Algorithms: Graph Representation (Undirected and Unweighted)

What is Graph?

- Graph (denoted by **G**) consists of a set of vertices (denoted by **V**) and a set of edges (denoted by **E**). These set of edges describes how the vertices are connected to each other.
  - G = (V, E)
  - Number of nodes/vertices are often denoted by **n.**
  - Number of edges/connections are often denoted by **m.**

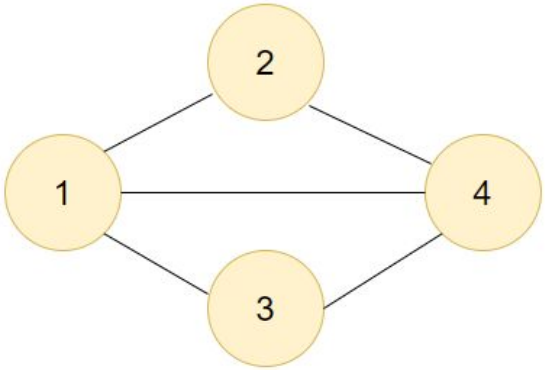
- Examples of **vertices**: cities, houses, objects, computers etc.
- Examples of **edges**: roads, cables, pipes, etc.
- Examples of **graphs**: Social networks where peoples are nodes and connections between them are edges.

- Edges can be of different types
  - Directed edge
  - Undirected edge
  - Weighted edge
  - Unweighted edge

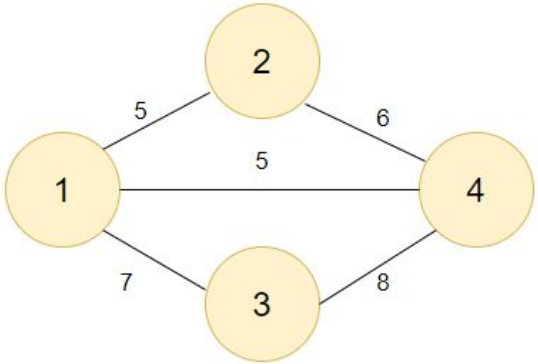
Why study Graph?

- Lots of algorithmic problems are related to graphs.
- If some problems are converted to a graph problem, they become easy to solve.
- Some Popular graph problems are:
  - Shortest Path Problems
  - Traversing problem
  - Maximum Network flow problem.
  - Travelling salesman problem

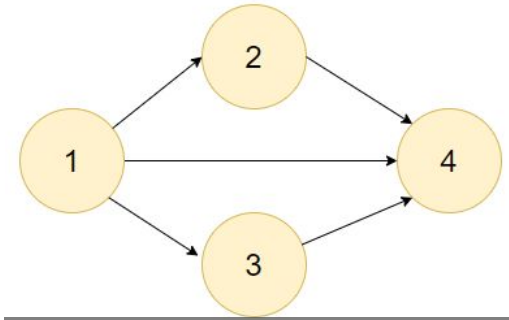
TYPES OF GRAPHS



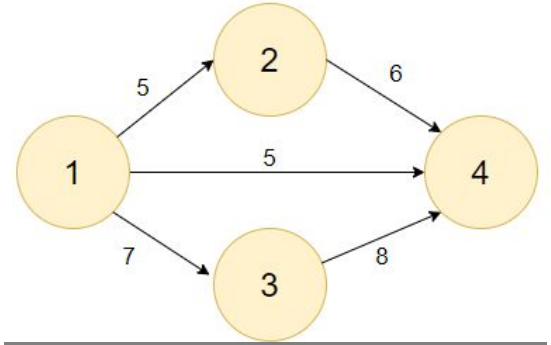
Unweighted and Undirected



Weighted but Undirected



Unweighted but directed

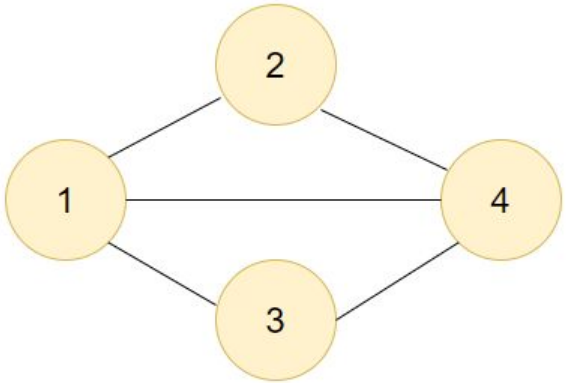


Weighted and Directed

Graph Representation

- There are different ways to represent a graph. We will see
  - Adjacency Matrix
  - Adjacency List
  - Edge List

Adjacency Matrix (undirected)

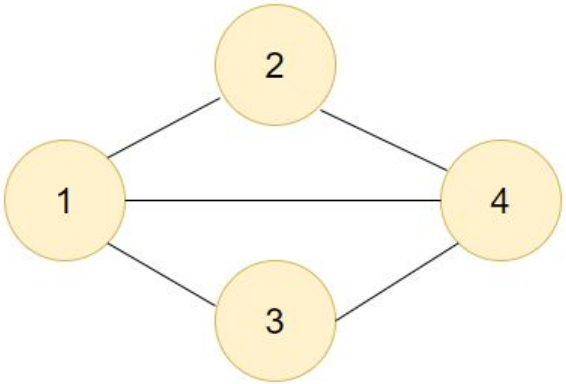


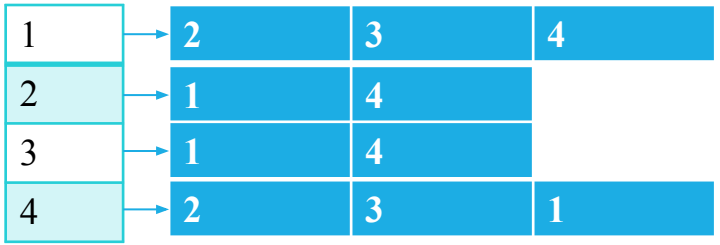
	1	2	3	4
1	0	**1**	**1**	**1**
2	**1**	0	0	**1**
3	**1**	0	0	**1**
4	**1**	0	**1**	0

Data Structure to Use: (n * n) Matrix.

Example: int adj [n][n];

Adjacency List (undirected)





Data Structure to Use:
2D vector or array of linked list.

Example:

```
vector<int> adj [n];
vector< vector<int> > adj;
```

Suggested Reading

- Algorithm Design by Jon Kleinberg, Eva Tardos
  - Chapter 3
    - Section: 3.1