



CS240 Algorithm Design and Analysis

Lecture 8

Network Flow (Cont.)

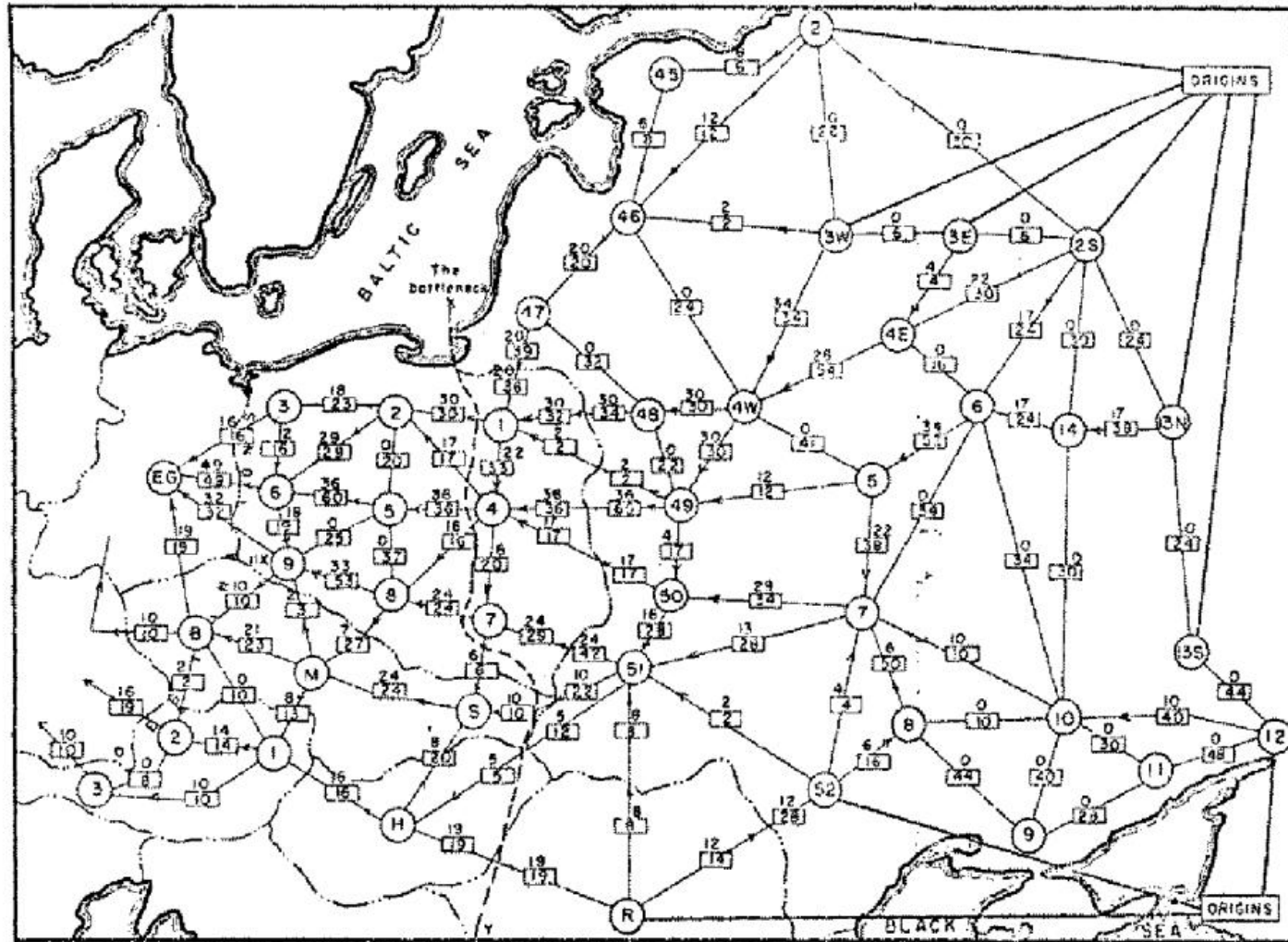
Quan Li
Fall 2025
2025.10.14



Applications of max-flow



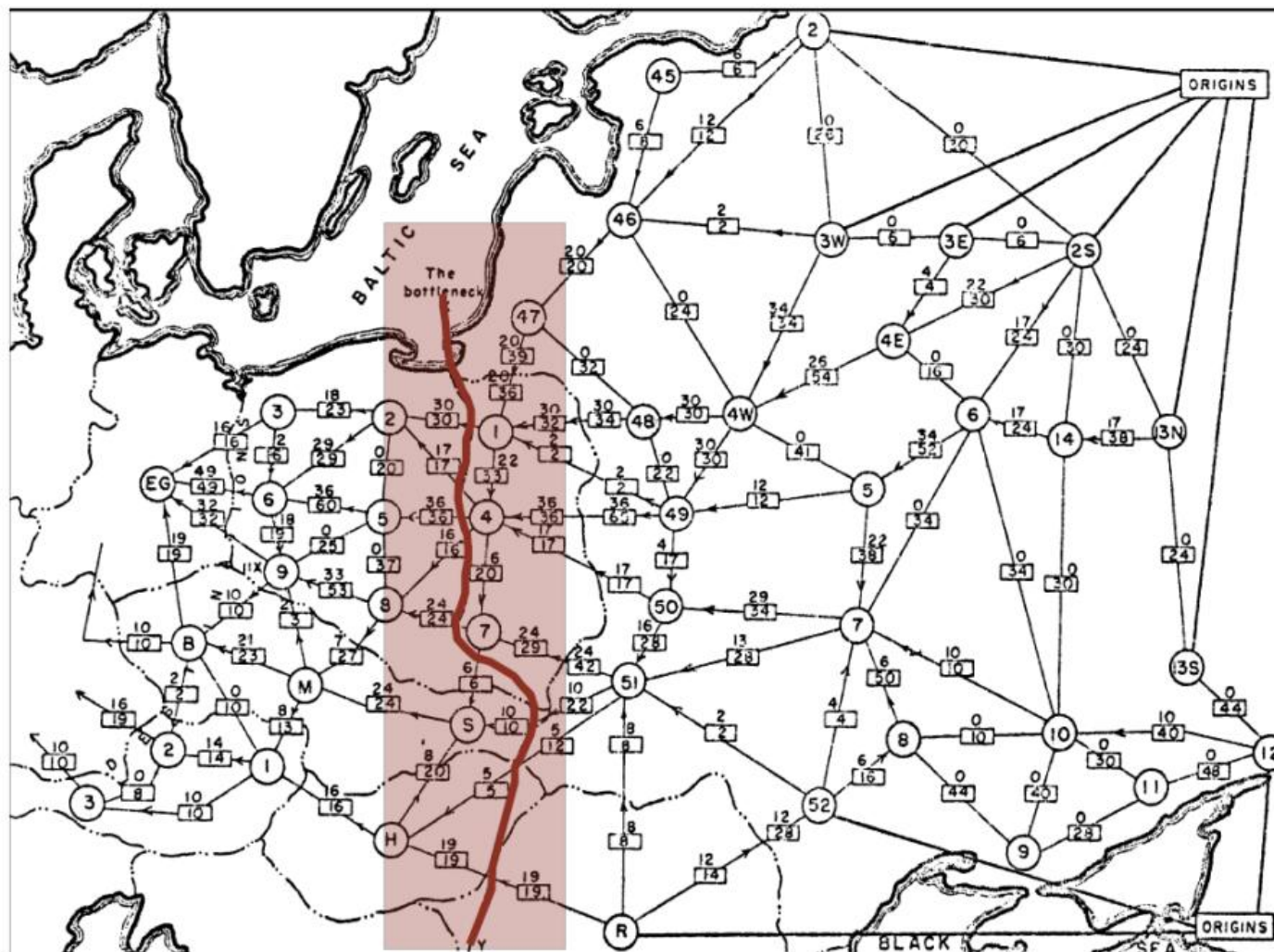
Soviet Rail Network, 1955



Reference: On the history of the transportation and maximum flow problems. Alexander Schrijver in Math Programming, 91: 3, 2002



Soviet Rail Network, 1955



Reference: On the history of the transportation and maximum flow problems. Alexander Schrijver in Math Programming, 91: 3, 2002



