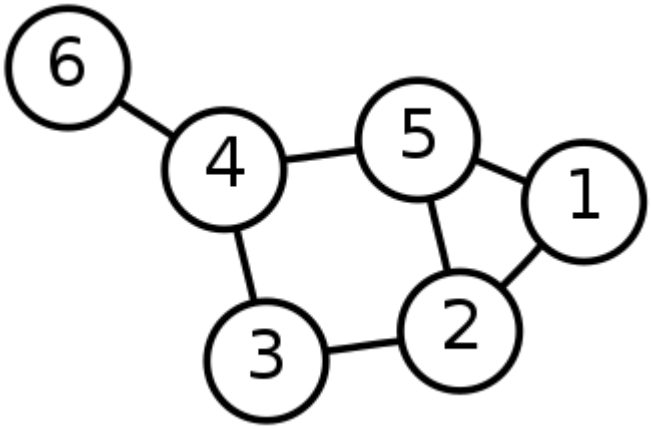
GRAPHS

A graph is collection of nodes or enitiaties connected by vertex. There is no rule of connection . The study of graph is called graph theory.



Mathematical Defination

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

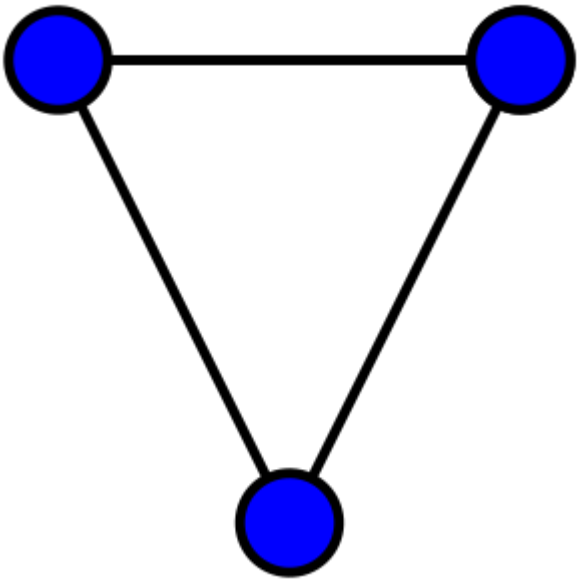
$G = (E, V)$

Here the order matters i.e $(a, b) \neq (b, a)$. First element is a set of edges or node and second a set of vertices

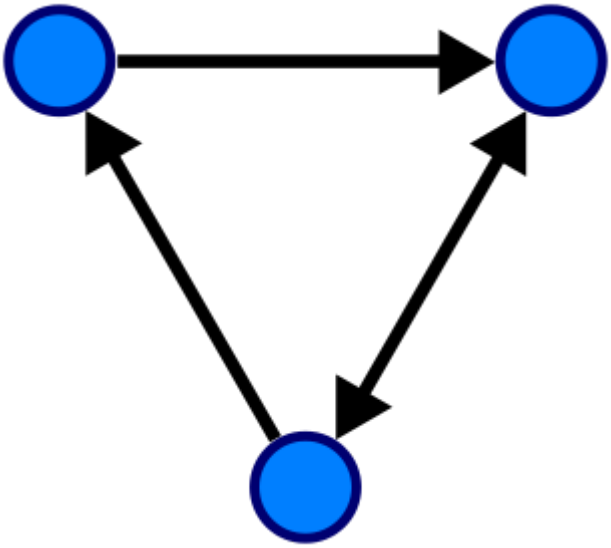
Types of Graphs

Edges can be :

1. Directed - connection is one way , one edge is origin and other is destination.



2. Undirected - connection is two way , edges are both origin as well as destination



A graph with all directed edges is called a directed graph or digraph . And graph with all undirected edges is called undirected graph. There is no special name for an undirected graph.

In other words Directed graph or digraph are graph in which edges are unidirectional or ordered pairs and undirected graph are graph in which edges are bidirectional or unordered pairs.

Now many real world systems and problems can be modelled using a graph. Graph can be used to represent any collection of objects that have some kind of *pair wise relationship*.

