南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Algorithm Design and Analysis (H)

CS216

Prof. Shiqi Yu (于仕琪)

yusq@sustech.edu.cn

http://faculty.sustech.edu.cn/yusq/

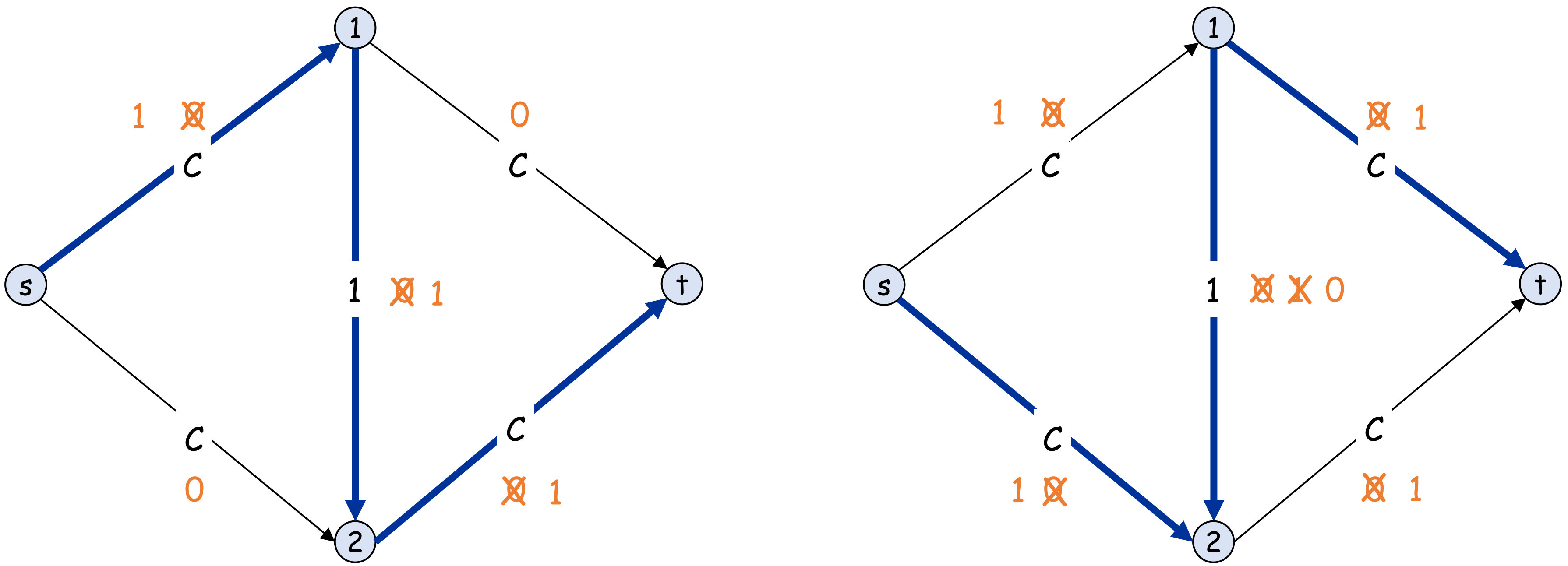
2. Choosing Good Augmenting Paths

Ford-Fulkerson: Exponential Number of Augmentations

- Q. Is generic Ford-Fulkerson algorithm polynomial in input size?

  m, n

- A. No. If max capacity is C, then algorithm can take C iterations.



Choosing Good Augmenting Paths

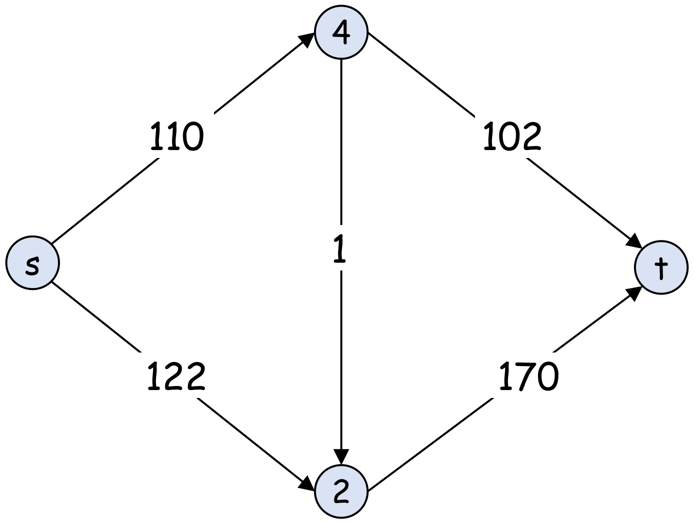
- Use care when selecting augmenting paths.
  - Some choices lead to exponential algorithms.
  - Clever choices lead to polynomial algorithms.
  - If capacities are irrational, algorithm not guaranteed to terminate!

- Goal: choose augmenting paths so that:
  - Can find augmenting paths efficiently.
  - Few iterations.

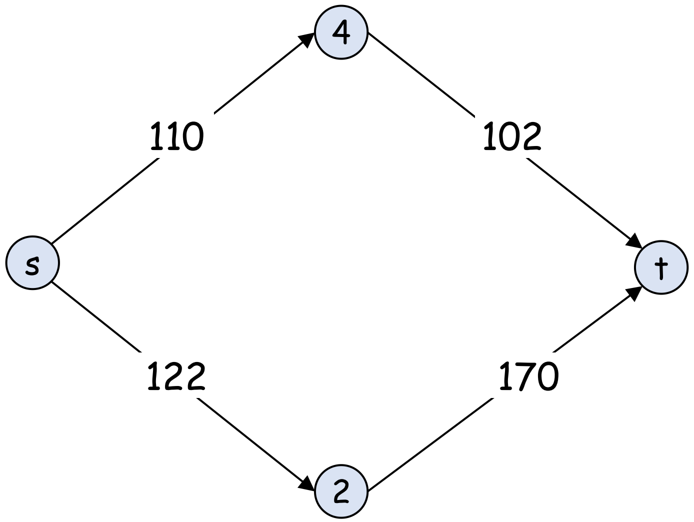
- Choose augmenting paths with: [Edmonds-Karp 1972, Dinitz 1970]
  - Max bottleneck capacity.
  - Sufficiently large bottleneck capacity.
  - Fewest number of edges.

Capacity Scaling

- Intuition. Choosing path with highest bottleneck capacity increases flow by max possible amount.
  - Don't worry about finding exact highest bottleneck path.
  - Maintain scaling parameter Δ.
  - Let $G_f(\Delta)$ be the subgraph of the residual graph consisting of only arcs with capacity at least Δ.



G_f



G_f (100)

Capacity Scaling

```
Scaling-Max-Flow(G, s, t, c) {
   foreach e ∈ E  f(e) ← 0
   Δ ← smallest power of 2 greater than or equal to C
   G_f ← residual graph

   while (Δ ≥ 1) {
      G_f(Δ) ← Δ-residual graph
      while (there exists augmenting path P in G_f(Δ)) {
         f ← augment(f, c, P)
         update G_f(Δ)
      }
      Δ ← Δ / 2
   }
   return f
}
```

Capacity Scaling: Correctness

- **Assumption.** All edge capacities are integers between 1 and C.

- **Integrality invariant.** All flow and residual capacity values are integral.

- **Correctness.** If the algorithm terminates, then f is a max flow.
- **Pf.**
  - By integrality invariant, when $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$.
  - Upon termination of $\Delta = 1$ phase, there are no augmenting paths. ▪

Capacity Scaling: Running Time

- Lemma 1. The outer while loop repeats $1 + \lceil \log_2 C \rceil$ times.
- Pf. Initially $C \le \Delta < 2C$. Δ decreases by a factor of 2 each iteration. ▪

- Lemma 2. Let f be the flow at the end of a Δ-scaling phase. Then the value of the maximum flow is at most $v(f) + m\,\Delta$. ← proof on next slide

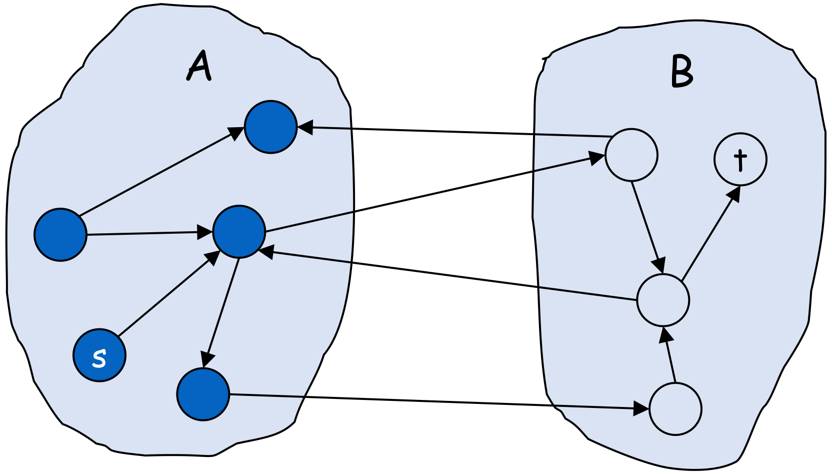
- Lemma 3. There are at most 2m augmentations per scaling phase.
  - Let f be the flow at the end of the previous scaling phase Δ'
  - Let f* be the maximum flow.
  - Lemma 2 $\Rightarrow$ $v(f^*) \le v(f) + m\,\Delta' = v(f) + m(2\Delta)$.
  - Each augmentation in a Δ-phase increases v(f) by at least Δ. ▪

- Theorem. The scaling max-flow algorithm finds a max flow in $O(m \log C)$ augmentations. It can be implemented to run in $O(m^2 \log C)$ time. ▪

Capacity Scaling: Running Time

- Lemma 2. Let f be the flow at the end of a Δ-scaling phase. Then value of the maximum flow is at most $v(f) + m\,\Delta$.
- Pf. (almost identical to proof of max-flow min-cut theorem)
  - We show that at the end of a Δ-phase, there exists a cut (A, B) such that $c(A, B) \leq v(f) + m\,\Delta$.
  - Choose A to be the set of nodes reachable from s in $G_f(\Delta)$.
  - By definition of A, $s \in A$.
  - By definition of f, $t \notin A$.