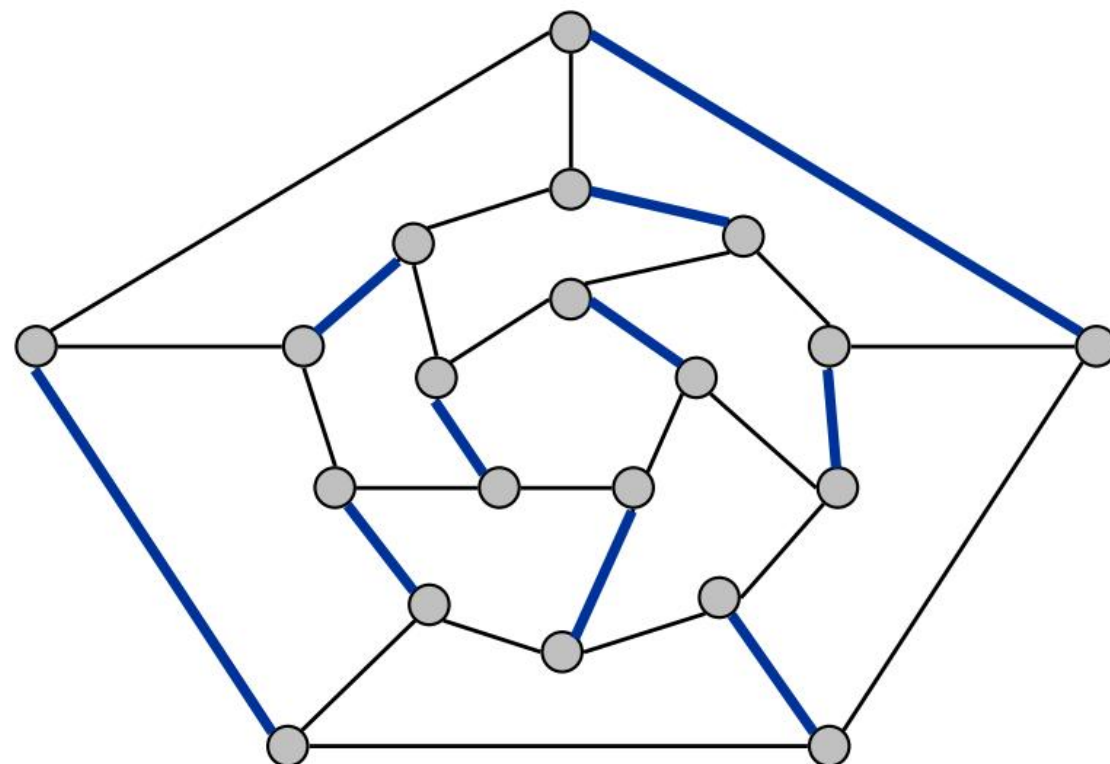
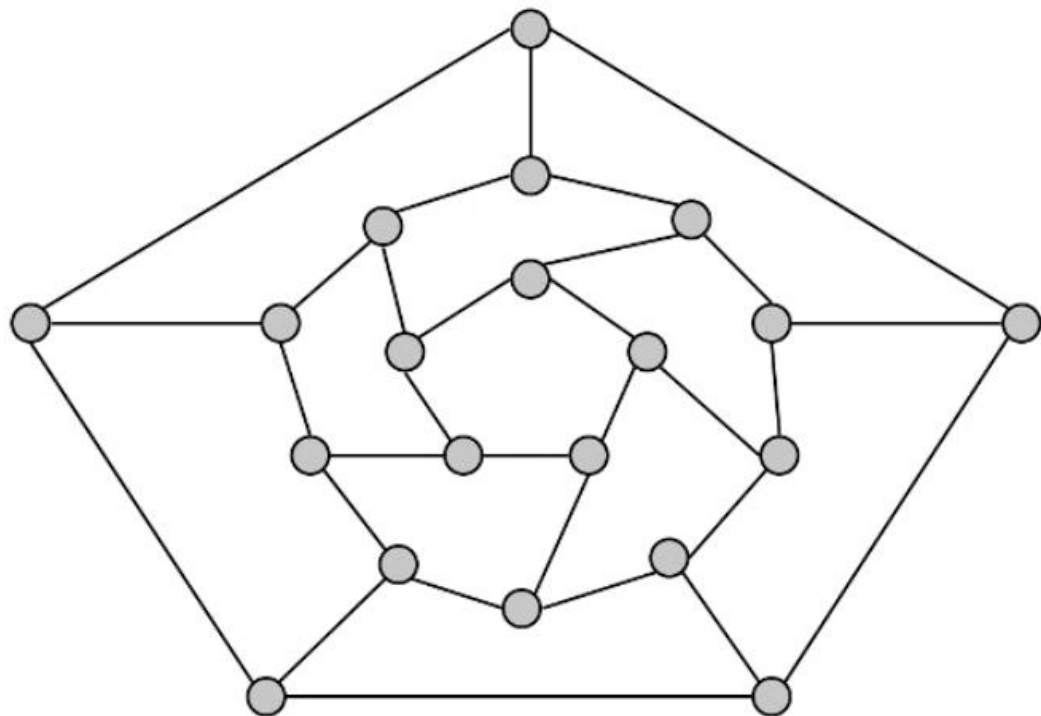
Bipartite Matching



Matching

- **Matching**

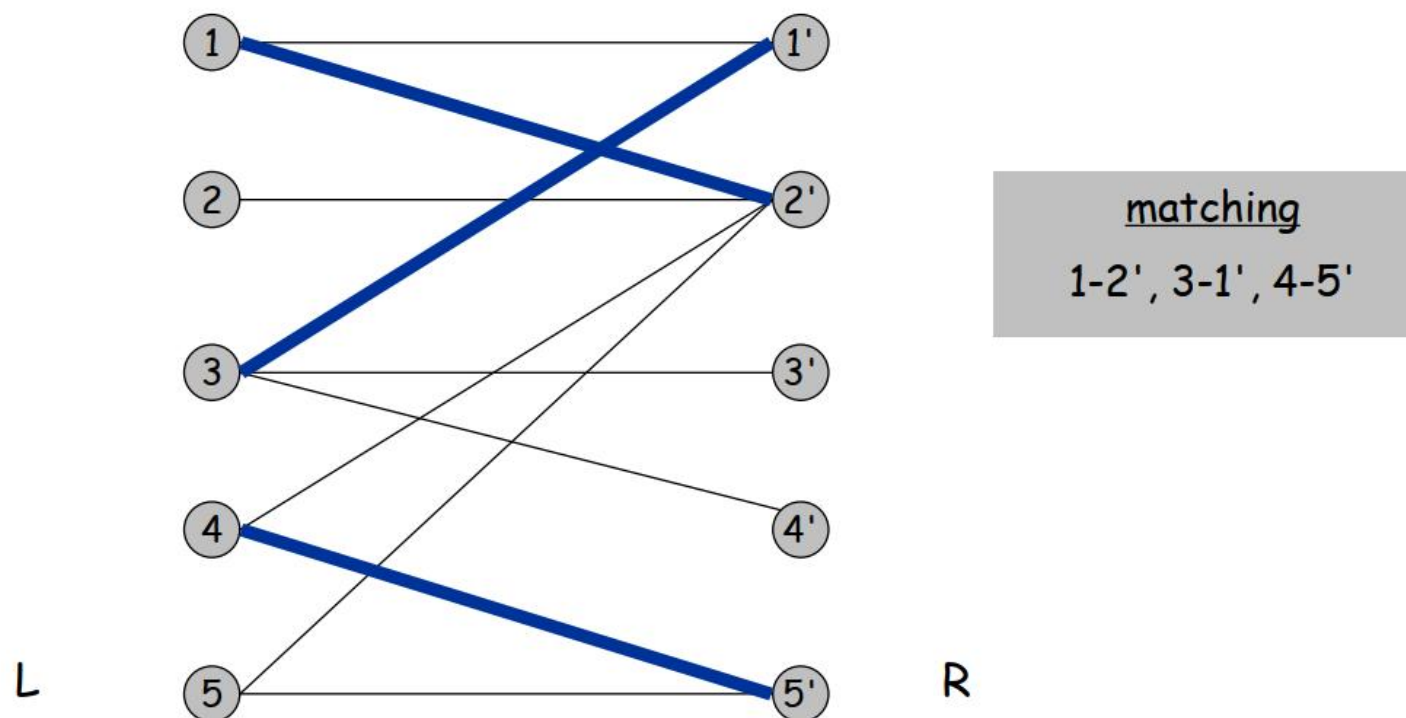
- Input: undirected graph $G = (V, E)$
- $M \subseteq E$ is a **matching** if each node appears in at most one edge in M
- Max matching: find a max cardinality matching





Bipartite Matching

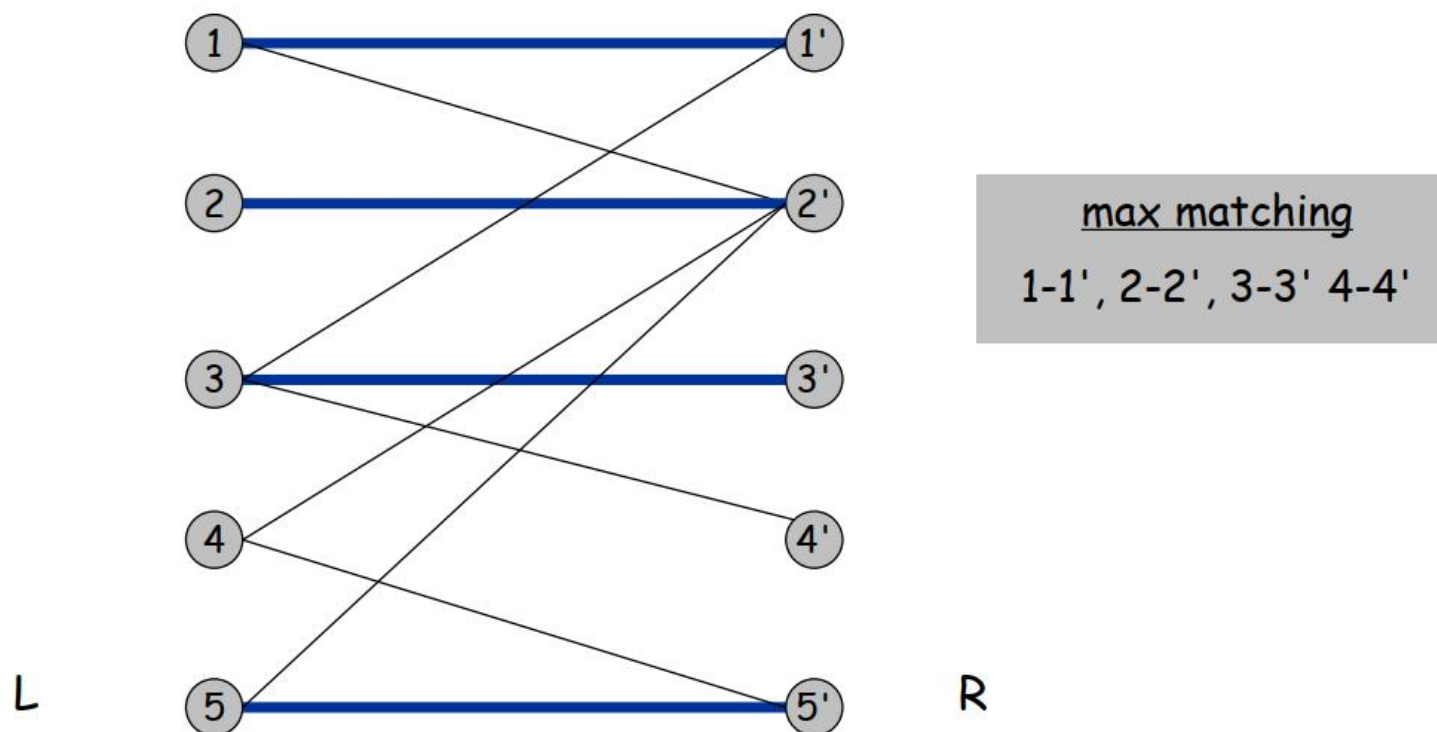
- Bipartite matching
 - Input: undirected, **bipartite** graph $G = (L \cup R, E)$
 - $M \subseteq E$ is a **matching** if each node appears in at most one edge in M
 - Max matching: find a max cardinality matching





