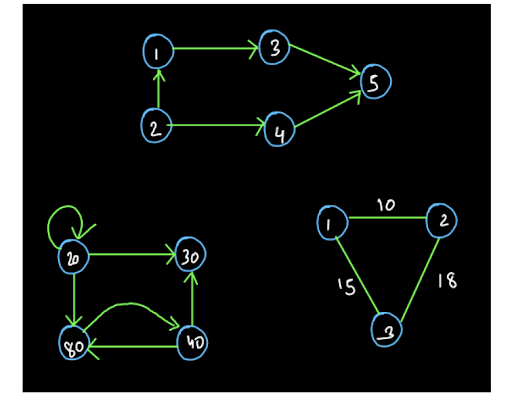
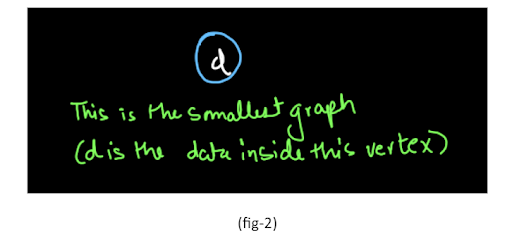
**1. PROBLEM DISCUSSION**

What is a graph data structure? A graph is a set of vertices and edges.

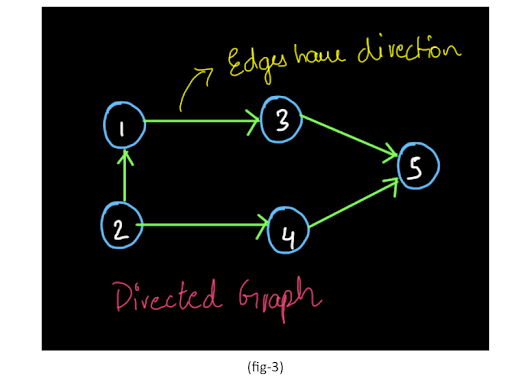
****

So, we can define the graph G = (V,E) where V is the set of vertices and E is the set of edges. As in the above diagram, the circles in which the data is stored are the vertices of the graphs and the lines connecting them together are called edges of the graphs. Now the question arises if a graph is a set of vertices and edges then what can be the minimum number of vertices and edges a graph can have? See, a graph is a data structure. This means that it is a way of storing data. Who stores data in a graph? As you saw in the above diagram, a vertex stores the data in the graph. So, presence of at least one vertex is necessary whereas presence of an edge is not. Therefore the smallest graph is a graph with only one vertex and zero edges.

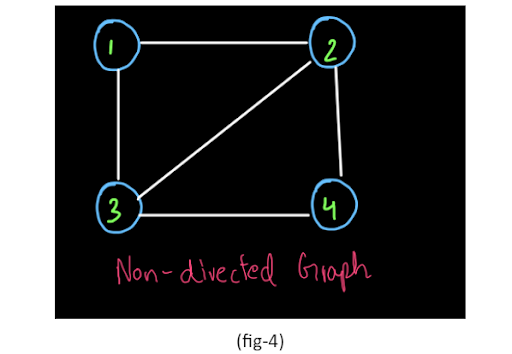
****

So, the set V i.e. the set of vertices for a graph G cannot be empty but the set E can be empty.

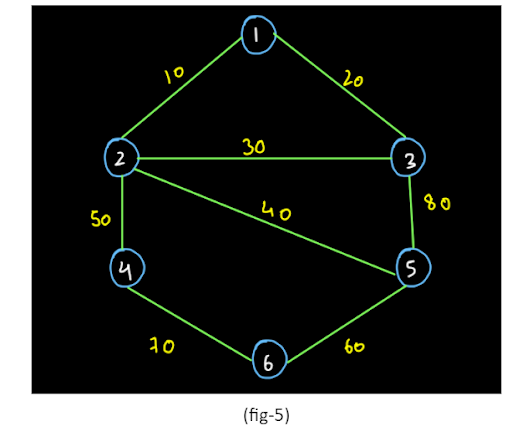
Directed Graph: If the edges of a graph has directions then the graph is known as directed graph:

****

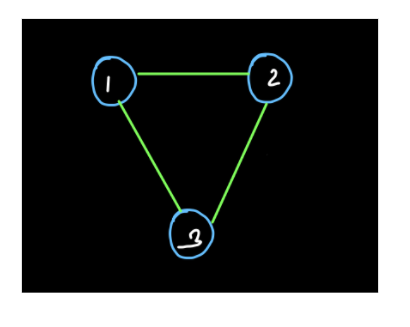
Non-directed Graph: A graph whose edges do not have direction is called an undirected/non-directed graph. They are assumed to be bi-directional.

