Graphs

Problem: Friends

We want to create a Simple program in which we can add friends, delete friends, view the friends, and view the friends of friends of a specific person

```
1. Add Friends
2. Delete a Friend
3. View all Friends
4. See the Friends of a Specific Person
5. See the Friends of Friends of a Specific Person
6. Exit
Enter your Option:
```

Problem: Friends

On pressing option 1.

```
Enter the name of the Person
Ali
Enter the name of his/her Friend
Aslam
```

Problem: Friends

On pressing option 2.

```
Enter the name of the Person you want to Delete
Ali
```

Problem: Friends

On pressing option 3.

```
Enter the start name
Ali
Ali Aslam Noor Bilal Fatima Khalid Haseeb Qaiser Asad
```

Problem: Friends

On pressing option 4.

```
Enter the name of the Person whose Friends you want to See
Ali
Aslam Noor
```

Problem: Friends

On pressing option 5.

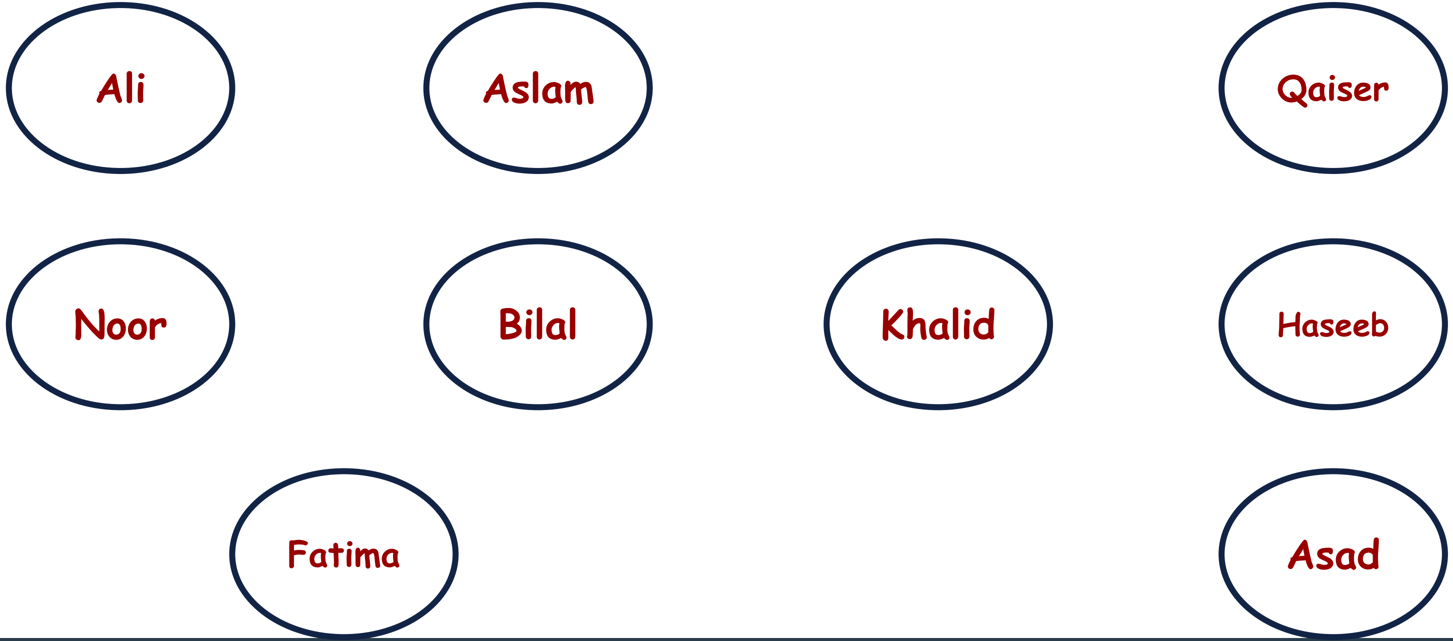
```
Enter the name of the Person whose Friends of Friends you want to See
Ali
Friend: Aslam
Friends of Friends: Bilal
Friend: Noor
Friends of Friends: Fatima
```

Problem: Friends

How can we store the informations of Friends effectively, so that all the shown options can be implemented efficiently?

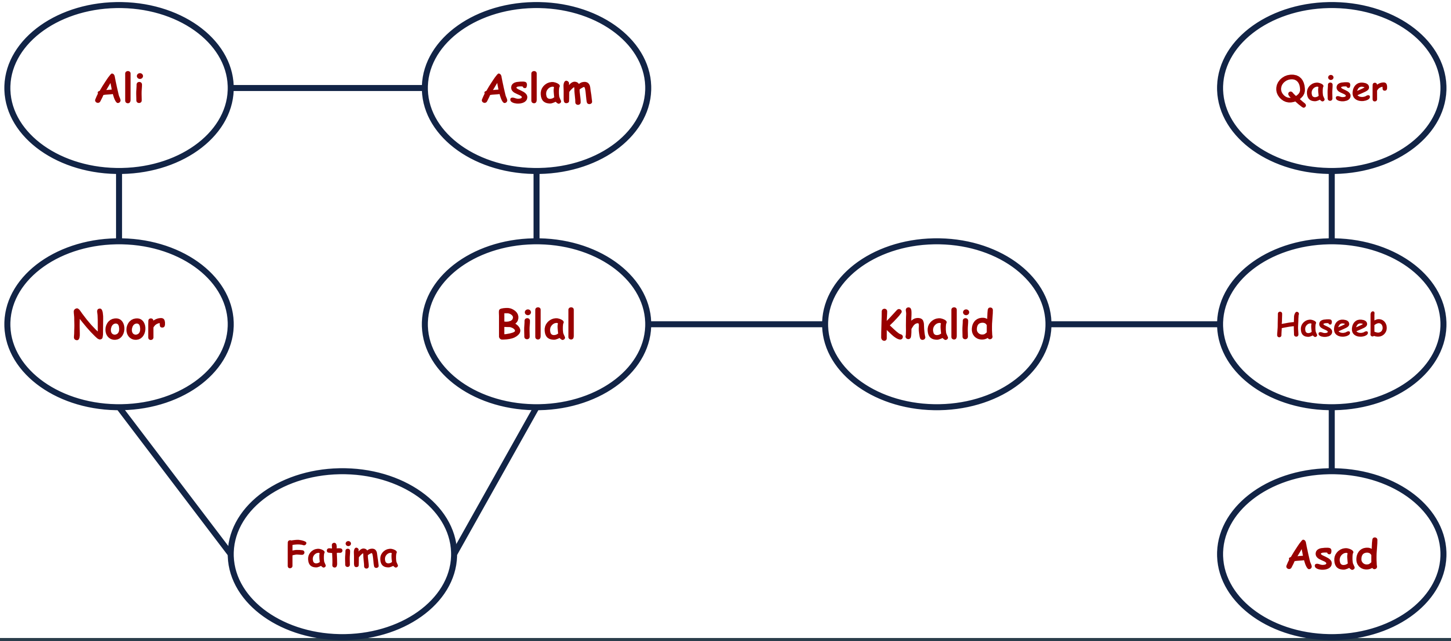
Problem: Solution

First of all, let's note down all the names of the people.



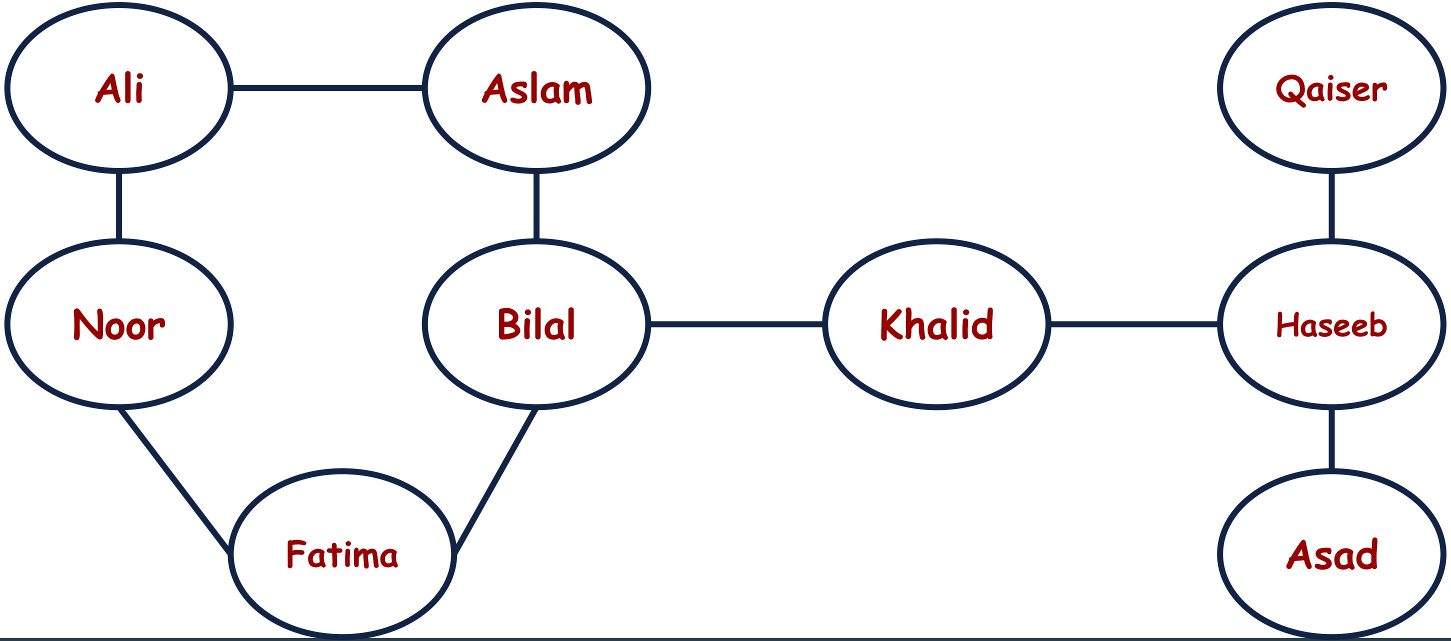
Problem: Solution

Let's connect all the friends through a Line.



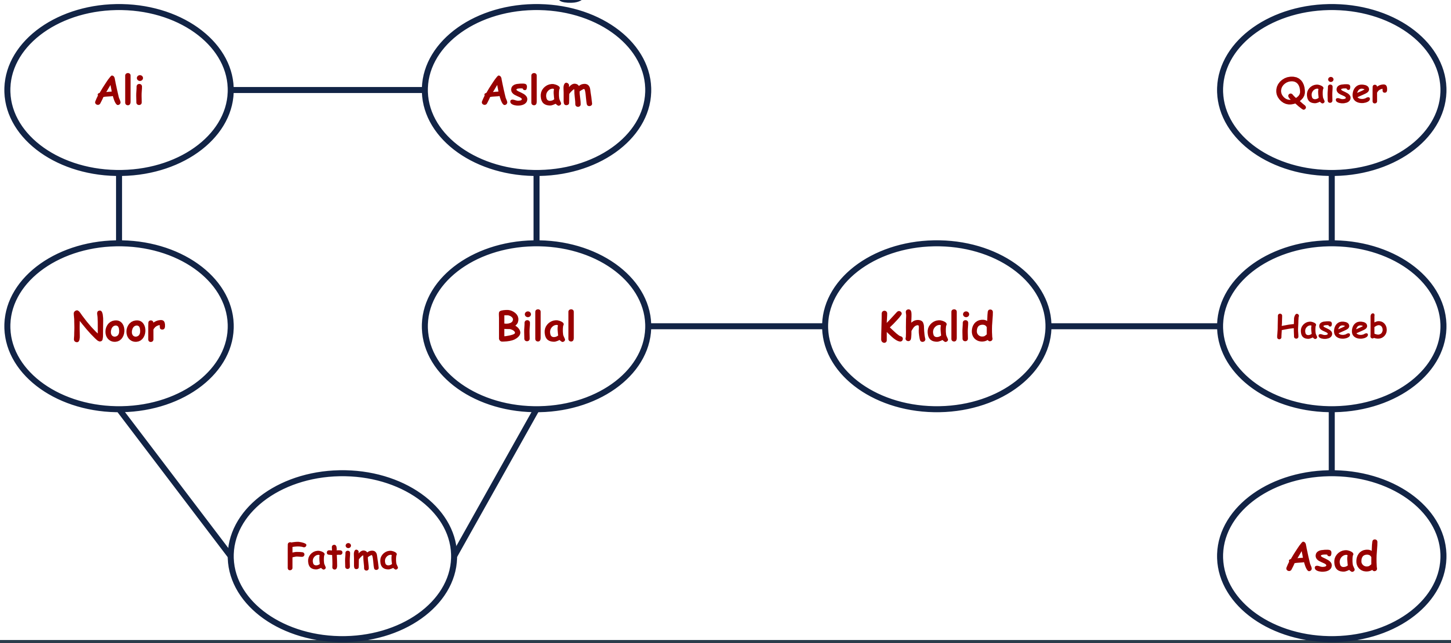
Graphs

This is a new Data Structure called Graph.