Bipartite Matching

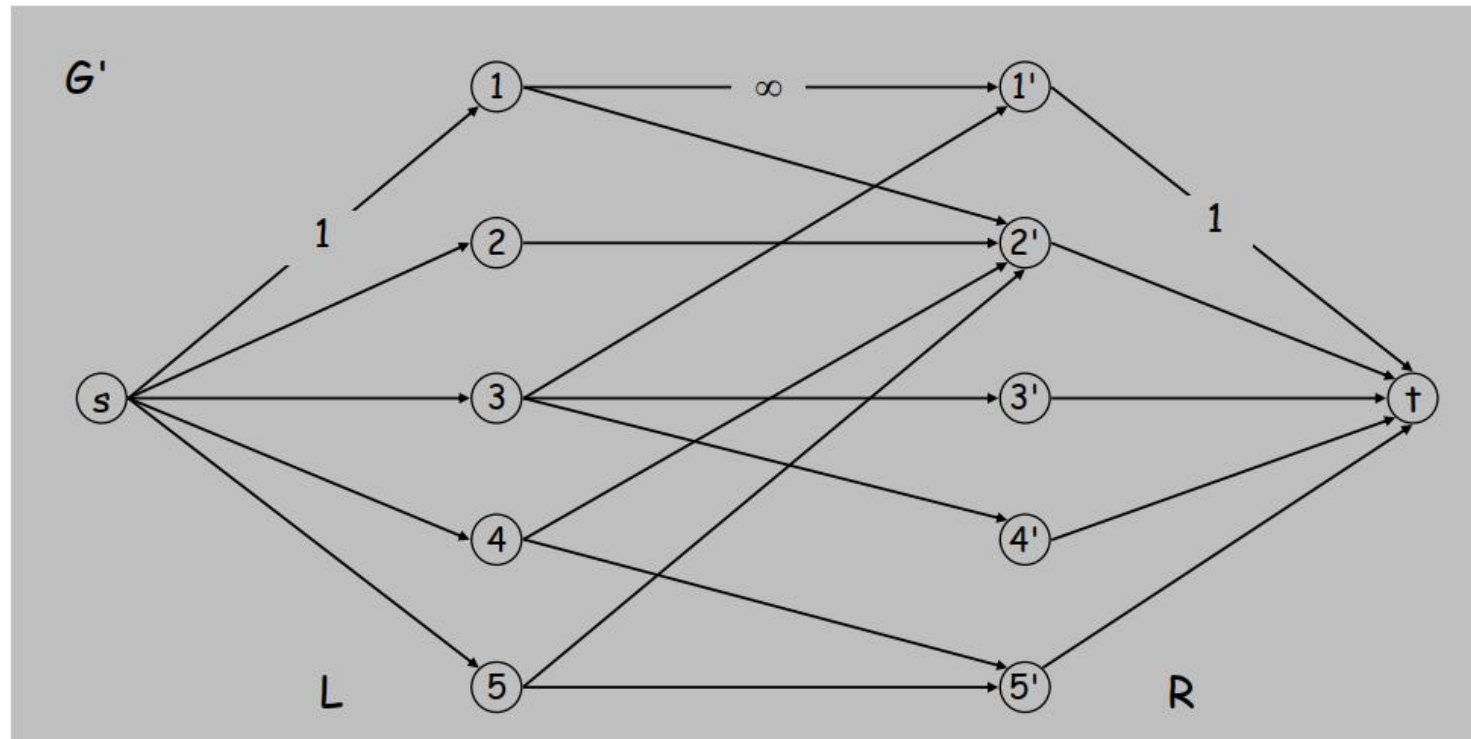
- Bipartite matching
 - Input: undirected, **bipartite** graph $G = (L \cup R, E)$
 - $M \subseteq E$ is a **matching** if each node appears in at most one edge in M
 - Max matching: find a max cardinality matching



Bipartite Matching

- **Max flow formulation**

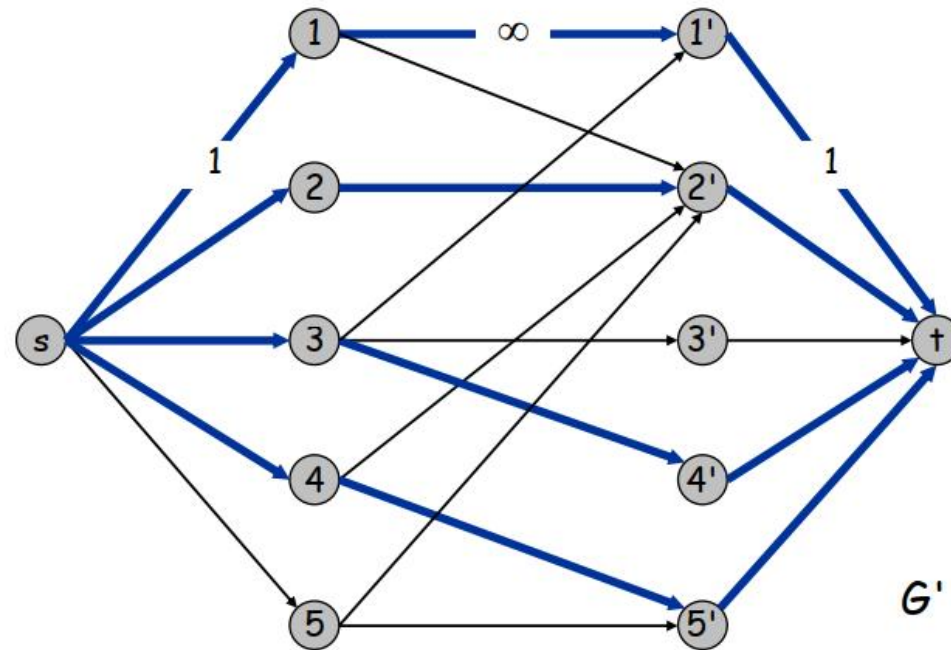
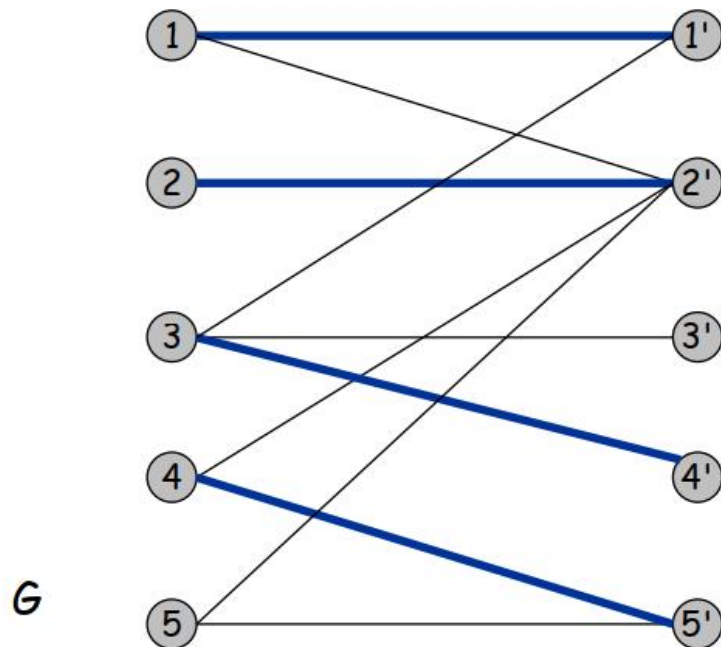
- Create digraph $G' = (L \cup R \cup \{s, t\}, E')$
- Direct all edges from L to R , and assign infinite (or unit) capacity
- Add source s , and unit capacity edges from s to each node in L
- Add sink t , and unit capacity edges from each node in R to t





