CS 445: Combinatorics

Shuai Li

John Hopcroft Center, Shanghai Jiao Tong University

https://shuaili8.github.io

https://shuaili8.github.io/Teaching/CS445/index.html

Basics

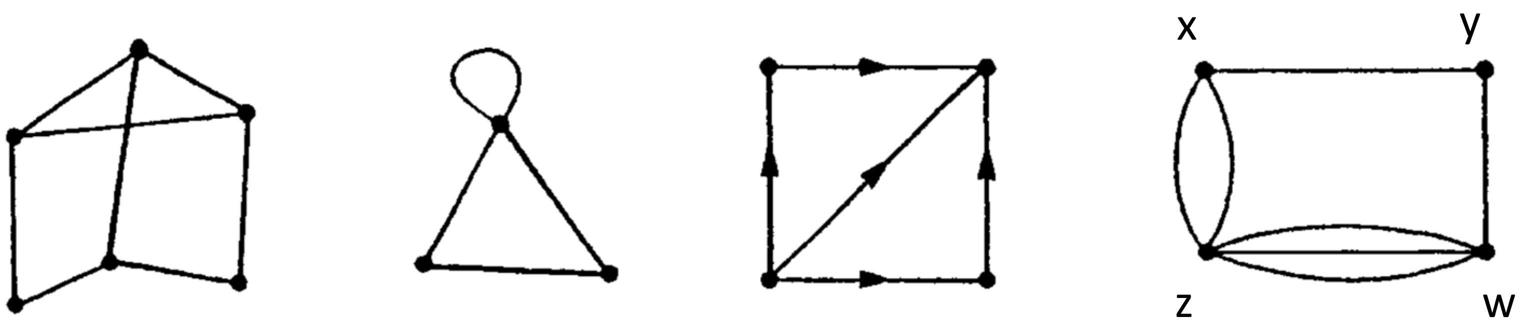
Graphs

We mainly focus on
Simple graph:
No loops, no multi-edges

- Definition A graph G is a pair (V, E)
  - V: set of vertices
  - E: set of edges
  - $e \in E$ corresponds to a pair of endpoints $x, y \in V$

edge	ends
a	x, z
b	y, w
c	x, z
d	z, w
e	z, w
f	x, y
g	z, w

Figure 1.1

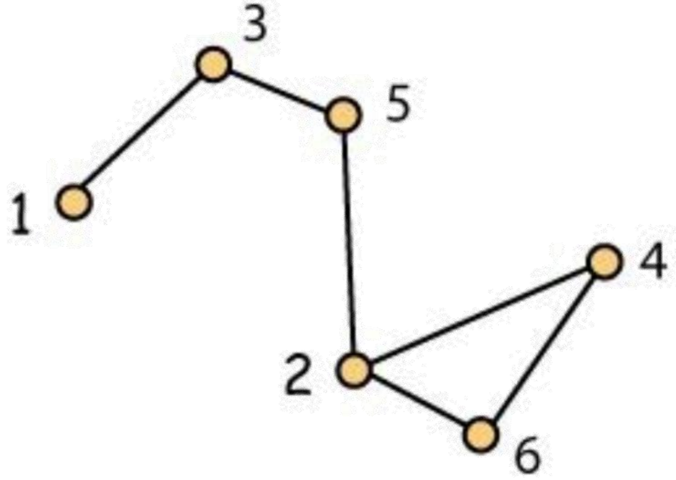


(i) graph (ii) graph with loop (iii) digraph (iv) multiple edges

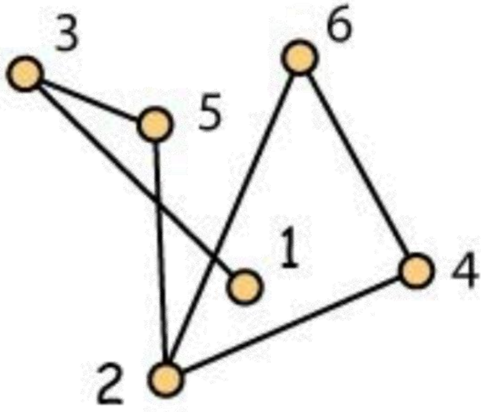
Figure 1.2

Graphs: All about adjacency

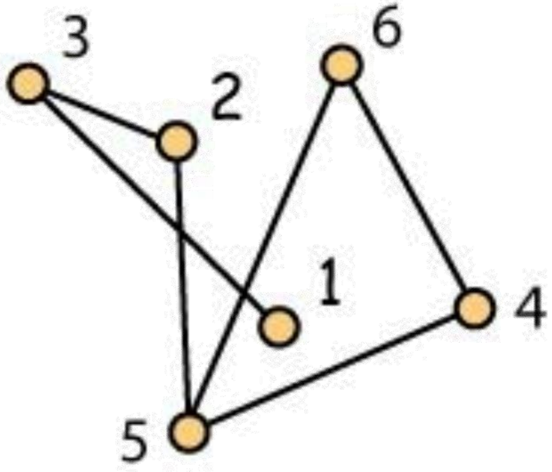
- Same graph or not



(a)



(b)



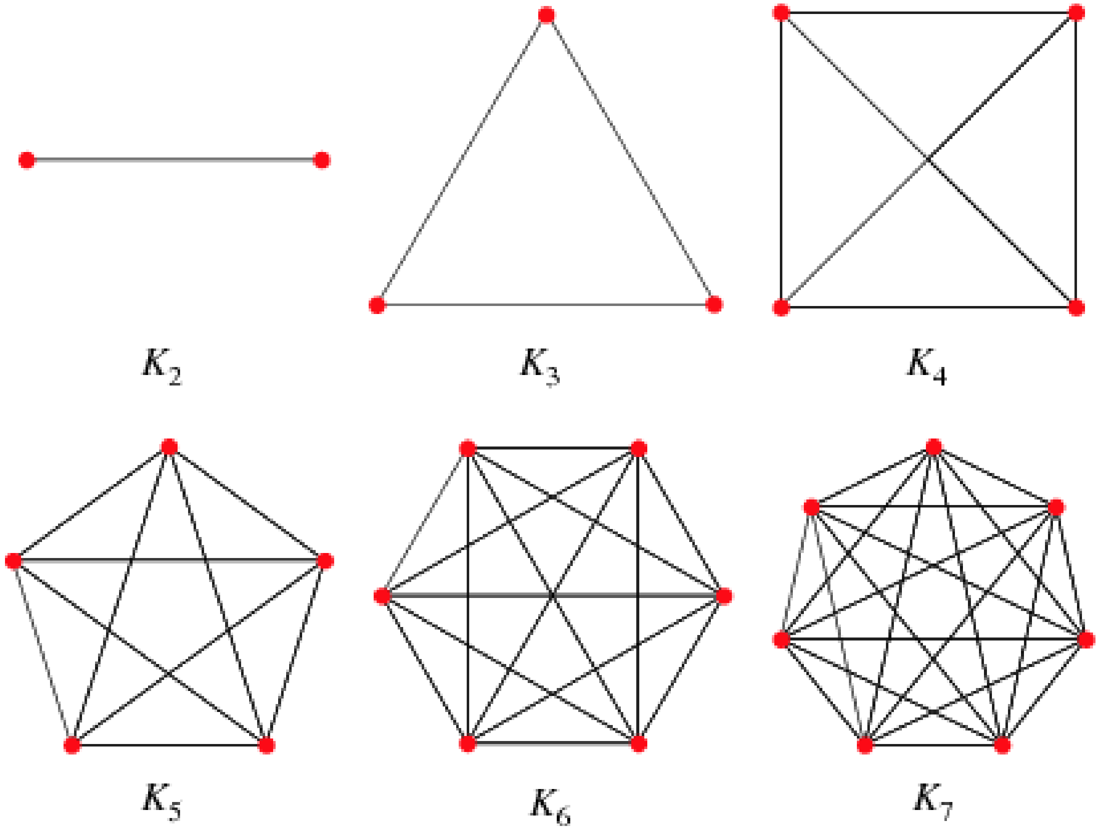
(c)

- Two graphs $G_1 = (V_1, E_1), G_1 = (V_2, E_2)$ are isomorphic if there is a bijection $f: V_1 \to V_2$ s.t.

$$e = \{a, b\} \in E_1 \Leftrightarrow f(e) := \{f(a), f(b)\} \in E_2$$

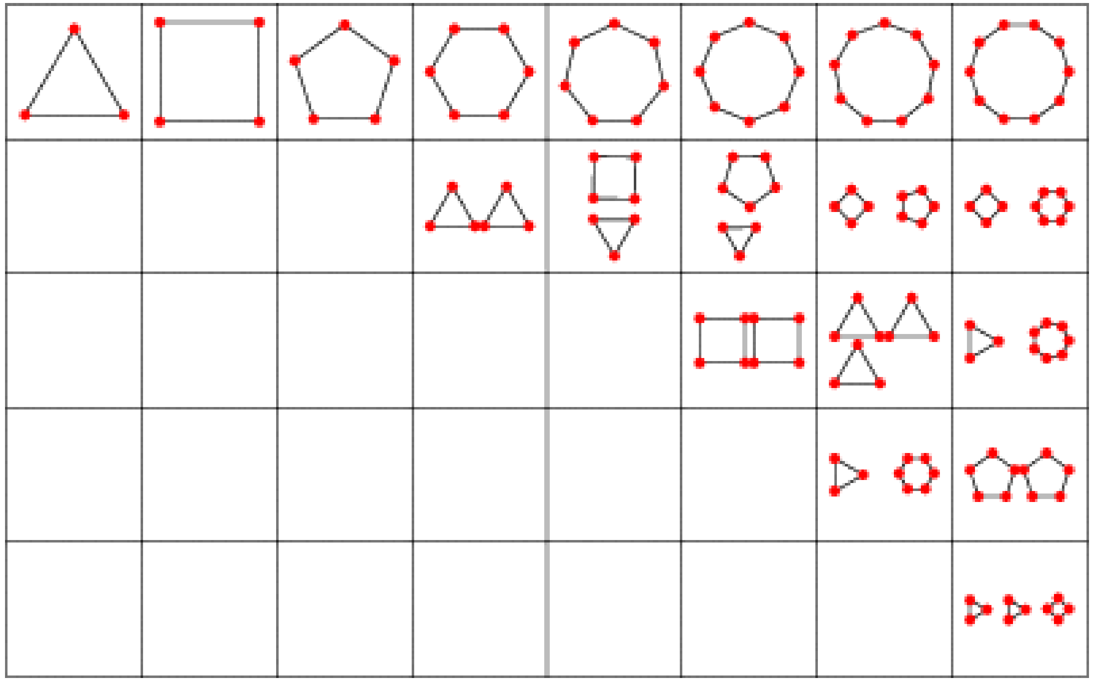
Example: Complete graphs

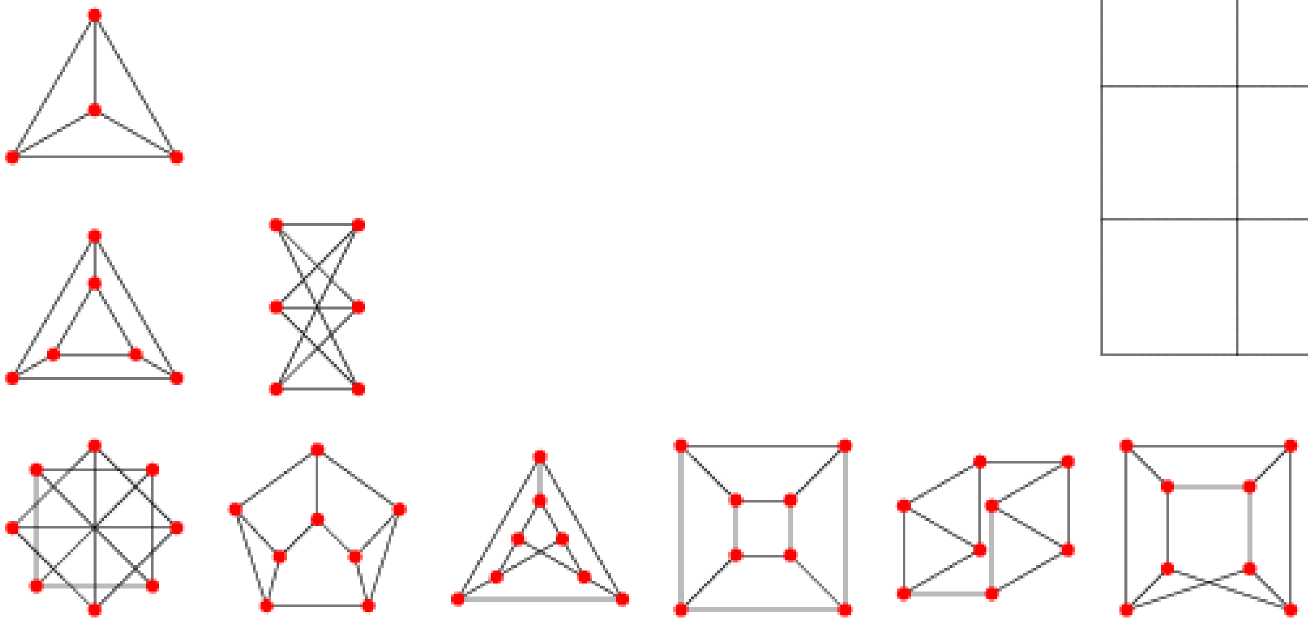
- There is an edge between every pair of vertices



Example: Regular graphs

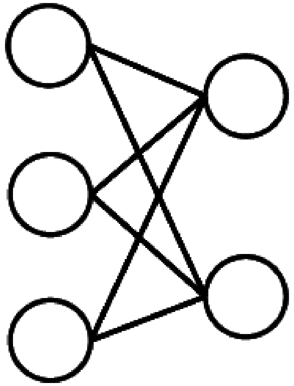
- Every vertex has the same degree



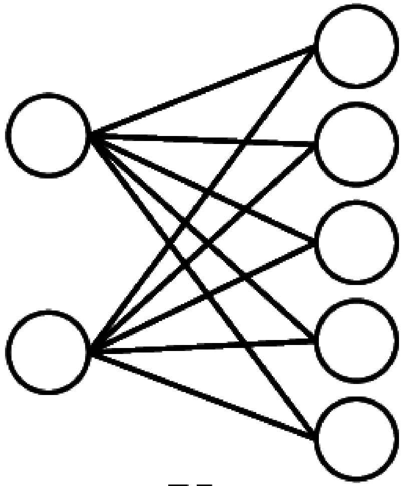


Example: Bipartite graphs

- The vertex set can be partitioned into two sets X and Y such that every edge in G has one end vertex in X and the other in Y
- Complete bipartite graphs



$\mathbf{K_{3,2}}$



$\mathbf{K_{2,5}}$

Example (1A, L): Peterson graph

- Show that the following two graphs are same/isomorphic

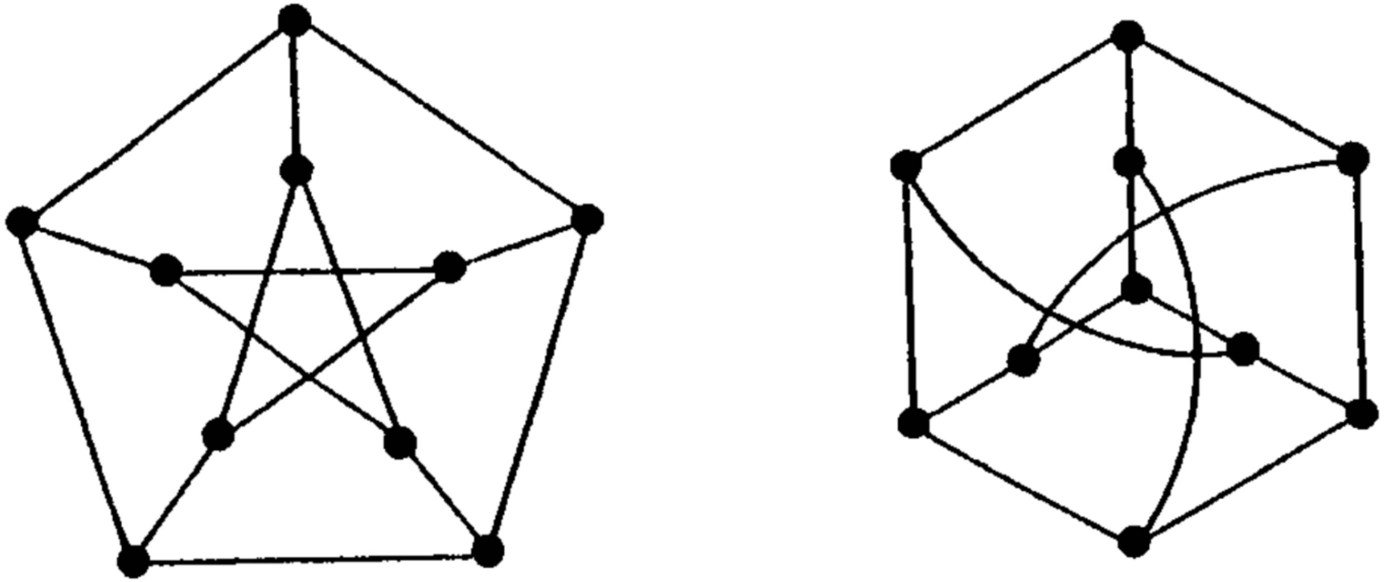
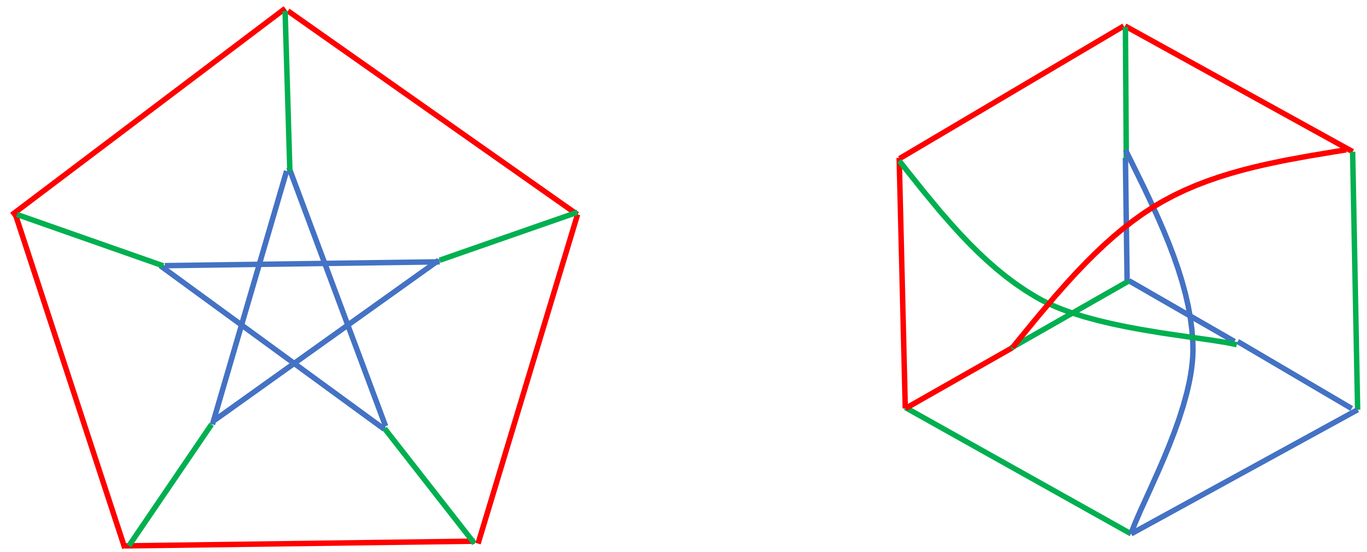


Figure 1.4

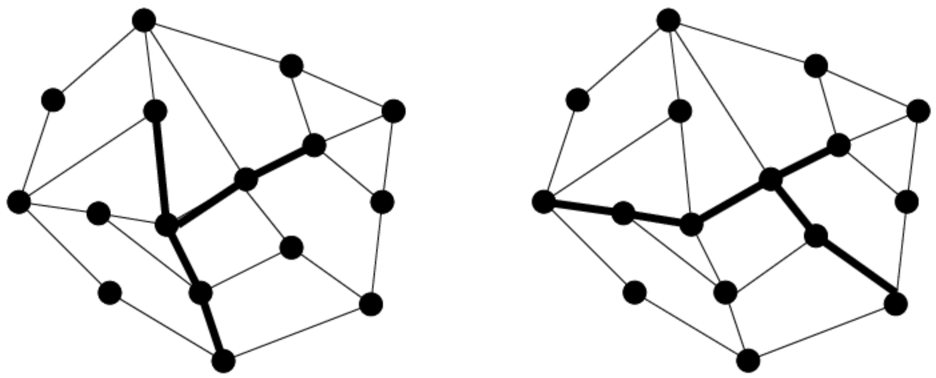
Example: Peterson graph (cont.)