  - Let B be the set of all vertices not in A, $t \in B$, so (A, B) is an s-t cut.
  - Consider two cases: e is an edge in G
    $e = (u, v)$, $u \in A$ and $v \in B$
    then $f(e) > c_e - \Delta$
    $e = (v, u)$, $u \in B$ and $v \in A$
    then $f(e) < \Delta$



original network

Capacity Scaling: Running Time

- Lemma 2. Let f be the flow at the end of a Δ-scaling phase. Then value of the maximum flow is at most $v(f) + m\,\Delta$.
- Pf. (almost identical to proof of max-flow min-cut theorem)
  - We show that at the end of a Δ-phase, there exists a cut (A, B) such that $c(A, B) \leq v(f) + m\,\Delta$.
  - Choose A to be the set of nodes reachable from s in $G_f(\Delta)$.
  - By definition of A, $s \in A$.
  - By definition of f, $t \notin A$.

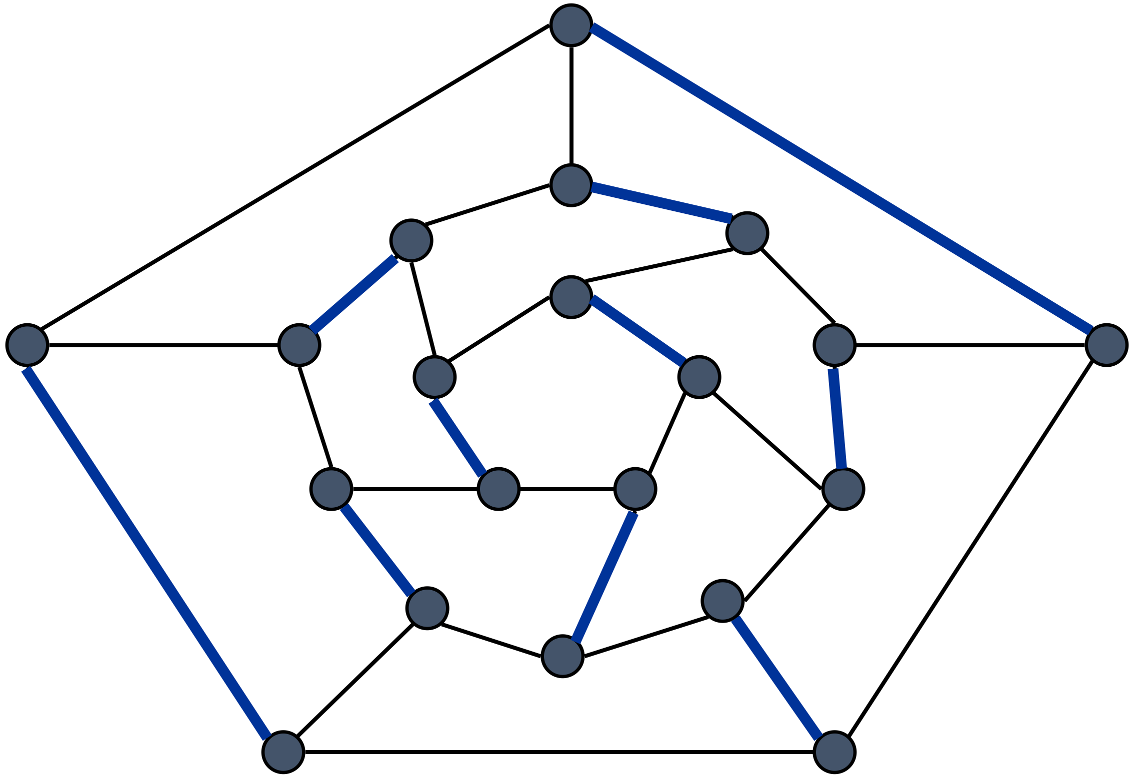
- Let B be the set of all vertices not in A, $t \in B$, so (A, B) is an s-t cut.
- Consider two cases: e is an edge in G
  - $e = (u, v)$, $u \in A$ and $v \in B$ then $f(e) > c_e - \Delta$
  - $e = (v, u)$, $u \in B$ and $v \in A$ then $f(e) < \Delta$

$$
\begin{aligned}
v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \\
&\geq \sum_{e \text{ out of } A} (c_e - \Delta) - \sum_{e \text{ into } A} \Delta \\
&= \sum_{e \text{ out of } A} c_e - \sum_{e \text{ out of } A} \Delta - \sum_{e \text{ into } A} \Delta \\
&\geq c(A, B) - m\Delta.
\end{aligned}
$$

3. Bipartite Matching

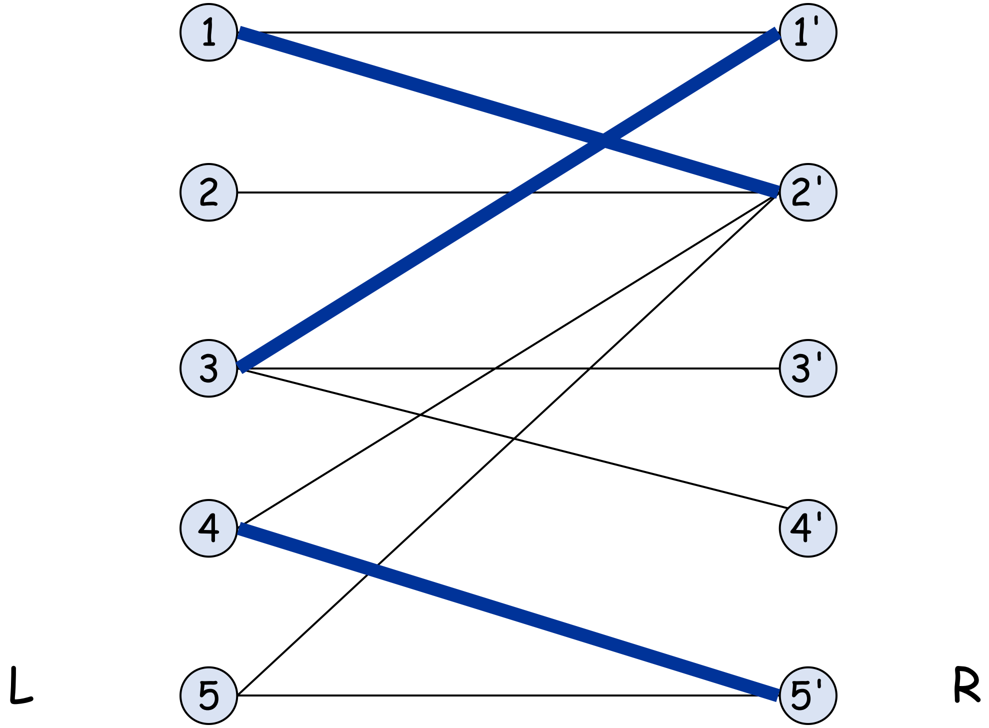
Matching

- 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.

