or sparse.
- For example:
  - We typically store a dense graph in something called **Adjacency Matrix.**
  - For a sparse graph we typically use something called **Adjacency List**.

Graph Properties
Path

- A path in a graph is:

Path :- a sequence of vertices where each adjacent pair is connected by an edge.

< A, B, F, H >



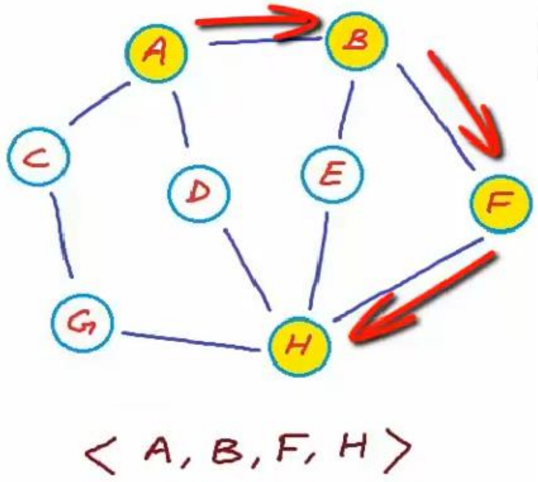
- <A, B, F, H> is a path in this graph.

Graph Properties
Path

- A path is called a simple path if:

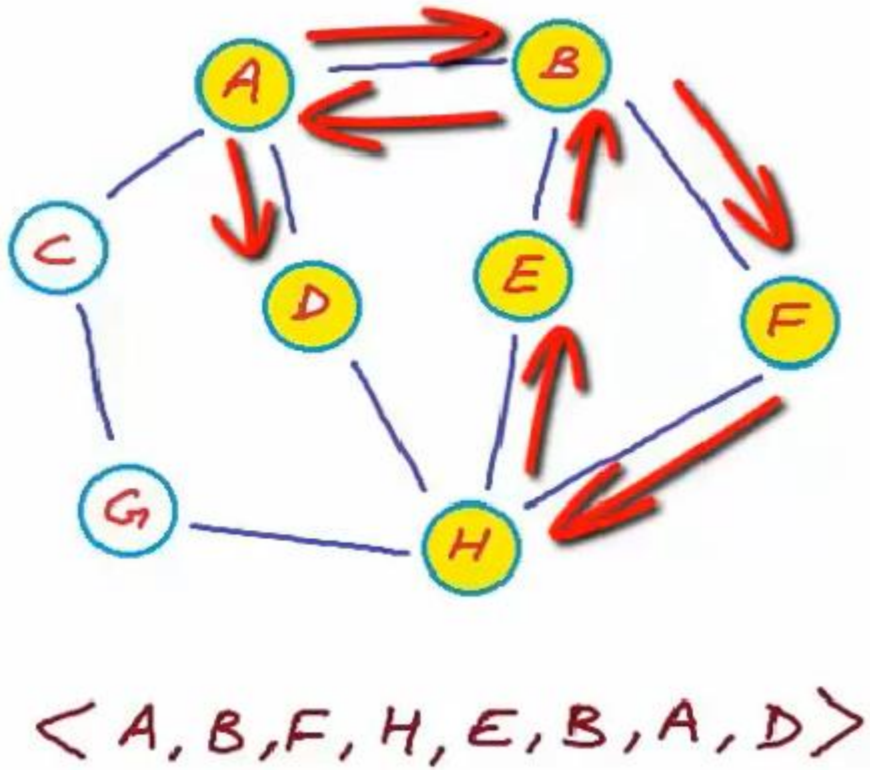
Simple path :- a path in which no vertices (and thus no edges) are repeated.

- <A, B, F, H> is a simple path.



Graph Properties
Path

- The following path is not a simple path (1 edge and 2 vertices are repeated).



