Graphs

A Graph is a non-linear data structure / ADT of a finite set of **vertices(or nodes)** and set of **edges** which connect a pair of vertices. The edges are lines or arcs that connect any two nodes in the graph.

In Mathematical terms:
A graph **G** is an ordered pair of set **V** of vertices and set **E** of edges. i.e **G = (V, E)**
Since it is an ordered pair, so sequence of items matter.

$|V| \rightarrow$ denotes number of vertexes
$|E| \rightarrow$ denotes number of edges

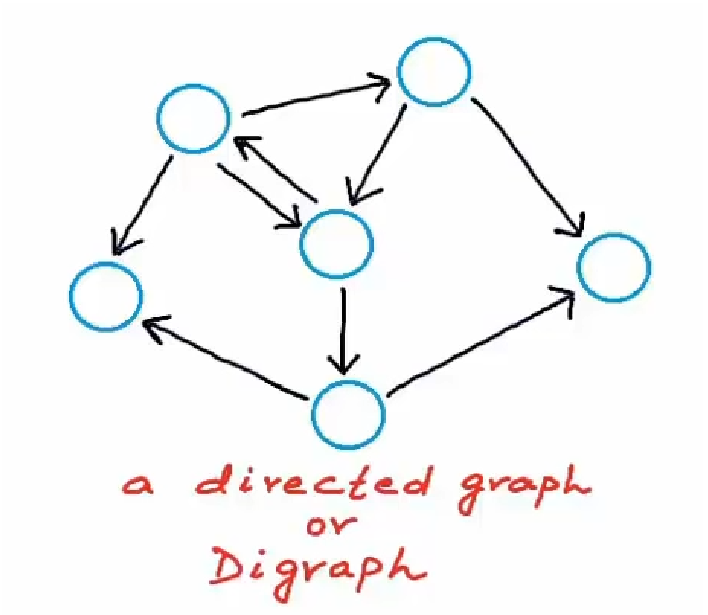
if $|V| = n$ then,
$0 \leq |E| \leq n(n-1)$, if directed
$0 \leq |E| \leq \frac{n(n-1)}{2}$, if un-directed

- **Ordered Pair :** A pair in which order matters and changing the order is not allowed is called an ordered pair. e.g.
  ```
  (a, b) != (b, a) if a != b
  ```
- **UnOrdered Pair :** A pair in which order doesn't matter is called an unordered pair. e.g.
  ```
  {a, b} = {b, a}
  ```

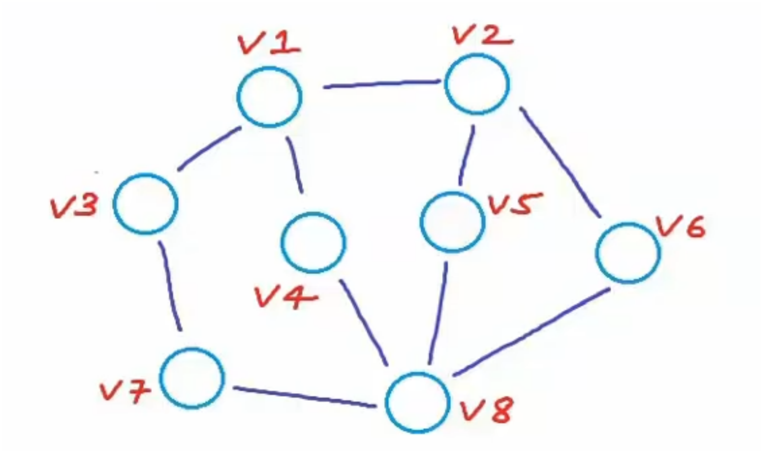
A **tree** is a special kind of graph. A graph with no cycles is called a tree. A tree is an acyclic connected graph.

Directed vs UnDirected Graph

Directed Graph consist of ordered pair of vertices **(u, v)**, where first vertex **u** is the origin and second vertex **v** is the destination. A directed graph is a graph in which all the edges are uni-directional i.e. the edges point in a single direction.



