Chap 4 Connectivity and Flow

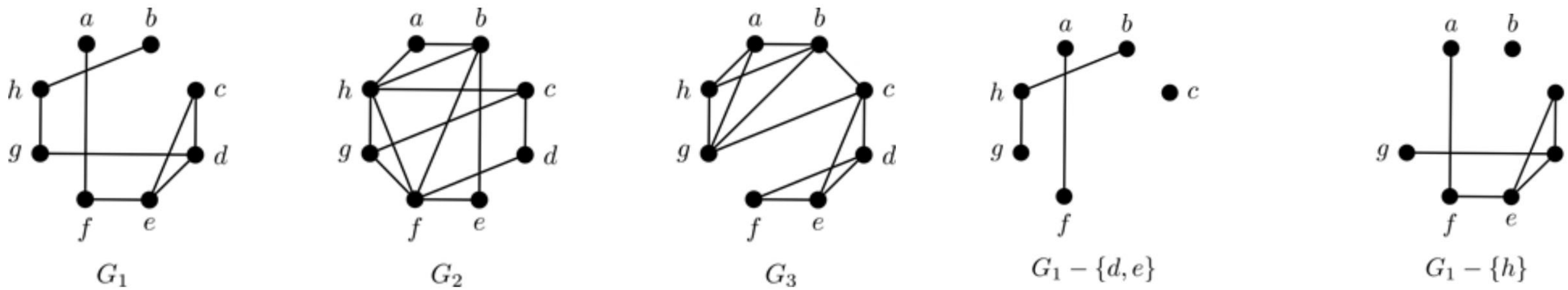
Yih-Lang Li (李毅郎)
Computer Science Department
National Yang-Ming Chiao-Tung University, Taiwan

The sources of most figure images are from the textbook

Outline

- Connectivity Measures
- Connectivity and Paths
- 2-Connected Graphs
- Network Flow
- Centrality Measures

4.1 Connectivity Measures



- **Definition 4.1** A *cut-vertex* of a graph G is a vertex v whose removal disconnects the graph, that is, G is connected but $G-v$ is not. A set S of vertices within a graph G is a *cut-set* if $G-S$ is disconnected.

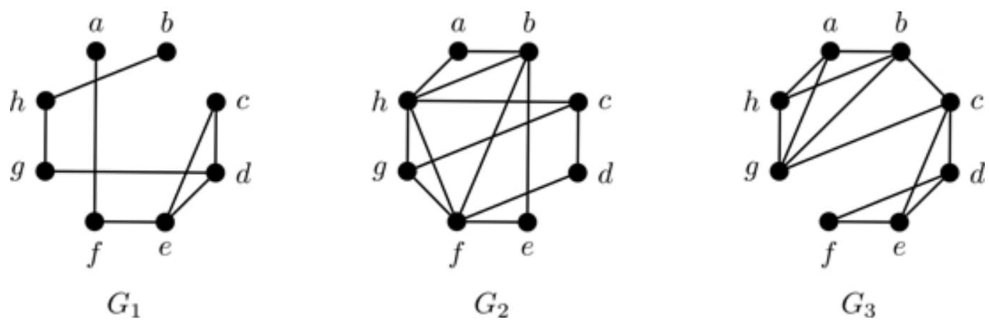
- **Definition 4.2** For any graph G, we say G is *k-connected* if the smallest cut-set is of size at least k.

  - ✓ Define the *connectivity* of G, $\kappa(G)=k$, to be the maximum k such that G is k-connected, that is there is a cut-set S of size k, yet no cut-set exists of size $k-1$ or less. Define $\kappa(K_n)=n-1$.

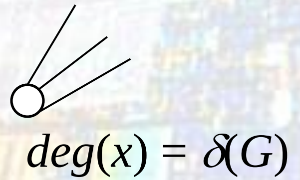
- **Example 4.1** Find $\kappa(G)$ for each of the graphs shown above (G_1, G_2, G_3).

k-Edge-Connected

- **Definition 4.3** A *bridge* in a graph $G=(V,E)$ is an edge e whose removal disconnects the graph, that is, G is connected but $G-e$ is not. An edge-cut is a set $F \subseteq E$ so that $G-F$ is disconnected.
- **Definition 4.4** We say G is *k-edge-connected* if the smallest edge-cut is of size at least k.
  - ✓ Define $\kappa'(G)=k$ to be the maximum k such that G is k-edge-connected, that is there exists a edge-cut F of size k, yet no edge-cut exists of size $k-1$.
- **Example 4.2** Find $\kappa'(G)$ for each of the graphs.

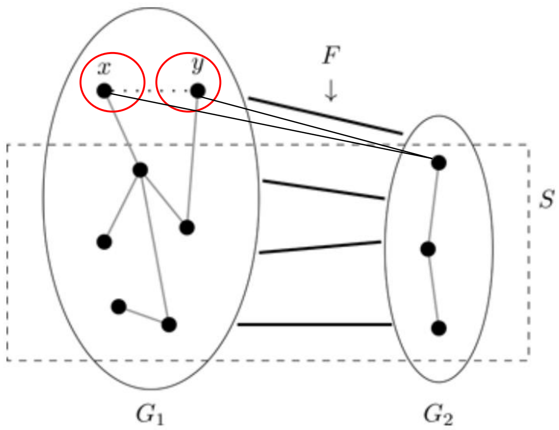


- **Theorem 4.5** (*Whitney's Theorem*) For any graph G, $\kappa(G) \le \kappa'(G) \le \delta(G)$.
  - ✓ $\kappa'(G) \le \delta(G)$
  - ✓ For complete graphs, $\kappa(G) = \kappa'(G)$



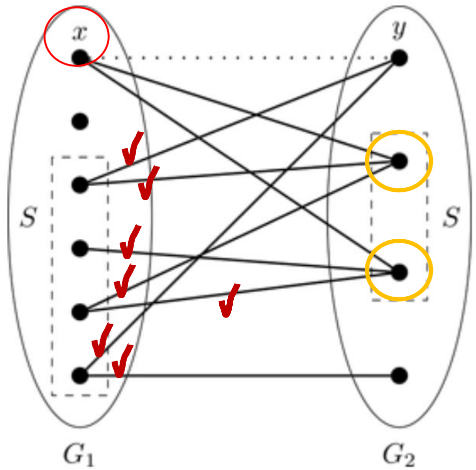
4.2 Connectivity and Paths

✓ For non-complete graphs, let F be the minimum edge cut set.



Complete connection between G_1 and G_2

$\kappa(G) \leq n-2 < n-1 \leq \kappa'(G)$.



$|S| \leq |F|$, G-S is disconnected

$\kappa(G) \leq |S| \leq |F| \leq \kappa'(G)$

✓ E_2 in F without x as its endpoint

E_1 in F has an endpoint of x

$E_1 \cup E_2 = F$

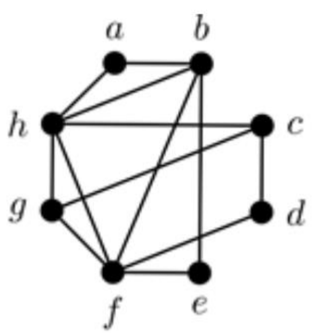
$E_1 : S \cap G_2 = 1 : 1$

$E_2 : S \cap G_1 = n : 1 \ (n \geq 1)$

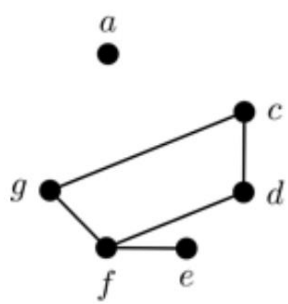
- **Theorem 4.6** A vertex v is a cut-vertex of a graph G if and only if there exist vertices x and y such that v is on every $x-y$ path.
- **Theorem 4.7** An edge e is a bridge of G if and only if there exist vertices x and y such that e is on every $x-y$ path.
- **Theorem 4.8** Every nontrivial connected graph contains at least two vertices that are not cut-vertices.