Graph Properties

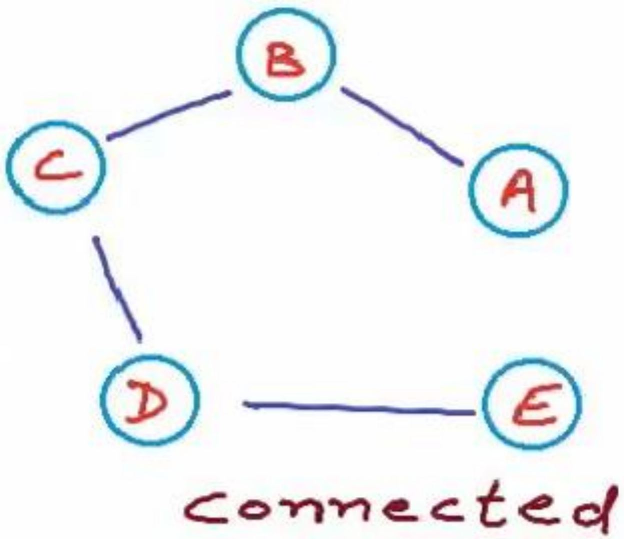
- A graph is called strongly connected if:

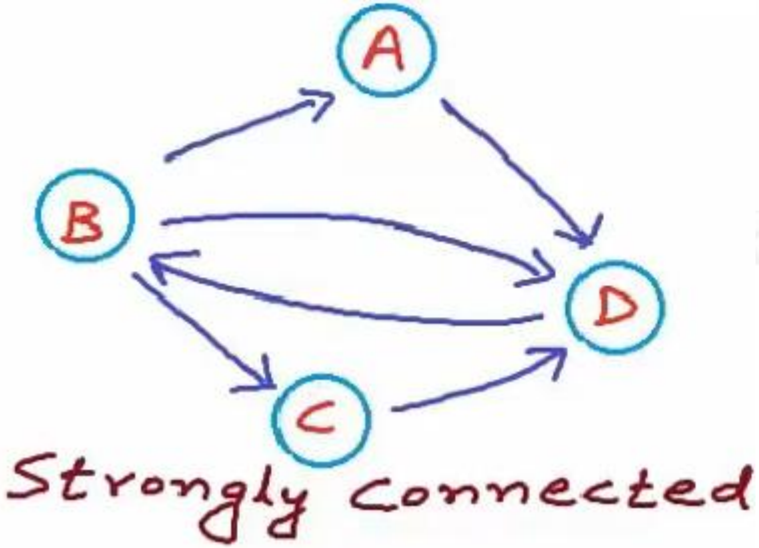
Strongly connected Graphs:
↳ if there is a path from any vertex to any other vertex.

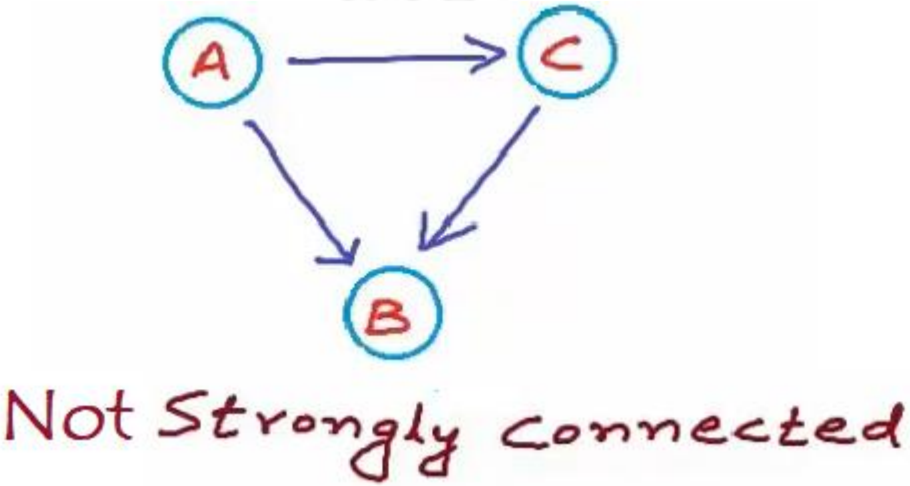
- If it's an undirected graph, we simply call it connected, and if it's a directed we call it strongly connected.

