The Design and Analysis of Algorithms

Lecture 4 Graph I

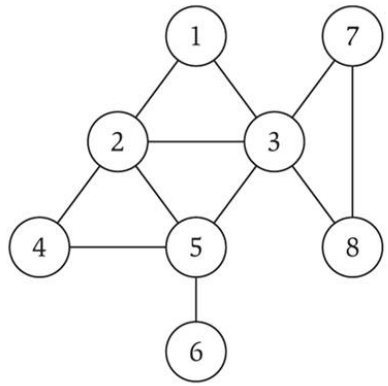
Zhenbo Wang

Department of Mathematical Sciences, Tsinghua University

Content

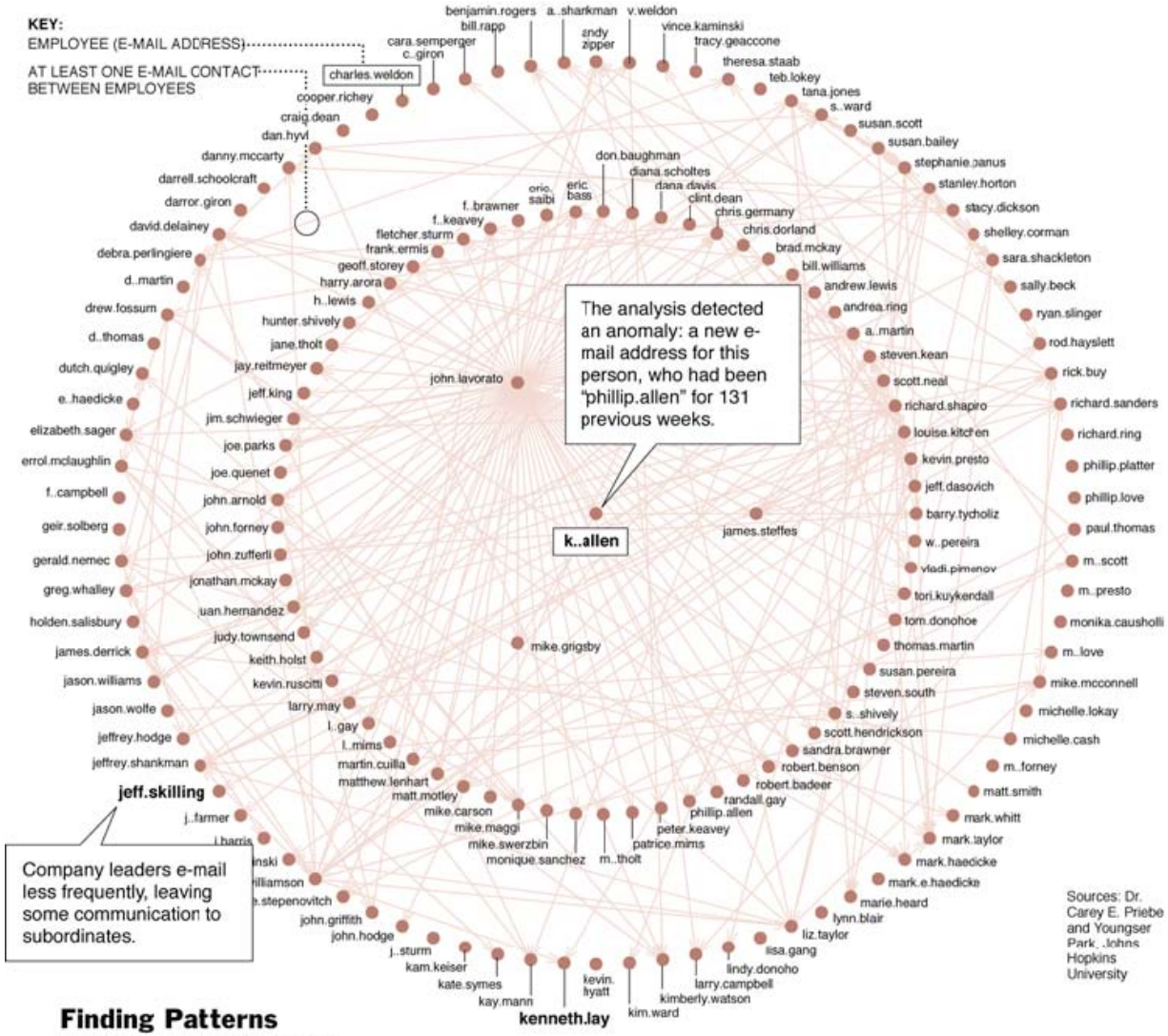
Undirected Graphs

- *Undirected graph*. $G = (V, E)$.
- $V =$ nodes.
- $E =$ edges between pairs of nodes.
- Graph size parameters: $n = |V|$, $m = |E|$.

