Ανάλυση Κοινωνικών Δικτύων (Social Network Analysis)

Graph Theory – Structure of Networks

Συμεών Παπαβασιλείου (papavass@mail.ntua.gr)
Βασίλειος Καρυώτης (vassilis@netmode.ntua.gr)

25 Οκτωβρίου, 2019

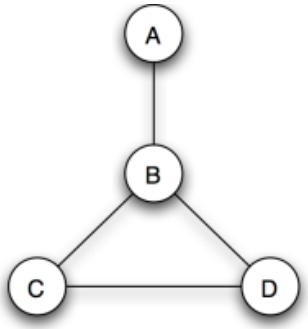
Complex interactions in highly connected systems

- Understanding highly connected networks and systems requires:
  - **Network structure**
  - Strategic behavior of actors
  - Feedback effects they produce across large populations

- **Structure:**
  - more or less densely interconnected
  - central core nodes containing most links
  - natural splits in multiple tightly-linked regions
  - Participants can be more central or more peripheral
  - **This requires a graph-theoretical framework to describe network formation and interconnectivity.**

Graph theory

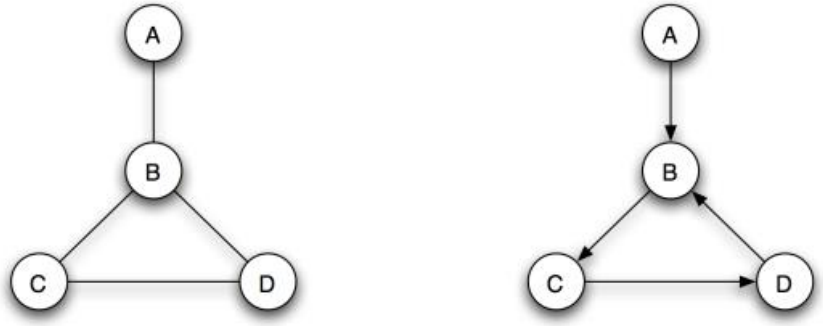
- A ***graph*** or a ***network*** is a way to specify relationships amongst a collection of items
- Mathematical model for representing network structures
- A graph consists of
  - Set of objects: *nodes*
  - Pairs of objects connected: *edges*



- Two nodes are neighbors if they are connected by an edge
- Symmetric links (an edge tha connects two nodes to each other)
- Assymetric relationships (node A points to node B but not vice versa)

Graph orientation

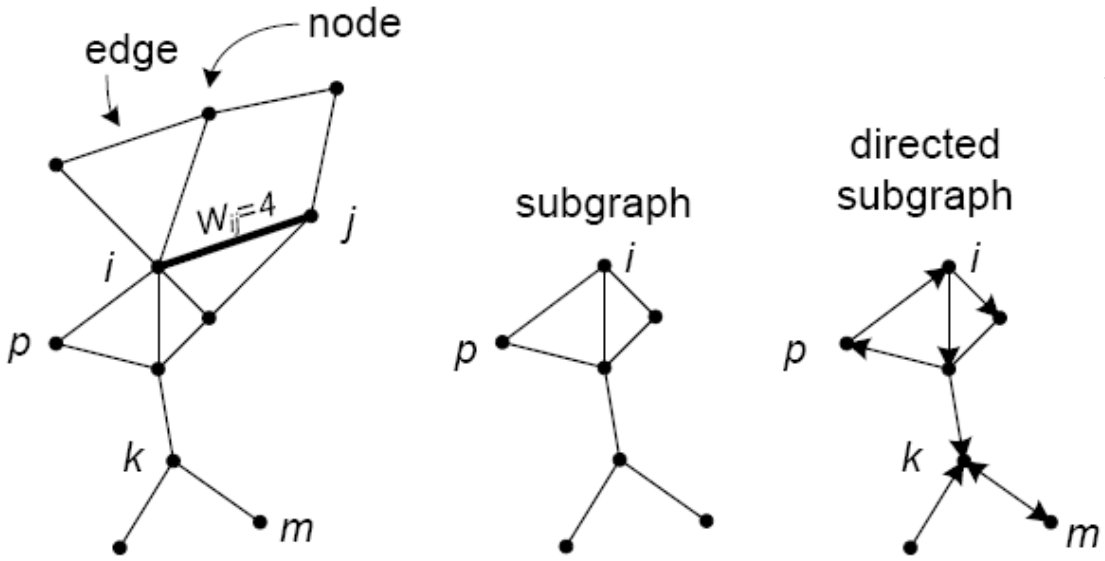
- Both graphs have 4 nodes (A,B,C,D) and 4 edges



- The left graph is *undirected*
  - Edges have no orientation (default assumption)
- The right graph is *directed*
  - Set of nodes with a set of dirceted edges. Each directed edge is a link from one node to another
  - Edges have an orientation, e.g. edge from B to C

Basic Definitions & Notation

- **Graph**: an ordered pair *G = (V;E): V* is the set of vertices/nodes with cardinality *n = |V|* and *E* is the set of edges with cardinality *|E|*.
  - Undirected: edges are unordered pairs of vertices $(i, j),\ \mathrm{i, j} \in V$
  - Directed : edges are ordered pairs of vertices, $(i, j) \neq (j, i)$
- **Examples**



i,j: adjacent vertices
i,j: incident with the edge *(i,j)*
A = [a_{ij}]: adjacency matrix, a_{ij} = 1 if there is a link from i to j, otherwise a_{ij} = 0.

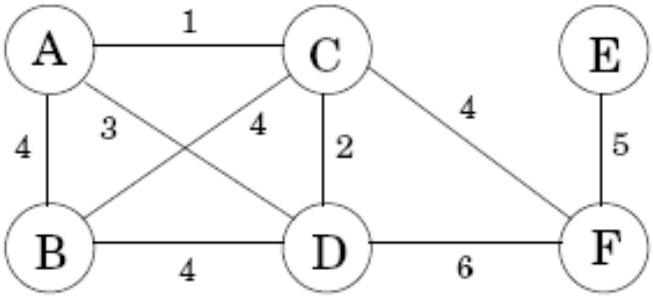
Weighted graphs

- Edges may also carry additional information
  - Signs (are we friends or enemies?)
  - Tie strength (how good are we as friends?)
  - Distance (how long is this road?)
  - Delay (how long does the transmission take?)
- In a ***weighted*** graph, every edge has an associated number called a ***weight***.
- In a ***signed*** graph, every edge has a + or a – sign associated with it.
- For showing the pattern of connections the actual placement of nodes is immaterial. All it matters is which nodes are linked to with others (and what is the weight if any).

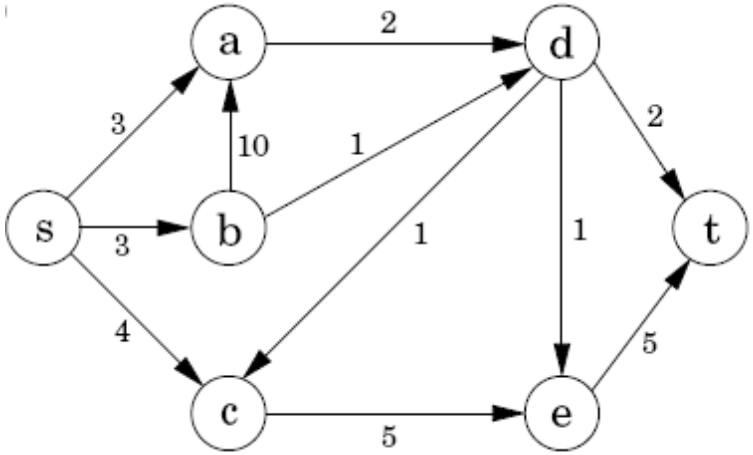
Weighted (Directed) Graphs

- A graph (directed or undirected) is **weighted**, if a measurable quantity (referred to as weight and usually denoted by *w*) is assigned to each edge $w: E \to \mathbb{R}$
- **Weight matrix** $W = [w_{ij}]$, w_{ij} is the weight of the link *(i, j).*
- *Weights:* costs, lengths or capacities in communications networks, distances, interest in social networks, cash flow in financial networks.