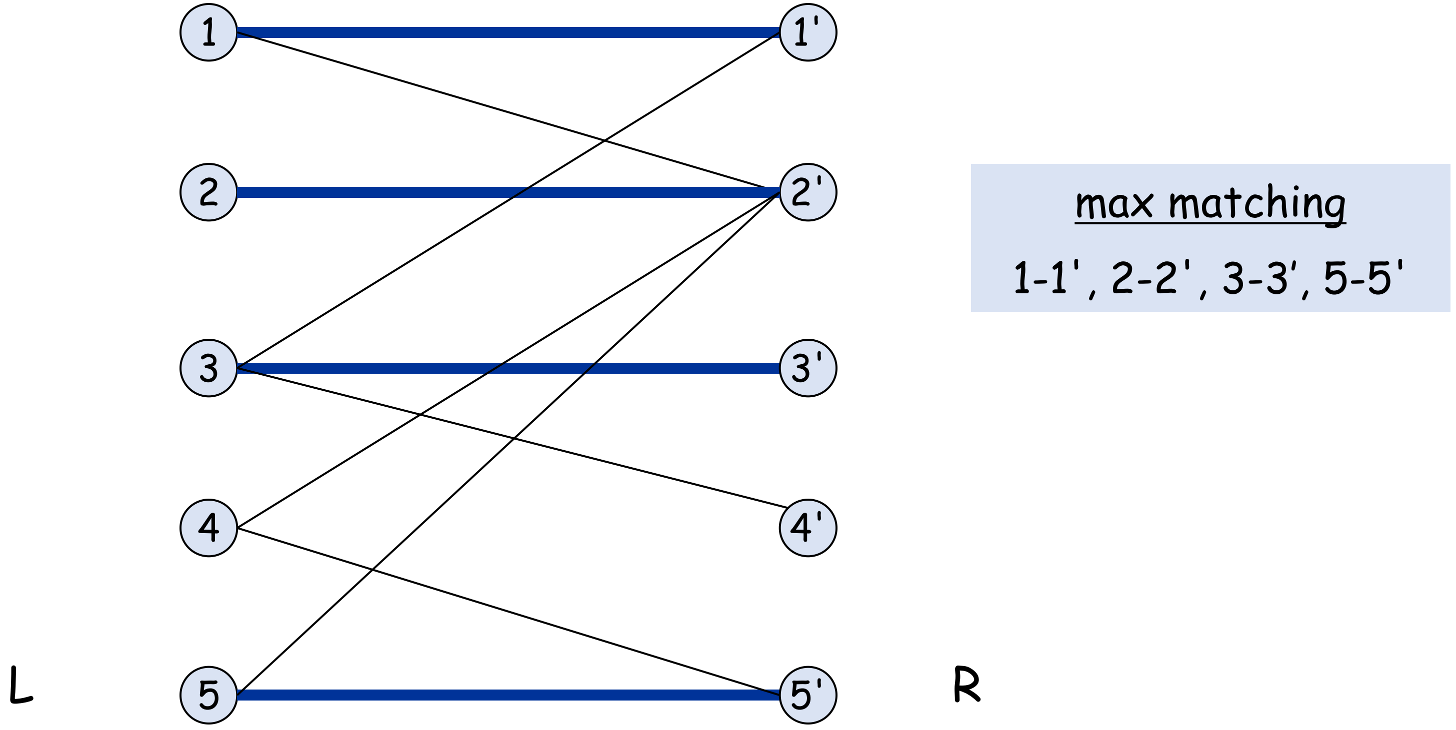


matching

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

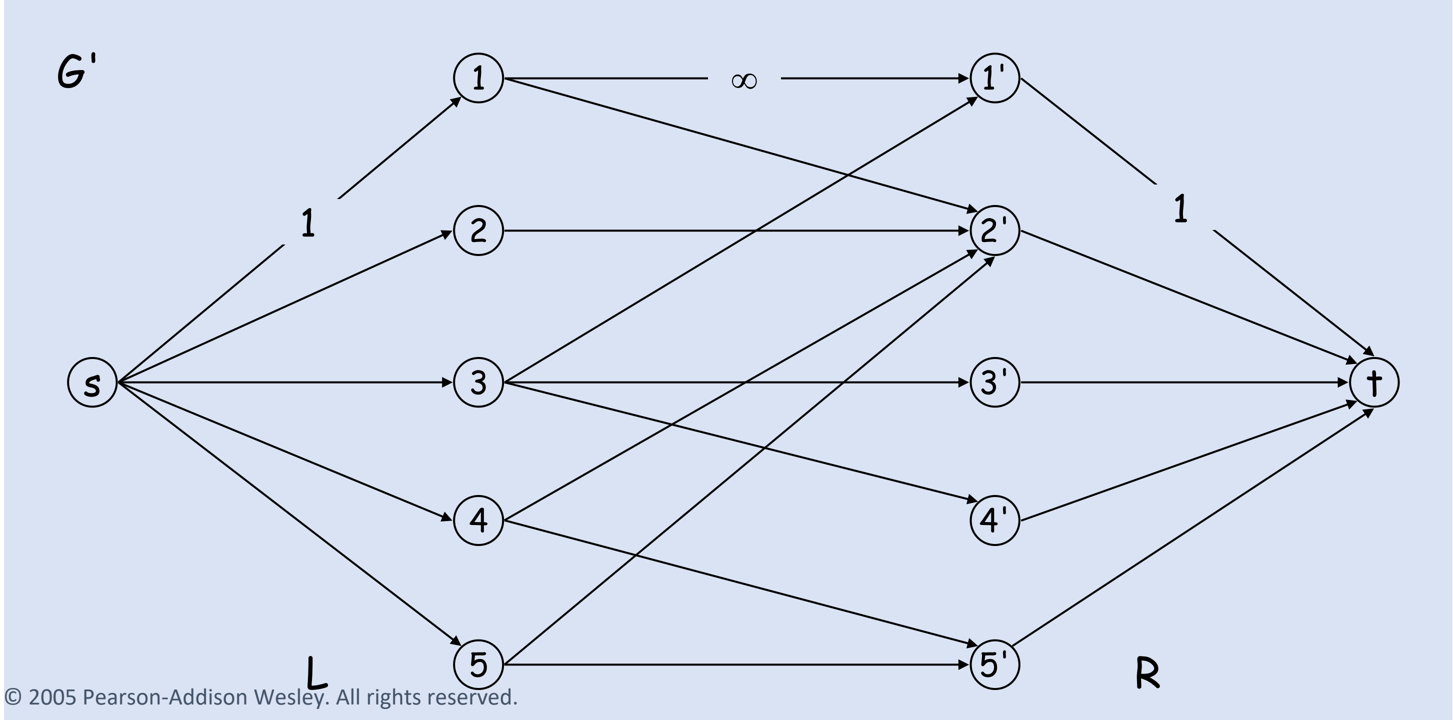
Bipartite Matching

- Bipartite matching.
  - Input: undirected, bipartite graph G = (L ∪ R, E).
  - M ⊆ 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 ∪ R ∪ {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.