Connectivity and Paths

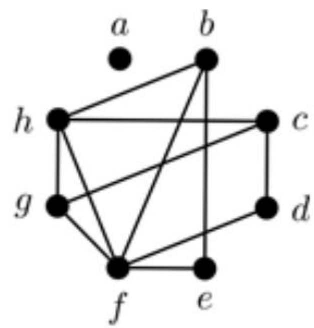
- **Theorem 4.9** An edge *e* is a bridge of *G* if and only if *e* lies on no cycle of *G*.
- **Definition 4.10** Let P_1 and P_2 be two paths within the same graph *G*. We say these paths are
  - ✓ *disjoint* if they have no vertices or edges in common.
  - ✓ *internally disjoint* if the only vertices in common are the starting and ending vertices of the paths.
  - ✓ *edge-disjoint* if they have no edges in common.
- Two edge-disjoint paths must be internally disjoint?
- **Definition 4.11** Let *x* and *y* be two vertices in a graph *G*. A set *S* (of either vertices or edges) separates *x* and *y* if *x* and *y* are in different components of $G-S$. When this happens, we say *S* is a separating set for *x* and *y*.



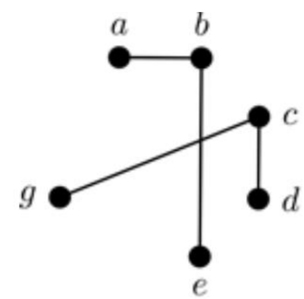
G_2



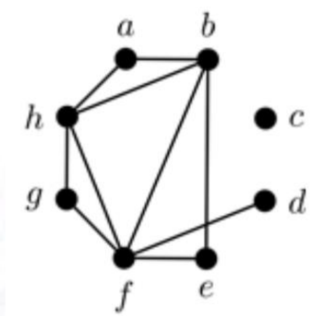
$G_2 - \{b, h\}$



$G_2 - \{ab, ah\}$



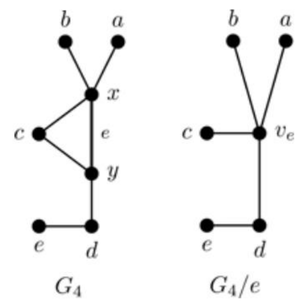
$G_2 - \{f, h\}$



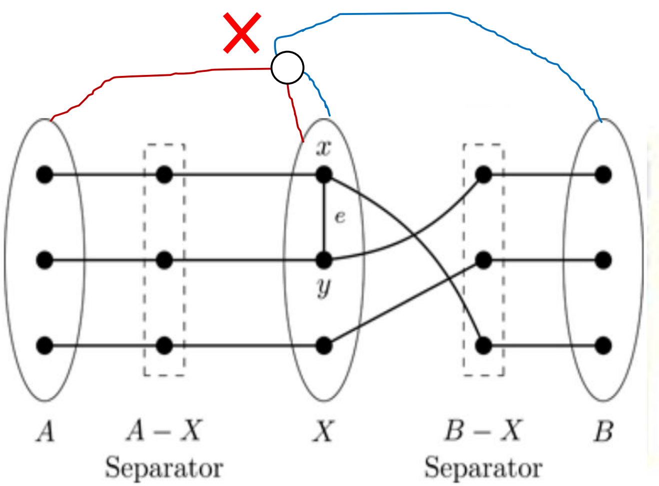
$G_2 - \{cd, cg, ch\}$

Menger's Theorem

- **Definition 4.12** Let $e=xy$ be an edge of a graph G. The contraction of e, denoted G/e, replaces the edge e with a vertex v_e so that any vertices adjacent to either x or y are now adjacent to v_e.

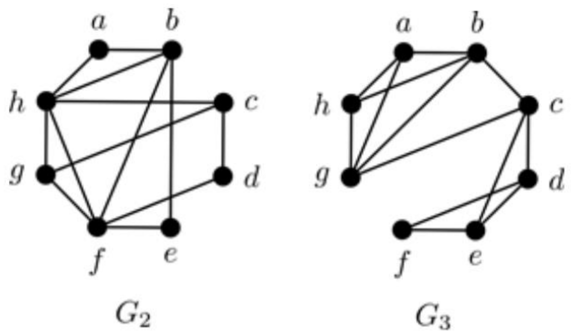


- **Theorem 4.13** (*Menger's Theorem*) Let x and y be nonadjacent vertices in G. Then the minimum number of vertices that separate x and y ($mivs(x,y)$) equals the maximum number of internally disjoint $x-y$ paths ($maidp(x,y)$) in G.
  - ✓ Let $k = mivs(A,B)$. Prove by induction on E
  - ✓ It holds for three vertices connected by two edges
  - ✓ Assume $maidp_G(A,B) < k$, neither for G/e
  - ✓ By IH, $|Y|=mivs_{G/e}(A,B)=maidp_{G/e}(A,B)<k$ and $v_e \in Y$
  - ✓ Let $X = Y - \{v_e\} \cup \{x, y\}$, then X is a $mivs_G(A, B)$
  - ✓ $mivs_{G-e}(A,X)=mivs_G(A,B)=k \rightarrow maidp_{G-e}(A, X)=k$
  - ✓ Similarly for $maidp_{G-e}(B, X)=k$



Menger's Theorem

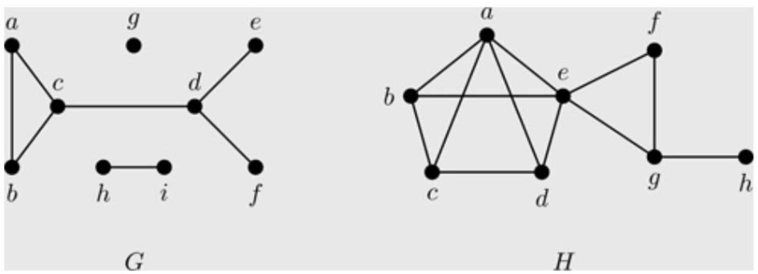
- **Theorem 4.14** A nontrivial graph G is k-connected if and only if for each pair of distinct vertices x and y there are at least k internally disjoint $x-y$ paths.
- *Edge version*
- **Theorem 4.15** Let x and y be distinct vertices in G. Then the minimum number of edges that separate x and y equals the maximum number of edge-disjoint $x-y$ paths in G.
- **Theorem 4.16** A nontrivial graph G is k-edge-connected if and only if for each pair of distinct vertices x and y there are at least k edge disjoint $x-y$ paths.

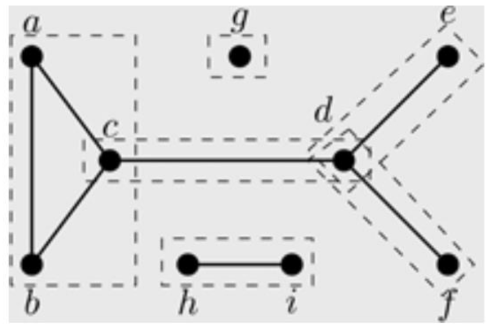


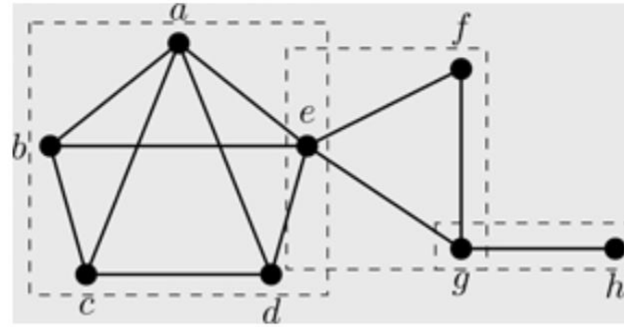
- **4.3 2-Connected Graphs**
- **Theorem 4.17** A graph G with at least 3 vertices is 2-connected if and only if G is connected and does not have any cut-vertices.
- **Corollary 4.18** A graph G with at least 3 vertices is 2-connected if and only if for every pair of vertices x and y there exists a cycle through x and y.