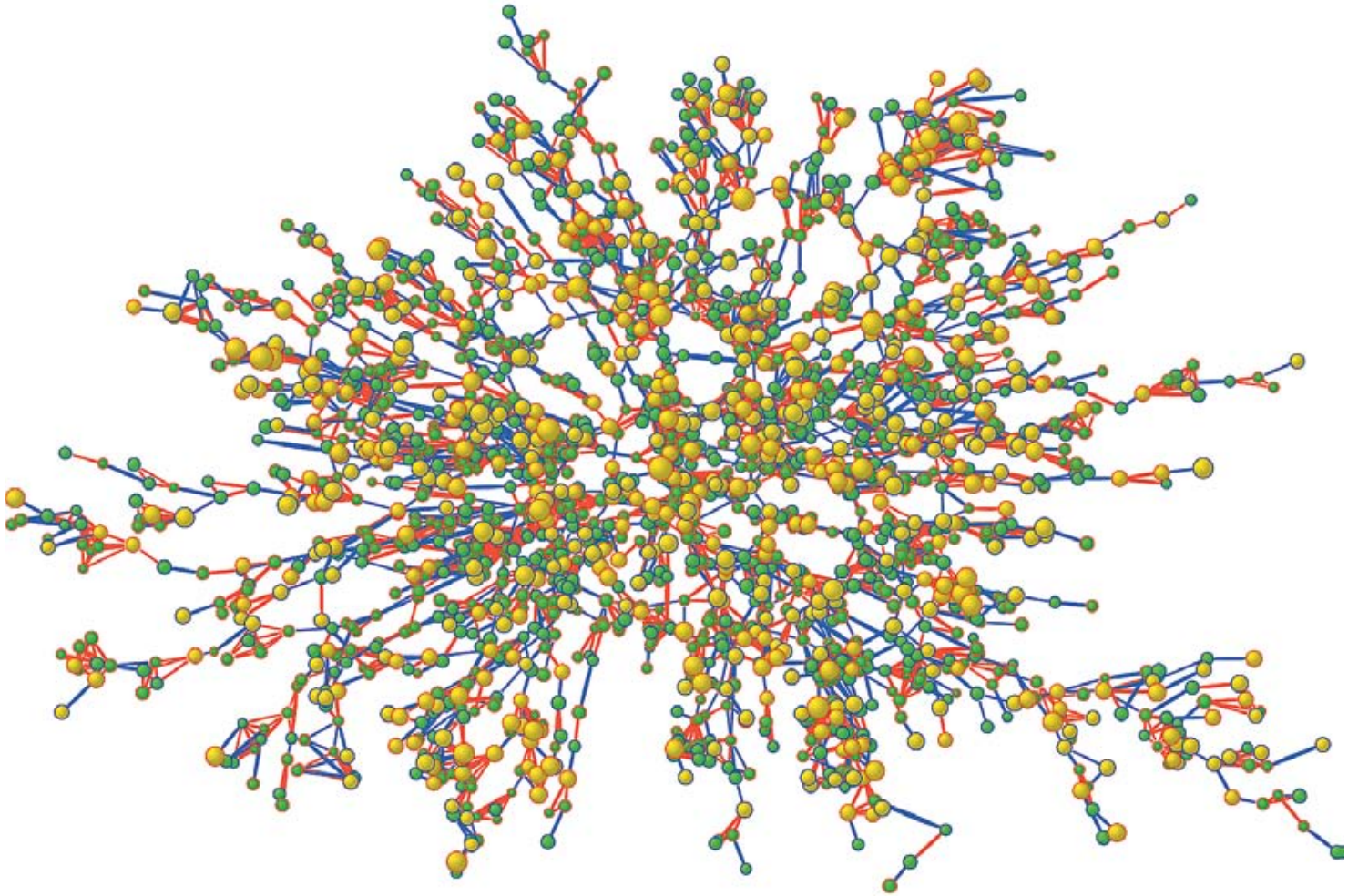


- $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$.
- $E = \{1–2, 1–3, 2–3, 2–4, 2–5, 3–5, 3–7, 3–8, 4–5, 5–6, 7–8\}$.
- $m = 11$, $n = 8$.