- Show that the following two graphs are same/isomorphic

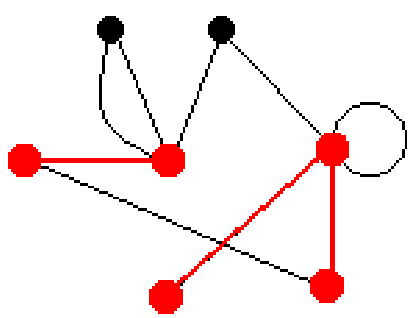


Subgraphs

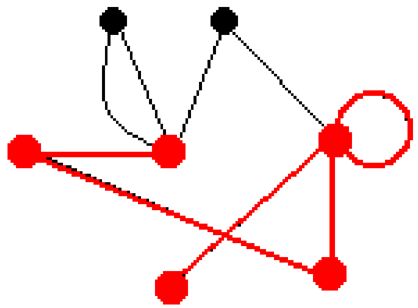
- A subgraph of a graph G is a graph H such that
$$V(H) \subseteq V(G), E(H) \subseteq E(G)$$
and the ends of an edge $e \in E(H)$ are the same as its ends in G
  - H is a spanning subgraph when $V(H) = V(G)$
  - The subgraph of G induced by a subset $S \subseteq V(G)$ is the subgraph whose vertex set is S and whose edges are all the edges of G with both ends in S



(a) (b)



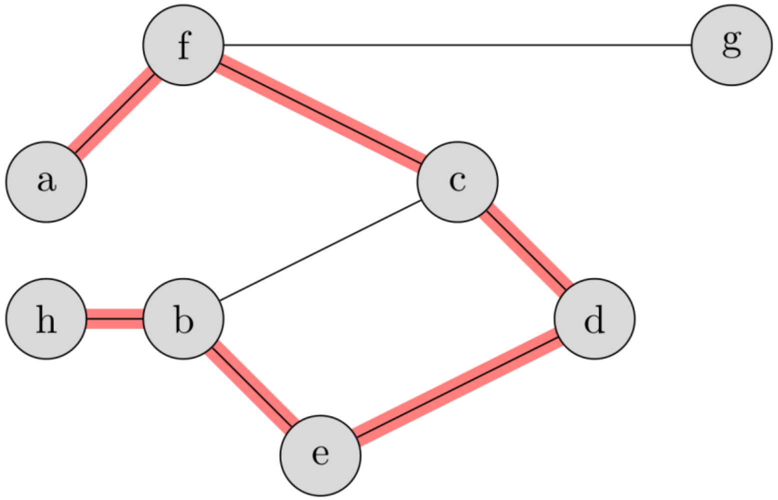
Subgraph (in red)



Induced Subgraph

Paths (路径)

- A path is a non-empty alternating sequence $v_0 e_1 v_1 e_2 \dots e_k v_k$ where vertices are all distinct
  - Or it can be written as $v_0 v_1 \dots v_k$ in simple graphs
- P^k: path of length k (the number of edges)

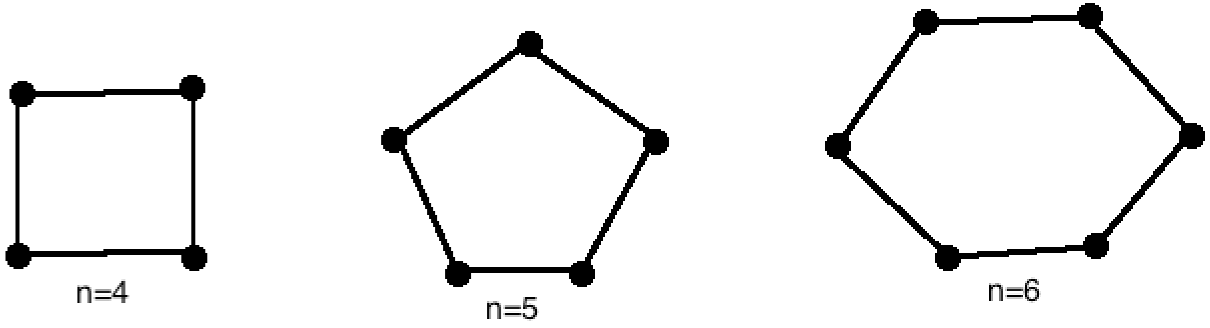


Walk (游走)

- A walk is a non-empty alternating sequence $v_0 e_1 v_1 e_2 \dots e_k v_k$
  - The vertices not necessarily distinct
  - The length = the number of edges
- Proposition (1.2.5, W) Every u-v walk contains a u-v path

Cycles (环)

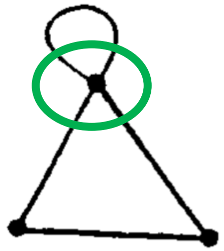
- If $P = x_0x_1 \dots x_{k-1}$ is a path and $k \geq 3$, then the graph $C \coloneqq P + x_{k-1}x_0$ is called a cycle
- C^k: cycle of length k (the number of edges/vertices)



- Proposition (1.2.15, W) Every closed odd walk contains an odd cycle

Neighbors and degree

- Two vertices $a \neq b$ are called adjacent if they are joined by an edge
  - $N(x)$: set of all vertices adjacent to x
    - neighbors of x
  - A vertex is isolated vertex if it has no neighbors
- The number of edges incident with a vertex x is called the degree of x
  - A loop contributes 2 to the degree

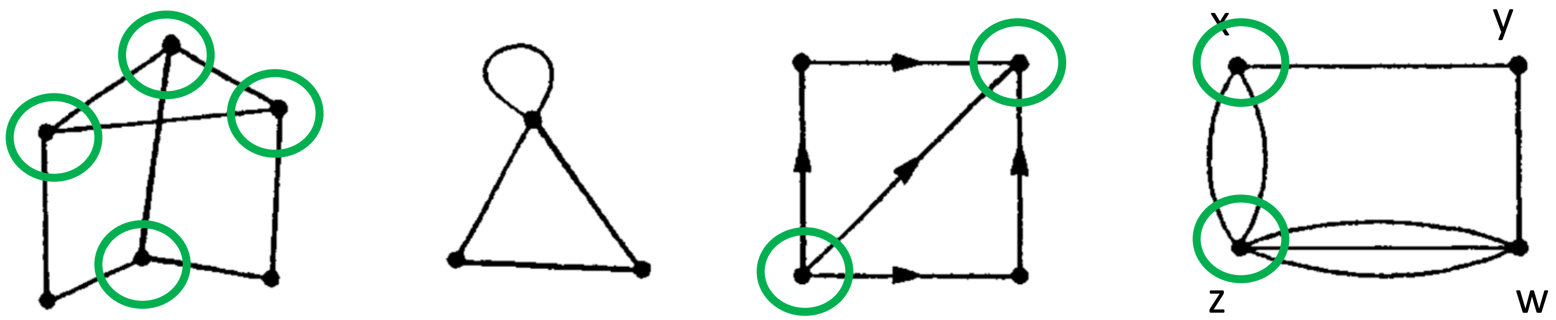


graph with loop

- A graph is finite when both $E(G)$ and $V(G)$ are finite sets

Handshaking Theorem (Euler 1736)

- Theorem A finite graph G has an even number of vertices with odd degree



(i) graph   (ii) graph with loop   (iii) digraph   (iv) multiple edges

Figure 1.2

Proof

- Theorem A finite graph G has an even number of vertices with odd degree.
- Proof The degree of x is the number of times it appears in the right column. Thus

$$\sum_{x \in V(G)} \deg(x) = 2|E(G)|$$

edge	ends
a	x, z
b	y, w
c	x, z
d	z, w
e	z, w
f	x, y
g	z, w

Figure 1.1

Degree

- Minimal degree of G: $\delta(G) = \min\{d(v) : v \in V\}$
- Maximal degree of G: $\Delta(G) = \max\{d(v) : v \in V\}$
- Average degree of G: $d(G) = \frac{1}{|V|} \sum_{v \in V} d(v) = \frac{2|E|}{|V|}$

- All measure the `density' of a graph

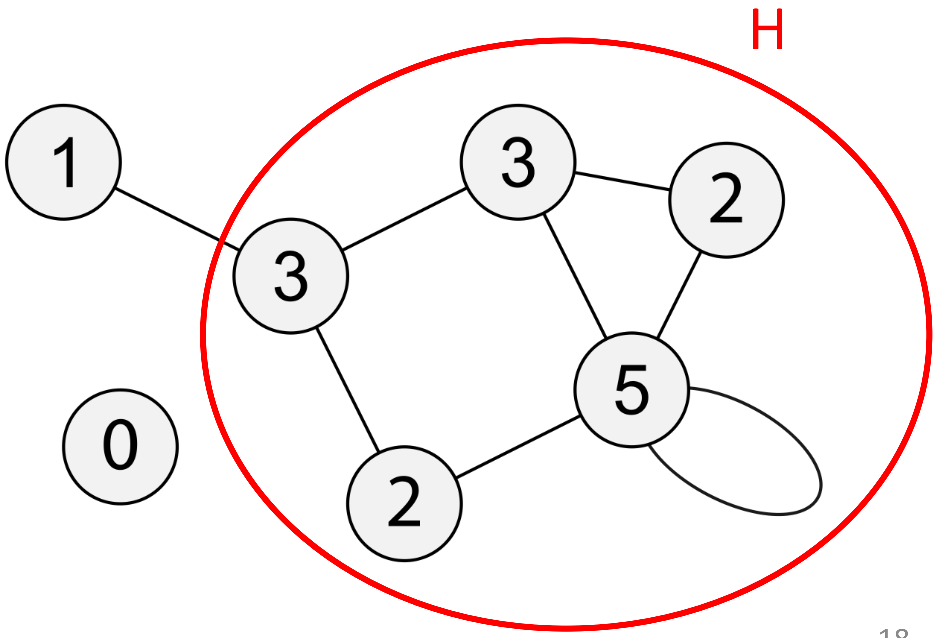
- $d(G) \geq \delta(G)$

Degree (global to local)

- Proposition (1.2.2, D) Every graph G with at least one edge has a subgraph H with

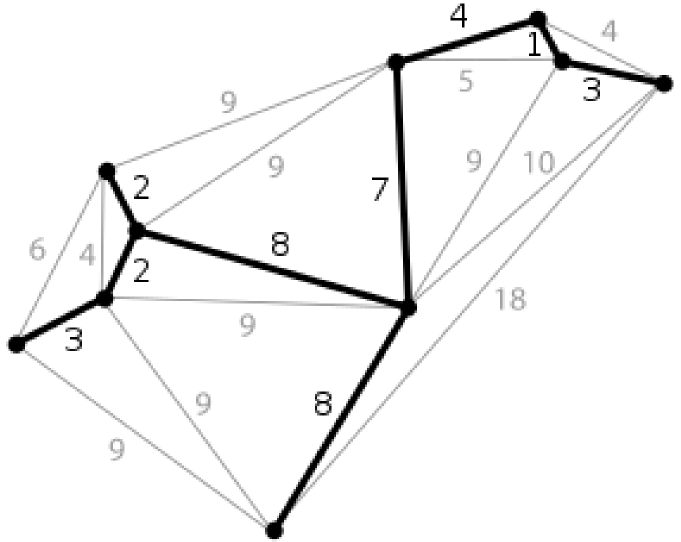
$$\delta(H) > \frac{1}{2}d(H) \geq \frac{1}{2}d(G)$$

- Example: $|G| = 7, d(G) = \frac{16}{7}$
- $\delta(H) = 2, d(H) = \frac{14}{5}$



Minimal degree guarantees long paths and cycles

- Proposition (1.3.1, D) Every graph G contains a path of length $\delta(G)$ and a cycle of length at least $\delta(G) + 1$, provided $\delta(G) \geq 2$.

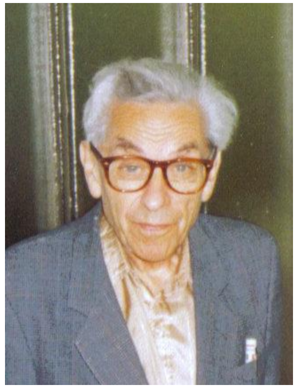


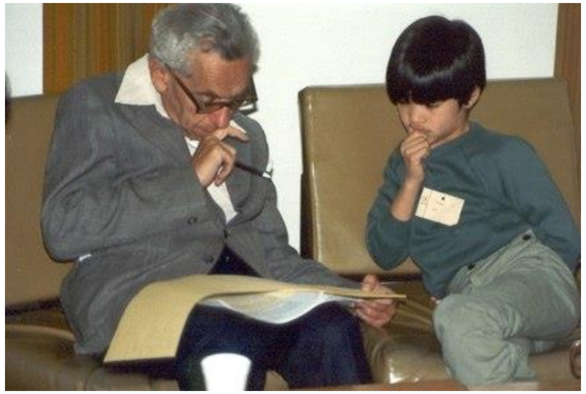
Distance and diameter

- The distance $d_G(x, y)$ in G of two vertices x, y is the length of a shortest $x{\sim}y$ path
  - if no such path exists, we set $d(x, y) := \infty$
- The greatest distance between any two vertices in G is the diameter of G

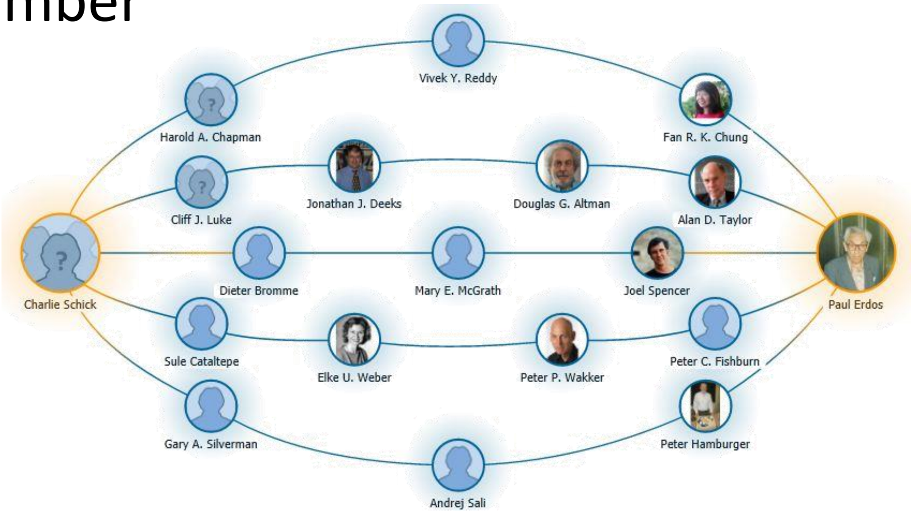
$$\operatorname{diam}(G) = \max_{x,y \in V} d(x, y)$$

Example -- Erdős number





- A well-known graph
  - vertices: mathematicians of the world
  - Two vertices are adjacent if and only if they have published a joint paper
  - The distance in this graph from some mathematician to the vertex Paul Erdős is known as his or her Erdős number

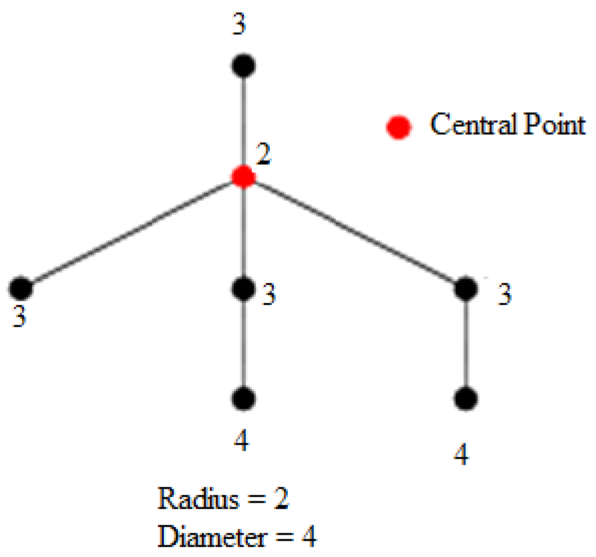


Radius and diameter

- A vertex is central in G if its greatest distance from other vertex is smallest, such greatest distance is the radius of G

$$\mathrm{rad(G)} := \min_{x \in V} \max_{y \in V} d(x, y)$$

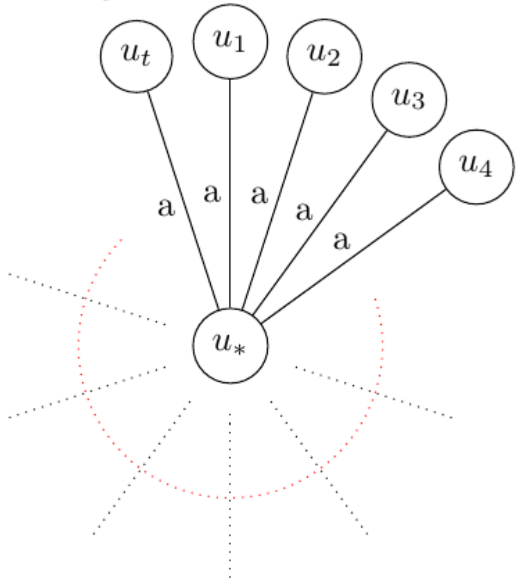
- Proposition (1.4, H; Ex1.6, D) $\mathrm{rad}(G) \le \mathrm{diam}(G) \le 2\,\mathrm{rad}(G)$



Radius and maximum degree control graph size

- Proposition (1.3.3, D) A graph G with radius at most r and maximum degree at most $\Delta \geq 3$ has fewer than $\frac{\Delta}{\Delta-2}(\Delta-1)^r$.

Figure 1: Star Graph



Lecture 2: Girth, Connectivity and Bipartite Graphs

Shuai Li

John Hopcroft Center, Shanghai Jiao Tong University

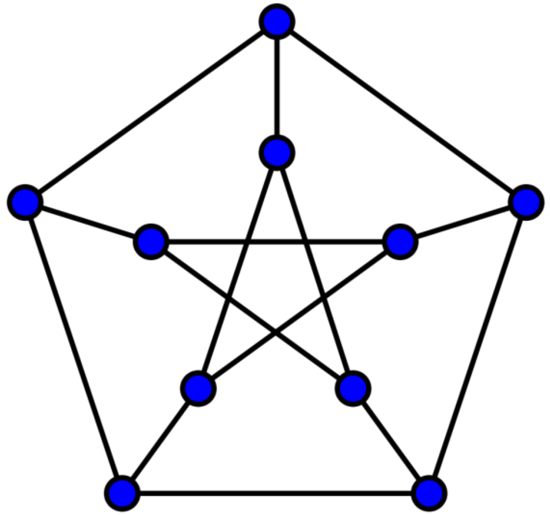
https://shuaili8.github.io

https://shuaili8.github.io/Teaching/CS445/index.html

Girth

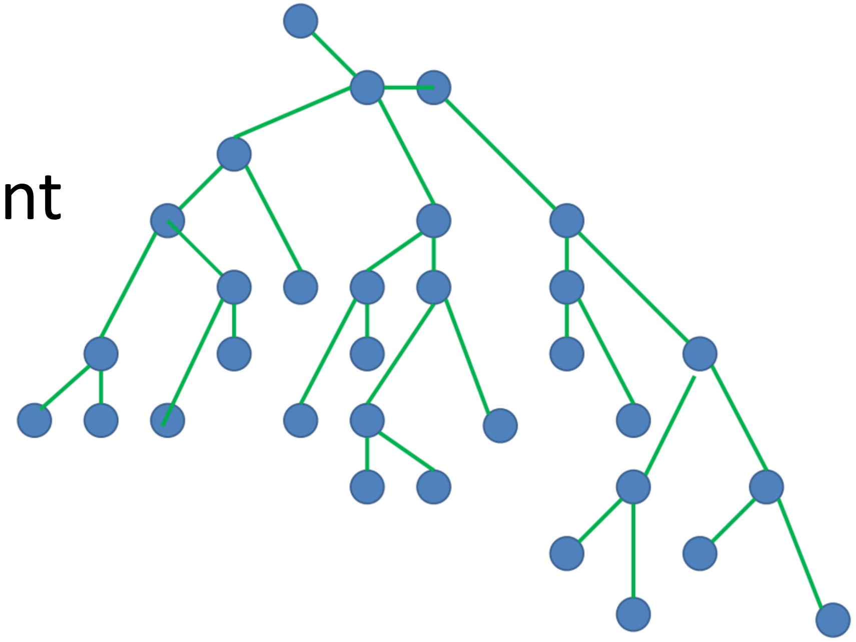
- The minimum length of a cycle in a graph G is the girth $g(G)$ of G

- Example: The Peterson graph is the unique 5-cage
  - cubic graph (every vertex has degree 3)
  - girth = 5
  - smallest graph satisfies the above properties



Girth (cont.)

- A tree has girth ∞
- Note that a tree can be colored with two different colors
- $\Longrightarrow$ A graph with large girth has small chromatic number?
- Unfortunately NO!
- Theorem (Erdős, 1959) For all k, l, there exists a graph G with $g(G) > l$ and $\chi(G) > k$



Girth and diameter

- Proposition (1.3.2, D) Every graph G containing a cycle satisfies $g(G) \leq 2\,\mathrm{diam}(G) + 1$

- When the equality holds?

Girth and minimal degree lower bounds graph size

- $n_0(\delta, g) := \begin{cases} 1 + \delta \sum_{i=0}^{r-1} (\delta - 1)^i, & \text{if } g = 2r + 1 \text{ is odd} \\ 2 \sum_{i=0}^{r-1} (\delta - 1)^i, & \text{if } g = 2r \text{ is even} \end{cases}$
- Exercise (Ex7, ch1, D) Let G be a graph. If $\delta(G) \geq \delta \geq 2$ and $g(G) \geq g$, then $|G| \geq n_0(\delta, g)$
- Corollary (1.3.5, D) If $\delta(G) \geq 3$, then $g(G) < 2 \log_2 |G|$

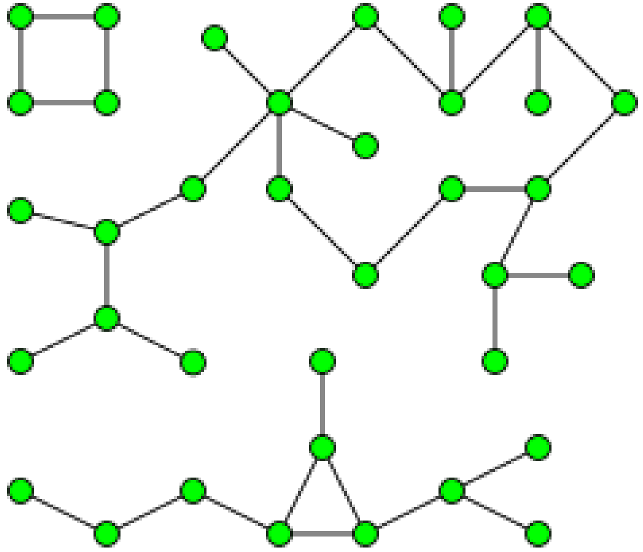
Triangle-free upper bounds # of edges

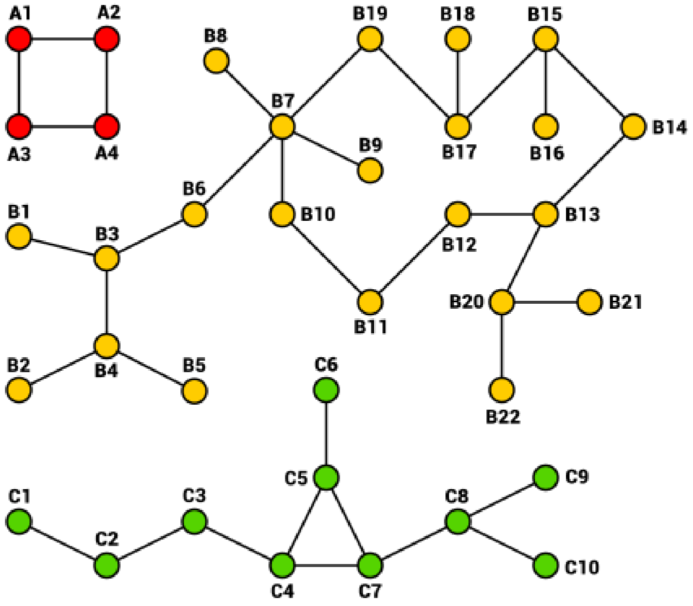
- Theorem (1.3.23, W, Mantel 1907) The maximum number of edges in an n-vertex triangle-free simple graph is $\lfloor n^2/4 \rfloor$

- The bound is best possible
- There is a triangle-free graph with $\lfloor n^2/4 \rfloor$ edges: $K_{\lfloor n/2 \rfloor, \lceil n/2 \rceil}$
- Extremal problems

Connected, connected component

- A graph G is connected if $G \neq \emptyset$ and any two of its vertices are linked by a path
- A maximal connected subgraph of G is a (connected) component





Quiz

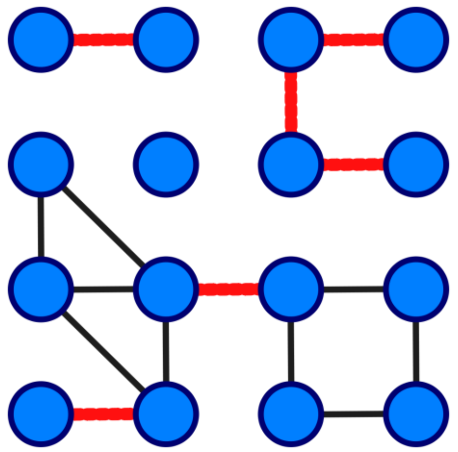
- Problem (1B, L) Suppose G is a graph on 10 vertices that is not connected. Prove that G has at most 36 edges. Can equality occur?
- More general (Ex9, S1.1.2, H) Let G be a graph of order n that is not connected. What is the maximum size of G?

Connected vs. minimal degree

- Proposition (1.3.15, W) If $\delta(G) \geq \frac{n-1}{2}$, then G is connected
- (Ex16, S1.1.2, H; 1.3.16, W)
  If $\delta(G) \geq \frac{n-2}{2}$, then G need not be connected

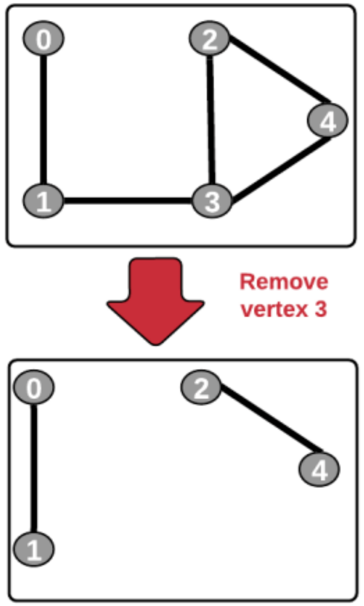
- Extremal problems
- “best possible” “sharp”

Add/delete an edge



- Components are pairwise disjoint; no two share a vertex
- Adding an edge decreases the number of components by 0 or 1
  - $\Rightarrow$ deleting an edge increases the number of components by 0 or 1
- Proposition (1.2.11, W)
  Every graph with n vertices and k edges has at least $n - k$ components
- An edge e is called a bridge if the graph $G - e$ has more components
- Proposition (1.2.14, W)
  An edge e is a bridge $\Leftrightarrow$ e lies on no cycle of G
  - Or equivalently, an edge e is not a bridge $\Leftrightarrow$ e lies on a cycle of G

Cut vertex and connectivity



- A node v is a cut vertex if the graph $G - v$ has more components
- A proper subset S of vertices is a vertex cut set if the graph $G - S$ is disconnected, or trivial (a graph of order 0 or 1)
- The connectivity, $\kappa(G)$, is the minimum size of a cut set of G
  - The graph is k-connected for any $k \leq \kappa(G)$

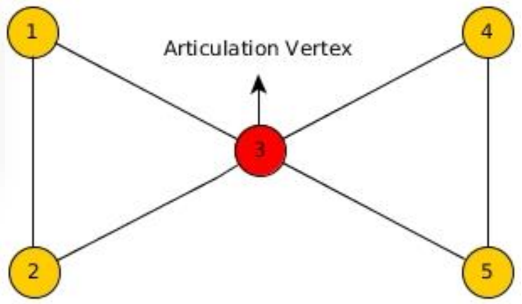
Connectivity properties

- $\kappa(K^n) = n - 1$
- If G is disconnected, $\kappa(G) = 0$
  - $\Rightarrow$ A graph is connected $\Leftrightarrow \kappa(G) \geq 1$
- If G is connected, non-complete graph of order n, then
$$1 \leq \kappa(G) \leq n - 2$$

Connectivity properties (cont.)