Bipartite Matching: Proof of Correctness

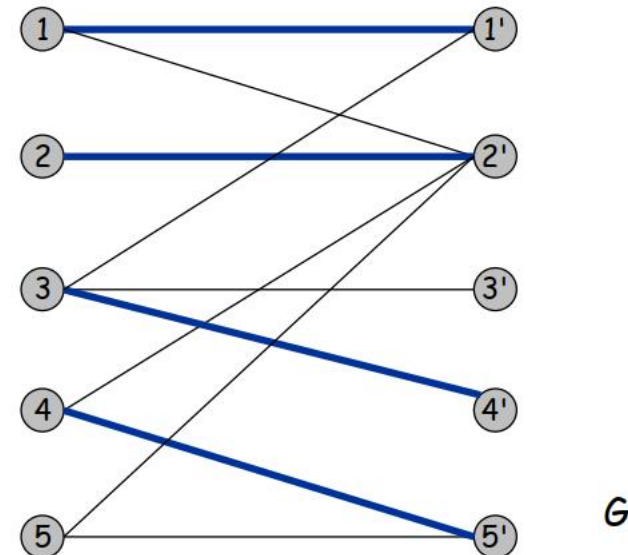
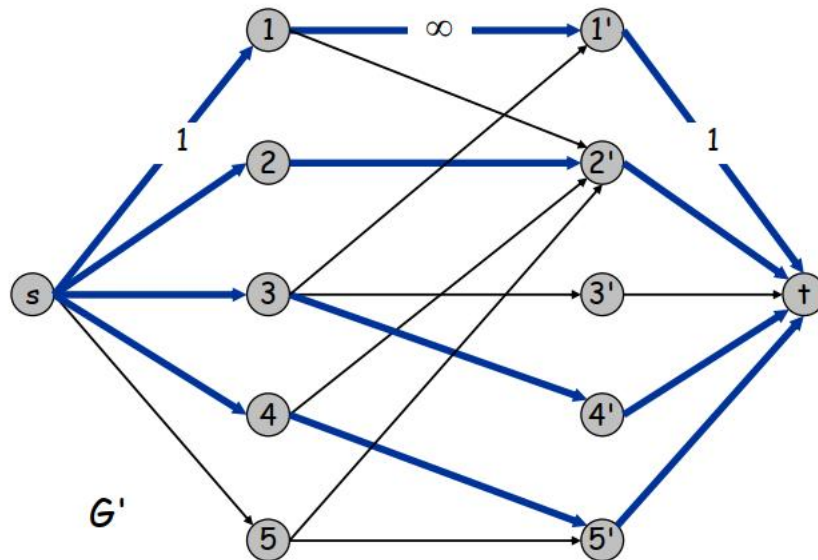
- **Theorem.** Max cardinality matching in G = value of max flow in G'
- **Pf. \Leftarrow**
 - Given max matching M of cardinality k
 - Consider flow f that sends 1 unit along each of k paths
 - f is a flow, and has cardinality k





Bipartite Matching: Proof of Correctness

- **Theorem.** Max cardinality matching in G = value of max flow in G'
- **Pf. \Rightarrow**
 - 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 $(L \cup s, R \cup t)$

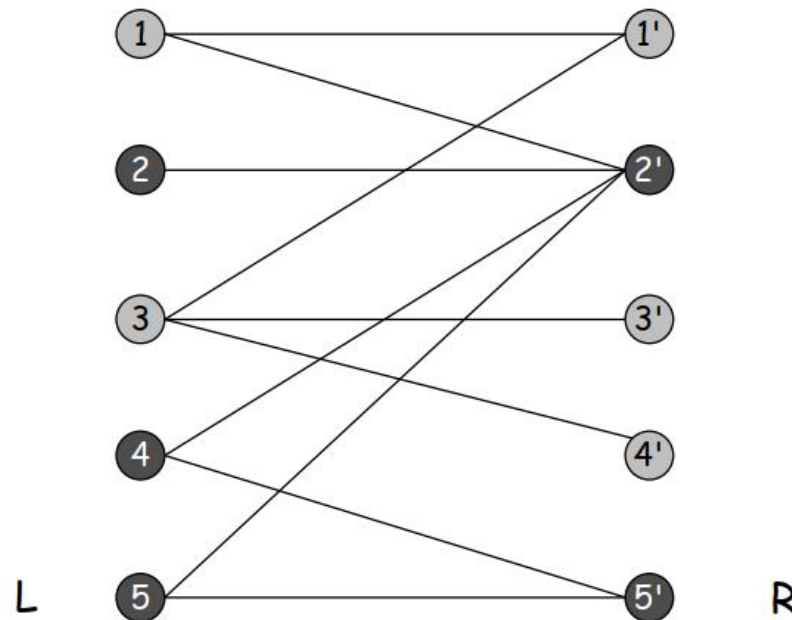


Perfect Matching

- **Def.** A matching $M \subseteq E$ is **perfect** 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 matchings**
 - Clearly we must have $|L| = |R|$
 - What other conditions are necessary?
 - What conditions are sufficient?

Perfect Matching

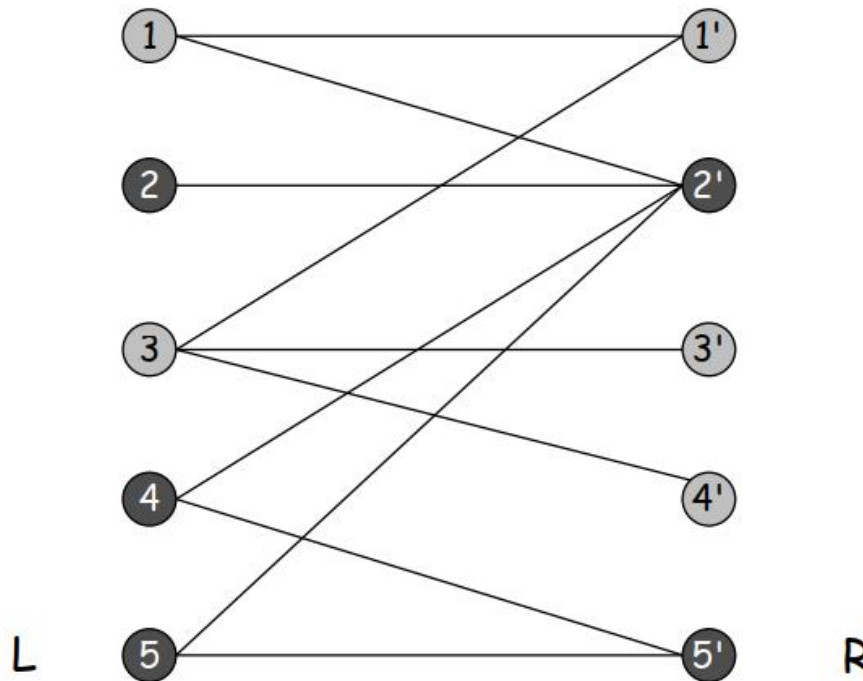
- **Notation.** Let S be a subset of nodes, and let $N(S)$ be the set of nodes adjacent to nodes in S
- **Observation.** If a bipartite graph $G = (L \cup R, E)$, has a perfect matching, then $|N(S)| \geq |S|$ for all subsets $S \subseteq L$
- **Pf.** Each node in S has to be matched to a different node in $N(S)$



No perfect matching:
 $S = \{ 2, 4, 5 \}$
 $N(S) = \{ 2', 5' \}.$

Marriage Theorem

- **Marriage Theorem.** [Frobenius 1917, Hall 1935] Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$. Then, G has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$
- **Pf.** \rightarrow This was the previous observation



No perfect matching:

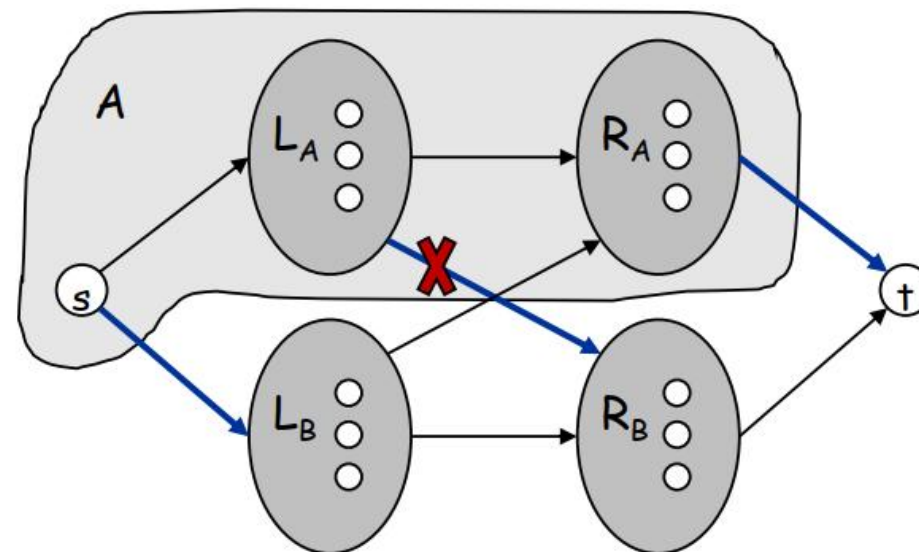
$$S = \{ 2, 4, 5 \}$$

$$N(S) = \{ 2', 5' \}.$$



Proof of Marriage Theorem

- **Marriage Theorem.** G has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$
- **Pf.** \leftarrow Suppose G does not have a perfect matching
 - Formulate as a max flow problem and let (A, B) be min cut in G'
 - Define $L_A = L \cap A$, $L_B = L \cap B$, $R_A = R \cap A$, $R_B = R \cap B$
 - $\text{Cap}(A, B) = v(f^*) = |M| < |L|$ (" $<$ ": because no perfect matching)
 - Since min cut can't use ∞ edges, no edge between L_A and R_B
 - $\text{Cap}(A, B) = |L_B| + |R_A|$
 - $N(L_A) \subseteq R_A$
 - $|N(L_A)| \leq |R_A|$
 $= \text{cap}(A, B) - |L_B|$
 $< |L| - |L_B|$
 $= |L_A|$
 - This contradicts the condition