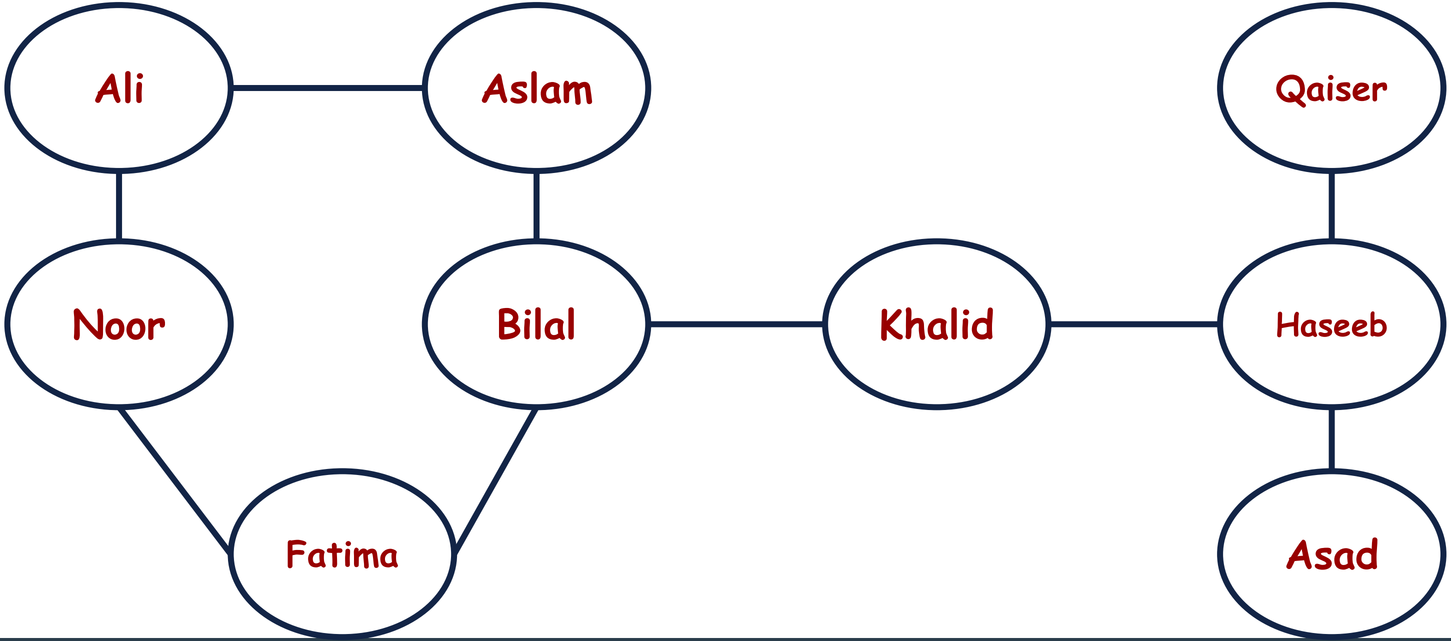
Graphs

A graph is a non-linear data structure made up of nodes/vertices and edges.



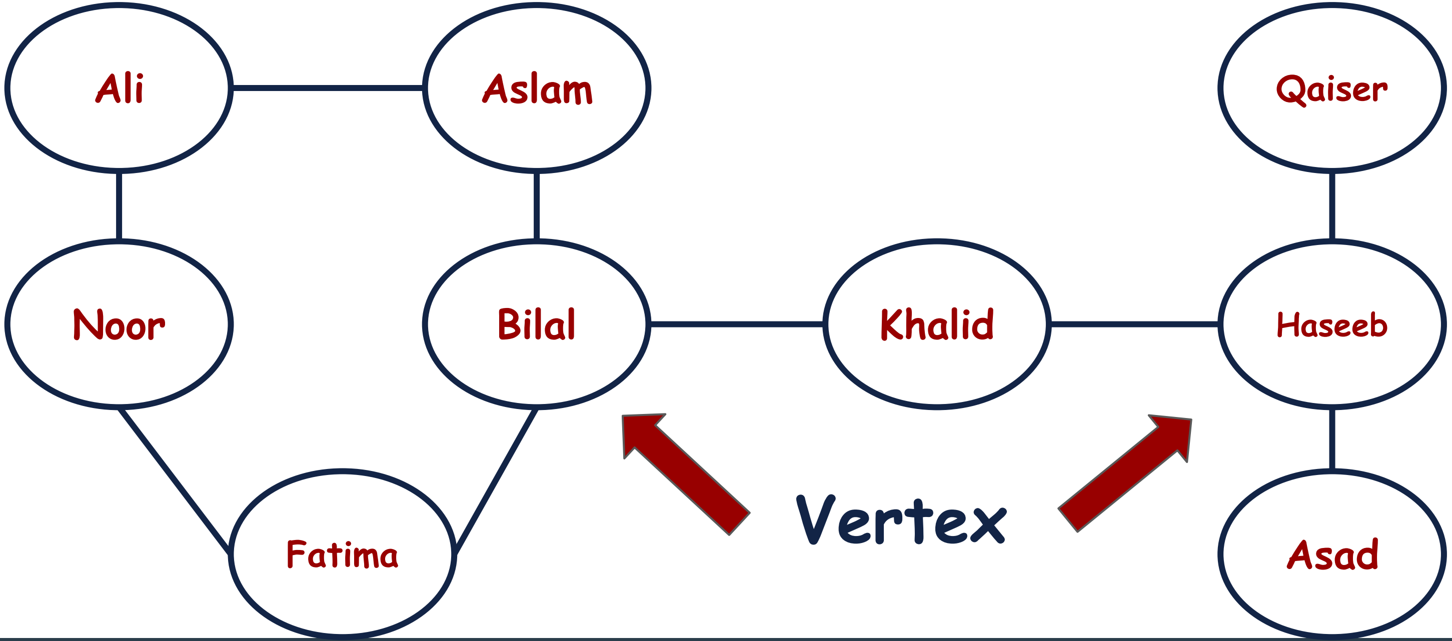
Graphs: Terminologies

Let's discuss all the terminologies of related to Graphs.



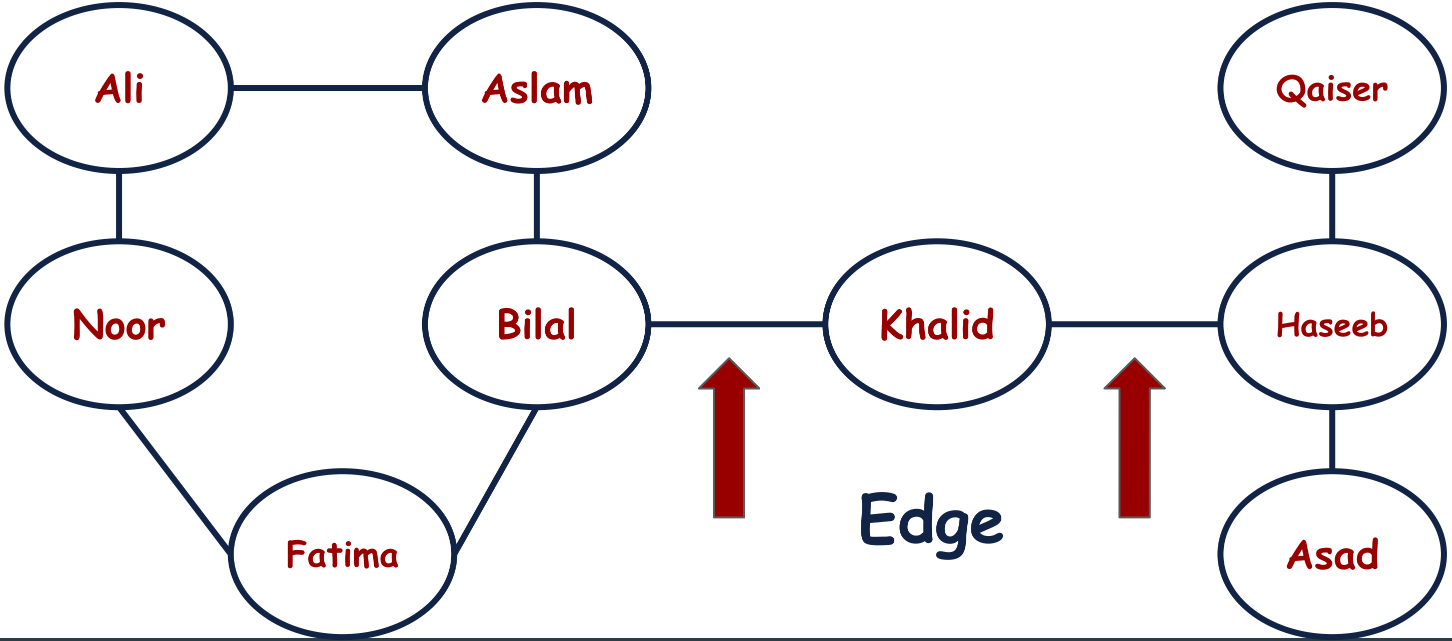
Graphs: Nodes/Vertices

The nodes are also known as vertices



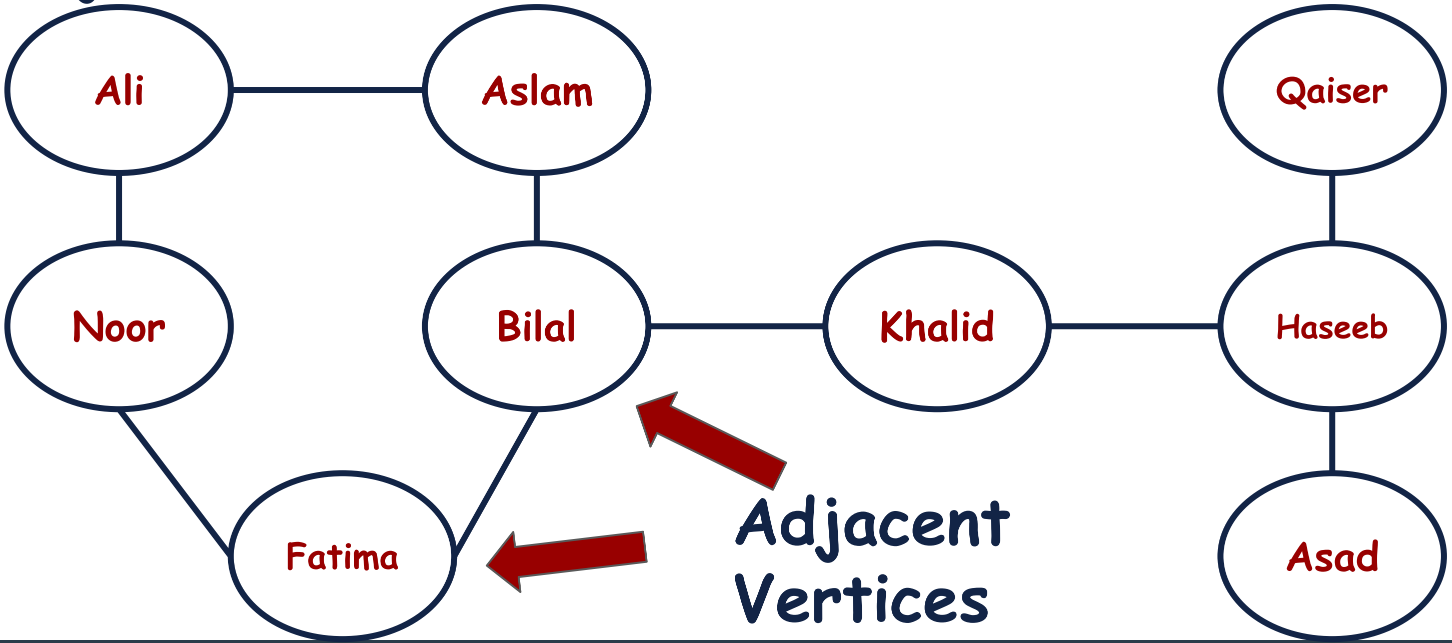
Graphs: Edge

The edges connect any two nodes in the graph.



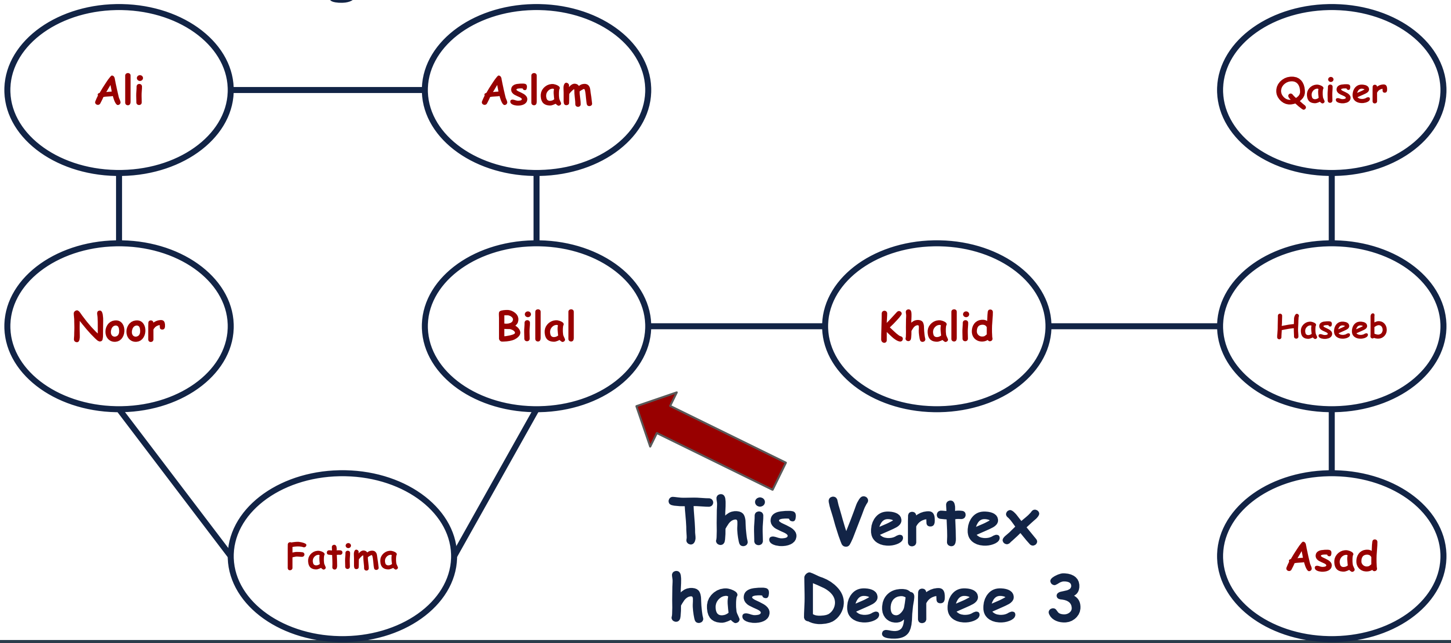
Graphs: Adjacent Vertices

If two vertices are endpoints of the same edge, they are adjacent.