One week of Enron emails



Social Network



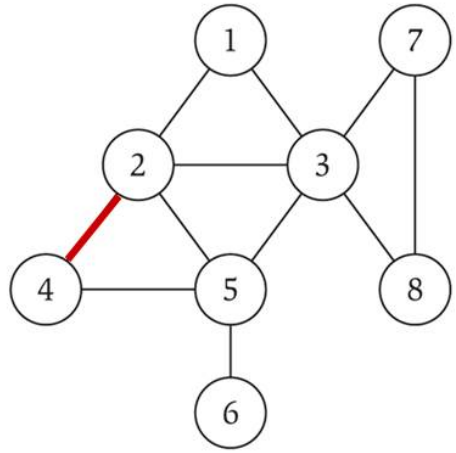
Some Graph Applications

graph	**node**	**edge**
communication	telephone	cable
circuit	gate, register, processor	wire
financial	stock, currency	transactions
transportation	street intersection, airport	highway, airway
internet	web pages	hyperlinks
social relationship	person, actor	friendship, movie cast
neural network	neuron	synapse
protein network	protein	protein-protein interaction
molecule	atom	bond

Graph Representation: Adjacency Matrix

- *Adjacency matrix*. n-by-n matrix with $A_{uv} = 1$ if (u, v) is an edge.
- Two representations of each edge.
- Space proportional to n^2.
- Checking if (u, v) is an edge takes $\Theta(1)$ time.
- Identifying all edges takes $\Theta(n^2)$ time.

Adjacency Matrix

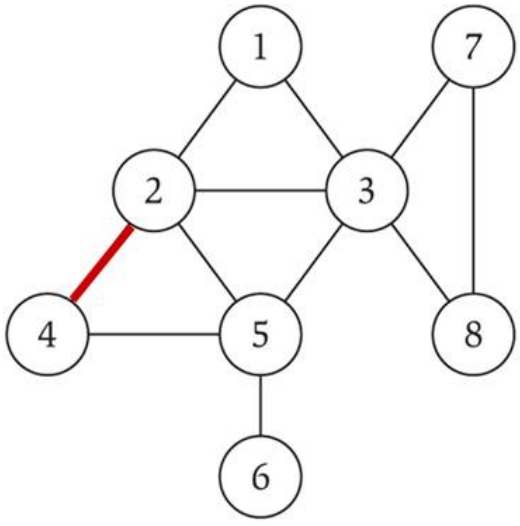


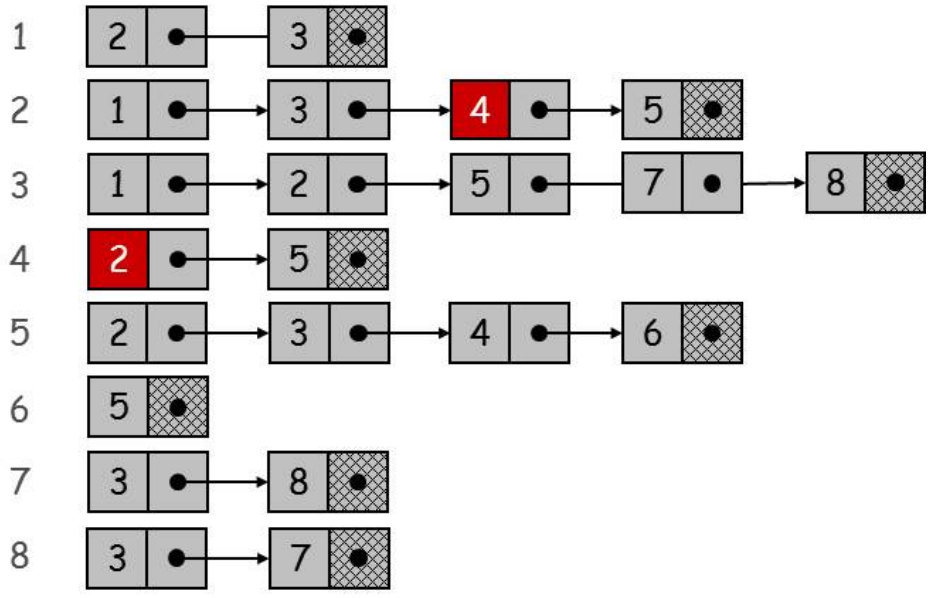
	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

Graph Representation: Adjacency List

- *Adjacency list*. Node indexed array of lists.
- Two representations of each edge.
- Space proportional to $m + n$.
- Checking if (u, v) is an edge takes $O(deg(u))$ time.
- Identifying all edges takes $(m + n)$ time.

Adjacency List

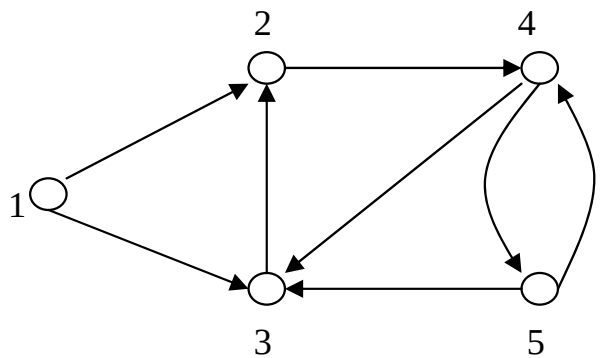




Directed Graphs

- *Directed graph*. $G = (V, E)$.