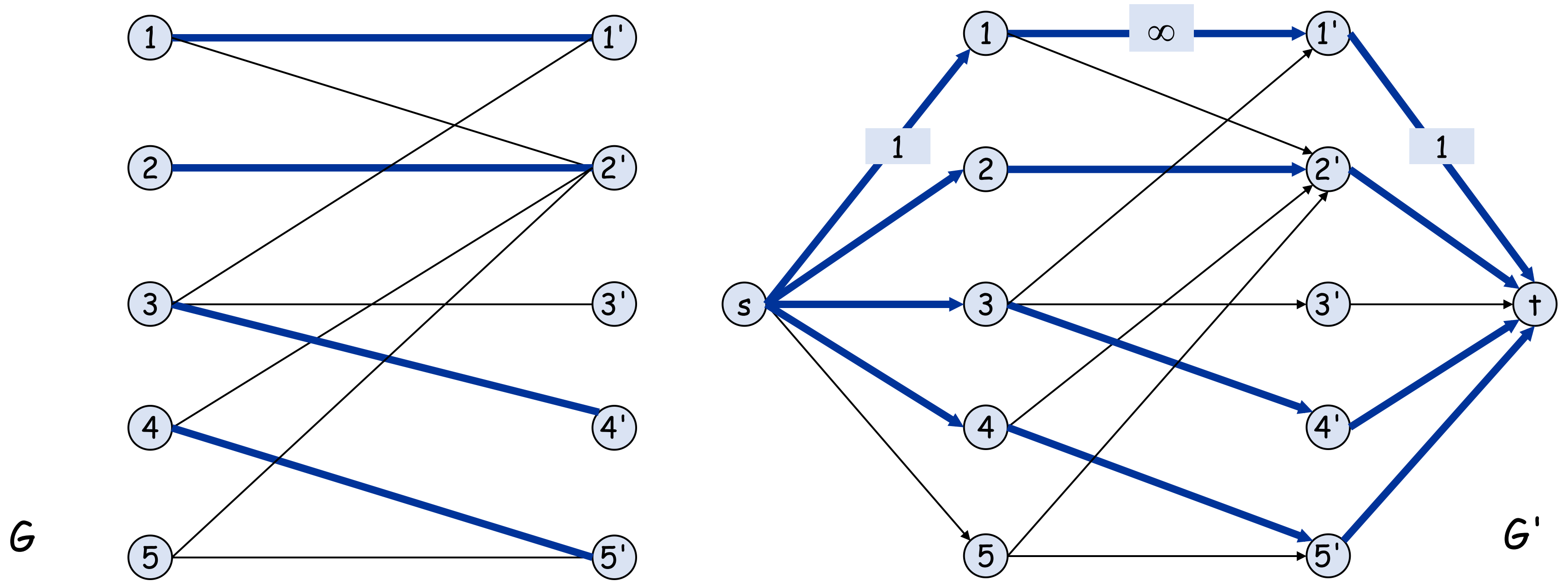


Bipartite Matching: Proof of Correctness

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. ▪

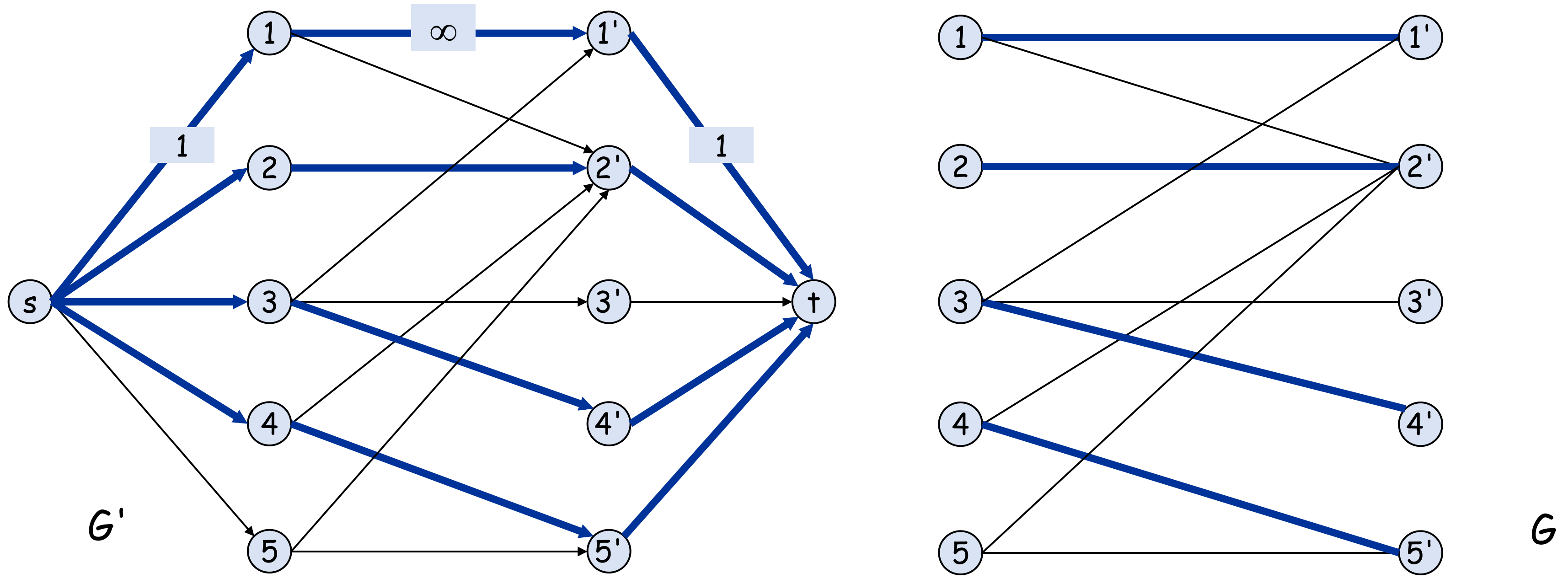


Bipartite Matching: Proof of Correctness

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

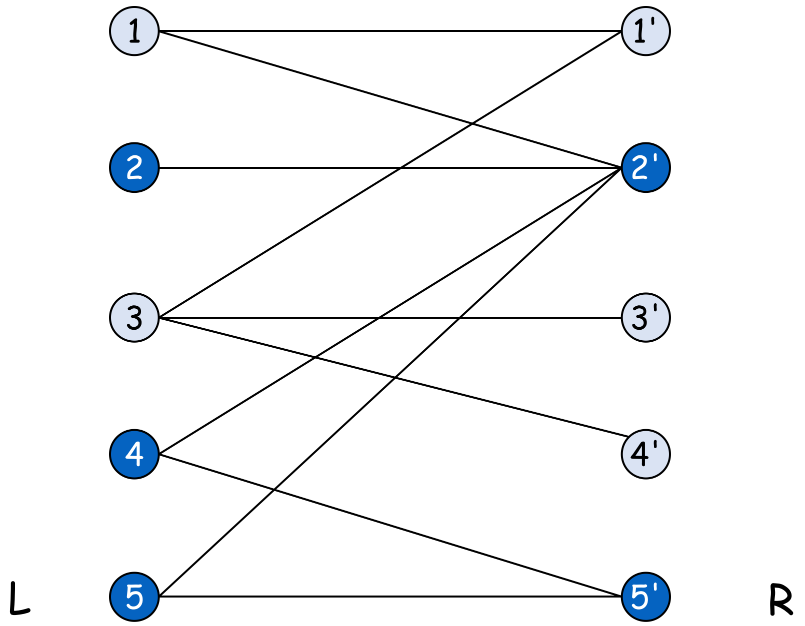
- Def. A matching $M \subseteq E$ is perfect if each node appears in exactly one edge in M.

- 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).

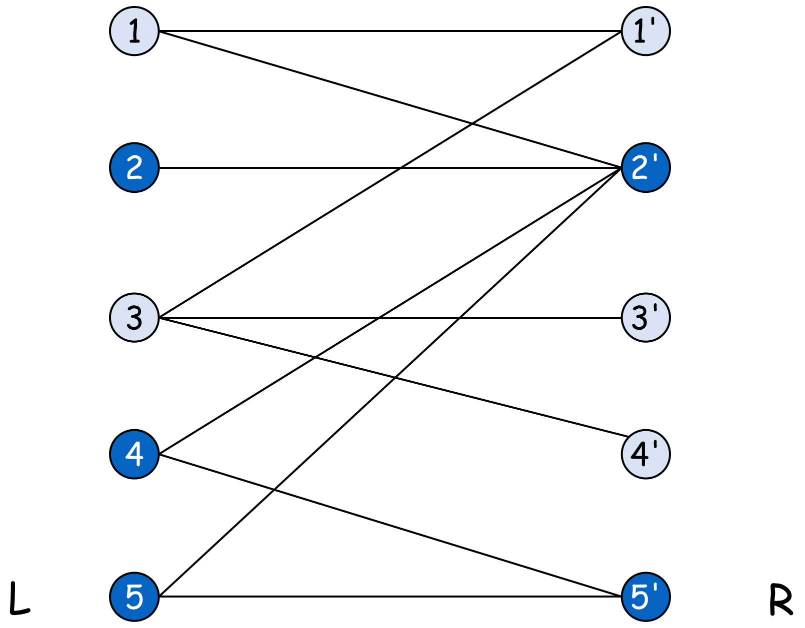


No perfect matching:
S = { 2, 4, 5 }
N(S) = { 2', 5' }.

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.

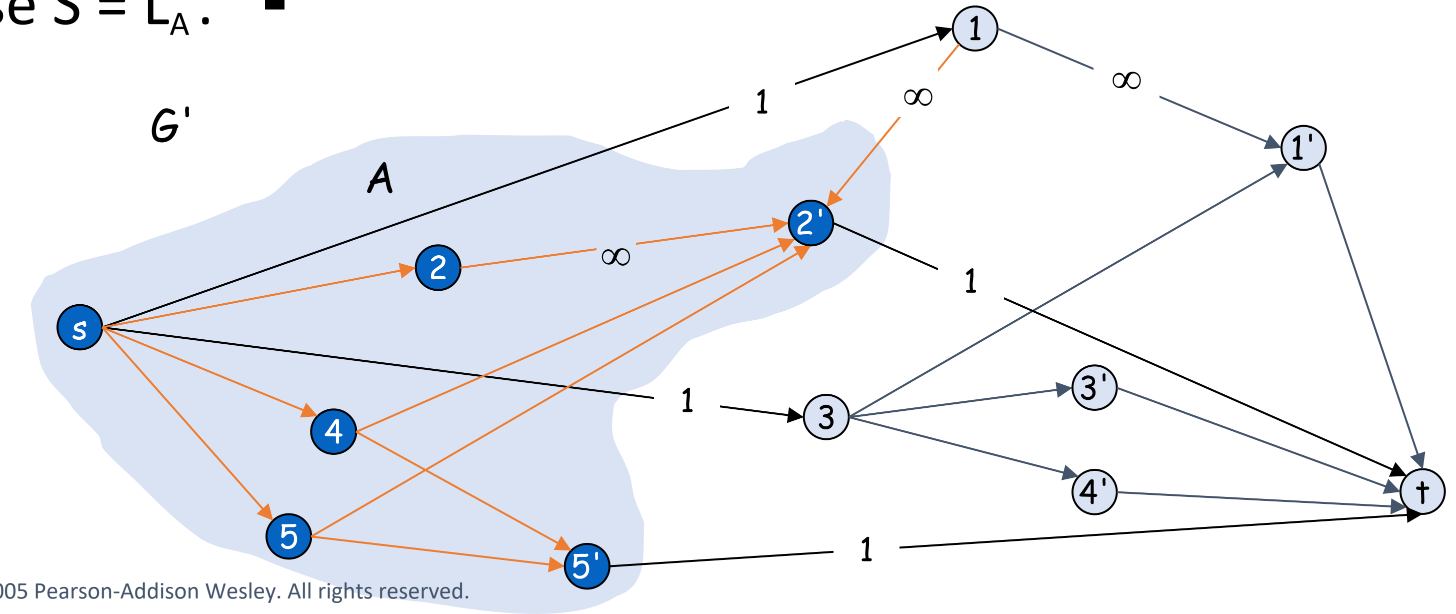


No perfect matching:
S = { 2, 4, 5 }
N(S) = { 2', 5' }.

