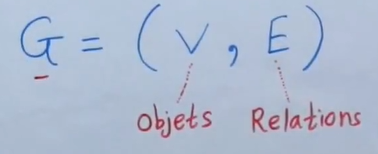
GRAPH



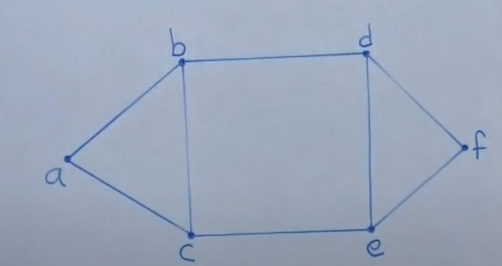
**Vertex** – is a node. Vertices – is plural

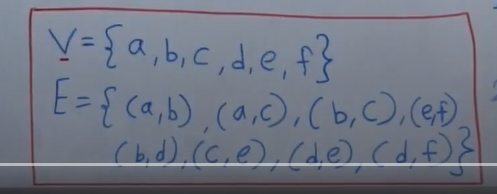
**Edge** – is an element of graph

V- set of vertices (objects)

E- set of edges

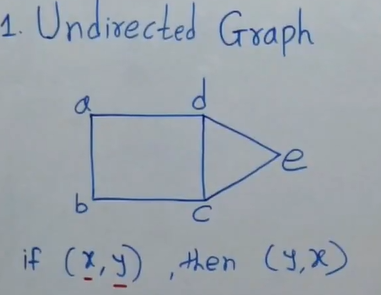
**Order of graph** – is number of vertices (objects, elements)





If consider an example of social network

(a,b), (a,c) -are friends, but (a,f) are not friends

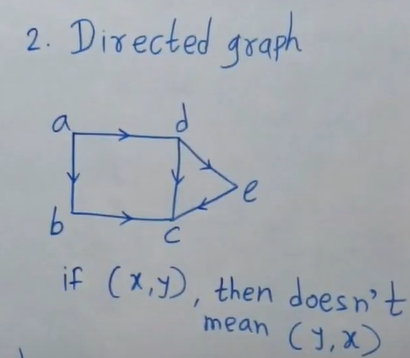


Undirected graph: if there is an edge between (a,b) => there is also an edge between (b,a). For example,

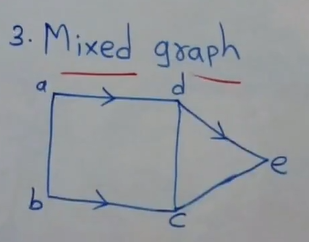
If [a] is a friend of [b], then [b] is also a friend of [a]

It is called **sibling relation**.

* (!) undirected graph always has a cycle, because you can simple go back and forth between any two neighbors
* Difference between trees and graphs that graphs do not have cycle



For example, it like a follower relation. If [a] follows [b], it does not mean [b] is also a follower of [a]

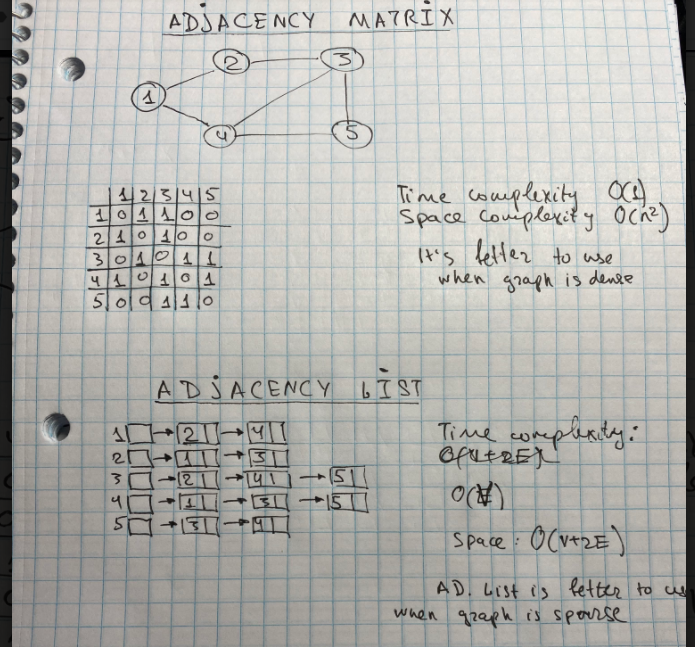


Example, of job scheduler

# **GRAPH PRESENTATION**

Graph can be present as:

* Adjacency matrix
* Adjacency list



DFS vs BFS

* DFS and BFS is a way of searching. It’s not only for graph. It is general term. It has a goal to search relations. It ‘s not only about graph, it a general term. It can be used to find a distance in a string.

**DFS** it is goes first deep and then goes back

Stack (LIFO) is used

**BFS** – goes wide

Queue (FIFO) is used

Steps:

Pull a node

Process if not seen and mark as seen when it is processed

Add childers

# **TYPES OF EDGES DFS TRAVERSAL**

There are four types of edges in DFS traversal

Tree edge – member of DFS traversal

Forward edge – G(x,y) where [y] appears after [x] and there is a path between [x] and [y]

Back edge - G(y,x) where [y] appears before [x] and there is a path between [x] and [y]

Cross edge - G(y,x) any edge from [x] to [y], where there is no path from [y] to [x]