4.3 2-Connected Components

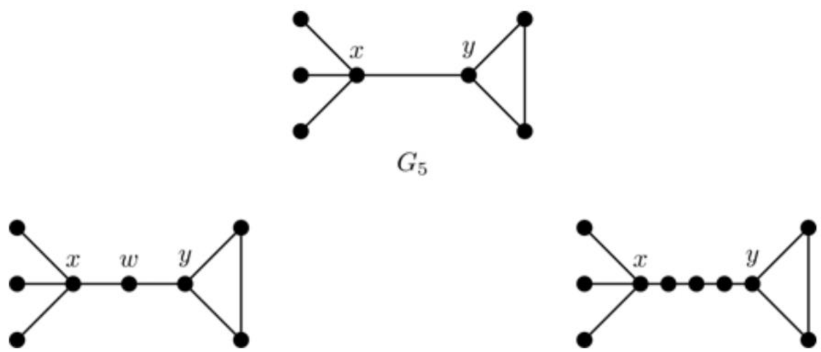
- **Definition 4.19** A block of a graph G is a maximal 2-connected subgraph of G, that is, a subgraph with as many edges as possible without a cut-vertex.
- **Example 4.3** Determine all blocks for the two graphs below.







- **Definition 4.20** Let $e=xy$ be an edge in a graph G. The subdivision of e adds a vertex v in the edge so as to replace it with the path $x\ v\ y$.

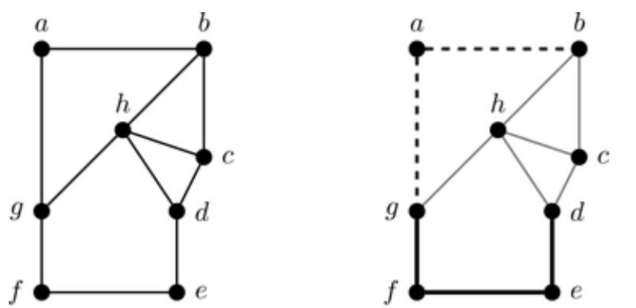


2-Connected Components

- **Theorem 4.21** Let G be a graph and G' the graph obtained by subdividing any edge of G. Then G is 2-connected if and only if G' is 2-connected.
  - ✓ $\rightarrow$ G is 2-connected, there is a cycle between any two vertices.
  - ✓ $\leftarrow$ if G is not 2-connected, there is a cut vertex v for the connection of some two vertices

G' $G'-v$ G' $G'-v$

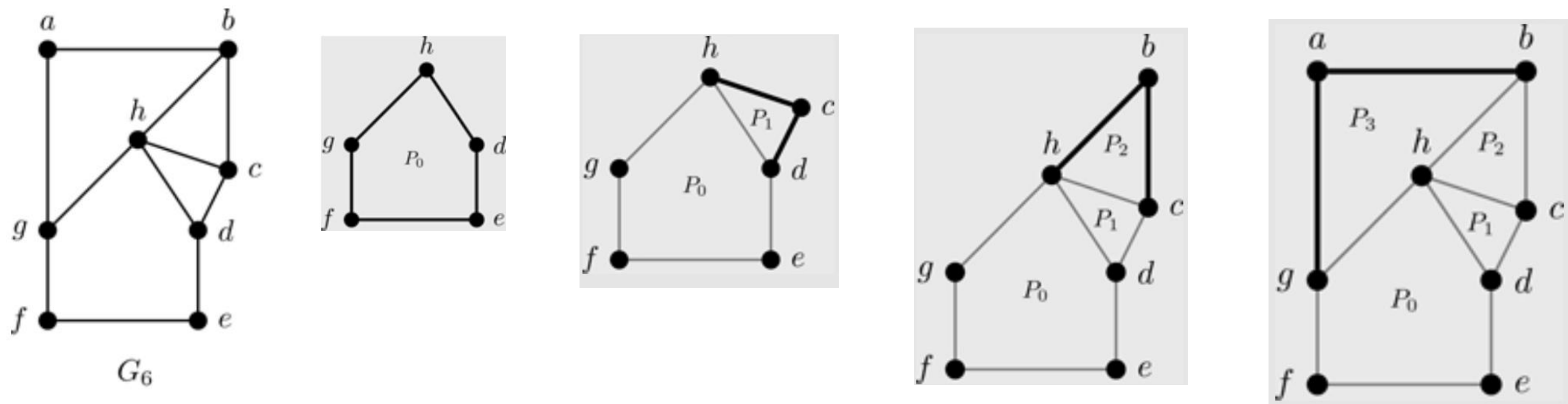
- **Definition 4.22** An ear of a graph is a path P that is contained within a cycle where only the endpoints of P can have degree more than 2 in the graph.



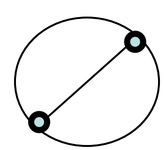
G_6 G_6 with ears highlighted

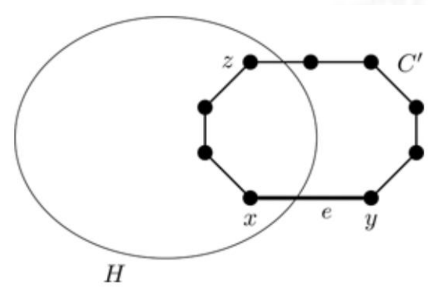
2-Connected Components

- **Definition 4.23** An ear decomposition of a graph G is a collection $P_0, P_1, \ldots P_k$ so that P_0 is a cycle, P_i is an ear of $P_0 \cup \cdots \cup P_{i-1}$ for all $i \geq 1$, and all edges and vertices are included in the collection.
- **Example 4.4** Find an ear decomposition of the graph G_6 shown above.



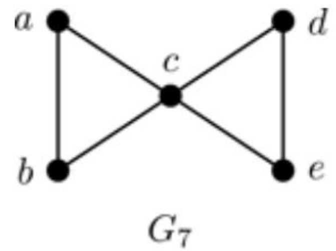
- **Theorem 4.24** A graph G is 2-connected if and only if it has an ear decomposition.
  - ✓ 2-connect→ a cycle → ear addition (H is maximal)
  - ✓ Conversely, cycle-> ear addition,

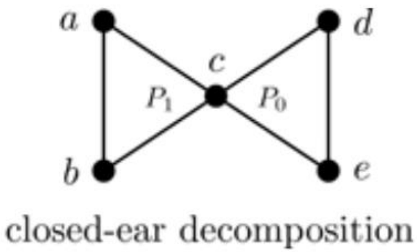




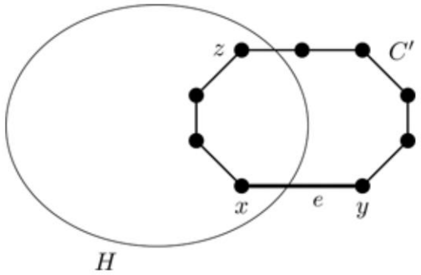
2-Edge-Connected

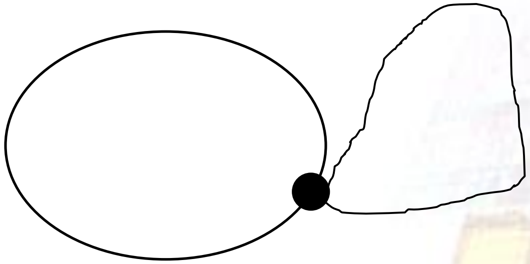
- **Definition 4.25** A *closed-ear* in a graph G is a cycle where all vertices have degree 2 in G except for one vertex on the cycle. A *closed-ear decomposition* is a collection $P_0, P_1, \ldots P_k$ so that P_0 is a cycle, P_i is either an ear or closed-ear of $P_0 \cup \cdots \cup P_{i-1}$ for all $i \geq 1$, and all edges and vertices are included in the collection.





