



# Design and Analysis of Algorithms

## Network Flow

**Si Wu**

School of CSE, SCUT

[cswusi@scut.edu.cn](mailto:cswusi@scut.edu.cn)

TA: [1684350406@qq.com](mailto:1684350406@qq.com)



# Topics

- **Image Segmentation**

(<https://www.v7labs.com/blog/semantic-segmentation-guide>)

- **Bipartite Matching**

- **Disjoint Paths**



# Image Segmentation

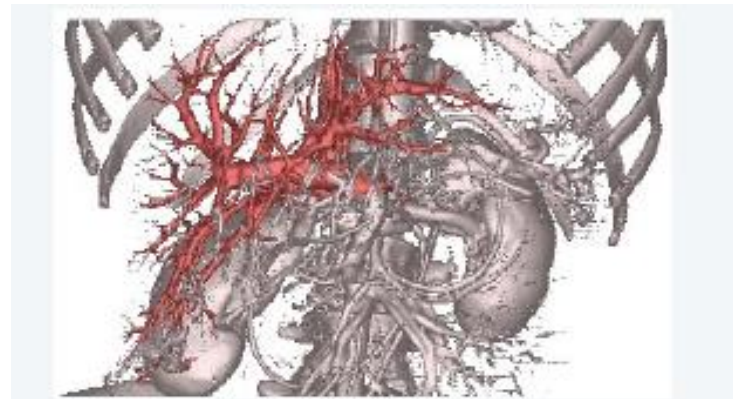
## Image segmentation.

- Central problem in image processing.
- Divide image into coherent regions.

**Ex.** Three people standing in front of complex background scene. Identify each person as a coherent object.



Semantic segmentation



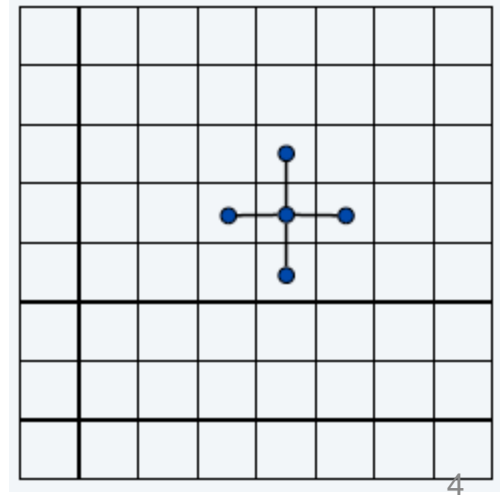
liver and hepatic vascularization segmentation



# Image Segmentation

## Foreground/background segmentation.

- Label each pixel in picture as belonging to foreground or background.
- $V$  = set of pixels,  $E$  = pairs of neighboring pixels.
- $a_i \geq 0$  is likelihood pixel  $i$  in foreground.
- $b_i \geq 0$  is likelihood pixel  $i$  in background.
- $p_{ij} \geq 0$  is separation penalty for labeling one of  $i$  and  $j$  as foreground, and the other as background.





# Image Segmentation

## Foreground/background segmentation.

- Label each pixel in picture as belonging to foreground or background.
- $V$  = set of pixels,  $E$  = pairs of neighboring pixels.
- $a_i \geq 0$  is likelihood pixel  $i$  in foreground.
- $b_i \geq 0$  is likelihood pixel  $i$  in background.
- $p_{ij} \geq 0$  is separation penalty for labeling one of  $i$  and  $j$  as foreground, and the other as background.

## Goals.

- Accuracy: if  $a_i > b_i$  in isolation, prefer to label  $i$  in foreground.
- Smoothness: if many neighbors of  $i$  are labeled foreground, we should be inclined to label  $i$  as foreground.
- Find partition  $(A, B)$  that maximizes:

$$\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E, |A \cap \{i,j\}|=1} p_{ij}$$



# Image Segmentation

Formulate as min-cut problem.

- Maximization
- No source or sink.
- Undirected graph.

Turn into minimization problem.

- Maximizing  $\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E, |A \cap \{i,j\}|=1} p_{ij}$
- Is equivalent to minimizing

$$\sum_{i \in V} a_i + \sum_{j \in V} b_j - \sum_{i \in A} a_i - \sum_{j \in B} b_j + \sum_{(i,j) \in E, |A \cap \{i,j\}|=1} p_{ij}$$

- Or alternatively  $\sum_{j \in B} a_j + \sum_{i \in A} b_i + \sum_{(i,j) \in E, |A \cap \{i,j\}|=1} p_{ij}$

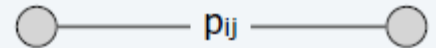


# Image Segmentation

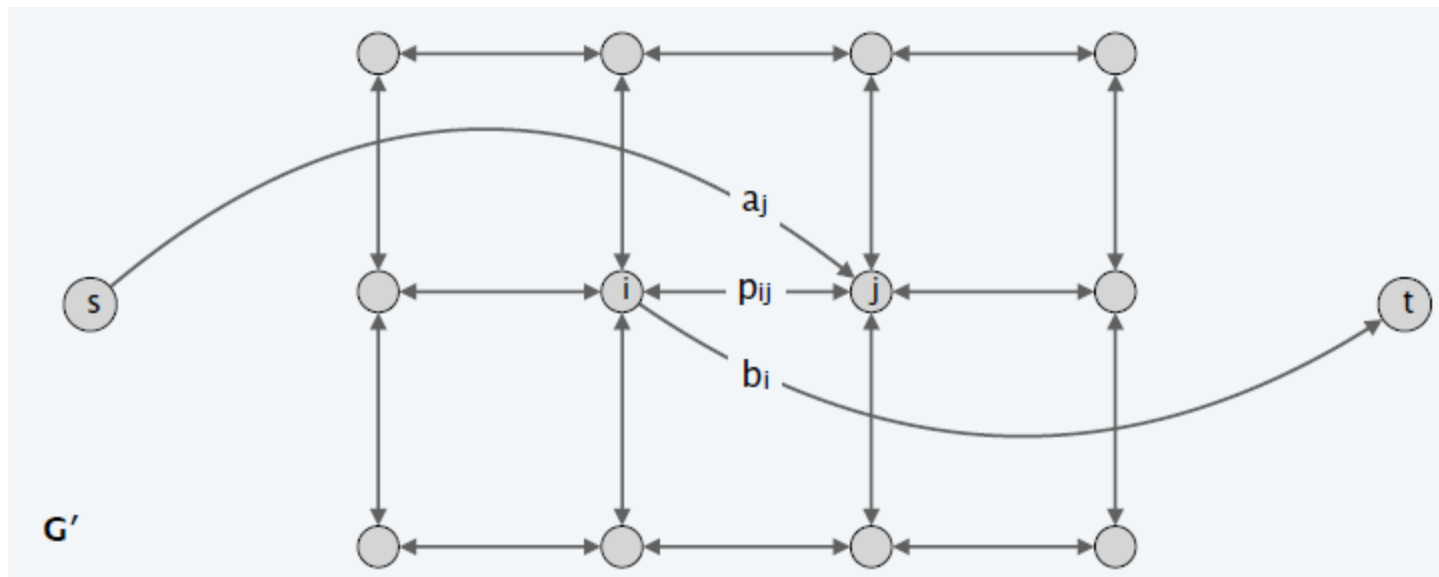
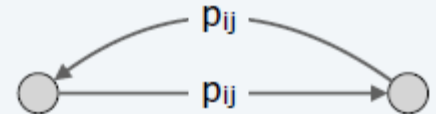
Formulate as min-cut problem  $G' = (V', E')$ .