Weighted undirected

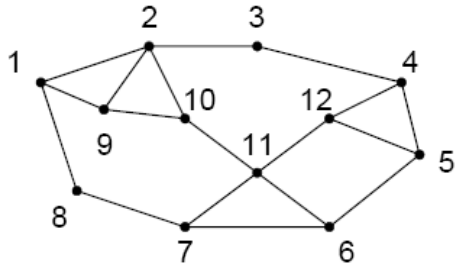


Weighted directed

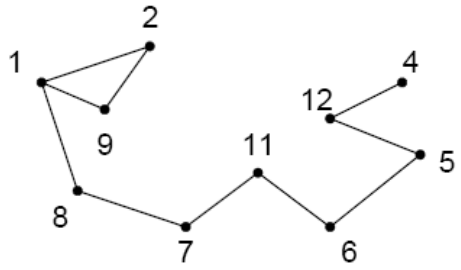


Subgraphs

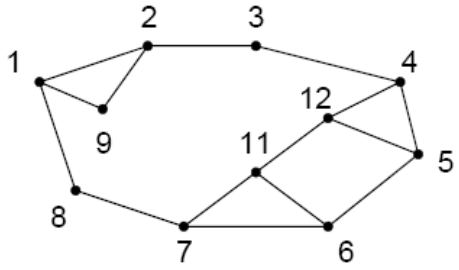
- **Subgraph:** A graph $G_0 = (V_0;E_0)$ is a subgraph of $G=(V;E)$ if $V_0 \subseteq V$ and, $E_0 \subseteq E$ where E_0 is defined on V_0, and this is denoted by $G_0 \subseteq G$
- **Induced**: contains all edges of G that link two nodes in V_0.
- **Spanning**: $V \equiv V_0$



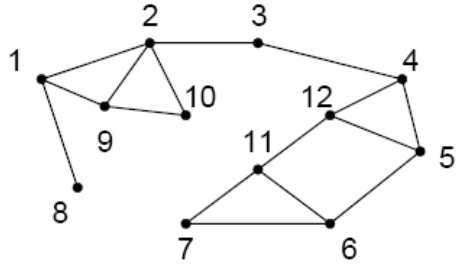
(a) original graph G



(b) subgraph of G



(c) induced subgraph of G



(d) spanning subgraph of G

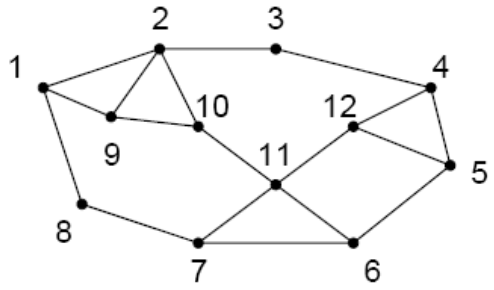
Node Degree & Strength

- Node degree
  - Undirected: number of neighbors k_i
  - Directed: number of in-/out-neighbors $k_i^{out} = \sum_{j=1}^{N} a_{ij}$

$$k_i^{in} = \sum_{j=1}^{N} a_{ji}$$

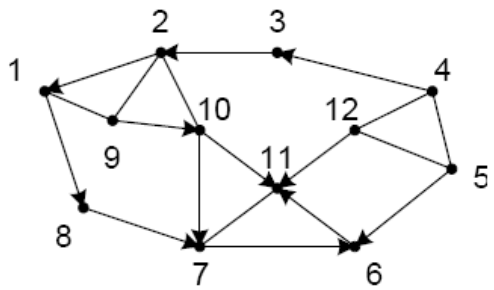
***d*-regular graph**: all vertices have degree equal to *d*

N(10)={2,9,11}
d(10)=k_{10}=3



(a) undirected graph

N(10)={2,7,9,11}
k_{10}^{in}=2,k_{10}^{out}=3



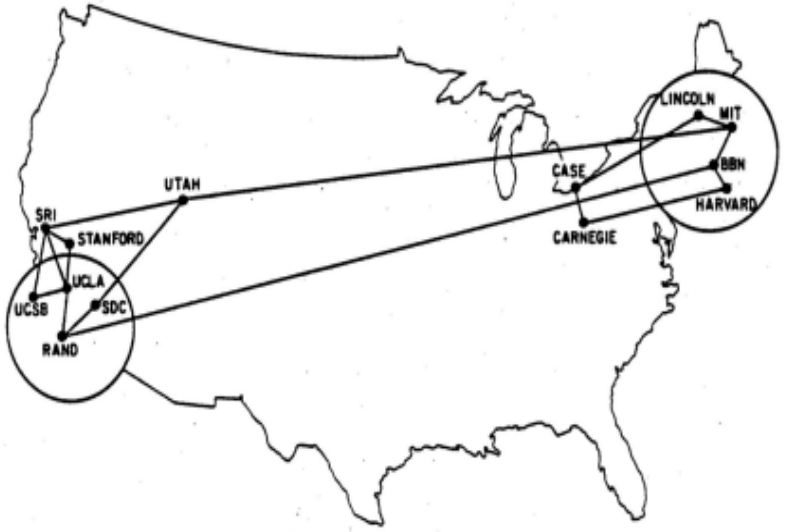
(b) directed graph

- Strength
  - Weighted graphs: Sum of neighboring link weights $s_i = \sum_{j=1}^{N} w_{ij}$
  - Directed graphs: in-/out-sum of link weights

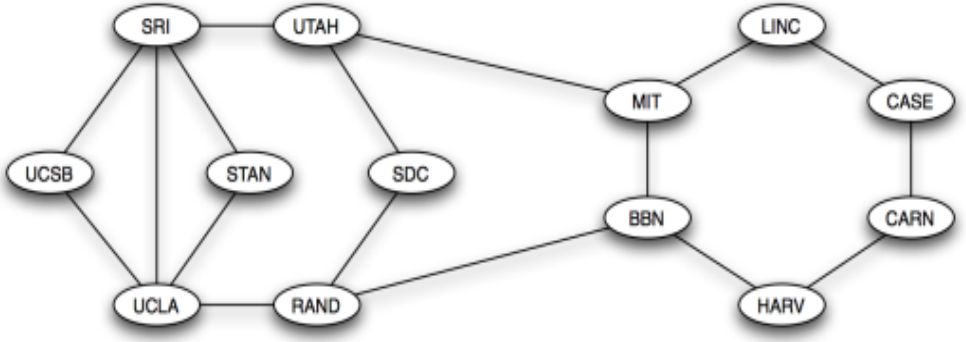
$$s_i^{out} = \sum_{j=1}^{N} w_{ij} \text{ and } s_i^{in} = \sum_{j=1}^{N} w_{ji}$$

ARPANET: Early Internet precursor

- December 1970 with13 nodes



For showing the pattern of connections the actual placement of nodes is immaterial. All it matters is which nodes are linked to with others (and what is the weight if any).



Graph representations

- Abstract graph theory is interesting in itself
- But in network science, items typically represent real-world entities
  - Some network abstractions are very commonly used. For example ARPANET is a communication network where nodes are computers or other devices that relay messages.
- Several indicative examples
  - Communication networks
    - telephone networks, Internet, cellular networks
  - Social networks
    - Nodes: People or groups of people;
    - Edges: friendship/contacts or social interaction
  - Information networks
    - Nodes: information sources such as Web sites or documents;
    - Edges: logical connections such as hyperlinks, citations, cross-references, etc.

Other networks: Transportation networks and structural networks

- Graph terminology often derived from transportation metaphors
e.g. “shortest path“, “flow“, “diameter“

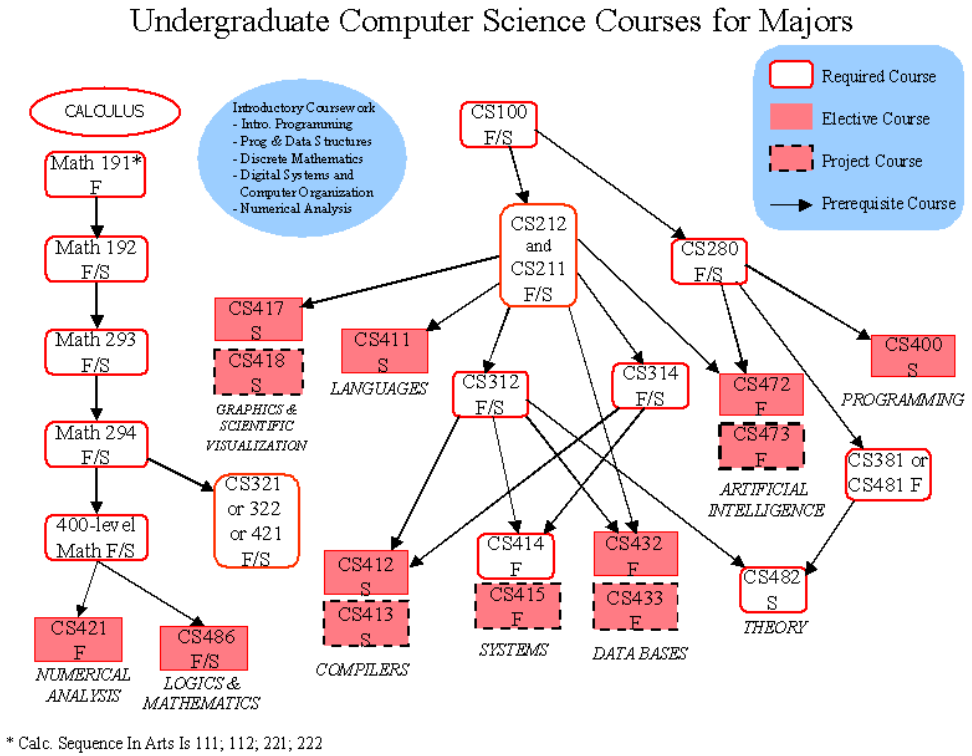
- Joints are nodes and physical linkages are edges

Dependency networks

Nodes are tasks and directed edges indicate that one task must be performed before others.

