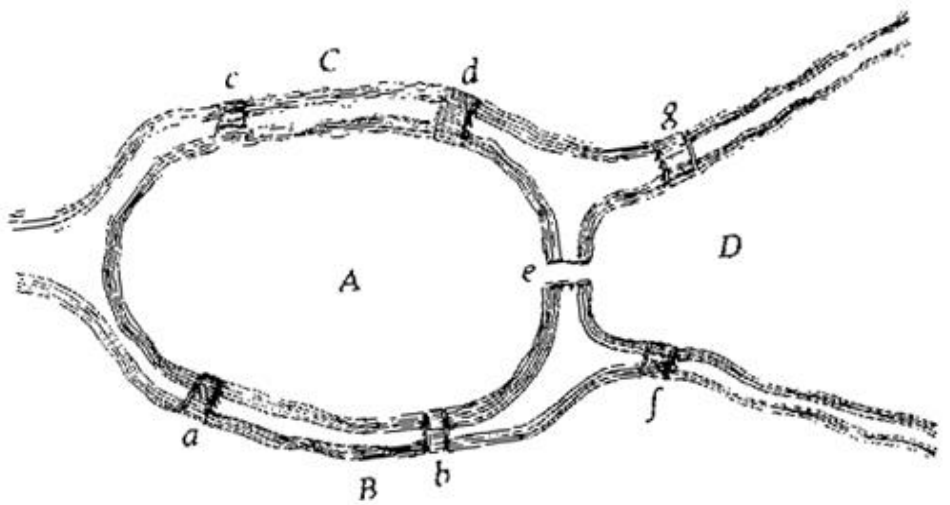
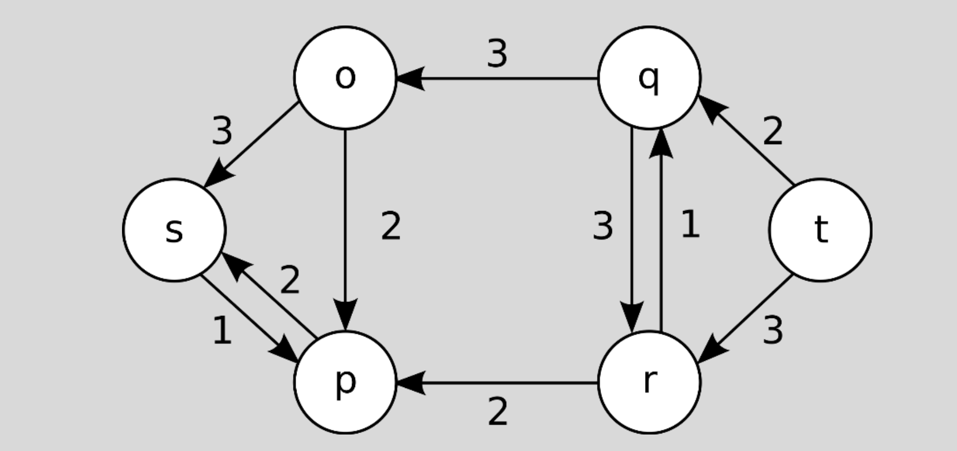
Lecture 9 – Flow networks II: Applications

THE UNIVERSITY OF SYDNEY





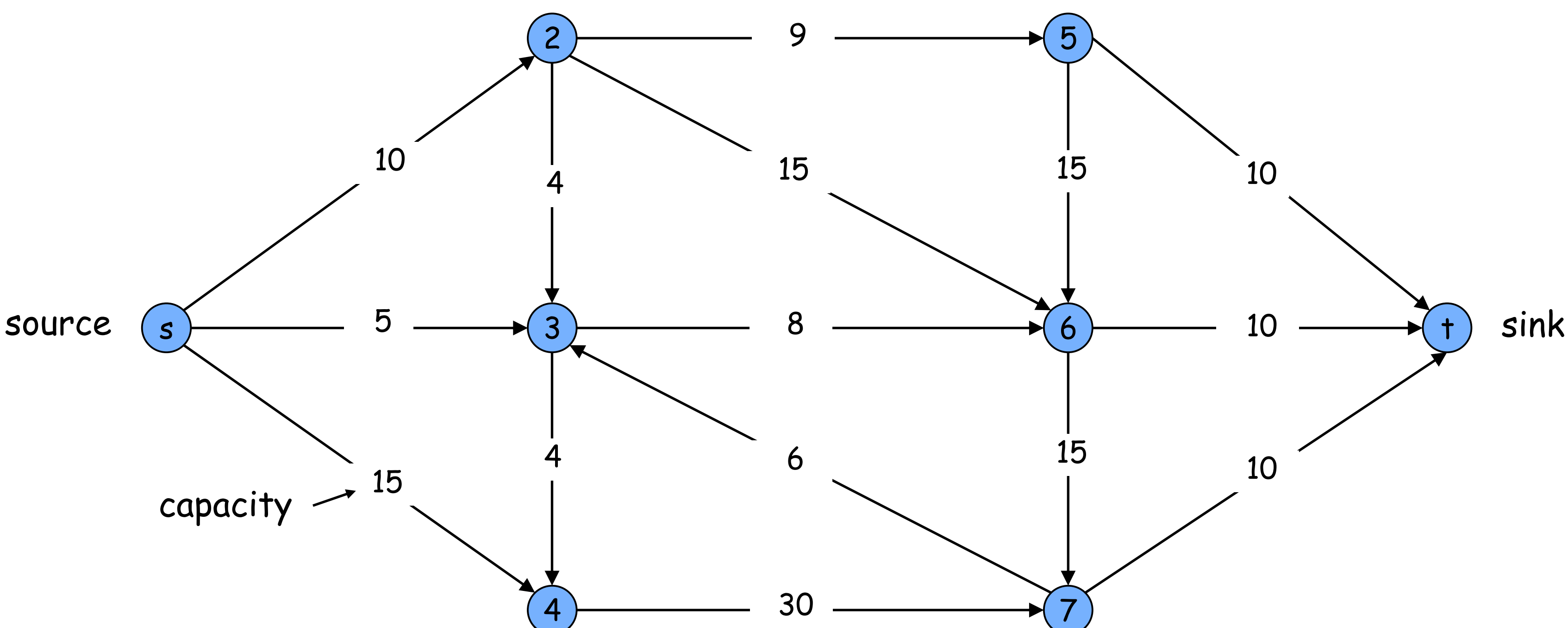
General techniques in this course

- Greedy algorithms [Lecture 3]
- Divide & Conquer algorithms [Lecture 4]
- Sweepline algorithms [Lecture 5]
- Dynamic programming algorithms [Lectures 6 and 7]
- Network flow algorithms [Lecture 8 and today]
  - Theory [Lecture 8]
  - Applications [today]

