Part II

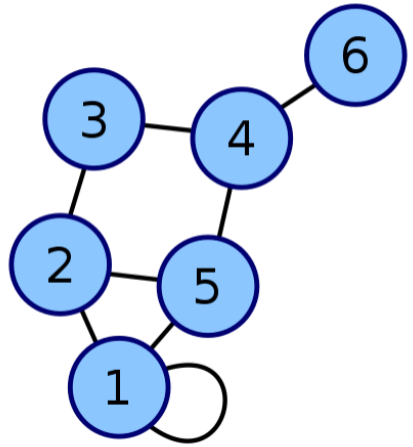
Reminder on Graph Theory

Graph Theory (GT)

- GT (mathematical subdomain) has significantly contributed to the evolution of SNA.
- Viewing social networks as graphs forms the basis for using Graph Theory.
  - Individuals, entities, organizations, … -> nodes, relationships, interactions, … -> edges, the importance of an individual, … -> degree, centrality, …
  - A better visualization of the structural properties of a network.
- A wide array of metrics, algorithms, methods, … are available in GT to analyze and study graphs.

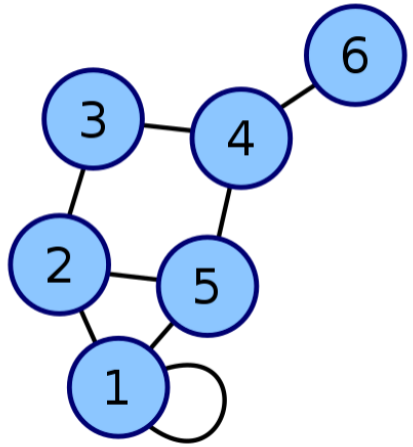
Graph Theory

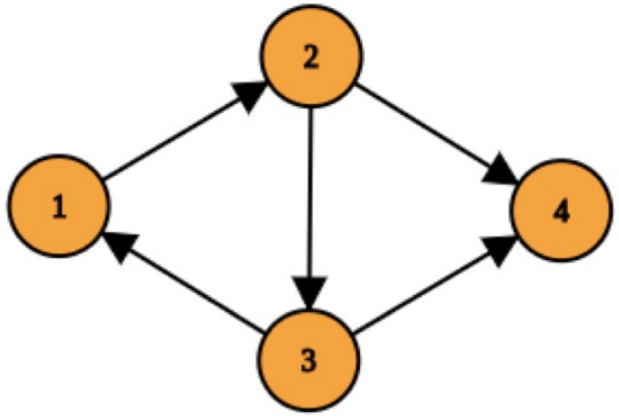
- **Graph**: A simple visualization (abstraction) of a set of connections between multiple entities.
- A graph G is defined by the pair G(V,E) (where V represents **nodes** and E represents **relations**).
  - node, vertex, actor
  - connection, edge, link, relation
- **Loop**: reflexive relation
- **Adjacent** nodes (neighbors): direct relation
- **Attributes**: nodes (role, type, ...) & connections (weight, sign, type of relation, ...)