  Edge (u, v) goes from node u (tail) to node v (head).



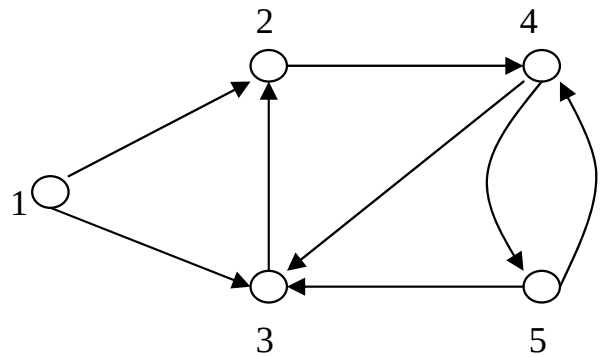
Graph Representation: Incidence Matrix

- *Incidence matrix*. n-by-m matrix B, one row represents one node and one column represents one edge.

$$
\begin{aligned}
B &= (b_{ue})_{n\times m} \in \{-1, 0, 1\}^{n\times m}, \\
b_{ue} &= \begin{cases} 1, & \exists\, v \in V, e = (u, v) \in E, \\ -1, & \exists\, v \in V, e = (v, u) \in E, \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}
$$

- Two representations of each edge.
- Space proportional to mn.
- *Property*. If a directed graph with n nodes is connected, then the rank of its incidence matrix is $n - 1$.

Incidence Matrix



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

Graph Representation: Forward Star

- *Forward star*. Store the two nodes of each edge.

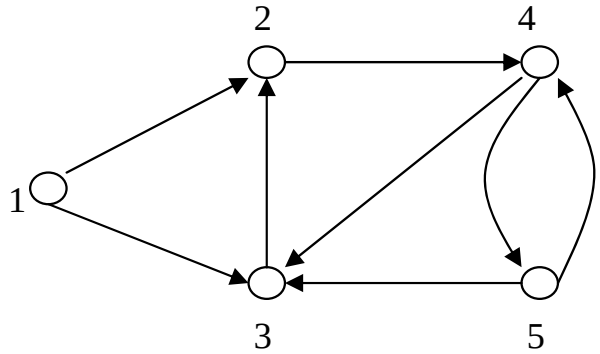
  Number the edges in a specific order: first those emanating from node 1, then those emanating from node 2, and so on.

  Maintain a pointer with each node u, that indicates the smallest numbered edge that emanates form node u.

- Reverse star.

- Forward and reverse star.

Forward Star



u	1	2	3	4	5	6
points	1	3	4	5	7	9

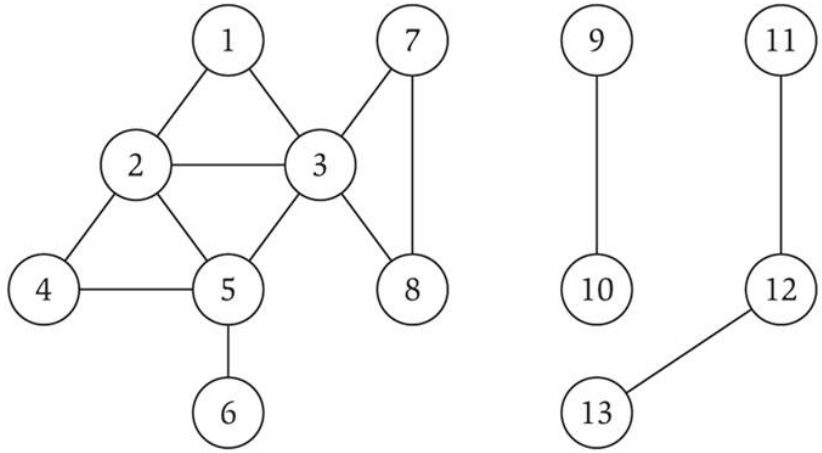
edges	1	2	3	4	5	6	7	8
tails	1	1	2	3	4	4	5	5
heads	2	3	4	2	3	5	3	4
wights	8	9	6	4	0	3	6	7

Paths and Connectivity

Def. A *path* in an undirected graph $G = (V, E)$ is a sequence of nodes $v_1, v_2, \cdots, v_k$ with the property that each consecutive pair v_{i-1}, v_i is joined by an edge in E.

Def. A path is *simple* if all nodes are distinct.

Def. An undirected graph is connected if for every pair of nodes u and v, there is a path between u and v.