> Examples : a social network like **Facebook** can be represented as an undirected graph. A user would be a node in the graph and if two users are friends, there would be an edge connecting them. A real social network would have millions and billions of nodes. Social Network is an undirected graph because of friendship is a mutual relationship. If I'm your friend, you are my friend too, so connections have to be two way now.

Weighted and Unweighted Graphs

Now sometimes all connections in a graph cannot be treated as equal, so we label edges is associated to connections as cost or way is called a weighted graph. If there is no cost distinction among edges, then graph is unweighted. These edges complicated graph theory.

Then we can have a self loop.

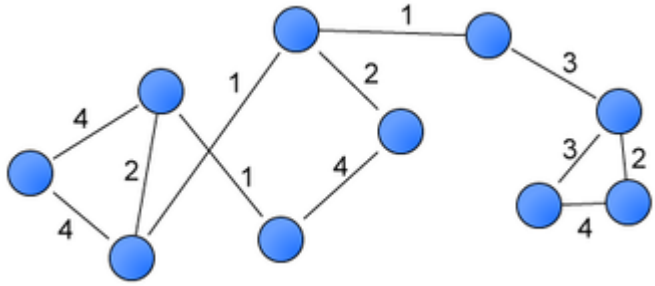
> For example, interlinked web pages on the Internet or the worldwide web can be represented as a directed graph. A page with a unique URL can be a node in a graph and we can have a directed edge if a page contains link to another page. Now we can have a self loop in this graph because it's very much possible for a web page to have a link to itself.

Now the next special type of edge is a multi edge graph. If nodes or edges occurs more than once in a graph, it is called a multi-edge graph. we can have a multi edge in both directed and undirected graphs.

> Example : Lets say we are representing flight network between cities as a graph. A city would be a node and we can have an edge if there is a direct flight connection between any two cities.

> But then there can be multiple flights between a pair of cities and these flights would have different names. And may have different costs if I want to keep the information about all the flights in my graph, I can draw multi edges. I can draw one directed edge for each flight and then I can label and edge with its cost or any other property.

If a graph contains no self loop or multi edge, it's called a simple graph.



Number of Edges

Given number of vertices in a simple graph, that is a graph with no self loop, multi edge. What would be minimum possible number of edges? Nodes can be totally disconnected, so minimum possible number of edges in a graph is zero .

Now if this is a directed graph, what do you think can be maximum number of edges here? Well, each node can have. Directed edges to all other nodes. In general if there are n vertices then maximum number of edges in a directed graph would be n(n-1).

Now what do you think would be the maximum for an undirected graph? In an undirected graph we can have only one bidirectional edge between a pair of nodes. We can't have two edges in different directions, so here the maximum would be half of the maximum for directed. Number of edges would be in the range zero to $N * (N - 1)/2$. Remember, this is true only if there is no self loop or multi edge.

Dense and Sparse Graph

A graph is called dense if number of edges in the graph is close to maximum possible number of edges. That is it the number of edges is of the order of square of

number of vertices. And a graph is called sparse if number of edges is really less. There is no defined boundary for what can be called dense or what can be called sparse. It all depends on context. But this is an important classification.

Path , Walk and a trail

Path in a graph is a sequence of vertices where each adjacent pair in the sequence is connected by an edge. Path is called simple path if no vertices are repeated and if vertices are not repeated, then edges will also not be repeated. So in a simple path both vertices and edges are not repeated.

In graph theory, there is some inconsistency in use of the term path. Most of the time when we say path, we mean a simple path.

If Repetition is possible we use the term walk so a path is basically a walk in which new vertices or edges are repeated.

A walk is called trail if vertices can be repeated, but edges cannot be repeated.

Strongly Connected Graph

A graph is called strongly connected if there is a path from every node to any other node in the graph.

Cycles in graphs

If a graph is closed it is called cyclic. A graph with no cycle is called an acyclic graph.

Representation of Graph into memory

A graph can be represented by two lists a vertex list and an edge list.

An edge is identified by it's two end point . It can be represented in a pair of vertex namely startVertex and an endVertex.

```
class Edge{
  String startVertex;
  String endVertex;
};
```

In undirected graph any vertex can be start or end.

What if we have some cost associated with this graph. How should we store this data .