Graph Properties

- Examples:

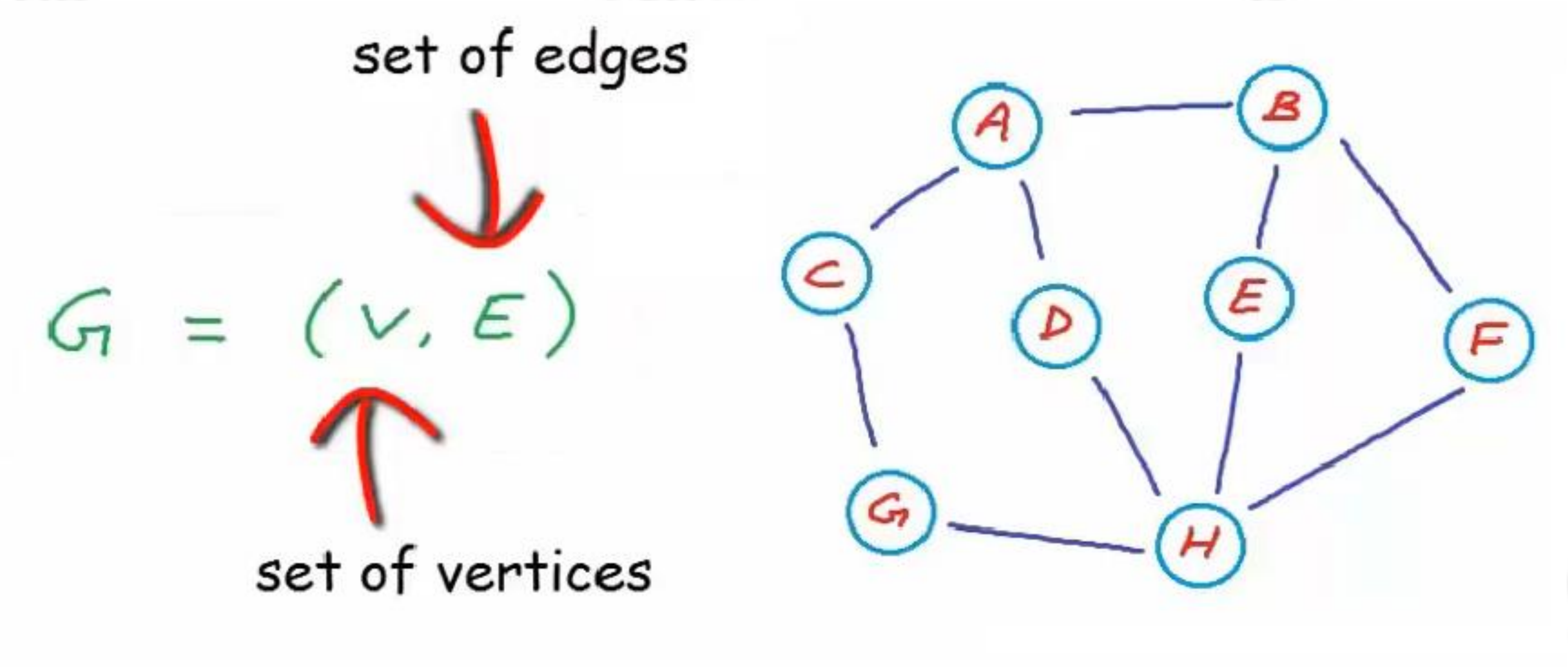






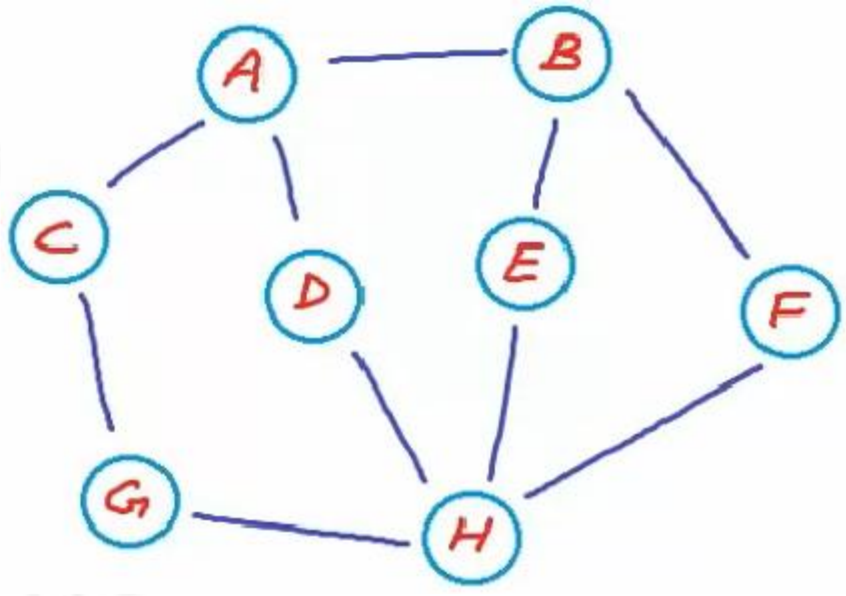
Graph Representation

- A graph as we know contains a set of vertices and a set of edges.



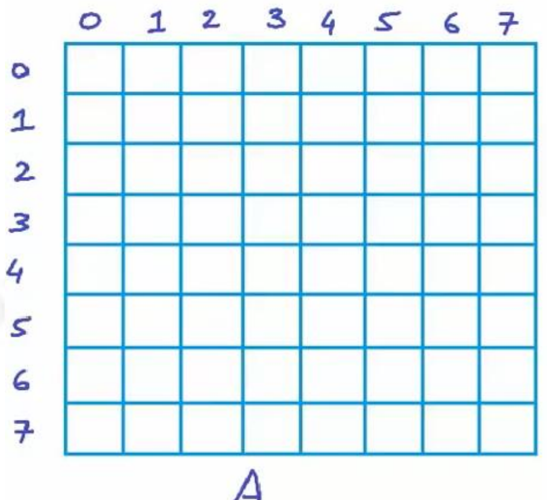