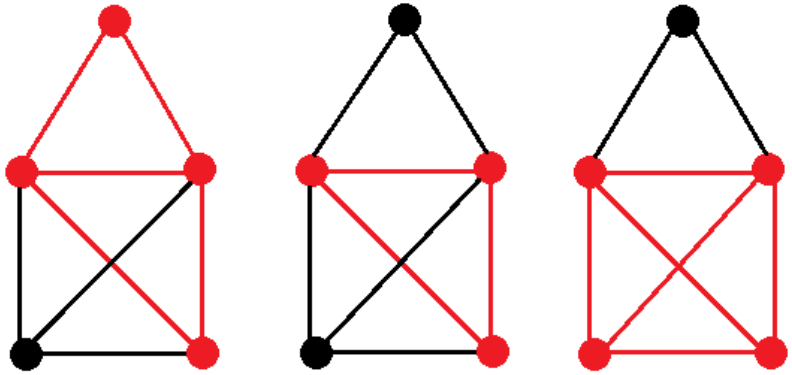
Examples include: software systems, indiustrial processes.

Interesting scheduling problems occur from analysis of such large depending networks



Cliques

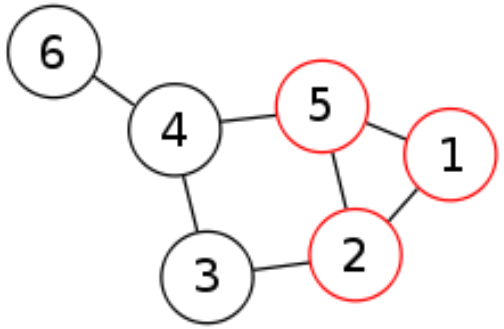
- **Clique**: subset of its vertices such that every two vertices in the subset are connected by an edge.
- **Maximal clique:** a clique that cannot be extended by including one more adjacent vertex.
- **Maximum clique:** is a clique of the largest possible size in a given graph.
- **Clique number, *ω(G)*, of a graph *G(V, E):*** the number of vertices in a maximum clique in *G*.



not a clique

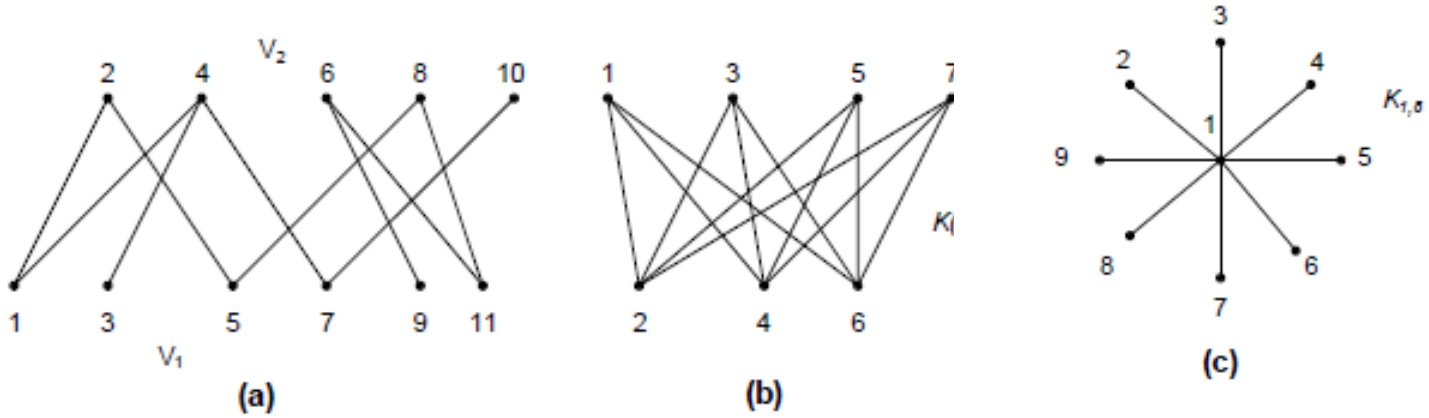
non-maximal clique

maximal clique



Bipartite Graphs

- **A bipartite graph *G*** is a graph whose vertex set *V* can be partitioned into two subsets V_1 and V_2, such that every edge of *G* joins only a node from V_1 with a node from V_2.
- **Graph vertices:** split into two distinct groups and relations take place only between members of different sets, e.g. advisors-advisees, doctors-patients, etc.
- **Complete bipartite:** all possible edges between V_1 and V_2 $(K_{m,n})$.
- **Star graph:** special case of bipartite.
- **Theorem:** A graph is bipartite iff it does not contain an odd cycle.

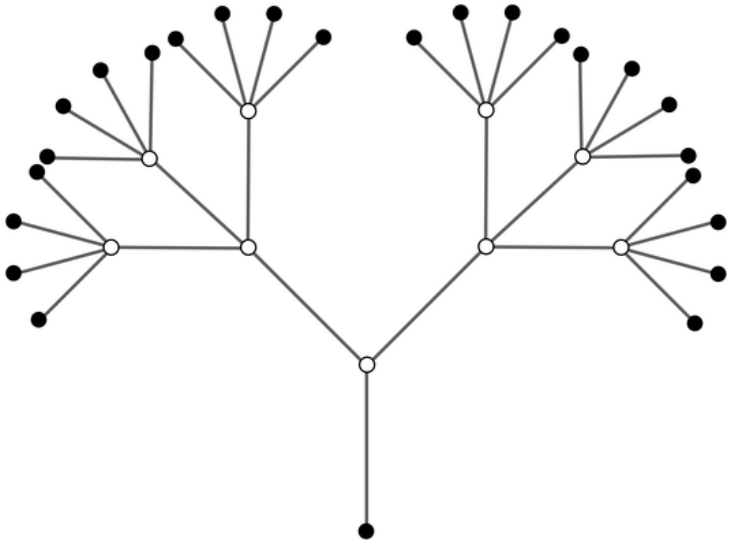


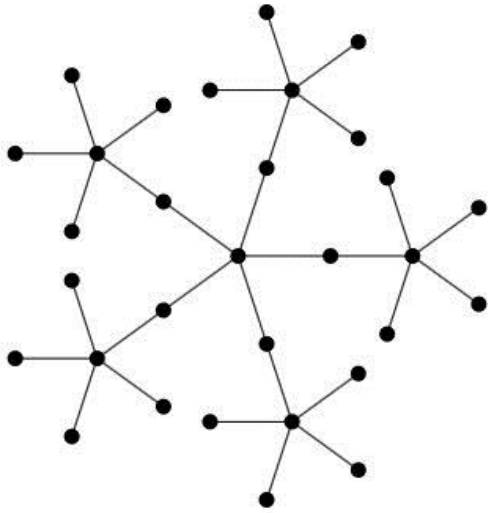
Trees

Theorem *The following statements are all equivalent for a graph G:*

1. *G is a tree.*
2. *G is a connected graph and every edge is a bridge.*
3. *G is a maximal acyclic graph; that is G is acyclic and if x and y are nonadjacent vertices of G, then $G + xy$ contains a cycle.*

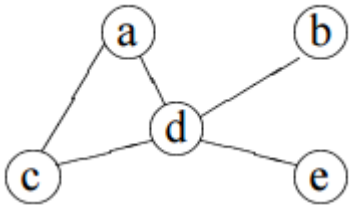
If omitted the network becomes divided.