- Next lecture: NP and intractability

Recap: Minimum Cut Problem

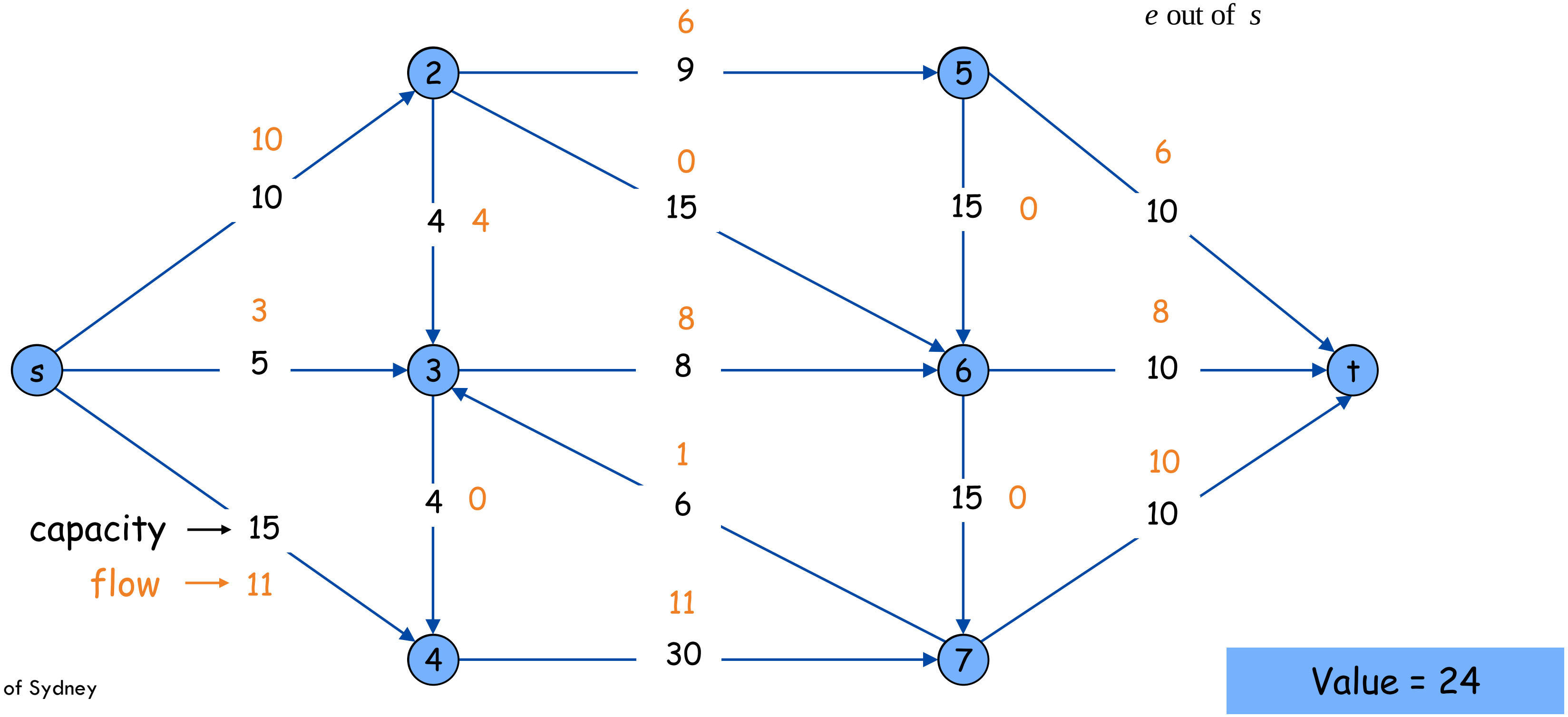
- Flow network
  - Abstraction for material flowing through the edges.
  - G = (V, E) = directed graph, no parallel edges.
  - Two distinguished nodes: s = source, t = sink.
  - c(e) = capacity of edge e.



Recap: Flows

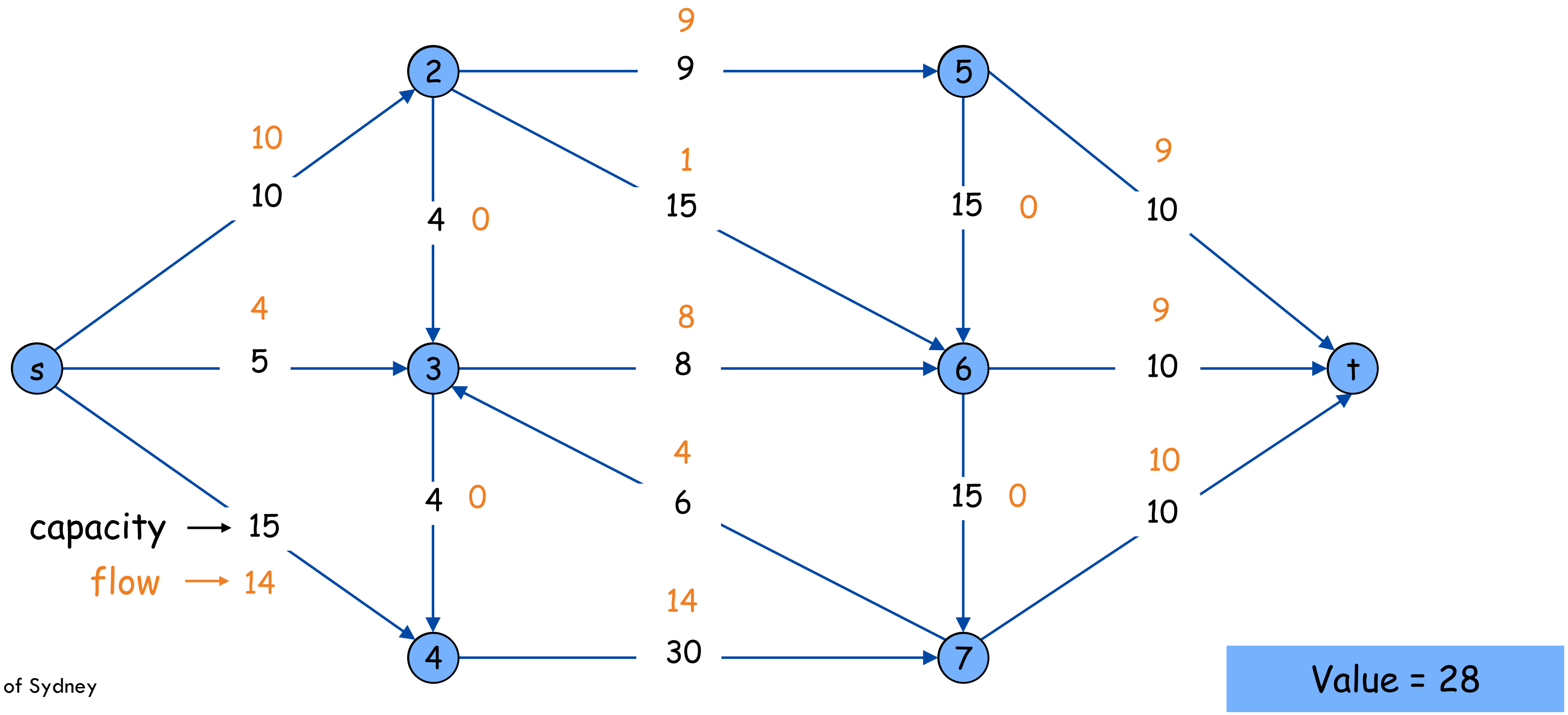
– **Definition:** An s-t flow is a function that satisfies:
  – For each e ∈ E: $0 \le f(e) \le c(e)$ (capacity)
  – For each v ∈ V – {s, t}: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

– **Definition:** The value of a flow f is: $v(f) = \sum_{e \text{ out of } s} f(e)$.