Graph Theory

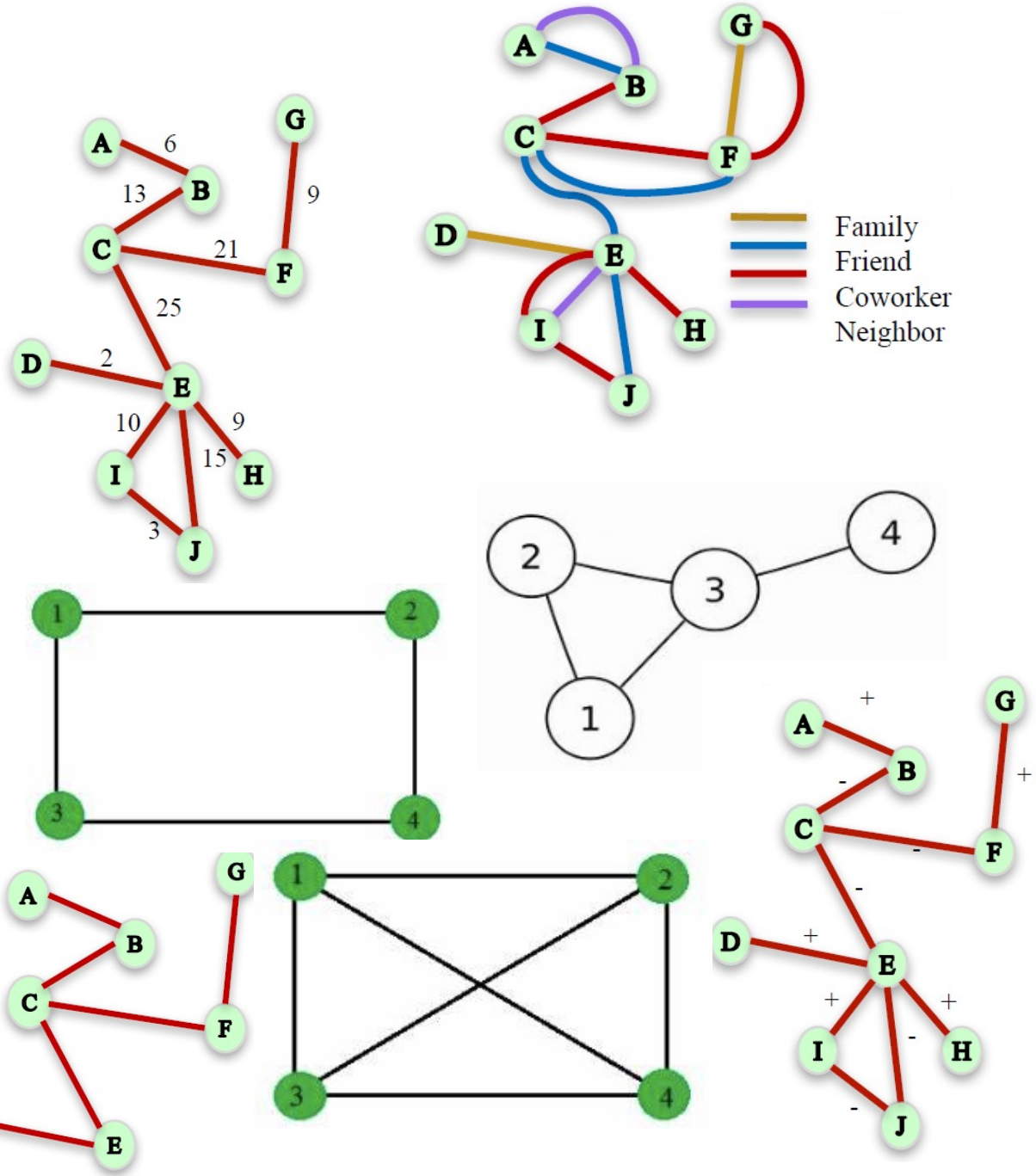
- **Undirected** graph: symmetric connections
- **Directed** graph: asymmetric connections
- **Order**: # of nodes
- **Size**: # of connections
- **Degree** (node): # of incident connections
- **Degree** (graph): maximum degree of all nodes
- **In-degree** (node): # of incoming connections
- **Out-degree** (node): # of outgoing connections