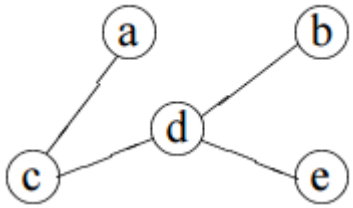


Spanning Trees

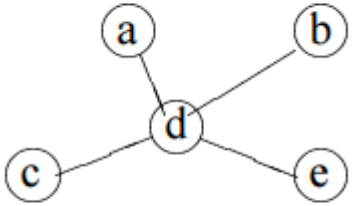
- **Spanning tree**: tree that contains every vertex.
- **Minimum weight spanning tree (MST):** spanning tree, which in addition has a minimum total edge weight sum.
- ***Every connected graph contains a spanning tree.***



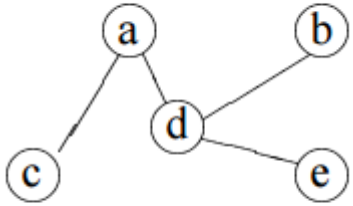
Graph



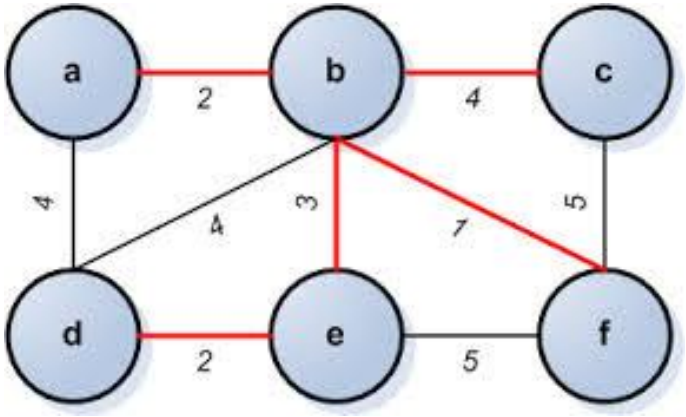
spanning tree 1



spanning tree 2



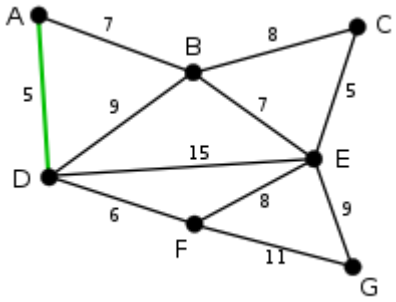
spanning tree 3



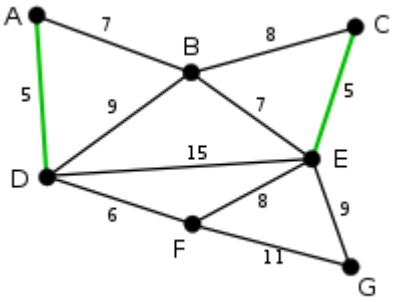
Spanning Trees – Kruskal's algorithm

- **Kruskal's algorithm**: minimum-spanning-tree algorithm which finds an edge of the least possible weight that connects any two trees in the forest.
  - *Greedy algorithm, adding increasing cost edge at each step.*

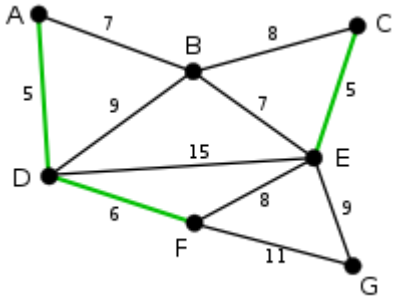
1



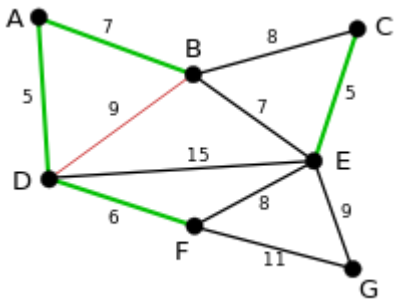
2



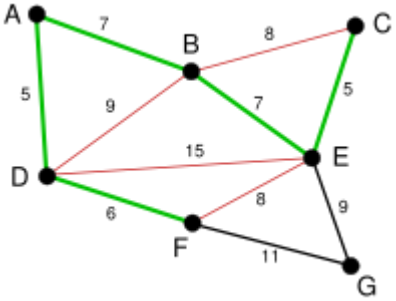
3



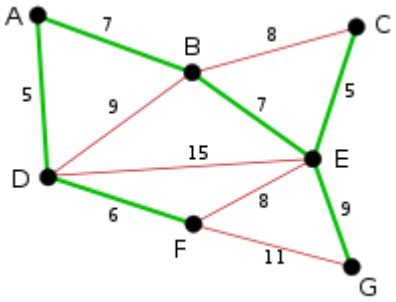
4



5

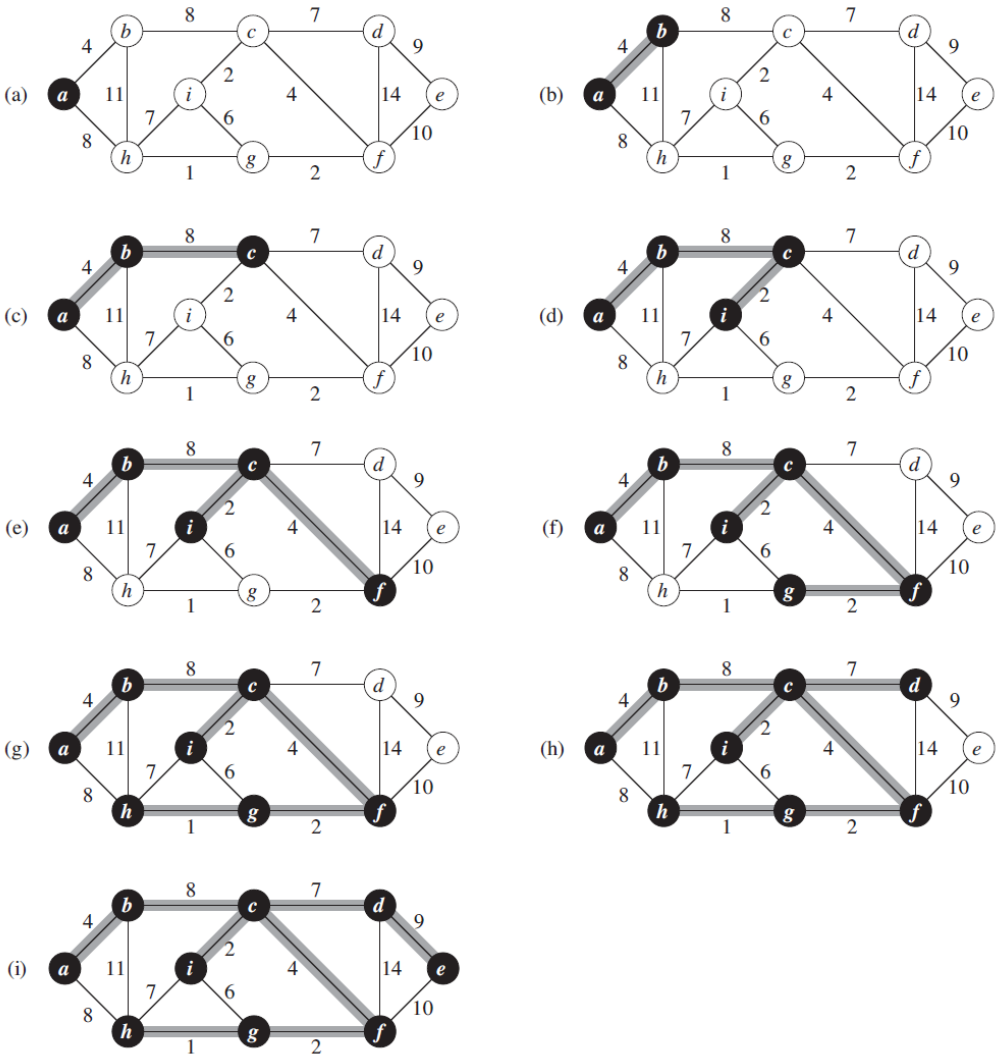


6



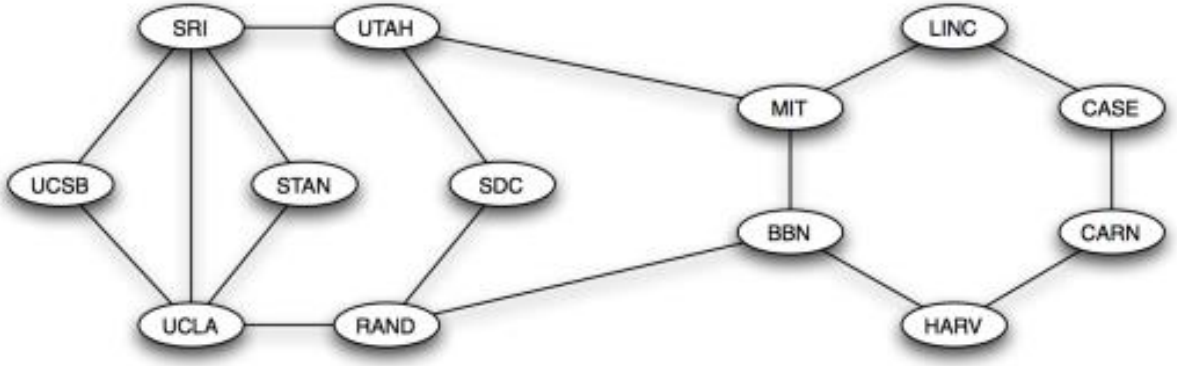
Spanning Trees – Prim's algorithm

- **Prim's algorithm**: starting with a specific node, finds minimum-spanning-tree by adding an edge of the least possible weight that maintains the tree up to the given step
  - *Greedy algorithm, adding increasing cost edge at each step while maintaining a connected tree discovered already*