Proof of Marriage Theorem

- 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'.
  - By max-flow min-cut, $c(A, B) < | L |$.
  - Define $L_A = L \cap A$, $L_B = L \cap B$, $R_A = R \cap A$.
  - $c(A, B) = | L_B | + | R_A |$.
  - Min cut : $N(L_A) \subseteq R_A$ because c(A,B) not increased if so
  - $|N(L_A)| \leq | R_A | = c(A, B) - | L_B | < | L | - | L_B | = | L_A |$.
  - Choose $S = L_A$. ■

$G' = (L \cup R \cup \{s, t\}, E')$



$L_A = \{2, 4, 5\}$
$L_B = \{1, 3\}$
$R_A = \{2', 5'\}$
$N(L_A) = \{2', 5'\}$

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]

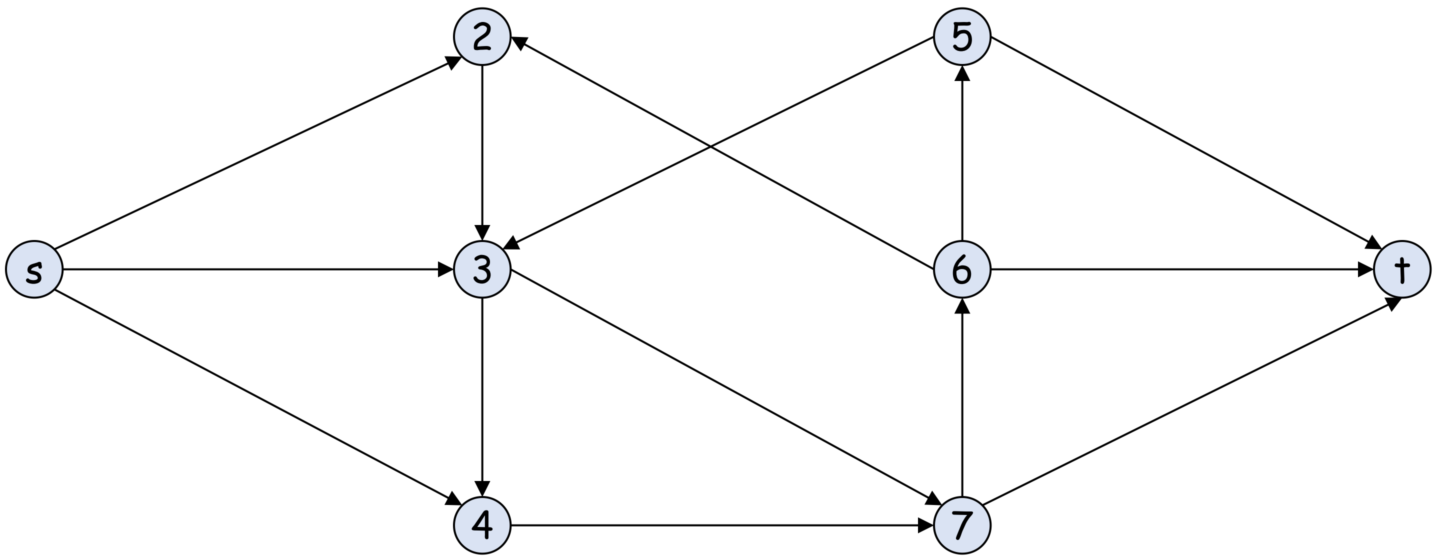
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

4. 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.

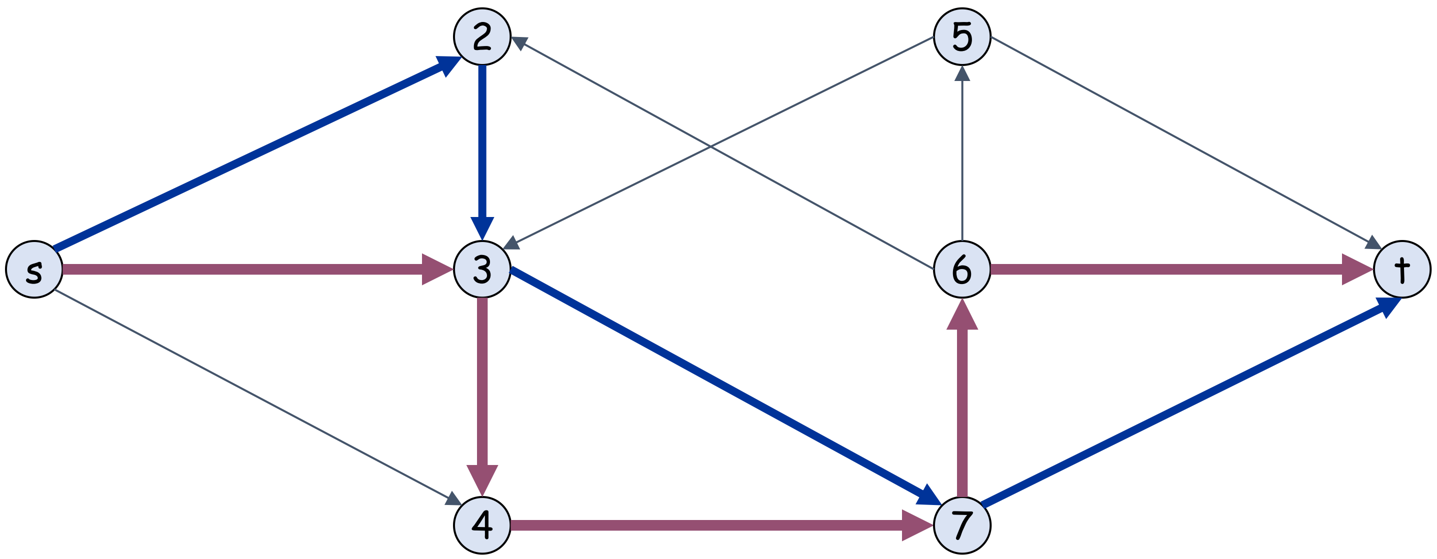
- Def. Two paths are edge-disjoint if they have no edge in common.



Edge Disjoint Paths

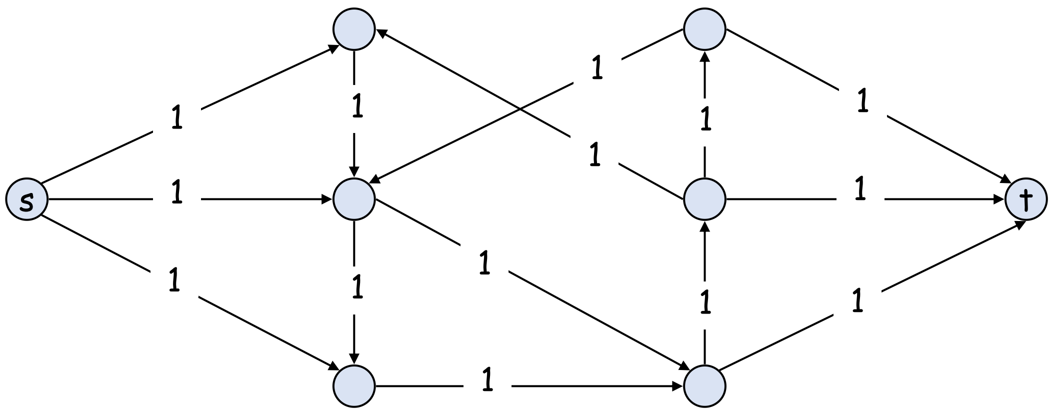
- Disjoint path problem. Given a digraph G = (V, E) and two nodes s and t, find the max number of edge-disjoint s-t paths.

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



Edge Disjoint Paths

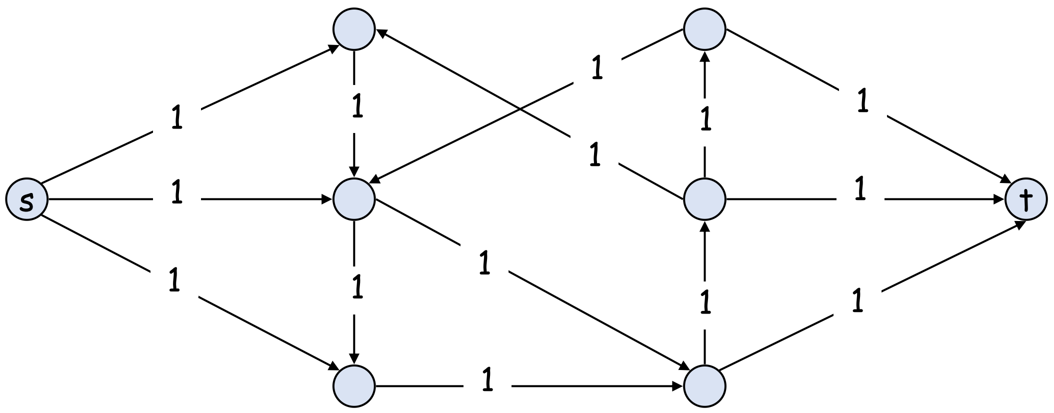
- Max flow formulation: assign unit capacity to every edge.