Graph Representation

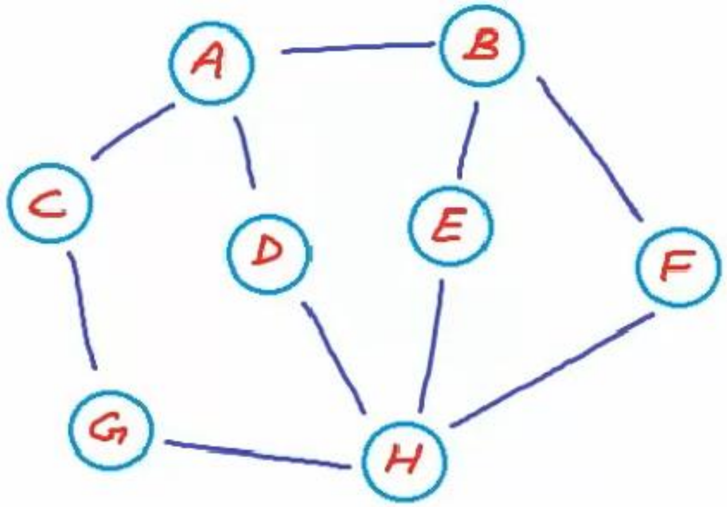
- To create and store a graph in computer's memory there are different ways.
- We will cover one of these ways which called Adjacency Matrix.
- Suppose we have the following unweighted graph.