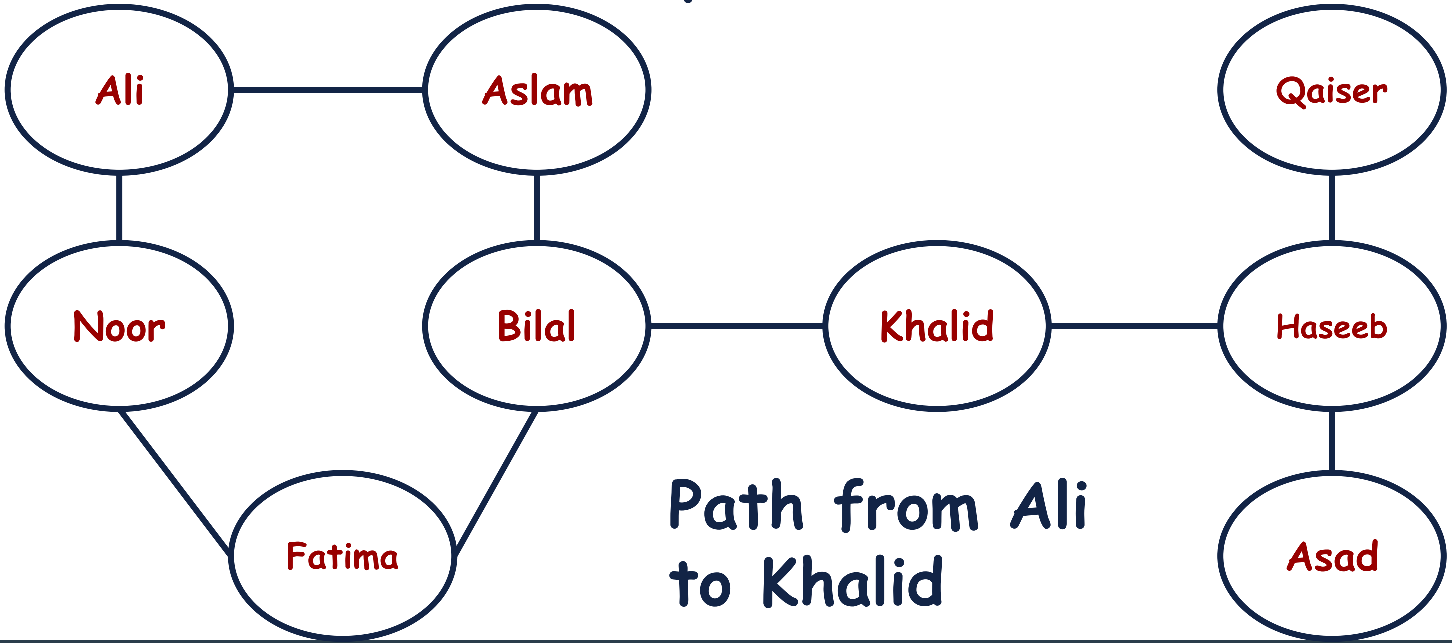
Graphs: Degree

The total number of edges connected to a vertex in a graph is its degree.



Graphs: Path

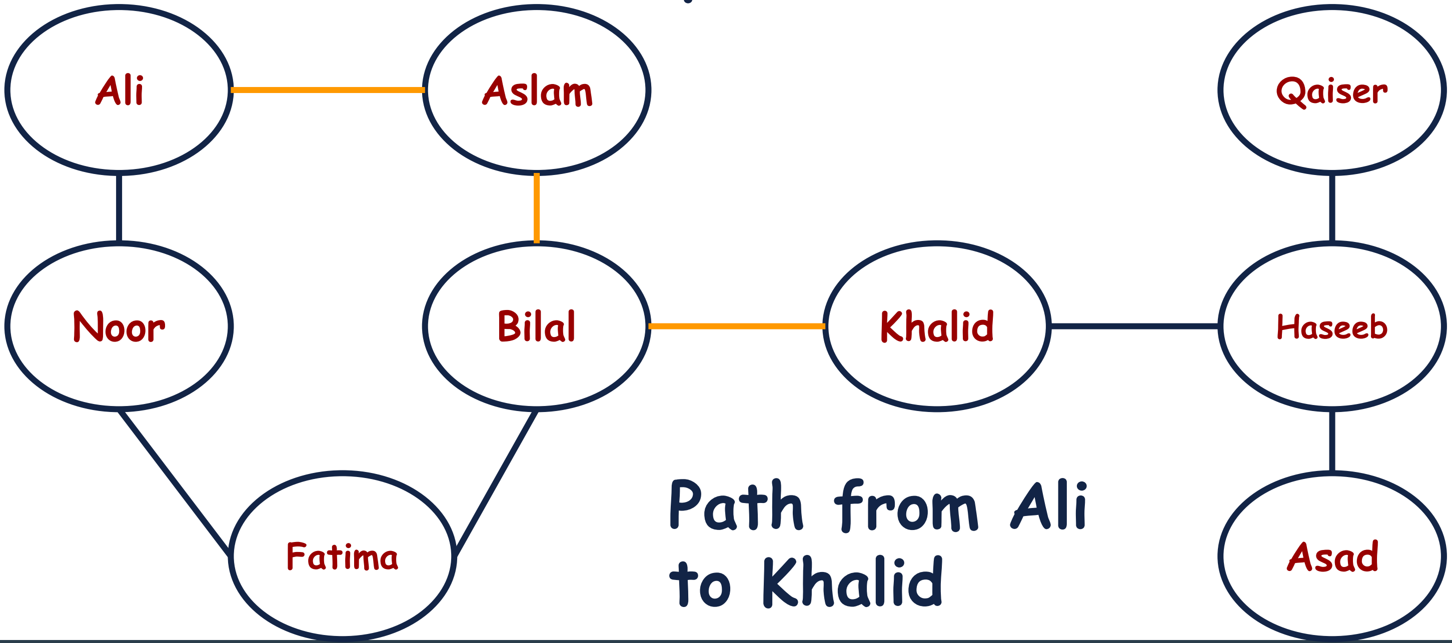
A sequence of edges that allows you to go from vertex A to vertex B is called a path.



Path from Ali to Khalid

Graphs: Path

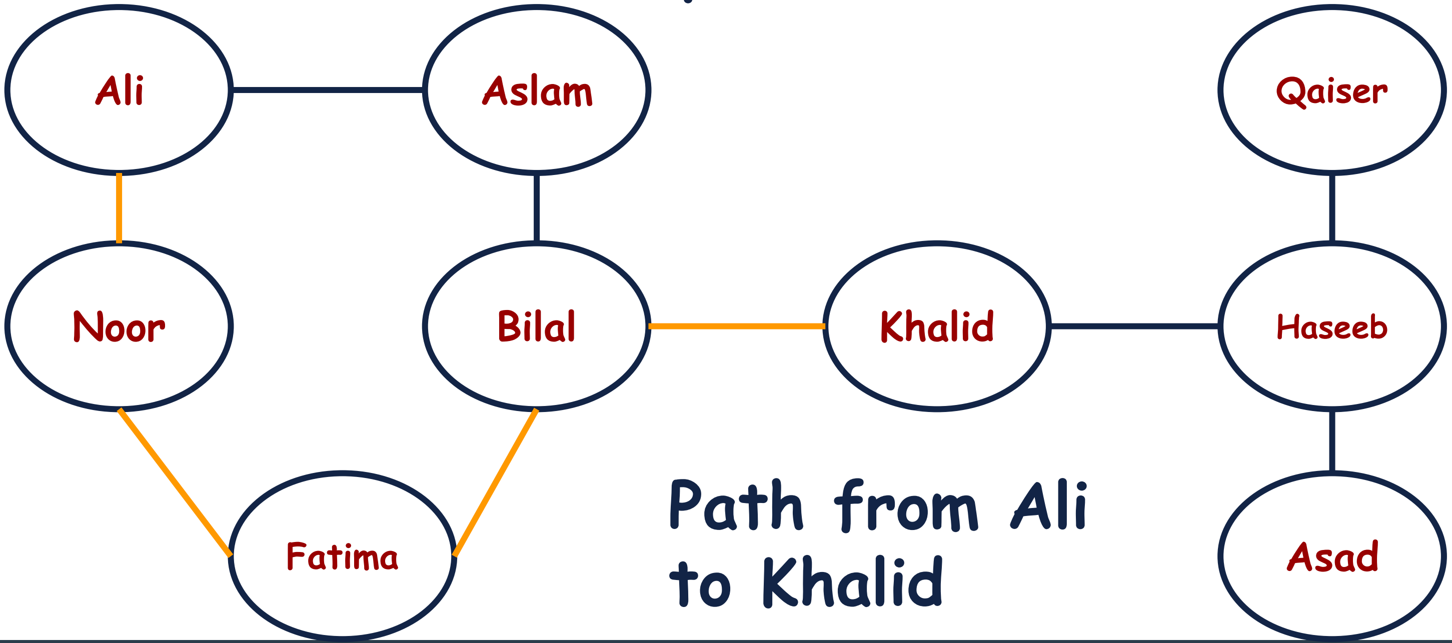
A sequence of edges that allows you to go from vertex A to vertex B is called a path.



Path from Ali to Khalid

Graphs: Path

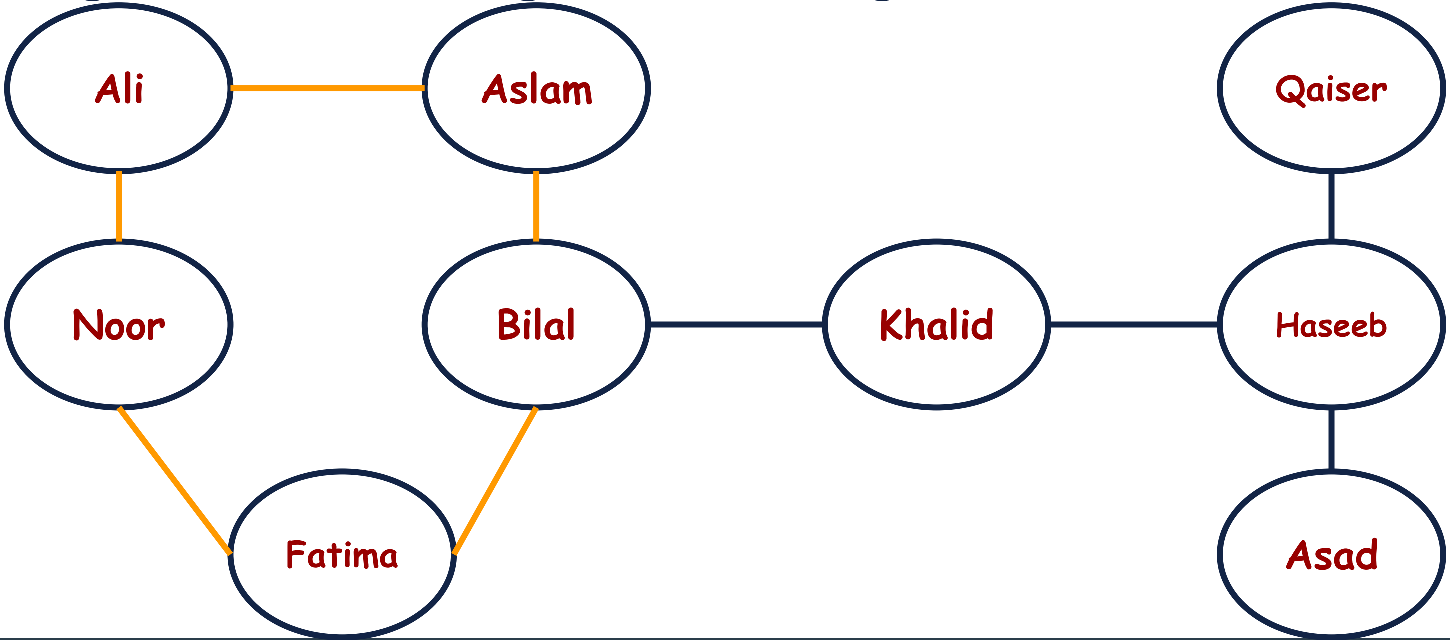
A sequence of edges that allows you to go from vertex A to vertex B is called a path.