Cycles

Def. A *cycle* is a path $v_1, v_2, \cdots, v_{k-1}, v_k$ in which $v_1 = v_k$, $k > 2$, and the first $k - 1$ nodes are all distinct.

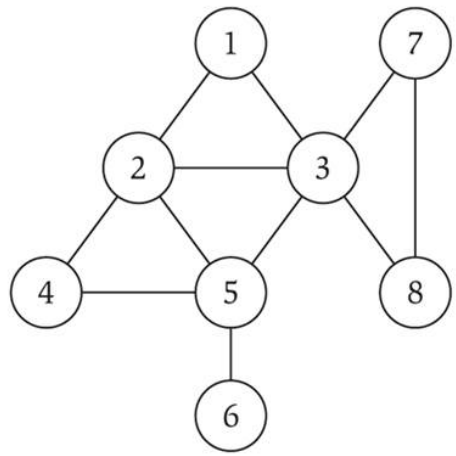


Figure 1: cycle $C = 1 - 2 - 4 - 5 - 3 - 1$.

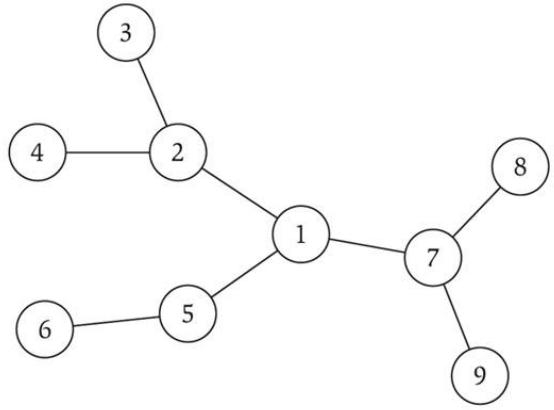
Trees

Def. An undirected graph is a tree if it is connected and does not contain a cycle.

Theorem 1

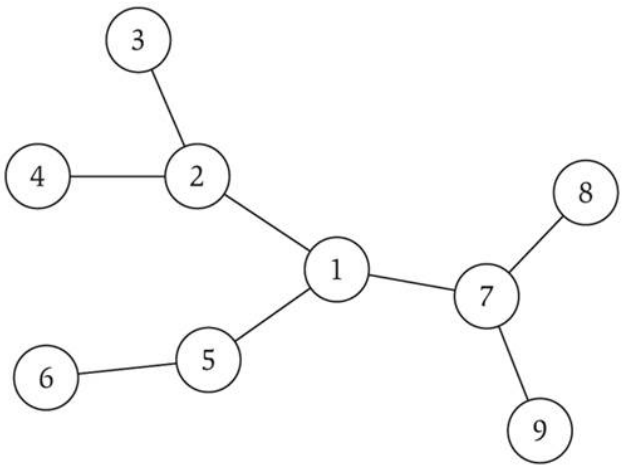
Let G be an undirected graph on n nodes. Any two of the following statements imply the third.

- *G is connected.*
- *G does not contain a cycle.*
- *G has n – 1 edges.*

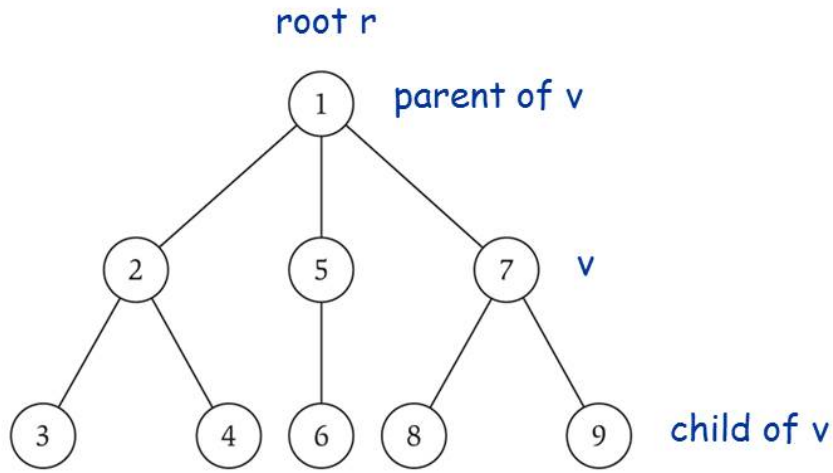


Rooted Trees

- *Rooted tree*. Given a tree T, choose a root node r and orient each edge away from r.