- Include node for each pixel.
- Use two antiparallel edges instead of undirected edge.
- Add source  $s$  to correspond to foreground.
- Add sink  $t$  to correspond to background.

edge in  $G$



two antiparallel edges in  $G'$





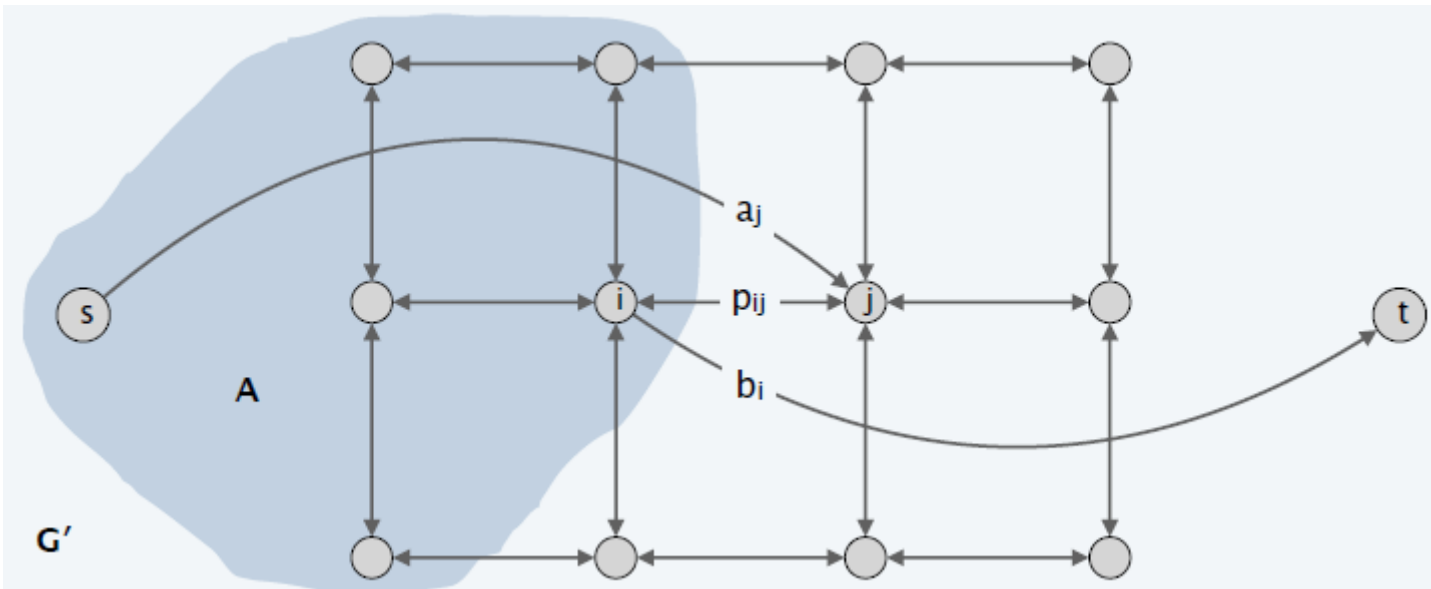
# Image Segmentation

Consider min cut  $(A, B)$  in  $G'$ .

- $A$  = foreground.

$$cap(A, B) = \sum_{j \in B} a_j + \sum_{i \in A} b_i + \sum_{(i,j) \in E, i \in A, j \in B} p_{ij}$$

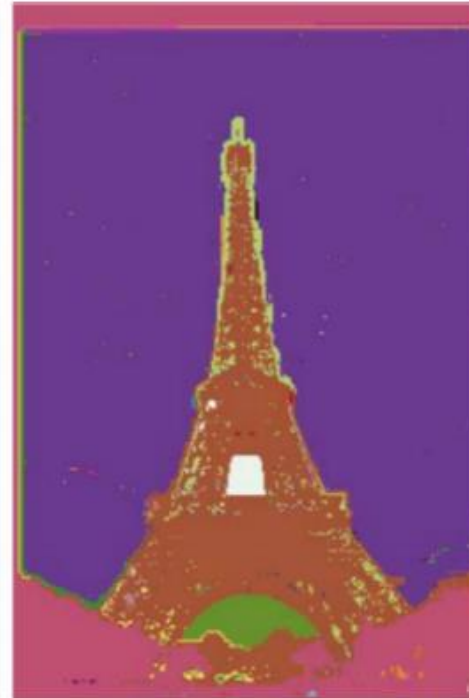
- The quantity we want to minimize.







# Image Segmentation

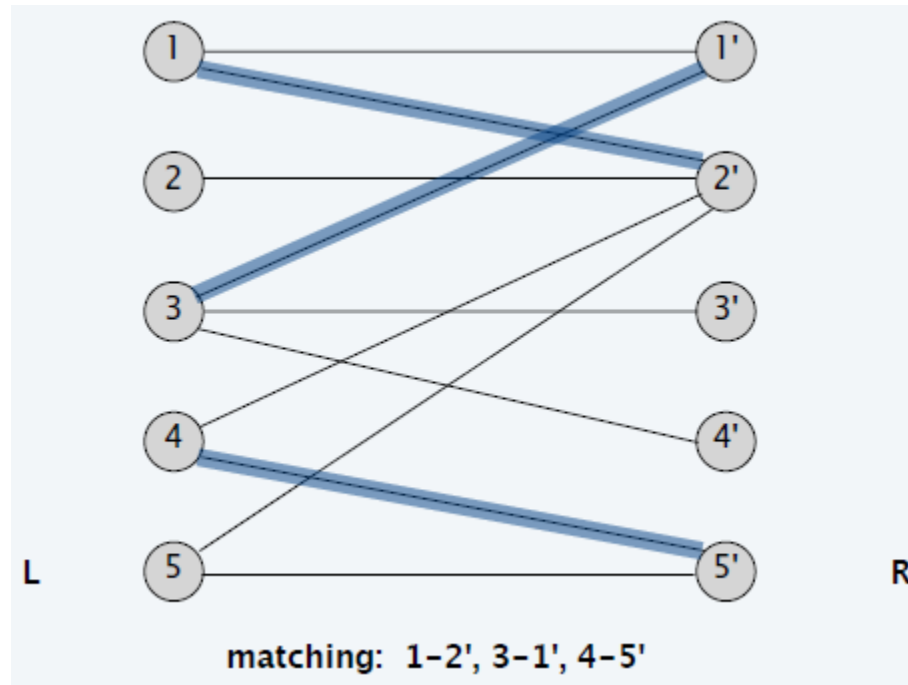




# Bipartite Matching

**Def.** A graph  $G$  is bipartite if the nodes can be partitioned into two subsets  $L$  and  $R$  such that every edge connects a node in  $L$  to one in  $R$ .

**Bipartite matching.** Given a bipartite graph  $G = (L \cup R, E)$ , find a max-cardinality matching.



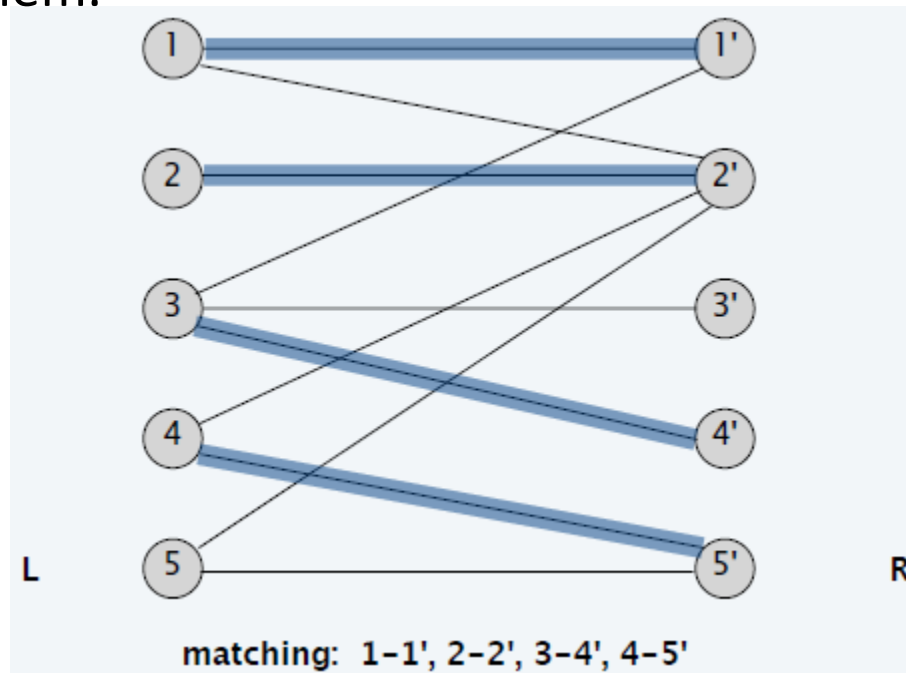


# Bipartite Matching

**Def.** A graph  $G$  is bipartite if the nodes can be partitioned into two subsets  $L$  and  $R$  such that every edge connects a node in  $L$  to one in  $R$ .

