

# Applications of max-flow

---

# Maximum Flow and Minimum Cut

Max flow and min cut: many applications / reductions.

- Data mining.
- Open-pit mining.
- Project selection.
- Airline scheduling.
- Bipartite matching.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Network reliability.
- Distributed computing.
- Egalitarian stable matching.
- Security of statistical data.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- Many many more . . .

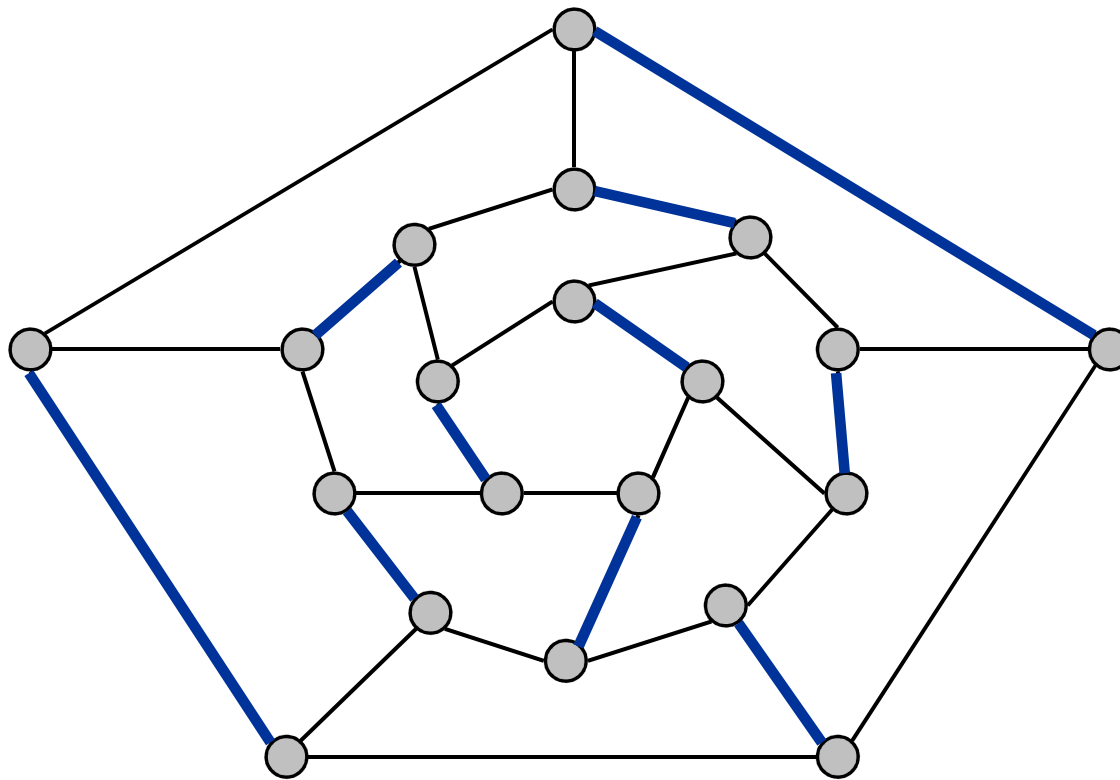
## 7.5 Bipartite Matching

---

# Matching

## Matching.

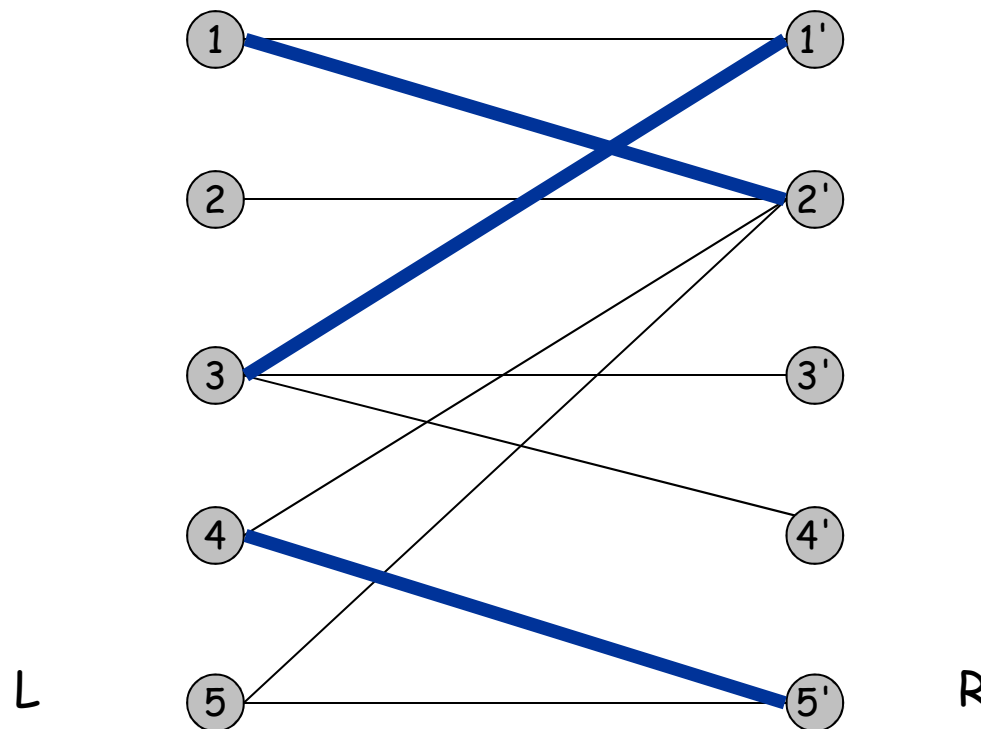
- Input: undirected graph  $G = (V, E)$ .
- $M \subseteq E$  is a **matching** if each node appears in at most one edge in  $M$ .
- Max matching: find a max cardinality matching.



# 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.

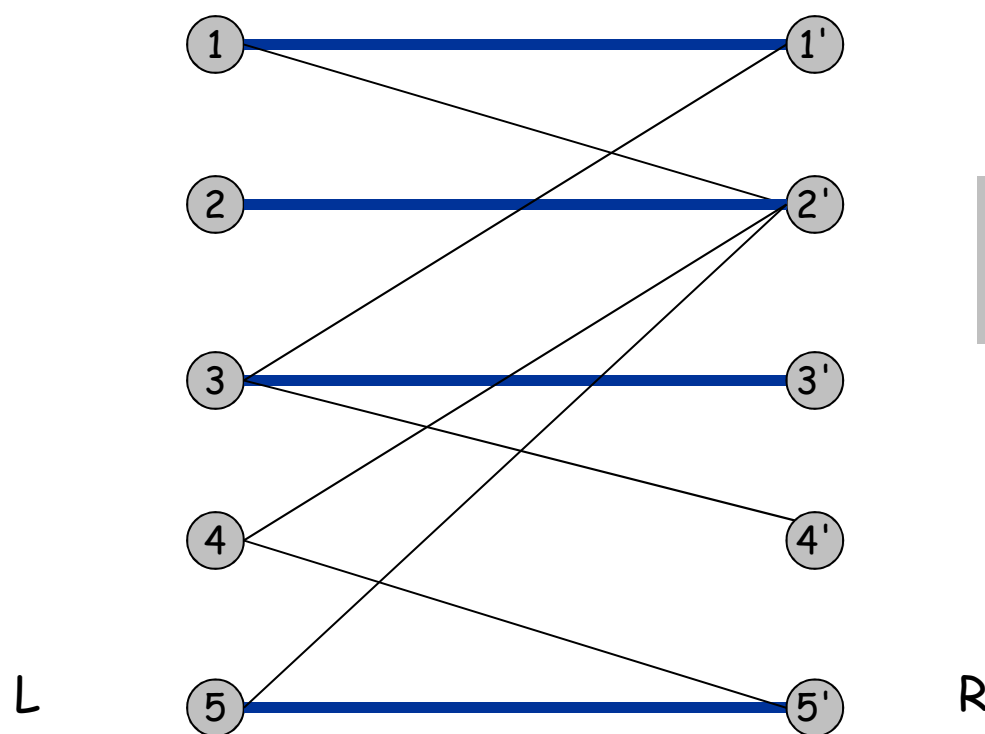


matching  
1-2', 3-1', 4-5'