For this we can have one more field in edge object .

```
class Edge{
  String startVertex;
  String endVertex;
  int weight;
};
```

Time and space (Memory) complexity is important for analysing this type of data.

Minimum no of item in list in vertex object is equal to number of nodes. space complexity is equal to O (|V|) where | V | is no of nodes. For a better design we can use int instead of strings in edge object. so here space complexity will be O(|V| + |E|) i.e the sum of number of vertices and edges.

```
class Vertex{
  int vertex;
}
class Edge{
  int startVertex;
  int endVertex;
  int weight;
};
```

Graph representation using 2d Array

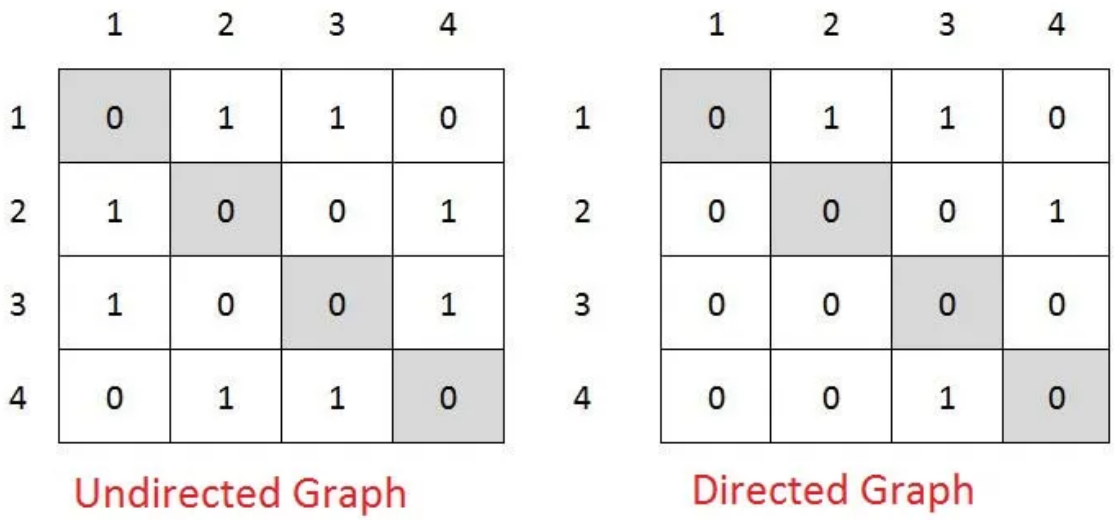
> How much time will you take to find all nodes adjacent to a given node ? Running time will be proportional to number of edges since we will perform linear search.

if |V| = n

$o <= |E| <= V(V-1)$

Time complexity in this case however is not efficient since we are performing linear search in most of the cases , since number of edges can be very very large .

So a better way to store the data would be using a 2-d Array or Matrix of size VxV



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

Undirected Graph

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

Directed Graph

This matrix is called adjacency matrix.

here 1 represent an edge , 0 represent no edge or no connection. To see all the edges in undirected graph we need to just see the upper half triangle in matrix.