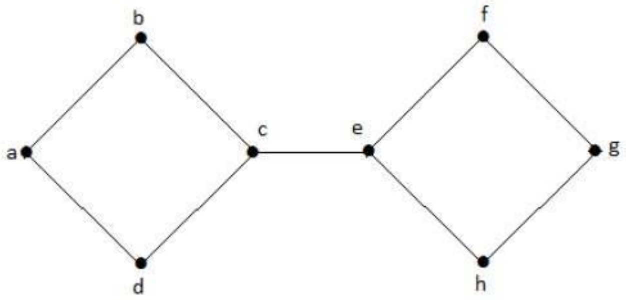
Proposition (1.2.14, W)
An edge e is a bridge $\Leftrightarrow$ e lies on no cycle of G

- Or equivalently, an edge e is not a bridge $\Leftrightarrow$ e lies on a cycle of G



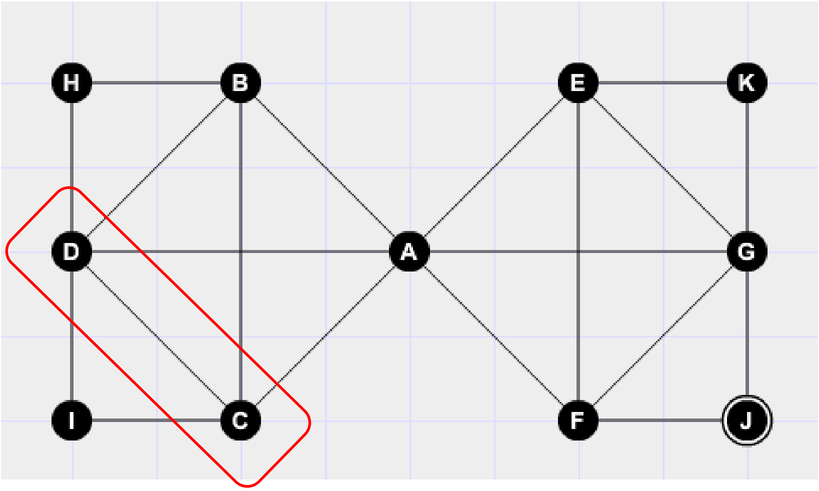
- $\kappa(G) \geq 2 \Leftrightarrow G$ is connected and has no cut vertices
- A vertex lies on a cycle $\not\Rightarrow$ it is not a cut vertex
  - $\Rightarrow$ (Ex13, S1.1.2, H) Every vertex of a connected graph G lies on at least one cycle $\not\Rightarrow \kappa(G) \geq 2$
  - (Ex14, S1.1.2, H) $\kappa(G) \geq 2$ implies G has at least one cycle

- (Ex12, S1.1.2, H) G has a cut vertex vs. G has a bridge



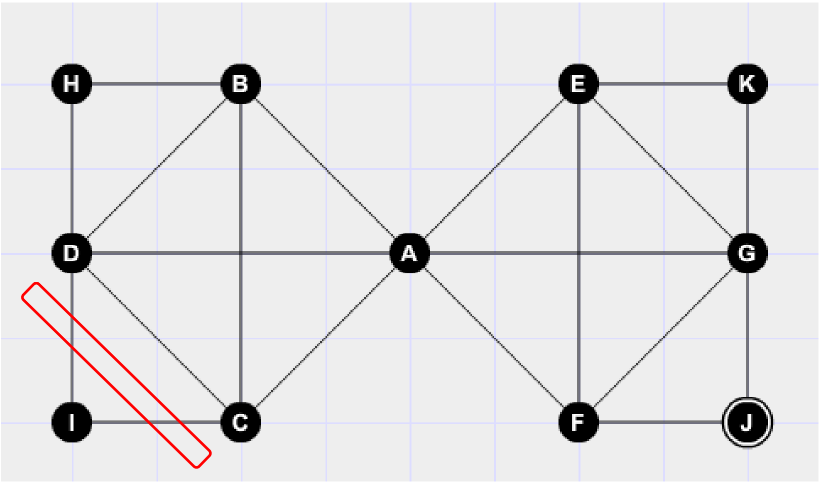
Connectivity and minimal degree

- (Ex15, S1.1.2, H)
- $\kappa(G) \leq \delta(G)$
- If $\delta(G) \geq n-2$, then $\kappa(G) = \delta(G)$



Edge-connectivity

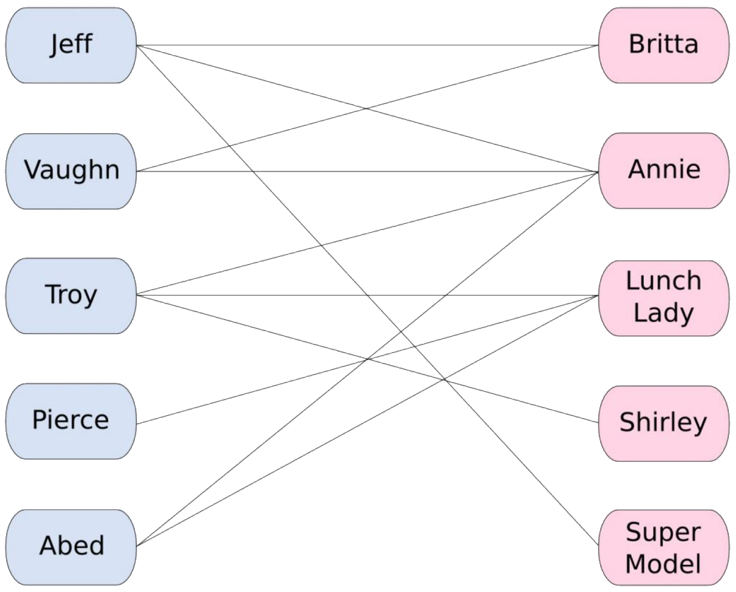
- A proper subset $F \subset E$ is edge cut set if the graph $G - F$ is disconnected
- The edge-connectivity $\lambda(G)$ is the minimal size of edge cut set
- $\lambda(G) = 0$ if G is disconnected
- Proposition (1.4.2, D) If G is non-trivial, then $\kappa(G) \leq \lambda(G) \leq \delta(G)$



Large average (minimal) degree implies local large connectivity

- Theorem (1.4.3, D, Mader 1972) Every graph G with $d(G) \geq 4k$ has a $(k+1)$-connected subgraph H such that $d(H) > d(G) - 2k$.

Bipartite graphs

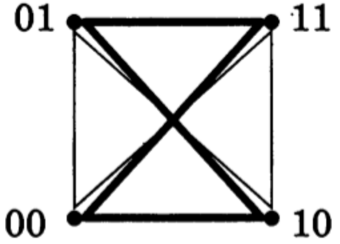


- Theorem (1.2.18, W, Kőnig 1936)
  A graph is bipartite $\Longleftrightarrow$ it contains no odd cycle

Proposition (1.2.15, W) Every closed odd walk contains an odd cycle

Complete graph is a union of bipartite graphs

- The union of graphs $G_1, \ldots, G_k$, written $G_1 \cup \cdots \cup G_k$, is the graph with vertex set $\bigcup_{i=1}^{k} V(G_i)$ and edge set $\bigcup_{i=1}^{k} E(G_i)$
- Consider an air traffic system with k airlines
  - Each pair of cities has direct service from at least one airline
  - No airline can schedule a cycle through an odd number of cities
  - Then, what is the maximum number of cities in the system?



- Theorem (1.2.23, W) The complete graph K_n can be expressed as the union of k bipartite graphs $\Leftrightarrow n \leq 2^k$

Bipartite subgraph is large

- Theorem (1.3.19, W) Every loopless graph G has a bipartite subgraph with at least $|E|/2$ edges

Lecture 3: Trees

Shuai Li

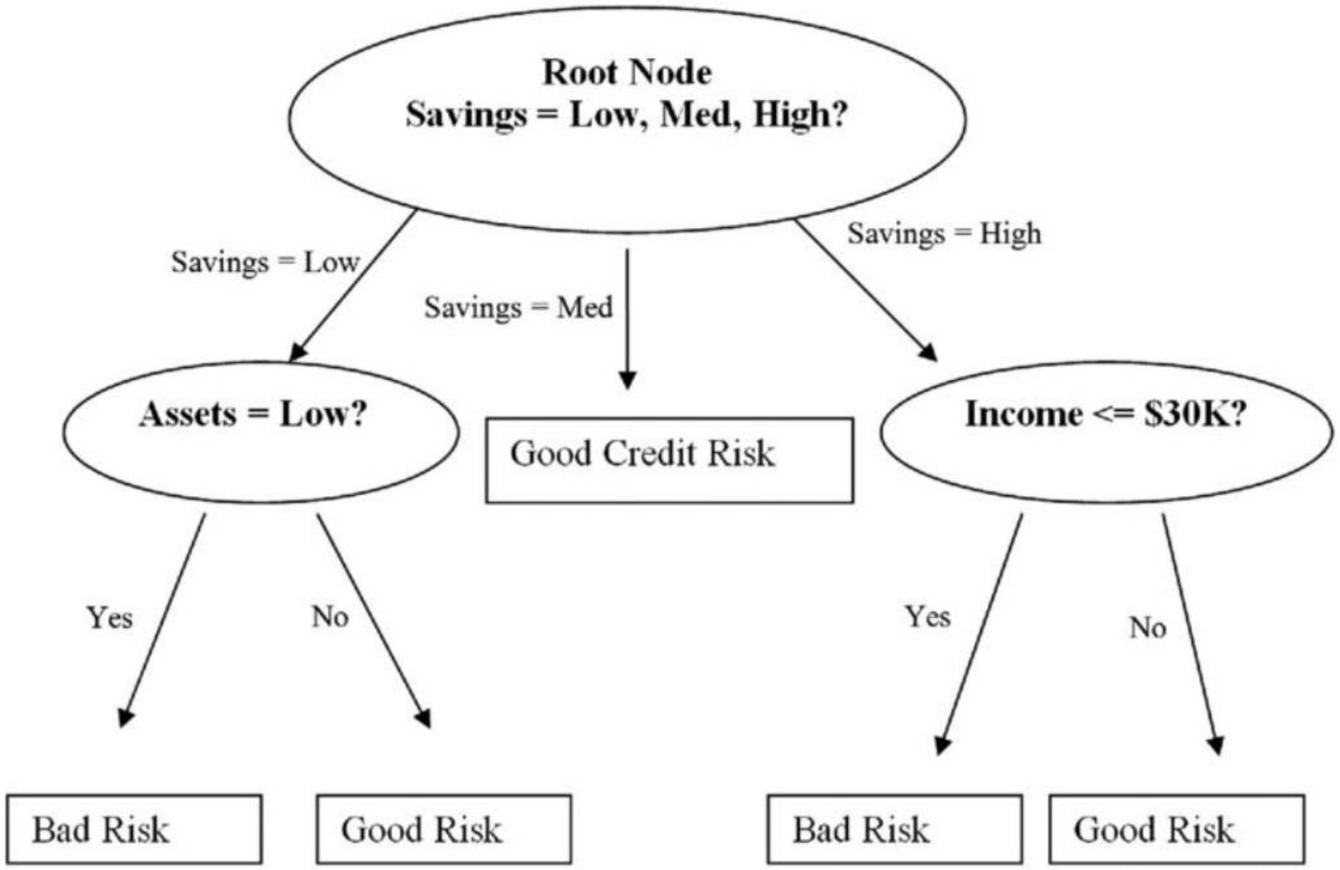
John Hopcroft Center, Shanghai Jiao Tong University

https://shuaili8.github.io

https://shuaili8.github.io/Teaching/CS445/index.html

Trees

- A tree is a connected graph T with no cycles



Properties

- Recall that **Theorem** (1.2.18, W, Kőnig 1936)
  A graph is bipartite $\Leftrightarrow$ it contains no odd cycle
- $\Rightarrow$(Ex 3, S1.3.1, H) A tree of order $n \geq 2$ is a bipartite graph

- Recall that **Proposition** (1.2.14, W)
  An edge e is a bridge $\Leftrightarrow$ e lies on no cycle of G
  - Or equivalently, an edge e is not a bridge $\Leftrightarrow$ e lies on a cycle of G
- $\Rightarrow$ Every edge in a tree is a bridge
- T is a tree $\Leftrightarrow$ T is minimally connected, i.e. T is connected but $T - e$ is disconnected for every edge $e \in T$

Equivalent definitions (Theorem 1.5.1, D)

- T is a tree of order n
  $\Leftrightarrow$ Any two vertices of T are linked by a unique path in T
  $\Leftrightarrow$ T is minimally connected
  - i.e. T is connected but $T - e$ is disconnected for every edge $e \in T$

  $\Leftrightarrow$ T is maximally acyclic
  - i.e. T contains no cycle but $T + xy$ does for any non-adjacent vertices $x, y \in T$

  $\Leftrightarrow$ (Theorem 1.10, 1.12, H) T is connected with $n - 1$ edges

  $\Leftrightarrow$ (Theorem 1.13, H) T is acyclic with $n - 1$ edges

Leaves of tree

- A vertex of degree 1 in a tree is called a leaf
- Theorem (1.14, H; Ex9, S1.3.2, H) Let T be a tree of order $n \geq 2$. Then T has at least two leaves
- (Ex3, S1.3.2, H) Let T be a tree with max degree Δ. Then T has at least Δ leaves
- (Ex10, S1.3.2, H) Let T be a tree of order $n \geq 2$. Then the number of leaves is

$$2 + \sum_{v:d(v)\geq 3} (d(v) - 2)$$

- (Ex8, S1.3.2, H) Every nonleaf in a tree is a cut vertex
- Every leaf node is not a cut vertex

The center of a tree is a vertex or ‘an edge’

- Theorem (1.15, H) In any tree, the center is either a single vertex or a pair of adjacent vertices

Any tree can be embedded in a ‘dense’ graph

- Theorem (1.16, H) Let T be a tree of order $k+1$ with k edges. Let G be a graph with $\delta(G) \geq k$. Then G contains T as a subgraph

Spanning tree

- Given a graph G and a subgraph T, T is a spanning tree of G if T is a tree that contains every vertex of G
- Example: A telecommunications company tries to lay cable in a new neighbourhood
- Proposition (2.1.5c, W) Every connected graph contains a spanning tree

Minimal spanning tree - Kruskal’s Algorithm

- Given: A connected, weighted graph G

1. Find an edge of minimum weight and mark it.
2. Among all of the unmarked edges that do not form a cycle with any of the marked edges, choose an edge of minimum weight and mark it
3. If the set of marked edges forms a spanning tree of G, then stop. If not, repeat step 2

Example

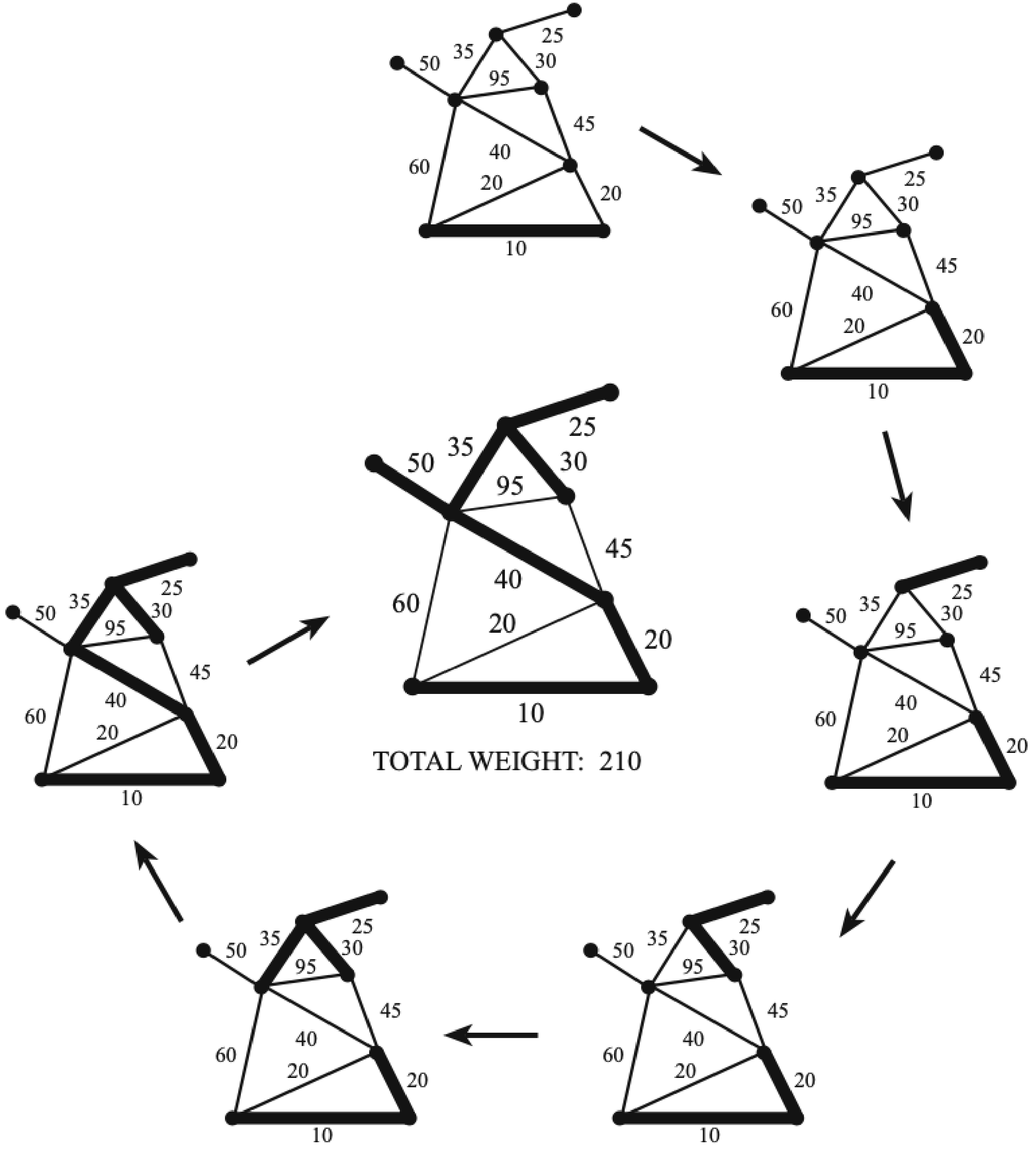


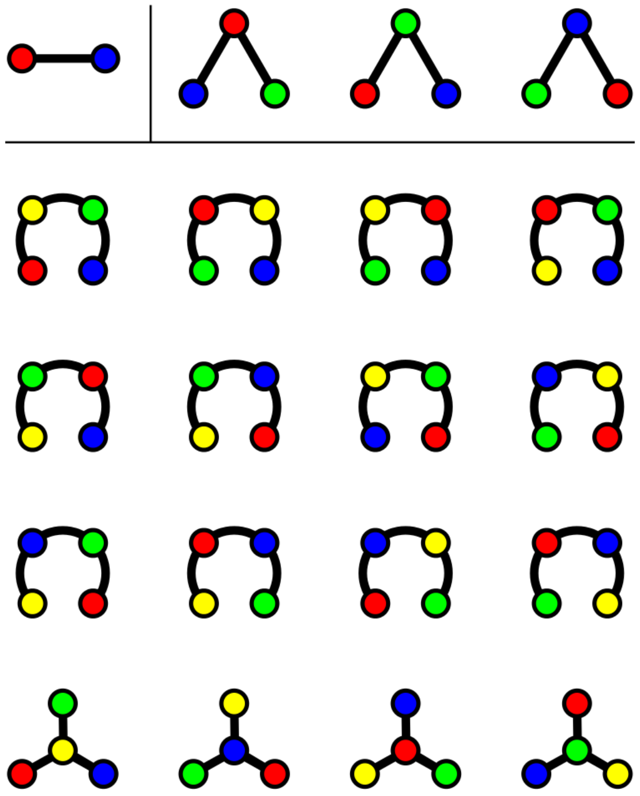
FIGURE 1.43. The stages of Kruskal's algorithm.

Theoretical guarantee of Kruskal’s algorithm

- Theorem (1.17, H) Kruskal’s algorithm produces a spanning tree of minimum total weight

Cayley's tree formula

- Theorem (1.18, H; 2.2.3, W). There are n^{n-2} distinct labeled trees of order n



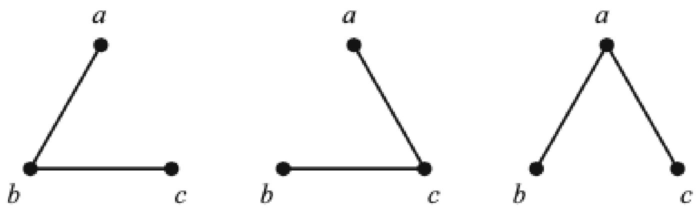


FIGURE 1.45. Labeled trees on three vertices.

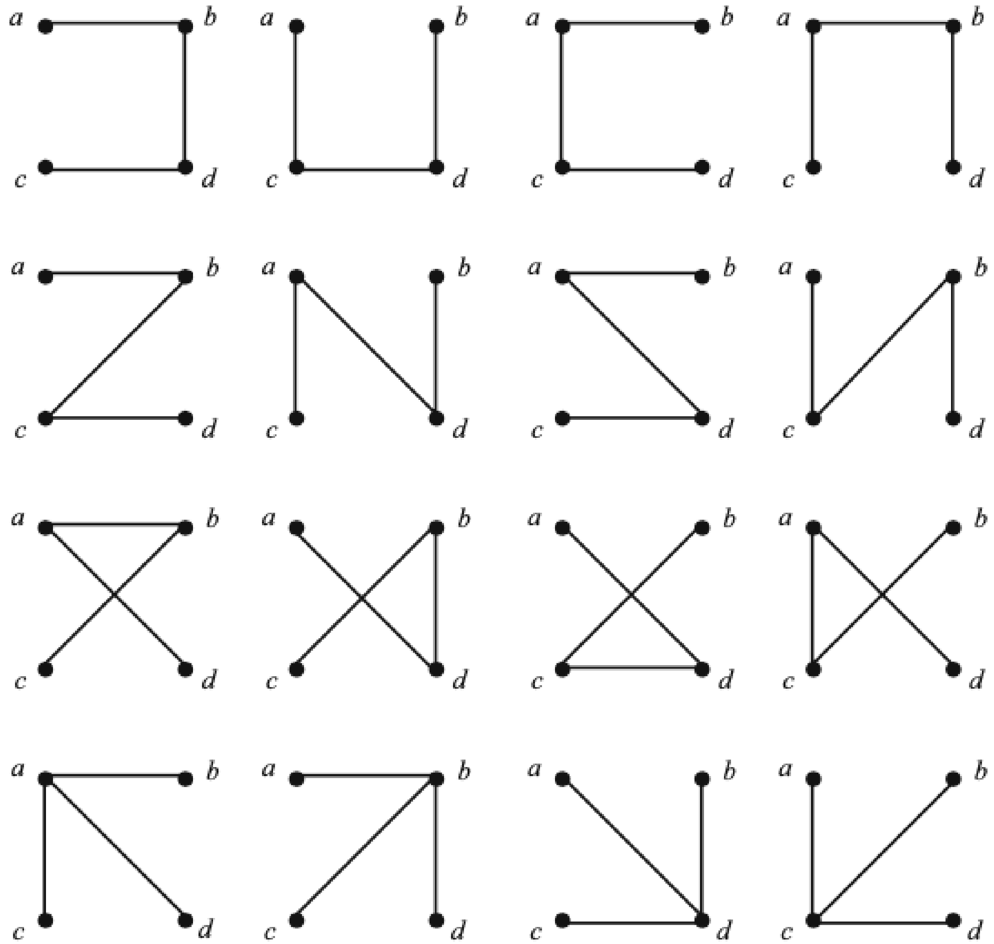


FIGURE 1.46. Labeled trees on four vertices.