Graph Representation

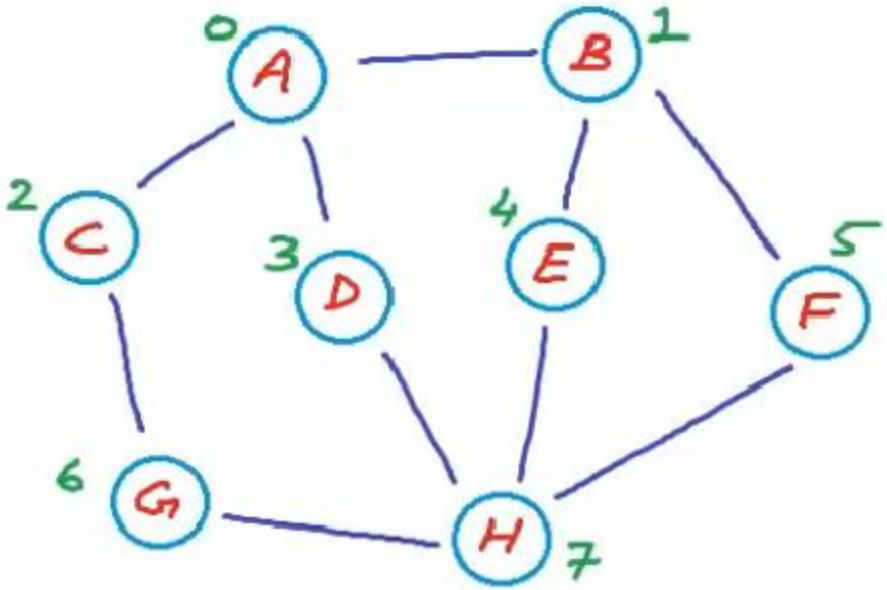
- We can store the edges in a two-dimensional array or matrix.
- We will create a two-dimensional array of size (V x V), where V is number of vertices.
- In our example graph, number of vertices is 8, so we will create an array of size (8 x 8) with the name **A**.





Graph Representation

- We can give each vertex a number starting from 0 to (V – 1) same as an array index.



Graph Representation

- Now, in **A** we can do the following:

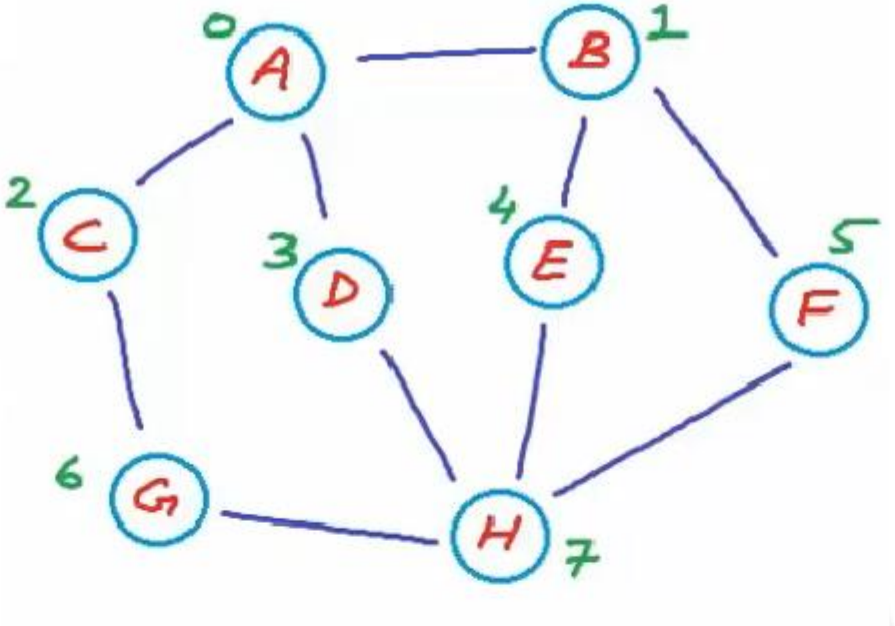
$$A_{ij} = \begin{cases} 1, & \text{if } \exists \text{ edge from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