a tree



the same tree, rooted at 1

Connectivity

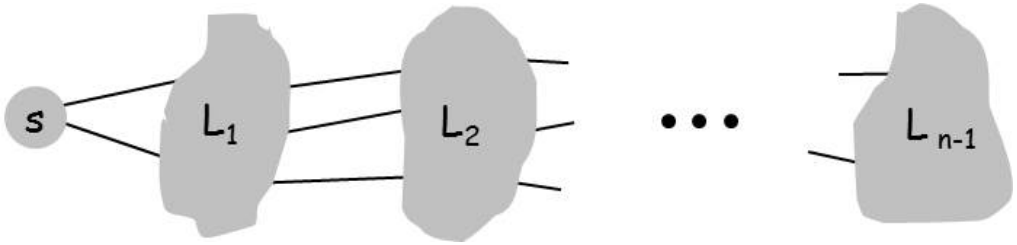
- *s – t connectivity problem*. Given two node *s* and *t*, is there a path between *s* and *t*?
- *s – t* shortest path problem. Given two node *s* and *t*, what is the length of the shortest path between *s* and *t*?
- Applications.

  Renren.

  Maze traversal.

  Fewest number of hops in a communication network.

Breadth First Search



- Explore outward from s in all possible directions, adding nodes one "layer" at a time.
- BFS algorithm.

  $L_0 = \{s\}$.

  $L_1 =$ all neighbors of L_0.

  $L_2 =$ all nodes that do not belong to L_0 or L_1, and that have an edge to a node in L_1.

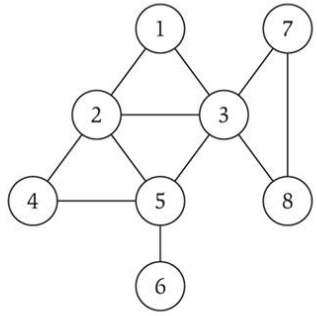
  ...

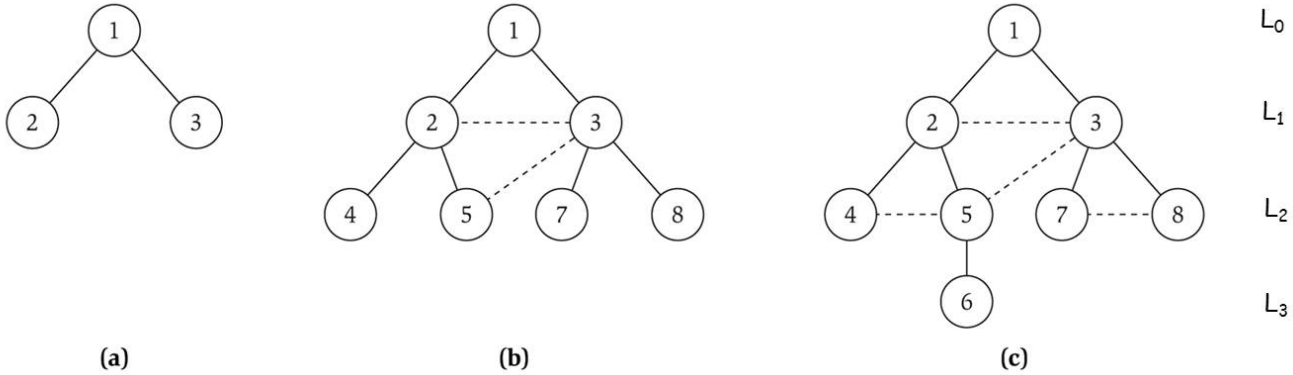
Breadth First Search

Theorem 2

For each i, L_i consists of all nodes at distance exactly i from s. There is a path from s to t iff t appears in some layer.

- *Property*. Let T be a BFS tree of $G = (V, E)$, and let (x, y) be an edge of G. Then, the level of x and y differ by at most 1.





Breadth First Search: Analysis

Theorem 3

The above implementation of BFS runs in $O(m + n)$ time if the graph is given by its adjacency representation.

Pf.

When we consider node u, there are $deg(u)$ incident edges (u, v).

Total time processing edges is $\sum_{u \in V} deg(u) = 2m$. □

Connected Component

- *Connected component*. Find all nodes reachable from s.
- BFS or DFS explores the connected component in $O(m + n)$ time.