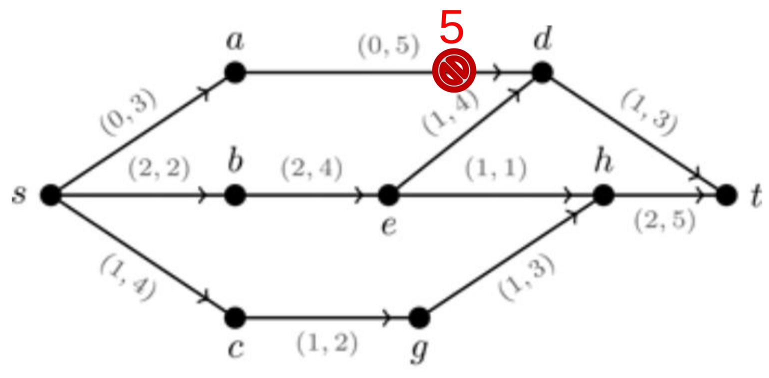
- **Theorem 4.26** A graph G is 2-edge-connected if and only if it has a closed-ear decomposition.





4.4 Network Flow

- **Definition 4.27** A network is a digraph where each arc e has an associated nonnegative integer $c(e)$, called a *capacity*. In addition, the network has a designated starting vertex s, called the *source*, and a designated ending vertex t, called the *sink*. A *flow f* is a function that assigns a value $f(e)$ to each arc of the network.



G_8

- **Definition 4.28** For a vertex v, let $f^-(v)$ represent the total flow entering v and $f^+(v)$ represent the total flow exiting v. A flow is *feasible* if it satisfies the following conditions: (1) $f(e)\geq 0$ for all edges e. (2) $f(e)\leq c(e)$ for all edges e. (3) $f^+(v)=f^-(v)$ for all vertices other than s and t. (4) $f^-(s)=f^+(t)=0$.

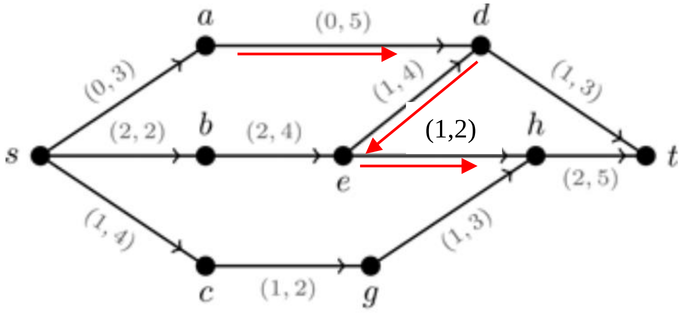
- **Definition 4.29** The value of a flow is defined as $|f|=f^+(s)=f^-(t)$, that is, the amount exiting the source which must also equal the flow entering the sink. A *maximum flow* is a feasible flow of largest value.

Network Flow

- **Definition 4.30** Let f be a flow along a network. The *slack* k of an arc is the difference between its capacity and flow; that is, $k(e)=c(e)-f(e)$.
- **Definition 4.31** A chain K is a path in a digraph where the direction of the arcs are ignored.
  - ✓ *sadeht* and *sadt*
- Vertices will be assigned two-part labels that aid in the creation of a chain on which the flow can be increased.
  - ✓ The first part of the label for vertex y will indicate one of two possibilities:

    x^- if there is a ***positive flow*** along $y \to x$,

    x^+ if there is ***slack*** along the arc $x \to y$,
  - ✓ The second part of the label will indicate the amount of flow that could be adjusted along the arc in question.



G_8

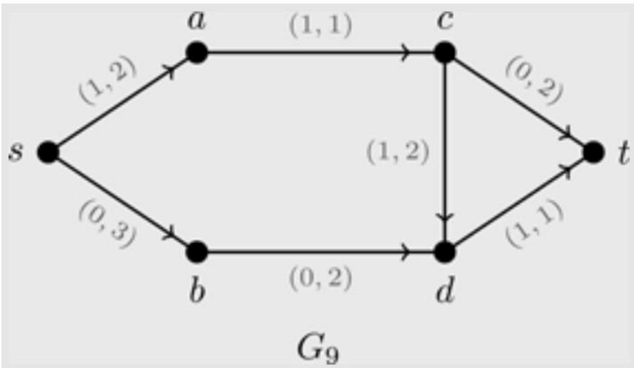
Network Flow

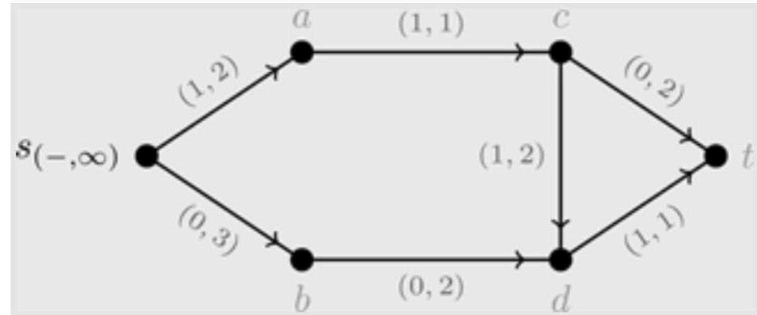
Augmenting Flow Algorithm

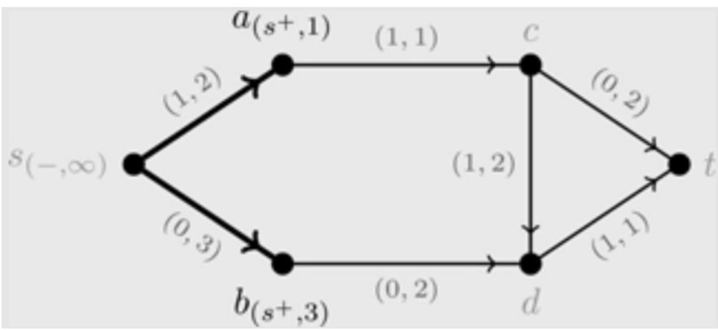
- **Input:** Network $G=(V, E, c)$, with designated source s and sink t, and each arc is given a capacity.
- **Steps:**
  1. Label s with $(-,\infty)$.
  2. Choose a labeled vertex x.
     - (a) For any arc yx, if $f(yx)>0$ and y is unlabeled, then label y with $(x^-,\sigma(y))$ where $\sigma(y)=\min\{\sigma(x),f(yx)\}$.
     - (b) For any arc xy, if $k(xy)>0$ and y is unlabeled, then label y with $(x^+,\sigma(y))$ where $\sigma(y)=\min\{\sigma(x),k(xy)\}$.
  3. If t has been labeled, go to Step (4). Otherwise, choose a different labeled vertex that has not been scanned and go to Step (2). If all labeled vertices have been scanned, then f is a maximum flow.
  4. Find an $s-t$ chain K of slack edges by backtracking from t to s. Along the edges of K, increase the flow by $\sigma(t)$ units if they are in the forward direction and decrease by $\sigma(t)$ units if they are in the backward direction. Remove all vertex labels except that of s and return to Step (2).
- **Output:** Maximum flow f.