- Theorem. Max number edge-disjoint s-t paths equals max flow value.
- Pf. $\leq$
  - Suppose there are k edge-disjoint paths $P_1, \ldots, P_k$.
  - 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

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



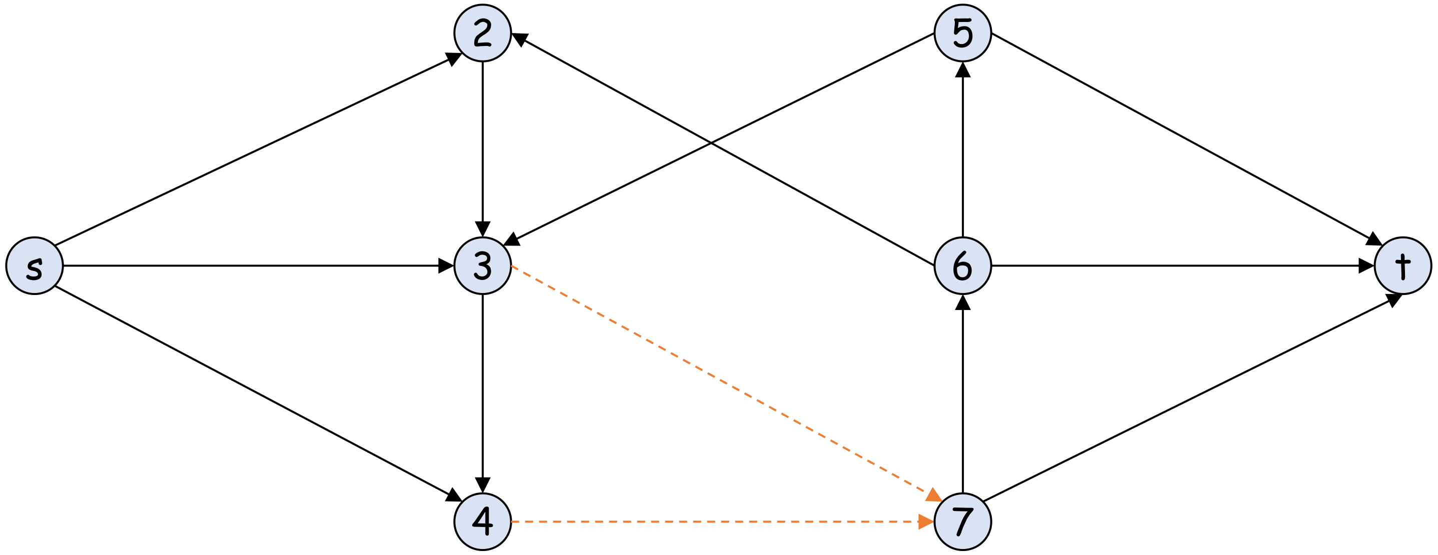
- Theorem. Max number edge-disjoint s-t paths equals max flow value.
- Pf. $\geq$
  - Suppose max flow value is k.
  - Integrality theorem $\Rightarrow$ there exists 0-1 flow f of value k.
  - Consider edge (s, u) with f(s, u) = 1.
    - ✓ by conservation, there exists an edge (u, v) with f(u, v) = 1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

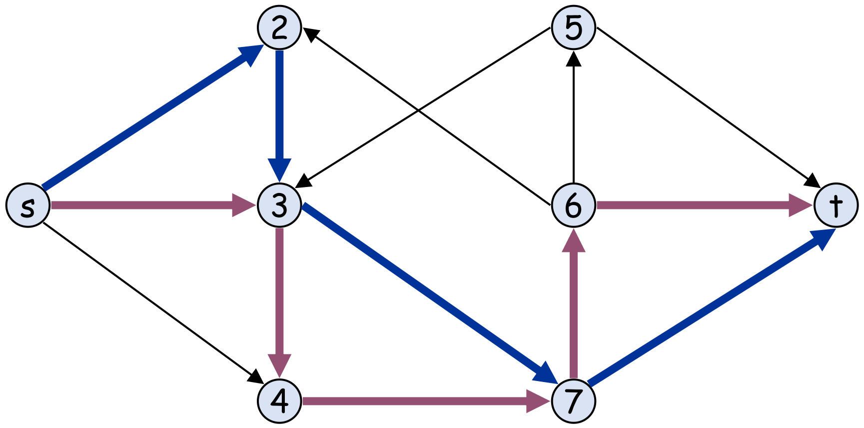
- Def. A set of edges $F \subseteq E$ disconnects t from s if every s-t path uses at least one edge in F.

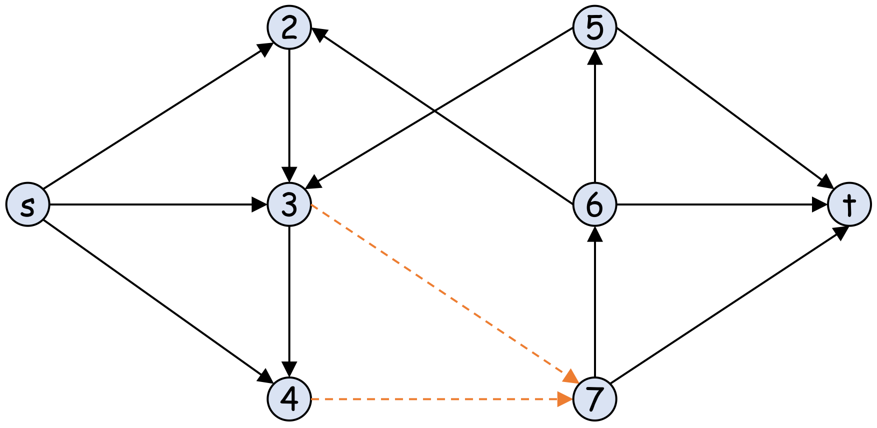


Edge Disjoint Paths and Network Connectivity

- 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 $|F| = k$.
  - Every s-t path uses at least one edge in F.
    Hence, the number of edge-disjoint paths is at most k. ▪

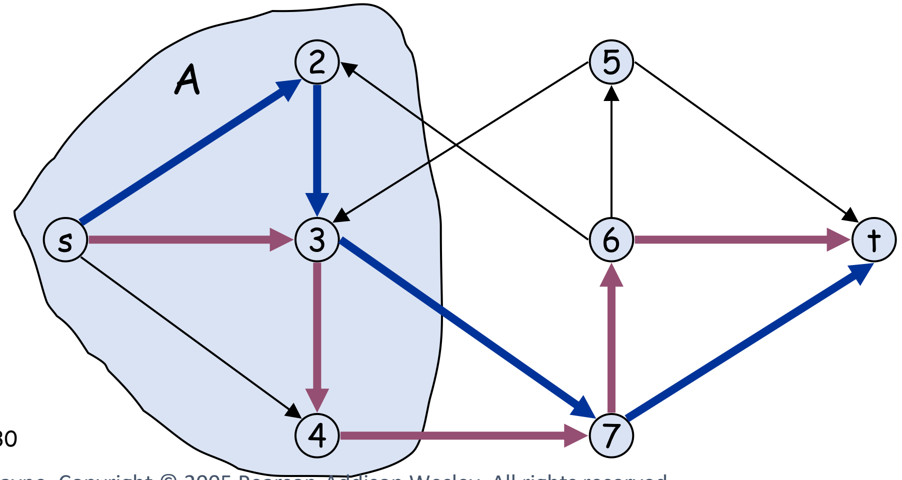


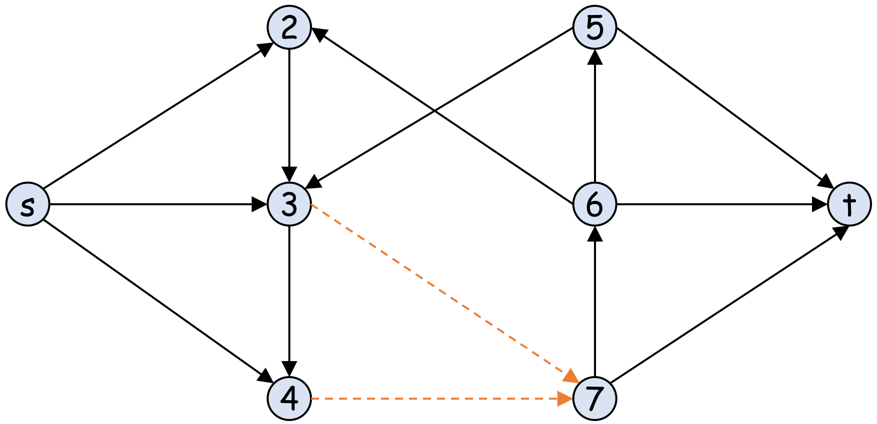


Disjoint Paths and Network Connectivity

- 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 $\Rightarrow$ exist cut (A, B) of capacity k.
  - Let F be set of edges going from A to B.
  - |F| = k and disconnects t from s. ▪





5. Extensions to Max Flow

Circulation with Demands

Circulation with demands.