Recap: Maximum Flow Problem

– Max flow problem. Find s-t flow of maximum value.

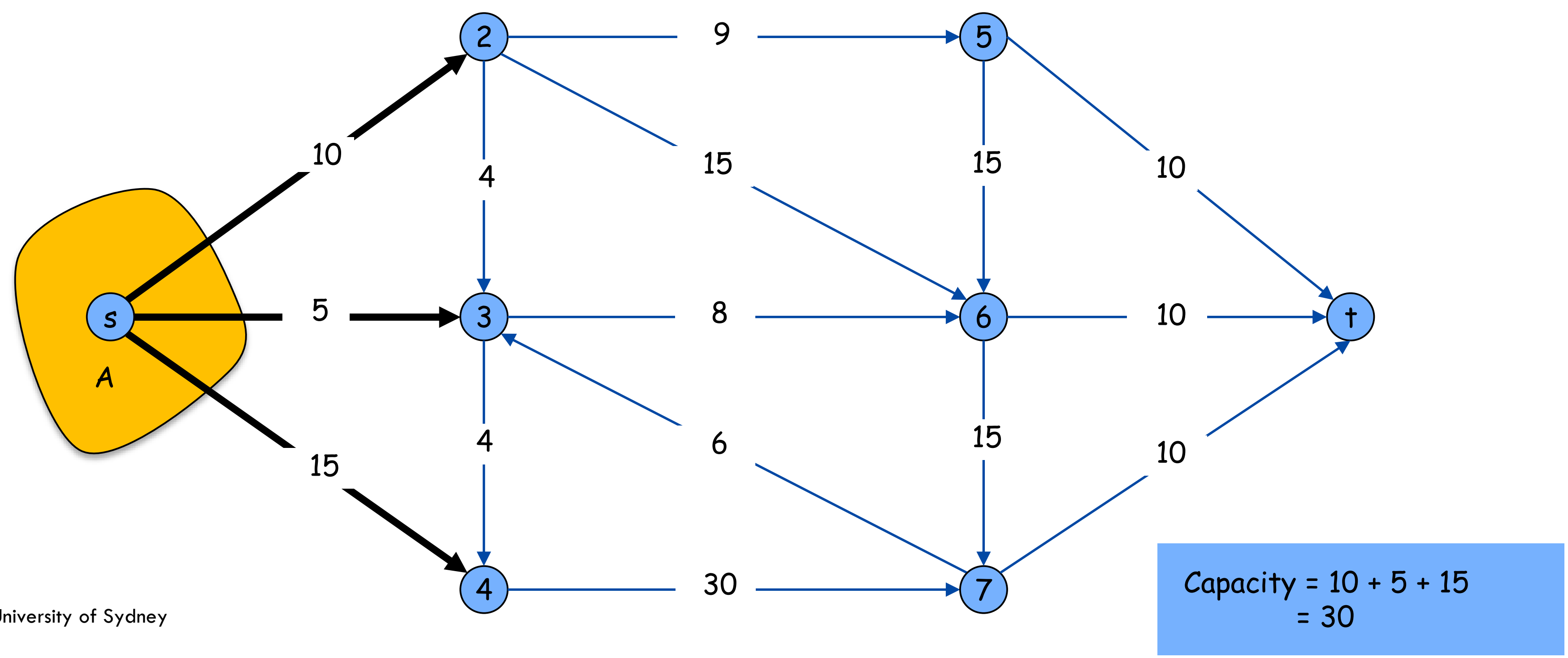


Recap: Cuts

Definitions:

– An s-t cut is a partition (A, B) of V with s ∈ A and t ∈ B.

– The capacity of a cut (A, B) is: $cap(A, B) = \sum_{e \text{ out of } A} c(e)$

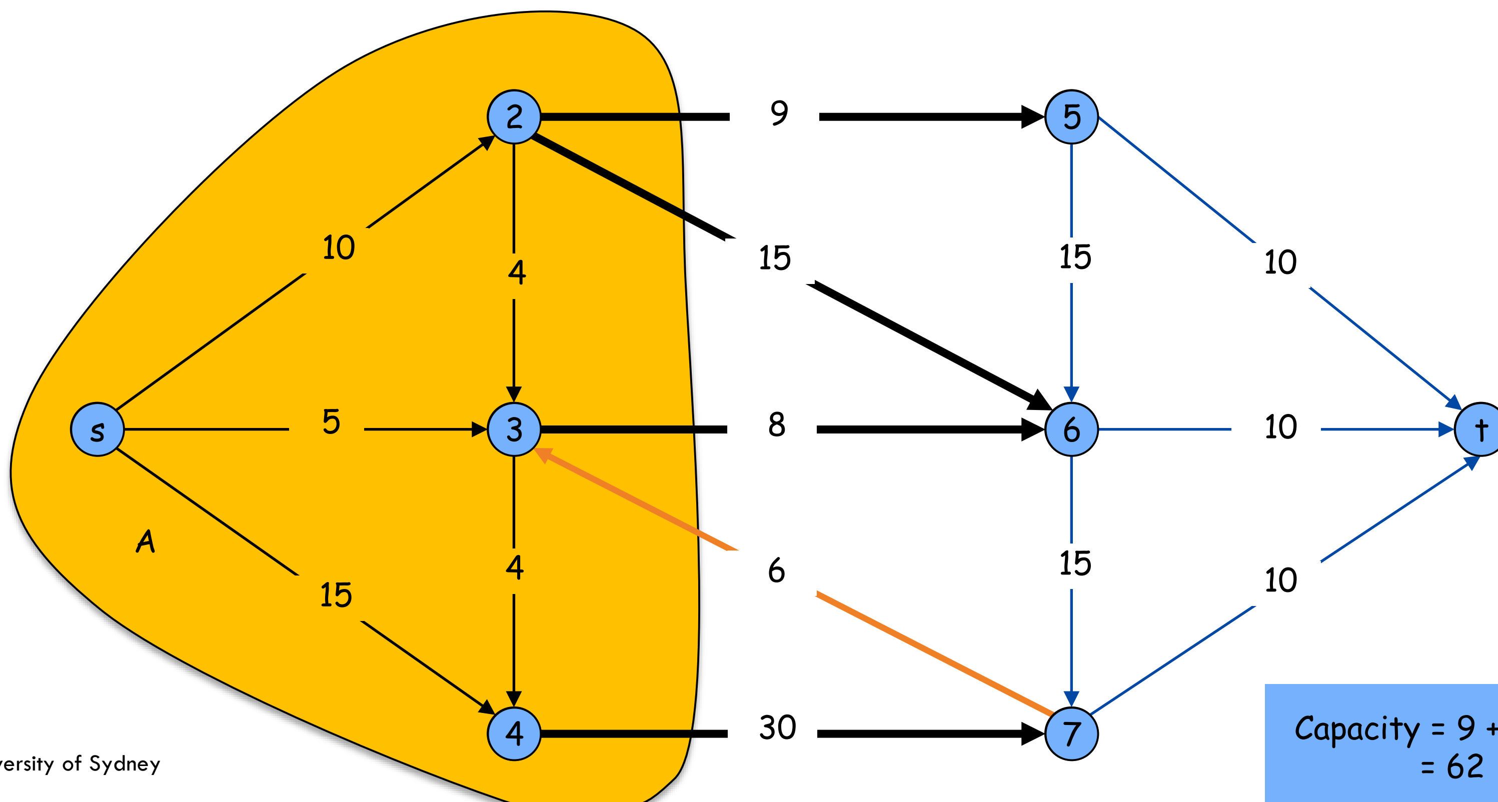


Recap: Cuts

Definitions:

– An s-t cut is a partition (A, B) of V with s ∈ A and t ∈ B.