****

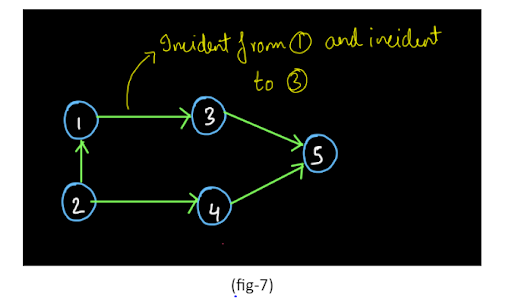
In the diagram shown above, we can assume any direction of edges that we want to. For example, 1 -> 2 and 2 -> 1 are both correct for the non-weighted graph. Note: The two vertices connected to each other via an edge are called adjacent vertices. Weighted Graph: A graph whose edges have weights mentioned on them is called as a weighted graph

****

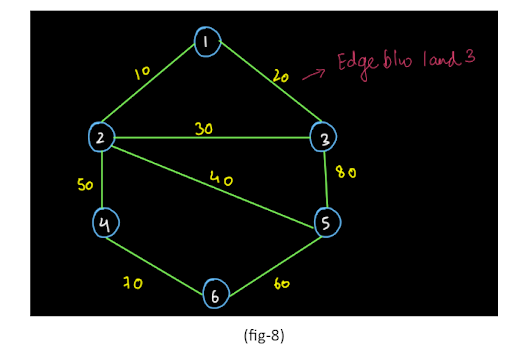
In the above example a weighted undirected graph is shown in the figure. What do these weights signify? See the edge 1 -> 2 has a weight 10. This means that the cost of travelling from 1 -> 2 or from 2 -> 1 is both 10. Non-weighted Graph: A graph whose edges do not have any weights mentioned on them is called a non-weighted graph.

****

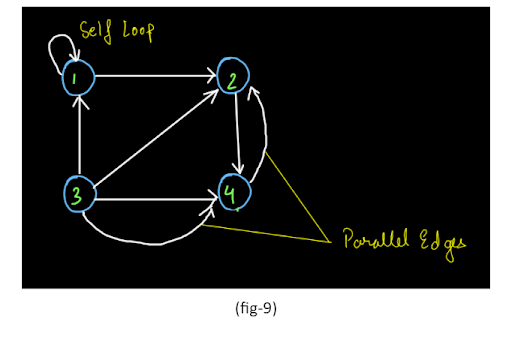
Incident to and Incident From: In a directed graph, an edge is said to be incident from a vertex if it emerges out of the vertex and it is said to be incident on a vertex if it points at that vertex.

****

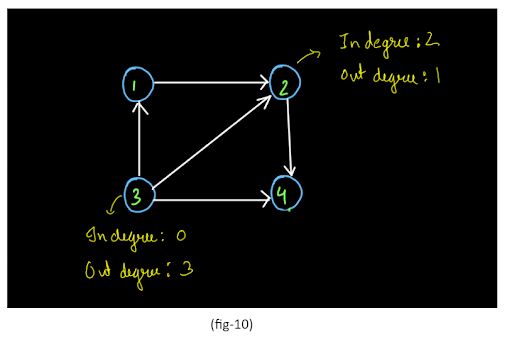
In a noon-directed graph, since the edges are bi-directional, the "incident to" and "incident from" phrases are not used. We just say that there is an edge between two vertices.

****

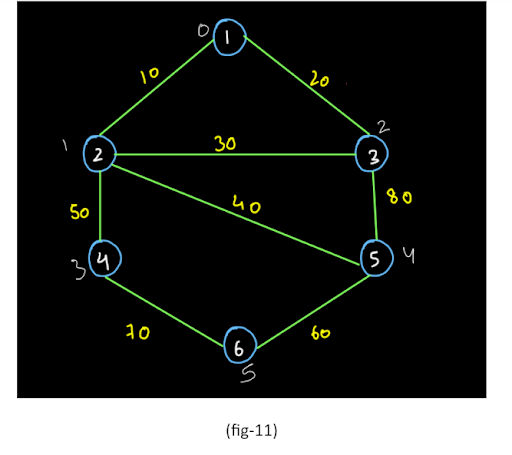
Self loop and Parallel Edges:

****

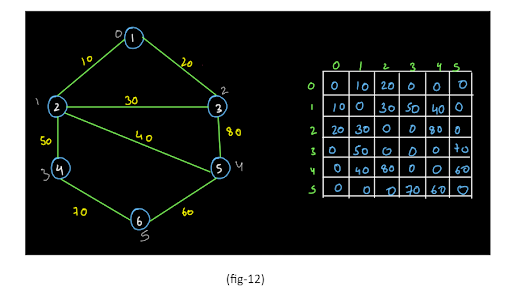
As you can see, the direction of the parallel edges can be the same or opposite. Both are called parallel edges. In-degree and Out-degree: In a directed graph (also known as digraph), the number of edges coming into the vertex is called as the in-degree of a vertex and the number of edges going out of a vertex is called as the out-degree of a vertex.