# 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.

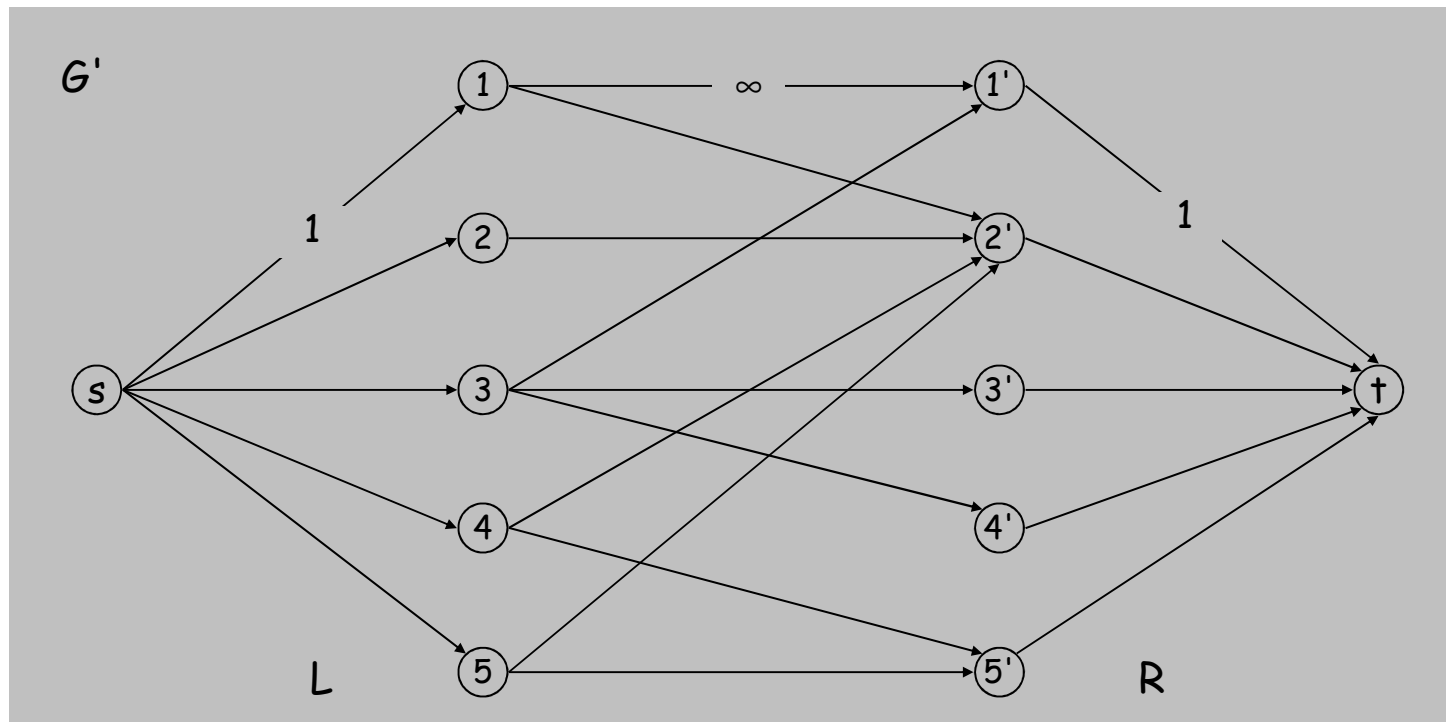


max matching  
1-1', 2-2', 3-3' 4-4'

## 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  $t$ .

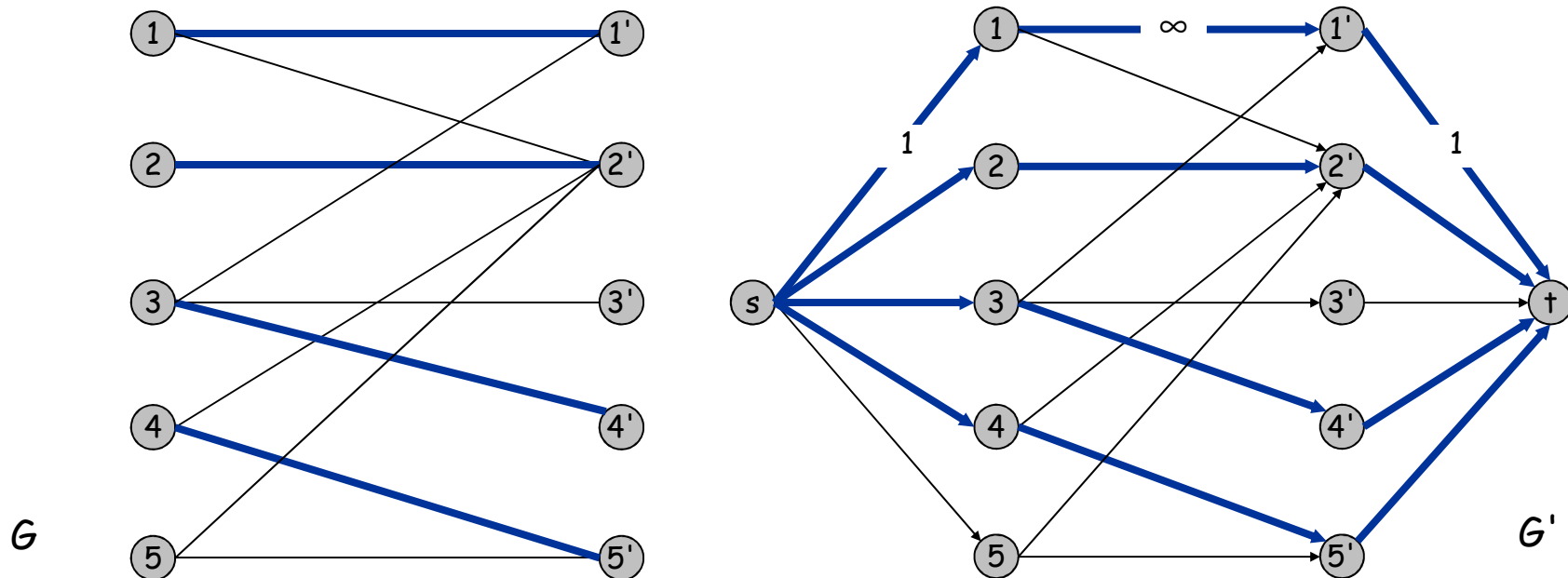


## Bipartite Matching: Proof of Correctness

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

**Pf.**  $\leq$

- 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  $k$ . ▪



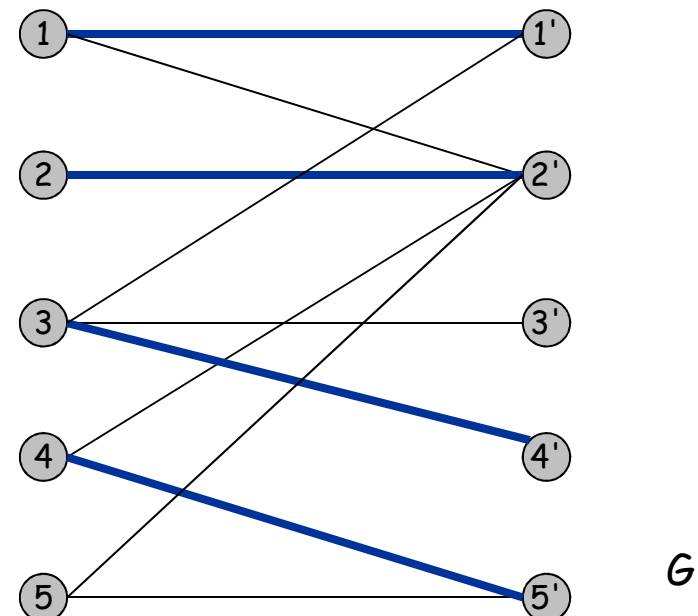
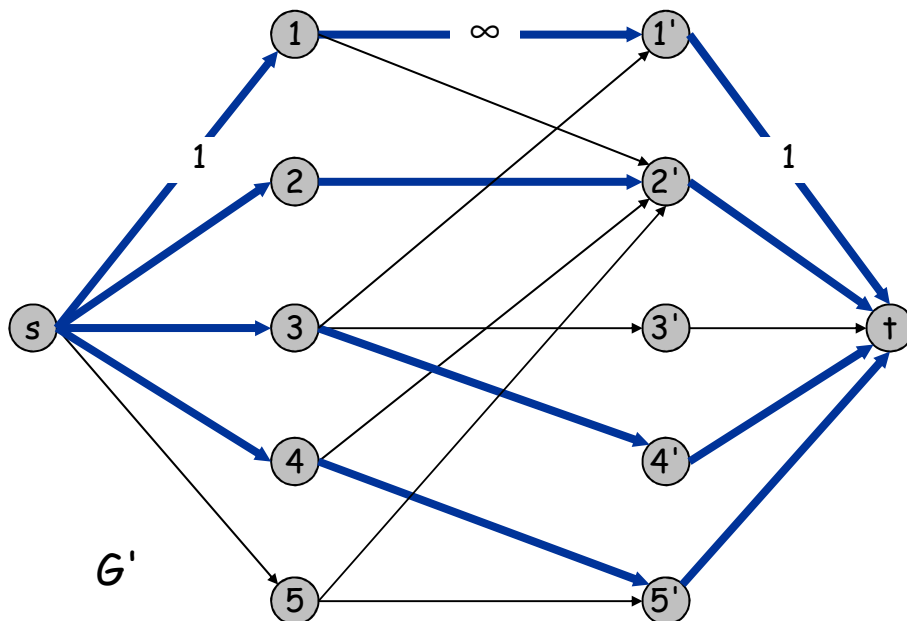


## Bipartite Matching: Proof of Correctness

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

**Pf.**  $\geq$

- 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

**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.

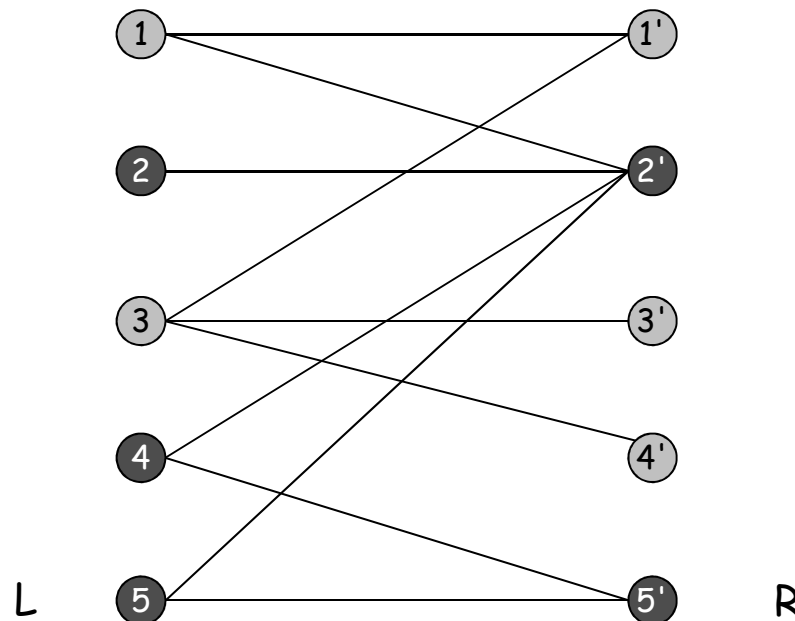
- Clearly we must have  $|L| = |R|$ .
- What other conditions are necessary?
- What conditions are sufficient?