Paths

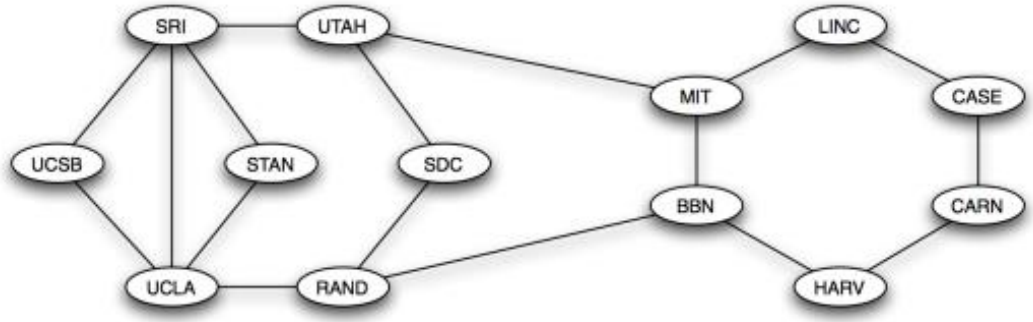
- **Path:** Sequence of nodes with the property that each consecutive pair in the sequence is connected by an edge (can also be defined as a sequence of edges)
- A path (in general) can repeat nodes. We will typically consider paths that do not repeat nodes – refer to as **simple path**.



- MIT – BBN – RAND – UCLA is a path (simple)
- SRI – STAN – UCLA – SRI – UTAH – MIT (traverse the same node multiple times – non-simple path)

Cycles

- **Cycles** are ring structures that begin and end in the same node (specifc form of non-simple path)
  - LINC – CASE – CARN – HARV – BBN – MIT – LINC is a cycle
- In the ARPANET example, every edge belongs to a cycle
  - By design. Why? If any edge fails there should be a way to go from any node to any other node.



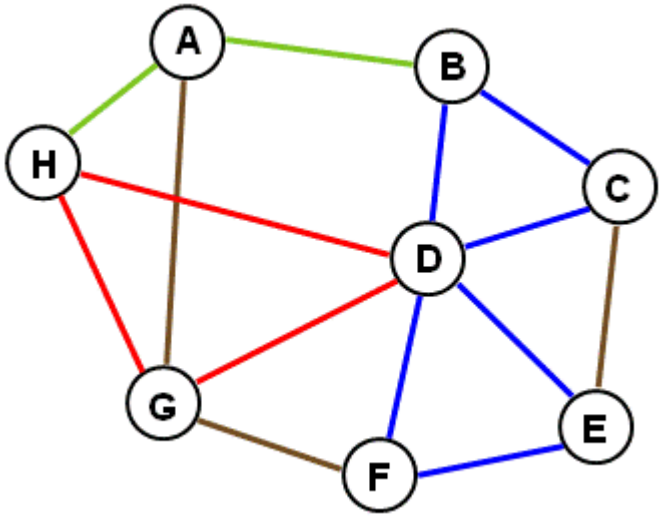
- In social networks cycles may also occur naturally (friend, of a friend, of a friend, ...)

Paths & Cycles - Example

- **Walk** of G: alternating sequence of vertices and edges
- Closed walk if $v_0 = v_n$;
- **Cycle**: closed path as C_n (cycle on n points, $n > 3$)
- **Length of walk** = # of edges of the network traversed

$$v_0, x_1, v_1, \ldots, v_{n-1}, x_n, v_n$$

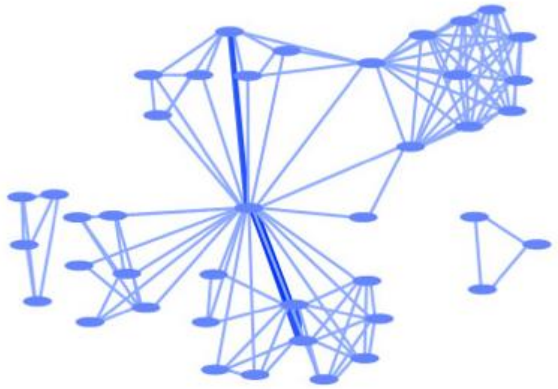
path H-A-B; closed walk with a repeated vertex B-D-E-F-D-C-B; cycle with no repeated edge H-D-G-H



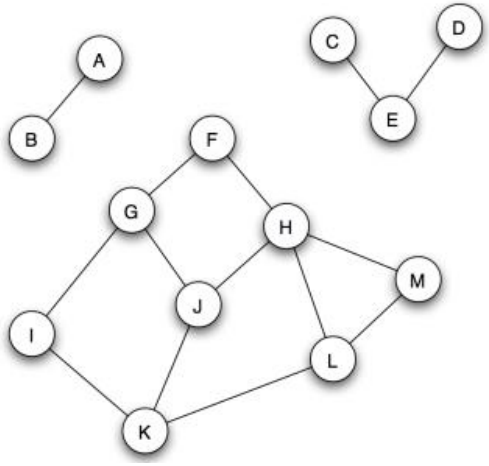
Connectivity

- Can every node in a graph be reached from any other node through a path?
  - If so, the graph is *connected*

  Therefore a graph is connected if for every pair of nodes there is a path between them.
- The ARPANET graph is connected
- In many cases, graphs may not be connected
  - Social networks (there may be two people for which is not possible to construct a path from one to the other)
  - Collaboration networks



Collaboration graph of a research center



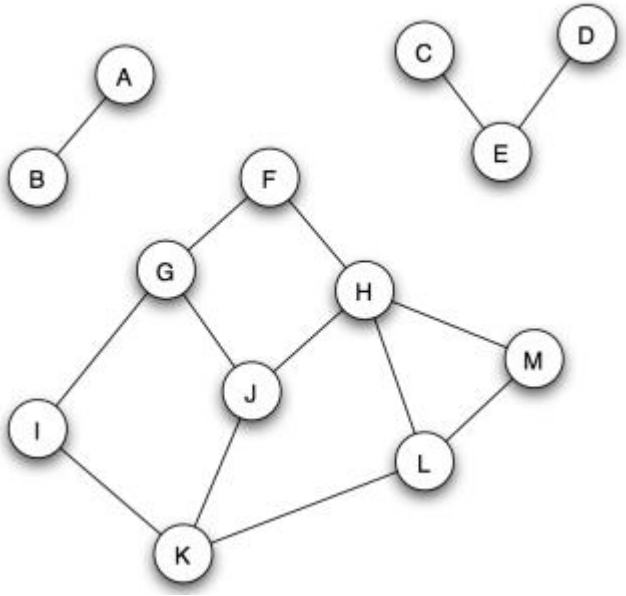
Is this graph connected?

- A,B not connected to other nodes
- C,D,E not connected to other nodes

Components

- **Observation:** If a graph is not connected then there exists a set of connected "pieces", i.e. groups of nodes, that each group is connected when considered as a graph in isolation, and not two groups overlap.
- In general, if a graph is not connected, it tends to break into pieces that themselves are connected
- **Formal:** A **connected component** of a graph is a subset of the nodes such that:
  - (i) every node in the subset has a path to every other node
  - (i) the subset is not part of some larger set with the property that every node can reach every other node.
- Intuitively:
  - (i) the component is internally connected
  - (ii) it is really "free-standing" piece of graph (not a connected part of a larger piece).