- Directed graph G = (V, E).
- Edge capacities c(e), e ∈ E.
- Node supply and demands d(v), v ∈ 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 ∈ E: $0 \leq f(e) \leq c(e)$ (capacity)
- For each v ∈ V: $f^{in}(v) - f^{out}(v) = d(v)$ (conservation)

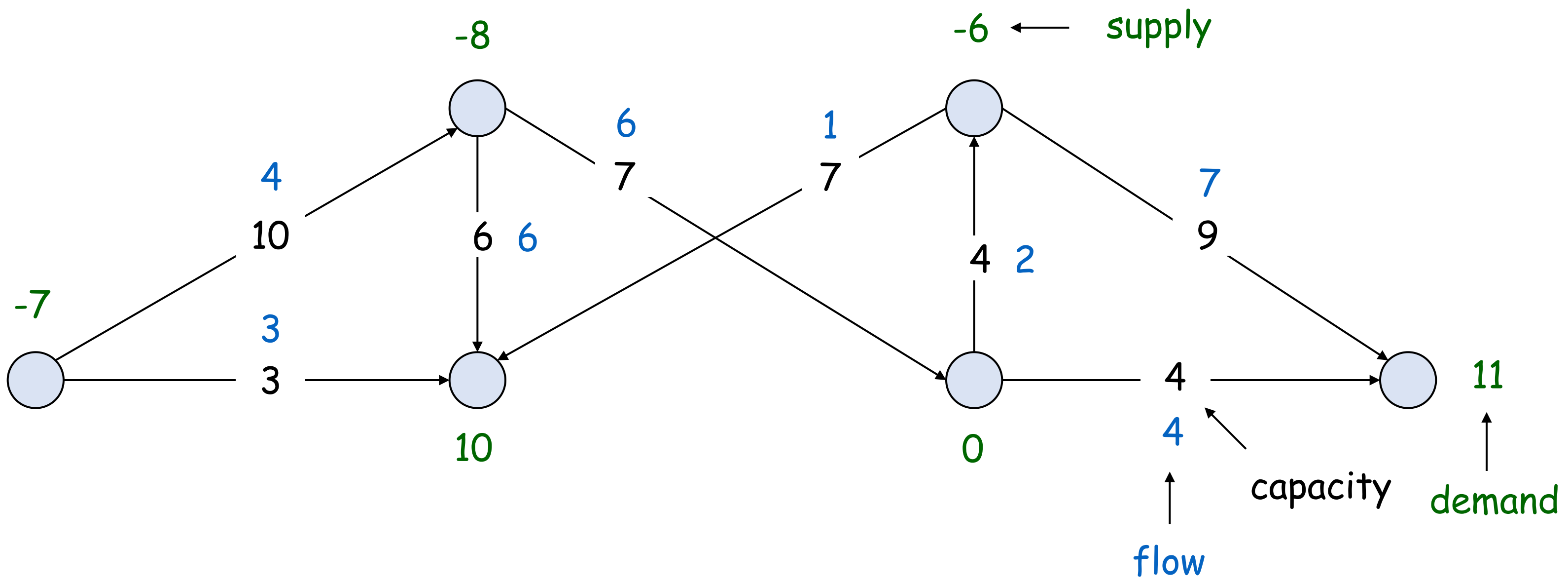
Circulation problem: given (V, E, c, d), does there exist a circulation?

Circulation with Demands

Necessary condition: sum of supplies = sum of demands.

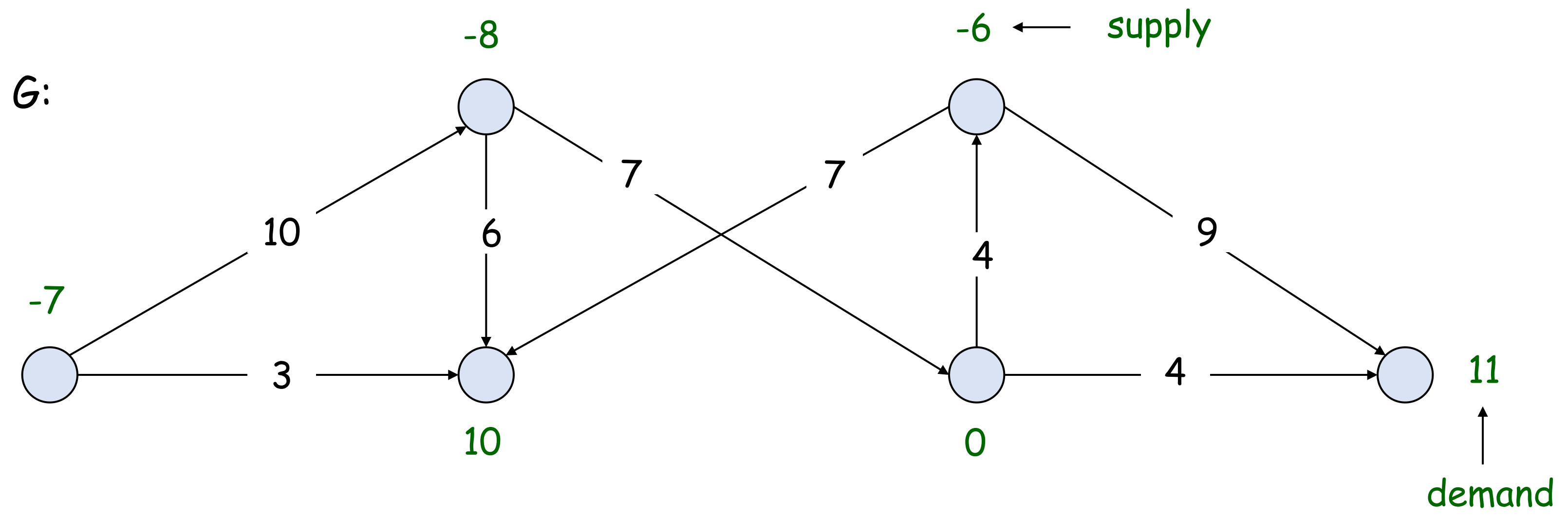
$$\sum_v d(v) = 0$$

Pf. Sum conservation constraints for every demand node v.



Circulation with Demands

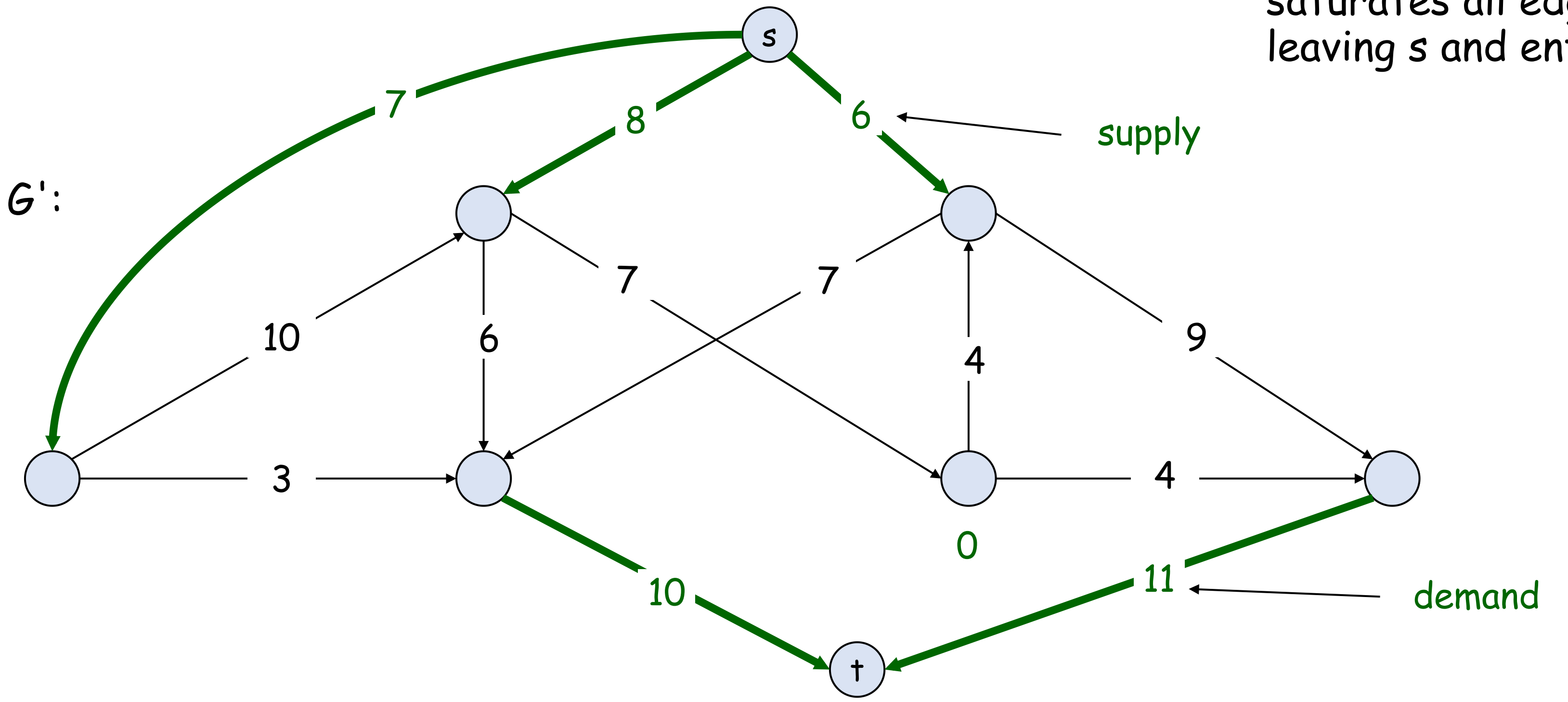
- Max flow formulation.



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.

saturates all edges leaving s and entering t



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

- 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 $v \in V$: $f^{in}(v) - f^{out}(v) = d(v)$ (conservation)

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

6. Survey Design

Survey Design

- Survey design.
  - Design survey asking n_1 consumers about n_2 products.