## 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:

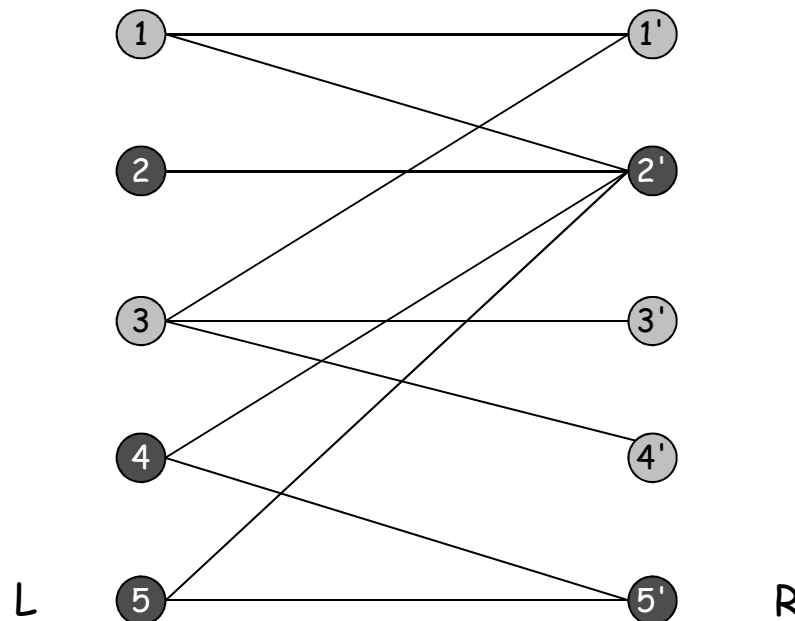
$S = \{ 2, 4, 5 \}$

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

## 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  $S \subseteq L$ .

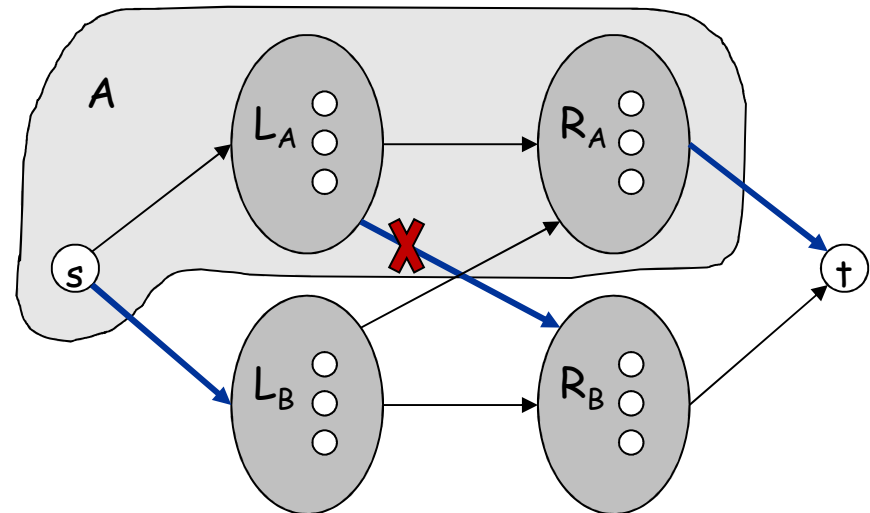
**Pf.**  $\Leftarrow$  Suppose  $G$  does not have a perfect matching.

- 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  $\infty$  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.



## Bipartite Matching: Running Time

Which max flow algorithm to use for bipartite matching?

- Generic augmenting path:  $O(m \text{ val}(f^*)) = O(mn)$ .
- Capacity scaling:  $O(m^2 \log C) = O(m^2)$ .
- Shortest augmenting path:  $O(m n^{1/2})$ .

Non-bipartite matching.

- Structure of non-bipartite graphs is more complicated, but well-understood. [Tutte-Berge, Edmonds-Galai]
- Blossom algorithm:  $O(n^4)$ . [Edmonds 1965]
- Best known:  $O(m n^{1/2})$ . [Micali-Vazirani 1980]