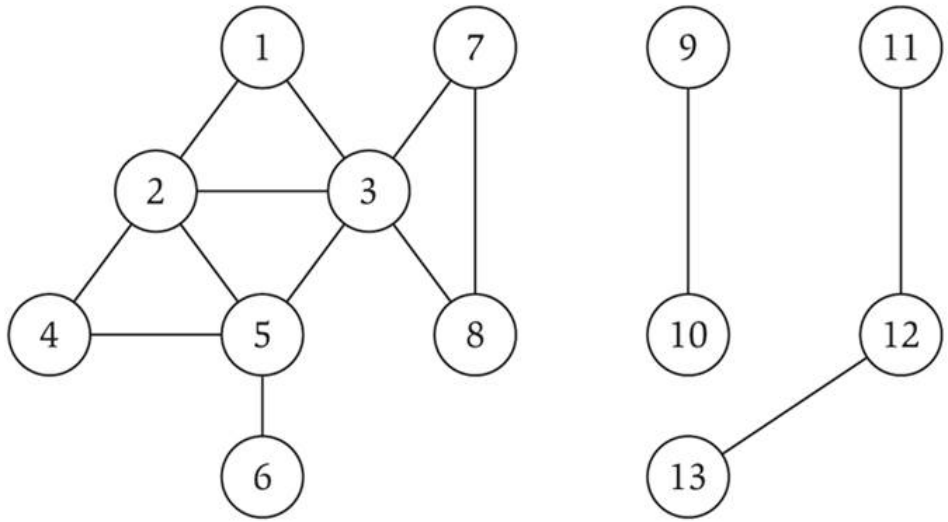


Figure 2: Connected component containing node 1 $= \{1, 2, 3, 4, 5, 6, 7, 8\}$.

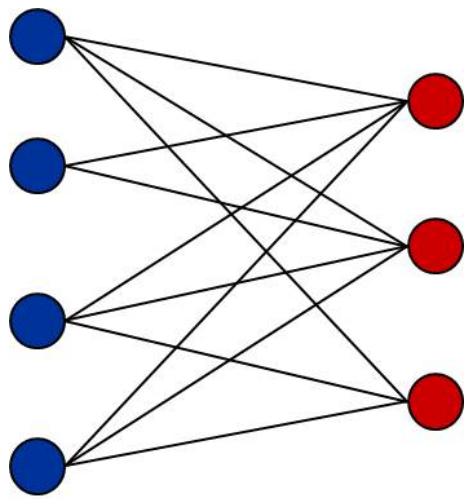
Bipartite Graphs

Def. An undirected graph $G = (V, E)$ is bipartite if the nodes can be colored red or blue such that every edge has one red and one blue end.

- *Applications*.

  Stable marriage: men = red, women = blue.

  Scheduling: machines = red, jobs = blue.



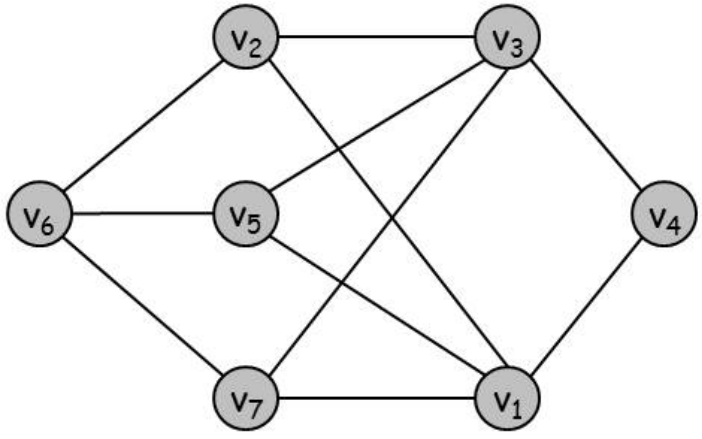
a bipartite graph

Testing Bipartiteness

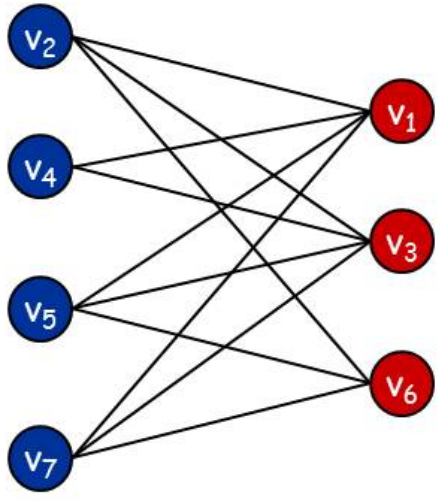
- *Testing bipartiteness*. Given a graph G, is it bipartite?
- Many graph problems become:

  easier if the underlying graph is bipartite (matching).

  tractable if the underlying graph is bipartite (independent set).