Path from Ali to Khalid

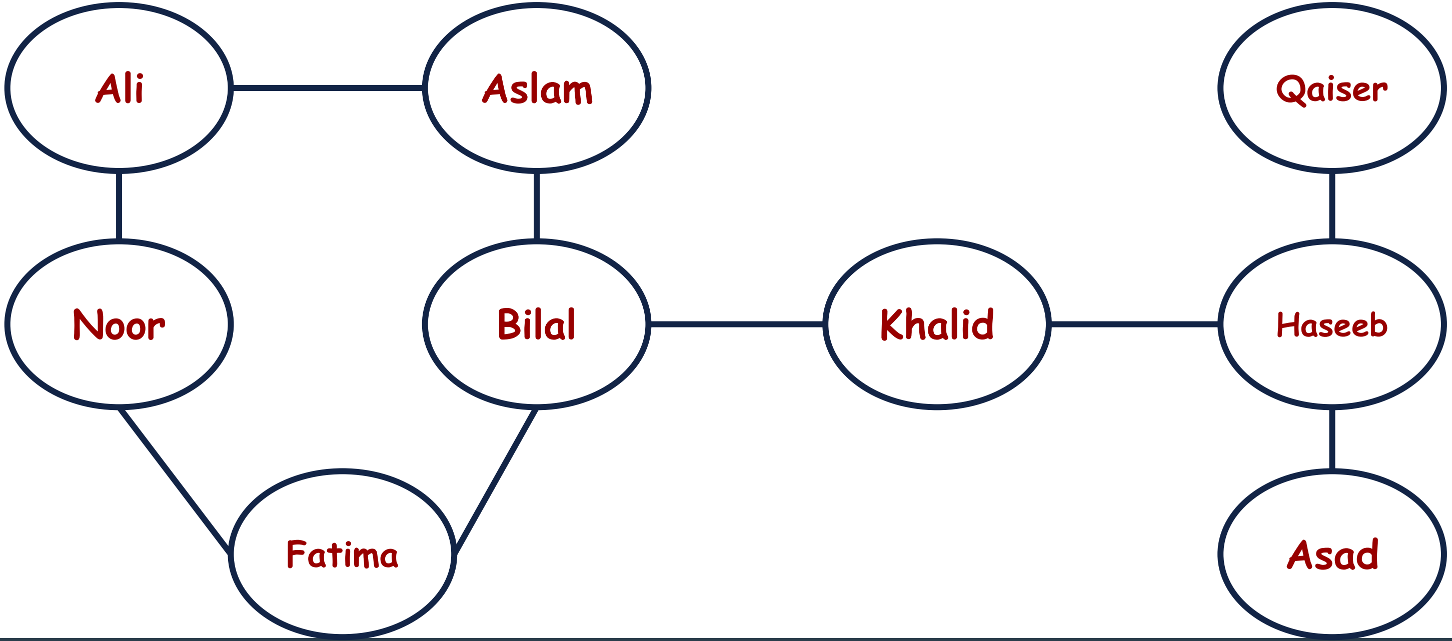
Graphs: Cycle

Traversing a graph such that we do not repeat a vertex and edge but starting and ending vertex are same.



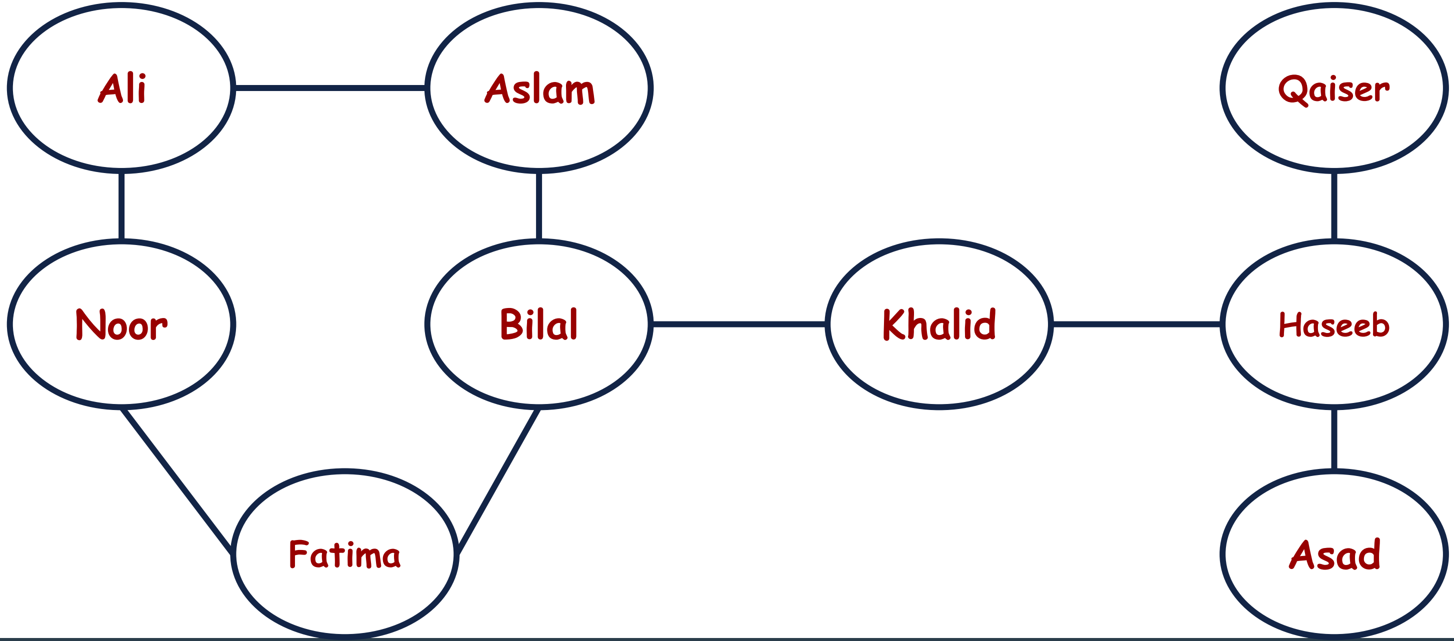
Graphs

In this example, Ali is the Friend of Aslam and Aslam is the friend of Ali. Such is the case in Facebook.



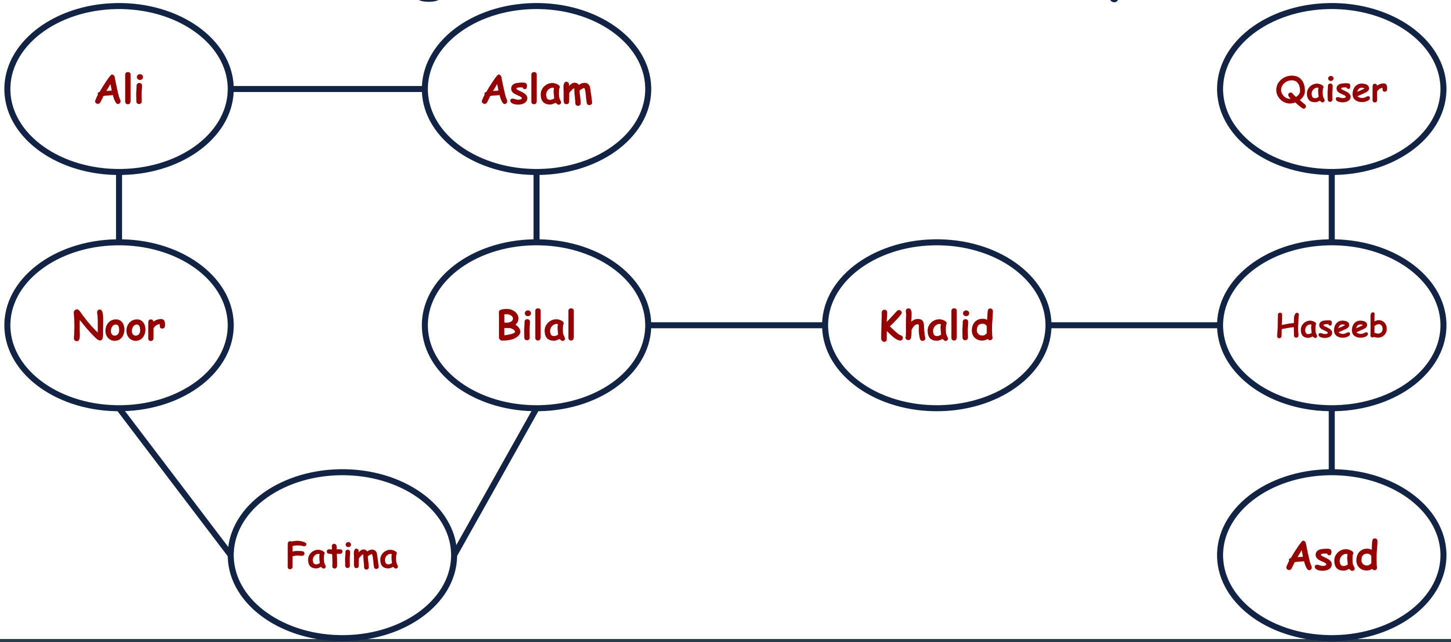
Graphs: Undirected Graphs

This type of Graph is called Undirected Graph.



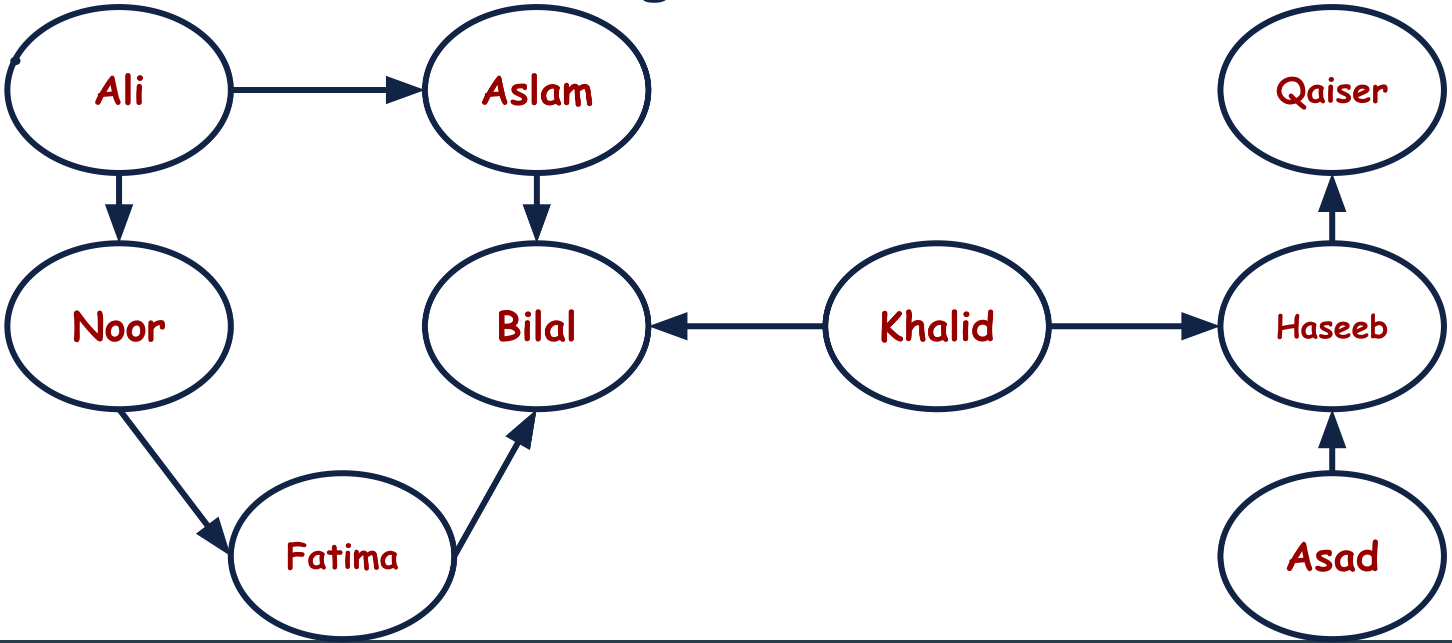
Graphs: Undirected Graphs

Undirected graphs have edges that do not have a direction. The edges indicate a two-way relationship.



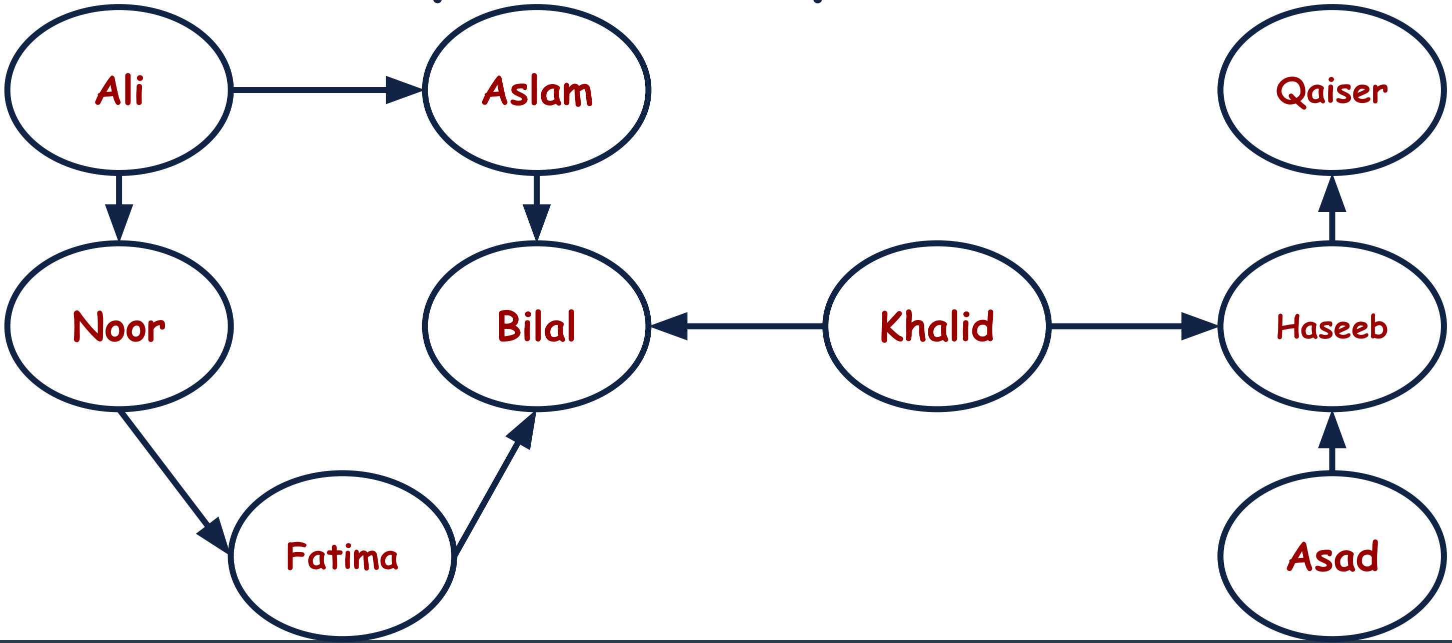
Graphs

Let's say we have instagram. And Ali is following Aslam but Aslam is not following Ali instead he is following Bilal.