Example

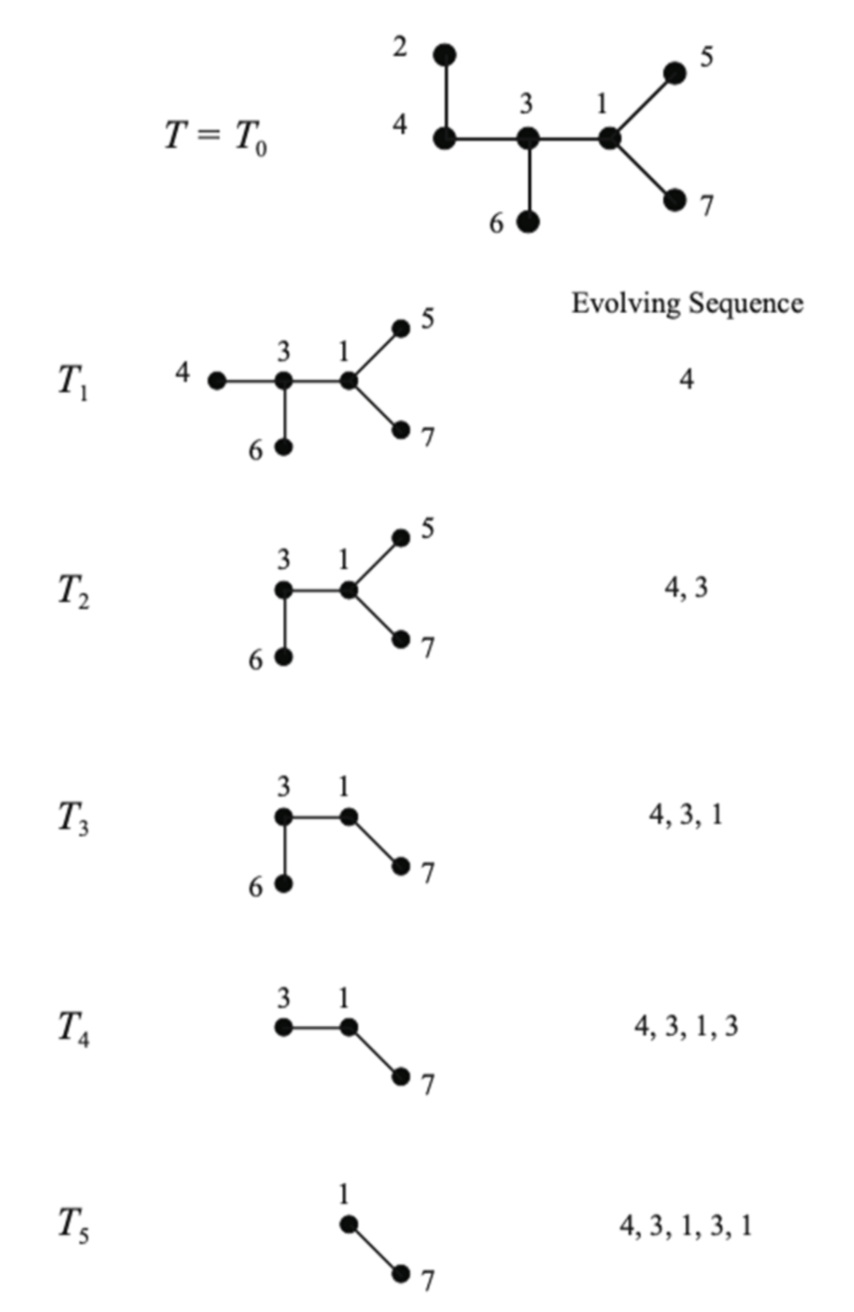


FIGURE 1.47. Creating a Prüfer sequence.

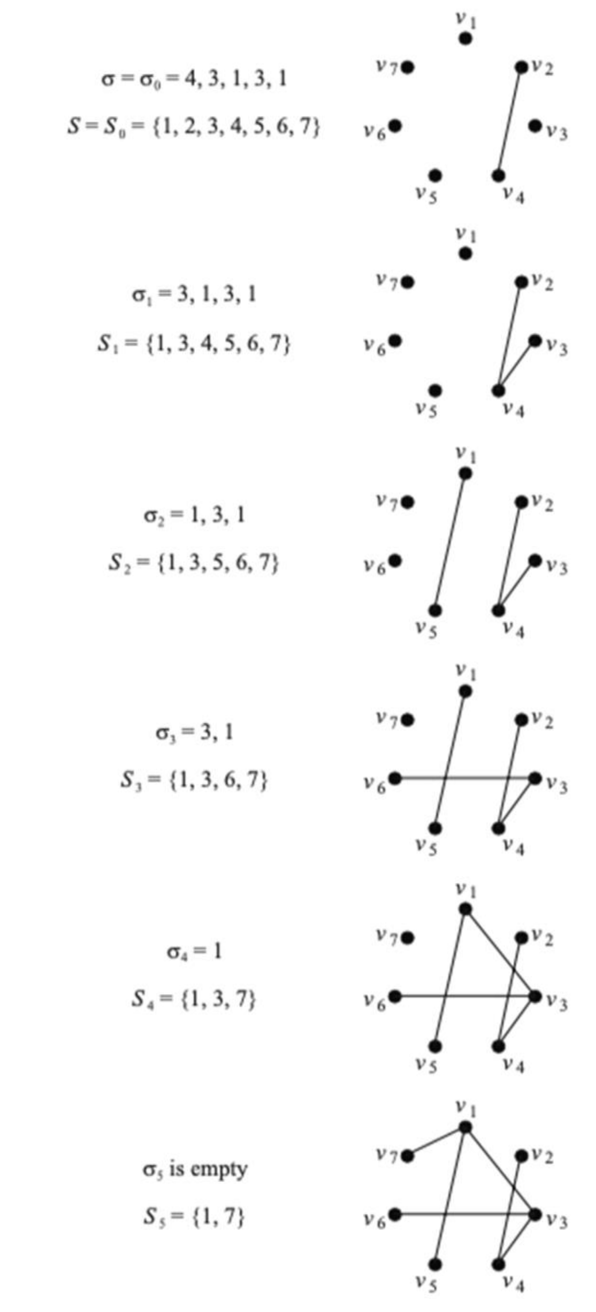
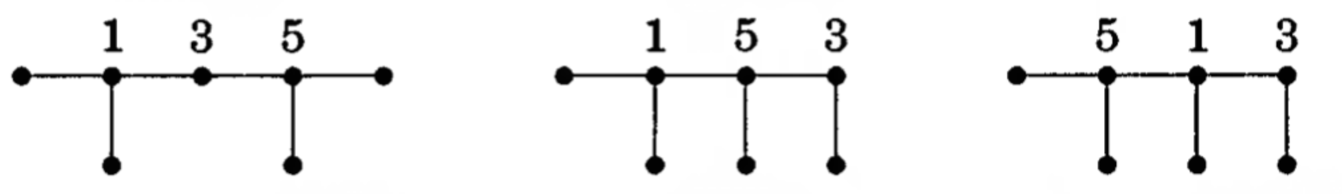


FIGURE 1.48. Building a labeled tree.

of trees with fixed degree sequence

- Corollary (2.2.4, W) Given positive integers $d_1, \ldots, d_n$ summing to $2n-2$, there are exactly $\frac{(n-2)!}{\prod(d_i-1)!}$ trees with vertex set $[n]$ such that vertex i has degree d_i for each i

- Example (2.2.5, W) Consider trees with vertices $[7]$ that have degrees $(3,1,2,1,3,1,1)$



Matrix tree theorem - cofactor

- For an $n \times n$ matrix A, the i, j cofactor of A is defined to be
$$(-1)^{i+j} \det(M_{ij})$$
where M_{ij} represents the $(n-1) \times (n-1)$ matrix formed by deleting row i and column j from A

3 × 3 generic matrix [edit]

Consider a 3×3 matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

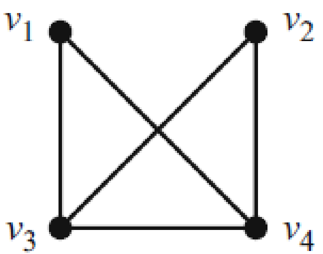
Its cofactor matrix is

$$\mathbf{C} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix},$$

Matrix tree theorem

- Theorem (1.19, H; 2.2.12, W; Kirchhoff) If G is a connected labeled graph with adjacency matrix A and degree matrix D, then the number of unique spanning trees of G is equal to the value of any cofactor of the matrix $D - A$
- If the row sums and column sums of a matrix are all 0, then the cofactors all have the same value
- Exercise Read the proof
- Exercise (Ex7, S1.3.4, H) Use the matrix tree theorem to prove Cayley's theorem

Example



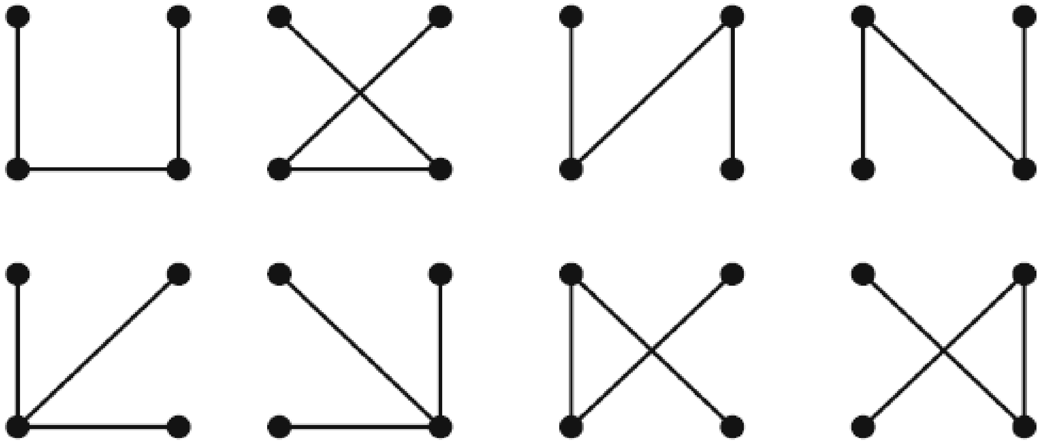


FIGURE 1.49. A labeled graph and its spanning trees.

The degree matrix D and adjacency matrix A are

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix},$$

and so

$$D - A = \begin{bmatrix} 2 & 0 & -1 & -1 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}.$$

The $(1,1)$ cofactor of $D - A$ is

$$\det \begin{bmatrix} 2 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix} = 8.$$

Score one for Kirchhoff!

- Exercise (Ex6, S1.3.4, H) Let e be an edge of K_n. Use Cayley's Theorem to prove that $K_n - e$ has $(n-2)n^{n-3}$ spanning trees

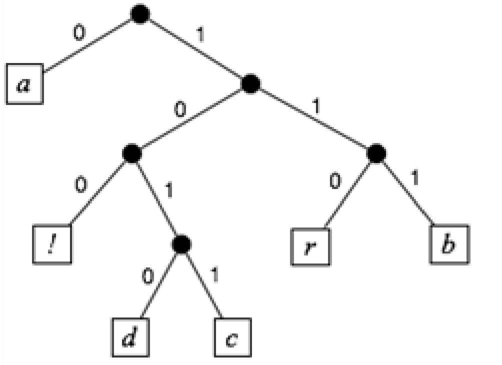
Wiener index

- In a communication network, large diameter may be acceptable if most pairs can communicate via short paths. This leads us to study the average distance instead of the maximum
- Wiener index $D(G) = \sum_{u,v \in V(G)} d_G(u, v)$
- Theorem (2.1.14, W) Among trees with n vertices, the Wiener index $D(T)$ is minimized by stars and maximized by paths, both uniquely
- Over all connected n-vertex graphs, $D(G)$ is minimized by K_n and maximized (2.1.16, W) by paths
  - (Lemma 2.1.15, W) If H is a subgraph of G, then $d_G(u, v) \leq d_H(u, v)$

Prefix coding

- A binary tree is a rooted plane tree where each vertex has at most two children
- Given large computer files and limited storage, we want to encode characters as binary lists to minimize (expected) total length
- Prefix-free coding: no code word is an initial portion of another

- Example: 11001111011



character	encoding
a	0
b	111
c	1011
d	1010
r	110
!	100

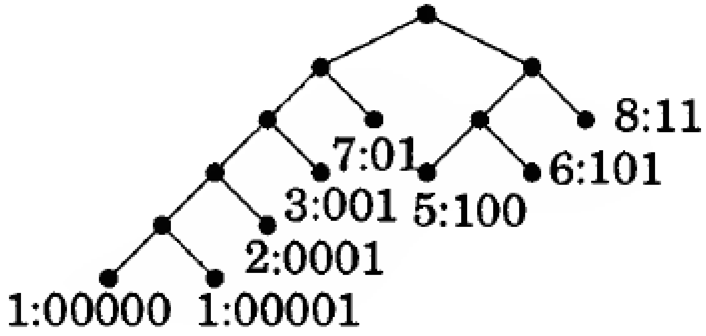
A binary prefix code for the alphabet $\{a, b, c, d, r, !\}$

Huffman's Algorithm (2.3.13, W)

- Input: Weights (frequencies or probabilities) $p_1, \dots, p_n$
- Output: Prefix-free code (equivalently, a binary tree)
- Idea: Infrequent items should have longer codes; put infrequent items deeper by combining them into parent nodes.
- Recursion: replace the two least likely items with probabilities p, p' with a single item of weight $p + p'$

Example (2.3.14, W)

a	5	100
b	1	00000
c	1	00001
d	7	01
e	8	11
f	2	0001
g	3	001
h	6	101



The average length is $\frac{5\times3+5+5+7\times2+\cdots}{33} = \frac{30}{11} < 3$

Huffman coding is optimal

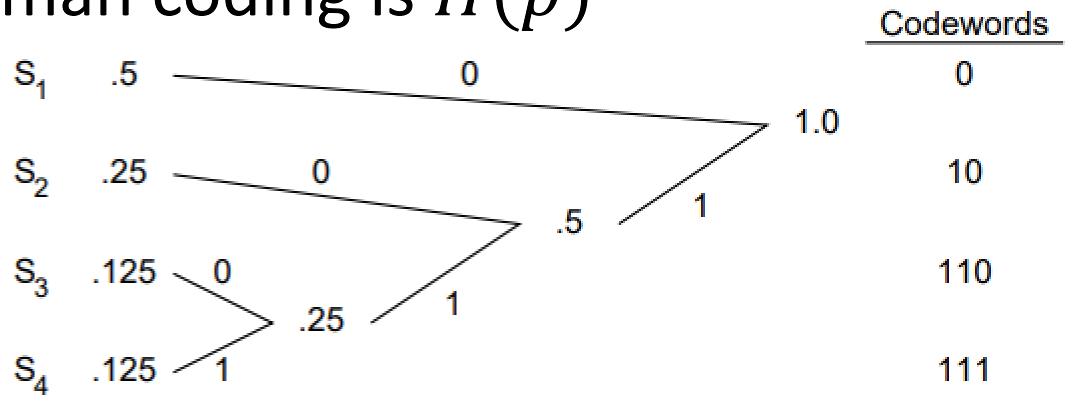
- Theorem (2.3.15, W) Given a probability distribution $\{p_i\}$ on n items, Huffman's Algorithm produces the prefix-free code with minimum expected length

Huffman coding and entropy

- The entropy of a discrete probability distribution $\{p_i\}$ is that

$$H(p) = -\sum_i p_i \log_2 p_i$$

- Exercise (Ex2.3.31, W) $H(p) \leq$ average length of Huffman coding $\leq H(p) + 1$
- Exercise (Ex2.3.30, W) When each p_i is a power of ½, average length of Huffman coding is $H(p)$



$$\begin{aligned} \text{average length} &= (1)\left(\frac{1}{2}\right) + (2)\left(\frac{1}{4}\right) + (3)\left(\frac{1}{8}\right) + (3)\left(\frac{1}{8}\right) \\ &= 1.75 \ \text{ bits/symbol} \end{aligned}$$

$$\begin{aligned} H &= \frac{1}{2}\log_2 2 + \frac{1}{4}\log_2 4 + \frac{1}{8}\log_2 8 + \frac{1}{8}\log_2 8 \\ &= \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{3}{8} \\ &= 1.75 \end{aligned}$$

Lecture 4: Circuits

Shuai Li

John Hopcroft Center, Shanghai Jiao Tong University

https://shuaili8.github.io

https://shuaili8.github.io/Teaching/CS445/index.html

Eulerian circuit

- A closed walk through a graph using every edge once is called an Eulerian circuit
- A graph that has such a walk is called an Eulerian graph

- Theorem (1.2.26, W) A graph G is Eulerian $\Leftrightarrow$ it has at most one nontrivial component and its vertices all have even degree
- (possibly with multiple edges)
- Proof "$\Longrightarrow$" That G must be connected is obvious.
  Since the path enters a vertex through some edge and leaves by another edge, it is clear that all degrees must be even

Key lemma

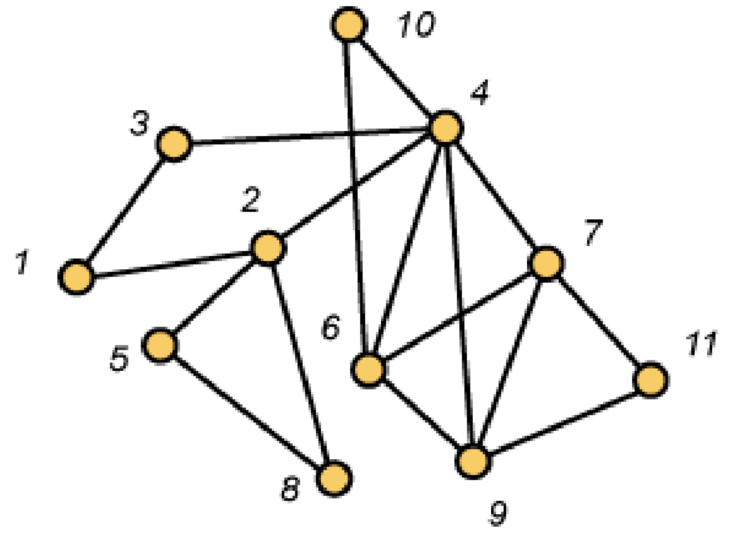
- Lemma (1.2.25, W) If every vertex of a graph G has degree at least 2, then G contains a cycle.

Proposition (1.3.1, D) Every graph G contains a path of length $\delta(G)$ and a cycle of length at least $\delta(G) + 1$, provided $\delta(G) \geq 2$.

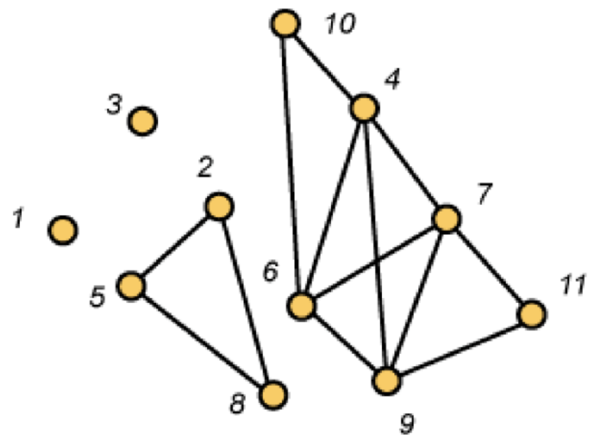
Hierholzer's Algorithm for Euler Circuits

1. Choose a root vertex r and start with the trivial partial circuit (r)
2. Given a partial circuit $(x_0, e_1, x_1, \ldots, x_{t-1}, e_t, x_t = x_0)$ that traverses not all edges of G, remove these edges from G
3. Let i be the least integer for which x_i is incident with one of the remaining edges
4. Form a greedy partial circuit among the remaining edges of the form $(x_i = y_0, e_1', y_1, \ldots, y_{s-1}, e_s', y_s = x_i)$
5. Expand the original circuit by setting $(x_0, e_1, \ldots, e_i, x_i = y_0, e_1', y_1, \ldots, y_{s-1}, e_s', y_s = x_i, e_{i+1}, \ldots, e_t, x_t = x_0)$
6. Repeat step 2-5

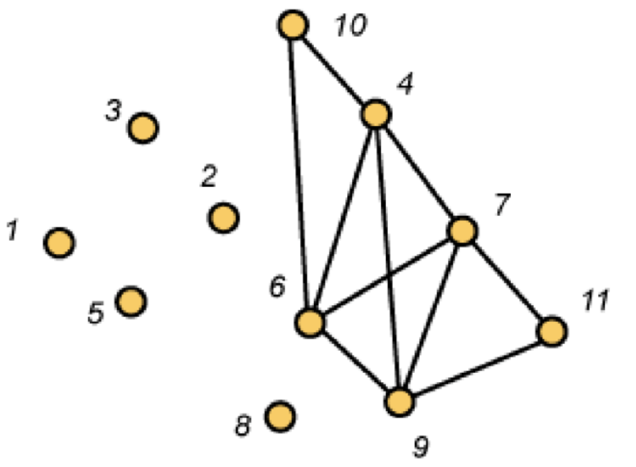
Example



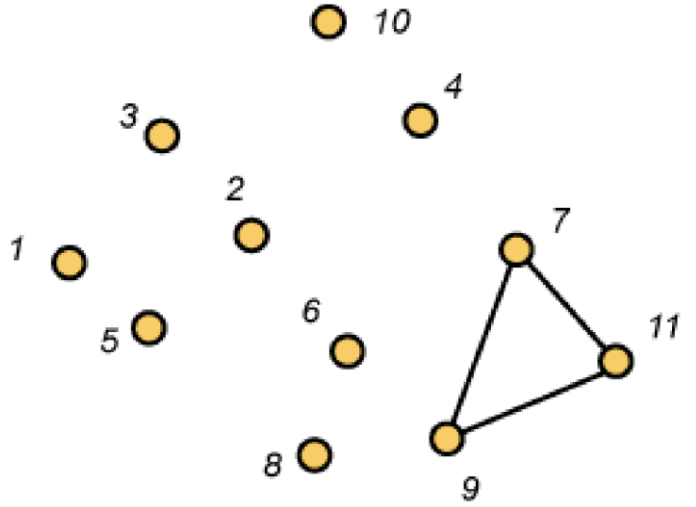
1. Start with the trivial circuit (1)
2. Greedy algorithm yields the partial circuit
   $(1,2,4,3,1)$
3. Remove these edges
4. The first vertex incident with remaining edges is 2
5. Greedy algorithms yields $(2,5,8,2)$
6. Expanding $(1,2,5,8,2,4,3,1)$
7. Remove these edges



Example (cont.)