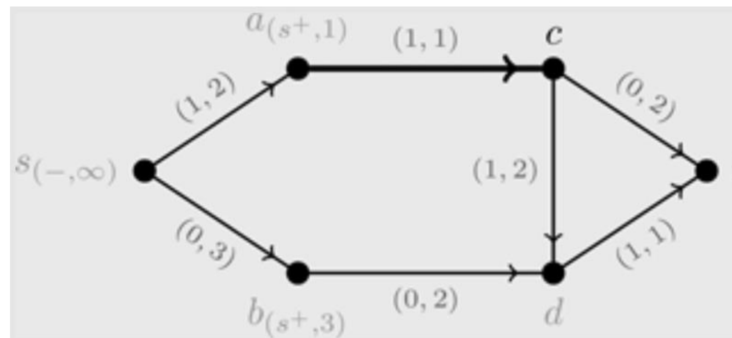
Network Flow

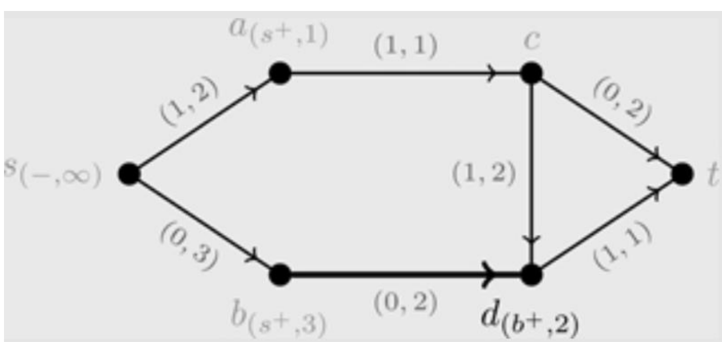
- **Example 4.5** Apply the Augmenting Flow Algorithm to the network G_9 shown below.