**Bipartite matching.** Given a bipartite graph  $G = (L \cup R, E)$ , find a max-cardinality matching.

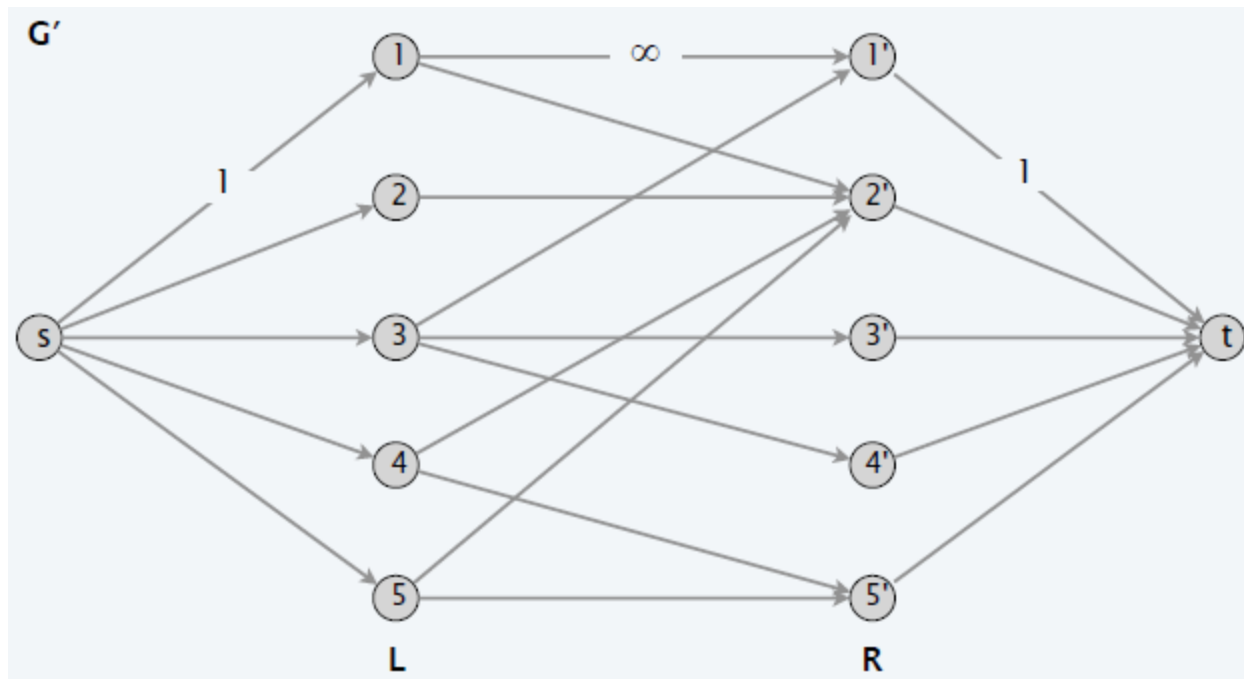
The Ford-Fulkerson algorithm can be implemented to solve the bipartite matching problem.





# Bipartite Matching: Max-Flow Formulation

- Create digraph  $G' = (L \cup R \cup \{s, t\}, E')$ .
- Direct all edges from  $L$  to  $R$ , and assign infinite (or unit) capacity.
- Add source  $s$ , and unit-capacity edges from  $s$  to each node in  $L$ .
- Add sink  $t$ , and unit-capacity edges from each node in  $R$  to  $t$ .



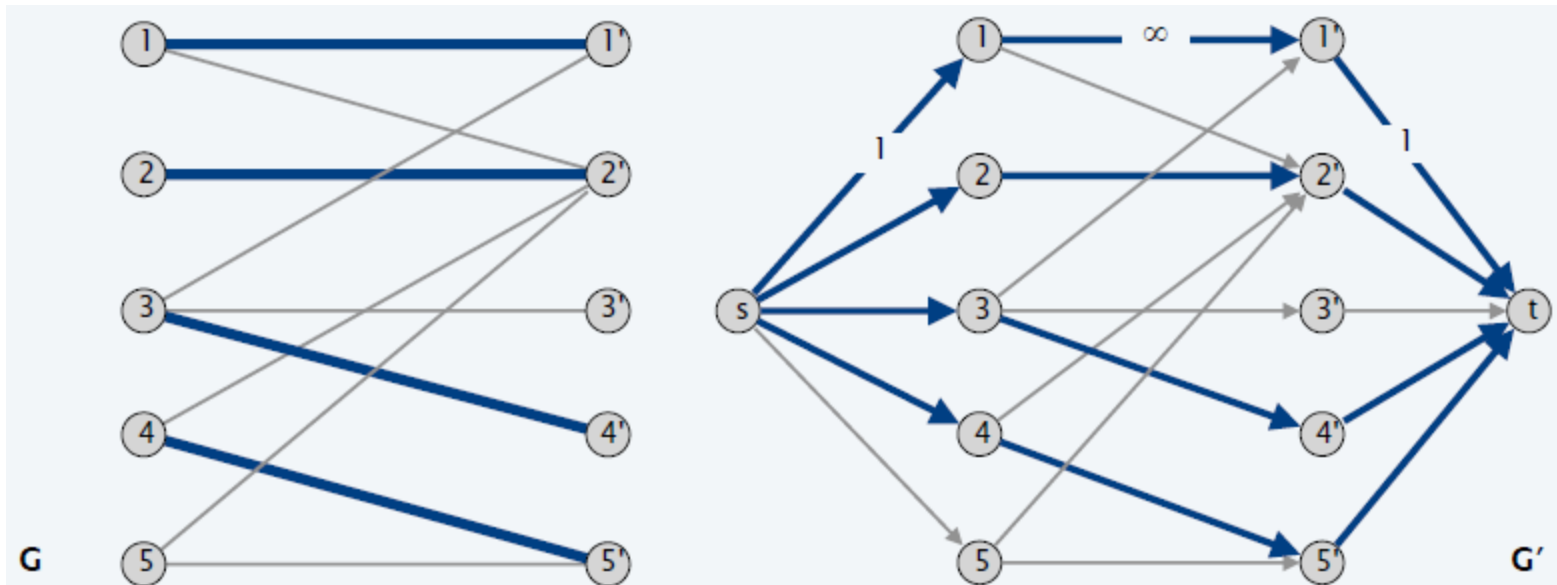


# Max-Flow Formulation: Proof of Correctness

**Theorem.** Max cardinality of a matching in  $G$  = value of max flow in  $G'$ .

**Pf.**  $\leq$

- Given a max matching  $M$  of cardinality  $k$ .
- Consider flow  $f$  that sends 1 unit along each of  $k$  paths.
- $f$  is a flow, and has value  $k$ .