– The capacity of a cut (A, B) is: $cap(A, B) = \sum_{e \text{ out of } A} c(e)$

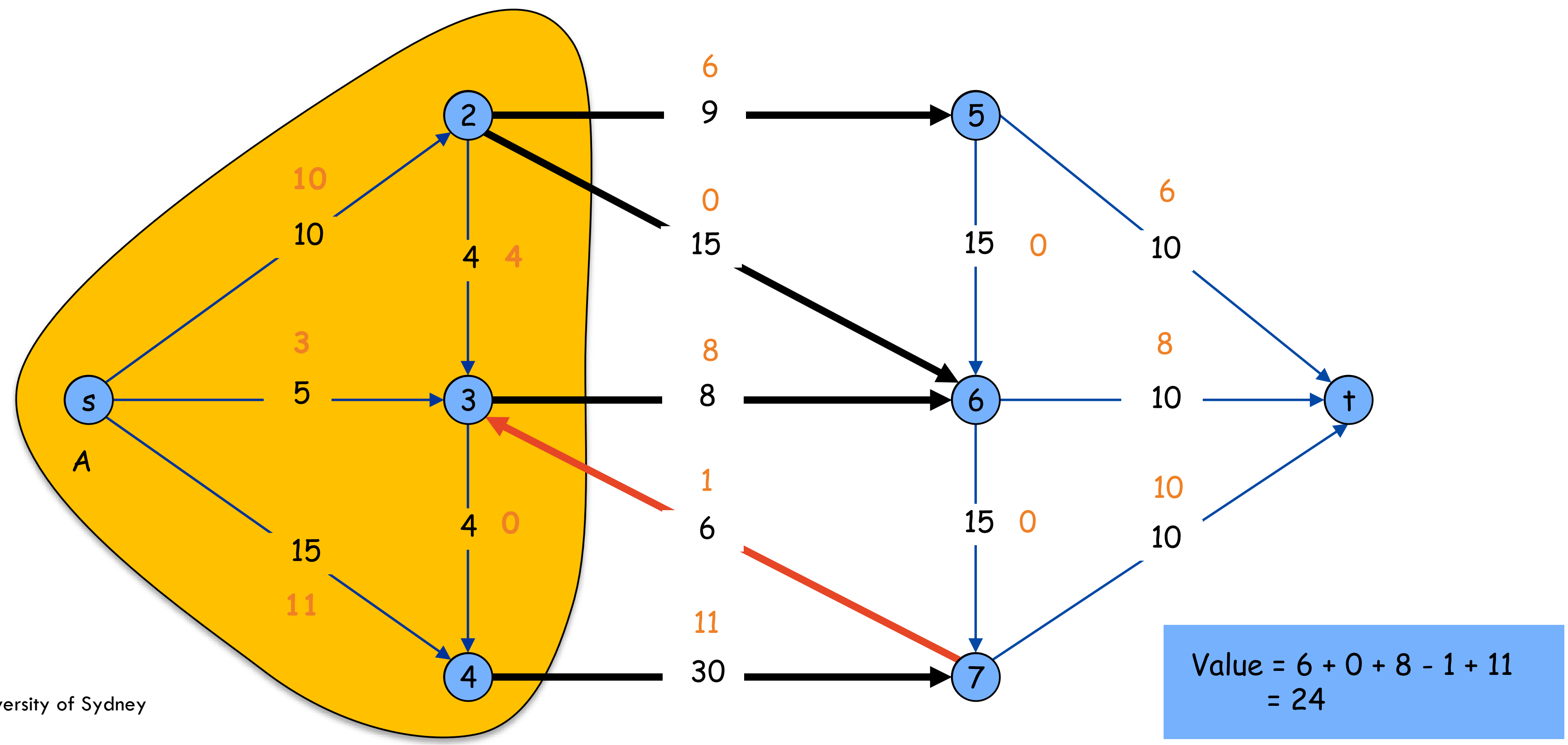


Capacity = 9 + 15 + 8 + 30
= 62

Recap: Flows and Cuts

– Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then, the net flow sent across the cut is equal to the amount leaving s.

$$\sum_{e \text{ out of } A} f(e) \; - \; \sum_{e \text{ in to } A} f(e) \; = \; v(f)$$



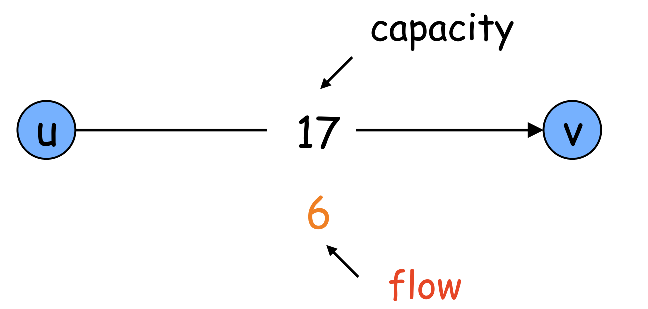
Augmenting Path Algorithm

```
Ford-Fulkerson(G,s,t) {
   foreach e ∈ E
        f(e) ← 0
   Gf ← residual graph

   while (there exists augmenting path P in Gf){
      f ← Augment(f,P)
      update Gf
   }
   return f
}
```

```
Augment(f,P) {
   b ← bottleneck(P,f)
   foreach e =(u,v) ∈ P {
      if e is a forward edge then
         increase f(e) in G by b
      else (e is a backward edge)
         decrease f(e) in G by b
   }
   return f
}
```

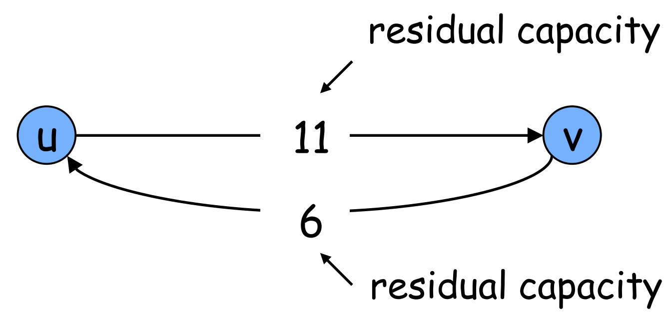
Recap: Residual Graph

– Original edge: e = (u, v) $\in$ E.
  – Flow f(e), capacity c(e).