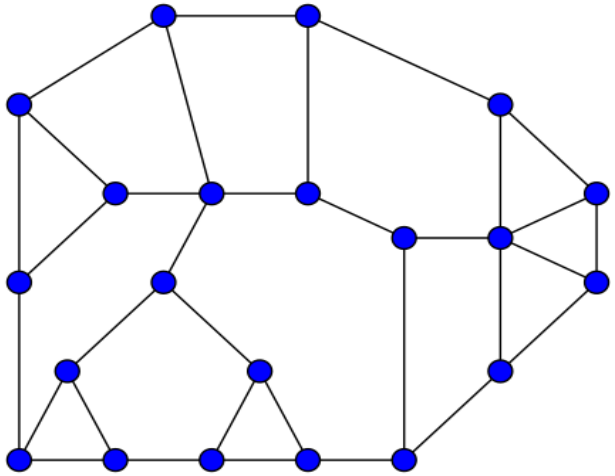
Graph Theory

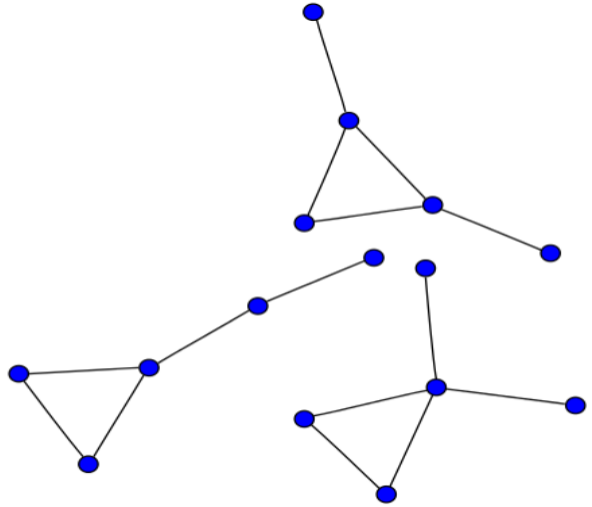
- **Simple** graph: no loops & no multiple connections
- **Multigraph**: multiple connections are possible
- **Weighted** graph: connections have labels or weights
- **Regular** graph: all nodes have the same degree
- **Complete** graph: every vertex is connected to all other nodes
- **Signed** graph: connections have opposite signs



Graph Theory

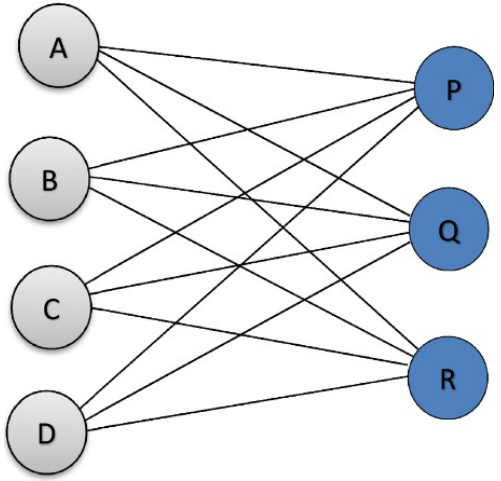
- **Connected** graph: every pair of nodes is connected by a chain/path
- **Disconnected** graph: set of distinct subgraphs (components)
- **Connected component**: a connected subgraph of a graph such that none of its nodes has a connection with the remaining nodes of the graph (isolated portions)

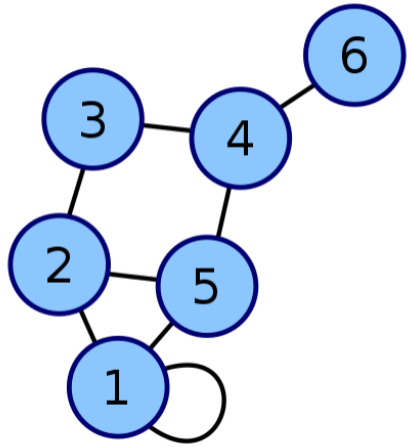




Graph Theory

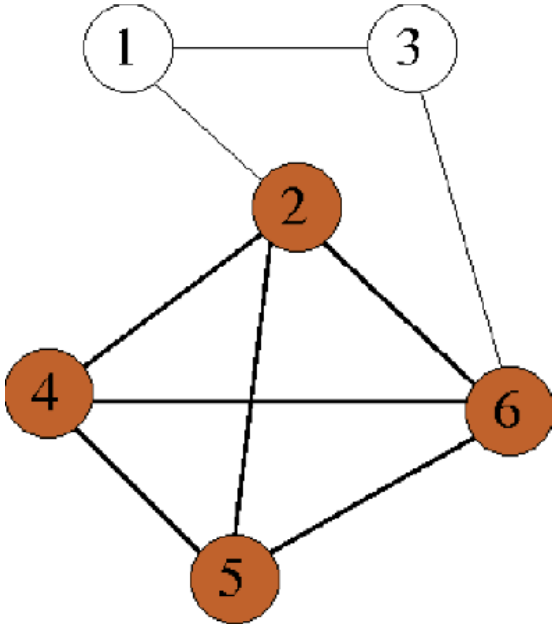
- **Bipartite** graph: two subsets of nodes with connections only between nodes from different sets
- **Partial** graph: all nodes with a subset of connections
- **Subgraph**: a subset of nodes and their associated connections
- **Complementary** graph: connections that do not exist in the original graph
- **Adjoint** graph: nodes become connections and vice versa