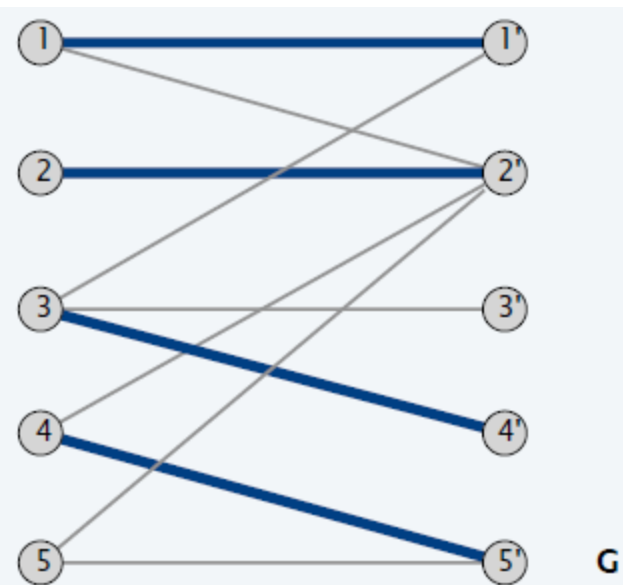
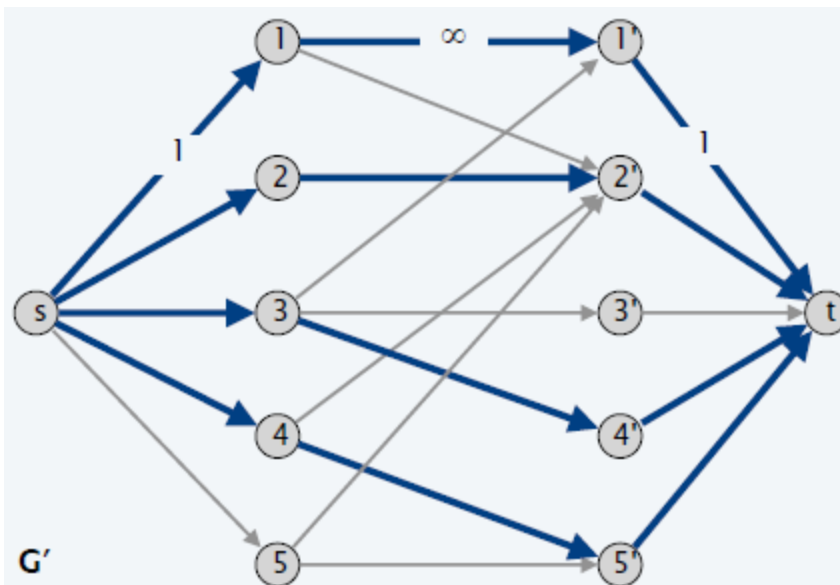


# Max-Flow Formulation: Proof of Correctness

**Theorem.** Max cardinality of a matching in  $G =$  value of max flow in  $G'$ .

**Pf.  $\geq$**

- Let  $f$  be a max flow in  $G'$  of value  $k$ .
- Integrality theorem  $\Rightarrow k$  is integral and can assume  $f$  is 0-1.
- Consider  $M =$  set of edges from  $L$  to  $R$  with  $f(e) = 1$ .
  - Each node in  $L$  and  $R$  participates in at most one edge in  $M$
  - $|M| = k$ : consider cut  $(L \cup \{s\}, R \cup \{t\})$ .





# Perfect Matching in a Bipartite Graph

**Def.** Given a graph  $G = (V, E)$ , a subset of edges  $M \subseteq E$  is a perfect matching if each node appears in exactly one edge in  $M$ .

**Q.** When does a bipartite graph have a perfect matching?

Structure of bipartite graphs with perfect matching.

- Clearly we must have  $|L| = |R|$ .
- What are the other conditions?

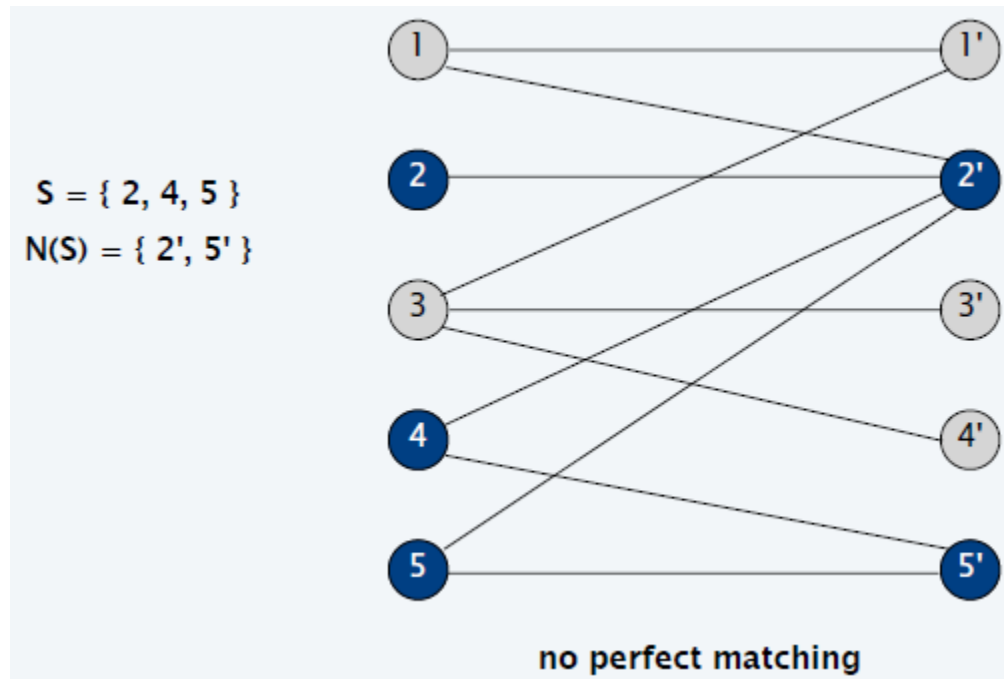


# Perfect Matching in a Bipartite Graph

**Notation.** Let  $S$  be a subset of nodes, and let  $N(S)$  be the set of nodes adjacent to nodes in  $S$ .

**Observation.** If a bipartite graph  $G = (L \cup R, E)$  has a perfect matching, then  $|N(S)| \geq |S|$  for all subsets  $S \subseteq L$ .

**Pf.** Each node in  $S$  has to be matched to a different node in  $N(S)$ .







# Bipartite Matching

**Bipartite matching.** Can solve via reduction to maximum flow.

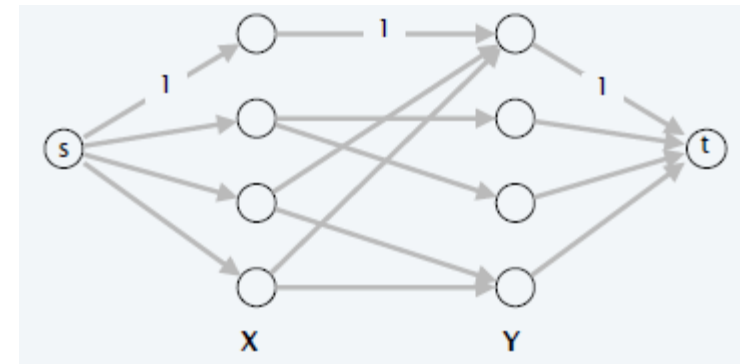
**Flow.** During Ford-Fulkerson, all residual capacities and flows are 0-1; flow corresponds to edges in a matching  $M$ .

**Residual graph  $G_M$**  simplifies to:

- If  $(x, y) \notin M$ , then  $(x, y)$  is in  $G_M$ .
- If  $(x, y) \in M$ , then  $(y, x)$  is in  $G_M$ .

**Augmenting path** simplifies to:

- Edge from  $s$  to an unmatched node  $x \in X$ ,
- Alternating sequence of unmatched and matched edges,
- Edge from unmatched node  $y \in Y$  to  $t$ .