## 7.6 Disjoint Paths

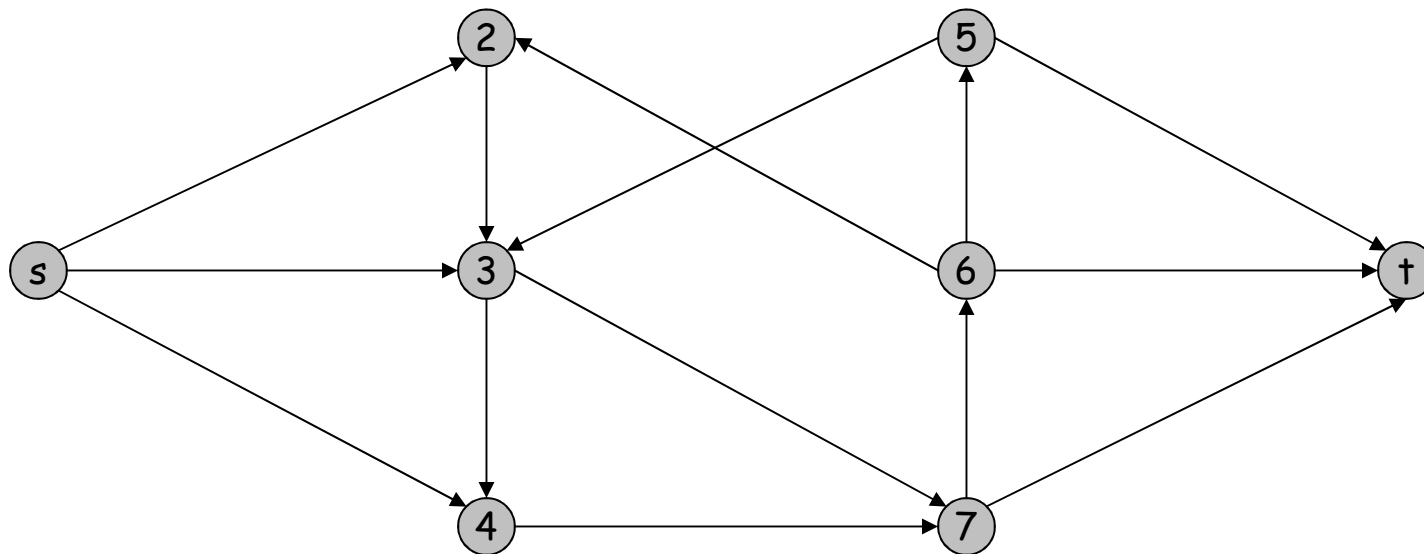
---

## 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.



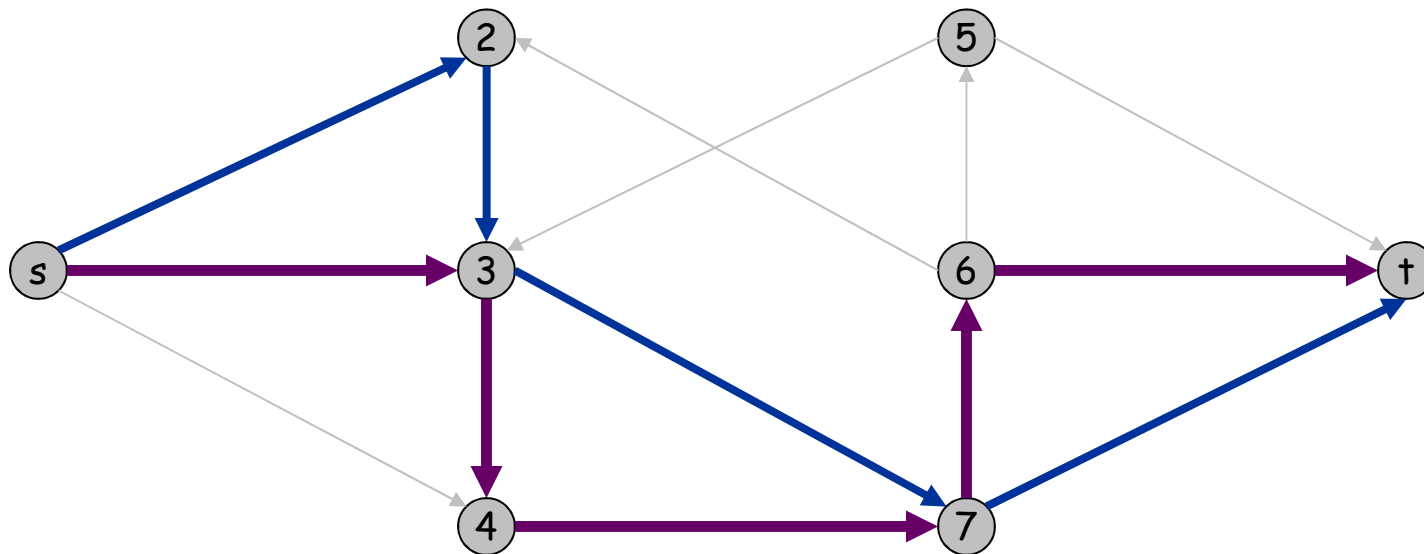


## Edge Disjoint Paths

**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.

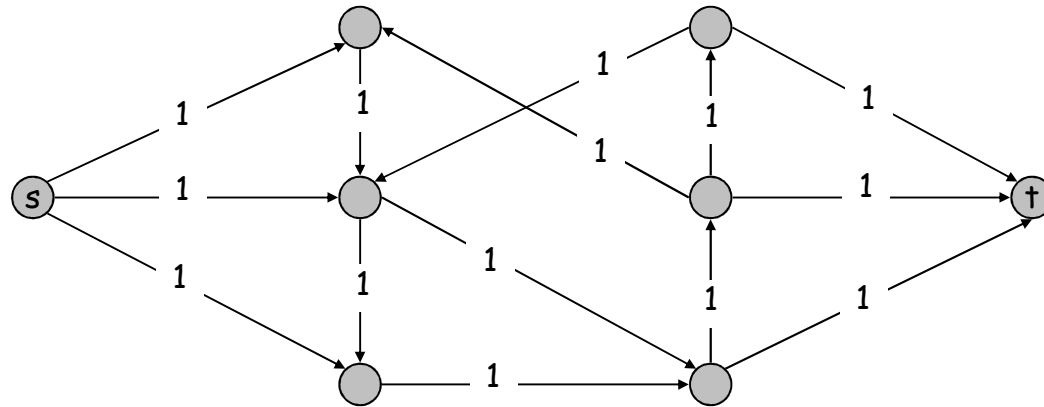
**Def.** Two paths are **edge-disjoint** if they have no edge in common.

**Ex:** communication networks.



## Edge Disjoint Paths

Max flow formulation: assign unit capacity to every edge.



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

## Edge Disjoint Paths

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

**Pf.**  $\leq$

- 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.**  $\geq$

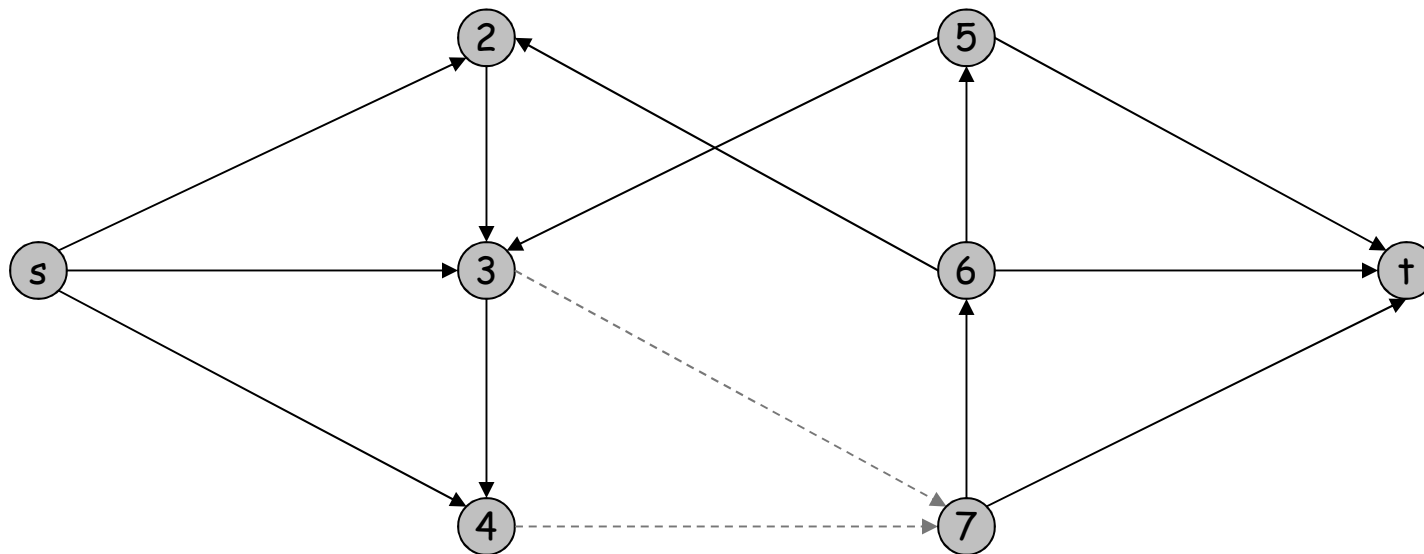
- Suppose max flow value is  $k$ .
- Integrality theorem  $\Rightarrow$  there exists 0-1 flow  $f$  of value  $k$ .
- Consider edge  $(s, u)$  with  $f(s, u) = 1$ .
  - by conservation, there exists an edge  $(u, v)$  with  $f(u, v) = 1$
  - continue until reach  $t$ , always choosing a new edge
  -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

## Network Connectivity

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

**Def.** A set of edges  $F \subseteq E$  **disconnects  $t$  from  $s$**  if all  $s$ - $t$  paths uses at least on edge in  $F$ .

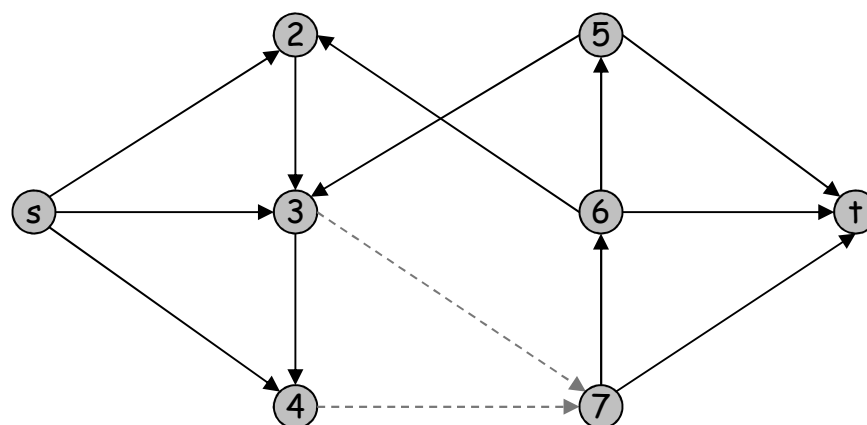
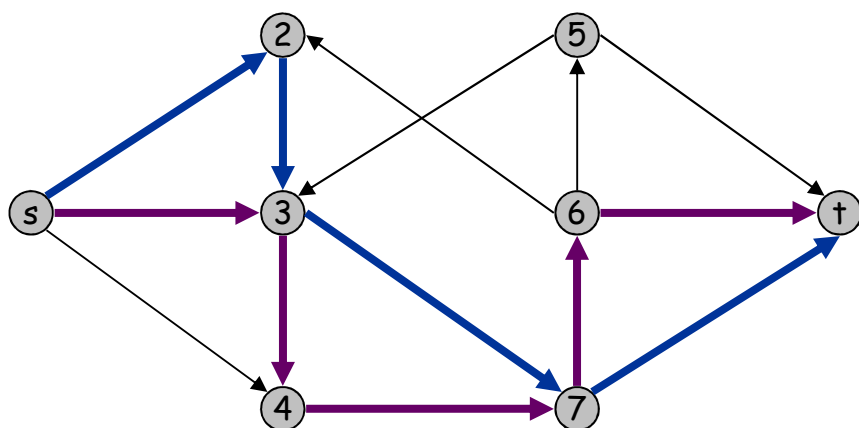


## Edge Disjoint Paths and Network Connectivity

**Theorem.** [Menger 1927] The max number of edge-disjoint  $s$ - $t$  paths is equal to the min number of edges whose removal disconnects  $t$  from  $s$ .