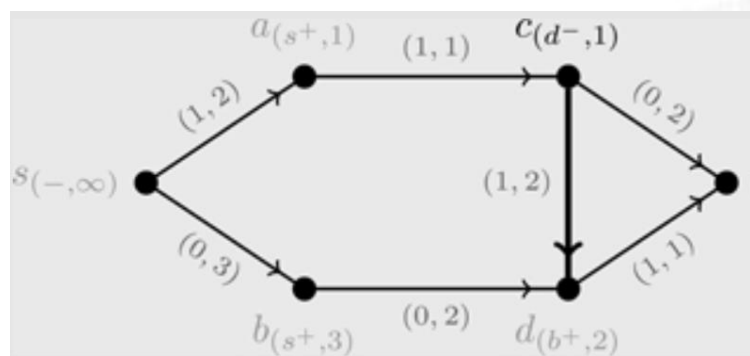




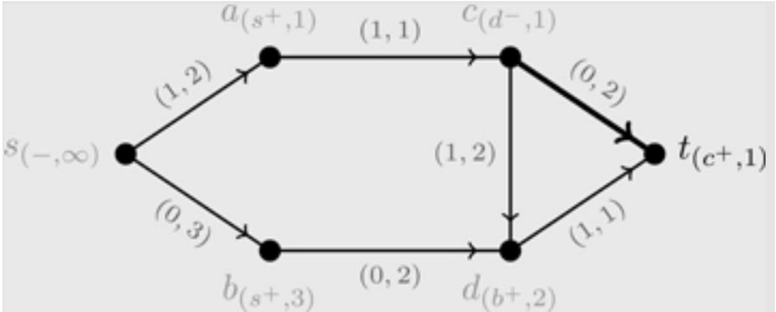


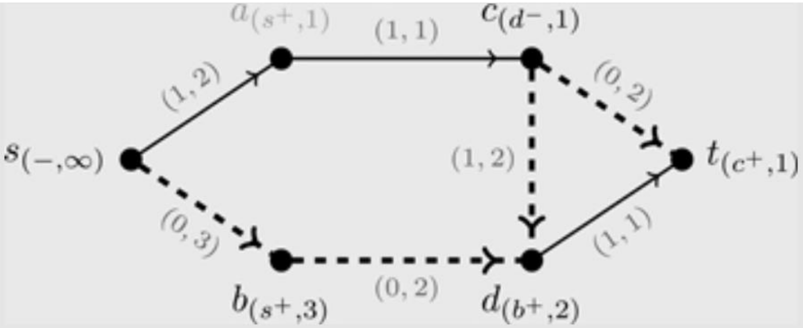


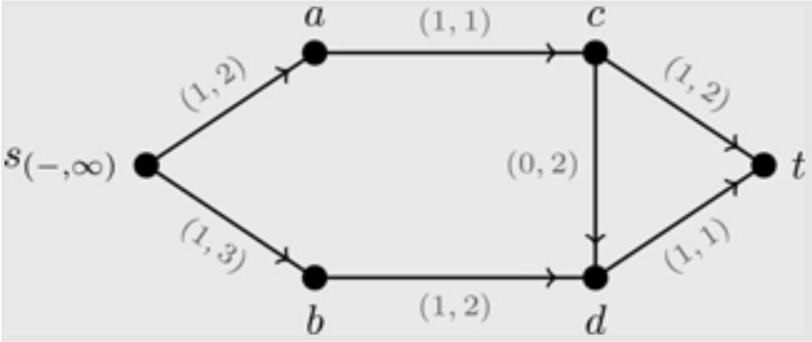


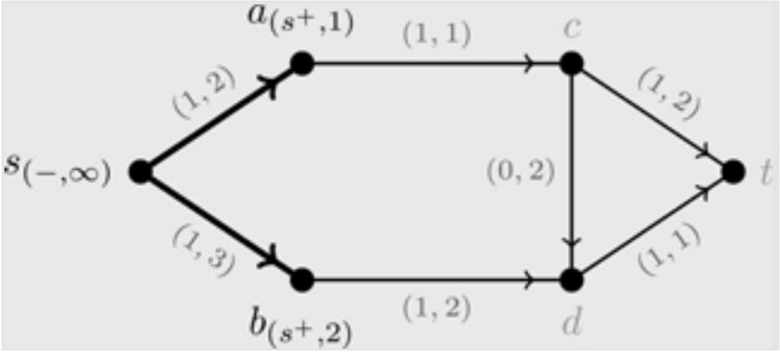


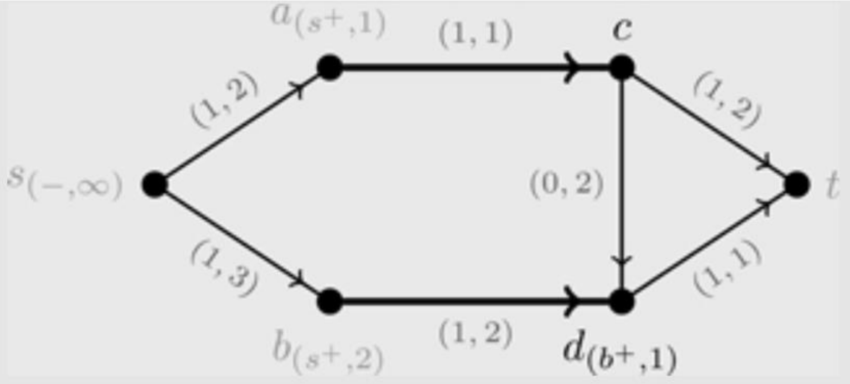
Network Flow





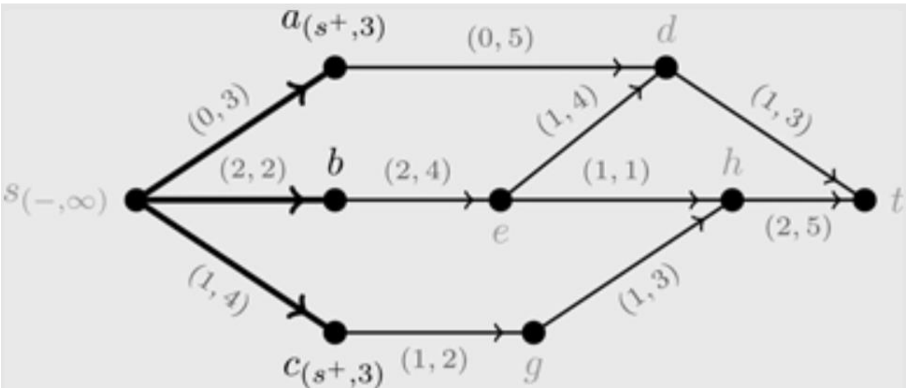


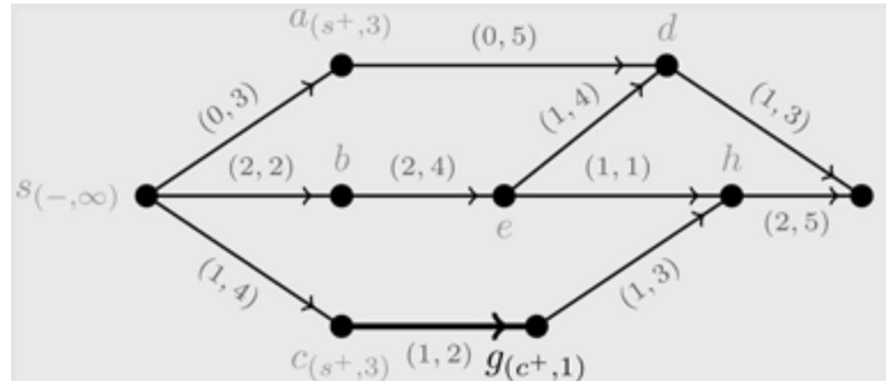


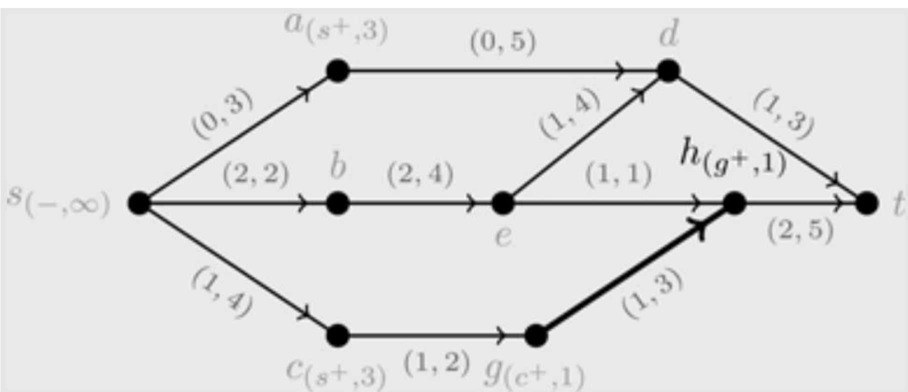


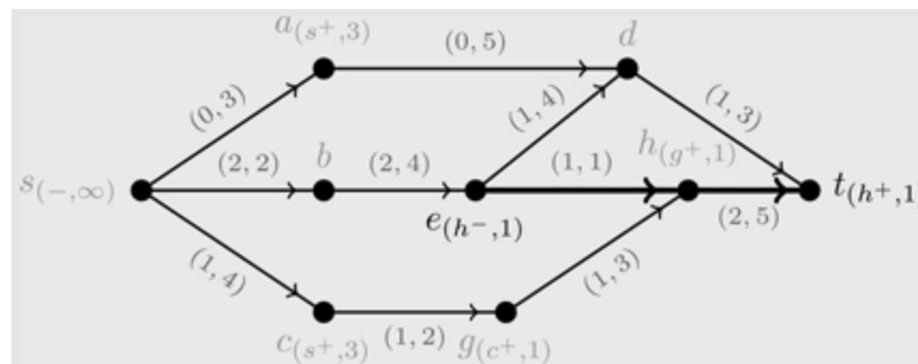
Network Flow

Example 4.6 Apply the Augmenting Flow Algorithm to the network G_8.