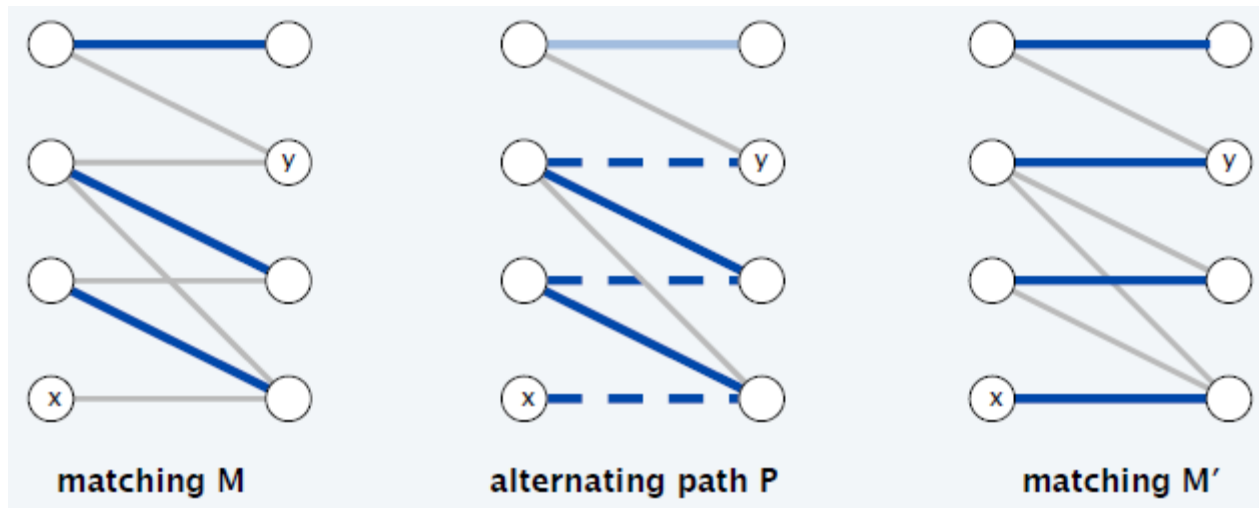




# Alternating Path

**Def.** An **alternating path**  $P$  with respect to a matching  $M$  is an alternating sequence of unmatched and matched edges, starting from an unmatched node  $x \in X$  and going to an unmatched node  $y \in Y$ .

**Key property.** Can use  $P$  to increase by one the cardinality of the matching.

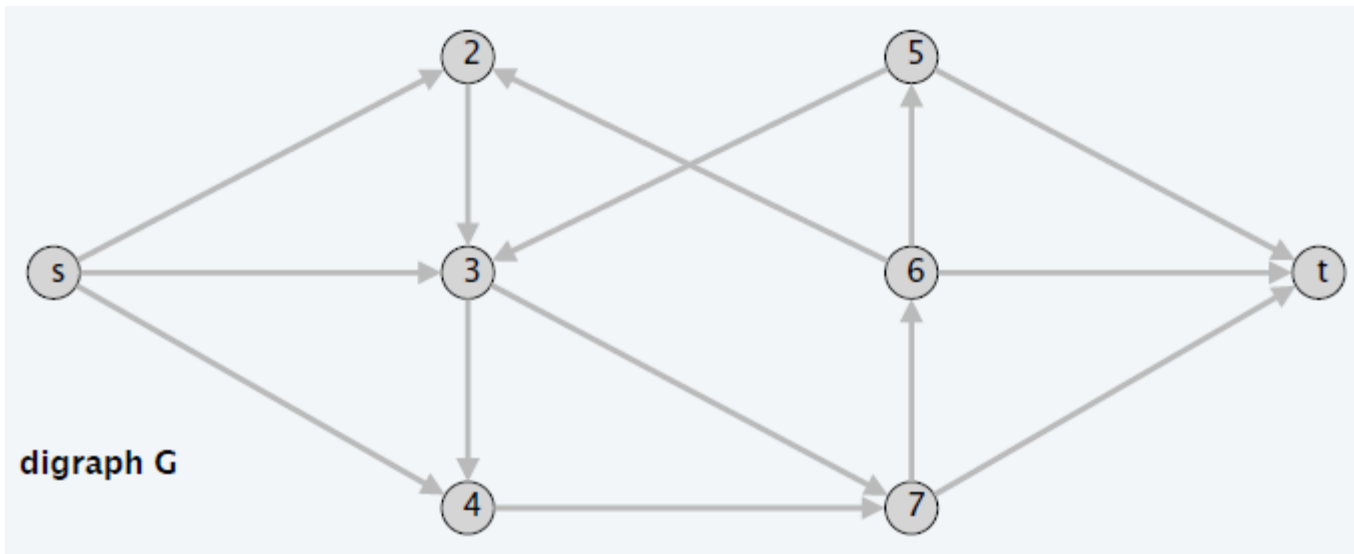




# Edge-Disjoint Paths

**Def.** Two paths are **edge-disjoint** if they have no edge in common.

**Disjoint path problem.** Given a digraph  $G = (V, E)$  and two nodes  $s$  and  $t$ , find the max number of edge-disjoint  $s \rightarrow t$  paths.

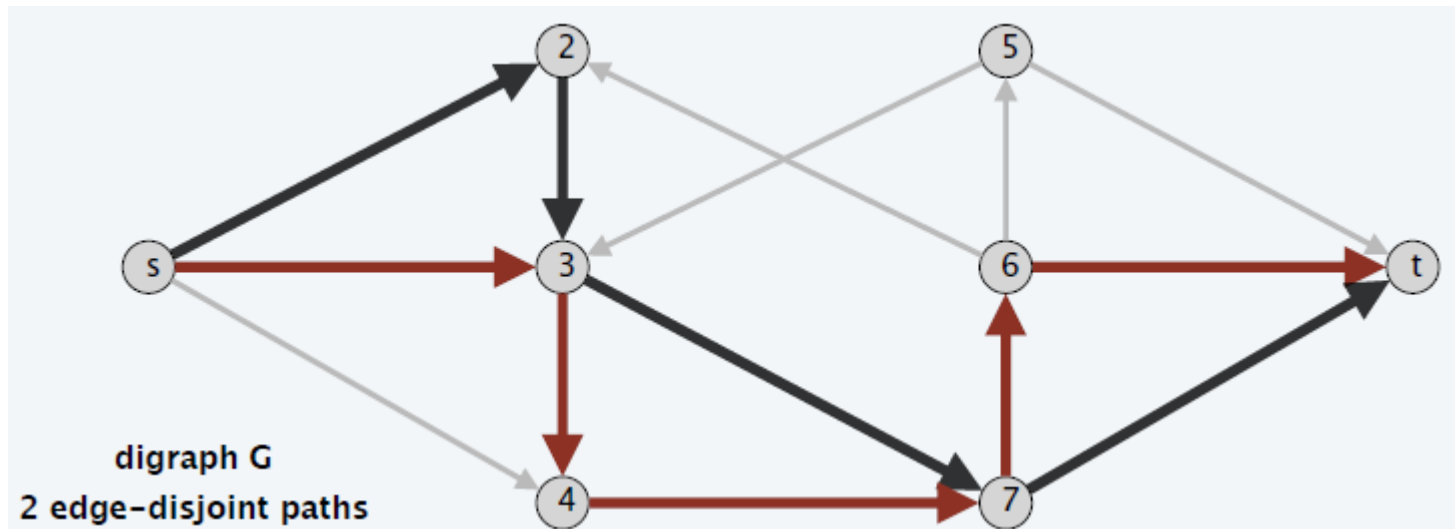




# Edge-Disjoint Paths

**Def.** Two paths are **edge-disjoint** if they have no edge in common.

**Disjoint path problem.** Given a digraph  $G = (V, E)$  and two nodes  $s$  and  $t$ , find the max number of edge-disjoint  $s \rightarrow t$  paths.