a bipartite graph G



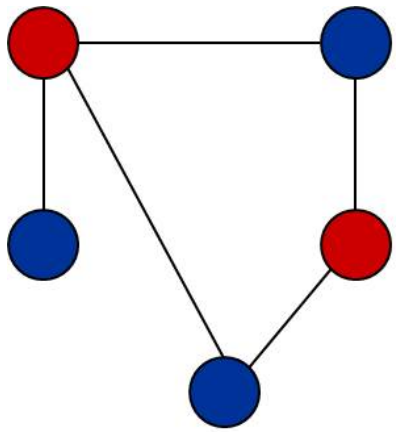
another drawing of G

An Obstruction to Bipartiteness

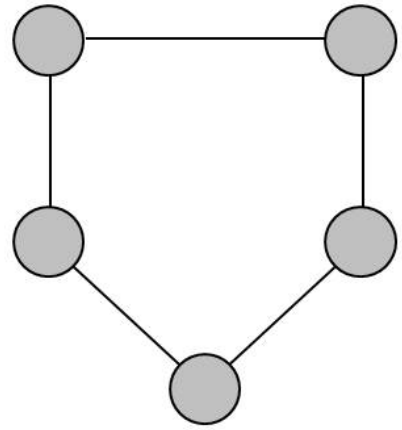
Lemma 4

If a graph G is bipartite, it cannot contain an odd length cycle.

Pf. Not possible to 2-color the odd cycle.



bipartite
(2-colorable)



not bipartite
(not 2-colorable)

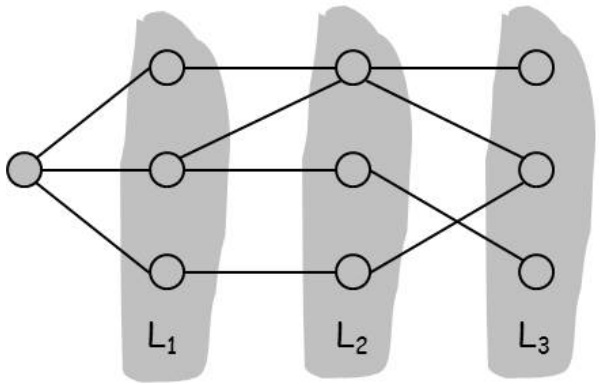
Bipartite Graphs

Lemma 5

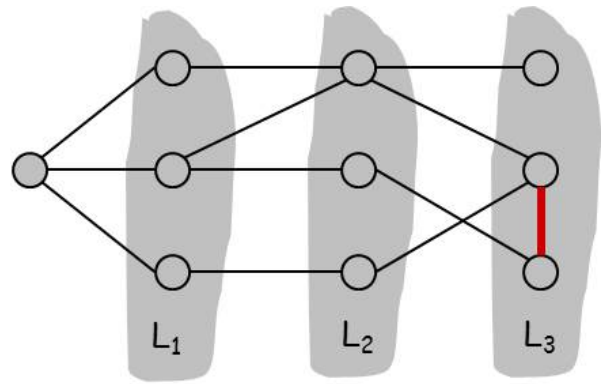
Let G be a connected graph, and let $L_0, \cdots, L_k$ be the layers produced by BFS starting at node s. Exactly one of the following holds.

1. *No edge of G joins two nodes of the same layer, and G is bipartite.*

2. *An edge of G joins two nodes of the same layer, and G contains an odd-length cycle (and hence is not bipartite).*



Case (i)

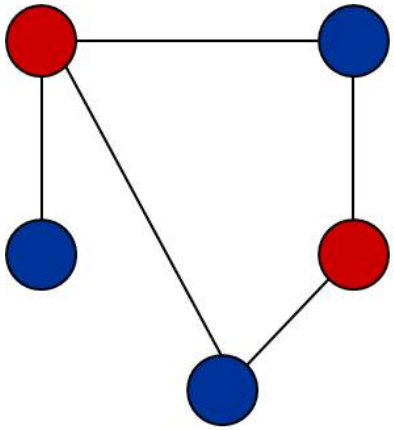


Case (ii)

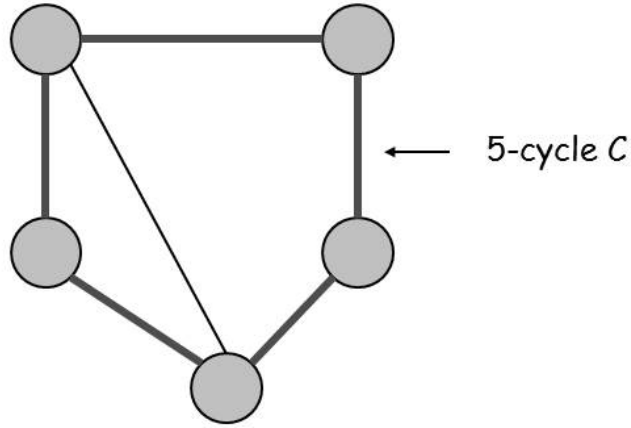
The Only Obstruction to Bipartiteness

Corollary 6

A graph G is bipartite iff it contain no odd length cycle.



bipartite
(2-colorable)



not bipartite
(not 2-colorable)

Homework

- Read Chapter 3 of the textbook.
- Exercises 5 & 6 in Chapter 3.