Components



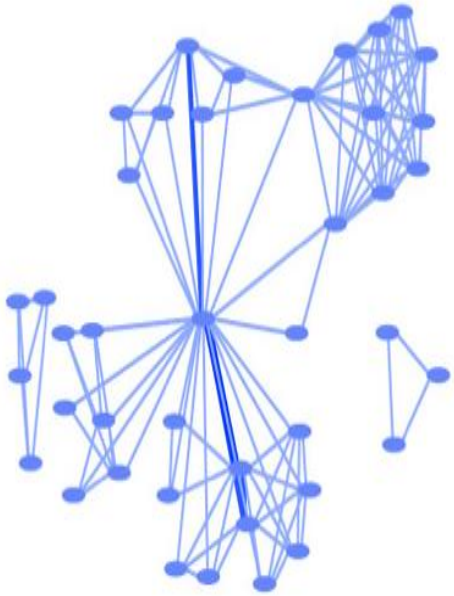
- Three connected components
  - {A,B}, {C,D,E}, {F,G,...,M}

Connectivity

- **node-connectivity** $\kappa = \kappa(G)$: min # of nodes whose removal results in disconnected/trivial graph
- **edge-connectivity** $\lambda = \lambda(G)$: min # of edges whose removal results in disconnected/trivial graph
- **connectivity pair**: ordered (α,β), $\alpha,\beta>0$, s.t. there exist α vertices β edges whose removal disconnects the graph, while no $\alpha-1$ nodes and $\beta-1$ edges have this property
- A graph is ***n*-connected** if $\kappa(G) \geq n$ and ***n*-edge-connected** if $\lambda(G) \geq n$

Components: Analysis and Discussion

- A first global way to look at graph structure
  - prominent node at the center and tightly-knit groups linked to this node but not to each other
  - we can understand what is holding a component together
  - the largest component will break into three distinct components if the prominent central node were removed
- Analyzing graphs in terms of densely connected regions and the boundaries of regions
  - for instance, only include edges with weights above a threshold, then gradually increase the threshold
  - the graph will fragment into more and more components

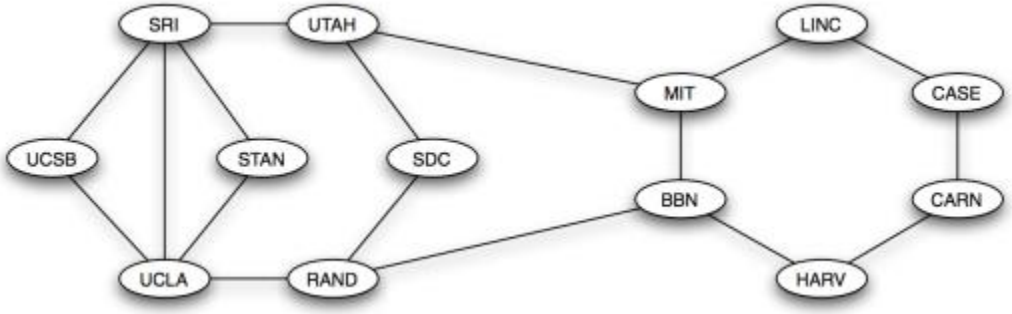


Giant components

- Large complex networks often are not connected but have what is called a very large connected component (**giant component**), which is a a connected component containing a significant fraction of all nodes.
- Rare that two or more of these will exist in a graph. Why? If two giant components existed then just a single edge from one member of one giant component would suffice to connect to the other.

Distance

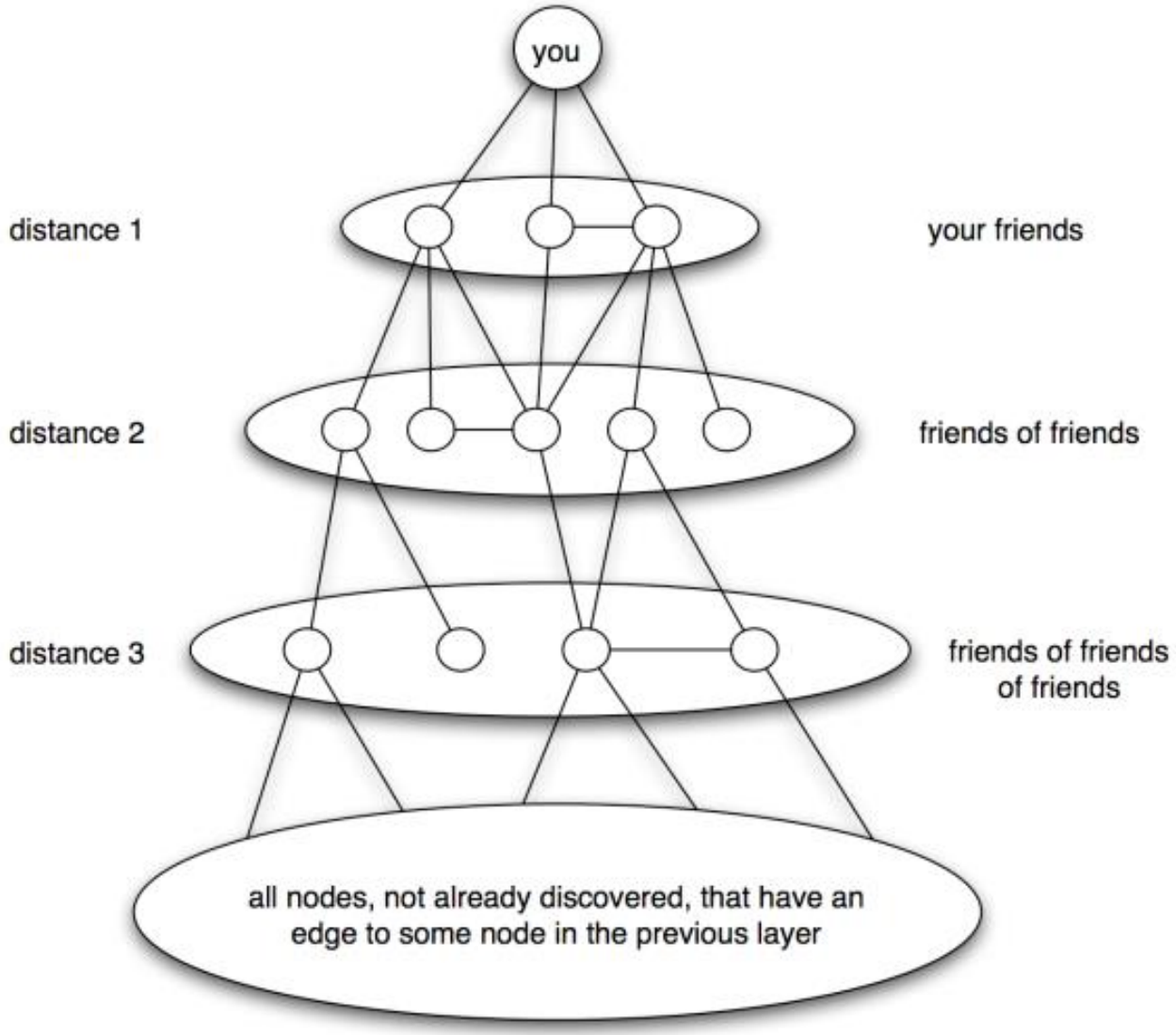
- How long is a path?
- **Length of a path** is the number of hops (steps) it conatins from beginning to end (i.e. number of edges in the sequence that comprises it)
- **Distance:** The length of the shortest path between two nodes (**Distance *d(u,v)*** between two vertices *u*, *v* in *G*: length of shortest path (SP) joining them)
  - Just number of edges (thought of as all edges having weight of 1)
  - Above definition can be generalized for other metrics based on the weight of each edge
- What's the distance between MIT and SDC?