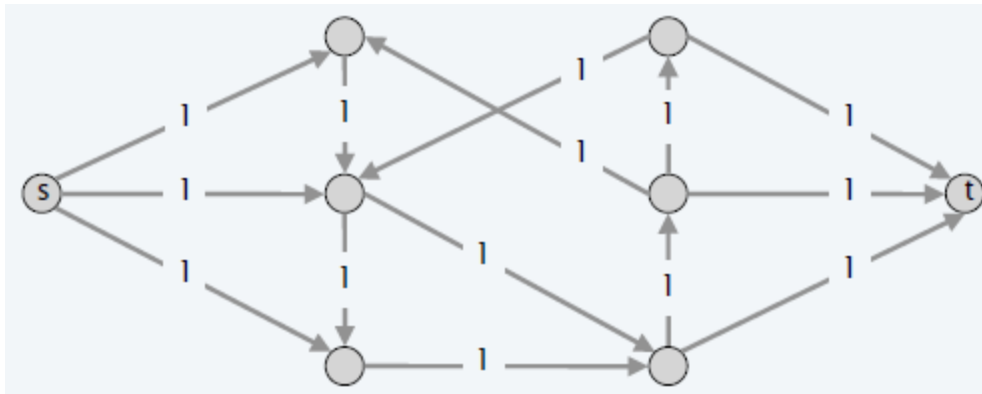
# Edge-Disjoint Paths

**Max-flow formulation.** Assign unit capacity to every edge.

**Theorem.** Max number of edge-disjoint  $s \rightarrow t$  paths equals value of max flow.

**Pf.  $\leq$**

- Suppose there are  $k$  edge-disjoint  $s \rightarrow t$  paths  $P_1, \dots, P_k$ .
- Set  $f(e) = 1$  if  $e$  participates in some path  $P_j$ , else set  $f(e) = 0$ .
- Since paths are edge-disjoint,  $f$  is a flow of value  $k$ .





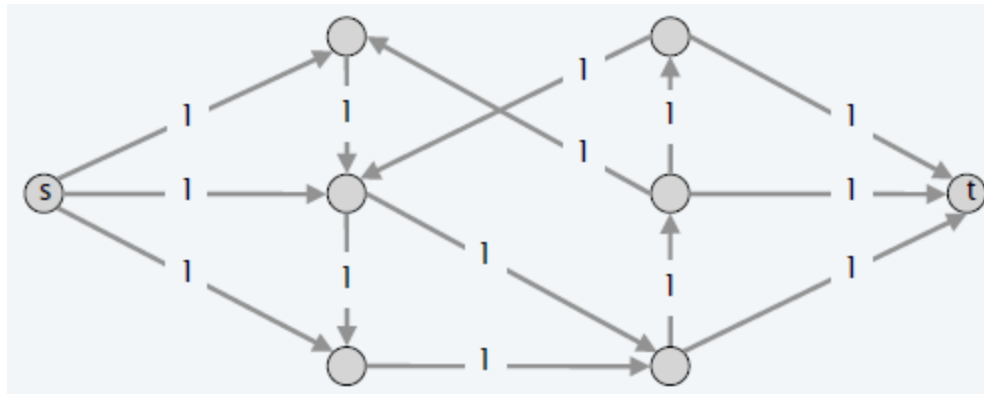
# Edge-Disjoint Paths

**Max-flow formulation.** Assign unit capacity to every edge.

**Theorem.** Max number of edge-disjoint  $s \rightarrow t$  paths equals value of max flow.

**Pf.  $\geq$**

- Suppose max flow value is  $k$ .
- Integrality theorem  $\Rightarrow$  there exists 0-1 flow  $f$  of value  $k$ .
- Consider edge  $(s, u)$  with  $f(s, u) = 1$ .
  - By flow conservation, there exists an edge  $(u, v)$  with  $f(u, v) = 1$
  - Continue until reach  $t$ , always choosing a new edge
- Produces  $k$  edge-disjoint paths.

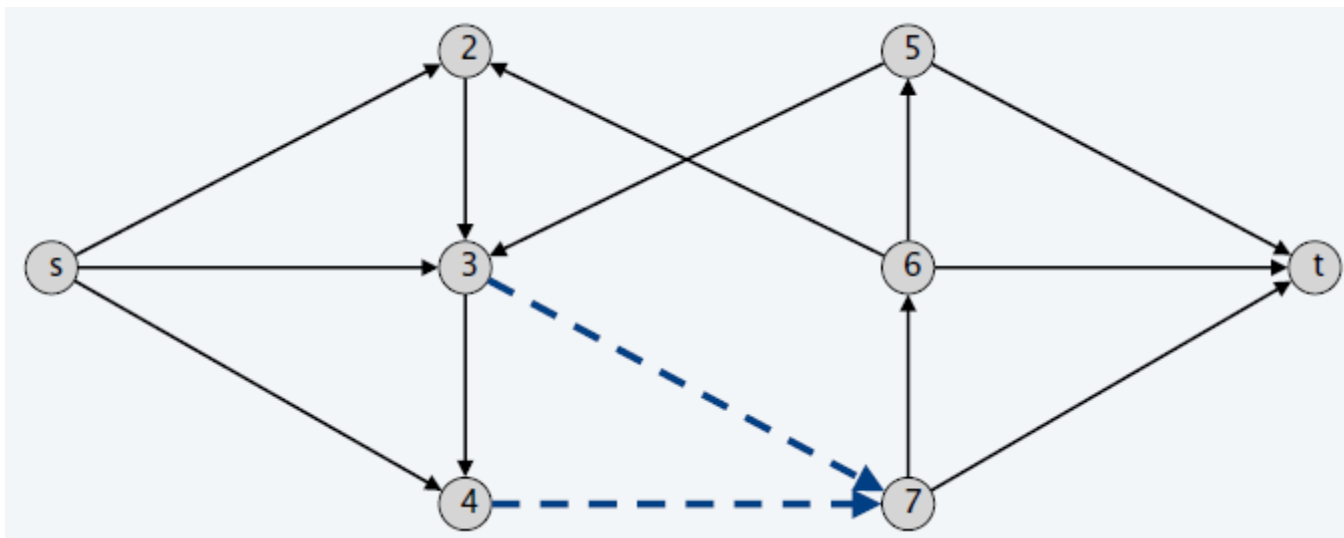




# Network Connectivity

**Def.** A set of edges  $F \subseteq E$  **disconnects  $t$  from  $s$**  if every  $s \rightarrow t$  path uses at least one edge in  $F$ .

**Network connectivity.** Given a digraph  $G = (V, E)$  and two nodes  $s$  and  $t$ , find min number of edges whose removal disconnects  $t$  from  $s$ .



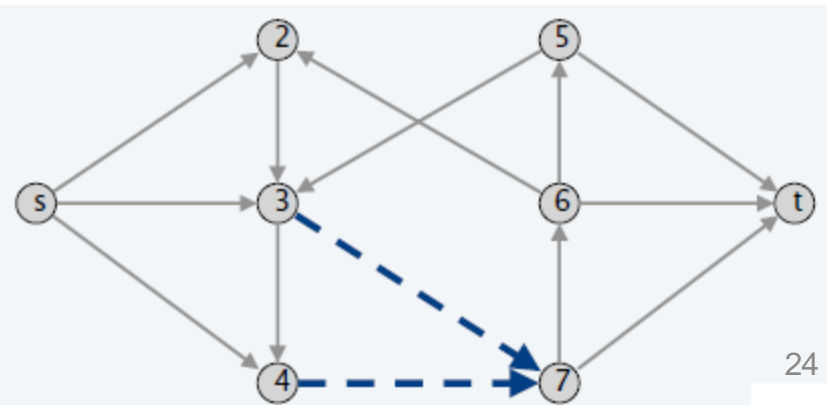
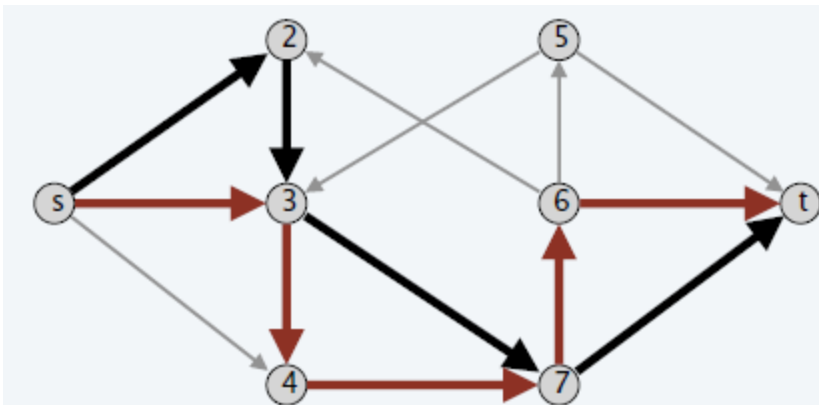


# Menger's Theorem

**Theorem.** The max number of edge-disjoint  $s \rightarrow t$  paths equals the min number of edges whose removal disconnects  $t$  from  $s$ .

