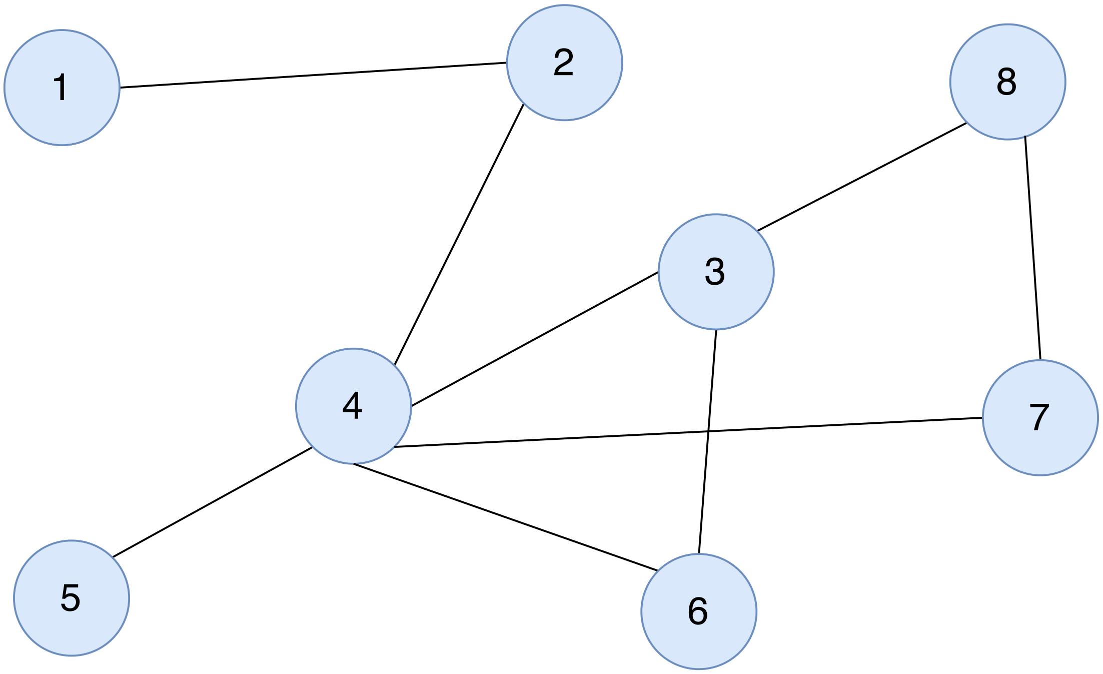
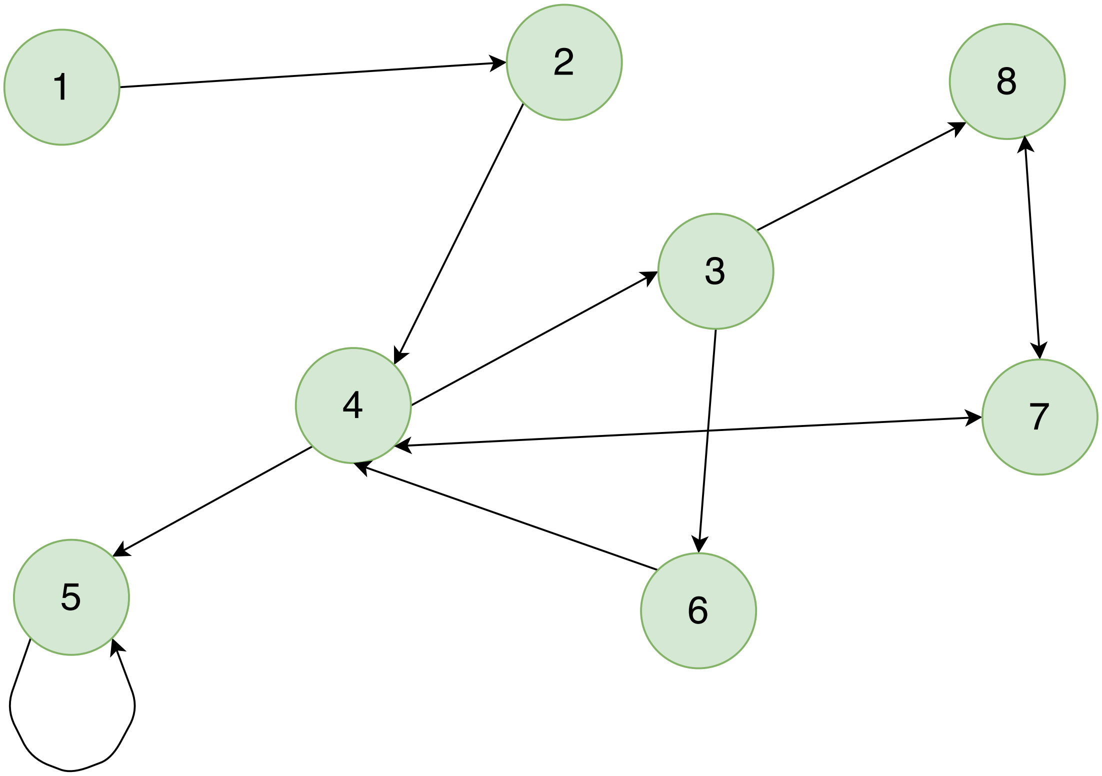
Graph Theory