– Residual edge.
  – "Undo" flow sent.
  – e = (u, v) and e^R = (v, u).
  – Residual capacity:

$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$



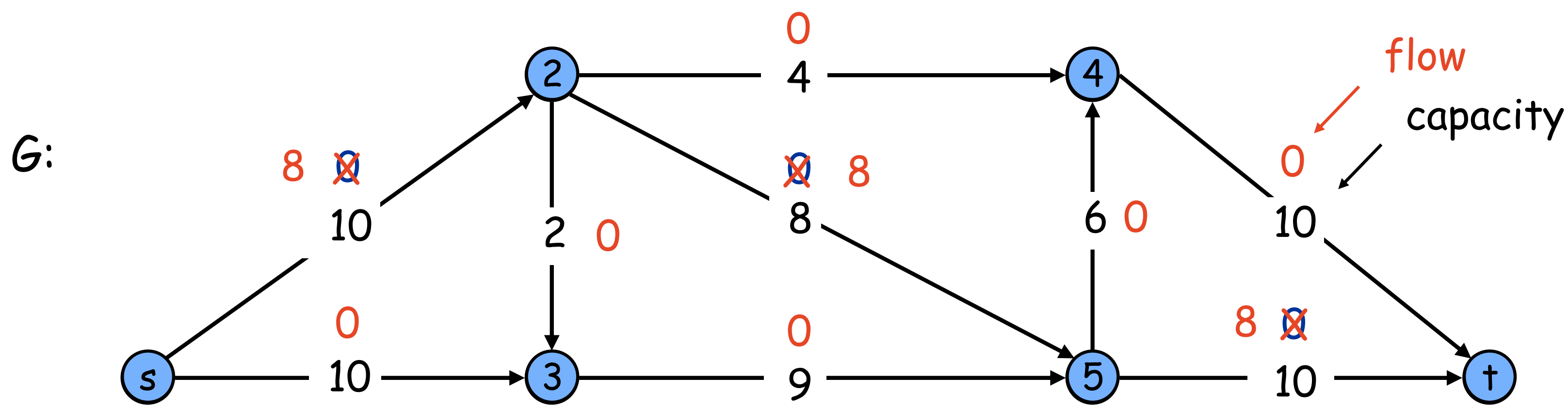
– Residual graph: G_f = (V, E_f).
  – Residual edges with positive residual capacity.
  – E_f = {e : f(e) < c(e)} $\cup$ {e^R : c(e) > 0}.

Augmenting Path Algorithm

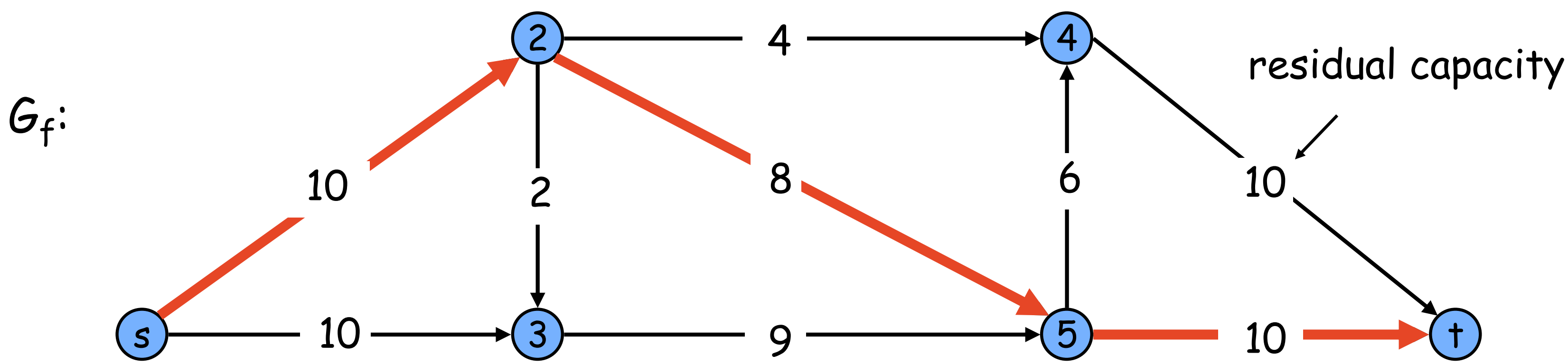
```
Ford-Fulkerson(G,s,t) {
   foreach e ∈ E
        f(e) ← 0
   Gf ← residual graph

   while (there exists augmenting path P in Gf){
      f ← Augment(f,P)
      update Gf
   }
   return f
}
```

```
Augment(f,P) {
   b ← bottleneck(P,f)
   foreach e =(u,v) ∈ P {
      if e is a forward edge then
         increase f(e) in G by b
      else (e is a backward edge)
         decrease f(e) in G by b
   }
   return f
}
```