Pr.  $\leq$

- Suppose the removal of  $F \subseteq E$  disconnects  $t$  from  $s$ , and  $|F| = k$ .
- Every  $s \rightarrow t$  path uses at least one edge in  $F$ .
- Hence, the number of edge-disjoint paths is  $\leq k$ .





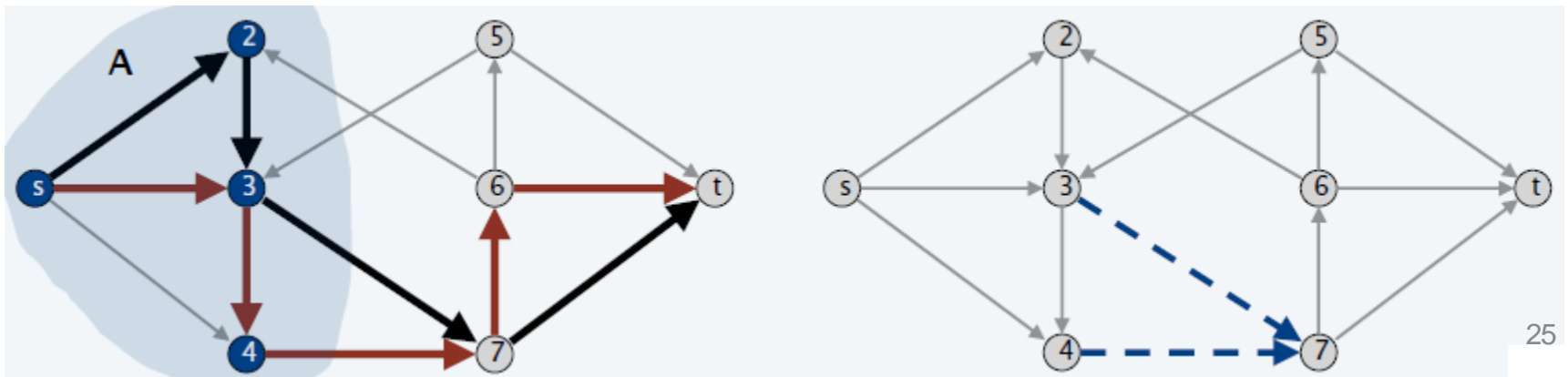


# Menger's Theorem

**Theorem.** The max number of edge-disjoint  $s \rightarrow t$  paths equals the min number of edges whose removal disconnects  $t$  from  $s$ .

Pr.  $\geq$

- Suppose max number of edge-disjoint paths is  $k$ .
- Then value of max flow =  $k$ .
- Max-flow min-cut theorem  $\Rightarrow$  there exists a cut  $(A, B)$  of capacity  $k$ .
- Let  $F$  be set of edges going from  $A$  to  $B$ .
- $|F| = k$  and disconnects  $t$  from  $s$ .

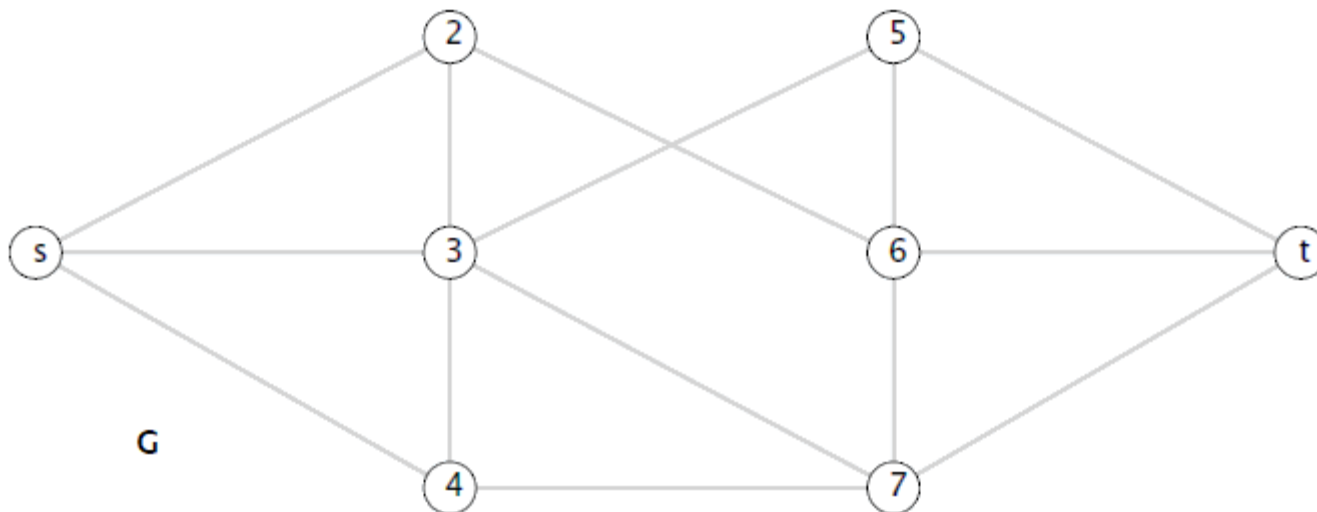




# Edge-Disjoint Paths in Undirected Graphs

**Def.** Two paths are **edge-disjoint** if they have no edge in common.

**Disjoint path problem in undirected graphs.** Given a graph  $G = (V, E)$  and two nodes  $s$  and  $t$ , find the max number of edge-disjoint  $s \rightarrow t$  paths.

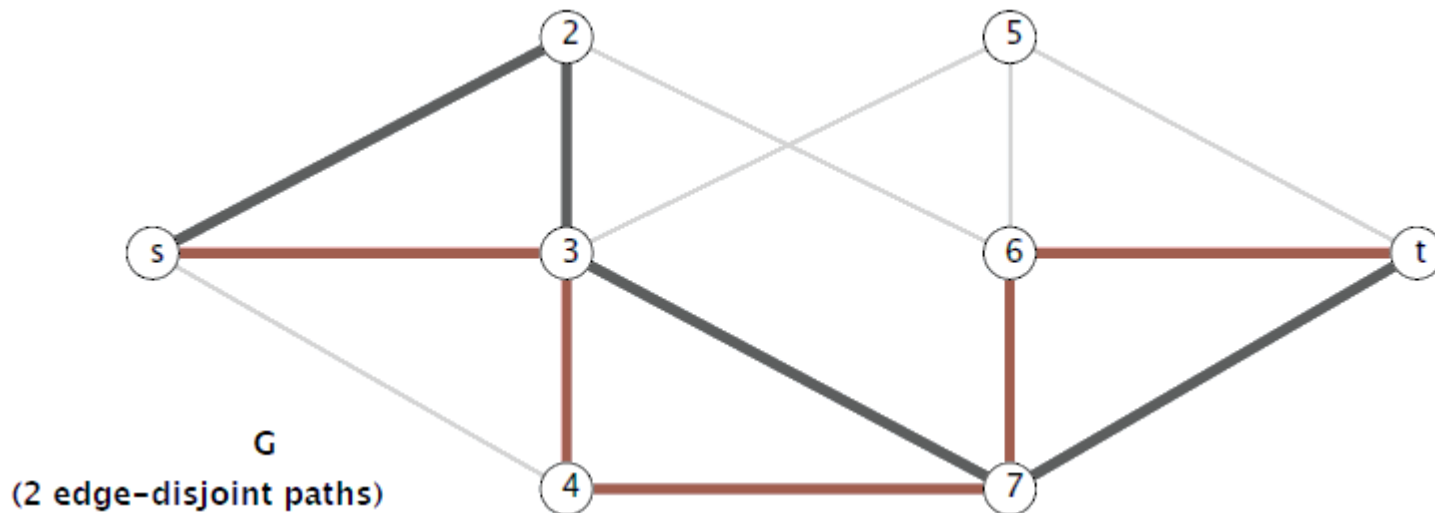




# Edge-Disjoint Paths in Undirected Graphs

**Def.** Two paths are **edge-disjoint** if they have no edge in common.

**Disjoint path problem in undirected graphs.** Given a graph  $G = (V, E)$  and two nodes  $s$  and  $t$ , find the max number of edge-disjoint  $s \rightarrow t$  paths.