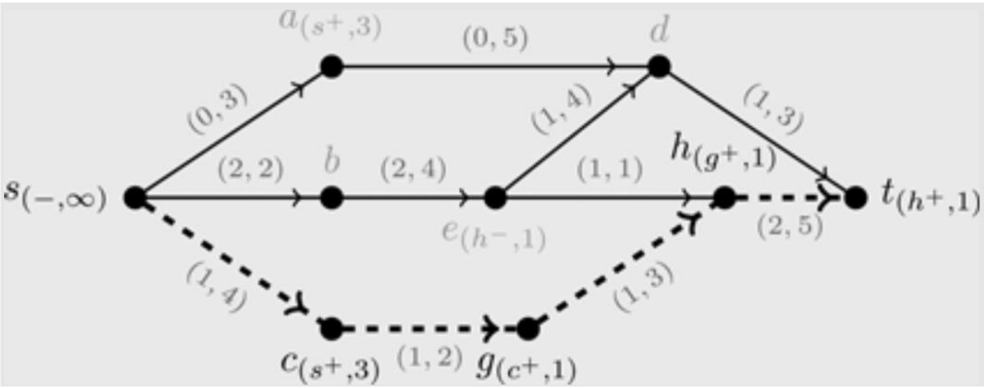


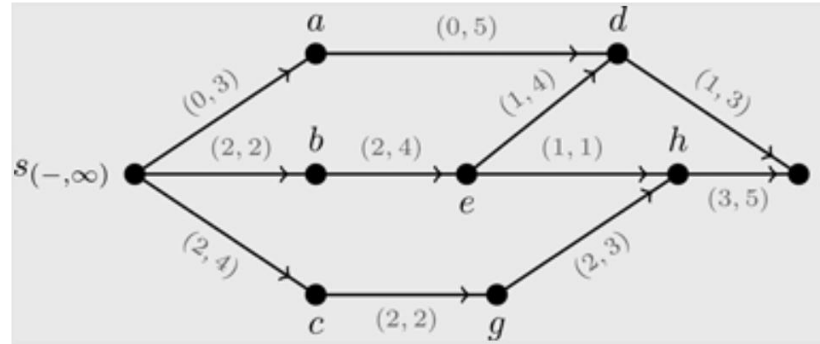


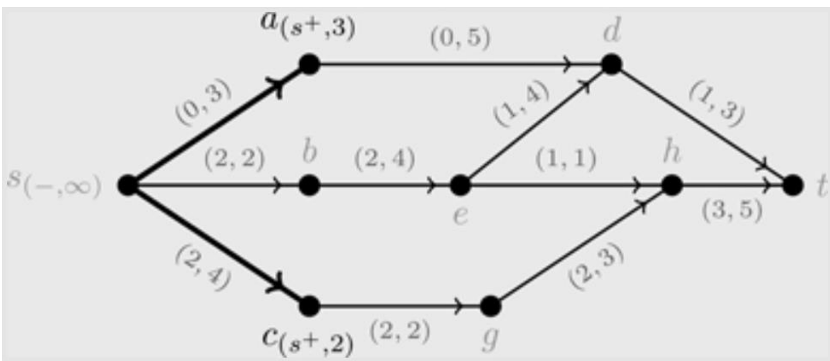


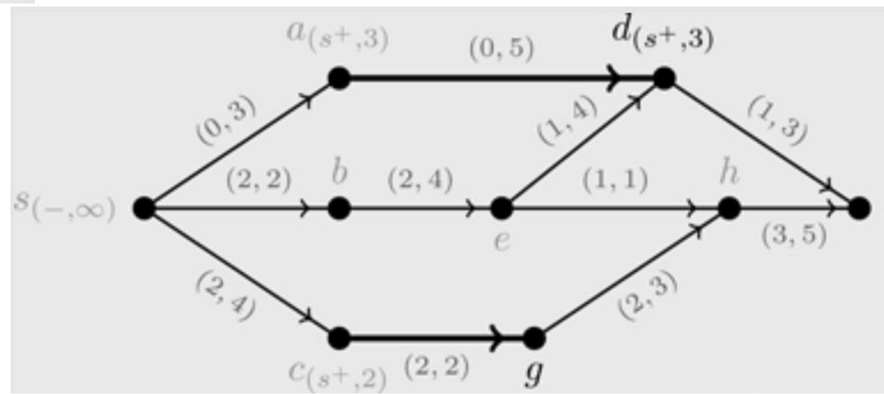


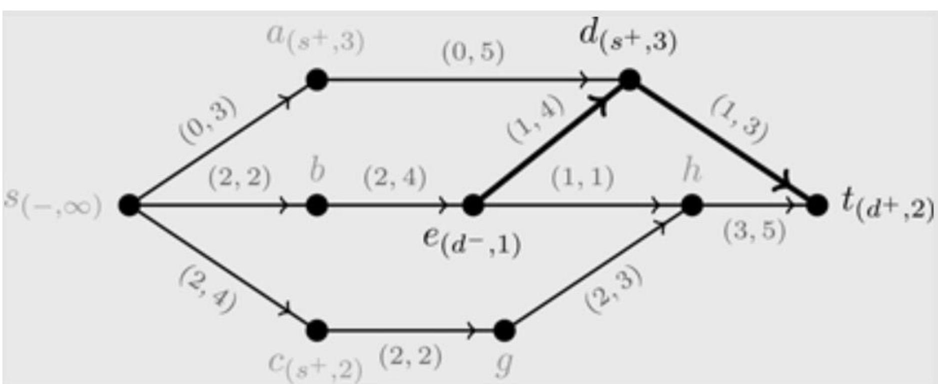
Network Flow

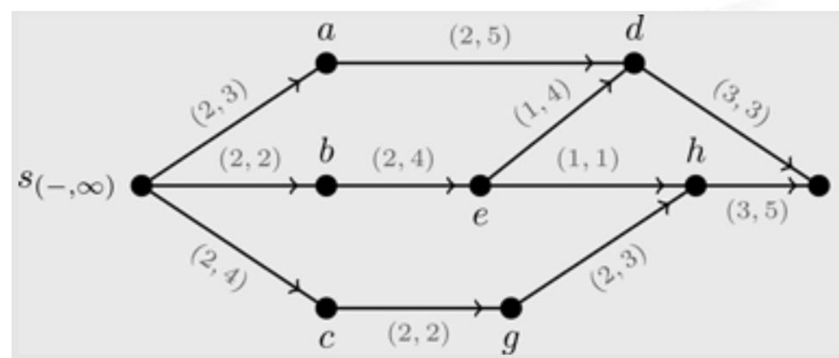




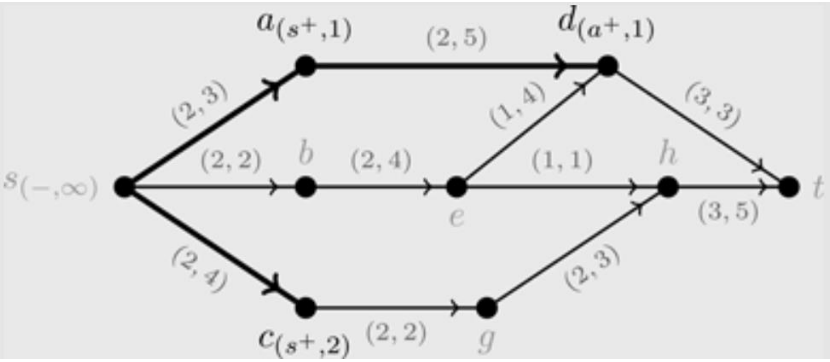


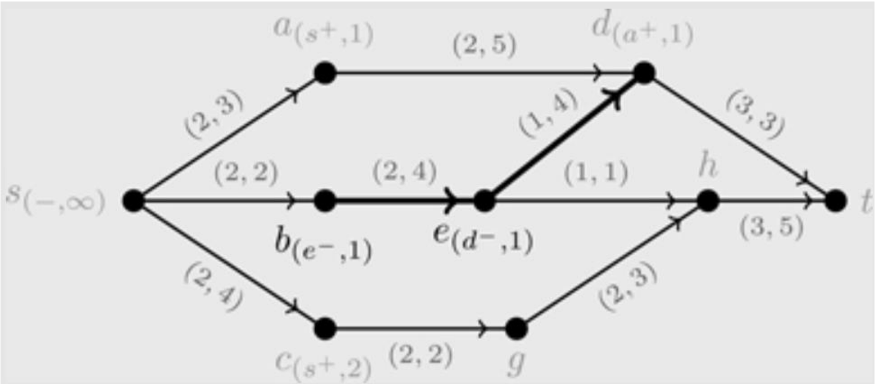




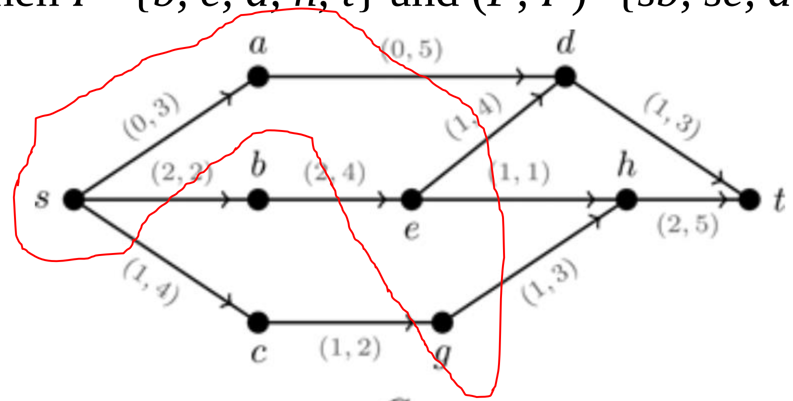


Network Flow





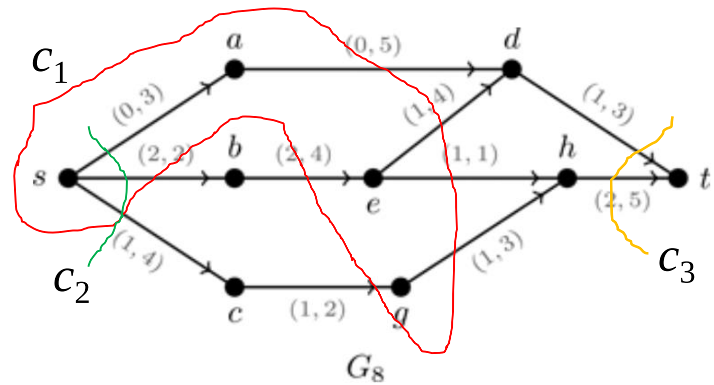
- **Definition 4.32** Let P be a set of vertices and $\bar{P}$ denote those vertices not in P (called the complement of P). A cut $(P, \bar{P})$ is the set of all arcs xy where x is a vertex from P and y is a vertex from $\bar{P}$. An *s−t cut* is a cut in which the source s is in P and the sink t is in $\bar{P}$.
  - ✓ Let P={s, a, e, g} then $\bar{P}$={b, c, d, h, t} and $(P, \bar{P})$={sb, sc, ad, ed, eh, gh} (not include be and cg).