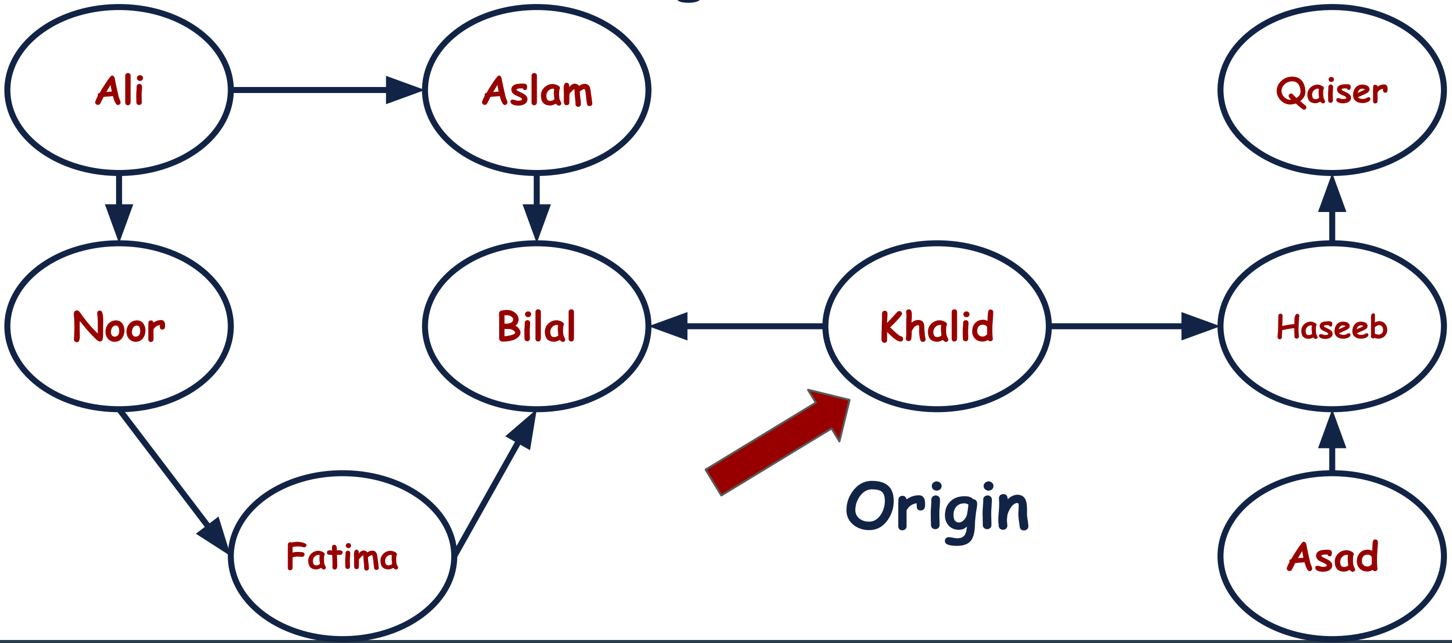
Graphs: Directed Graphs

Directed graphs have edges with direction. The edges indicate a one-way relationship.



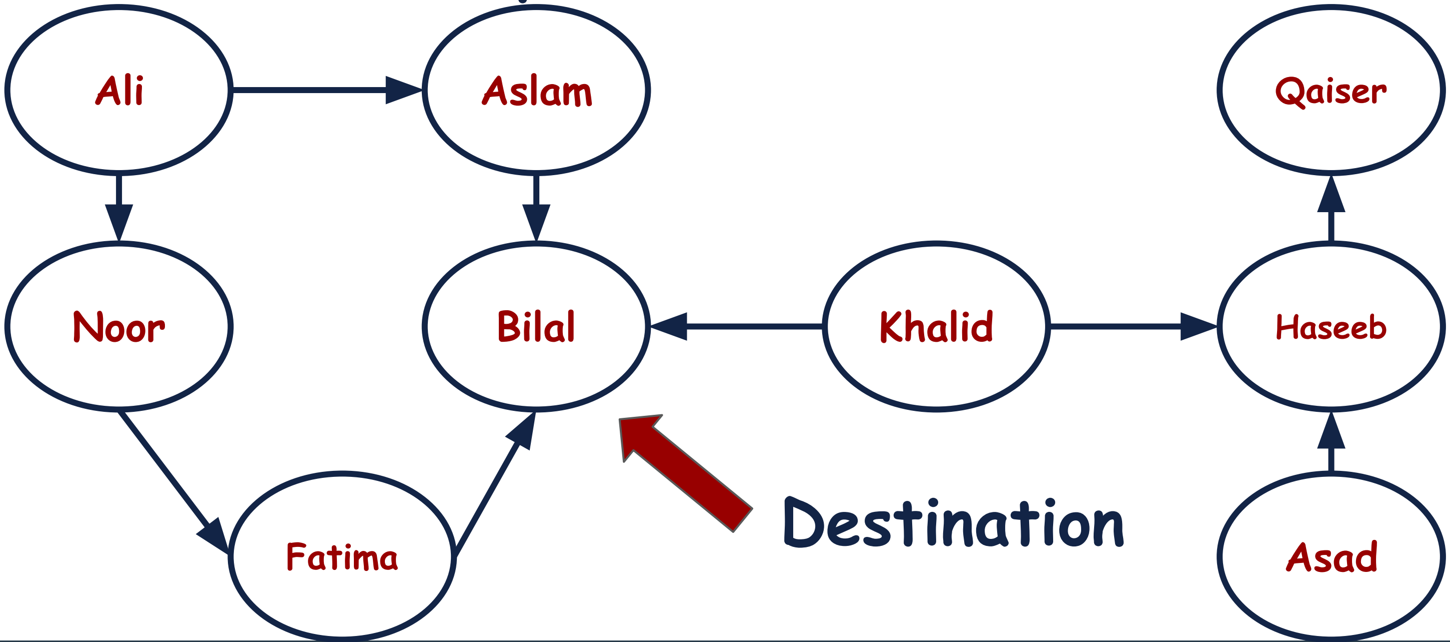
Graphs: Origin

In Case of Directed Graphs, the first endpoint of the edge is said to be the origin of it.



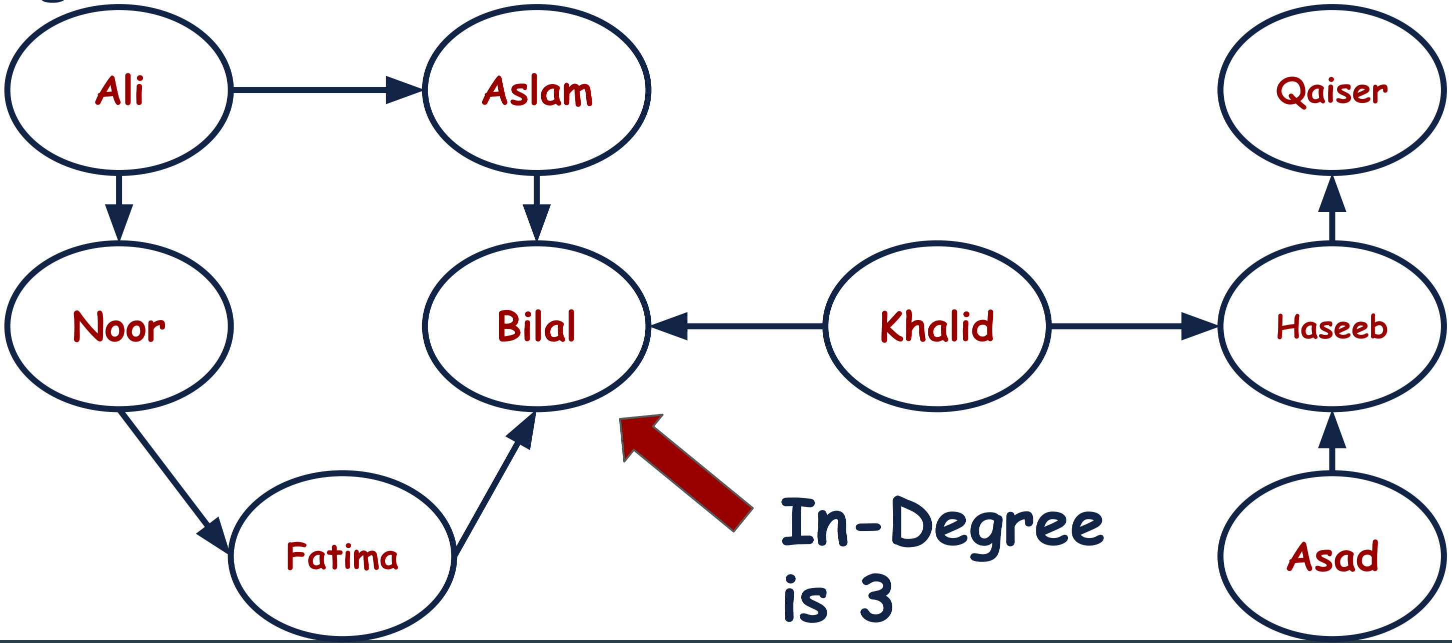
Graphs: Destination

If an edge is directed, its first endpoint is the origin of it and the other endpoint is the destination of the edge.



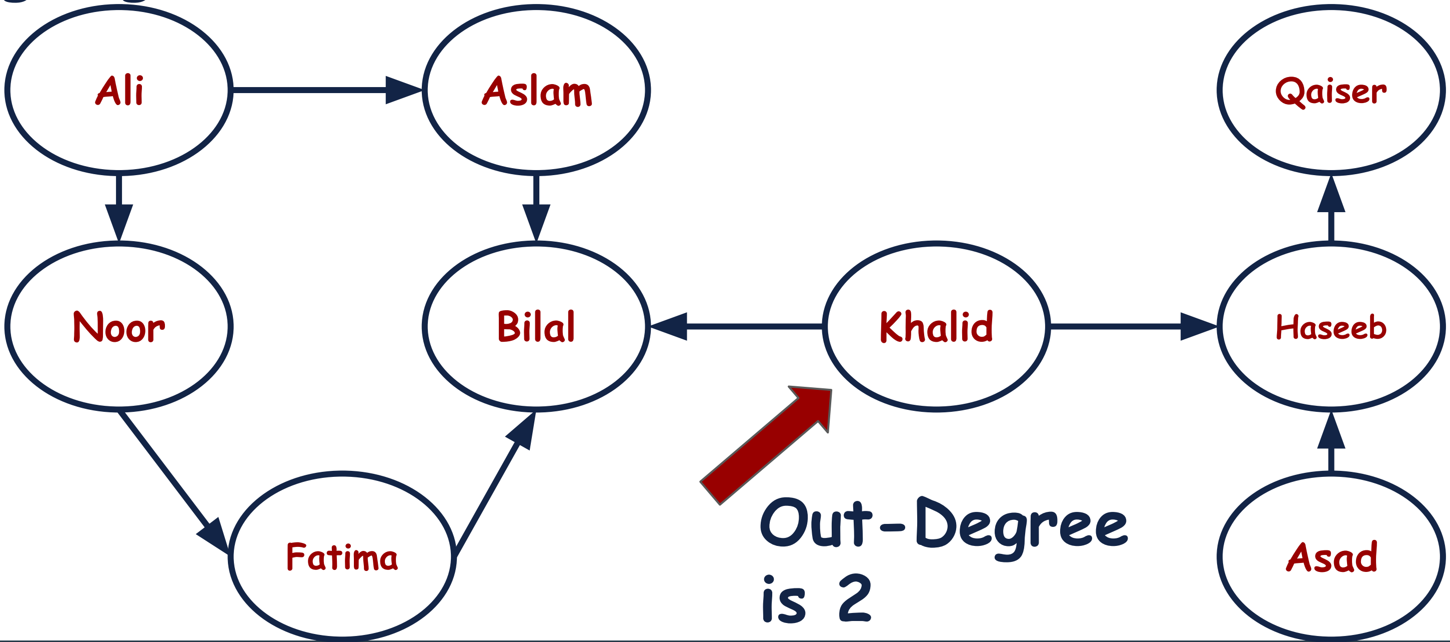
Graphs: In-Degree

In-degree of a vertex is the number of edges which are coming into the vertex.