Undirected Graph



Example:
Friends on Facebook

Directed Graph

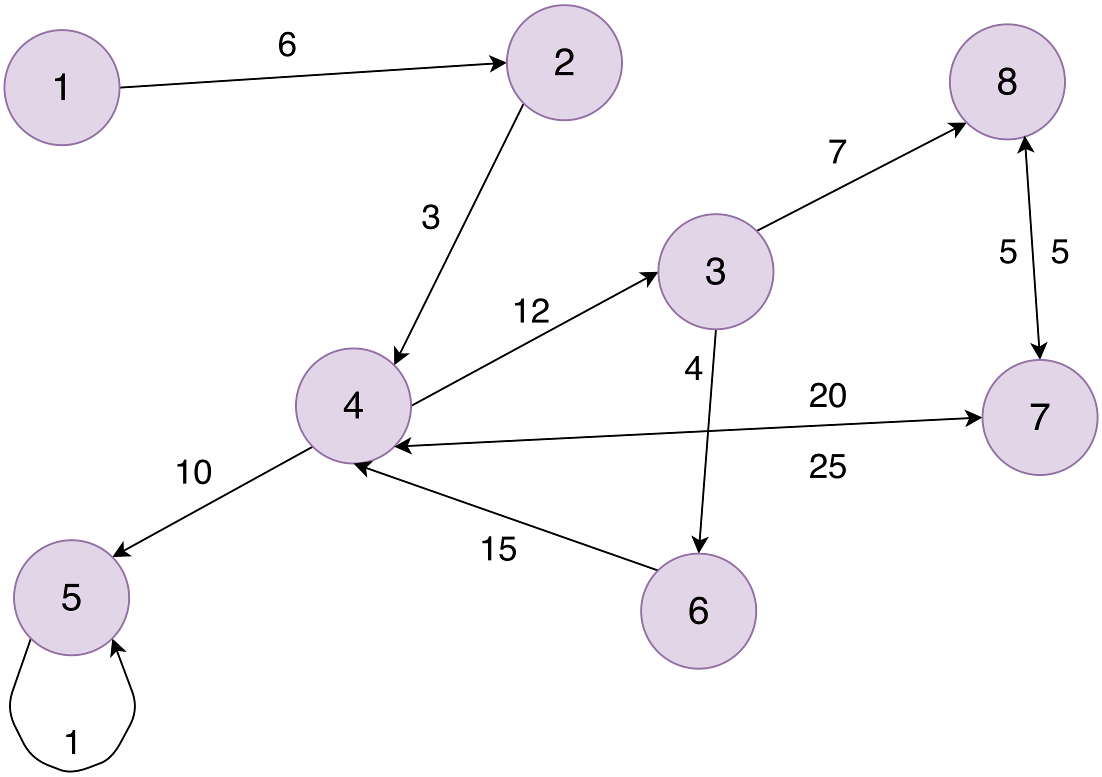


Example:
Followers on instagram
People giving each other gifts
Roads with 1 way streets

Weighted Graph

(Can Be **Undirected** or **Directed**)

Example:
Roads with Traffic

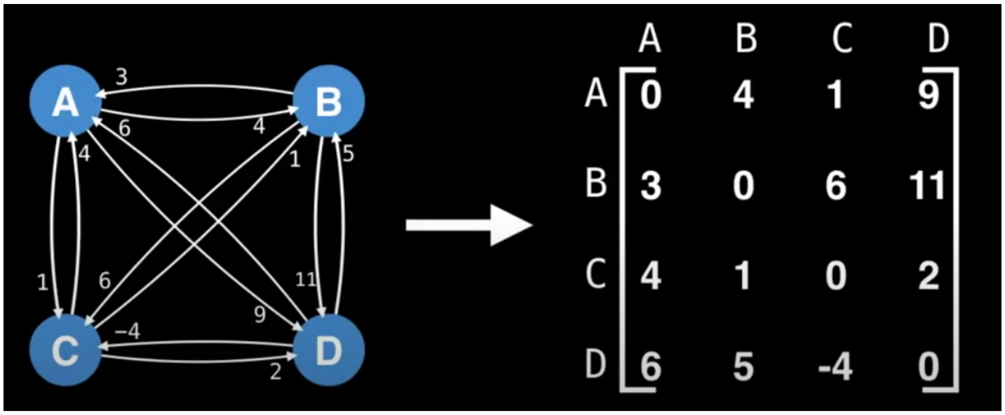


Other Types of Graphs

Tree
Rooted Tree
In-Tree (anti-arborescence)
Out-Tree (arborescence)
Directed Acyclic Graph (DAGs)
Bipartite Graph
Complete Graph