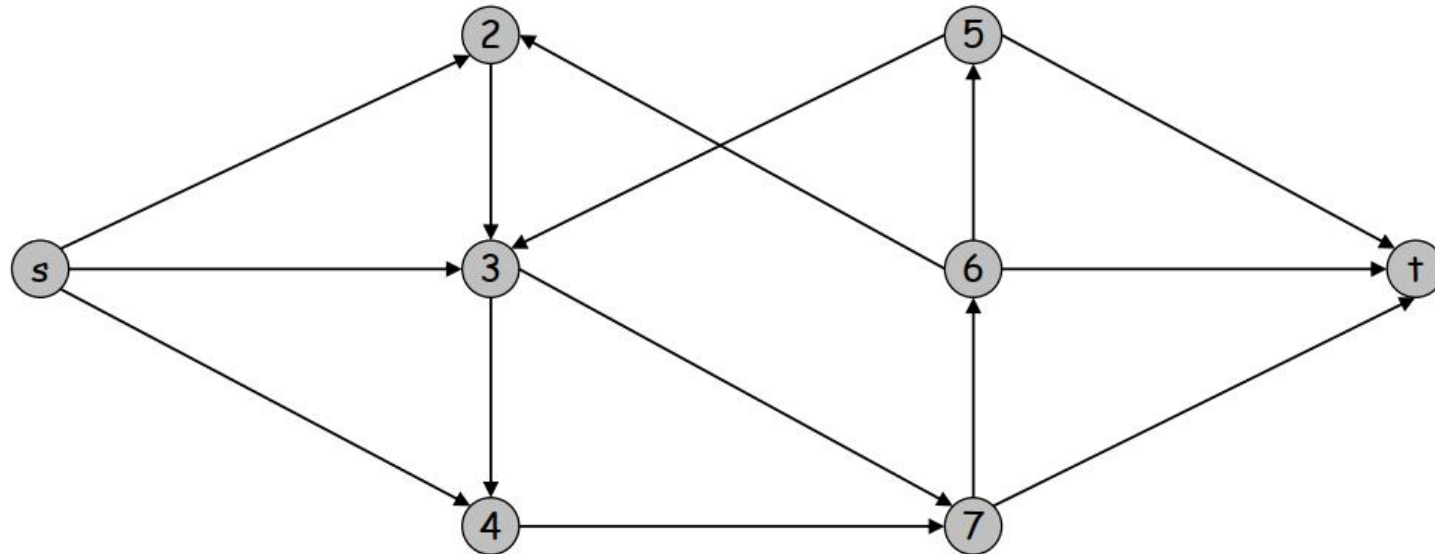




Disjoint Paths

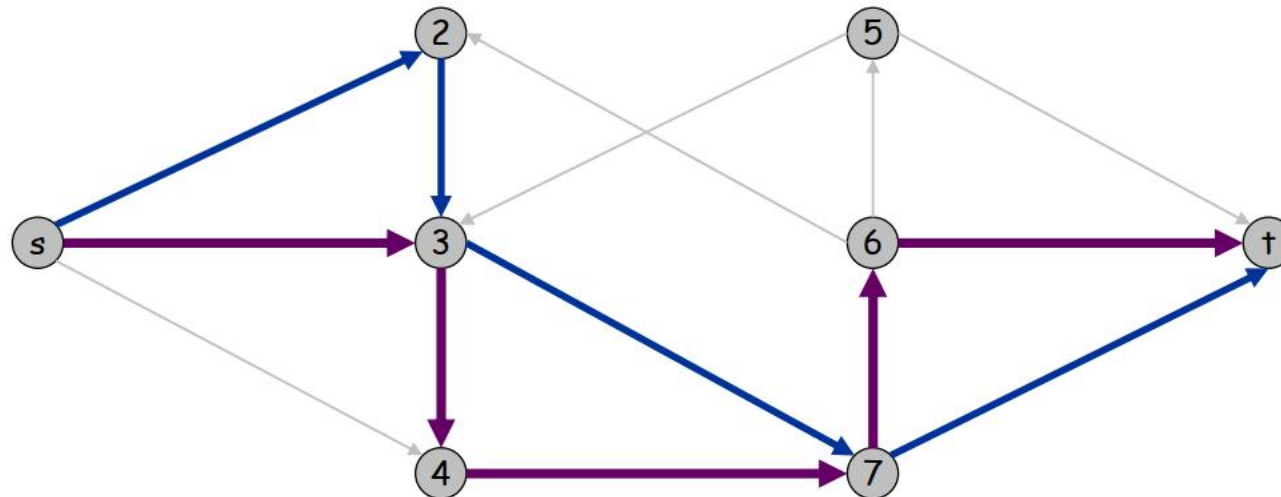
Edge Disjoint Paths

- **Disjoint path problem.** Given a digraph $G = (V, E)$ and two nodes s and t , find the max number of edge-disjoint s - t paths
- **Def.** Two paths are **edge-disjoint** if they have no edge in common
- **Ex:** communication networks



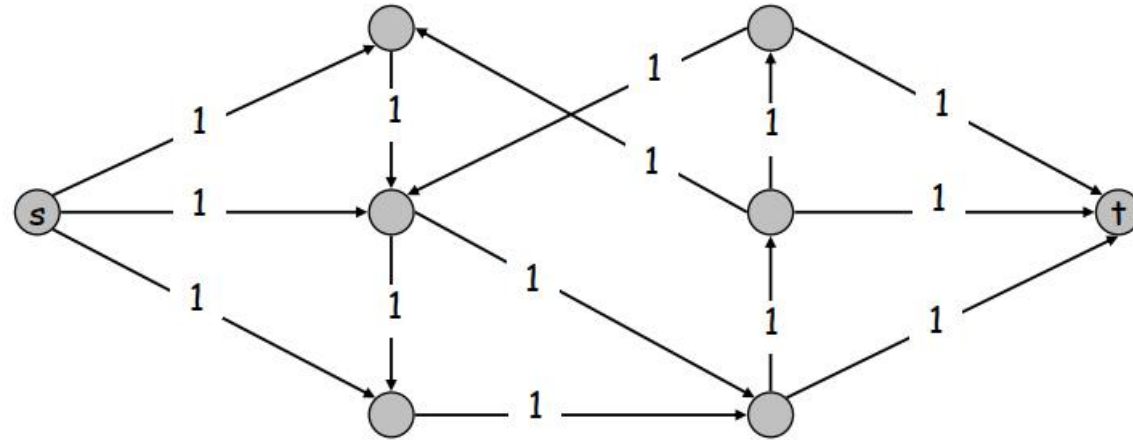
Edge Disjoint Paths

- **Disjoint path problem.** Given a digraph $G = (V, E)$ and two nodes s and t , find the max number of edge-disjoint s - t paths
- **Def.** Two paths are **edge-disjoint** if they have no edge in common
- **Ex:** communication networks



Edge Disjoint Paths

- **Max flow formulation:** assign unit capacity to every edge



- **Theorem.** Max number edge-disjoint s-t paths equals to max flow value

Edge Disjoint Paths

- **Theorem.** Max number edge-disjoint s-t paths equals max flow value
- **Pf. \Leftarrow**
 - Suppose there are k edge-disjoint paths P_1, \dots, P_k
 - Set $f(e) = 1$ if e participates in some path P_i ; else set $f(e) = 0$
 - Since paths are edge-disjoint, f is a flow of value k

Edge Disjoint Paths

- **Theorem.** Max number edge-disjoint s - t paths equals max flow value
- **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 conservation, there exists an edge (u, v) with $f(u, v) = 1$
 - Continue until reach t , always choosing a new edge
 - So we get a s - t path
- Reduce the flow to 0 along the path, so we get a flow of value $k-1$
- Repeat the process for k times, then we get k (not necessarily simple) edge-disjoint paths

Can eliminate cycles to get simple paths if desired





Extensions to Max Flow



Circulation with Demands

- **Circulation with demands**

- Directed graph $G = (V, E)$
- Edge capacities $c(e), e \in E$
- Node supply and demands $d(v), v \in V$



demand if $d(v) > 0$; supply if $d(v) < 0$; transshipment if $d(v) = 0$

- **Def.** A **circulation** is a function that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)

- For each $v \in V$ $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

- **Circulation problem:** given (V, E, c, d) , does there exist a circulation?