Undirected Graph consist of ordered pair of vertices **{u, v}**
An undirected graph is a graph in which all the edges are bi-directional i.e. the edges do not point in any specific direction. We can always redraw an undirected graph as directed graph (by representing each edge as 2 directed edges) but not vise-versa.



```
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}
```

Weighted vs Unweighted Graph

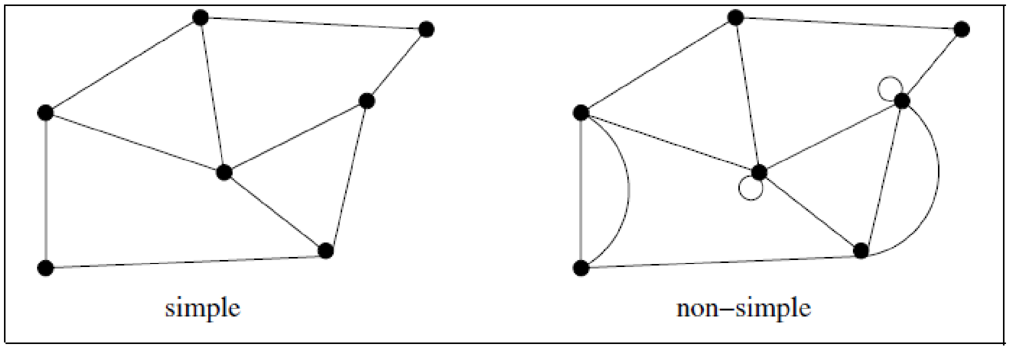
Each edge (or vertex) in a weighted graph G is assigned a numerical value, or weight. The edges of a road network graph might be weighted with their length, drive-time, or speed limit, depending upon the application. In unweighted graphs, there is no cost distinction between various edges and vertices.

The difference between weighted and unweighted graphs becomes particularly apparent in finding the shortest path between two vertices. For unweighted graphs, the shortest path must have the fewest number of edges, and we can assume the weight to be same for all edges in the graph. Shortest paths in weighted graphs requires more sophisticated algorithms.

Simple vs Non-simple Graphs

Certain types of edges complicate the task of working with graphs. A self-loop is an edge (x, x) involving only one vertex. An edge (x, y) is a multiedge if it occurs more than once in the graph. Both of these structures require special care in implementing graph algorithms. Hence any graph that avoids them is called simple.

Mathematicians sometimes refer to graphs with parallel edges as multigraphs and graphs with no parallel edges or self-loops as simple graphs.



Sparse vs Dense

Graphs are sparse when only a small fraction of the possible vertex pairs actually have edges defined between them. Graphs where a large fraction of the vertex pairs define edges are called dense. There is no official boundary between what is called sparse and what is called dense, but typically dense graphs have a quadratic number of edges, while sparse graphs are linear or log linear - $|V| * \log |V|$ in size.

Graphs with all edges present are called complete graphs.

Cyclic vs Acyclic

A *cyclic graph* is a directed graph which contains a path from at least one node back to itself. In simple terms cyclic graphs contain a cycle. A *simple cycle* is a cycle with no repeated edges or vertices (except the requisite repetition of the first and last vertices).

A *acyclic graph* is a directed graph which contains absolutely no cycle, that is no node can be traversed back to itself.

Directed Acyclic Graphs are called **DAG**s. They arise naturally in scheduling problems, where a directed edge (x, y) indicates that activity x must occur before y. An operation called topological sorting orders the vertices of a DAG to respect these precedence constraints. Topological sorting is typically the first step of any algorithm on a DAG.