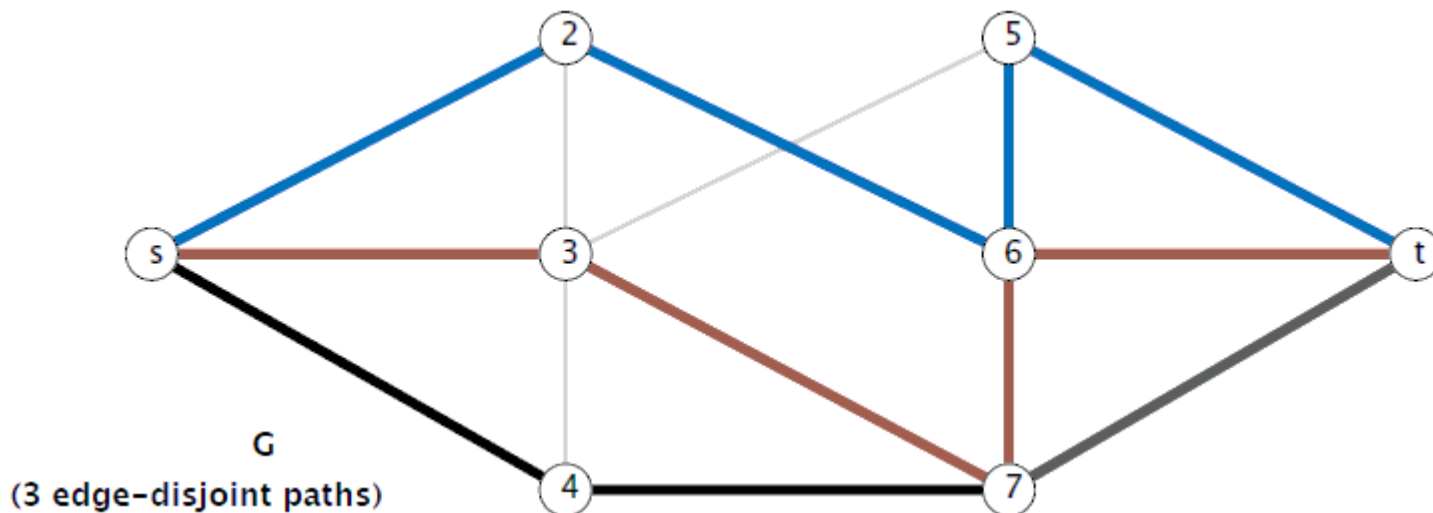




# Edge-Disjoint Paths in Undirected Graphs

**Def.** Two paths are **edge-disjoint** if they have no edge in common.

**Disjoint path problem in undirected graphs.** Given a graph  $G = (V, E)$  and two nodes  $s$  and  $t$ , find the max number of edge-disjoint  $s \rightarrow t$  paths.



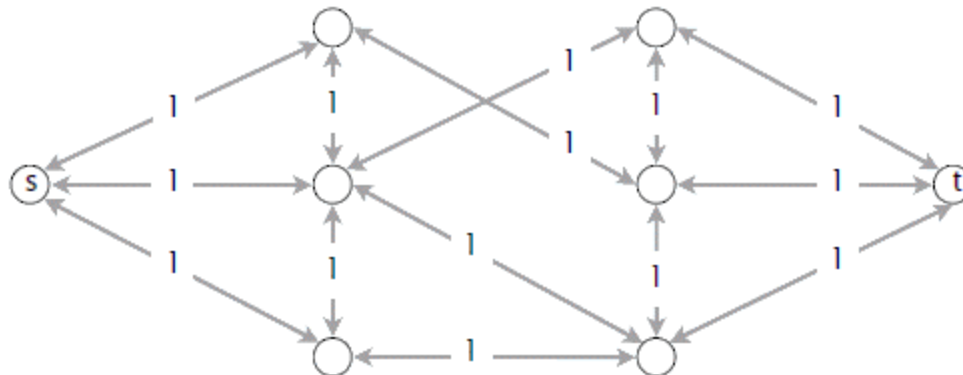


# Edge-Disjoint Paths in Undirected Graphs

**Max-flow formulation.** Replace each edge with two antiparallel edges and assign unit capacity to every edge.

**Observation.** Two paths  $P_1$  and  $P_2$  may be edge-disjoint in the digraph but not edge-disjoint in the undirected graph.

If  $P_1$  uses edge  $(u, v)$  and  $P_2$  uses its antiparallel edge  $(v, u)$





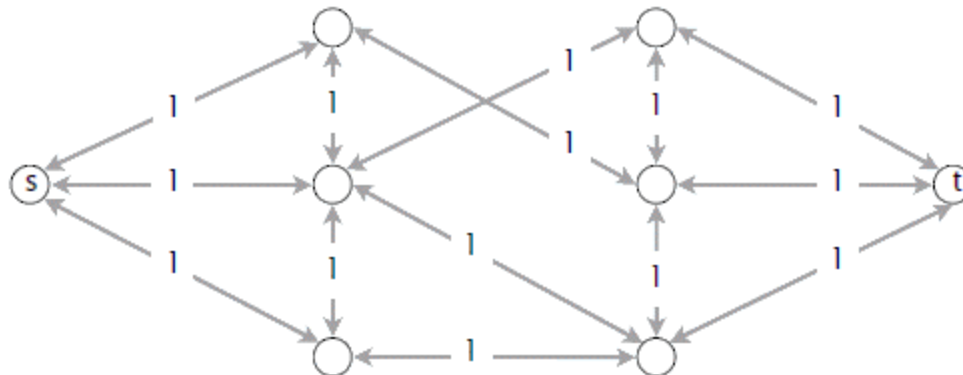
# Edge-Disjoint Paths in Undirected Graphs

**Max-flow formulation.** Replace each edge with two antiparallel edges and assign unit capacity to every edge.

**Lemma.** In any flow network, there exists a maximum flow  $f$  in which for each pair of antiparallel edges  $e$  and  $e'$ , either  $f(e) = 0$  or  $f(e') = 0$  or both. Moreover, integrality theorem still holds.

**Pf.** [by induction on number of such pairs of antiparallel edges]

- Suppose  $f(e) > 0$  and  $f(e') > 0$  for a pair of antiparallel edges  $e$  and  $e'$ .
- Set  $f(e) = f(e) - \delta$  and  $f(e') = f(e') - \delta$ , where  $\delta = \min\{f(e), f(e')\}$ .
- $f$  is still a flow of the same value but has one less such pair.





# Edge-Disjoint Paths in Undirected Graphs

**Max-flow formulation.** Replace each edge with two antiparallel edges and assign unit capacity to every edge.

**Lemma.** In any flow network, there exists a maximum flow  $f$  in which for each pair of antiparallel edges  $e$  and  $e'$ , either  $f(e) = 0$  or  $f(e') = 0$  or both. Moreover, integrality theorem still holds.

**Theorem.** Max number edge-disjoint  $s \rightarrow t$  paths equals value of max flow.