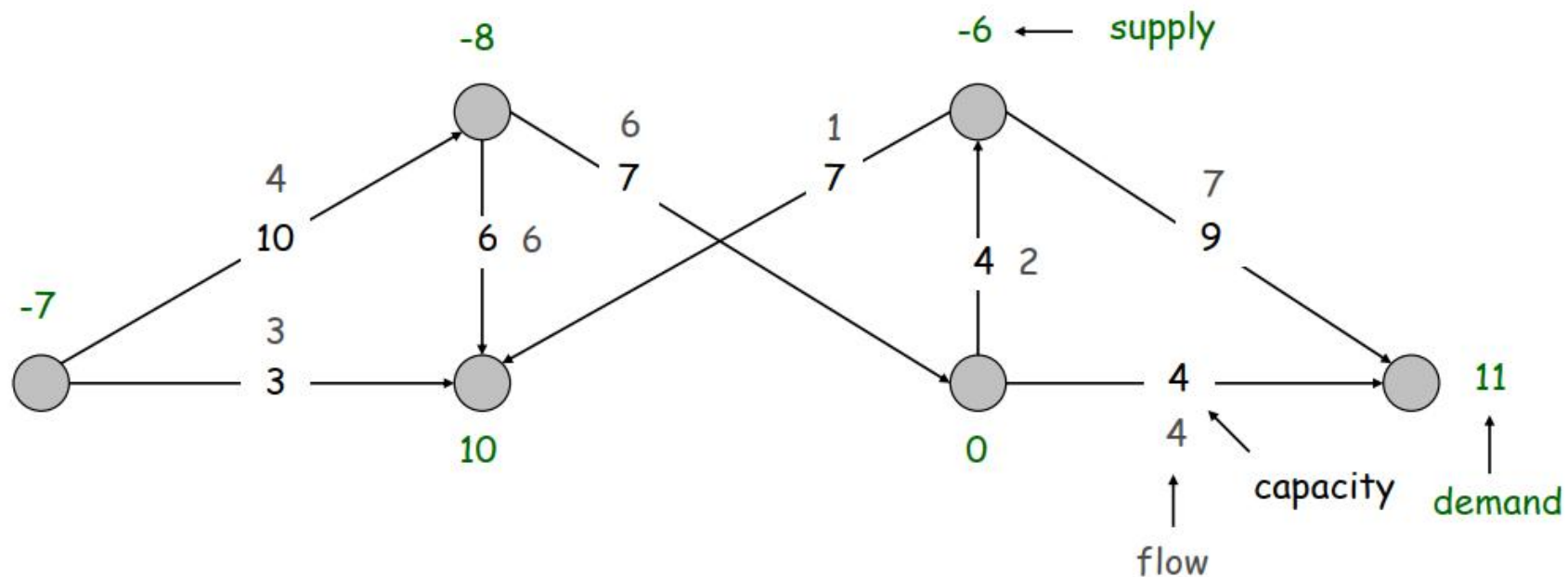


Circulation with Demands

- **Necessary condition:** sum of supplies = sum of demands

$$\sum_{v: d(v) > 0} d(v) = \sum_{v: d(v) < 0} -d(v) =: D$$

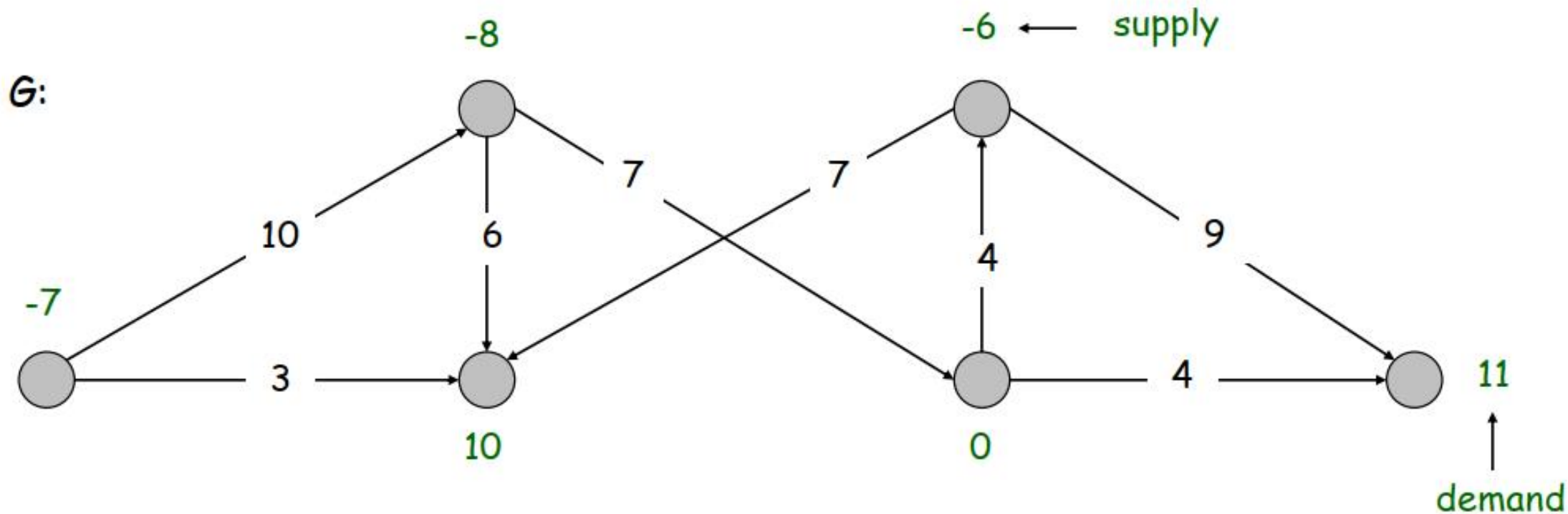
- **Pf.** Sum conservation constraints for every demand node v





Circulation with Demands

- Max flow formulation



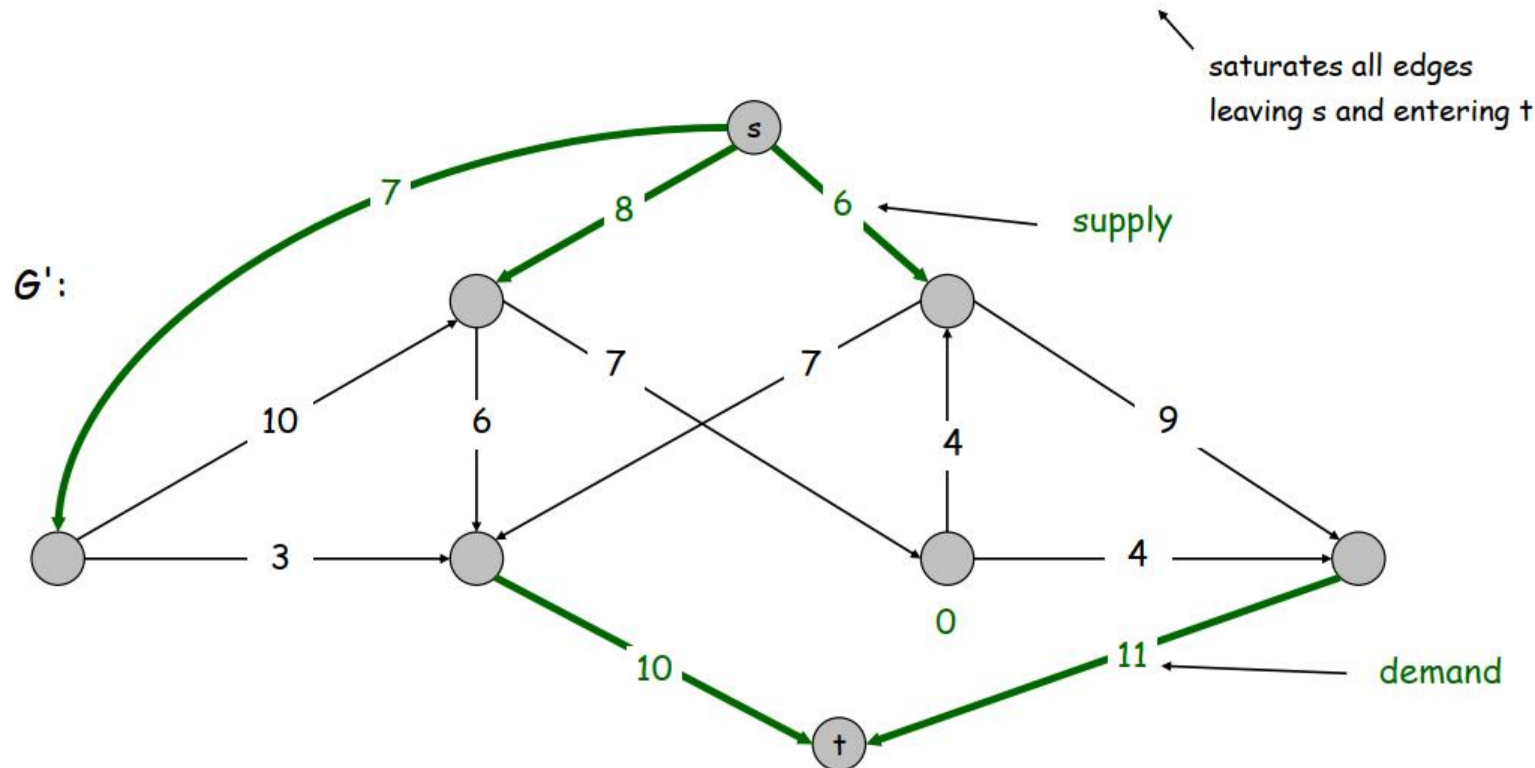


Circulation with Demands

- **Max flow formulation**

- Add new source s and sink t
- For each v with $d(v) < 0$, add edge (s, v) with capacity $-d(v)$
- For each v with $d(v) > 0$, add edge (v, t) with capacity $d(v)$

- **Claim.** G has circulation iff G' has max flow of value D





Circulation with Demands

- **Integrality theorem.** If all capacities and demands are integers, and there exists a circulation, then there exists one that is integer-valued
- **Pf.** Follows from max flow formulation and integrality theorem for max flow
- **Characterization.** Given (V, E, c, d) , there does **not** exist a circulation iff there exists a node partition (A, B) such that $\sum_{v \in B} d_v > \text{cap}(A, B)$



demand by nodes in B exceeds supply
of nodes in B plus max capacity of
edges going from A to B

- **Pf idea.** Look at max flow and min cut in G'



Circulation with Demands and Lower Bounds

- **Feasible circulation**

- Directed graph $G = (V, E)$
- Edge capacities $c(e)$ and **lower bounds** $l(e)$, $e \in E$
- Node supply and demands $d(v)$, $v \in V$

- **Def.** A **circulation** is a function that satisfies:

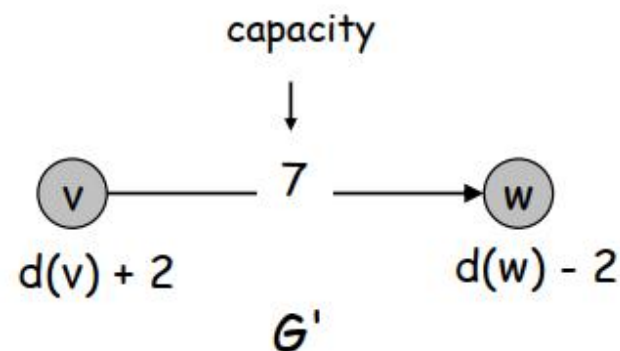
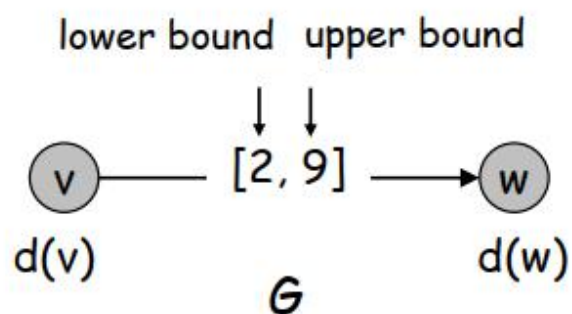
- For each $e \in E$: $l(e) \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V$: $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

- **Circulation problem with lower bounds.** Given (V, E, l, c, d) , does there exist a circulation?