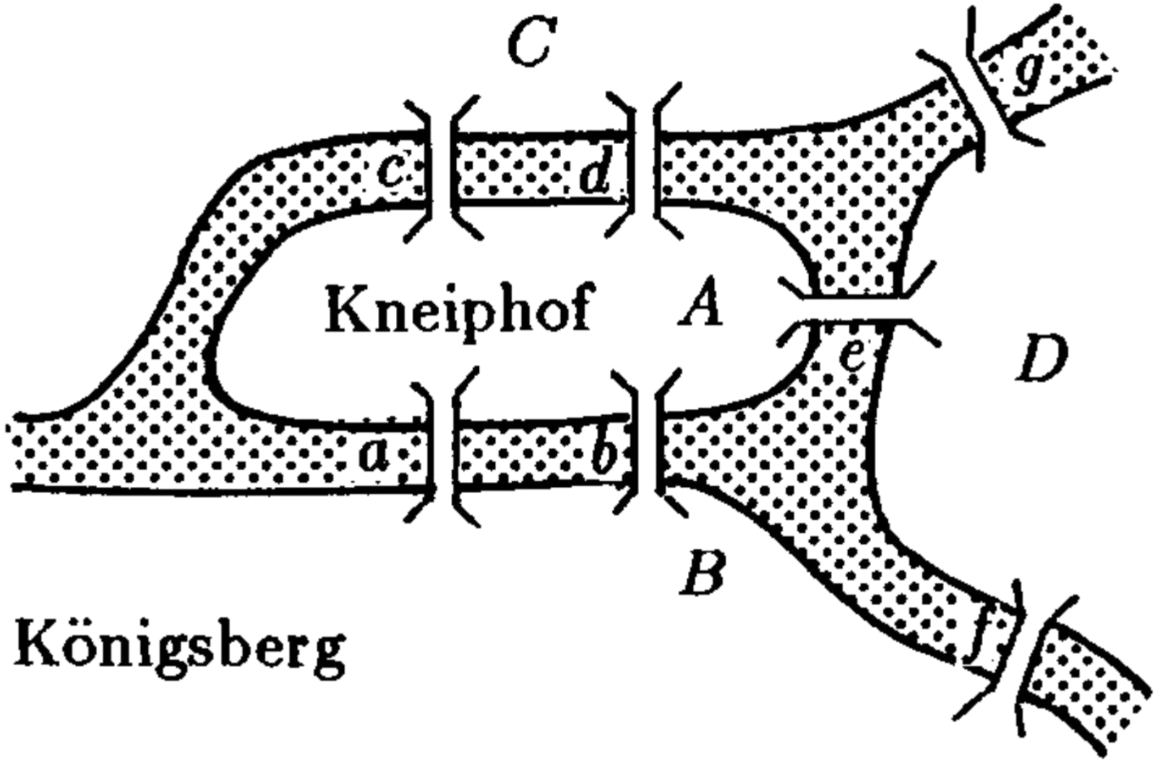


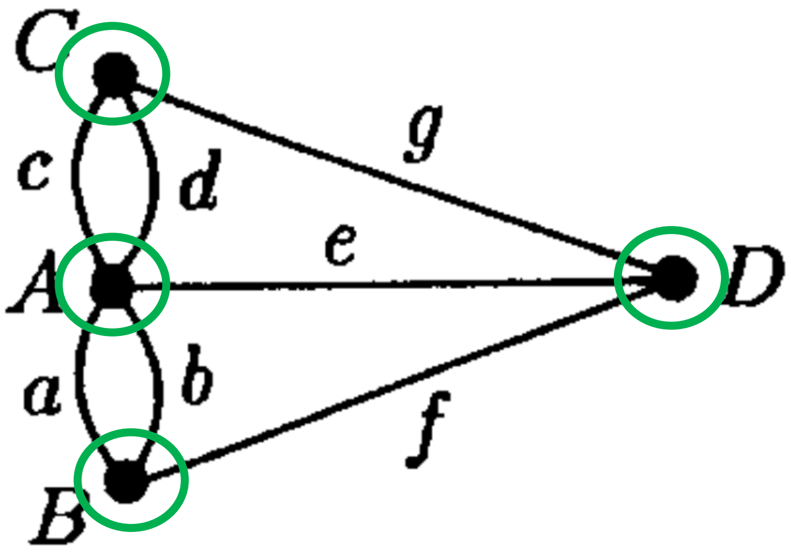
6. Expanding (1,2,5,8,2,4,3,1)
7. Remove these edges
8. First vertex incident with remaining edges is 4
9. Greedy algorithm yields (4,6,7,4,9,6,10,4)
10. Expanding (1,2,5,8,2,4,6,7,4,9,6,10,4,3,1)
11. Remove these edges
12. First vertex incident with remaining edges is 7
13. Greedy algorithm yields (7,9,11,7)
14. Expanding (1,2,5,8,2,4,6,7,9,11,7,4,9,6,10,4,3,1)



Eulerian circuit

- Theorem (1.2.26, W) A graph G is Eulerian $\Leftrightarrow$ it has at most one nontrivial component and its vertices all have even degree





Other properties

- Proposition (1.2.27, W) Every even graph decomposes into cycles

- The necessary and sufficient condition for a directed Eulerian circuit is that the graph is connected and that each vertex has the same ‘in-degree’ as ‘out-degree’

TONCAS

- TONCAS: The obvious necessary condition is also sufficient
- Theorem (1.2.26, W) A graph G is Eulerian $\Leftrightarrow$ it has at most one nontrivial component and its vertices all have even degree
- Proposition (1.3.28, W) The nonnegative integers $d_1, \ldots, d_n$ are the vertex degrees of some graph $\Leftrightarrow \sum_{i=1}^{n} d_i$ is even
- (Possibly with loops)
- Otherwise $(2,0,0)$ is not realizable
- **1.3.63.** (!) Let $d_1, \ldots, d_n$ be integers such that $d_1 \geq \cdots \geq d_n \geq 0$. Prove that there is a loopless graph (multiple edges allowed) with degree sequence $d_1, \ldots, d_n$ if and only if $\sum d_i$ is even and $d_1 \leq d_2 + \cdots + d_n$. (Hakimi [1962])

Hamiltonian path/circuits

- A path P is Hamiltonian if $V(P) = V(G)$
  - Any graph contains a Hamiltonian path is called traceable
- A cycle C is called Hamiltonian if it spans all vertices of G
  - A graph is called Hamiltonian if it contains a Hamiltonian circuit
- In the mid-19th century, Sir William Rowan Hamilton tried to popularize the exercise of finding such a closed path in the graph of the dodecahedron (正十二面体)

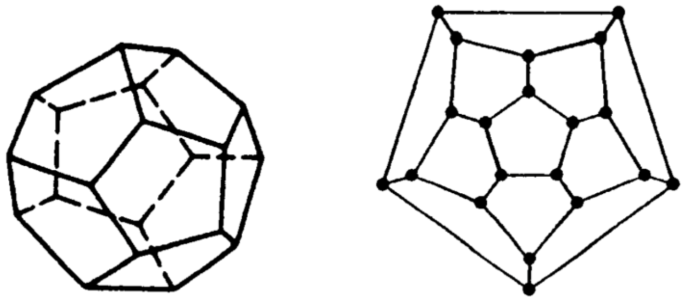
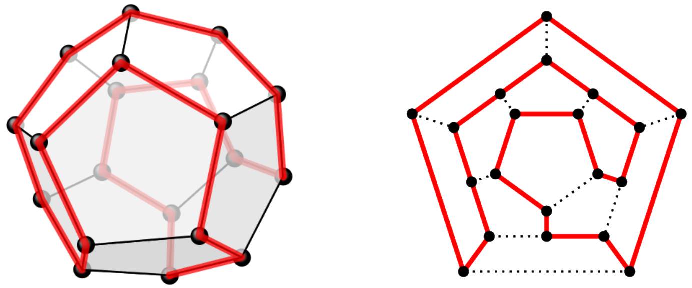


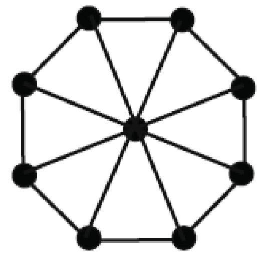
Figure 1.9



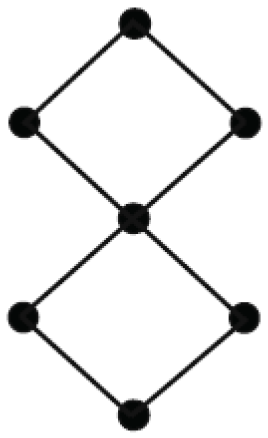
Degree parity is not a criterion

Theorem (1.2.26, W) A graph G is Eulerian $\Leftrightarrow$ it has at most one nontrivial component and its vertices all have even degree

- Hamiltonian graphs
  - all even degrees C_{10}
  - all odd degrees K_{10}
  - a mixture G_1
- non-Hamiltonian graphs
  - all even G_2
  - all odd $K_{5,7}$
  - mixed P_9



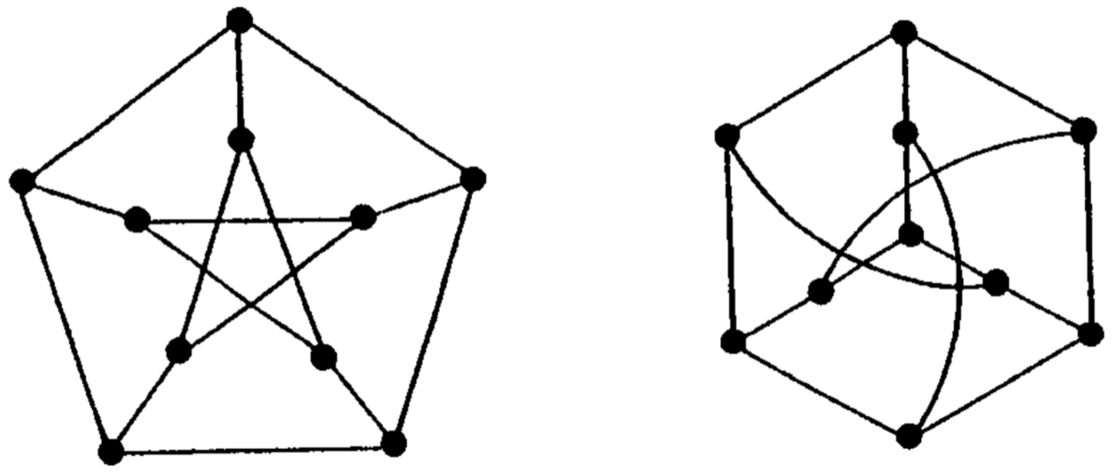
G_1



G_2

Example

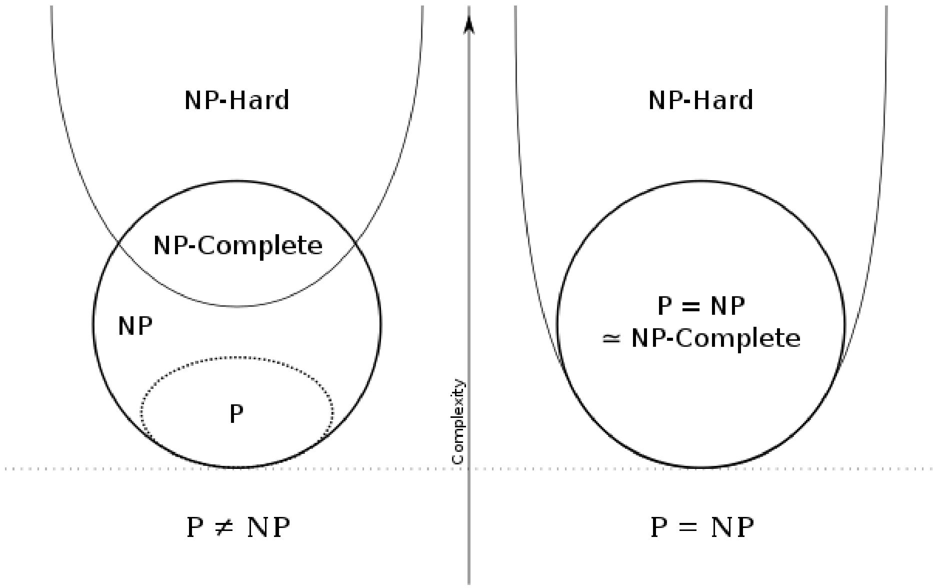
- The Petersen graph has a Hamiltonian path but no Hamiltonian cycle



- Determining whether such paths and cycles exist in graphs is the Hamiltonian path problem, which is NP-complete

P, NP, NPC, NP-hard

- P The general class of questions for which some algorithm can provide an answer in polynomial time
- NP (nondeterministic polynomial time) The class of questions for which an answer can be *verified* in polynomial time
- NP-Complete
    1. c is in NP
    2. Every problem in NP is reducible to c in polynomial time
- NP-hard
    - ~~c is in NP~~
    - Every problem in NP is reducible to c in polynomial time



Large minimal degree implies Hamiltonian

- Theorem (1.22, H, Dirac) Let G be a graph of order $n \geq 3$. If $\delta(G) \geq n/2$, then G is Hamiltonian

Proposition (1.3.15, W) If $\delta(G) \geq \frac{n-1}{2}$, then G is connected

(Ex16, S1.1.2, H) (1.3.16, W)
If $\delta(G) \geq \frac{n-2}{2}$, then G need not be connected

- The bound is tight
(Ex12b, S1.4.3, H) $G = K_{r,r+1}$ is not Hamiltonian
Exercise The condition when $K_{r,s}$ is Hamiltonian
- The condition is not necessary
  - C_n is Hamiltonian but with small minimum (and even maximum) degree

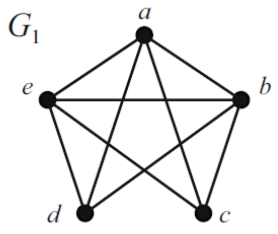
Generalized version

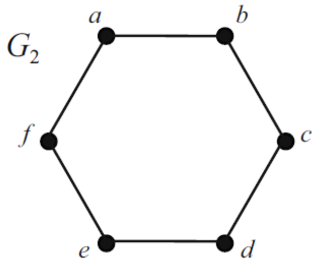
- Exercise (Theorem 1.23, H, Ore; Ex3, S1.4.3, H) Let G be a graph of order $n \geq 3$. If $\deg(x) + \deg(y) \geq n$ for all pairs of nonadjacent vertices x, y, then G is Hamiltonian

> Theorem (1.22, H, Dirac) Let G be a graph of order $n \geq 3$. If $\delta(G) \geq n/2$, then G is Hamiltonian

Independence number & Hamiltonian

- A set of vertices in a graph is called independent if they are pairwise nonadjacent
- The independence number of a graph G, denoted as $\alpha(G)$, is the largest size of an independent set
- Example: $\alpha(G_1) = 2, \alpha(G_2) = 3$
- Theorem (1.24, H) Let G be a connected graph of order $n \geq 3$. If $\kappa(G) \geq \alpha(G)$, then G is Hamiltonian





(Ex14, S1.1.2, H) $\kappa(G) \geq 2$ implies G has at least one cycle

Independence number & Hamiltonian 2

Theorem (1.24, H) Let G be a connected graph of order $n \geq 3$. If $\kappa(G) \geq \alpha(G)$, then G is Hamiltonian

- The result is tight: $\kappa(G) \geq \alpha(G) - 1$ is not enough
  - $K_{r,r+1}$: $\kappa = \mathrm{r}, \alpha = r + 1$
  - Exercise (Ex4, S1.4.3, H) Peterson graph: $\kappa = 3, \alpha = 4$

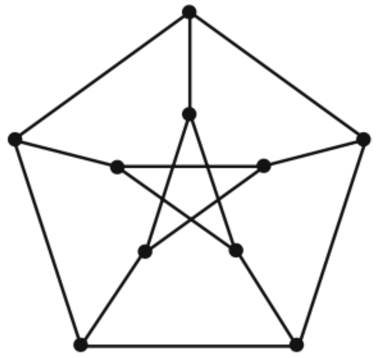
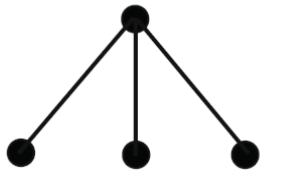
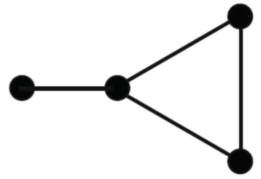


FIGURE 1.63. The Petersen Graph.

Pattern-free & Hamiltonian



$K_{1,3}$



Z_1

- G is H-free if G doesn't contain a copy of H as induced subgraph
- Theorem (1.25, H) If G is 2-connected and $\{K_{1,3}, Z_1\}$-free, then G is Hamiltonian

(Ex14, S1.1.2, H) $\kappa(G) \geq 2$ implies G has at least one cycle

- The condition 2-connectivity is necessary
- (Ex2, S1.4.3, H) If G is Hamiltonian, then G is 2-connected

Lecture 5: Matchings

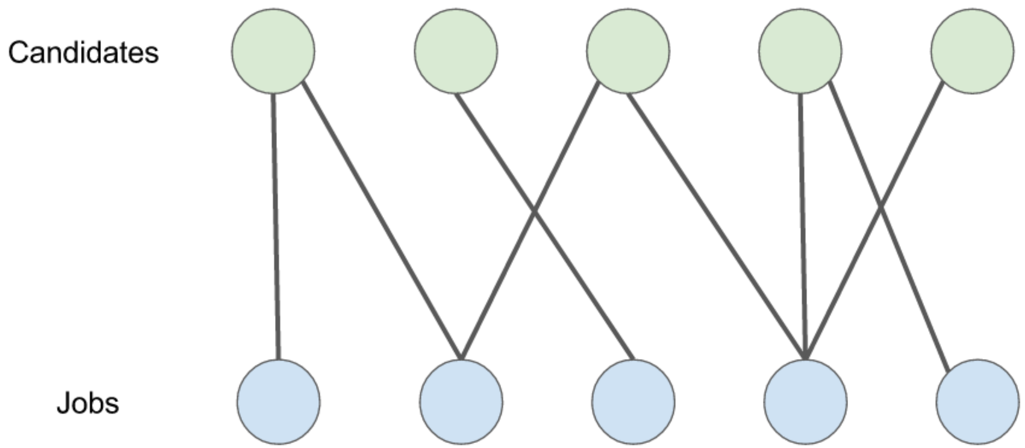
Shuai Li

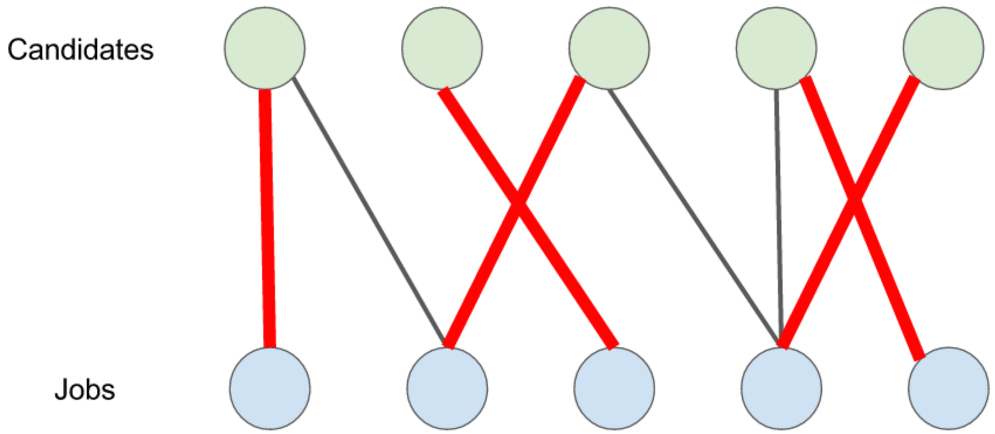
John Hopcroft Center, Shanghai Jiao Tong University

https://shuaili8.github.io

https://shuaili8.github.io/Teaching/CS445/index.html

Motivating example



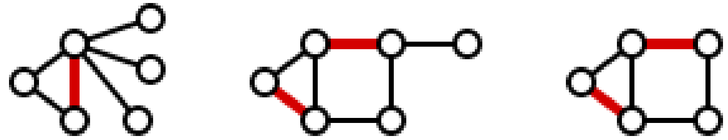


Definitions

- A matching is a set of independent edges, in which no pair of edges shares a vertex
- The vertices incident to the edges of a matching M are M-saturated (饱和的); the others are M-unsaturated
- A perfect matching in a graph is a matching that saturates every vertex
- Example (3.1.2, W) The number of perfect matchings in $K_{n,n}$ is $n!$
- Example (3.1.3, W) The number of perfect matchings in K_{2n} is
$$f_n = (2n-1)(2n-3)\cdots 1 = (2n-1)!!$$

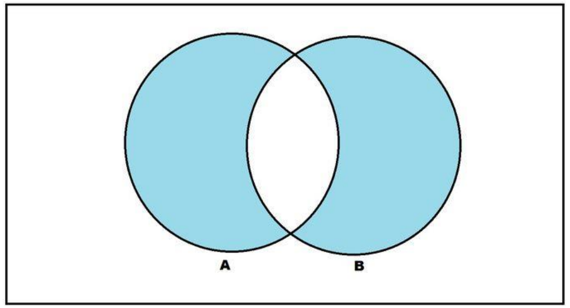
Maximal/maximum matchings 极大/最大

- A maximal matching in a graph is a matching that cannot be enlarged by adding an edge
- A maximum matching is a matching of maximum size among all matchings in the graph
- Example: P_3, P_5

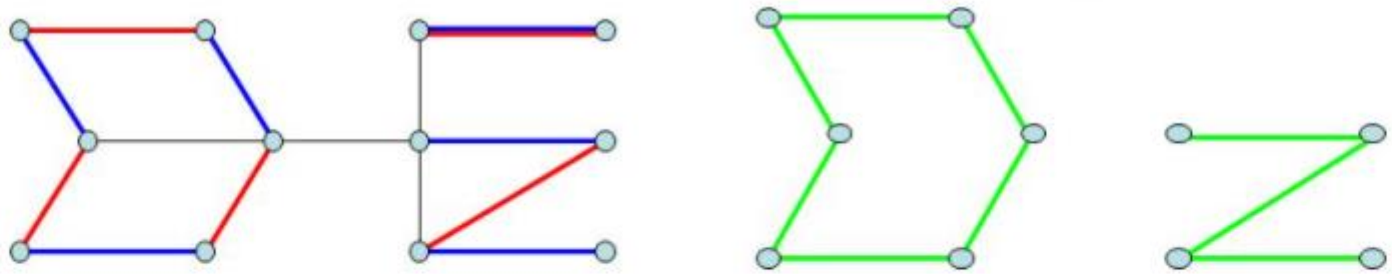


- Every maximum matching is maximal, but not every maximal matching is a maximum matching

Symmetric difference of matchings



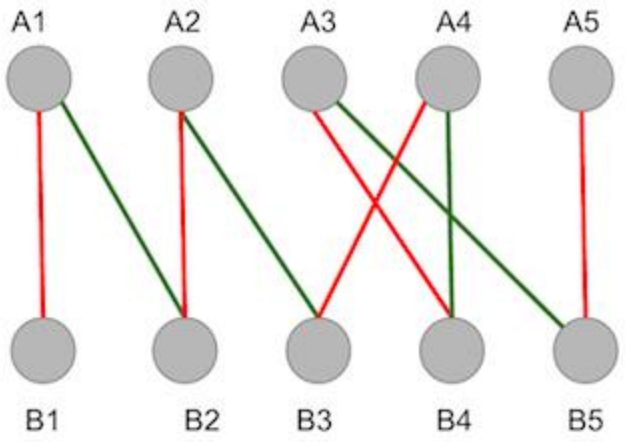
- The symmetric difference of M, M' is $M \Delta M' = (M - M') \cup (M' - M)$
- Lemma (3.1.9, W) Every component of the symmetric difference of two matchings is a path or an even cycle

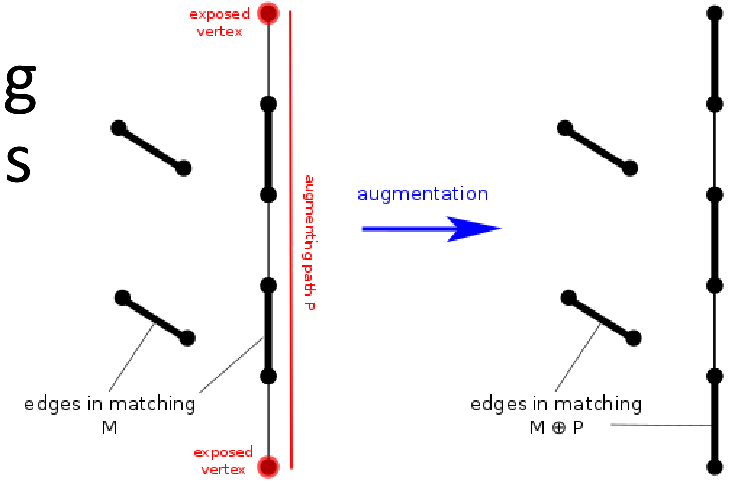


Maximum matching and augmenting path

- Given a matching M, an M-alternating path is a path that alternates between edges in M and edges not in M
- An M-alternating path whose endpoints are M-unsaturated is an M-augmenting path
- Theorem (3.1.10, W; 1.50, H; Berge 1957) A matching M in a graph G is a maximum matching in G $\Leftrightarrow$ G has no M-augmenting path

Lemma (3.1.9, W) Every component of the symmetric difference of two matchings is a path or an even cycle





Hall's theorem (TONCAS)

- Theorem (3.1.11, W; 1.51, H; 2.1.2, D; Hall 1935) Let G be a bipartite graph with partition X, Y.
  G contains a matching of $X \Leftrightarrow |N(S)| \geq |S|$ for all $S \subseteq X$

> Theorem (3.1.10, W; 1.50, H; Berge 1957) A matching M in a graph G is a maximum matching in $G \Leftrightarrow G$ has no M-augmenting path

- Exercise. Read the other two proofs in Diestel.
- Corollary (3.1.13, W; 2.1.3, D) Every k-regular ($k > 0$) bipartite graph has a perfect matching

General regular graph

- Corollary (2.1.5, D) Every regular graph of positive even degree has a 2-factor
  - A k-regular spanning subgraph is called a k-factor
  - A perfect matching is a 1-factor

Theorem (1.2.26, W) A graph G is Eulerian $\Leftrightarrow$ it has at most one nontrivial component and its vertices all have even degree

Corollary (3.1.13, W; 2.1.3, D) Every k-regular ($k > 0$) bipartite graph has a perfect matching

Application to SDR

- Given some family of sets X, a system of distinct representatives for the sets in X is a 'representative' collection of distinct elements from the sets of X

$$S_1 = \{2, 8\},$$
$$S_2 = \{8\},$$
$$S_3 = \{5, 7\},$$
$$S_4 = \{2, 4, 8\},$$
$$S_5 = \{2, 4\}.$$

The family $X_1 = \{S_1, S_2, S_3, S_4\}$ does have an SDR, namely $\{2, 8, 7, 4\}$. The family $X_2 = \{S_1, S_2, S_4, S_5\}$ does not have an SDR.