Pf.  $\leq$

- Suppose the removal of  $F \subseteq E$  disconnects  $t$  from  $s$ , and  $|F| = k$ .
- All  $s$ - $t$  paths use at least one edge of  $F$ . Hence, the number of edge-disjoint paths is at most  $k$ . ▪

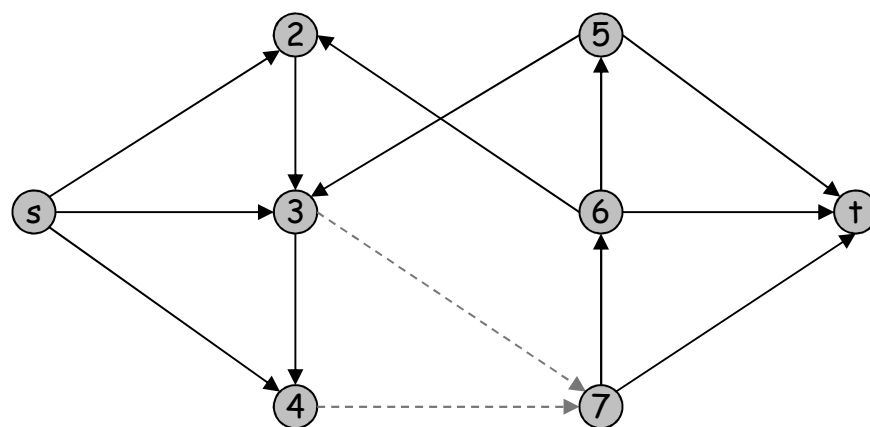
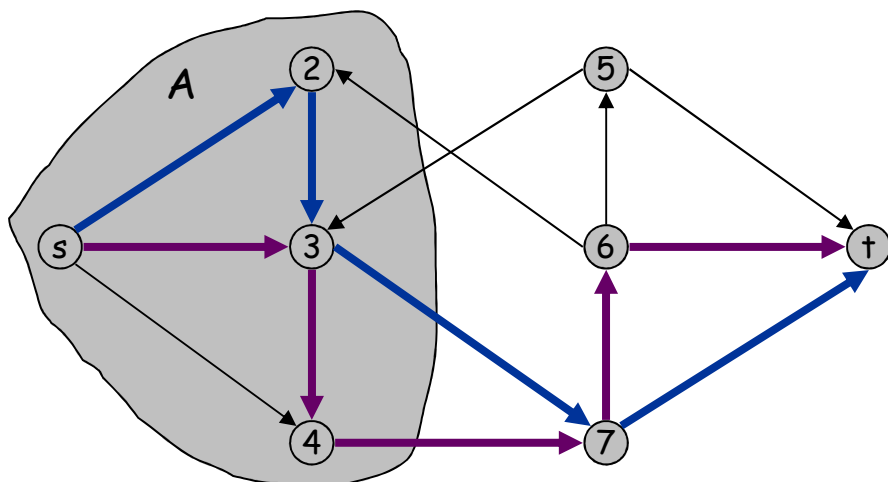


## Disjoint Paths and Network Connectivity

**Theorem.** [Menger 1927] The max number of edge-disjoint  $s$ - $t$  paths is equal to the min number of edges whose removal disconnects  $t$  from  $s$ .

**Pf.**  $\geq$

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



## 7.7 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)

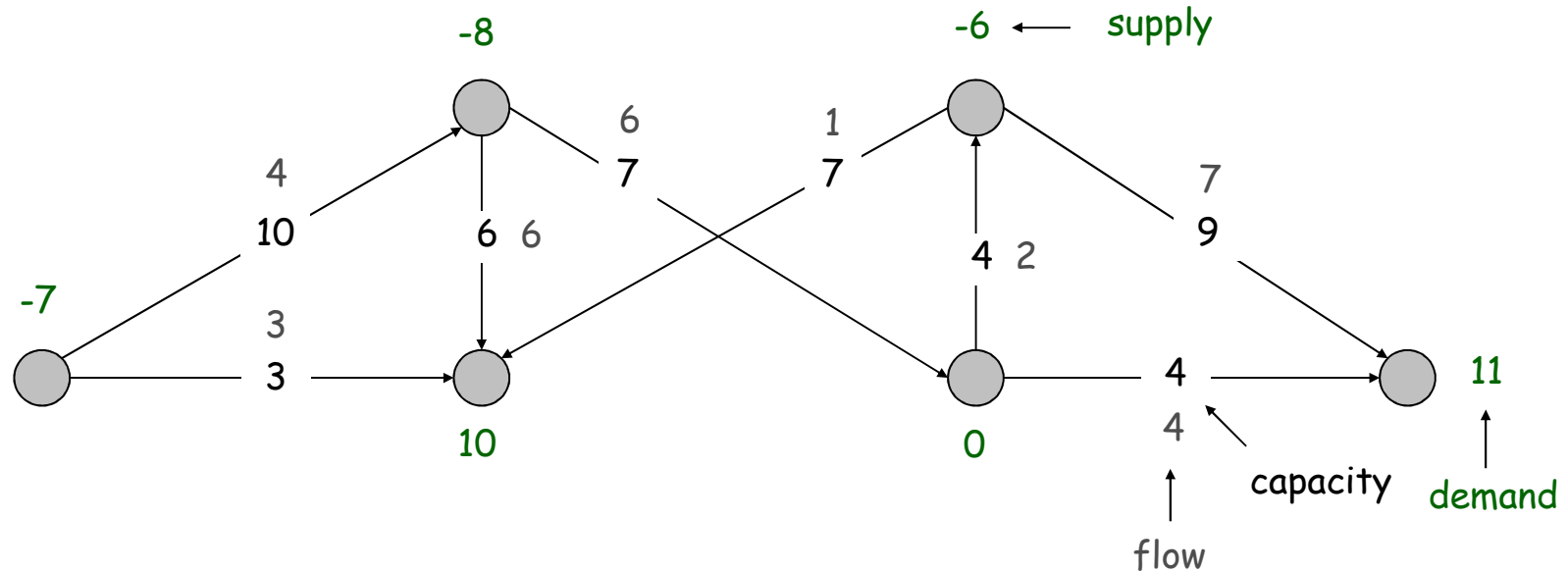
**Circulation problem:** given  $(V, E, c, d)$ , does there exist a circulation?

## Circulation with Demands

Necessary condition: sum of supplies = sum of demands.

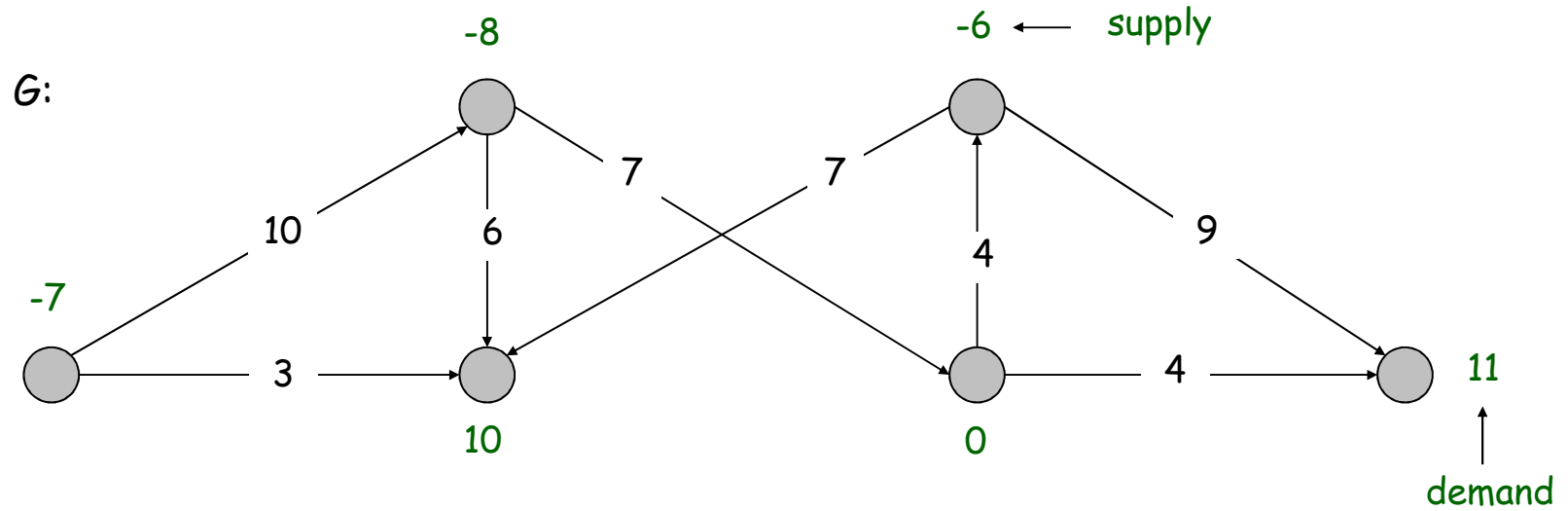
$$\sum_{v: d(v) > 0} d(v) = \sum_{v: d(v) < 0} -d(v) =: D$$

Pf. Sum conservation constraints for every demand node  $v$ .