Graphs: Out-Degree

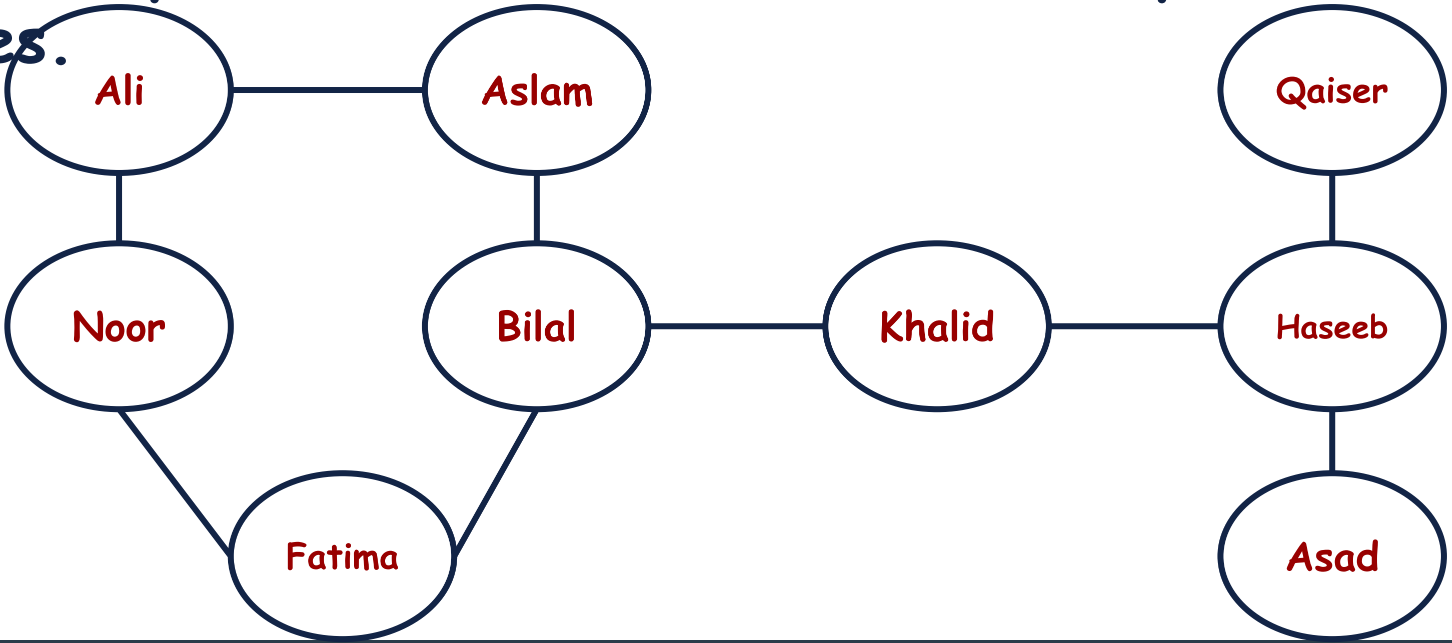
Out-degree of a vertex is the number of edges which are going out from the vertex.



Out-Degree is 2

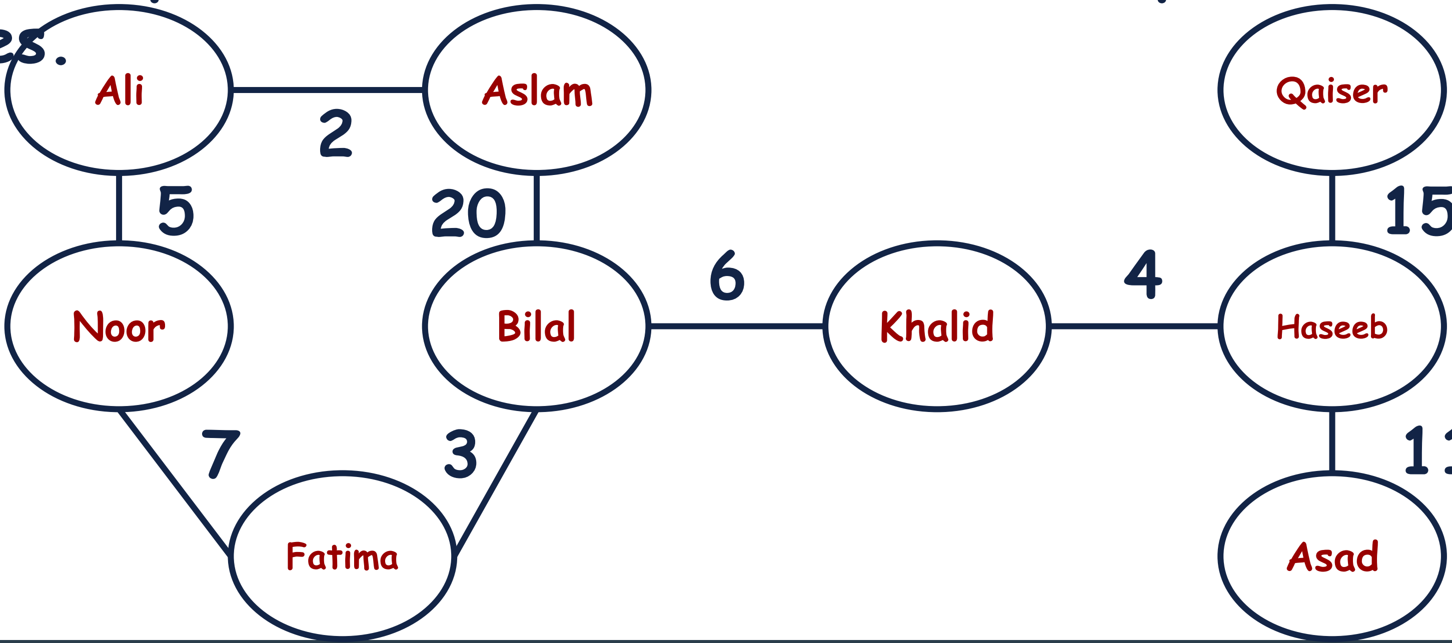
Graphs

Facebook is maintaining the information of years of friendship between friends. It can be represented on edges.



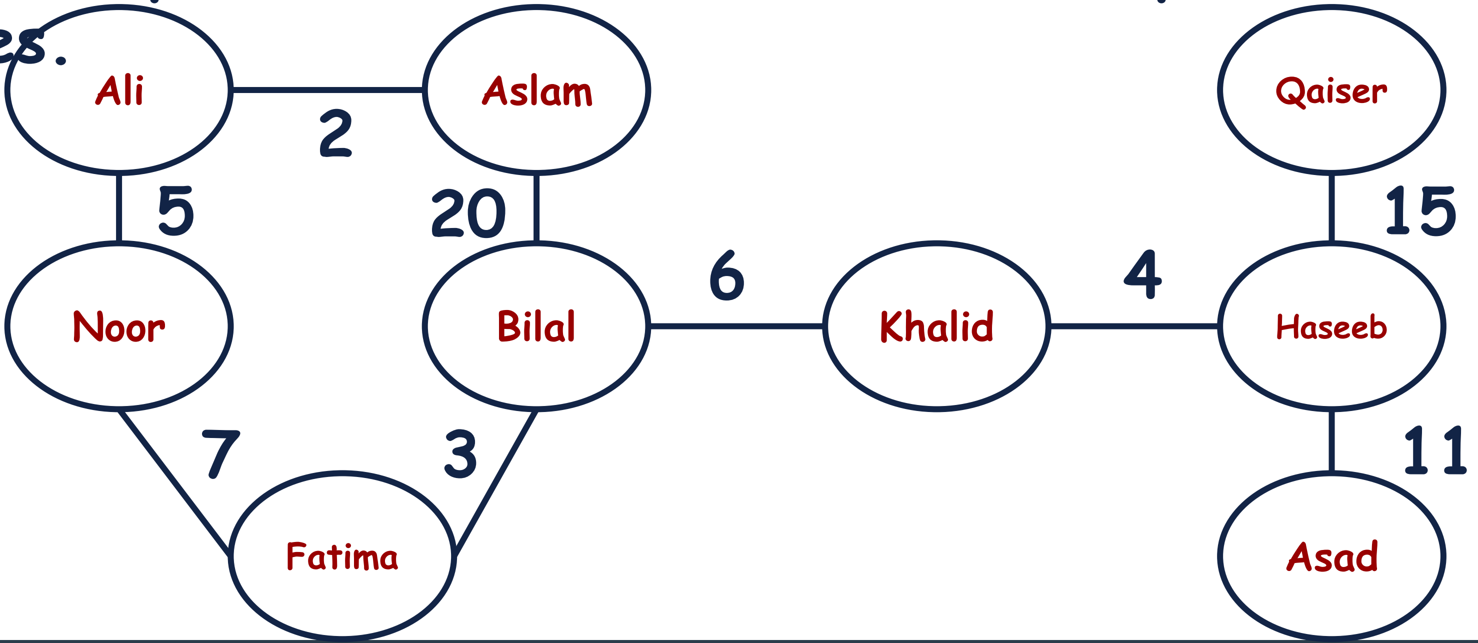
Graphs

Facebook is maintaining the information of years of friendship between friends. It can be represented on edges.



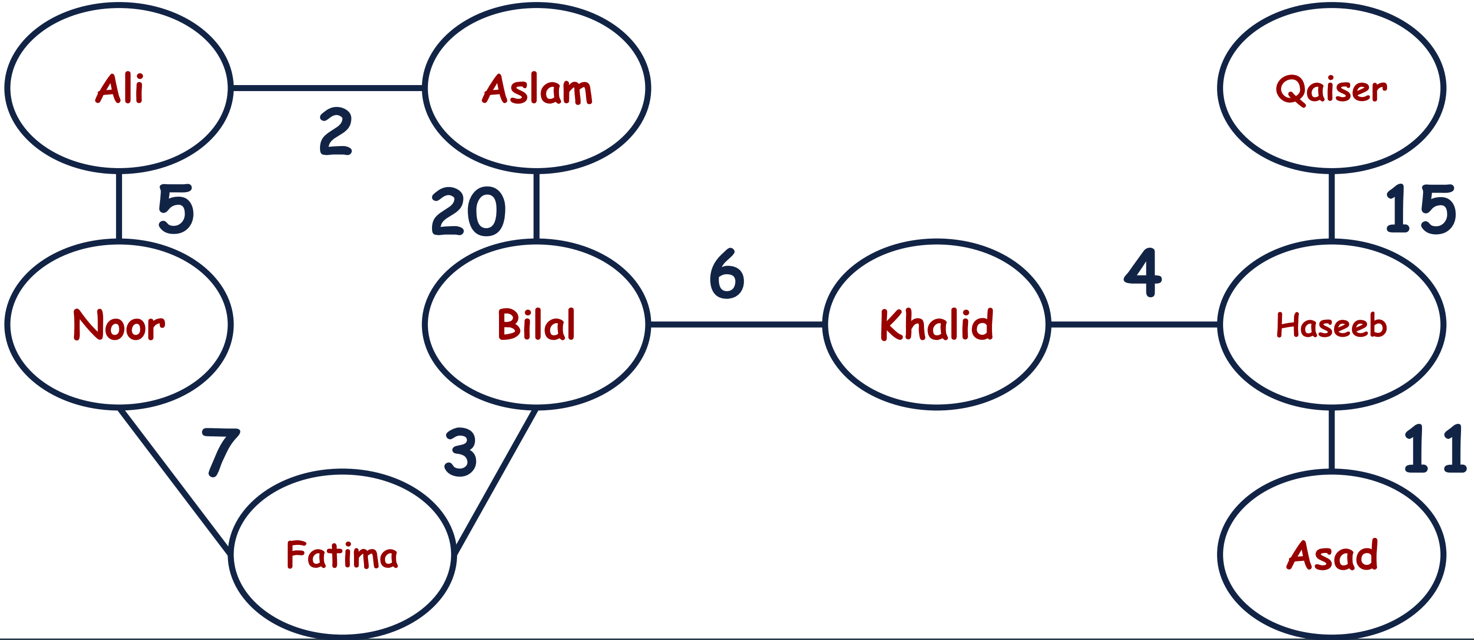
Graphs

Facebook is maintaining the information of years of friendship between friends. It can be represented on edges.



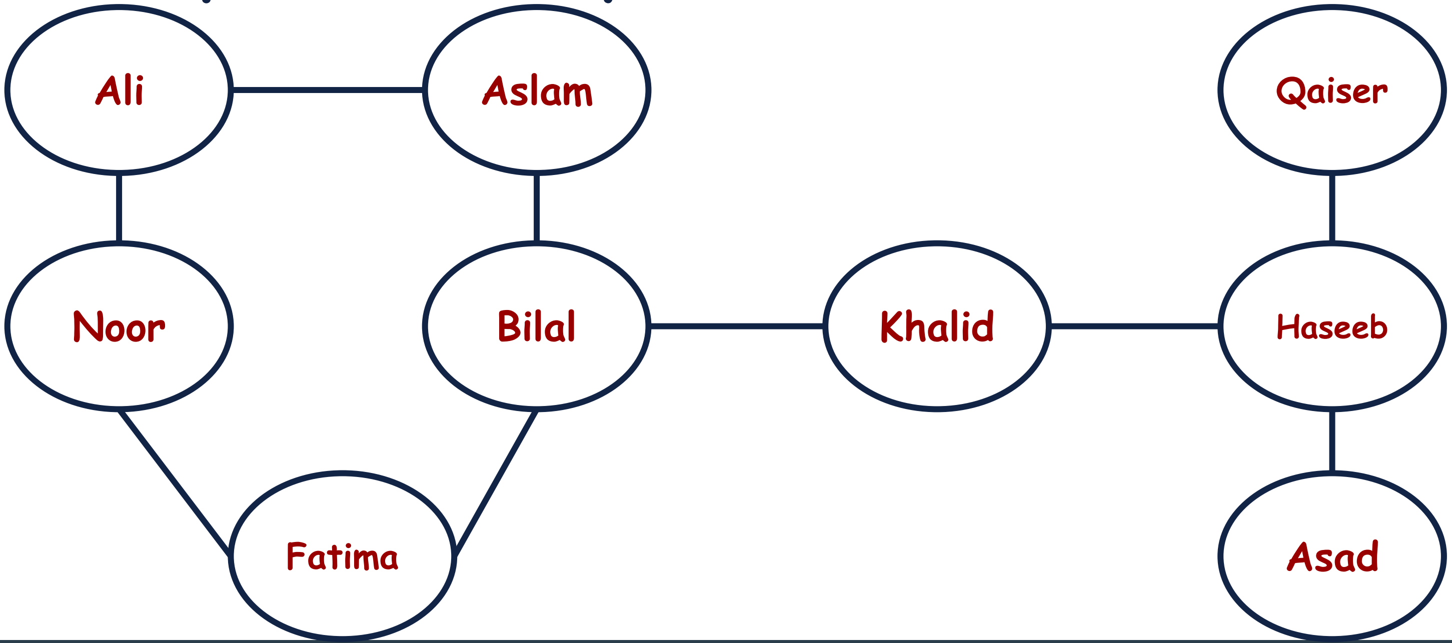
Graphs: Weighted Graphs

Such graphs are called as Weighted Graphs.