Circulation with Demands and Lower Bounds

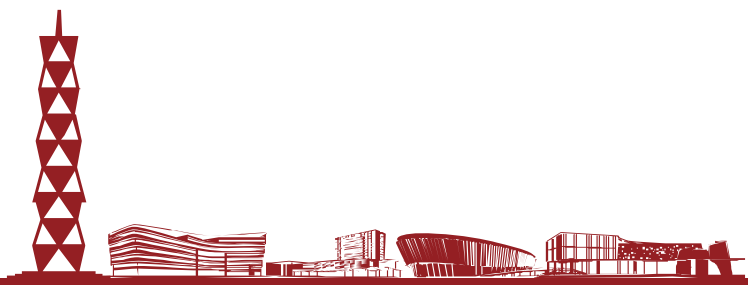
- **Idea.** Model lower bounds with demands
 - Send $\ell(e)$ units of flow along edge e
 - Update demands of both endpoints



- **Theorem.** There exists a circulation in G iff there exists a circulation in G' . If all demands, capacities, and lower bounds in G are integers, then there is a circulation in G that is integer-valued
- **Pf sketch.** $f(e)$ is a circulation in G iff $f'(e) = f(e) - \ell(e)$ is a circulation in G'



Survey Design

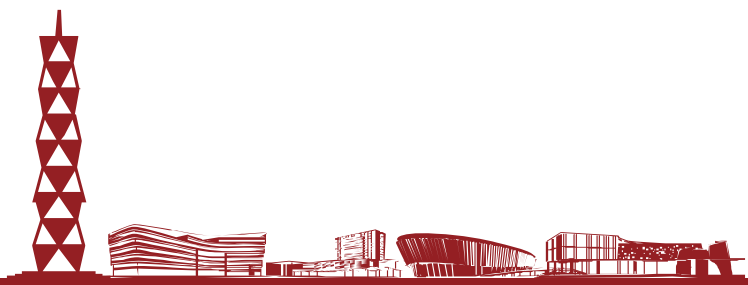




Survey Design



- **Survey design**
 - Design survey asking n_1 consumers about n_2 products
 - Can only survey consumer i about a product j if they own it
 - Ask consumer i between c_i and c'_i questions
 - Ask between p_j and p'_j consumers about product j
- **Goal.** Design a survey that meets these specs, if possible

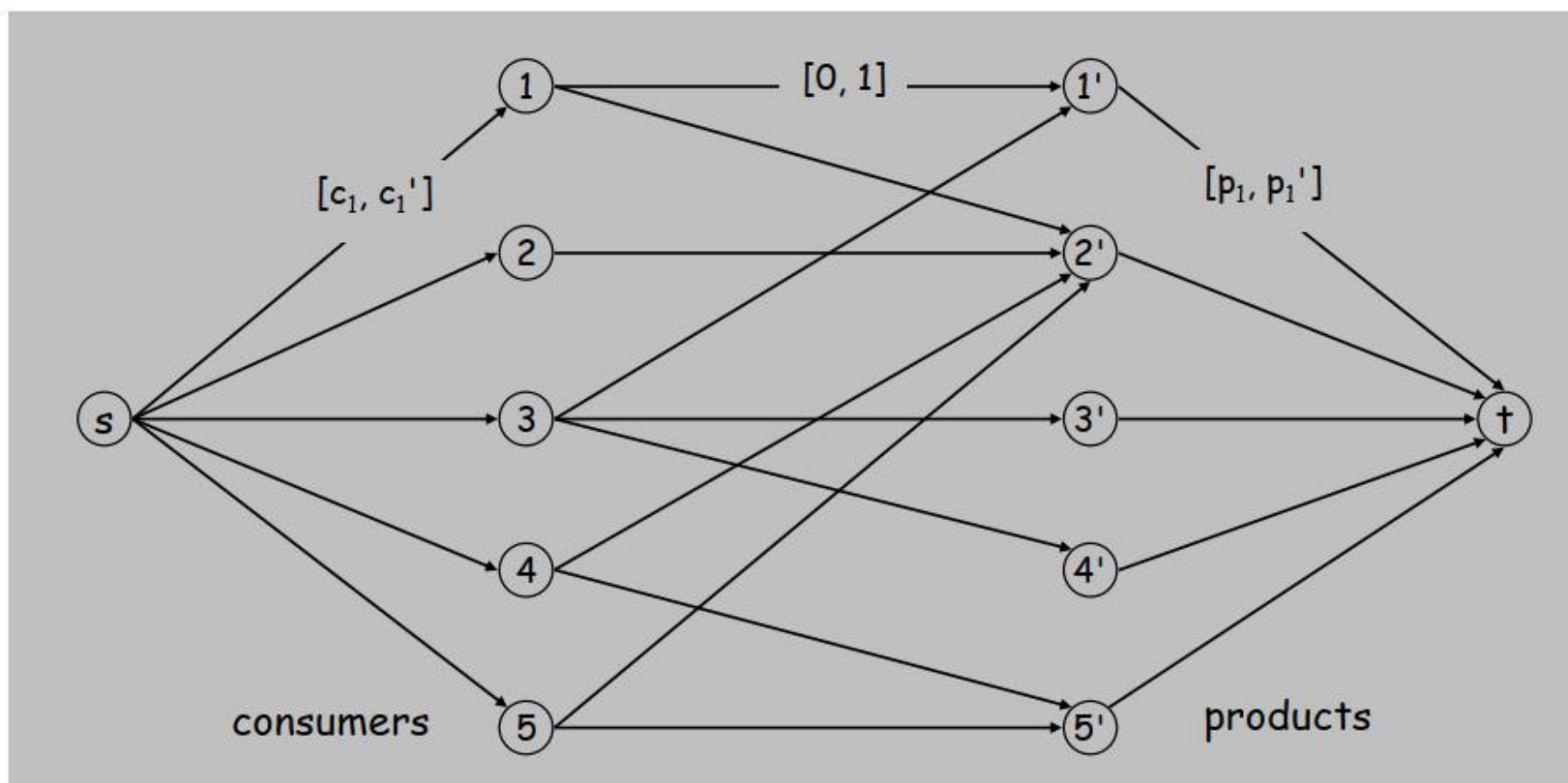




Survey Design



- **Algorithm.** Formulate as a flow-network?
 - Include an edge (i, j) if customer own product i
 - Goal: find a flow that satisfies edge upper & lower bounds. How?





Survey Design



- **Algorithm.** Formulate as a circulation problem with lower bounds
 - Include an edge (i, j) if customer own product i
 - Integer circulation \rightarrow feasible survey design