G_8

Network Flow

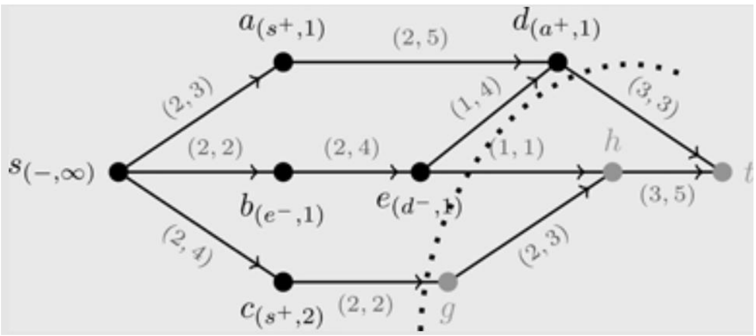
- **Definition 4.33** The *capacity* of a cut, $c(P, \bar{P})$, is defined as the sum of the capacities of the arcs that comprise the cut.
  - ✓ $c_1 = 19$, $c_2 = 9$, $c_3 = 8$

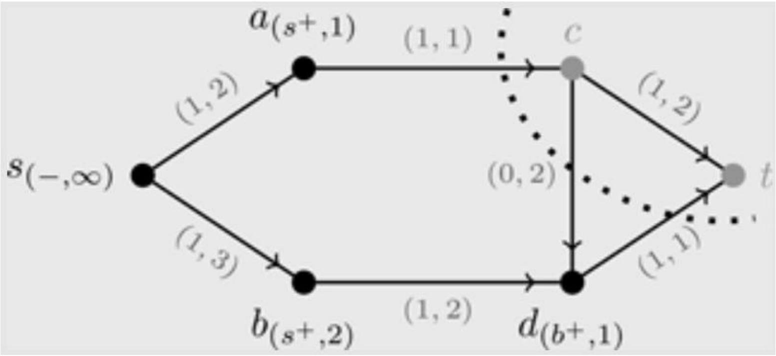


- **Theorem 4.34** (*Max Flow–Min Cut*) In any directed network, the value of a maximum $s\!-\!t$ flow equals the capacity of a minimum $s\!-\!t$ cut.
- **Min-Cut Method**
  - ✓ Let $G=(V, A, c)$ be a network with a designated source s and sink t and each arc is given a capacity c.
  - ✓ Apply the Augmenting Flow Algorithm.
  - ✓ Define an $s\!-\!t$ cut $(P, \bar{P})$ where P is **the set of labeled vertices from the final implementation** of the algorithm.
  - ✓ $(P, \bar{P})$ is a minimum $s\!-\!t$ cut for G.

Network Flow

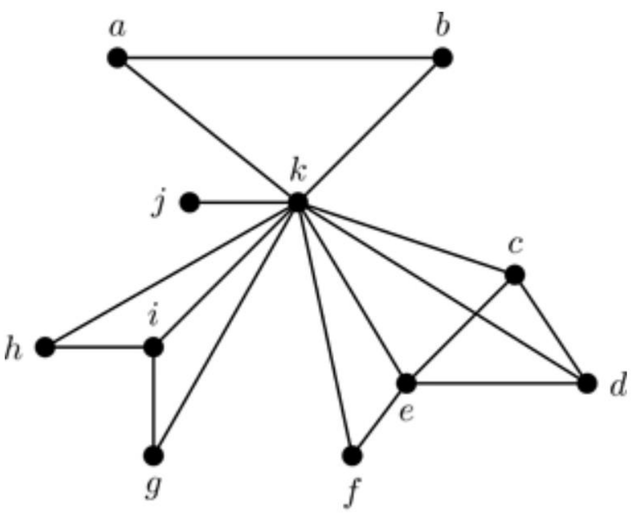
- **Example 4.7** Use the Min-Cut Method to find a minimum $s-t$ cut for the network G_8 and the network G_9.





Centrality Measures

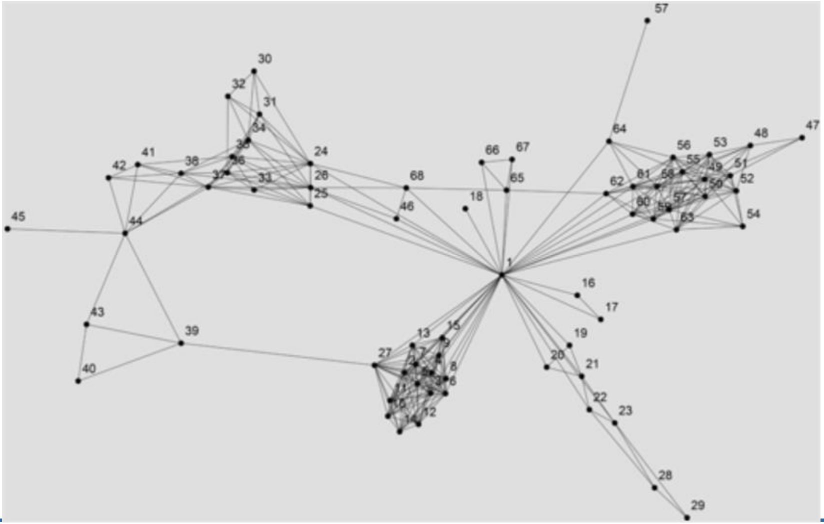
- Instead of only relying on path distance, we may want to characterize vertices based on other metrics indicating their relative importance within a graph. These measures are often called *network centralities*, where network here simply means a connected graph.
- **Definition 4.35** Given a graph G, the degree centrality of a vertex v is defined as $C_d(v)=deg(v)$.
- **Definition 4.36** Given a graph G, the closeness centrality of a vertex v is defined as $C_c(v)=\frac{1}{k}\sum_y \frac{1}{d(v,y)}$, where k is the number of vertices connected to v.
  - ✓ $C_c(v)=0$ as v is an isolated vertex, and $C_c(v)=1$ if v is adjacent to every vertex.
  - ✓ $a=b=f=g=h=0.6$, $c=d=i=0.65$, $j=0.55$, $k=1$



G_{10}

Centrality Measures

- **Definition 4.37** Given a graph G, the *betweenness centrality* of a vertex v is defined as $C_b(v) = \frac{1}{2}\sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}}$,
  - ✓ the sum is taken over all distinct pairs s and t,
  - ✓ σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of these paths that pass through v. Note: we set this ratio to be 0 if there are no paths from s to t.
  - ✓ $C_b(v)$ ranges from 0 to $(n-1)(n-2)/2$
- **Example 4.8** The graph below represents a friendship network. Visually, it would appear that vertex 1 seems to play a central role here. We will verify this using the centrality measures listed above.



C_d		C_c		C_b	
1	35	1	0.624	1	1670
49	16	27	0.466	24	255
27	16	26	0.463	26	238
*	15	24	0.463	27	217
*	15	58	0.453	39	160
*	15	25	0.450	58	141
*	15	57	0.448	44	125
*	15	**	0.444	25	117