## Circulation with Demands

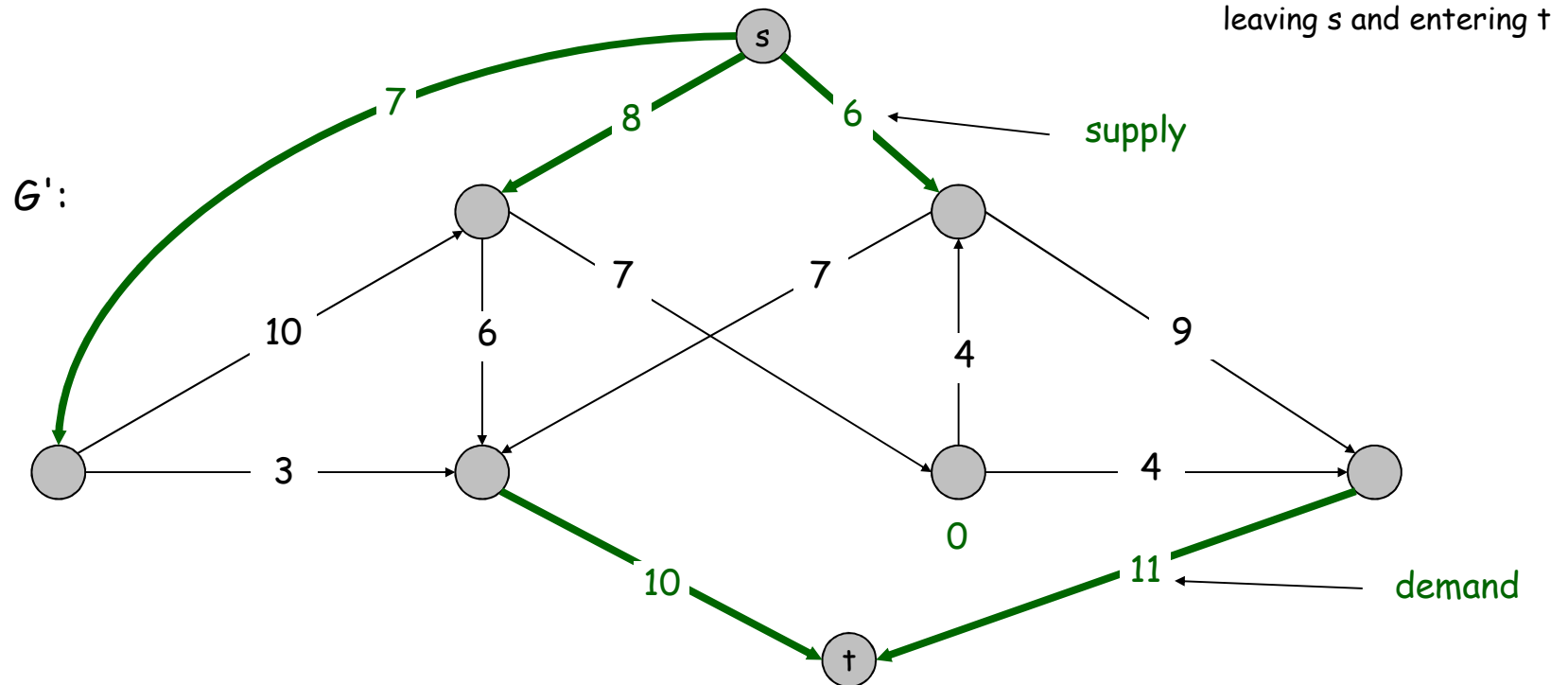
Max flow formulation.



## 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  $\sum_{v \in B} d_v > \text{cap}(A, B)$

**Pf idea.** Look at min cut in  $G'$ .

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

## Circulation with Demands and Lower Bounds

Feasible circulation.

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

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

- For each  $e \in E$ :  $\ell(e) \leq f(e) \leq c(e)$  (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  $(V, E, \ell, c, d)$ , does there exists 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  $G'$ .

## 7.8 Survey Design

---



# Survey Design

## Survey design.

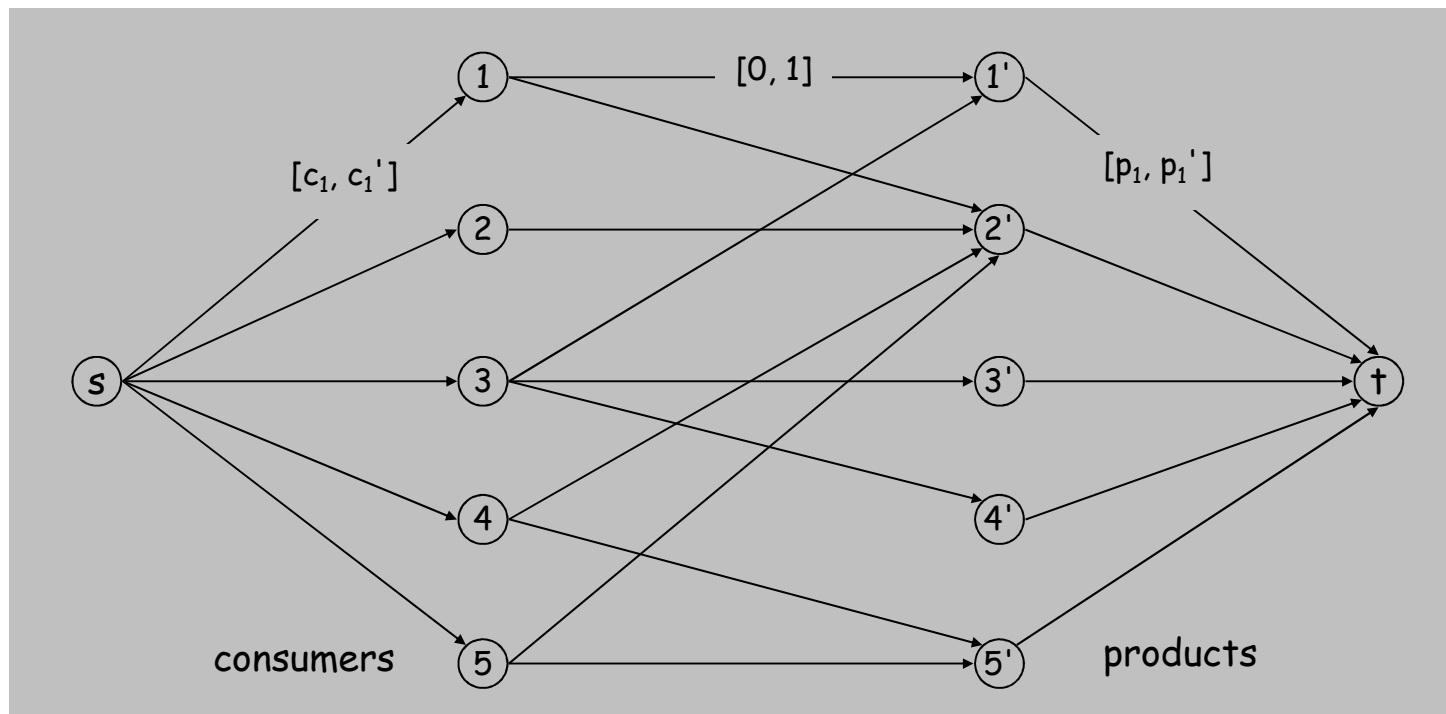
- Design survey asking  $n_1$  consumers about  $n_2$  products.
- Can only survey consumer  $i$  about a product  $j$  if they own it.
- Ask consumer  $i$  between  $c_i$  and  $c_i'$  questions.
- Ask between  $p_j$  and  $p_j'$  consumers about product  $j$ .

**Goal.** Design a survey that meets these specs, if possible.

## Survey Design

**Algorithm.** Formulate as a flow-network?

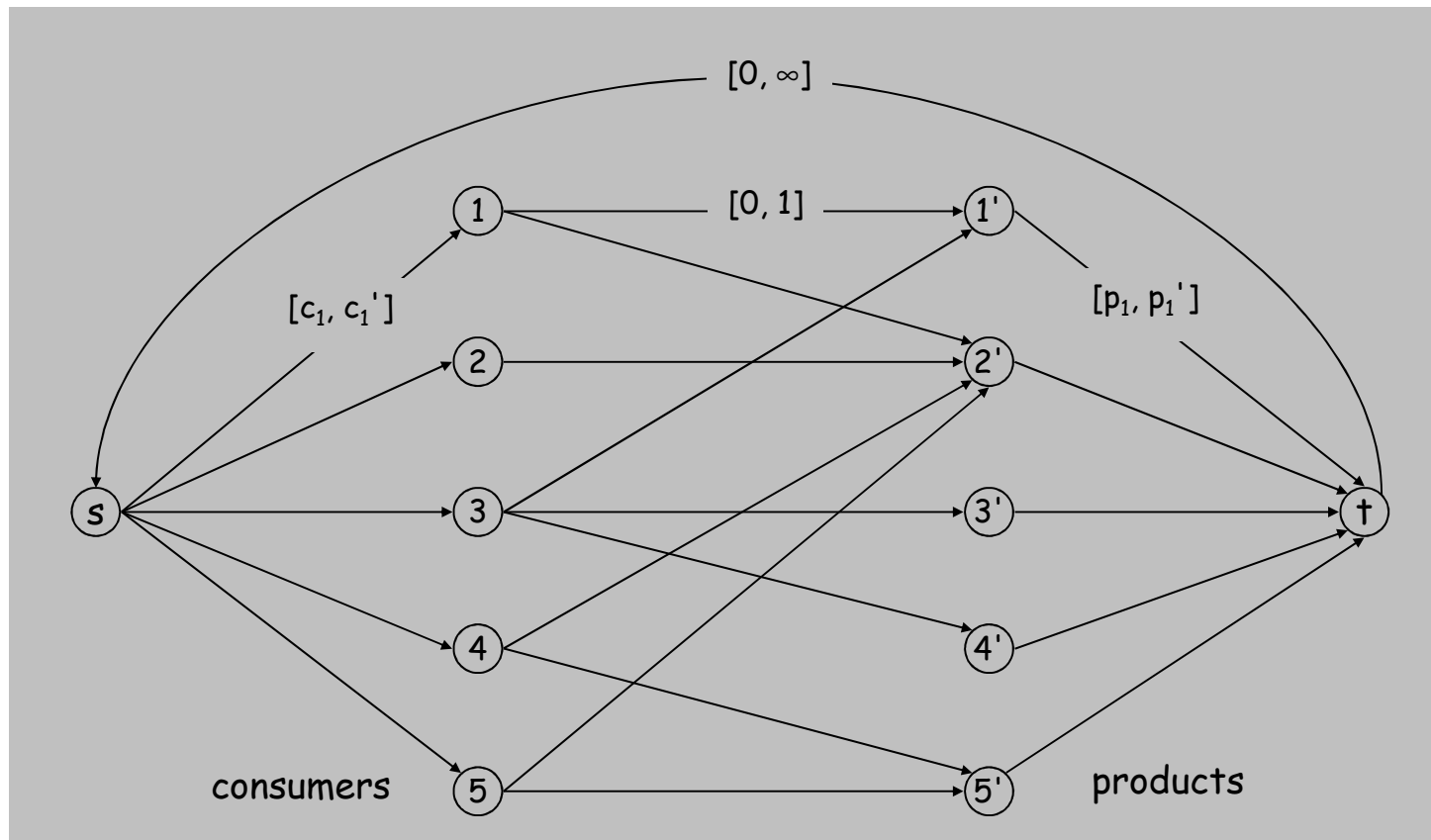
- Include an edge  $(i, j)$  if customer own product  $i$ .
- Goal: find a flow that satisfies edge upper&lower bounds. How?