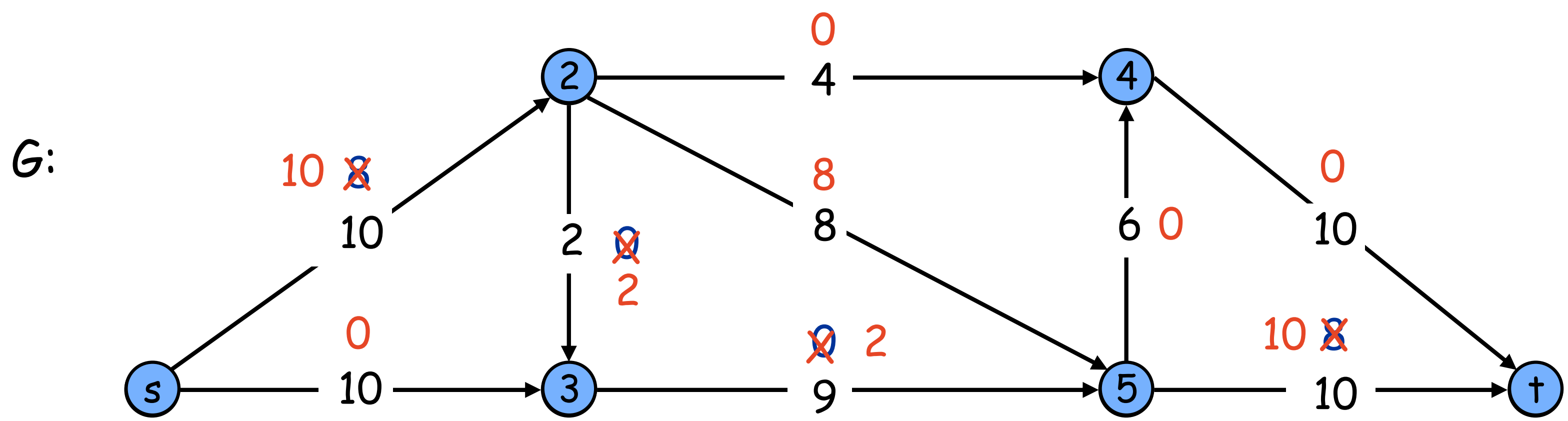
Recap: Ford-Fulkerson Algorithm



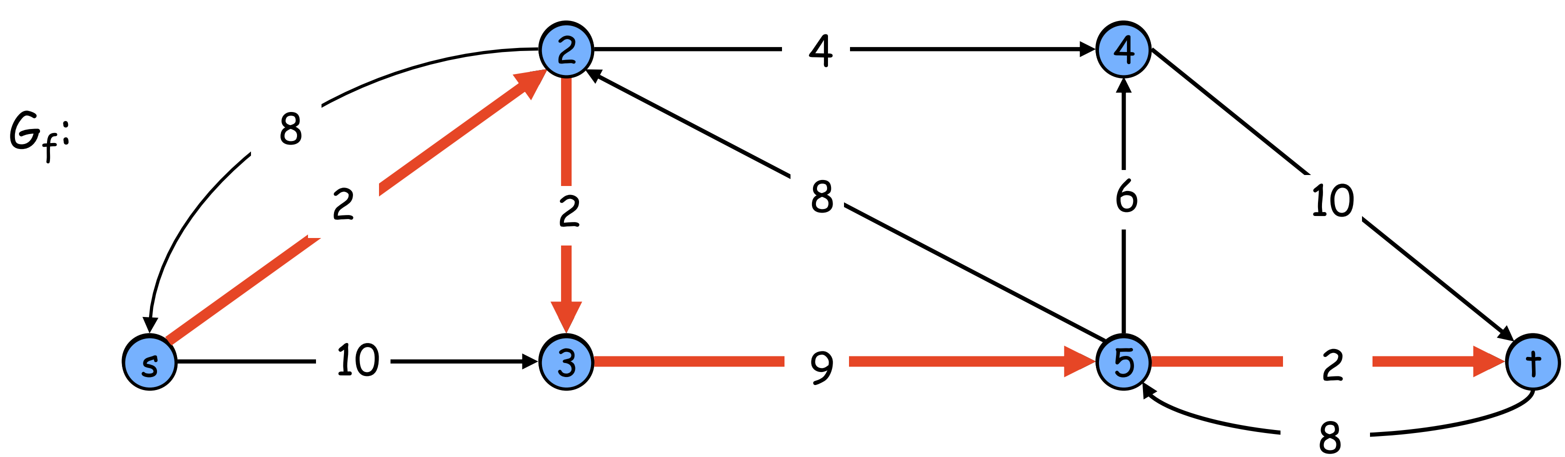
Flow value = 0



Recap: Ford-Fulkerson Algorithm



Flow value = 8



Recap: Max-Flow Min-Cut Theorem

– Augmenting path theorem: Flow f is a max flow if and only if there are no augmenting paths in the residual graph.

– Max-flow min-cut theorem: The value of the max flow is equal to the value of the min cut. [Ford-Fulkerson 1956]

– Integrality. If all capacities are integers then every flow value f(e) and every residual capacities c_f (e) remains an integer throughout the algorithm.

Recap: Running Time

Notation: $C = \sum_{e \text{ out of } s} c(e)$

Observation: C is an upper bound on the maximum flow.

Theorem. Ford-Fulkerson runs in $O(Cm)$ time.

Theorem. The scaling max-flow algorithm finds a max flow $O(m^2 \log C)$ time.

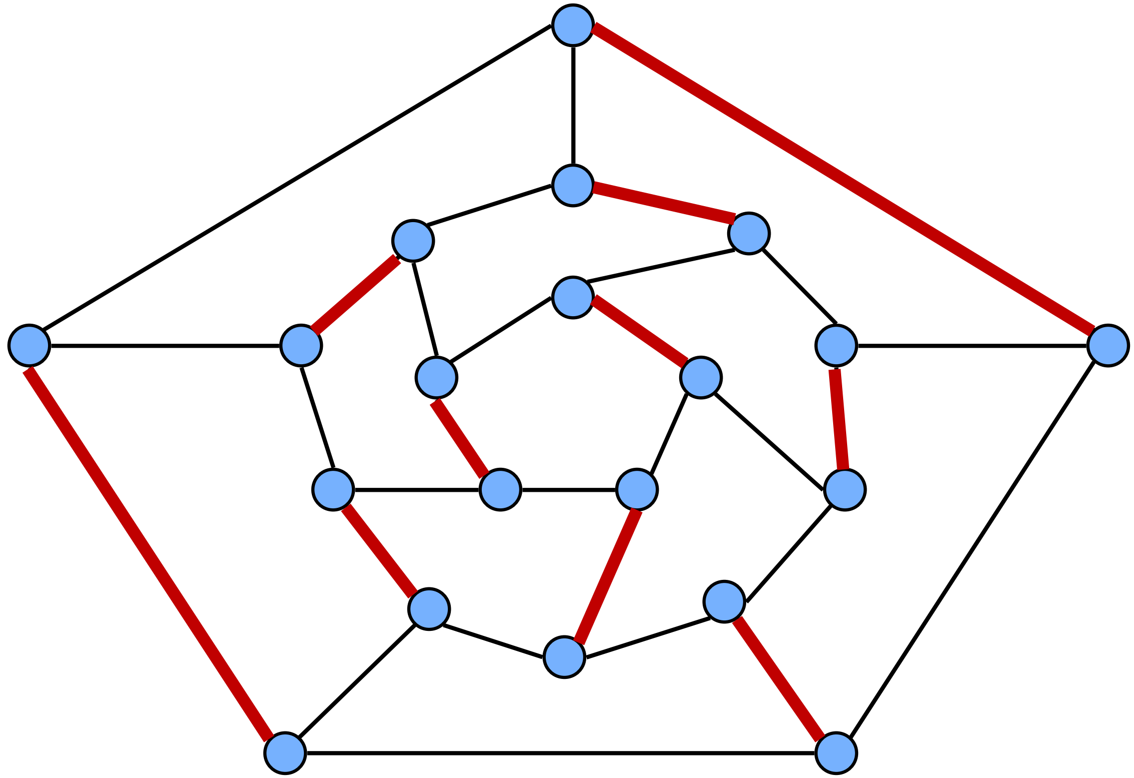
Applications of max flow

- Bipartite matching
- Perfect matching
- Disjoint paths
- Network connectivity
- Circulation problems
- Image segmentation
- Baseball elimination
- Project selection

7.5 Bipartite Matching

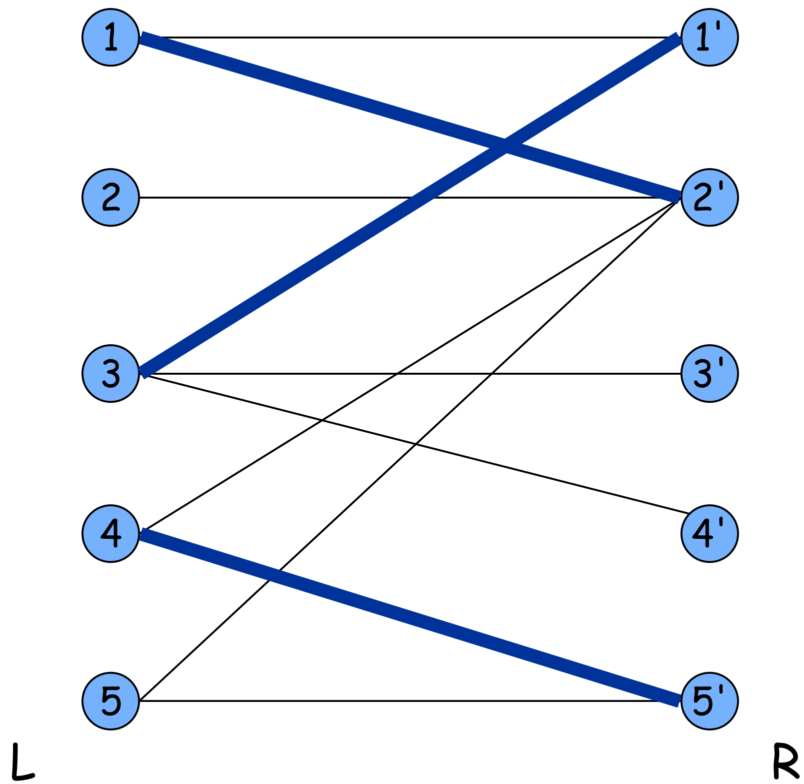
Matching

- 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

- 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.