Graph Representation

- So, **A** will be as the following:

	0	1	2	3	4	5	6	7
0	0	1	1	1	0	0	0	0
1	1	0	0	0	1	1	0	0
2	1	0	0	0	0	0	1	0
3	1	0	0	0	0	0	0	1
4	0	1	0	0	0	0	0	1
5	0	1	0	0	0	0	0	1
6	0	0	1	0	0	0	0	1
7	0	0	0	1	1	1	1	0

A



Graph Representation

- Notice that matrix **A** is symmetric.

$$
\begin{array}{c|cccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
5 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
6 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
7 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0
\end{array}
\rightarrow A_{ij} = A_{ji}
$$

A

- In general, for undirected graph the matrix is symmetric because $A_{ij} = A_{ji}$
- We are having two positions filled for each edge.

Graph Representation

- This representation of a graph in which edges are stored on a matrix or two-dimensional array is called Adjacency Matrix representation.
- Matrix **A** in our example is called an Adjacency Matrix.

Adjacency Matrix

	0	1	2	3	4	5	6	7
0	0	1	1	1	0	0	0	0
1	1	0	0	0	1	1	0	0
2	1	0	0	0	0	0	1	0
3	1	0	0	0	0	0	0	1
4	0	1	0	0	0	0	0	1
5	0	1	0	0	0	0	0	1
6	0	0	1	0	0	0	0	1
7	0	0	0	1	1	1	1	0

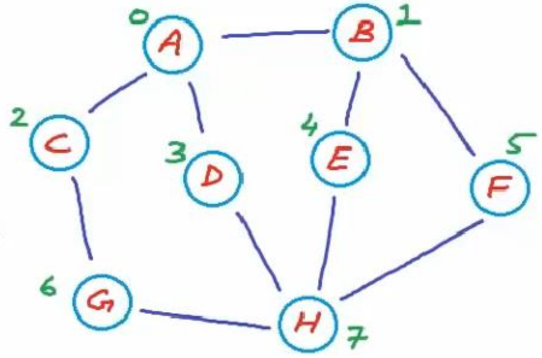
A

Graph Representation Operations

- When we use the Adjacency Matrix representation, what do you think would be the time cost of finding all nodes adjacent to a given node?

finding adjacent nodes

- For example, we need to find all nodes adjacent to node F.



Graph Representation Operations

- If we are given a name of a node, we first need to know it's index, in this case F at index 5.

Adjacency Matrix

	0	1	2	3	4	5	6	7
0	0	1	1	1	0	0	0	0
1	1	0	0	0	1	1	0	0
2	1	0	0	0	0	0	1	0
3	1	0	0	0	0	0	0	1
4	0	1	0	0	0	0	0	1
5	0	1	0	0	0	0	0	1
6	0	0	1	0	0	0	0	1
7	0	0	0	1	1	1	1	0

A

Graph Representation Operations

- Then we can go to the row with that index in the adjacency matrix.

Adjacency Matrix

	0	1	2	3	4	5	6	7
0	0	1	1	1	0	0	0	0
1	1	0	0	0	1	1	0	0
2	1	0	0	0	0	0	1	0
3	1	0	0	0	0	0	0	1
4	0	1	0	0	0	0	0	1
5	0	1	0	0	0	0	0	1
6	0	0	1	0	0	0	0	1
7	0	0	0	1	1	1	1	0

A

Graph Representation Operations

- Now we can scan this complete row to find all the adjacent nodes.