****

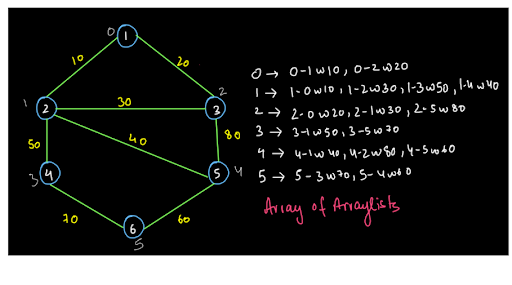
For a non-directed graph, since the edges are bi-directional therefore the in-degree and out-degree do not make much sense as both will always be equal. So, for a vertex v which belongs to a non-directed graph G, we have only degree i.e. number of edges the vertex is connected to. So, these are a few important concepts and definitions regarding the graph data structure that you need to know for now. We will learn more concepts and definitions and algorithms as we move forward in this section. Now, let us talk about the representation of a graph. Representation of a Graph: Consider the graph given below:

****

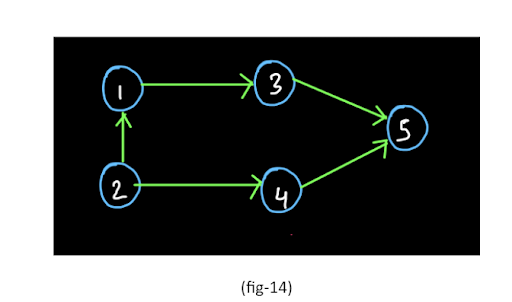
Let us try to represent this graph into something called an adjacency matrix. Adjacency Matrix: We can represent the graph shown in the above example into an adjacency matrix as shown below:

****

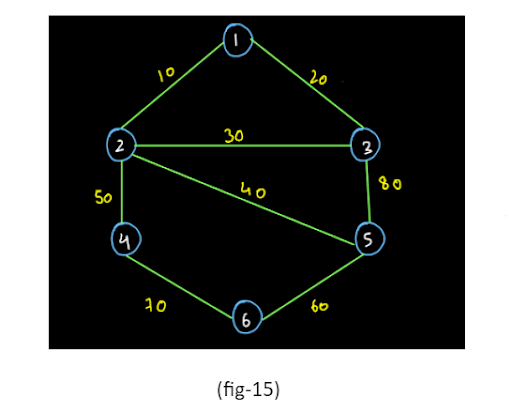
Let us try to understand how we have stored the data. First of all we see that all the elements in the diagonal of this matrix are 0. Why? In the diagonal the value of row and column is equal. This means that the diagonals represent the self loops. Since there are no self loops in the graph shown above, we have all the elements of the diagonal as 0. Now let us start from the 1st column in the 0th row (since the 0th column in the first row is a part of the diagonal). This element represents an edge from the 0th vertex (i.e. row) to the 1st vertex (i.e. column) in the graph. The vertex with data=1 is the 0th vertex and the element with data =2 is the first vertex in the graph. Since there is an edge from 0th vertex to the 1st vertex in the graph, this element will not be 0. Since the weight of that edge is 10, we insert 10 into m[0][1]. We have taken a weighted non-directed graph in our example. Also,, we have assumed that there will not be any parallel edges for this representation. Why? Think!! Since the graph is a weighted graph, we have inserted the weight of the edge i.e. 10. If it were a non-weighted graph, we would have inserted 1 denoting that there is an edge present from 0th vertex to the 1st vertex. Also, this is a non-directed graph. So, the edges are bi-directional. This means the value inserted at ith row and jth column will also be inserted at jth row and ith column. (Since the edge of weight 10 is both from vertex 0 to 1 and from vertex 1 to 0). So, this is how we fill the adjacency matrix. We have filled the entire matrix in the diagram shown above. We recommend you try to fill this matrix on your own by looking at the graph. Let us learn another representation called the adjacency list representation. Note:Use the adjacency matrix representation only if the number of vertices is less than 10 thousand. Adjacency List Representation The other popular way of representing a graph is called an adjacency list. This is an array of arraylists. How? Have a look at the diagram given below:

****

We have made an array of size 6 since there are six vertices. This array will store the arraylists at every index. For instance, at index 0, the arraylist is as given above. The first element is 0-1w10 meaning that the source vertex is vertex 0 and the destination is the vertex 1 and the weight of the edge is 10. The source vertex for all the elements in the arraylist at index 0 will be 0 only. This means that the source vertex for elements in the arraylist at index i will be i only. So, you can clearly understand further elements in this array of arraylists. If you have any doubts, refer to the INTRODUCTION TO GRAPHS AND REPRESENTATION VIDEO to clear all your doubts. Where can we use graphs? Consider the graph given below:

****

Let us say the vertices represent a particular city. So, we want to find out whether there is a path from city 2 to city 5. These kinds of problems can be solved using the graph data structure.

****

ConsoleJava

Now, let us say that we have this kind of a city network and the weights of the edges represent the distance between the cities (vertices). Let's say we have to find the shortest path from city 1 to city 6 (in terms of number of kilometers). It may also happen that we want to find the shortest path from city 1 to city 6 in

terms of the number of cities that we have to go through for visiting city 6. These kinds of problems are also solved using graphs.

The problems such as finding the minimum number of wires to connect a LAN network (minimum cost spanning tree) and many more are also solved using graph data structure.