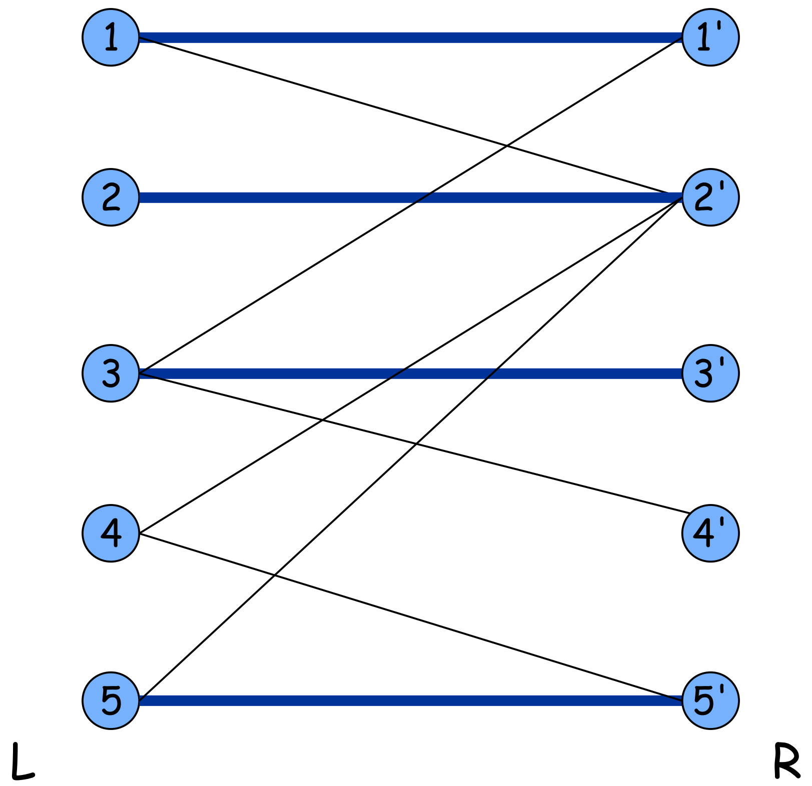


<u>matching</u>

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

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.



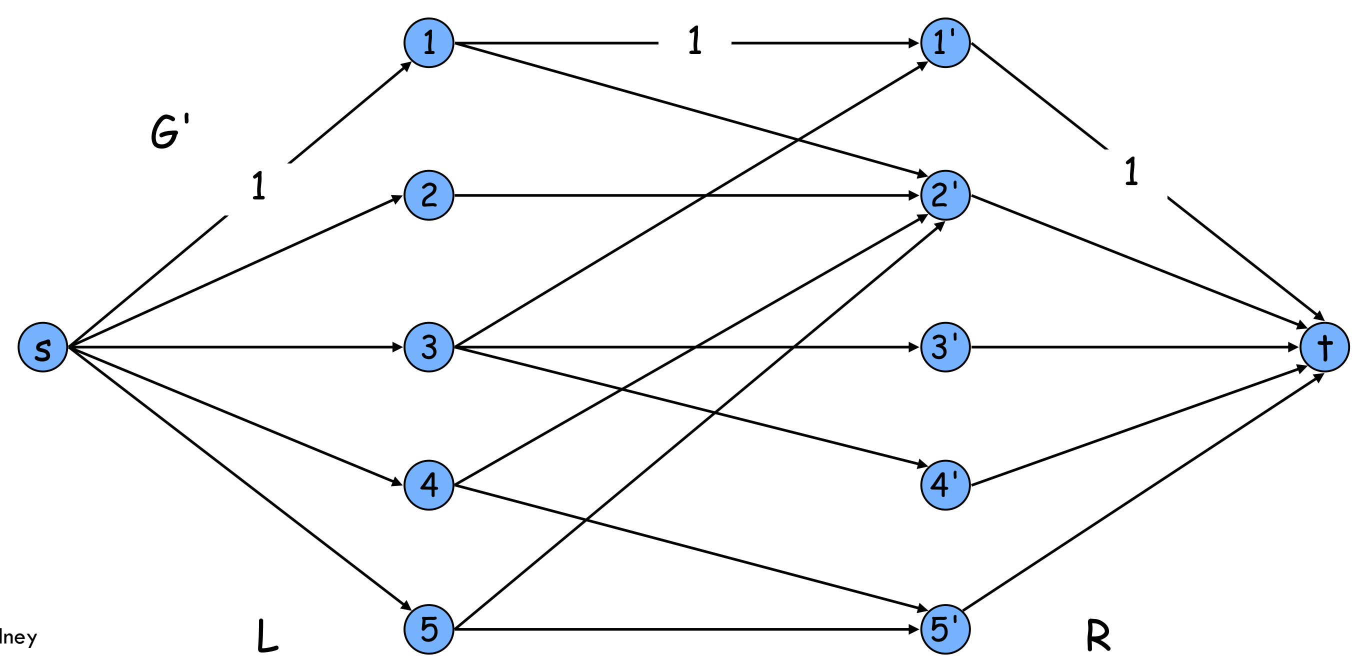
max matching

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

Bipartite Matching

Max flow formulation.

– Create digraph $G' = (L \cup R \cup \{s, t\}, E')$.
– Direct all edges from L to R, and assign 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.