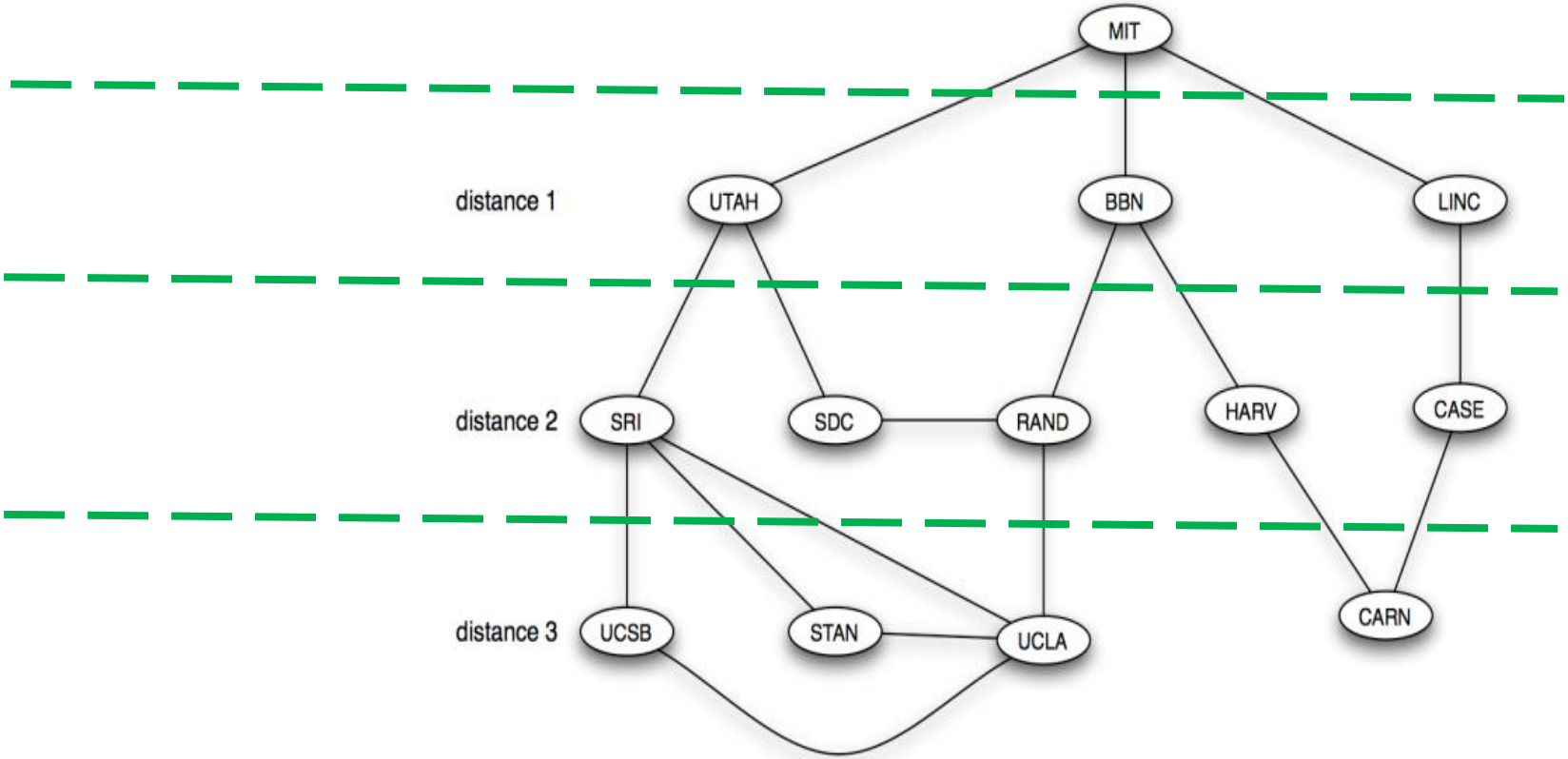
Distance: Breadth-first search (BFS)

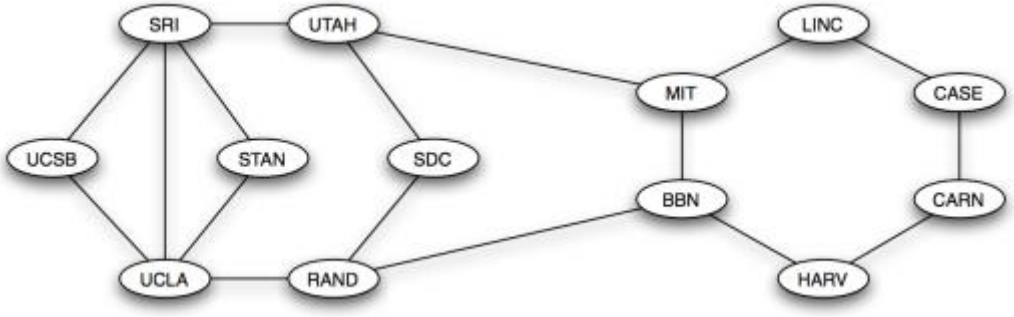
- Systematic method to determine distances (metric is number of edges)
- From the given node (*root*)
  - Find all actual friends that are distance 1. That is, find all nodes that are directly connected
    - These are labeled as “distance 1“
  - Find all of their friends (not counting people who are already friends of yours). That is, find all nodes that are directly connected to nodes at distance 1
    - If these nodes are not at distance 1, we label them as “distance 2“
  - ...
  - Find all nodes that are directly connected to nodes at distance *j*
    - If these nodes are not already of distance at most *j*+1, we label them as „distance *j*+1“
- Serves as a useful framework to conceptually organize the structure of the graph, by arranging nodes based on their distances from a fixed starting point, and creating a **multi-layer structure** where: each new created layer is built from nodes that (i) have not already been discovered in earlier layers, and (ii) have an edge to some node in the previous layer.

Distance: Breadth-first search (BFS)



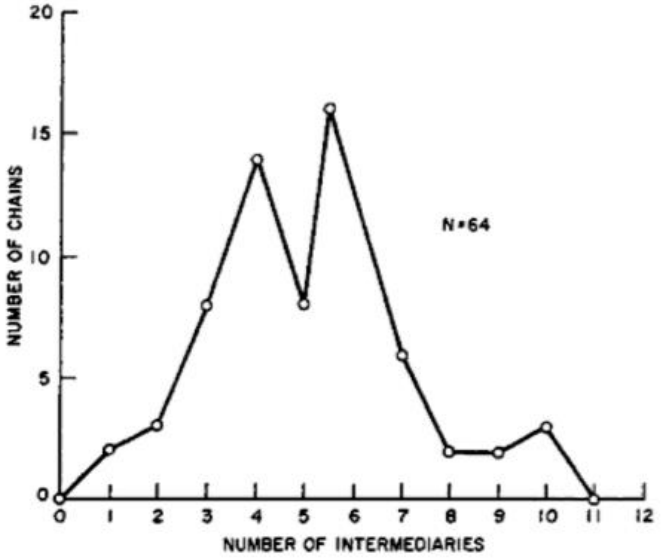
Distance: Breadth-first search (BFS)





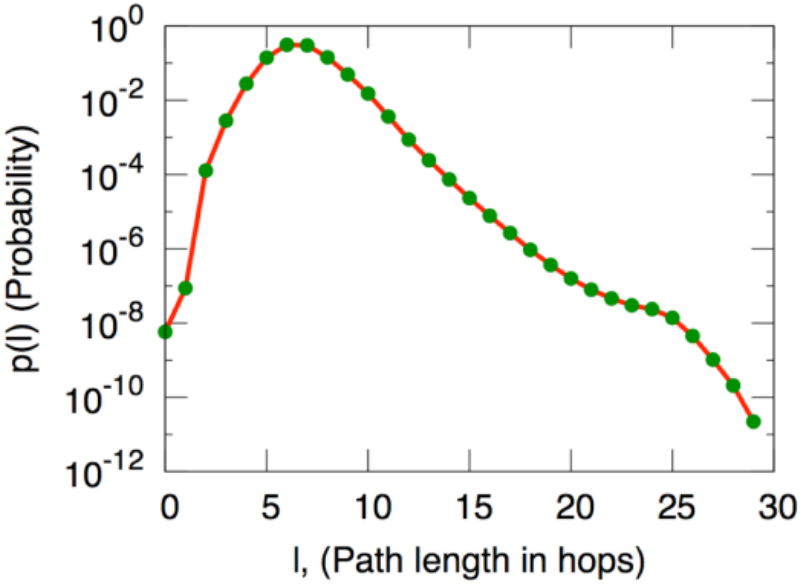
Small-world phenomenon-Six degrees of separation

- **Social networks tend to have very short paths between arbitrary pais of people**
- First experiment done by Stanley Milgram in 1960s (research budget $680)
  - 296 randomly chosen starters. Asked to send a letter to a target (in Boston), by forwarding to someone they know personally and so on. Number of steps counted.
- median hop number of 6 for successful chains – six degrees of separation
  - This study has since been largely discredited



Six degrees of separation

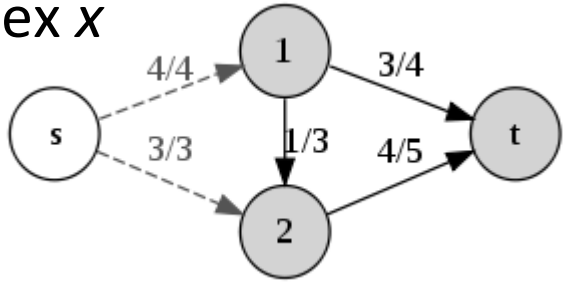
- Modern experiment by Leskovec and Horvitz in 2008
- Look at the 240 million user accounts of Microsoft Instant Messenger
- Complete snapshop – no missing data
- Found a giant component with very small distances
- A random sample of 1000 users were tested and performed Breadth-first search
  - Why do they look only at a sample? Due to time and computational constraints and feasibility.
- Estimated average distance of 6.6, median of 7