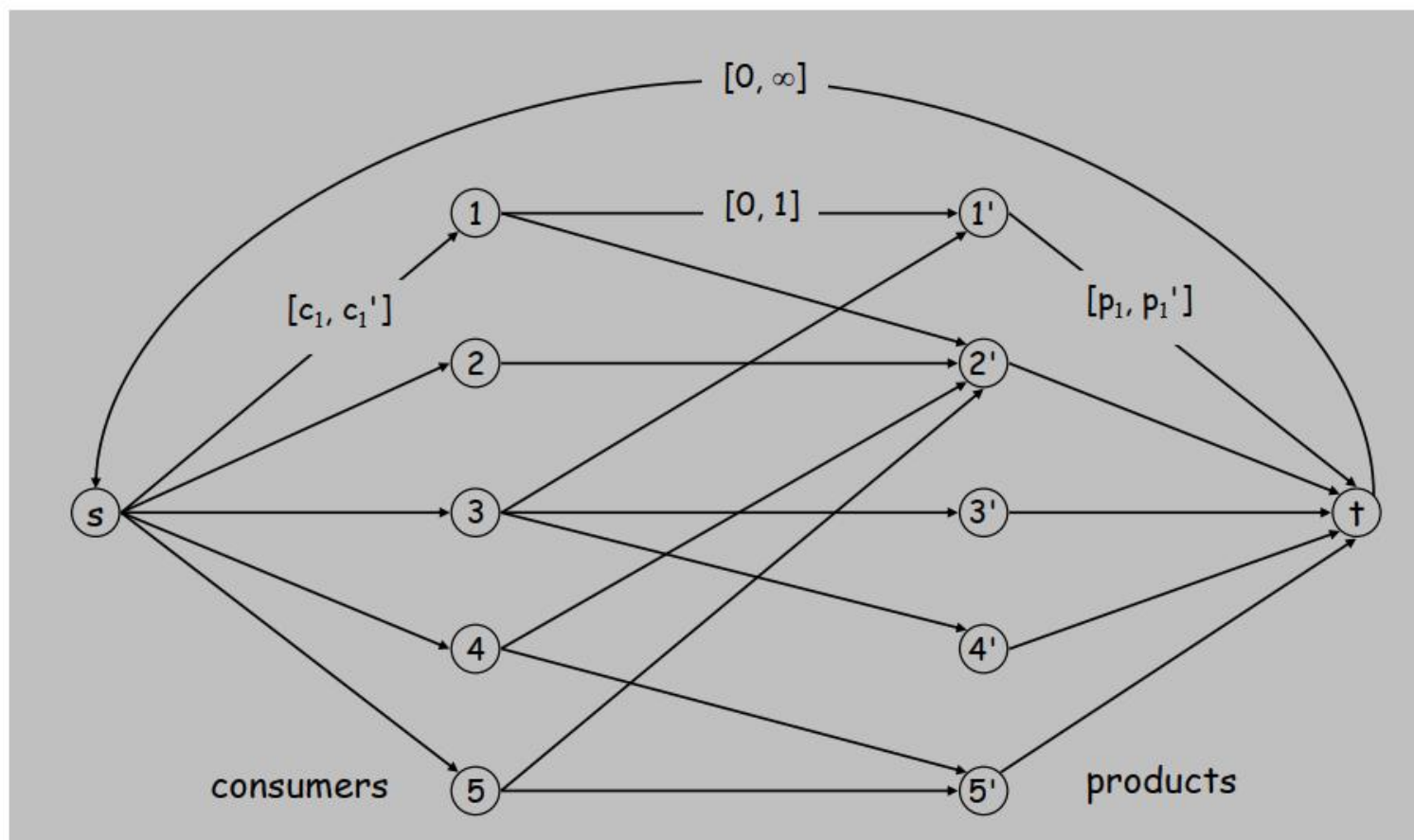




Image Segmentation

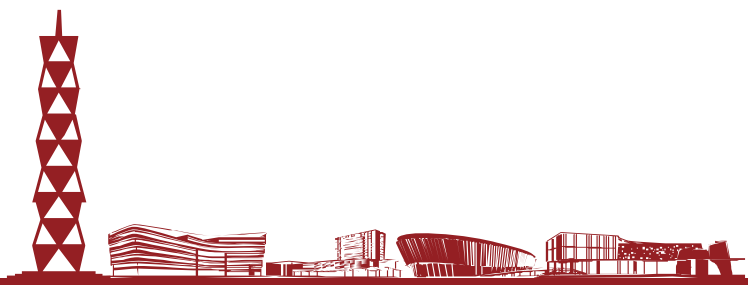




Image Segmentation



- **Image segmentation**

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

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



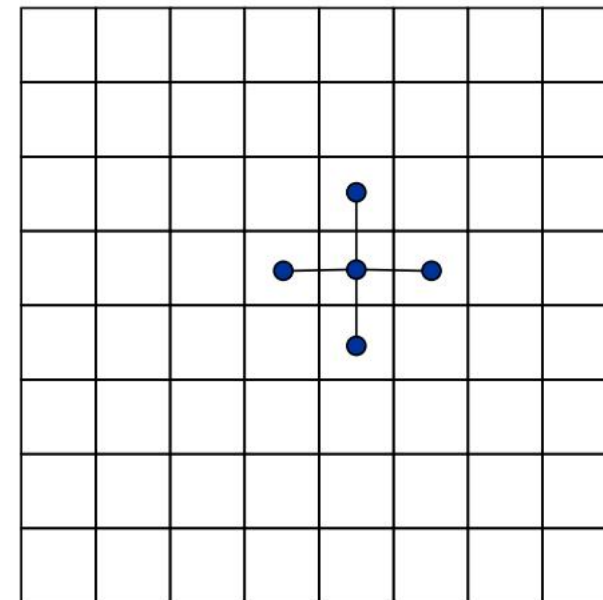


Image Segmentation



- **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:

↗ ↖
foreground background

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



Image Segmentation



- **Formulate as min cut problem**

- Maximization
- No source or sink
- Undirected graph

- **Turn into minimization problem**

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

is equivalent to minimizing
$$\underbrace{\left(\sum_{i \in V} a_i + \sum_{j \in V} b_j \right)}_{\text{a constant}} - \sum_{i \in A} a_i - \sum_{j \in B} b_j + \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

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



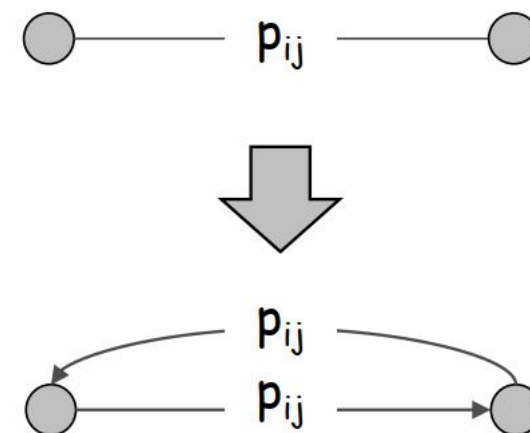
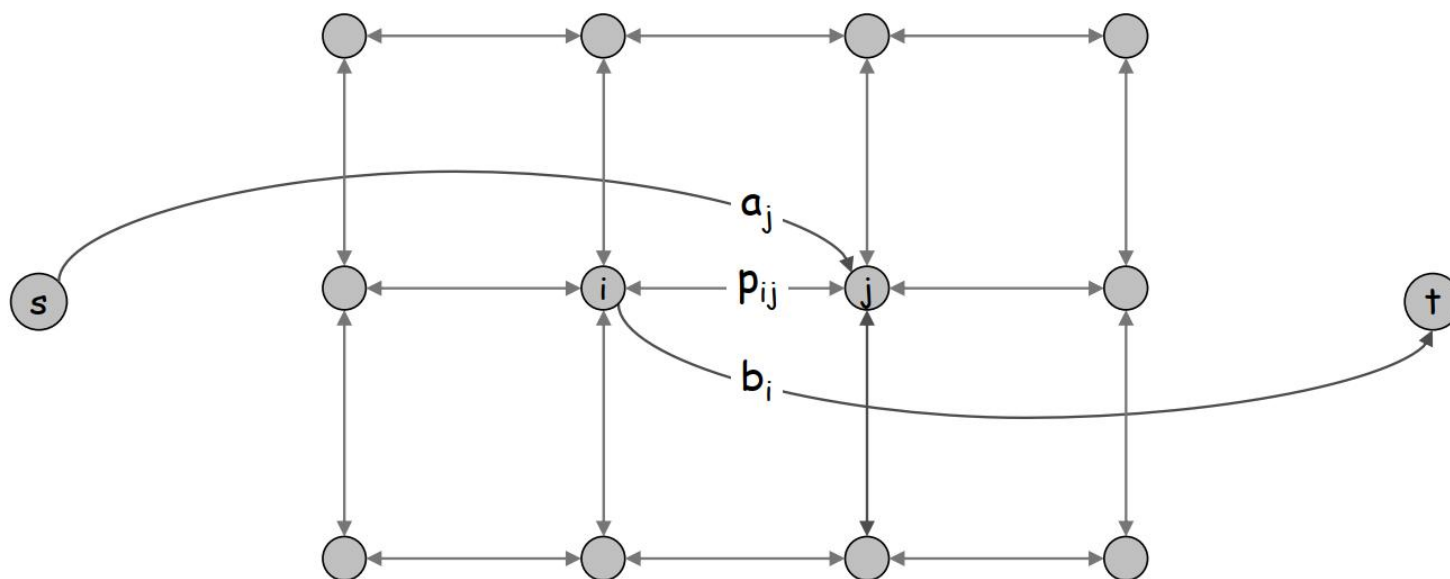


Image Segmentation



- **Formulate as min cut problem**

- $G' = (V', E')$
- Add source to correspond to foreground; add sink to correspond to background
- Use two anti-parallel edges instead of undirected edge



G'





Image Segmentation



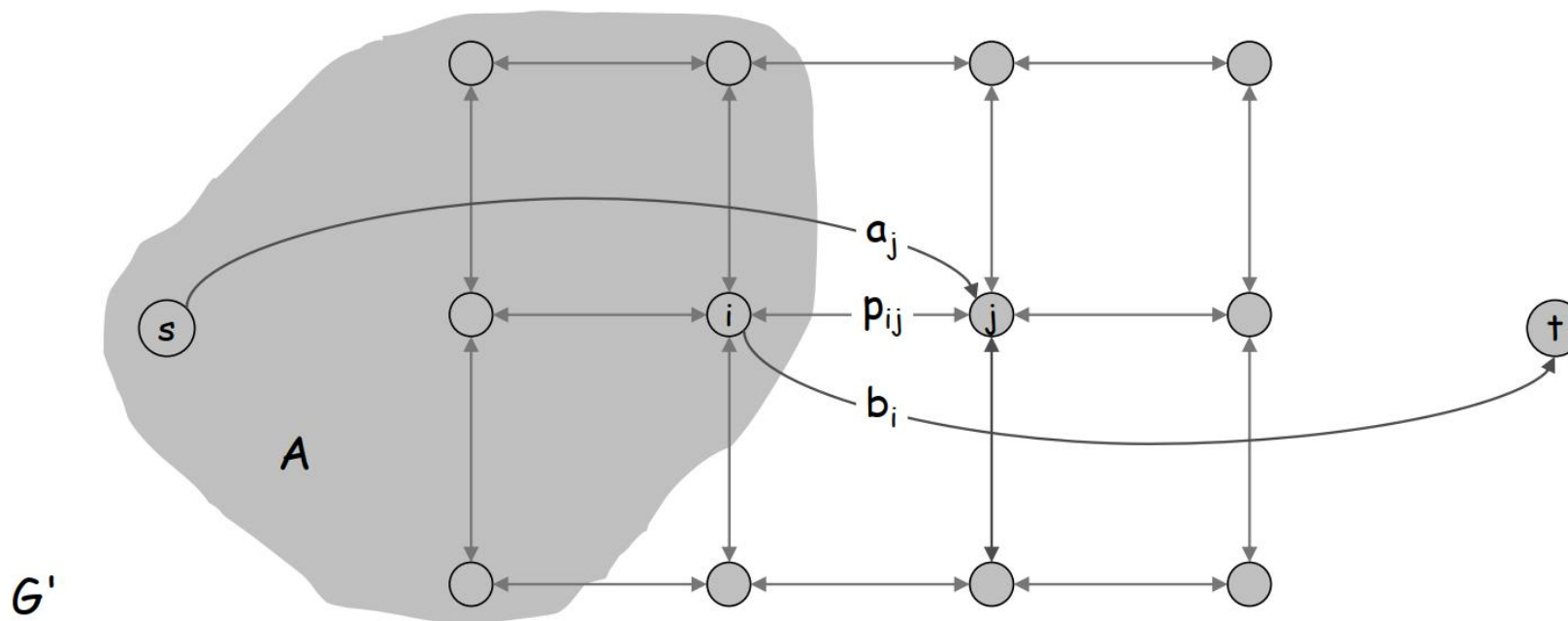
- Consider min cut (A, B) in G'

- A = foreground

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

← if i and j on different sides, p_{ij} counted exactly once

- Precisely the quantity we want to minimize





**Next Time:
Network Flow (Cont.)**