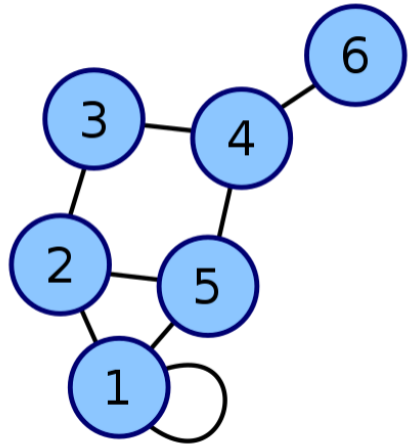
Graph Theory

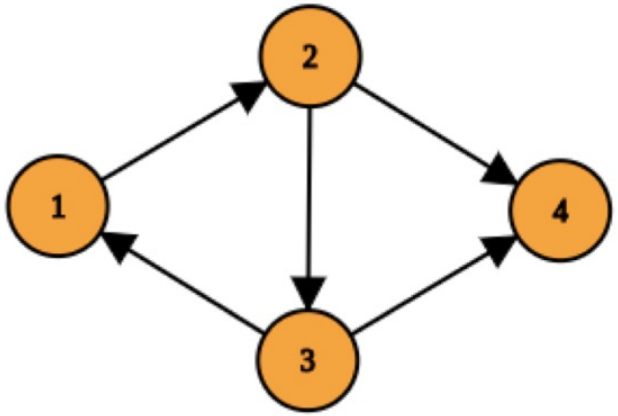
- **Clique**: a complete subgraph
- **Maximal clique**: the clique of maximum order
- **Independent set** (stable): a subset of nodes with no connections between them
- **Dyad**: a subgraph consisting of a connection between two nodes
- **Triad**: a subgraph consisting of connections among three nodes



Graph Theory

- **Chain/Walk**: a sequence of nodes connected by edges
- **Path**: a sequence of nodes connected by directed edges (arcs)
- **Cycle**: a simple closed chain
- **Circuit**: a simple closed path

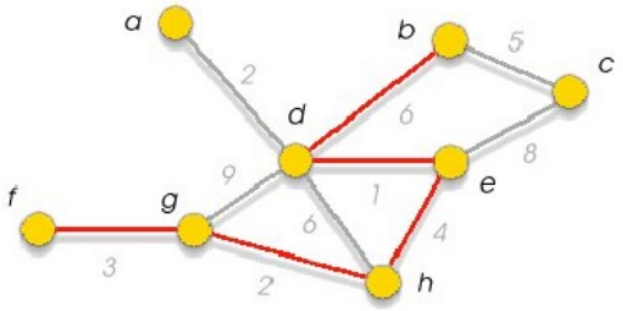




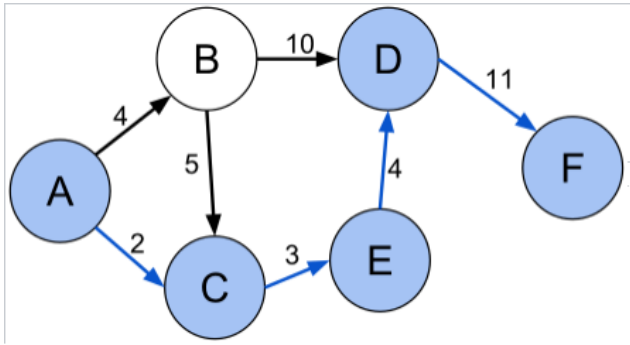
Graph theory

- **Shortest path**: a sequence of edges/arcs connecting two nodes with minimal distance (Dijkstra, Bellman-Ford algorithms, …)
- **Distance**: length of the shortest path between two nodes
- **Diameter**: the maximum value of distances between nodes

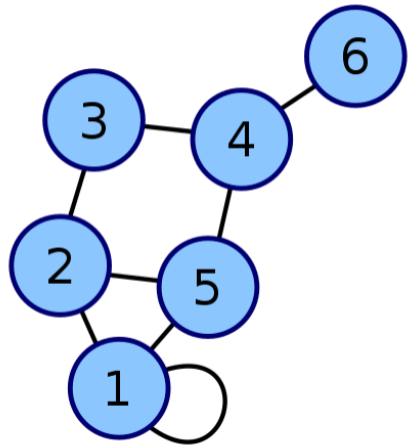
The SP between f & b is: {(f,g), (g,h), (h,e), (e,d), (d,b)}, SP(f,b)=? / Diameter = ?