Adjacency Matrix

	0	1	2	3	4	5	6	7
0	0	1	1	1	0	0	0	0
1	1	0	0	0	1	1	0	0
2	1	0	0	0	0	0	1	0
3	1	0	0	0	0	0	0	1
4	0	1	0	0	0	0	0	1
5	0	1	0	0	0	0	0	1
6	0	0	1	0	0	0	0	1
7	0	0	0	1	1	1	1	0

A

Graph Representation Operations

- Scanning a row in the adjacency matrix will cost us time proportional to the number of vertices **V**, because in a row we have exactly **V** columns.
- So overall, time cost of this operation is:

Operation	Time-cost
finding adjacent nodes	O(v)

Graph Representation Operations

- When we use the Adjacency Matrix representation, what do you think would be the time cost of finding if two nodes are connected or not?

finding if two nodes are connected

- The nodes are given to us as indices or names.
- We simply need to look at value in a particular row and particular column.

Graph Representation Operations

- This will cost us a constant time.
- You can access any value in any cell in a two-dimensional array in constant time.
- So, if indices are given, the time cost of this operation is:

Operation	Time-cost
finding if two nodes are connected	O(1)

Graph Representation
Weighted Graph

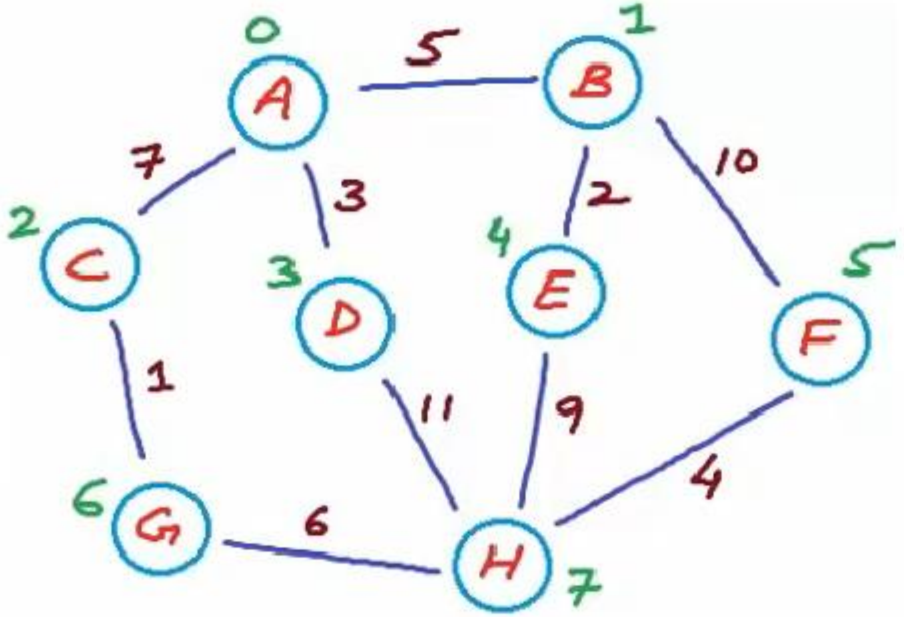
- If we want to store a weighted graph in adjacency matrix then A_{ij} can be set as weight of an edge.
- For nonexistent edges, we can have a default value like a large number or maximum possible integer that is never expected to be an edge weight.
- We can choose the default value as infinity, minus infinity, or any other value that would never be a valid edge weight.

Graph Representation
Weighted Graph

Adjacency Matrix

	0	1	2	3	4	5	6	7
0	∞	5	7	3	∞	∞	∞	∞
1	5	∞	∞	∞	2	10	∞	∞
2	7	∞	∞	∞	∞	∞	1	∞
3	3	∞	∞	∞	∞	∞	∞	11
4	∞	2	∞	∞	∞	∞	∞	9
5	∞	10	∞	∞	∞	∞	∞	4
6	∞	∞	1	∞	∞	∞	∞	6
7	∞	∞	∞	6	11	9	4	∞

A



Graph Representation
Notes

- In the adjacency matrix representation, we have gone high in memory usage.
- We are using exactly V^2 space.
- We are not just storing the information that these two nodes are connected; we are also storing that these two nodes are not connected.
- This is redundant information.
- If a graph is dense, then this is good.
- But if the graph is sparse, then we are wasting a lot of memory.

Any Questions???...