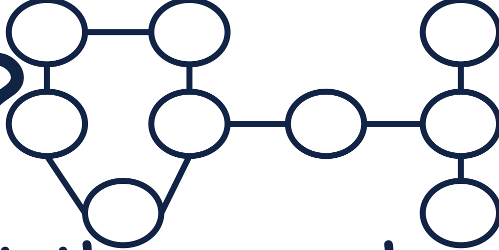


Graphs: How to Implement?

Now the main question is how to represent these graphs in the computer Memory?



Graphs: How to Implement?

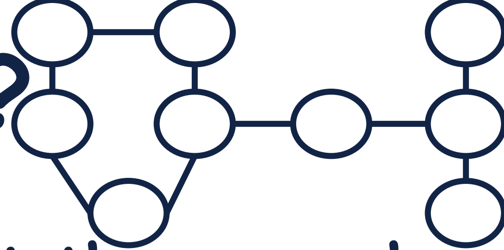


Now the main question is how to represent these graphs in the computer Memory?

A Graph is a collection of set of Vertices and set of Edges.

$$G = (V, E)$$

Graphs: How to Implement?



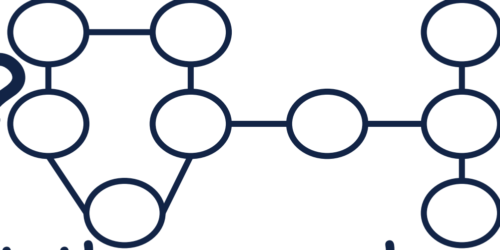
Now the main question is how to represent these graphs in the computer Memory?

A Graph is a collection of set of Vertices and set of Edges.

$$G = (V, E)$$

Vertices: {Ali, Aslam, Noor, Bilal, Fatima, Khalid, Haseeb, Qaiser, Asad}

Graphs: How to Implement?



Now the main question is how to represent these graphs in the computer Memory?

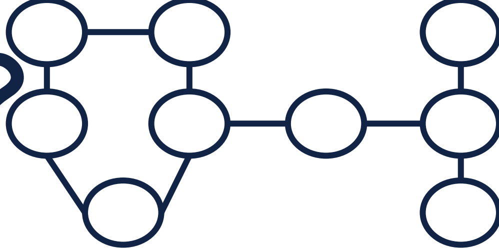
A Graph is a collection of set of Vertices and set of Edges.

$$G = (V, E)$$

Vertices: {Ali, Aslam, Noor, Bilal, Fatima, Khalid, Haseeb, Qaiser, Asad}

Edges: {Ali-Aslam, Aslam-Bilal, Ali-Noor, Noor-Fatima, Bilal-Fatima, Bilal-Khalid, Khalid-Haseeb, Haseeb-Qaiser, Haseeb-Asad}

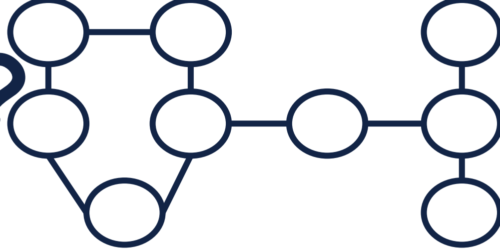
Graphs: How to Implement?



Vertices: {Ali, Aslam, Noor, Bilal, Fatima, Khalid, Haseeb, Qaiser, Asad}
Edges: {Ali-Aslam, Aslam-Bilal, Ali-Noor, Noor-Fatima, Bilal-Fatima, Bilal-Khalid, Khalid-Haseeb, Haseeb-Qaiser, Haseeb-Asad}

Now, we can make 2 vectors.
One for vertices and one for edges.

Graphs: How to Implement?



Vertices: {Ali, Aslam, Noor, Bilal, Fatima, Khalid, Haseeb, Qaiser, Asad}

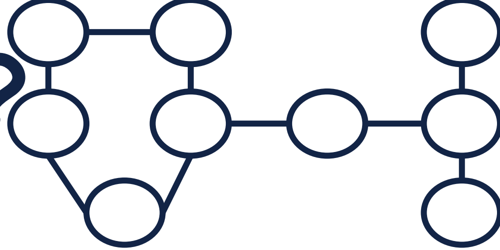
Edges: {Ali-Aslam, Aslam-Bilal, Ali-Noor, Noor-Fatima, Bilal-Fatima, Bilal-Khalid, Khalid-Haseeb, Haseeb-Qaiser, Haseeb-Asad}

Now, we can make 2 vectors.
One for vertices and one for edges.

```
vector<string> vertices;
vector<Edge> edges;
```

```
struct Edge
{
    string startVertex;
    string endVertex;
};
```

Graphs: How to Implement?



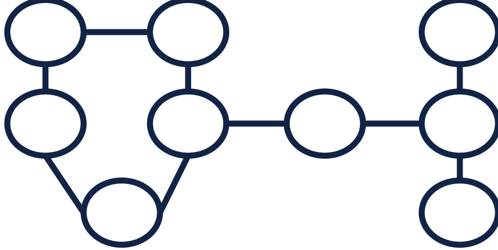
vertices

0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

Graphs: Edge List



vertices

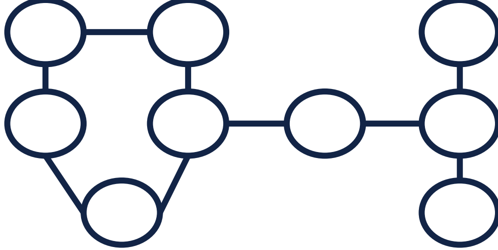
0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

This representation of Graph is called Implementation with Edge List.

Graphs: Edge List



vertices

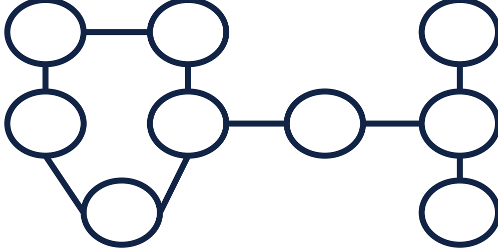
0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

What is the space complexity in this case?

Graphs: Edge List



vertices

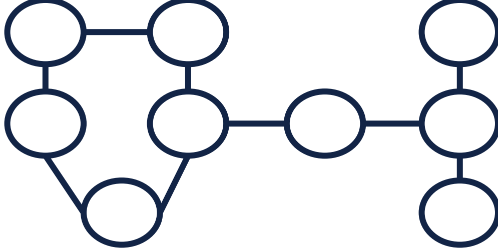
0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

Vertices vector contains all the vertices of the graph therefore its space complexity is $O(|V|)$

Graphs: Edge List



vertices

0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

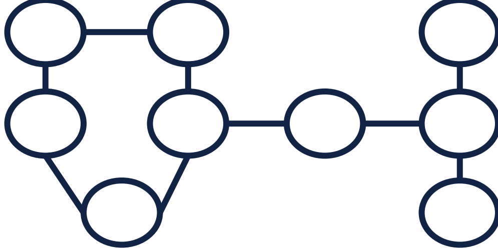
edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

Vertices vector contains all the vertices of the graph therefore its space complexity is O(|V|)

|V| represents that all the elements present in set V

Graphs: Edge List



vertices

0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

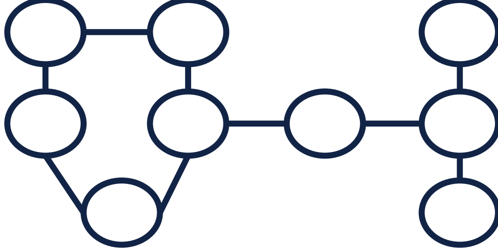
edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

edges vector contains all the edges of the graph therefore its space complexity is $O(|E|)$

$|E|$ represents that all the elements present in set E

Graphs: Edge List



vertices

0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

Overall space complexity is $O(|V+E|)$

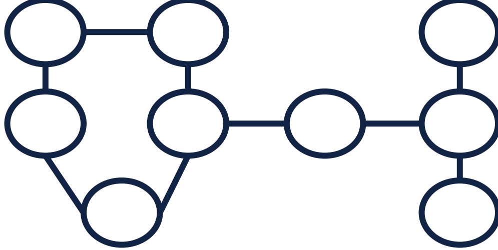
Graphs: Edge List

vertices

0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

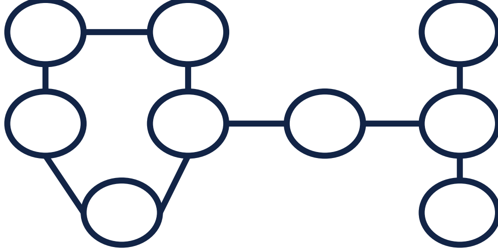
edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad



Now what will be the time complexity to find the friends of Ali?

Graphs: Edge List



vertices

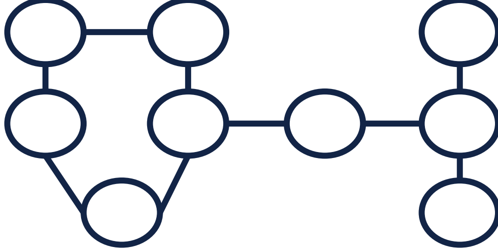
0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

We will have to search all the edges list therefore the time complexity will be $O(|E|)$

Graphs: Edge List



vertices

0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

This implementation is not efficient in case of time complexity.

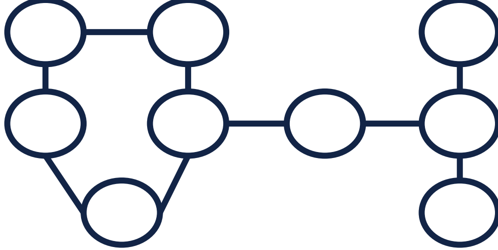
Graphs: Edge List

vertices

0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

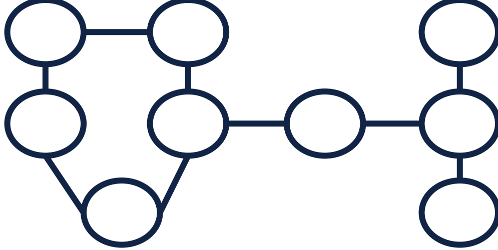
edges

0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad



Because the number of edges can be the equal to the square of vertices present in the worst case.

Graphs: Edge List



vertices

0	Ali
1	Aslam
2	Noor
3	Bilal
4	Fatima
5	Khalid
6	Haseeb
7	Qaiser
8	Asad

edges

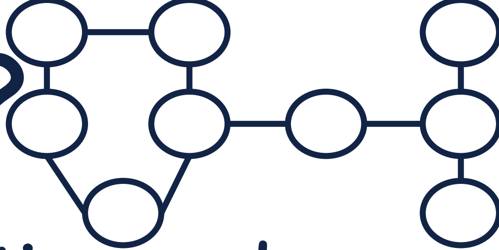
0	Ali	Aslam
1	Aslam	Bilal
2	Ali	Noor
3	Noor	Fatima
4	Bilal	Fatima
5	Bilal	Khalid
6	Khalid	Haseeb
7	Haseeb	Qaiser
8	Haseeb	Asad

Can we have better time complexity than this?

Graphs: How to Implement?

Now, instead of making 2 vectors for vertices and edges, can we use a vertex vector and a 2D array for edges?

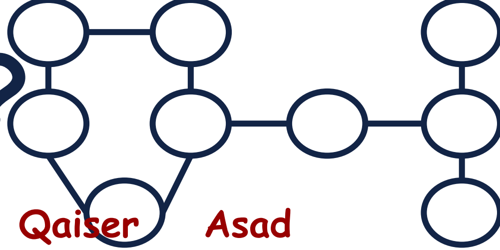
Graphs: How to Implement?



Now, instead of making 2 vectors for vertices and edges, can we use a vertex vector and a 2D array for edges?

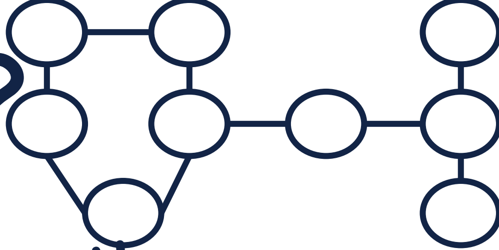
Rows and columns will represent the Vertices.

Graphs: How to Implement?



	Ali	Aslam	Noor	Bilal	Fatima	Khalid	Haseeb	Qaiser	Asad
Ali									
Aslam									
Noor									
Bilal									
Fatima									
Khalid									
Haseeb									
Qaiser									
Asad									

Graphs: How to Implement?



Now, fill this matrix with zeros. Only make the ones where the edge exist between the corresponding vertices.

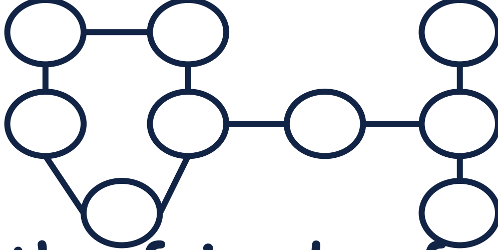
Graphs: How to Implement?

	Ali	Aslam	Noor	Bilal	Fatima	Khalid	Haseeb	Qaiser	Asad
Ali	0	1	1	0	0	0	0	0	0
Aslam	1	0	0	1	0	0	0	0	0
Noor	1	0	0	0	1	0	0	0	0
Bilal	0	1	0	0	1	1	0	0	0
Fatima	0	0	1	1	0	0	0	0	0
Khalid	0	0	0	1	0	0	1	0	0
Haseeb	0	0	0	0	0	1	0	1	1
Qaiser	0	0	0	0	0	0	1	0	0
Asad	0	0	0	0	0	0	1	0	0

Graphs: Adjacency Matrix

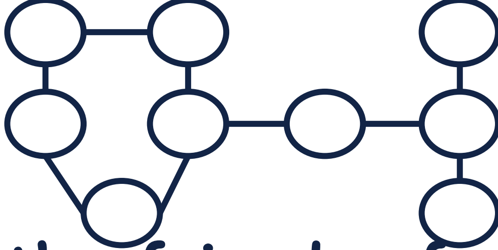
This representation of the Graph is called as the Adjacency Matrix representation of Graph.