- Theorem(1.52, H) Let $S_1, S_2, \dots, S_k$ be a collection of finite, nonempty sets. This collection has SDR $\Leftrightarrow$ for every $t \in [k]$, the union of any t of these sets contains at least t elements

Theorem (3.1.11, W; 1.51, H; 2.1.2, D; Hall 1935) Let G be a bipartite graph with partition X, Y.
G contains a matching of $X \Leftrightarrow |N(S)| \geq |S|$ for all $S \subseteq X$

König Theorem
Augmenting Path Algorithm

Vertex cover

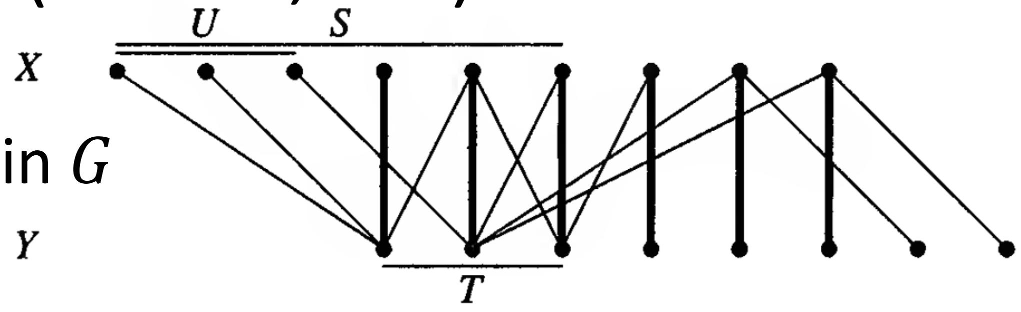
- A set $U \subseteq V$ is a (vertex) cover of E if every edge in G is incident with a vertex in U
- Example:
    - Art museum is a graph with hallways are edges and corners are nodes
    - A security camera at the corner will guard the paintings on the hallways
    - The minimum set to place the cameras?

König-Egeváry Theorem (Min-max theorem)

- Theorem (3.1.16, W; 1.53, H; 2.1.1, D; König 1931; Egeváry 1931) Let G be a bipartite graph. The maximum size of a matching in G is equal to the minimum size of a vertex cover of its edges

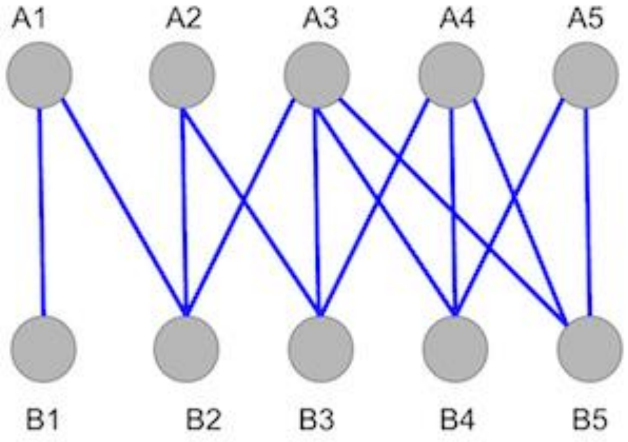
> Theorem (3.1.10, W; 1.50, H; Berge 1957) A matching M in a graph G is a maximum matching in $G \Leftrightarrow G$ has no M-augmenting path

Augmenting path algorithm (3.2.1, W)

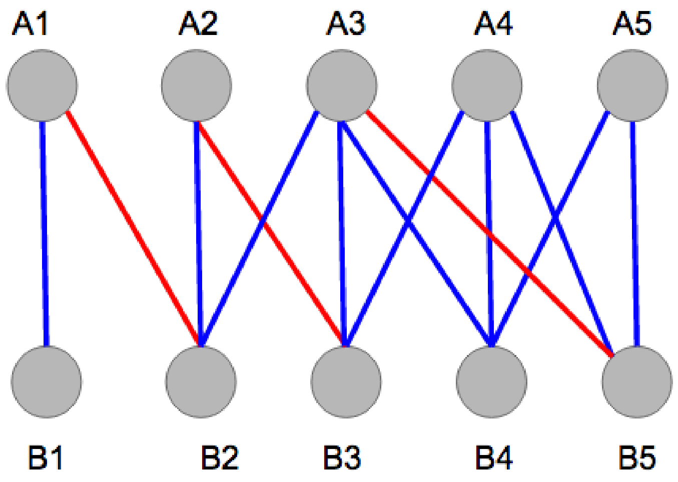


- **Input**: G is Bipartite with X, Y, a matching M in G
  $U = \{M\text{−unsaturated vertices in } X\}$
- **Idea**: Explore M-alternating paths from U
  letting $S \subseteq X$ and $T \subseteq Y$ be the sets of vertices reached
- **Initialization**: $S = U, T = \emptyset$ and all vertices in S are unmarked
- **Iteration**:
  - If S has no unmarked vertex, stop and report $T \cup (X - S)$ as a minimum cover and M as a maximum matching
  - Otherwise, select an unmarked $x \in S$ to explore
    - Consider each $y \in N(x)$ such that $xy \notin M$
      - If y is unsaturated, terminate and report an M-augmenting path from U to y
      - Otherwise, $yw \in M$ for some w
        - include y in T (reached from x) and include w in S (reached from y)
    - After exploring all such edges incident to x, mark x and iterate.

Example



Red: A random matching



Theoretical guarantee for Augmenting path algorithm

- Theorem (3.2.2, W) Repeatedly applying the Augmenting Path Algorithm to a bipartite graph produces a matching and a vertex cover of equal size

Weighted Bipartite Matching
Hungarian Algorithm

Weighted bipartite matching

- The maximum weighted matching problem is to seek a perfect matching M to maximize the total weight $w(M)$
- Bipartite graph
  - W.l.o.g. Assume the graph is $K_{n,n}$ with $w_{i,j} \geq 0$ for all $i, j \in [n]$
  - Optimization:

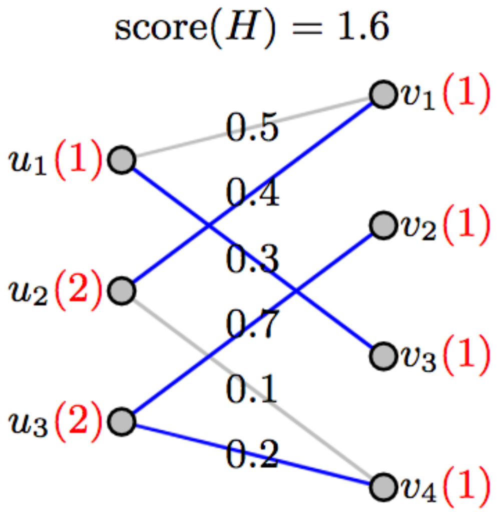
$$\max w(M_a) = \sum_{i,j} a_{i,j} w_{i,j}$$

$$s.t.\ a_{i,1} + \cdots + a_{i,n} \leq 1 \text{ for any } i$$

$$a_{1,j} + \cdots + a_{n,j} \leq 1 \text{ for any } j$$

$$a_{i,j} \in \{0,1\}$$

  - Integer programming
  - General IP problems are NP-Complete



(Weighted) cover

- A (weighted) cover is a choice of labels $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ such that $u_i + v_j \geq w_{i,j}$ for all i, j
  - The cost $c(u, v)$ of a cover (u, v) is $\sum_i u_i + \sum_j v_j$
  - The minimum weighted cover problem is that of finding a cover of minimum cost
- Optimization problem

$$\min c(u, v) = \sum_i u_i + \sum_j v_j$$

$$s.t.\, u_i + v_j \geq w_{i,j} \text{ for any } i, j$$

$$u_i, v_j \geq 0 \text{ for any } i, j$$

Duality

(IP)

$$\max \sum_{i,j} a_{i,j} w_{i,j}$$
$$s.t.\, a_{i,1} + \cdots + a_{i,n} \le 1 \text{ for any } i$$
$$a_{1,j} + \cdots + a_{n,j} \le 1 \text{ for any } j$$
$$a_{i,j} \in \{0,1\}$$

(Linear programming)

$$\max \sum_{i,j} a_{i,j} w_{i,j}$$
$$s.t.\, a_{i,1} + \cdots + a_{i,n} \le 1 \text{ for any } i$$
$$a_{1,j} + \cdots + a_{n,j} \le 1 \text{ for any } j$$
$$a_{i,j} \ge 0$$

(Dual)

$$\min \sum_{i} u_i + \sum_{j} v_j$$
$$s.t.\, u_i + v_j \ge w_{i,j} \text{ for any } i, j$$
$$u_i, v_j \ge 0$$

- Weak duality theorem
  - For each feasible solution a and (u, v)

$$\sum_{i,j} a_{i,j} w_{i,j} \le \sum_{i} u_i + \sum_{j} v_j$$

thus $\max \sum_{i,j} a_{i,j} w_{i,j} \le \min \sum_i u_i + \sum_j v_j$

Duality (cont.)

- Strong duality theorem
  - If one of the two problems has an optimal solution, so does the other one and that the bounds given by the weak duality theorem are tight

$$\max \sum_{i,j} a_{i,j} w_{i,j} = \min \sum_{i} u_i + \sum_{j} v_j$$

- Lemma (3.2.7, W) For a perfect matching M and cover (u, v) in a weighted bipartite graph G, $c(u, v) \geq w(M).$
  $c(u, v) = w(M) \Leftrightarrow$ M consists of edges $x_i y_j$ such that $u_i + v_j = w_{i,j}$
  In this case, M and (u, v) are optimal.

Equality subgraph

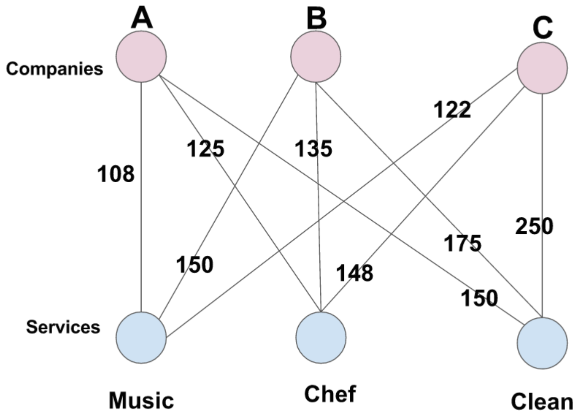
- The equality subgraph $G_{u,v}$ for a cover (u, v) is the spanning subgraph of $K_{n,n}$ having the edges $x_i y_j$ such that $u_i + v_j = w_{i,j}$
  - So if $c(u, v) = w(M)$ for some perfect matching M, then M is composed of edges in $G_{u,v}$
  - And if $G_{u,v}$ contains a perfect matching M, then (u, v) and M (whose weights are $u_i + v_j$) are both optimal

Hungarian algorithm

- **Input**: Weighted $K_{n,n} = B(X, Y)$
- **Idea**: Iteratively adjusting the cover (u, v) until the equality subgraph $G_{u,v}$ has a perfect matching
- **Initialization**: Let (u, v) be a cover, such as $u_i = \max_j w_{i,j}$, $v_j = 0$

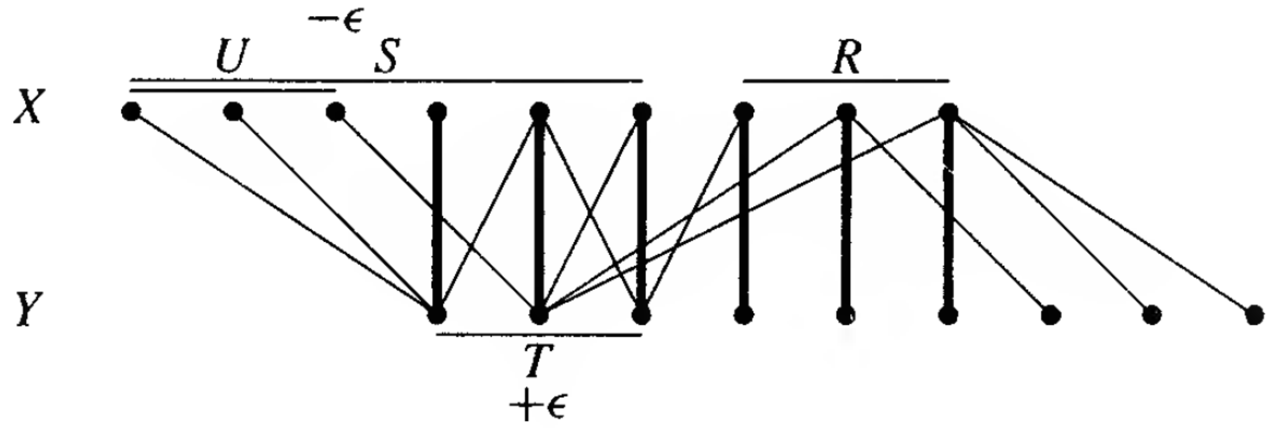
(Dual)

$$\min \sum_i u_i + \sum_j v_j$$
$$s.t. u_i + v_j \geq w_{i,j} \text{ for any } i, j$$
$$u_i, v_j \geq 0$$

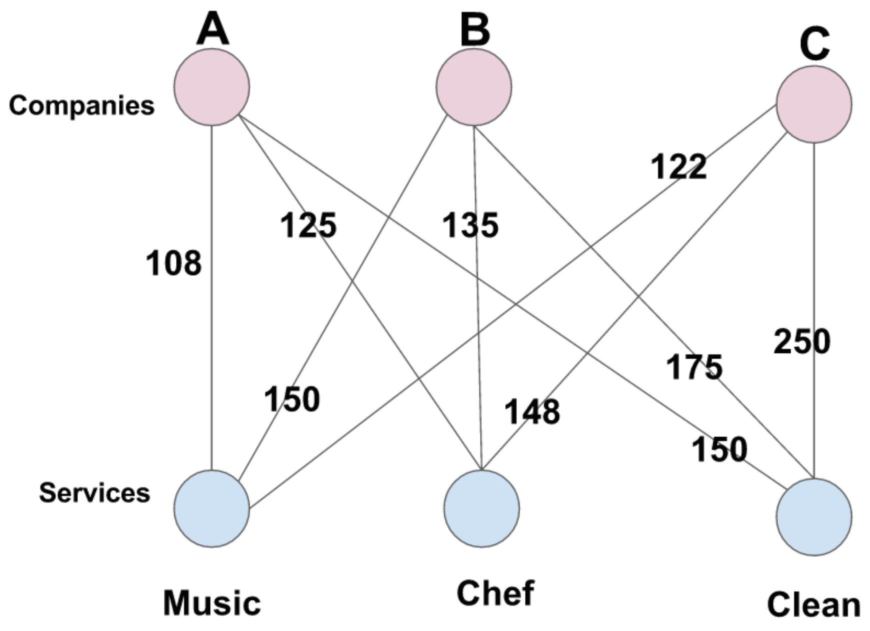


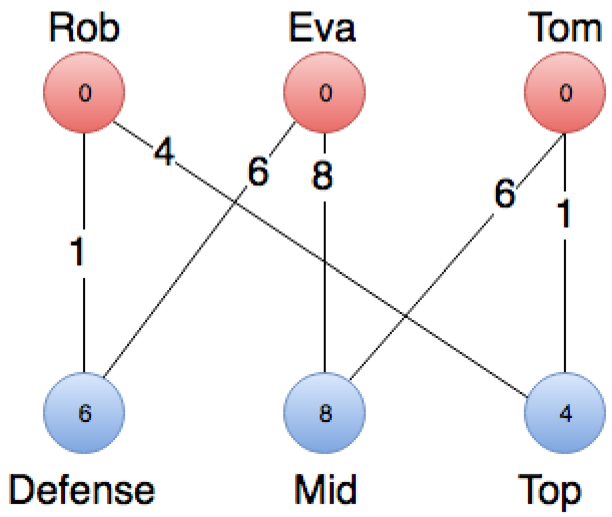
Hungarian algorithm (cont.)

- **Iteration**: Find a maximum matching M in $G_{u,v}$
  - If M is a perfect matching, stop and report M as a maximum weight matching
  - Otherwise, let Q be a vertex cover of size $|M|$ in $G_{u,v}$
    - Let $R = X \cap Q, T = Y \cap Q$
    $$\epsilon = \min\{u_i + v_j - w_{i,j} : x_i \in X - R, y_j \in Y - T\}$$
    - Decrease u_i by ϵ for $x_i \in X - R$ and increase v_j by ϵ for $y_j \in T$
  - Form the new equality subgraph and repeat



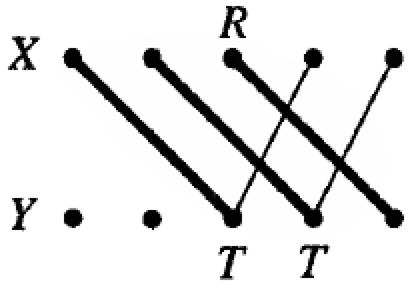
Example


A
B
C
Companies
108
125
135
122
150
148
175
250
150
Services
Music
Chef
Clean

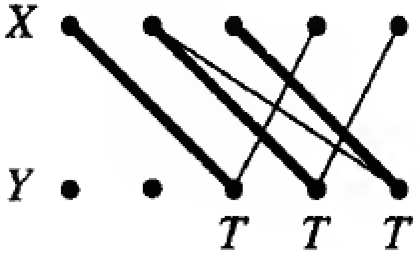

Rob
Eva
Tom
0
0
0
4
6
8
6
1
1
6
8
4
Defense
Mid
Top

Example 2: Excess matrix

$$
\begin{pmatrix} 4 & 1 & 6 & 2 & 3 \\ 5 & 0 & 3 & 7 & 6 \\ 2 & 3 & 4 & 5 & 8 \\ 3 & 4 & 6 & 3 & 4 \\ 4 & 6 & 5 & 8 & 6 \end{pmatrix}
\rightarrow
\begin{array}{ccc}
 & \begin{matrix} 0 & 0 & 0 & 0 & 0 \end{matrix} & \\
\begin{matrix} 6 \\ 7 \\ 8 \\ 6 \\ 8 \end{matrix} &
\begin{pmatrix} 2 & 5 & \underline{0} & 4 & 3 \\ 2 & 7 & 4 & \underline{0} & 1 \\ 6 & 5 & 4 & 3 & \underline{0} \\ 3 & 2 & 0 & 3 & 2 \\ 4 & 2 & 3 & 0 & 2 \end{pmatrix} &
\begin{matrix} \\ \\ R \\ \\ \\ \end{matrix} \\
 & \begin{matrix} & & T & T & \end{matrix} &
\end{array}
$$



$$
\begin{array}{cc}
 & \begin{matrix} 0 & 0 & 1 & 1 & 0 \end{matrix} \\
\begin{matrix} 5 \\ 6 \\ 8 \\ 5 \\ 7 \end{matrix} &
\begin{pmatrix} 1 & 4 & \underline{0} & 4 & 2 \\ 1 & 6 & 4 & \underline{0} & 0 \\ 6 & 5 & 5 & 4 & \underline{0} \\ 2 & 1 & 0 & 3 & 1 \\ 3 & 1 & 3 & 0 & 1 \end{pmatrix} \\
 & \begin{matrix} & & T & T & T \end{matrix}
\end{array}
$$



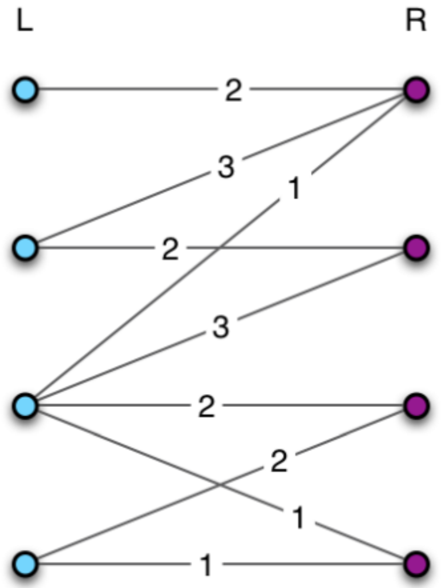
$$
\rightarrow
\begin{array}{cc}
 & \begin{matrix} 0 & 0 & 2 & 2 & 1 \end{matrix} \\
\begin{matrix} 4 \\ 5 \\ 7 \\ 4 \\ 6 \end{matrix} &
\begin{pmatrix} 0 & 3 & \underline{0} & 4 & 2 \\ \underline{0} & 5 & 4 & 0 & 0 \\ 5 & 4 & 5 & 4 & \underline{0} \\ 1 & \underline{0} & 0 & 3 & 1 \\ 2 & 0 & 3 & \underline{0} & 1 \end{pmatrix}
\end{array}
$$

Optimal value is the same
But the solution is not unique

Theoretical guarantee for Hungarian algorithm

- Theorem (3.2.11, W) The Hungarian Algorithm finds a maximum weight matching and a minimum cost cover

Example 3



Back to (unweighted) bipartite graph

- The weights are binary $0,1$
- Hungarian algorithm always maintain integer labels in the weighted cover, thus the solution will always be $0,1$
- The vertices receiving label 1 must cover the weight on the edges, thus cover all edges
- So the solution is a minimum vertex cover

Stable Matchings

Stable matching

- A family $(\leq_v)_{v \in V}$ of linear orderings $\leq_v$ on $E(v)$ is a set of preferences for G
- A matching M in G is stable if for any edge $e \in E \setminus M$, there exists an edge $f \in M$ such that e and f have a common vertex v with $e <_v f$
  - Unstable: There exists $xy \in E \setminus M$ but $xy', x'y \in M$ with $xy' <_x xy$ $x'y <_y xy$

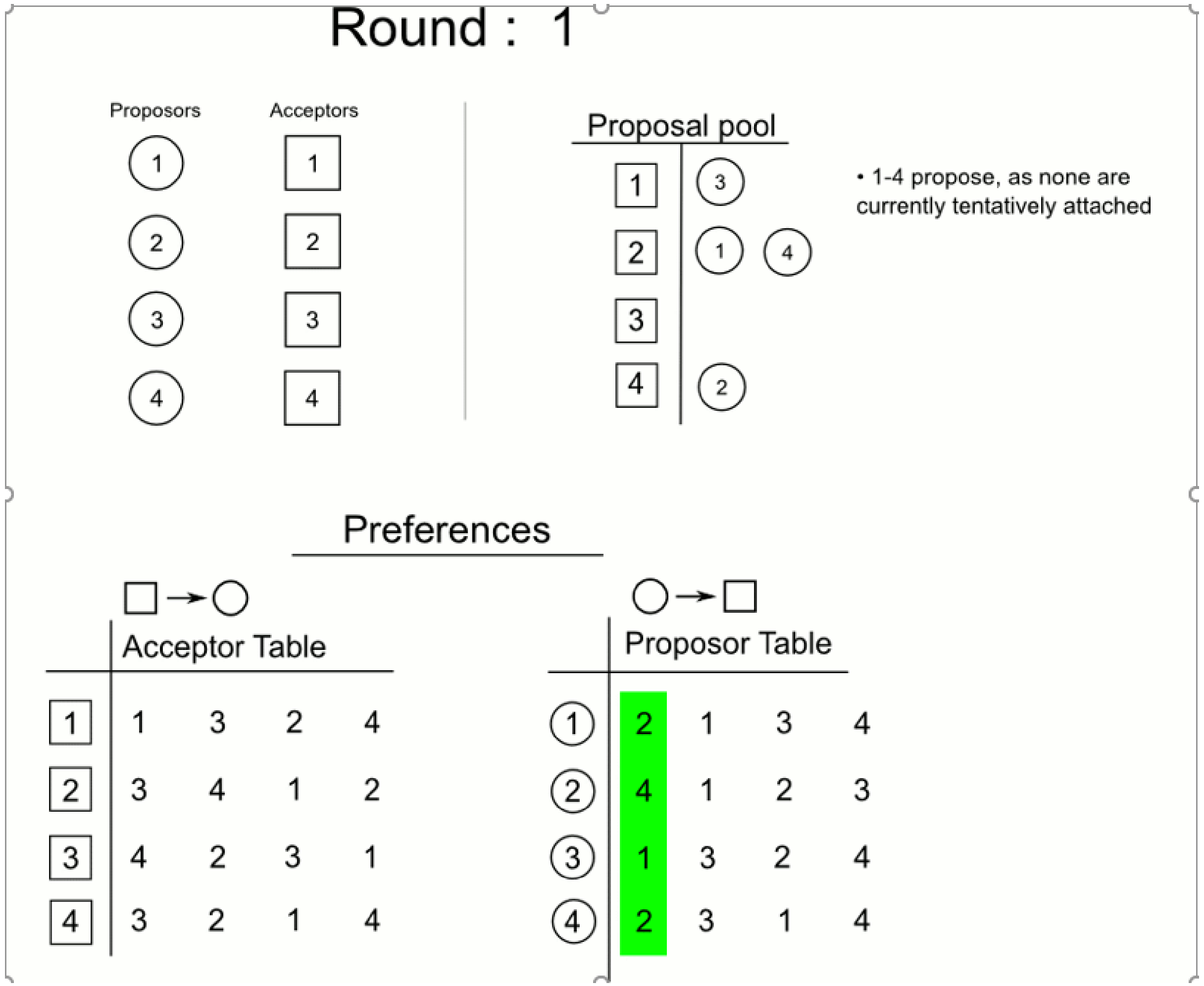
3.2.16. Example. Given men x, y, z, w, women a, b, c, d, and preferences listed below, the matching $\{xa, yb, zd, wc\}$ is a stable matching. ∎