## Survey Design

**Algorithm.** Formulate as a circulation problem with lower bounds.

- Include an edge  $(i, j)$  if customer own product  $i$ .
- Integer circulation  $\Leftrightarrow$  feasible survey design.



## 7.10 Image Segmentation

---

# Image Segmentation

## Image segmentation.

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

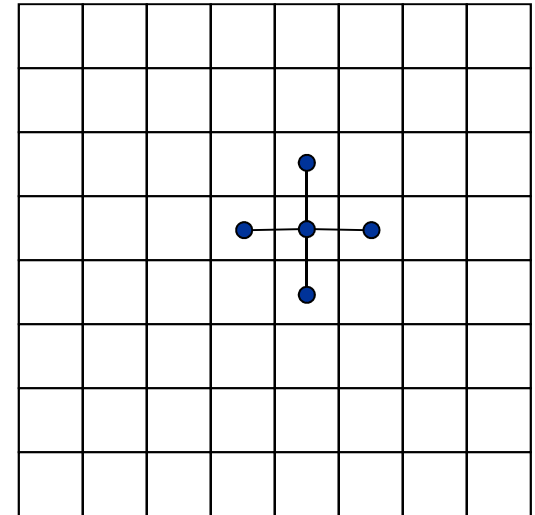
**Ex:** Two people standing in front of complex background scene.  
Identify each person as a coherent object.



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

$\nearrow$   
foreground

$\nwarrow$   
background

# Image Segmentation

Formulate as min cut problem.

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

Turn into minimization problem.

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

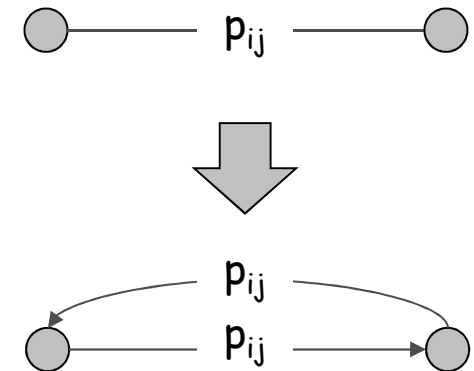
is equivalent to minimizing 
$$\underbrace{\left( \sum_{i \in V} a_i + \sum_{j \in V} b_j \right)}_{\text{a constant}} - \sum_{i \in A} a_i - \sum_{j \in B} b_j + \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}| = 1}} p_{ij}$$

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

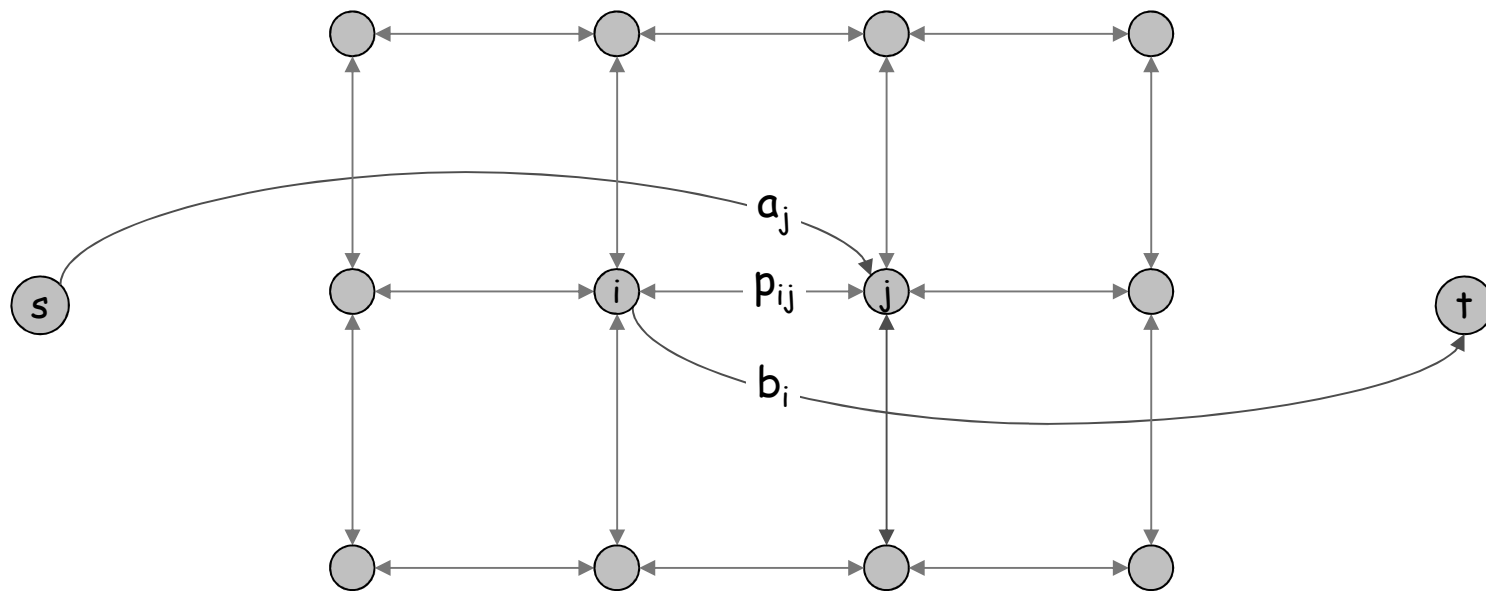
# Image Segmentation

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

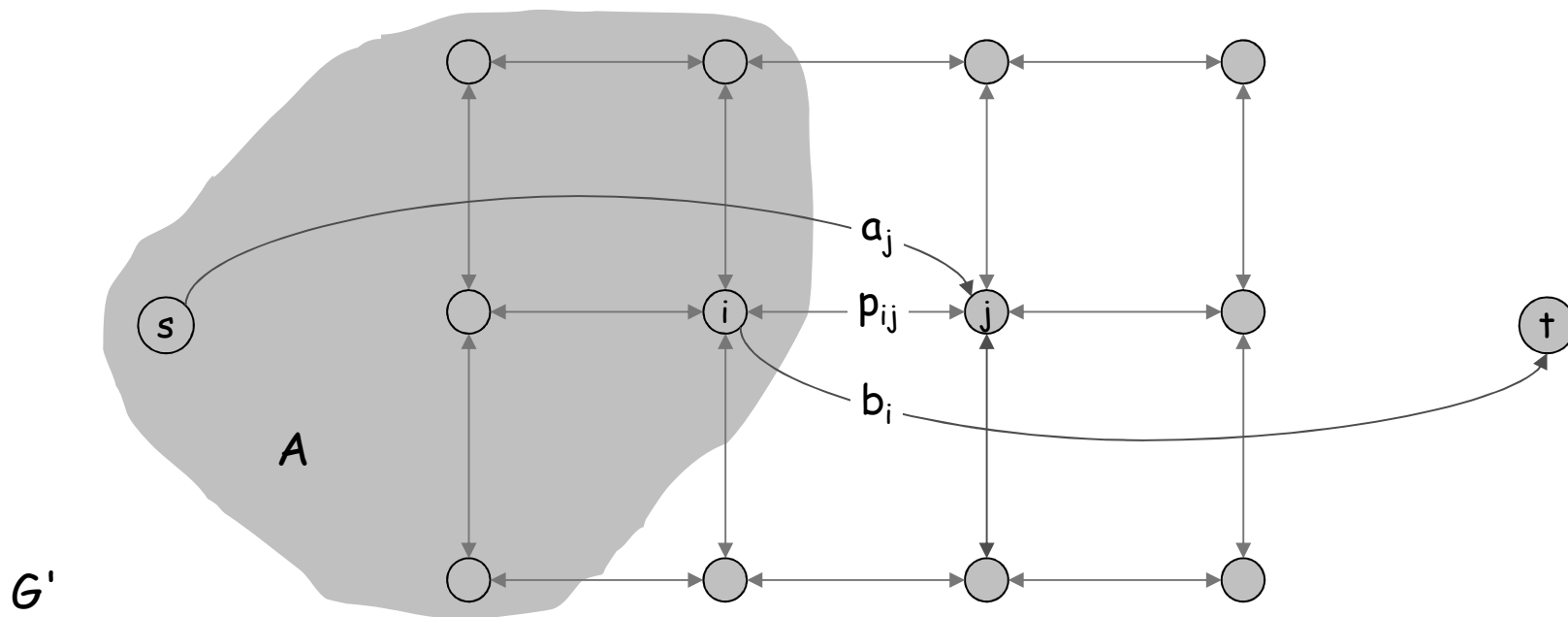
Consider min cut  $(A, B)$  in  $G'$ .

- $A$  = 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.