Graphs: Adjacency Matrix



Now, what is the time complexity to find the friends of Ali?

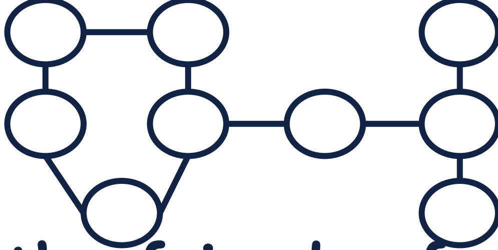
Graphs: Adjacency Matrix



Now, what is the time complexity to find the friends of Ali?

We have to search the vertices list to find the index of the vertex and then go to the specific index in the 2D array and then we just have to traverse all the columns and print the names where the value is 1.

Graphs: Adjacency Matrix



Now, what is the time complexity to find the friends of Ali?

We have to search the vertices list to find the index of the vertex and then go to the specific index in the 2D array and then we just have to traverse all the columns and print the names where the value is 1.

Time Complexity will be $O(|V| + |V|)$.

Graphs: Adjacency Matrix

Now, what is the time complexity to find the friends of Ali?

We have to search the vertices list to find the index of the vertex and then go to the specific index in the 2D array and then we just have to traverse all the columns and print the names where the value is 1.

Time Complexity will be $O(|V|)$.

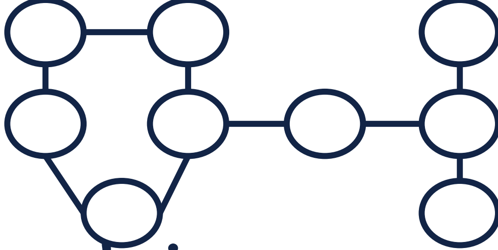
Graphs: Adjacency Matrix

With the Adjacency Matrix representation, the time complexity has been improved.
But what about the space complexity?

Graphs: Adjacency Matrix

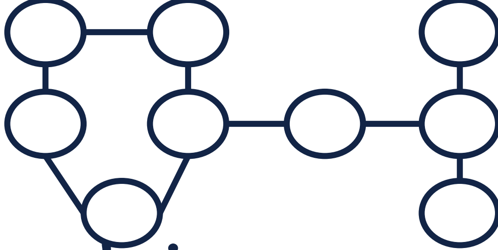
Now, instead of the $O(|V| + |E|)$, we are defining a 2D matrix where rows and columns are equal to the number of vertices. Therefore, the space complexity to store edges has become $O(|V^2|)$.

Graphs: Adjacency Matrix



Adjacency Matrix representation of the graph is feasible if the graph is dense, i.e, there is an edge from a specific vertex to almost all the vertices of the graph.

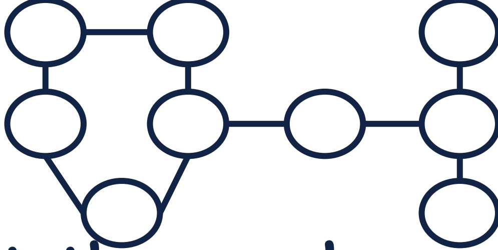
Graphs: Adjacency Matrix



Adjacency Matrix representation of the graph is feasible if the graph is dense, i.e, there is an edge from a specific vertex to almost all the vertices of the graph.

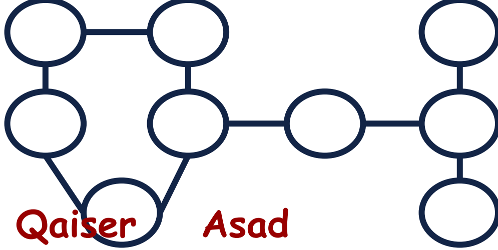
But most of the real life graphs are sparse, therefore, adjacency matrix representation is not that efficient.

Graphs: Adjacency Matrix



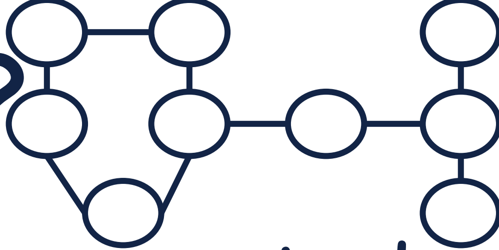
Now, the question is how can we represent the graph such that the time and space complexity are efficient. I.e, the space complexity is $O(|V| + |E|)$ like the edge list representation and time complexity to perform the operations is like the adjacency matrix or maybe better than adjacency matrix representation.

Graphs: Adjacency Matrix



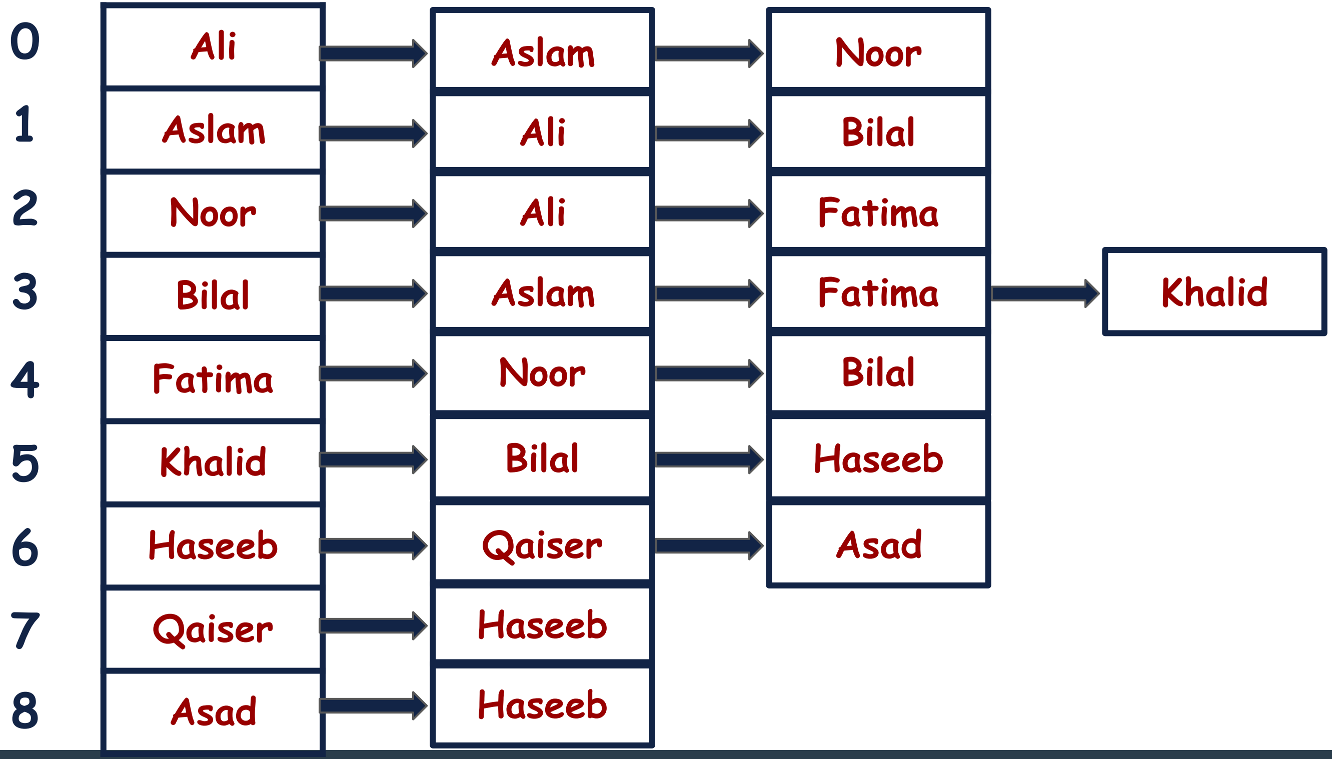
	Ali	Aslam	Noor	Bilal	Fatima	Khalid	Haseeb	Qaiser	Asad
Ali	0	1	1	0	0	0	0	0	0
Aslam	1	0	0	1	0	0	0	0	0
Noor	1	0	0	0	1	0	0	0	0
Bilal	0	1	0	0	1	1	0	0	0
Fatima	0	0	1	1	0	0	0	0	0
Khalid	0	0	0	1	0	0	1	0	0
Haseeb	0	0	0	0	0	1	0	1	1
Qaiser	0	0	0	0	0	0	1	0	0
Asad	0	0	0	0	0	0	1	0	0

Graphs: How to Implement?



Can we do something like from every vertex we extend a linked list that will represent the edges to other vertices?

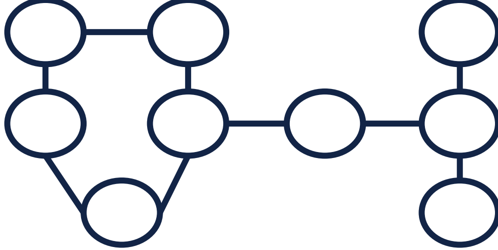
Graphs: How to Implement?



Graphs: Adjacency List

This representation of graph is called as implementation with Adjacency List.

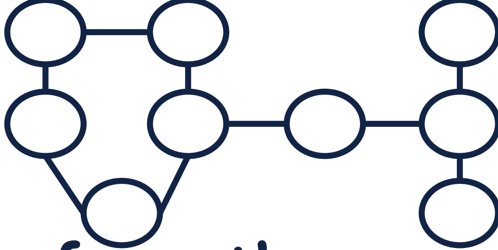
Graphs: Adjacency List



Now, we have following 2 vectors.
Vector containing all the vertices. Another vector of vector for storing the edges.

```
vector<string> vertices;
vector<vector<string>> edjList;
```

Graphs: Adjacency List

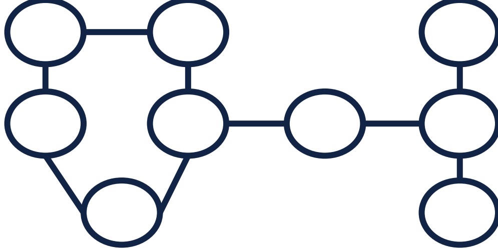


In this we have to first search the vertex from the vertices list and then go to the specific index of adjacency list.

Graphs: Adjacency List

Is there any better way to implement the adjacency list so that we don't have to search for the indexes?

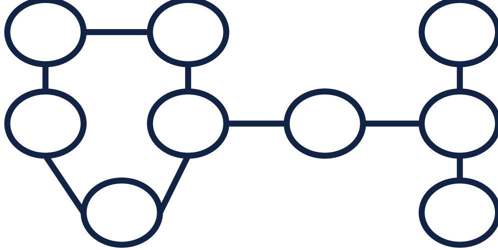
Graphs: Adjacency List



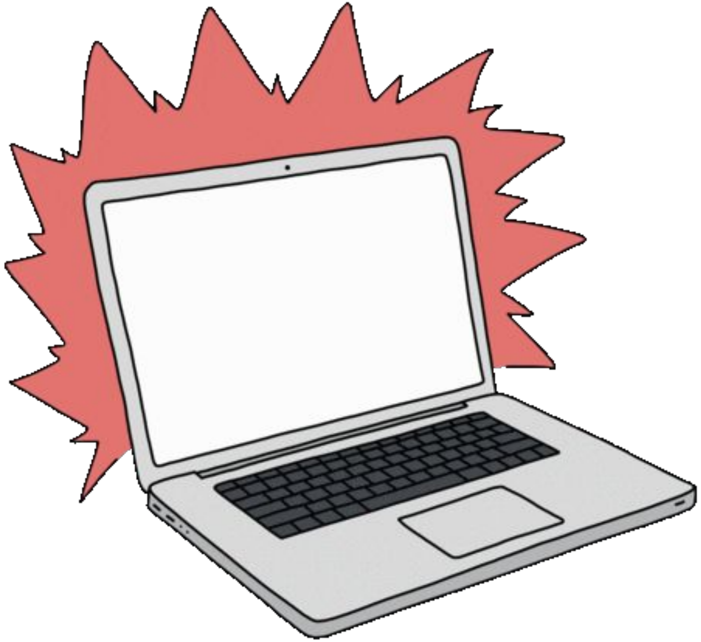
We can do that with the help of hashMaps.
Keys will be the vertices and values will be the edges connected to that corresponding vertices.

```
unordered_map<string, vector<string>> adjList;
```

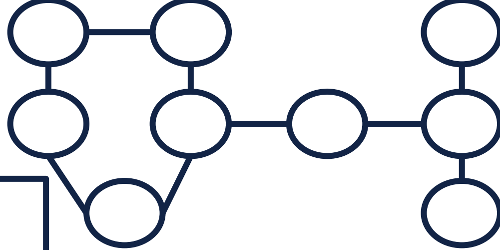
Graphs: Adjacency List



Let's implement the friends graph now.



Graphs: Adjacency List



```
class Graph
{
    unordered_map<string, vector<string>> adjList;

public:
    void addEdge(string s, string d)
    {
        adjList[s].push_back(d);
        adjList[d].push_back(s);
    }

    void print()
    {
        for (auto lst : adjList)
        {
            cout << lst.first << " ";
        }
    }
};
```

Learning Objective

Students should be able to **store** the graphs into computer memory efficiently to solve real life problems.

Self Assessment

https://leetcode.com/problems/find-the-town-judge/
https://leetcode.com/problems/find-center-of-star-graph/

Self Assessment

Project Ideas on Graphs:

- https://prinsli.com/application-of-graph-theory-in-real-life/
- https://www.xomnia.com/post/graph-theory-and-its-uses-with-examples-of-real-life-problems/