Network Flow

- Finite directed graph with source *s*, sink *t*
- **Flow**: nonnegative function on the edges, where the value $f(\overrightarrow{xy}) = f(x,y)$ is the amount of flow traversing edge-arc $\overrightarrow{xy}$
- Total flow leaving vertex *x* = total flow into vertex *x*

$$\sum_{y \in \Gamma^+(x)} f(x,y) = \sum_{z \in \Gamma^-(x)} f(z,x)$$



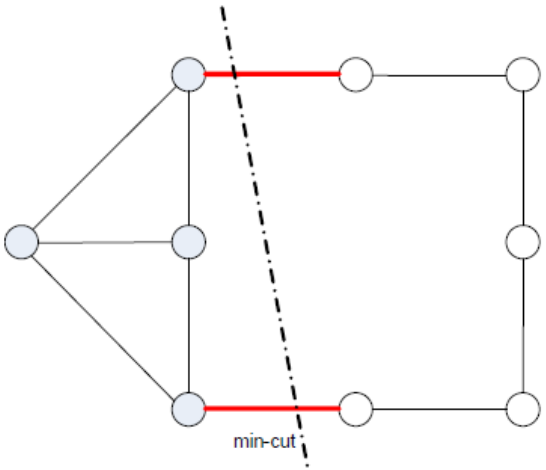
- Total flow leaving the source equals the total flow entering the sink

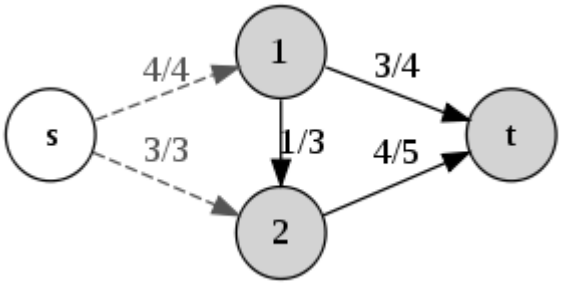
$$\sum_{y \in \Gamma^+(s)} f(s,y) - \sum_{y \in \Gamma^-(s)} f(y,s) = \sum_{y \in \Gamma^-(t)} f(y,t) - \sum_{y \in \Gamma^+(t)} f(t,y)$$

- **Capacity *c(x,y)>0***: the flow of this arc cannot exceed *c(x,y)* for any reason

Max Flow-Min Cut Theorem

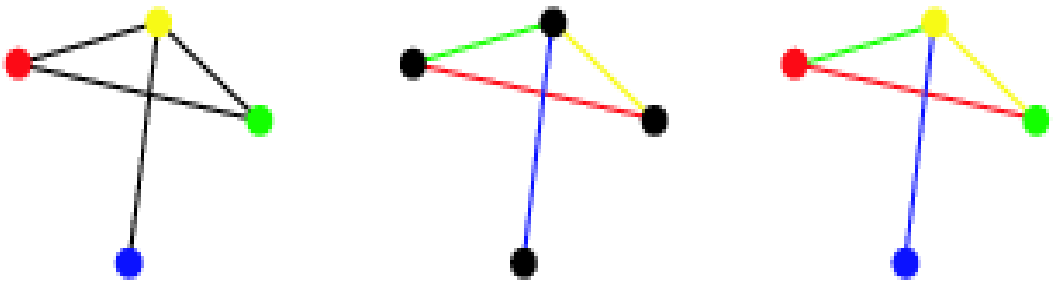
- The max. flow value from *s* to *t* is equal to the min. of the capacities of cuts separating *s* from *t*
  - **Cut**: a partition of vertices into two disjoint subsets
  - **Cut-set**: set of edges whose endpoints are in different subsets of the partition
  - Enables the computation of the max. transferring capacity of the network
  - Enables to locate the bottleneck part of the graph restricting capacity
- **Thm**: max. of flow value from a set of sources to a set of sinks = min. of capacities of cuts separating the sources from the sinks





Graph Coloring

- The assignment of labels or colors to the edges or vertices of a graph
- **Edge Coloring** - an assignment of labels or colors to each edge of a graph such that adjacent edges (or the edges bounding different regions) must receive different colors.
- **Vertex Coloring** - an assignment of labels or colors to each vertex of a graph such that no edge connects two identically colored vertices



vertex-colored graph *edge-colored graph* *vertex- and edge-colored graph*

Coverage & Coloring

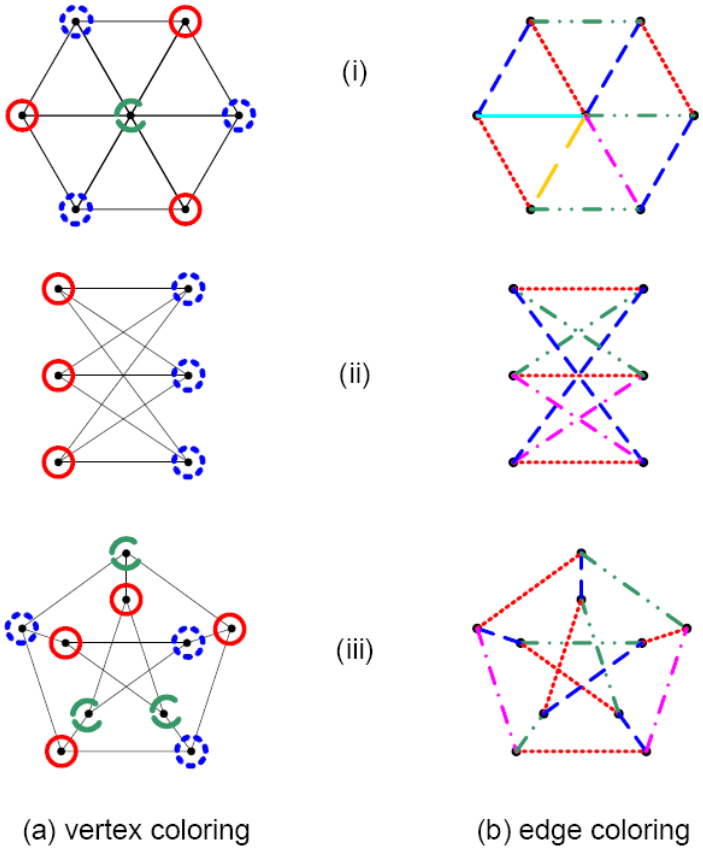
- **Coloring**: assignments of different colors to the vertices (edges) so that adjacent vertices (edges) have distinct colors
- Coloring ↔ coverage
- **Chromatic number *χ(G)***

min. # of colors in a vertex coloring

- **Edge-chromatic number *χ'(G)***

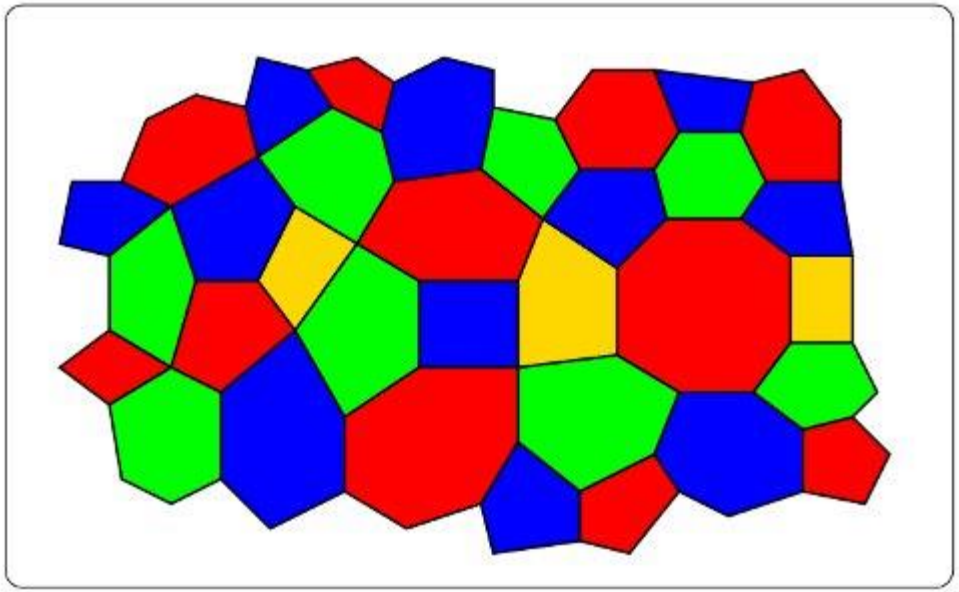
min. # of colors in a edge coloring

- ***n*-coloring** of *G* uses *n* colors
- ***n*-chromatic** if $\chi(G) = n$
- Different color classes define independent sets



The Four Color Problem

- The four color problem states that any map in a plane can be colored using four-colors in such a way that regions sharing a common boundary (other than a single point) do not share the same color.



4–Colored Map