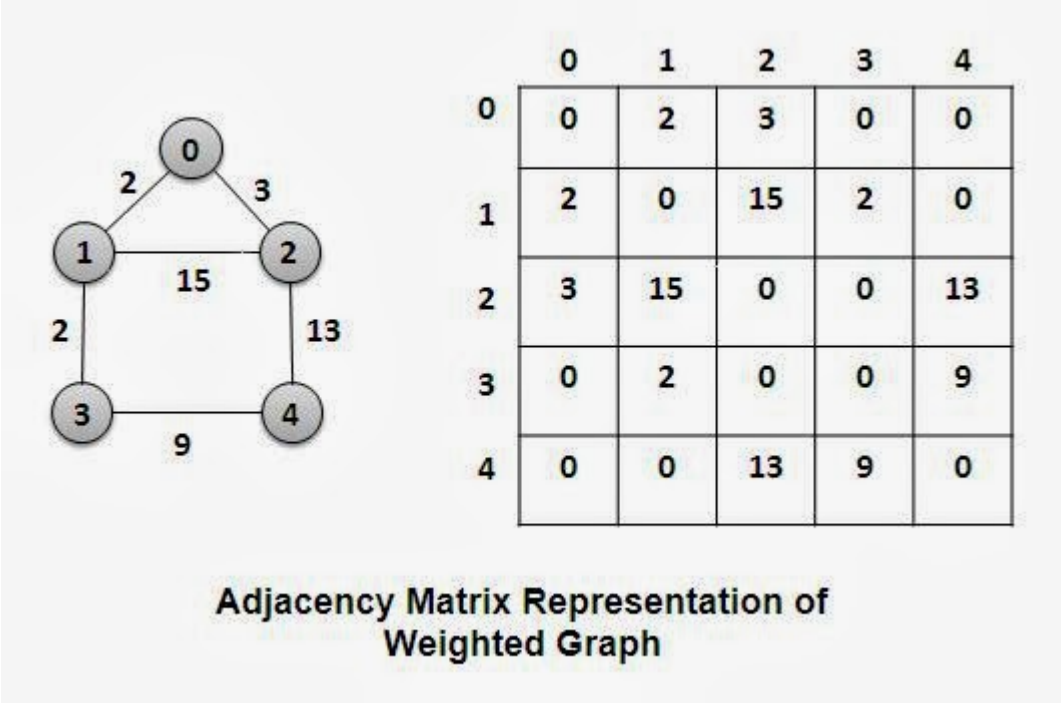
For directed graph we need to go through entire matrix to see all the edges.

time complexity here is O (V) .

Look up if there exist a connection between two nodes take constant time or O(V) when indices is not given .

For weighted adjacency matrix , we represent the connection or vertices by a number equivalent to weight and if there is no connection we assign a really large number or value that isn't possible.

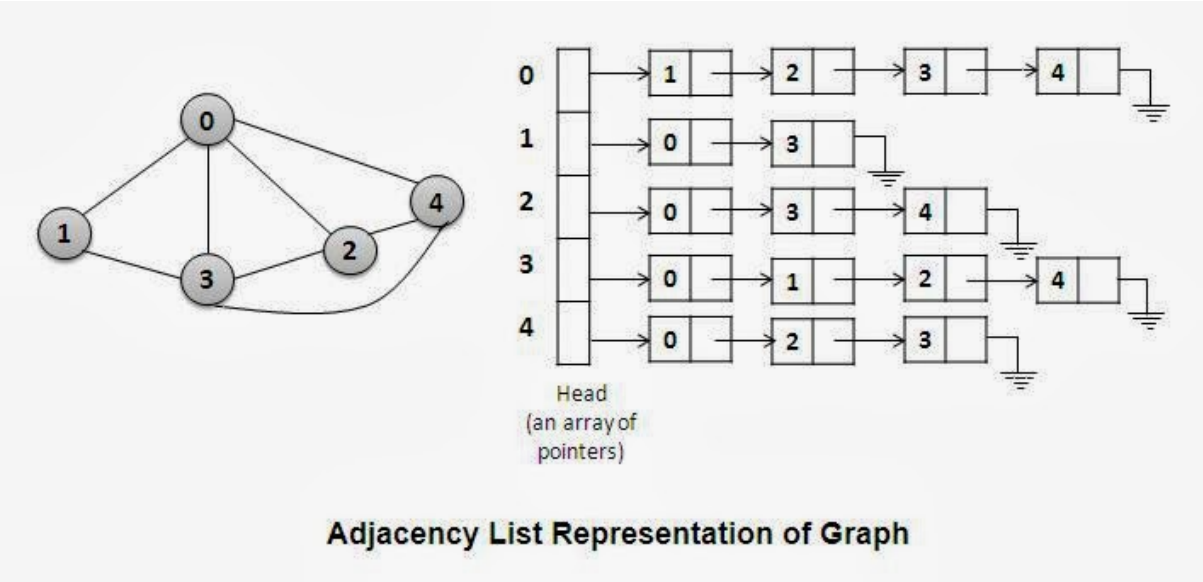
However the space complexity increases in this case.



Adjacency Matrix Representation of Weighted Graph

Adjacency List

Here we store all the connection from one node to all as a list only if it exist.



Adjacency List Representation of Graph