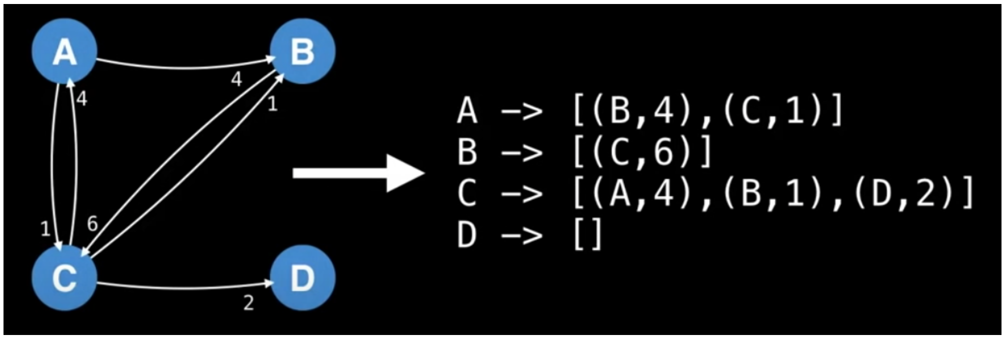
How To Represent a Graph

Adjacent Matrix



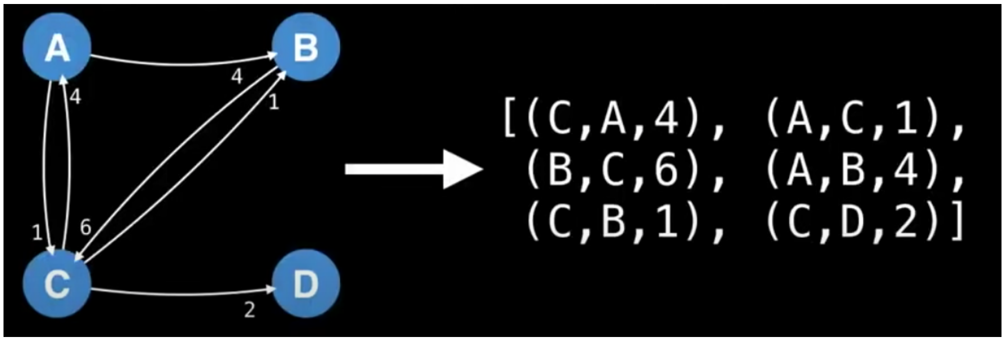
$$\begin{array}{c|cccc|} & A & B & C & D \\ A & 0 & 4 & 1 & 9 \\ B & 3 & 0 & 6 & 11 \\ C & 4 & 1 & 0 & 2 \\ D & 6 & 5 & -4 & 0 \end{array}$$

Adjacent List



```
A -> [(B,4),(C,1)]
B -> [(C,6)]
C -> [(A,4),(B,1),(D,2)]
D -> []
```

Edge List



```
[(C,A,4), (A,C,1),
 (B,C,6), (A,B,4),
 (C,B,1), (C,D,2)]
```

There are many different ways to represent a graph. The representation of a graph will always depend on its size, its type, whether it is directed, whether it is weighted, and more. For an undirected graph, I use a dictionary to represent the nodes.

Algorithms Used For Graphs

Depth-first search	(Connectivity)
Breadth-first search	(Connectivity)
Dijkstra's	(Shortest path)
A*	(Shortest path)
Bellman-Ford	(Shortest path) Or (Detect Negative Cycles)
Floyd-Warshall	(Shortest path) Or (Detect Negative Cycles)
Tarjan's	(Strongly Connected Components)
Kosaraju's	(Strongly Connected Components)
Held-Karp	(approximation) Used to solve Traveling Salesman Problems (TSPs)
Branch and Bound	(approximation) Used to solve Traveling Salesman Problems (TSPs)
Kruskal's	(Finding Minimum Spanning Trees (MSTs))
Prim's	(Finding Minimum Spanning Trees (MSTs))
Boruvka's	(Finding Minimum Spanning Trees (MSTs))
Ford-Fulkerson	(Netowork Flow: Max Flow)
Edmonds-Karp	(Netowork Flow: Max Flow)
Dinic's	(Netowork Flow: Max Flow)