Men $\{x, y, z, w\}$	Women $\{a, b, c, d\}$
$x : a > b > c > d$	$a : z > x > y > w$
$y : a > c > b > d$	$b : y > w > x > z$
$z : c > d > a > b$	$c : w > x > y > z$
$w : c > b > a > d$	$d : x > y > z > w$

Gale-Shapley Proposal Algorithm

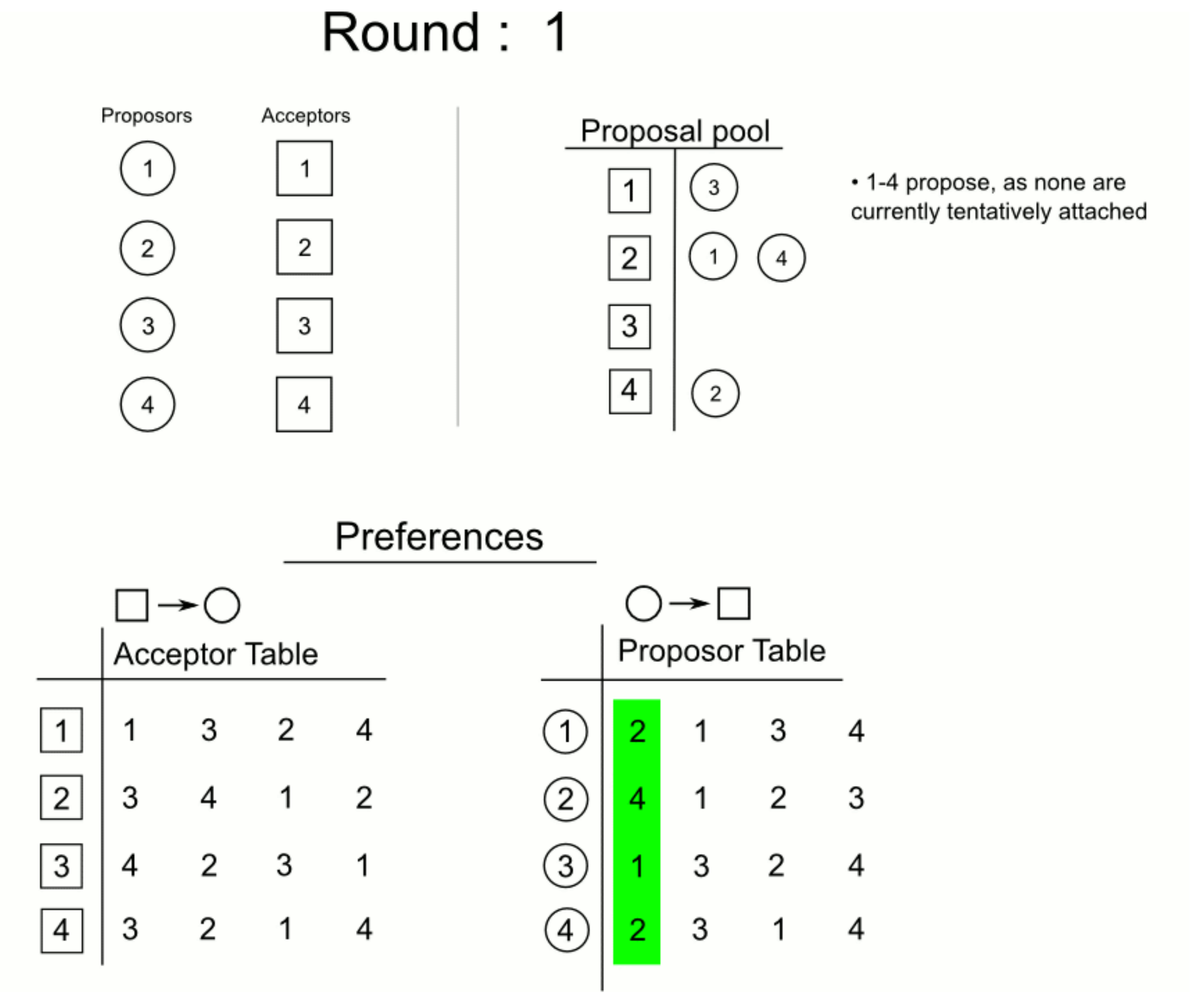
- **Input**: Preference rankings by each of n men and n women
- **Idea**: Produce a stable matching using proposals by maintaining information about who has proposed to whom and who has rejected whom
- **Iteration**: Each man proposes to the highest woman on his preference list who has not previously rejected him
  - If each woman receives exactly one proposal, stop and use the resulting matching
  - Otherwise, every woman receiving more than one proposal rejects all of them except the one that is highest on her preference list
  - Every woman receiving a proposal says “maybe” to the most attractive proposal received

Example



Round : 1
Proposors
Acceptors
1
2
3
4
Proposal pool
1 3
2 1 4
3
4 2
• 1-4 propose, as none are currently tentatively attached
Preferences
Acceptor Table
1 1 3 2 4
2 3 4 1 2
3 4 2 3 1
4 3 2 1 4
Proposor Table
1 2 1 3 4
2 4 1 2 3
3 1 3 2 4
4 2 3 1 4

Example (gif)


Round : 1
Proposors
Acceptors
Proposal pool
• 1-4 propose, as none are currently tentatively attached
Preferences
Acceptor Table
1 3 2 4
3 4 1 2
4 2 3 1
3 2 1 4
Proposor Table
2 1 3 4
4 1 2 3
1 3 2 4
2 3 1 4

Theoretical guarantee for the Proposal Algorithm

- Theorem (3.2.18, W, Gale-Shapley 1962) The Proposal Algorithm produces a stable matching
- Who proposes matters (jobs/candidates)
- Exercise Among all stable matchings, every man is happiest in the one produced by the male-proposal algorithm and every woman is happiest under the female-proposal algorithm

3.2.16. Example. Given men x, y, z, w, women a, b, c, d, and preferences listed below, the matching $\{xa, yb, zd, wc\}$ is a stable matching. ■

Men $\{x, y, z, w\}$	Women $\{a, b, c, d\}$
$x : a > b > c > d$	$a : z > x > y > w$
$y : a > c > b > d$	$b : y > w > x > z$
$z : c > d > a > b$	$c : w > x > y > z$
$w : c > b > a > d$	$d : x > y > z > w$

Matchings in General Graphs

Perfect matchings

- K_{2n}, C_{2n}, P_{2n} have perfect matchings
- Corollary (3.1.13, W; 2.1.3, D) Every k-regular ($k > 0$) bipartite graph has a perfect matching

- Theorem(1.58, H) If G is a graph of order $2n$ such that $\delta(G) \geq n$, then G has a perfect matching

Theorem (1.22, H, Dirac) Let G be a graph of order $n \geq 3$. If $\delta(G) \geq n/2$, then G is Hamiltonian

Tutte's Theorem (TONCAS)

- Let $q(G)$ be the number of connected components with odd order
- Theorem (1.59, H; 2.2.1, D; 3.3.3, W)
  Let G be a graph of order $n \geq 2$. G has a perfect matching $\Leftrightarrow q(G - S) \leq |S|$ for all $S \subseteq V$

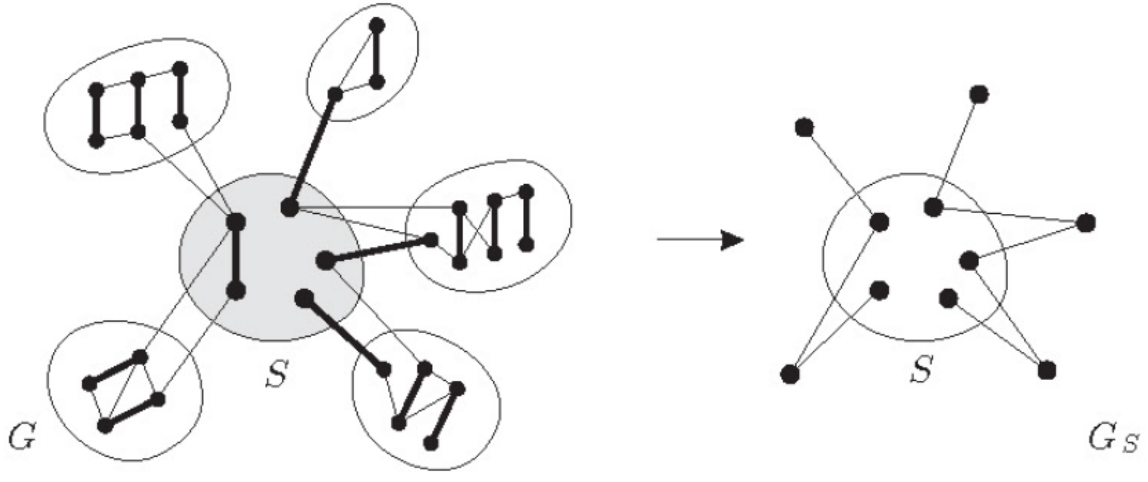


Fig. 2.2.1. Tutte's condition $q(G-S) \leqslant |S|$ for $q = 3$, and the contracted graph G_S from Theorem 2.2.3.

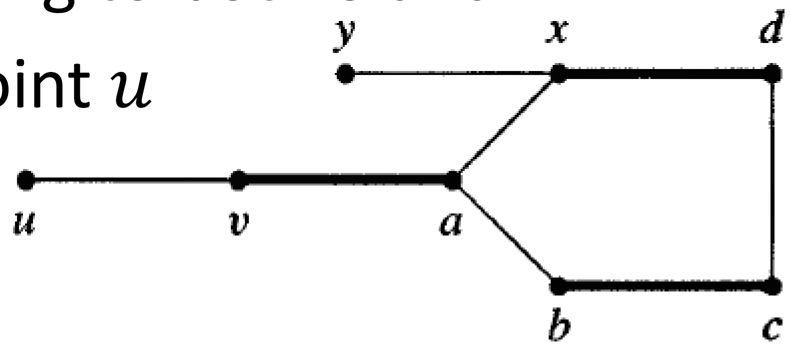
Petersen's Theorem

- Theorem (1.60, H; 2.2.2, D;3.3.8, W)
  Every bridgeless, 3-regular graph contains a perfect matching

Theorem (1.59, H; 2.2.1, D; 3.3.3, W)
Let G be a graph of order $n \geq 2$. G has a perfect matching $\Leftrightarrow q(G - S) \leq |S|$ for all $S \subseteq V$

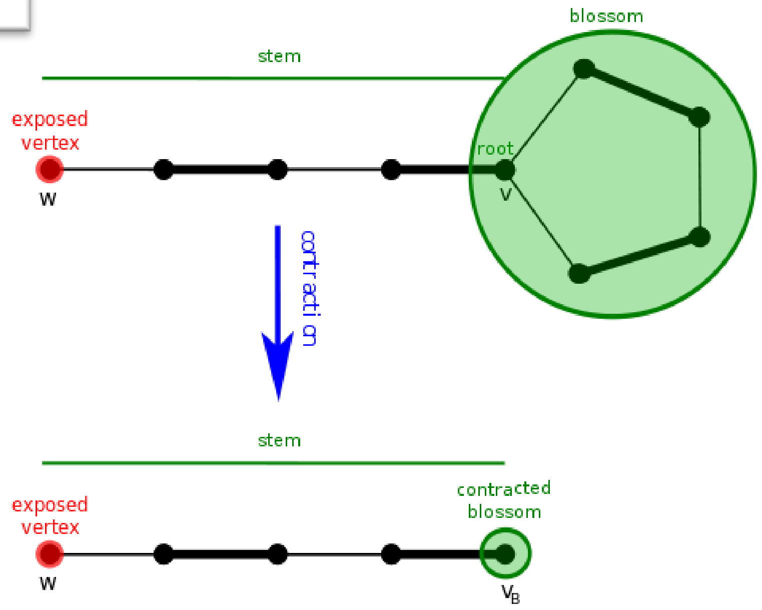
Find augmenting paths in general graphs

- Different from bipartite graphs, a vertex can belong to both S and T
- Example: How to explore from M-unsaturated point u

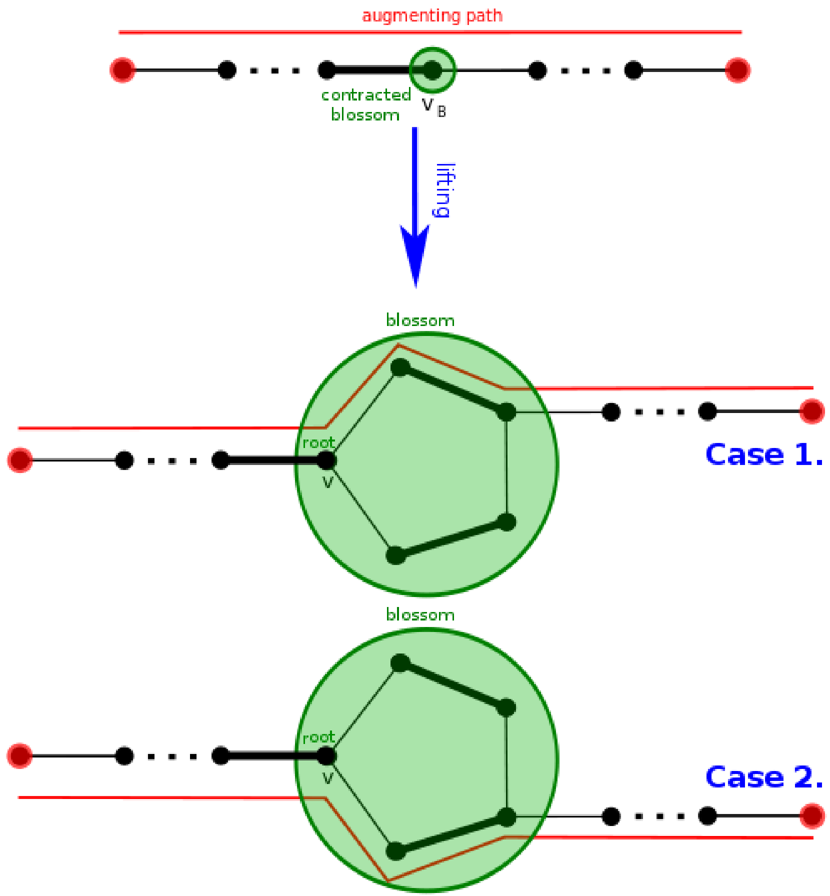


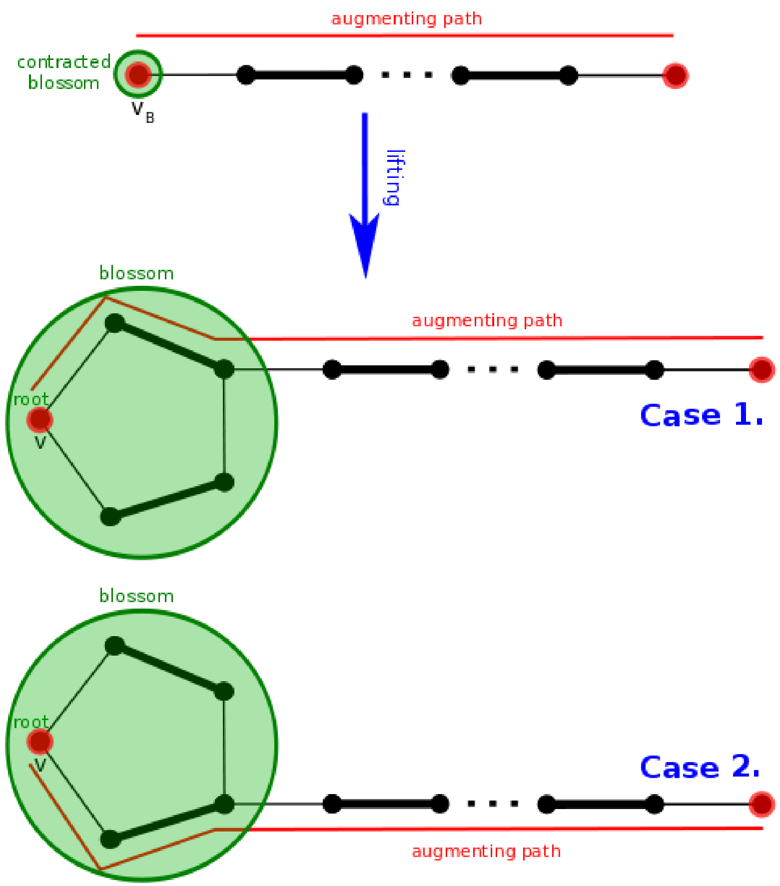
> Theorem (3.1.10, W; 1.50, H; Berge 1957) A matching M in a graph G is a maximum matching in G $\Leftrightarrow$ G has no M-augmenting path

- Flower/stem/blossom



Lifting

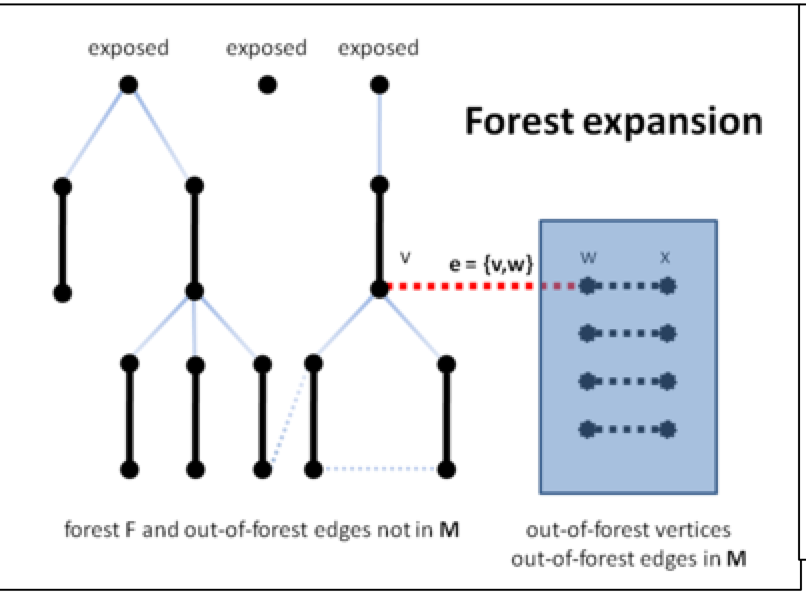

augmenting path
contracted blossom
v_B
lifting
blossom
root
v
Case 1.
blossom
root
v
Case 2.

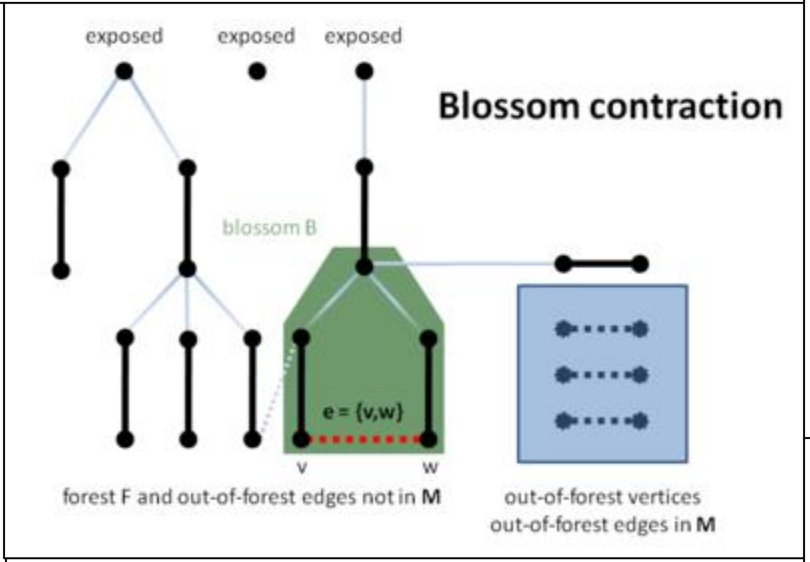

augmenting path
contracted blossom
v_B
lifting
blossom
augmenting path
root
v
Case 1.
blossom
root
v
Case 2.
augmenting path

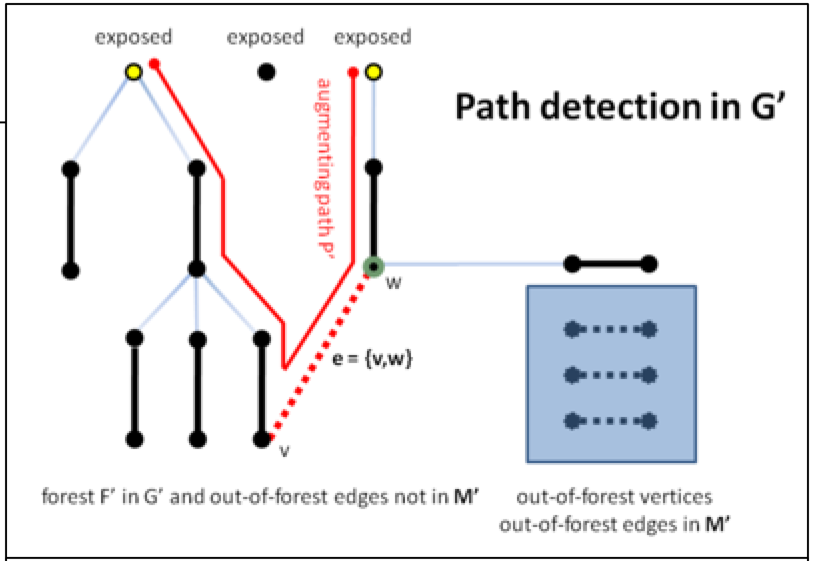
Edmonds' blossom algorithm (3.3.17, W)

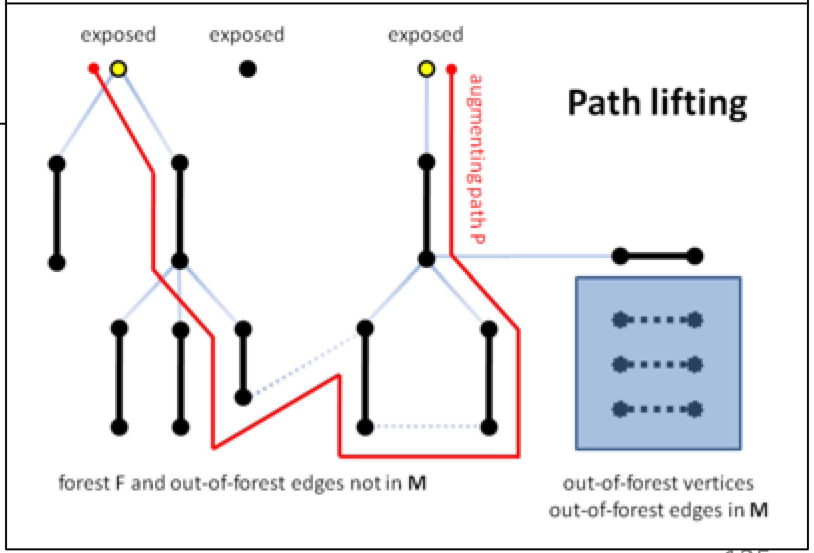
- **Input**: A graph G, a matching M in G, an M-unsaturated vertex u
- **Idea**: Explore M-alternating paths from u, recording for each vertex the vertex from which it was reached, and contracting blossoms when found
  - Maintain sets S and T analogous to those in Augmenting Path Algorithm, with S consisting of u and the vertices reached along saturated edges
  - Reaching an unsaturated vertex yields an augmentation.
- **Initialization**: $S = \{u\}$ and $T = \emptyset$
- **Iteration**: If S has no unmarked vertex, stop; there is no M-augmenting path from u
  - Otherwise, select an unmarked $v \in S$. To explore from v, successively consider each $y \in N(v)$ s.t. $y \notin T$
    - If y is unsaturated by M, then trace back from y (expanding blossoms as needed) to report an M-augmenting u, y-path
    - If $y \in S$, then a blossom has been found. Suspend the exploration of v and contract the blossom, replacing its vertices in S and T by a single new vertex in S. Continue the search from this vertex in the smaller graph.
    - Otherwise, y is matched to some w by M. Include y in T (reached from v), and include w in S (reached from y)
  - After exploring all such neighbors of v, mark v and iterate