## 7.11 Project Selection

---

# Project Selection

## Projects with prerequisites.

can be positive or negative

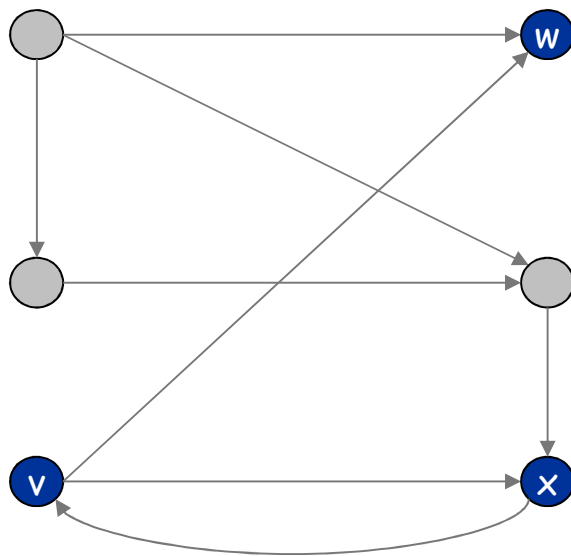
- Set  $P$  of possible projects. Project  $v$  has associated revenue  $p_v$ .
  - some projects generate money: create e-commerce interface, design web page
  - others cost money: upgrade computers, get site license
- Set of prerequisites  $E$ . If  $(v, w) \in E$ , can't do project  $v$  unless also do project  $w$ .
- A subset of projects  $A \subseteq P$  is **feasible** if the prerequisite of every project in  $A$  also belongs to  $A$ .

**Project selection.** Choose a feasible subset of projects to maximize revenue.

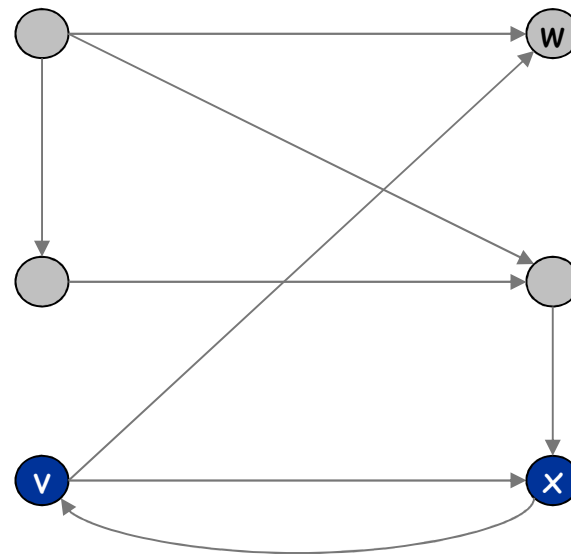
## Project Selection: Prerequisite Graph

### Prerequisite graph.

- Include an edge from  $v$  to  $w$  if can't do  $v$  without also doing  $w$ .
- $\{v, w, x\}$  is feasible subset of projects.
- $\{v, x\}$  is infeasible subset of projects.



feasible

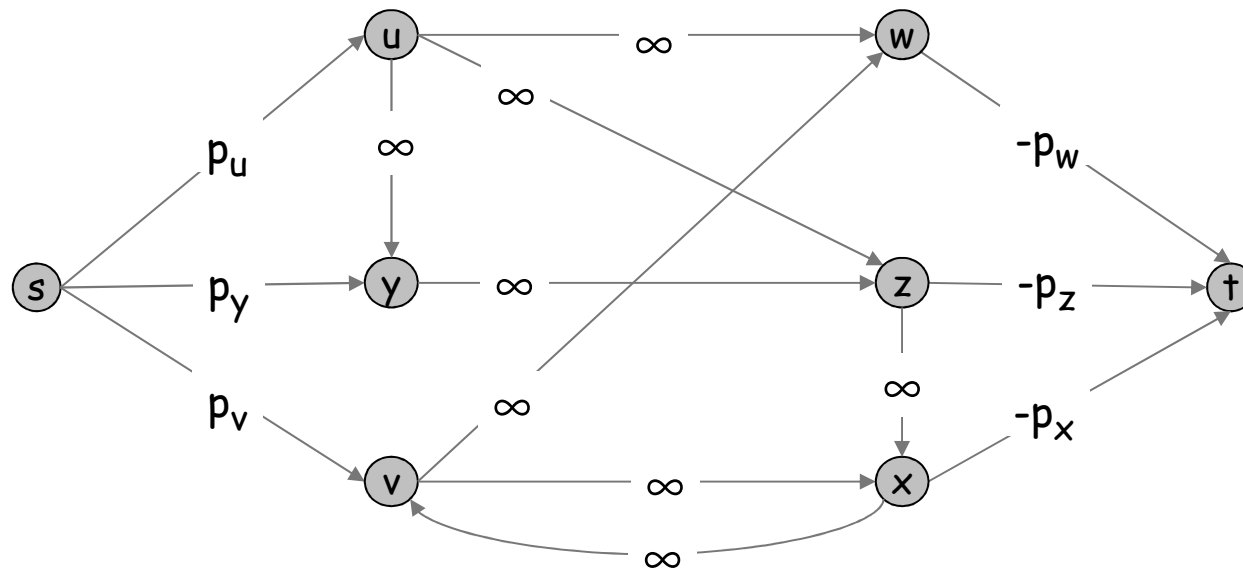


infeasible

## Project Selection: Min Cut Formulation

### Min cut formulation.

- Assign capacity  $\infty$  to all prerequisite edges.
- Add edge  $(s, v)$  with capacity  $p_v$  if  $p_v > 0$ .
- Add edge  $(v, t)$  with capacity  $-p_v$  if  $p_v < 0$ .
- For notational convenience, define  $p_s = p_t = 0$ .



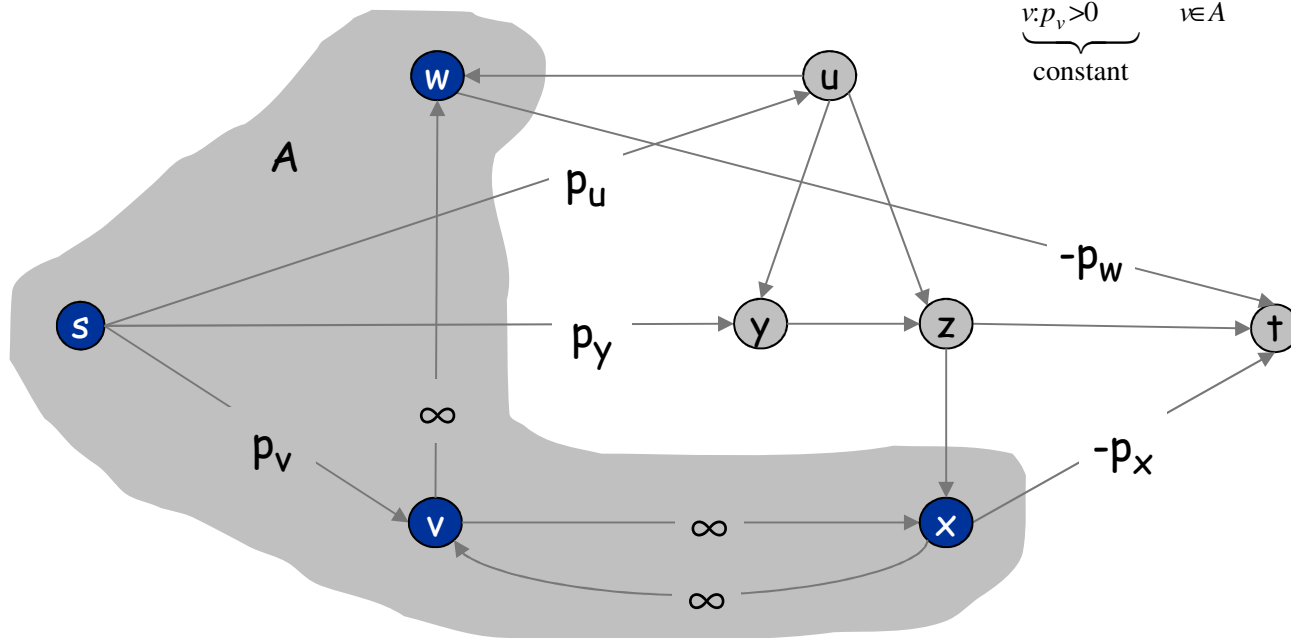
## Project Selection: Min Cut Formulation

**Claim.**  $(A, B)$  is min cut iff  $A - \{s\}$  is optimal set of projects.

- Infinite capacity edges ensure  $A - \{s\}$  is feasible.

- Max revenue because:

$$\begin{aligned} \text{cap}(A, B) &= \sum_{v \in B: p_v > 0} p_v + \sum_{v \in A: p_v < 0} (-p_v) \\ &= \sum_{v \in B: p_v > 0} p_v + \sum_{v \in A: p_v > 0} p_v - \sum_{v \in A: p_v > 0} p_v + \sum_{v \in A: p_v < 0} (-p_v) \\ &= \underbrace{\sum_{v: p_v > 0} p_v}_{\text{constant}} - \sum_{v \in A} p_v \end{aligned}$$



# Open Pit Mining

Open-pit mining. (studied since early 1960s)

- Blocks of earth are extracted from surface to retrieve ore.
- Each block  $v$  has net value  $p_v = \text{value of ore} - \text{processing cost}$ .
- Can't remove block  $v$  before  $w$  or  $x$ .