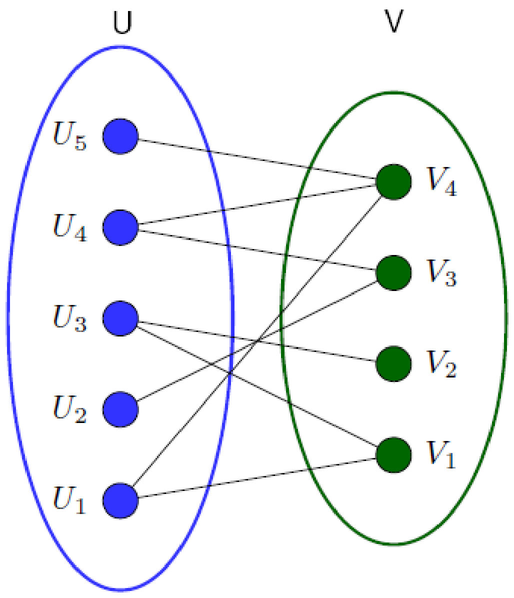
Bipartite Graph

In graph theory, a bipartite graph is a graph whose vertices can be divided into two disjoint and independent sets U and V such that every edge connects a vertex in U to one in V. Vertex sets U and V are usually called the parts of the graph. A bipartite graph is a graph that does not contain any odd-length cycles.

The two sets U and V may be thought of as a coloring of the graph with two colors: if one colors all nodes in U blue, and all nodes in V green, each edge has endpoints of differing colors, as is required in the graph coloring problem. In contrast, such a coloring is impossible in the case of a non-bipartite graph, such as a *triangle*: after one node is colored blue and another green, the third vertex of the triangle is connected to vertices of both colors, preventing it from being assigned either color.

One often writes $G = (U, V, E)$ to denote a bipartite graph whose partition has the parts U and V, with E denoting the edges of the graph.

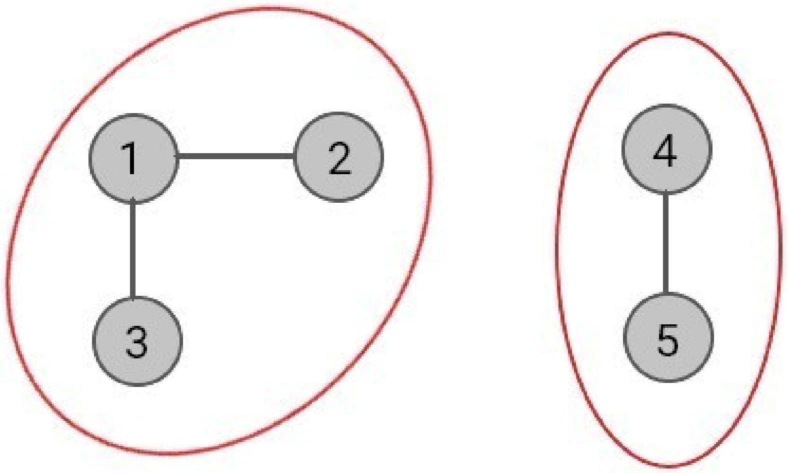
Every *tree* is bipartite.



Definitions from graph theory

A **path** is a sequence of vertices with the property that each vertex in the sequence is adjacent to the vertex next to it. A path that does not repeat vertices is called a *simple path*. The length of a path or a cycle is its number of edges.

A graph is **connected** if there is a path from every vertex to every other vertex in the graph. A graph that is not connected consists of a set of *connected components*, which are maximal connected sub-graphs.



There are 2 connected components in above un-directed graph
3 - 1 - 2 & 4 - 5

- When an edge connects two vertices, the vertices are said to be *adjacent* to each other and that the edge is *incident* on both vertices.
- *Degree* of a vertex is the number of edges incident on it.
- A *forest* is a disjoint set of trees. A *spanning tree* of a connected graph is a subgraph that contains all of that graph's vertices and is a single tree.
- A *simple cycle* is a cycle with no repeated vertices (except for the beginning and ending vertex).
- Articulation points, bridges, and biconnected components
  Let $G = (V, E)$ be a connected, undirected graph.
  - An *articulation point* of G is a vertex whose removal disconnects G.
  - A *bridge* of G is an edge whose removal disconnects G.
  - A *biconnected component* of G is a maximal set of edges such that any two edges in the set lie on a common simple cycle. A graph G is biconnected iff it contains no articulation point.