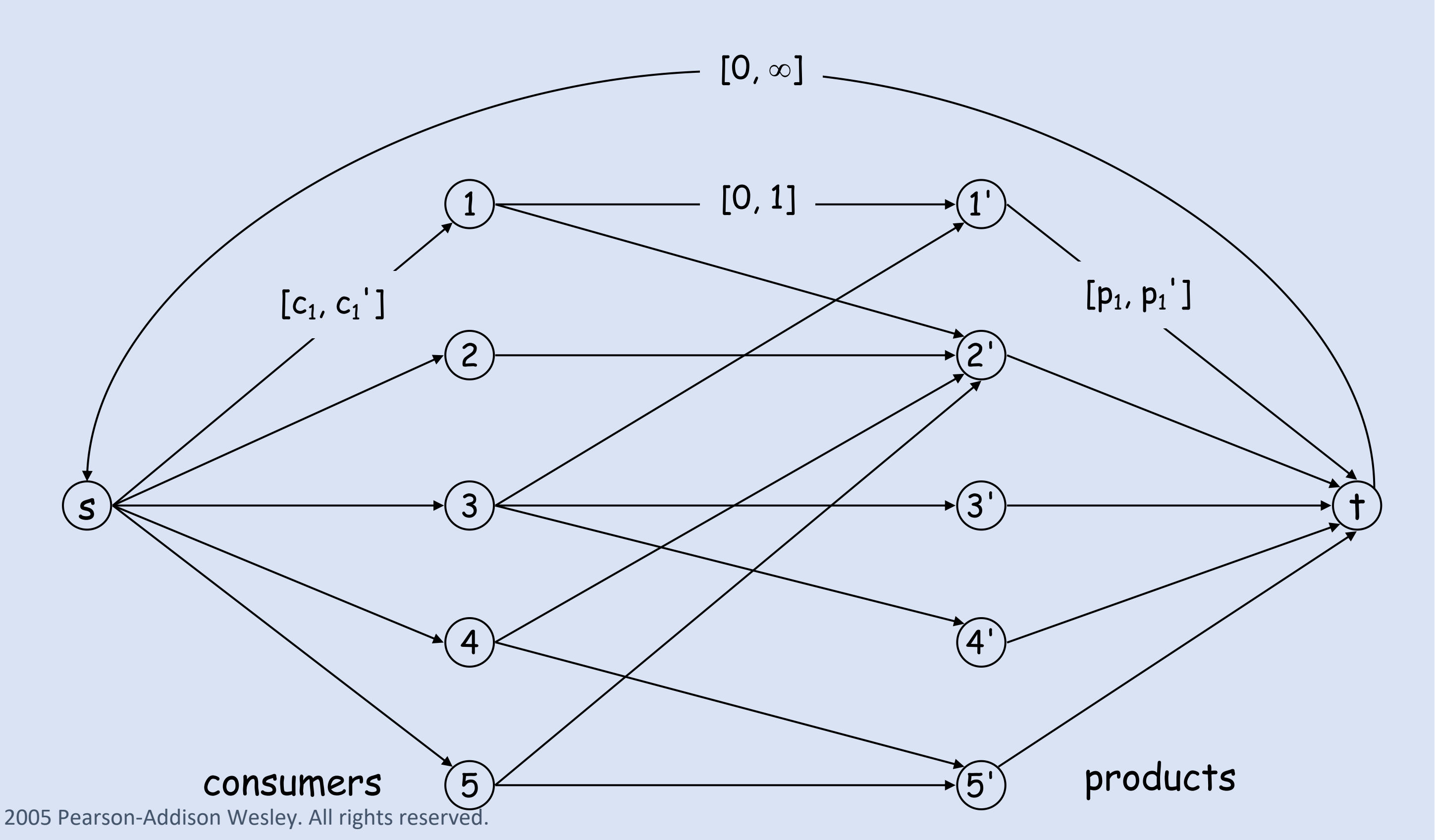
    one survey question per product
  - Can only survey consumer i about product j if they own it.
  - Ask consumer i between c_i and c_i' questions.
  - Ask between p_j and p_j' consumers about product j.

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

- Bipartite perfect matching. Special case when $c_i = c_i' = p_i = p_i' = 1$.

Survey Design

- Algorithm. Formulate as a circulation problem with lower bounds.
  - Include an edge (i, j) if consumer j owns product i.
  - Integer circulation $\Leftrightarrow$ feasible survey design.



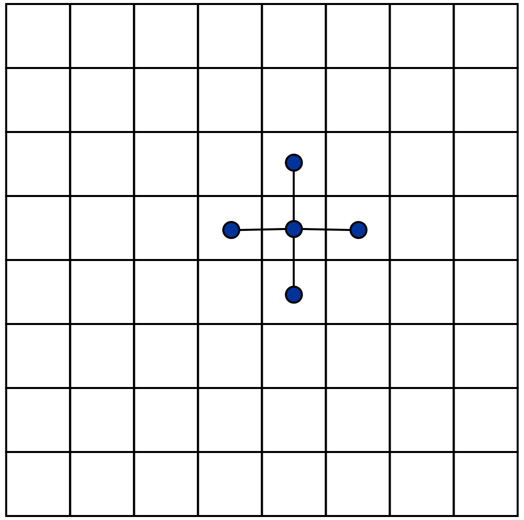
7. Image Segmentation

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.

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

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

Image Segmentation

Formulate as min cut problem.

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

Turn into minimization problem.

- Maximizing: $q(A,B) = Q - \sum_{i \in A} b_i - \sum_{j \in B} a_j - \sum_{(i,j) \in E, |A \cap \{i,j\}|=1} p_{ij}$

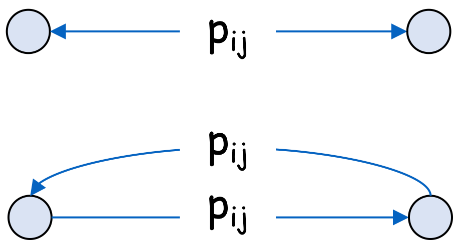
  where $Q = \sum_i (a_i + b_i)$

is equivalent to minimizing:

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

- or alternatively

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.

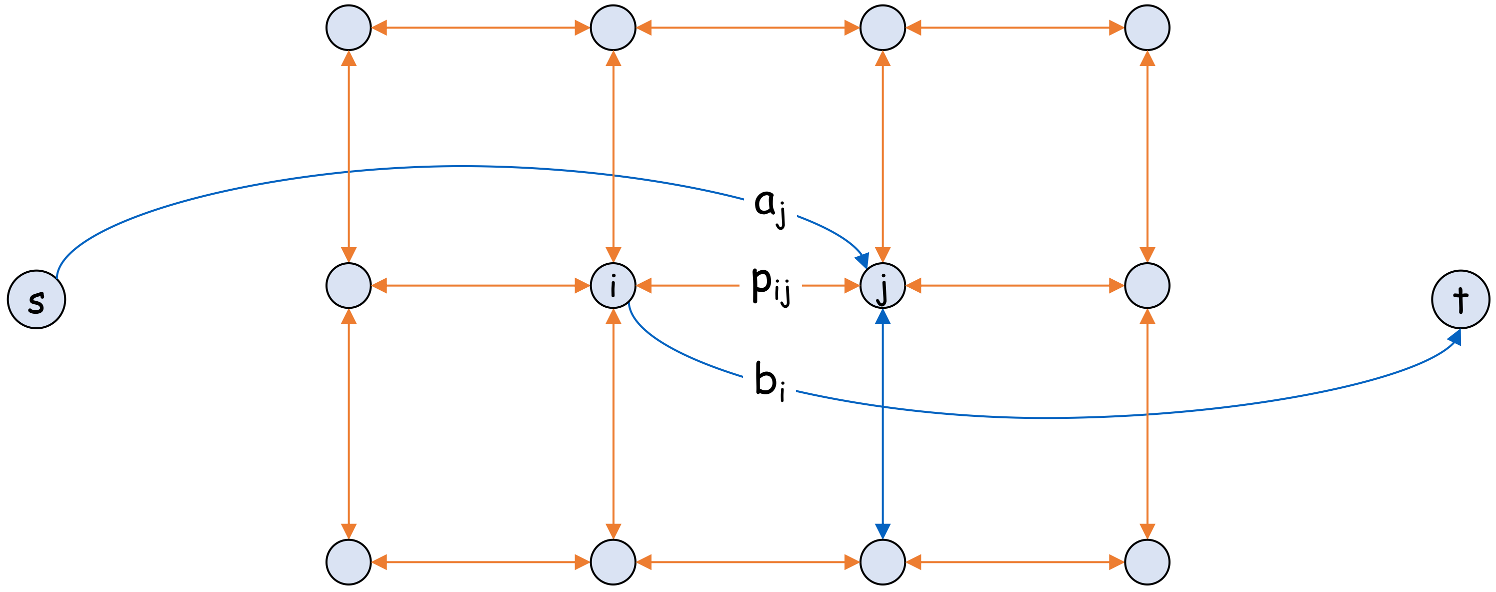


Image Segmentation

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

- A = foreground.
- $c(A,B) = \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{(i,j) \in E, |A \cap \{i,j\}|=1} p_{ij} = q'(A,B)$
- Precisely the quantity we want to minimize.

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