The SP between A & F is: {(A,C), (C,E), (E,D), (D,F)}, SP(A,F)=? / Diameter = ?



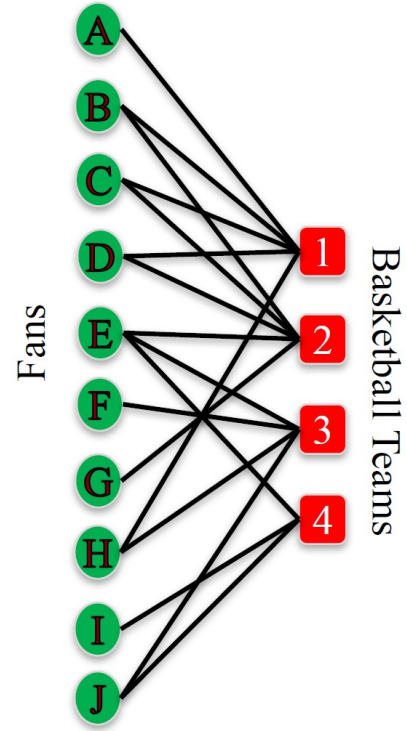
Diameter ?

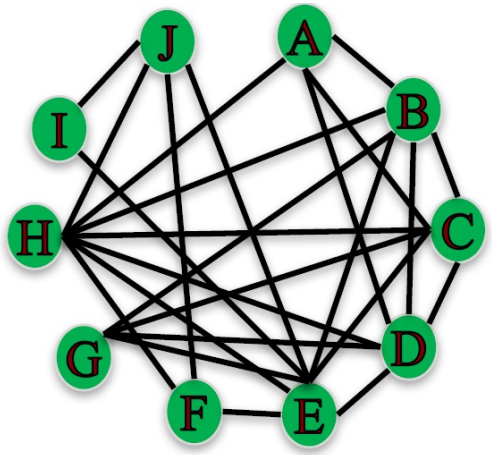


Distance(2,6) ?

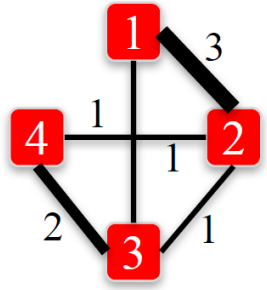
Graph theory

- **L-bipartite/R-bipartite** graph: formed by nodes from group L/R, where a pair of nodes is connected if they have a common neighbor in R/L in the bipartite graph.
- **Why**? To measure similarity and predict future connections (application in e-commerce).
- **Weights** can be assigned to links based on the number of common neighbors.





L-bipartite

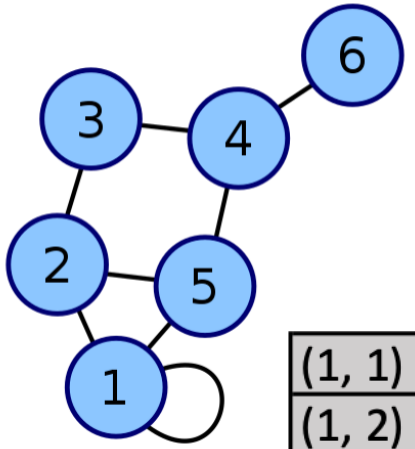


R-bipartite

Graph theory

- **Adjacency matrix**: a n×n matrix where element aij is the number of edges (weight) connecting node i to node j.
- **Adjacency list**: a list of n nodes, each with a list of its neighbors.
- **Edge list**: a list of all edges + list of nodes.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$



1	1	2	5
2	1	3	5
3	2	4	
4	3	5	6
5	1	2	4
6	4		

(1, 1)	1
(1, 2)	2
(1, 5)	3
(2, 3)	4
(2, 5)	5
(3, 4)	6
(4, 5)	
(4, 6)	