Illustration

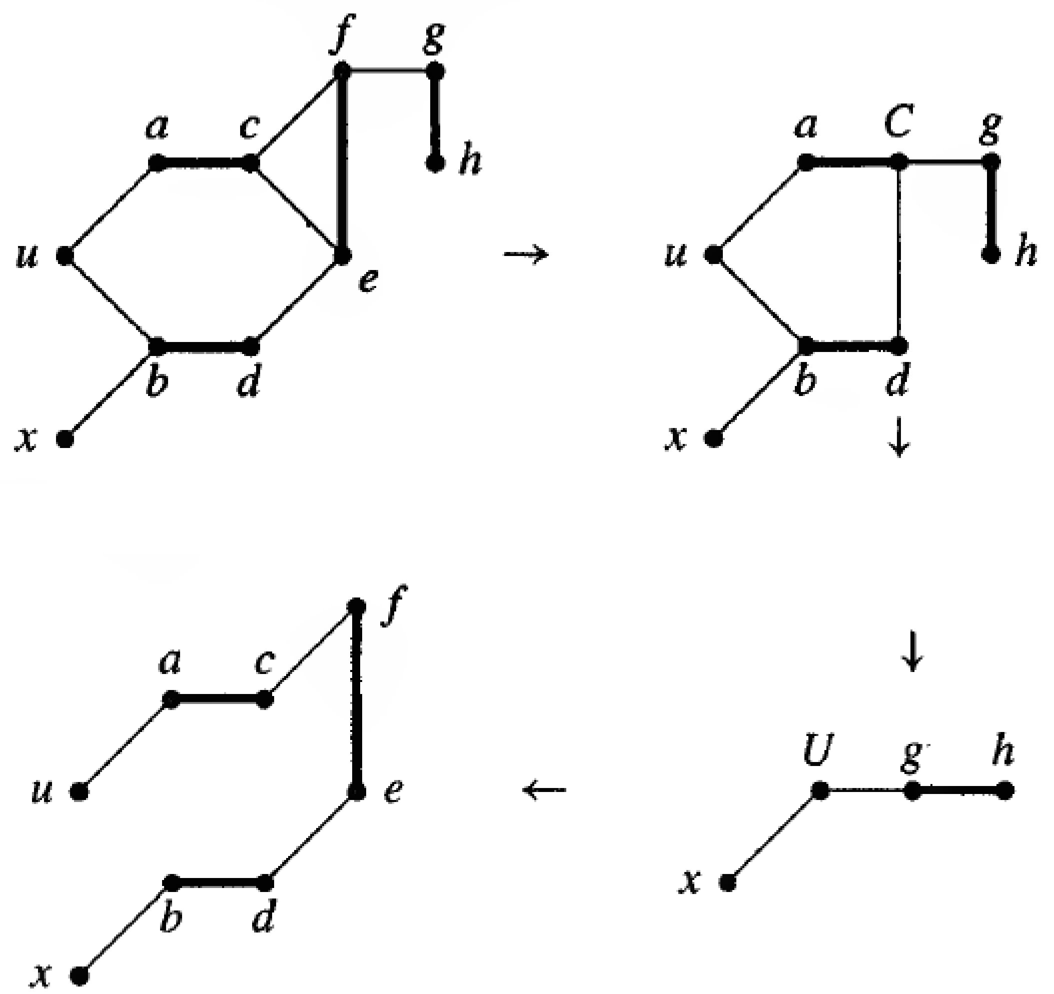




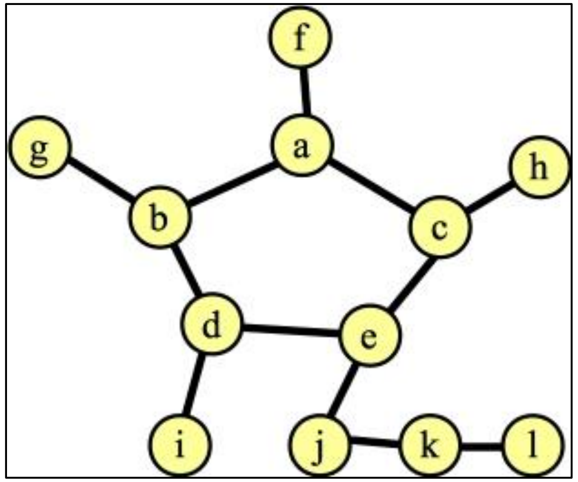


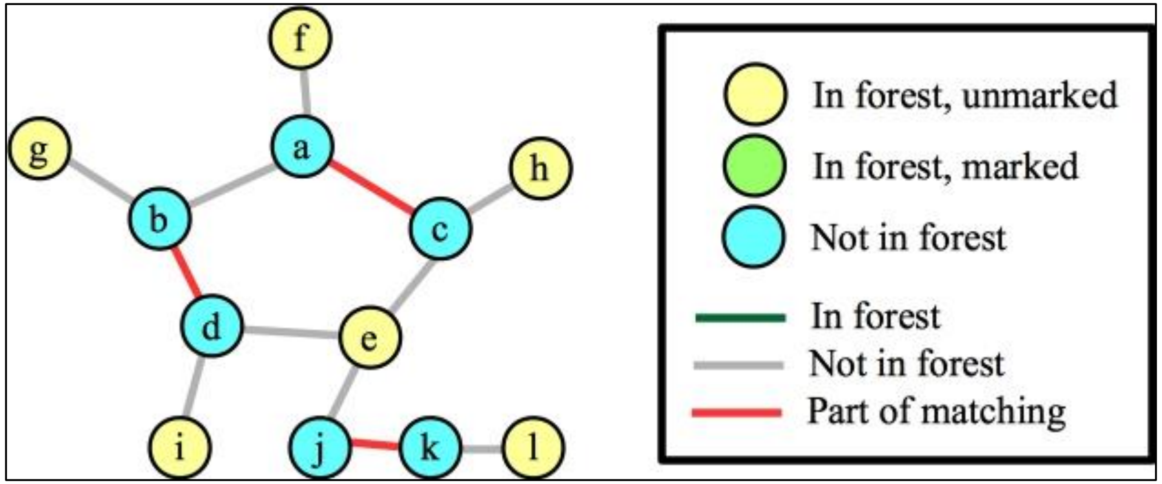


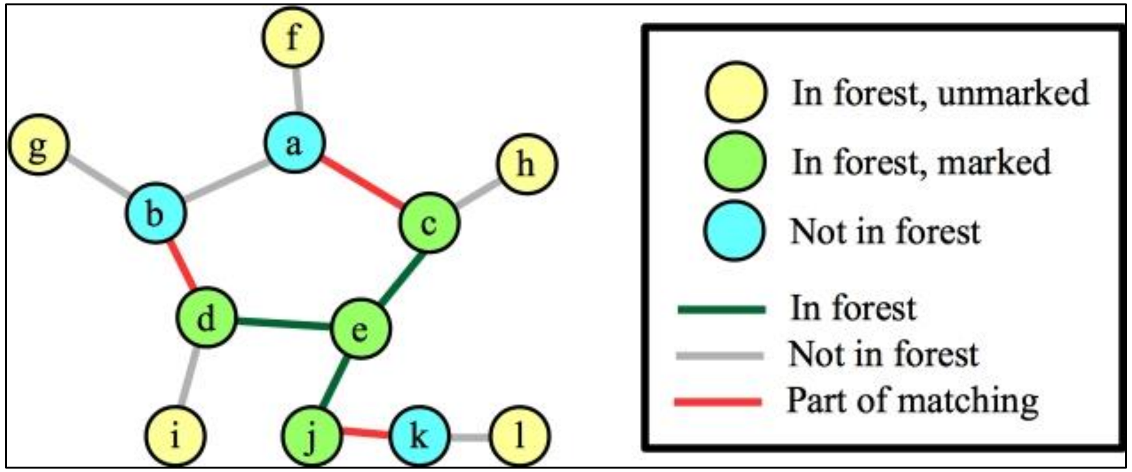
Example



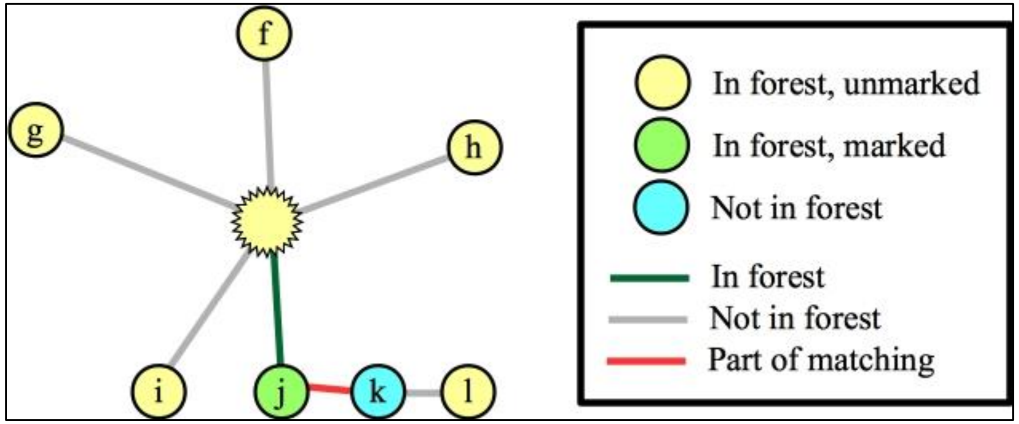
Example 2

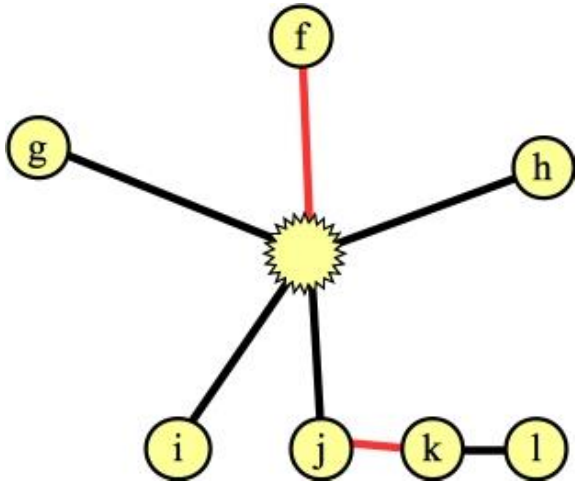

f
g
a
h
b
c
d
e
i
j
k
l

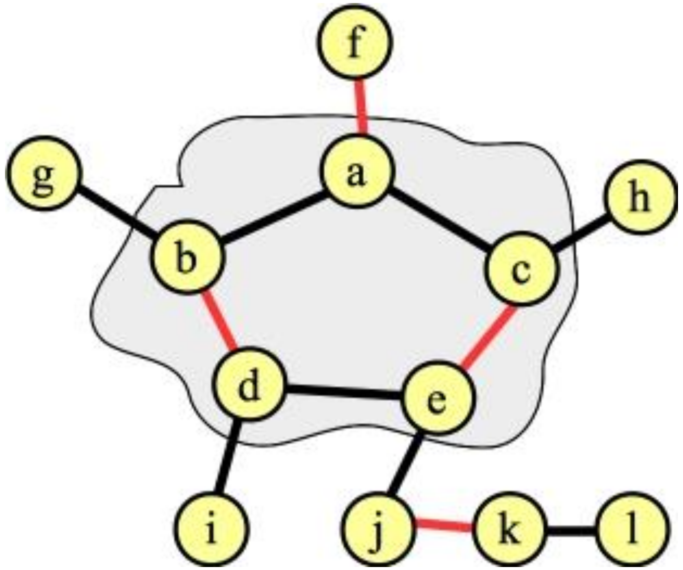

f
g
a
h
b
c
d
e
i
j
k
l
In forest, unmarked
In forest, marked
Not in forest
In forest
Not in forest
Part of matching


f
g
a
h
b
c
d
e
i
j
k
l
In forest, unmarked
In forest, marked
Not in forest
In forest
Not in forest
Part of matching

Example 2 (cont.)







Lecture 6: More on Connectivity

Shuai Li

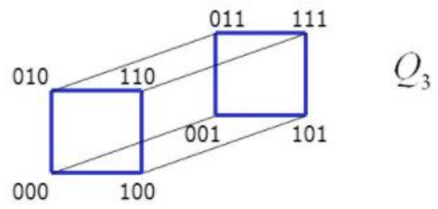
John Hopcroft Center, Shanghai Jiao Tong University

https://shuaili8.github.io

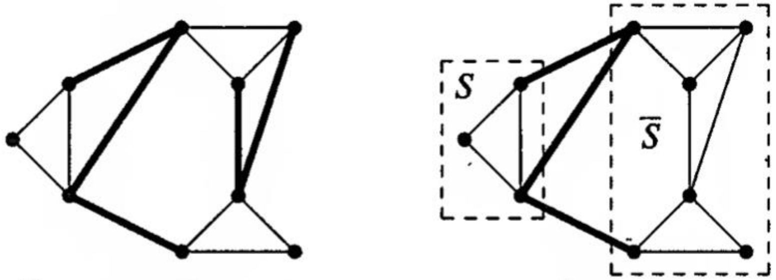
https://shuaili8.github.io/Teaching/CS445/index.html

Vertex cut set and connectivity

- A proper subset S of vertices is a vertex cut set if the graph $G - S$ is disconnected
- The connectivity, $\kappa(G)$, is the minimum size of a vertex set S of G such that $G - S$ is disconnected or has only one vertex
  - The graph is k-connected if $k \leq \kappa(G)$
- $\kappa(K_n) := n - 1$
- If G is disconnected, $\kappa(G) = 0$
  - $\Rightarrow$ A graph is connected $\Leftrightarrow \kappa(G) \geq 1$
- If G is connected, non-complete graph of order n, then
$$1 \leq \kappa(G) \leq n - 2$$
- For convention, $\kappa(K_1) = 0$
- Example (4.1.3, W) For k-dimensional cube $Q_k = \{0,1\}^k$, $\kappa(Q_k) = k$



Edge-connectivity



- A disconnecting set of edges is a set $F \subseteq E(G)$ such that $G - F$ has more than one component
  - A graph is k-edge-connected if every disconnecting set has at least k edges
  - The edge-connectivity of G, written $\lambda(G)$, is the minimum size of a disconnecting set
- Given $S, T \subseteq V(G)$, we write $[S, T]$ for the set of edges having one endpoint in S and the other in T
  - An edge cut is an edge set of the form $[S, S^c]$ where S is a nonempty proper subset of $V(G)$
- Every edge cut is a disconnecting set, but not vice versa
- Remark (4.1.8, W) Every minimal disconnecting set of edges is an edge cut

Connectivity and edge-connectivity

- Proposition (1.4.2, D) If G is non-trivial, then $\kappa(G) \leq \lambda(G) \leq \delta(G)$
- If $\delta(G) \geq n-2$, then $\kappa(G) = \delta(G)$

  that is $\kappa(G) = \lambda(G) = \delta(G)$

- Theorem (4.1.11, W) If G is a 3-regular graph, then $\kappa(G) = \lambda(G)$

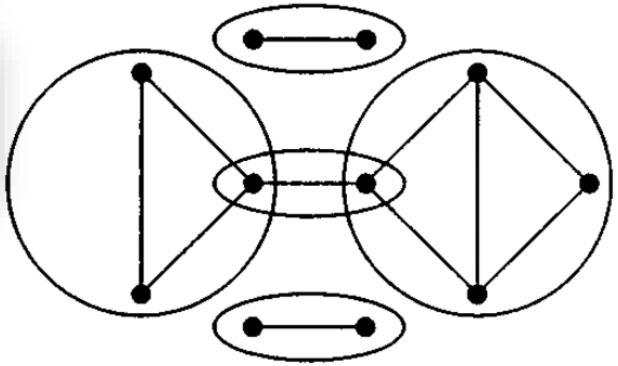
Properties of edge cut

- When $\lambda(G) < \delta(G)$, a minimum edge cut cannot isolate a vertex
- Similarly for (any) edge cut
- Proposition (4.1.12, W) If S is a set of vertices in a graph G, then
$$|[S, S^c]| = \sum_{v \in S} d(v) - 2e(G[S])$$
- Corollary (4.1.13, W) If G is a simple graph and $|[S, S^c]| < \delta(G)$, then $|S| > \delta(G)$
    - $|S|$ must be much larger than a single vertex

Blocks

- A block of a graph G is a maximal connected subgraph of G that has no cut-vertex. If G itself is connected and has no cut-vertex, then G is a block
- Example

Proposition (1.2.14, W)
An edge e is a bridge $\Leftrightarrow$ e lies on no cycle of G
- Or equivalently, an edge e is not a bridge $\Leftrightarrow$ e lies on a cycle of G



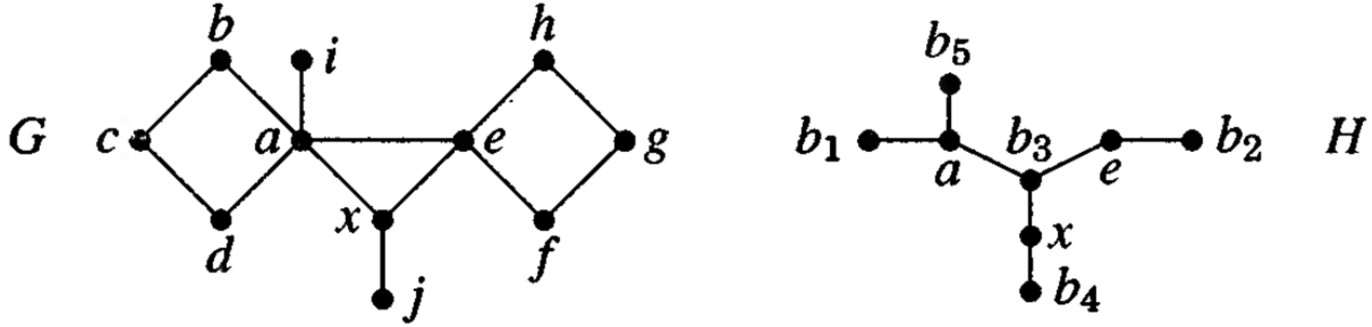
- An edge of a cycle cannot itself be a block
  - An edge is block $\Leftrightarrow$ it is a bridge
  - The blocks of a tree are its edges
- If a block has more than two vertices, then it is 2-connected
  - The blocks of a loopless graph are its isolated vertices, bridges, and its maximal 2-connected subgraphs

Intersection of two blocks

- Proposition (4.1.19, W) Two blocks in a graph share at most one vertex
  - When two blocks share a vertex, it must be a cut-vertex
- Every edge is a subgraph with no cut-vertex and hence is in a block. Thus blocks in a graph decompose the edge set

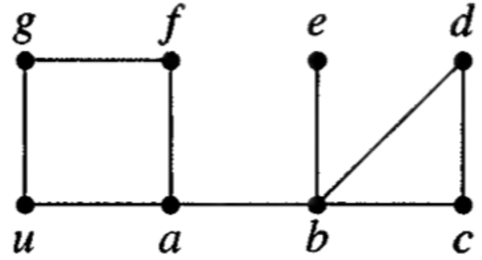
Block-cutpoint graph

- The block-cutpoint graph of a graph G is a bipartite graph H in which one partite set consists of the cut-vertices of G, and the other has a vertex b_i for each block B_i of G. We include vb_i as an edge of $H \Leftrightarrow v \in B_i$



- (Ex34, S4.1, W) When G is connected, its block-cutpoint graph is a tree

Depth-first search (DFS)



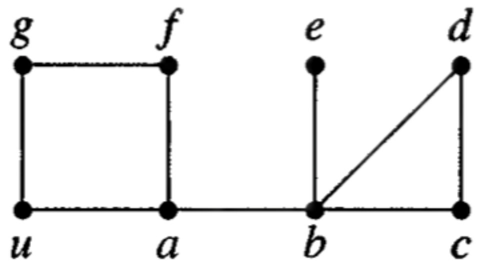
- Depth-first search

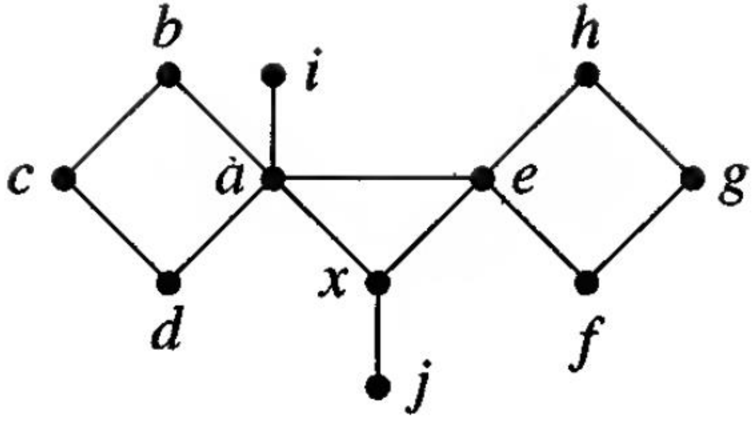
- Lemma (4.1.22, W) If T is a spanning tree of a connected graph grown by DFS from u, then every edge of G not in T consists of two vertices v, w such that v lies on the u, w-path in T

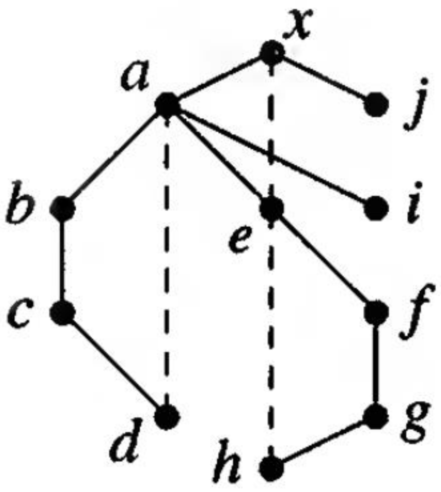
Finding blocks by DFS

- **Input**: A connected graph G
- **Idea**: Build a DFS tree T of G, discarding portions of T as blocks are identified. Maintain one vertex called ACTIVE
- **Initialization**: Pick a root $x \in V(H)$; make x ACTIVE; set $T = \{x\}$
- **Iteration**: Let v denote the current active vertex
  - If v has an unexplored incident edge vw, then
    - If $w \notin V(T)$, then add vw to T, mark vw explored, make w ACTIVE
    - If $w \in V(T)$, then w is an ancestor of v; mark vw explored
  - If v has no more unexplored incident edges, then
    - If $v \neq x$ and w is a parent of v, make w ACTIVE. If no vertex in the current subtree T' rooted at v has an explored edge to an ancestor above w, then $V(T') \cup \{w\}$ is the vertex set of a block; record this information and delete $V(T')$
    - if $v = x$, terminate

Example







Strong orientation

- Theorem (2.5, L; 4.2.14, W; Robbins 1939) A graph has a strong orientation, i.e. an orientation that is a strongly connected digraph $\Leftrightarrow$ it is 2-edge-connected
  - A directed graph is strongly connected if for every pair of vertices (v, w), there is a directed path from v to w
  - Proposition (2.4, L) Let $xy \in T$ which is not a bridge in G and x is a parent of y. Then there exists an edge in G but not in T joining some descendant a of y and some ancestor b of x

- The blocks of a loopless graph are its isolated vertices, bridges, and its maximal 2-connected subgraphs

Lemma (4.1.22, W) If T is a spanning tree of a connected graph grown by DFS from u, then every edge of G not in T consists of two vertices v, w such that v lies on the u, w-path in T