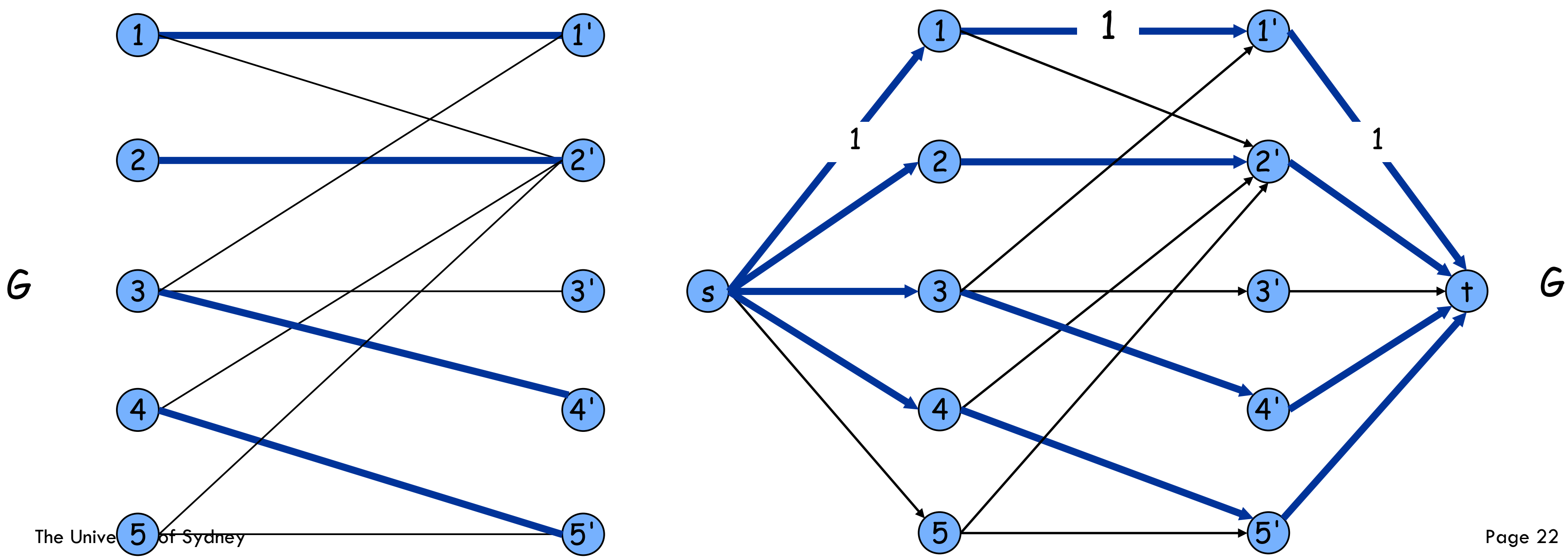


Bipartite Matching: Proof of Correctness

Theorem: Max cardinality matching in G $\Leftrightarrow$ value of max flow in G'.

Proof: $\Rightarrow$

- Assume max matching M has cardinality k.
- Consider a flow f that sends 1 unit along each of the k paths, defined by the edges in M.
- f is a flow, and it has value k. ■

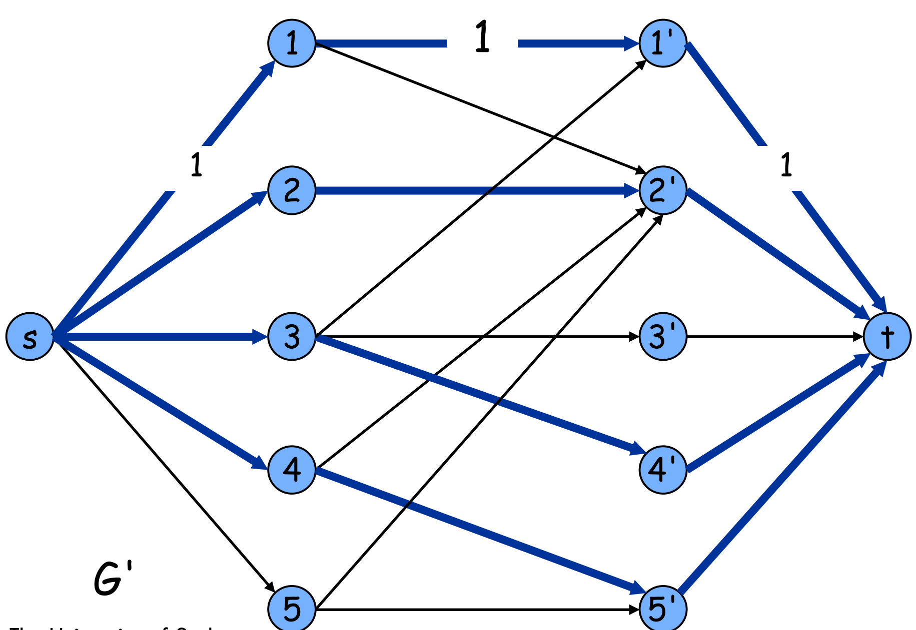


Bipartite Matching: Proof of Correctness

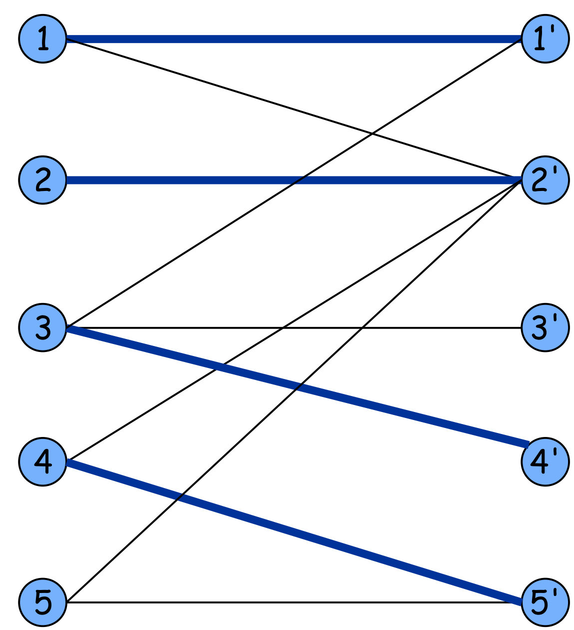
Theorem: Max cardinality matching in G $\Leftrightarrow$ value of max flow in G'.

Proof: $\Leftarrow$

- Let f be a max flow in G' of value k.
- Integrality theorem $\Rightarrow$ k is integral so f(e) is 0 or 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 ($L \cup s$, $R \cup t$)



G'



G

Perfect Matching

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

Question: When does a bipartite graph have a perfect matching?

Structure of bipartite graphs with perfect matchings.

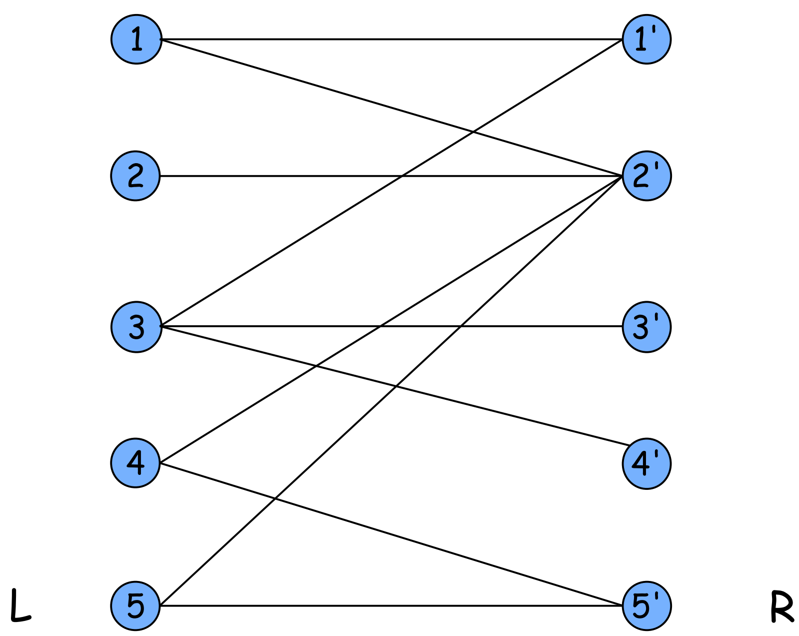
- Clearly we must have $|L| = |R|$.
- What other conditions are necessary?
- What conditions are sufficient?

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 $S \subseteq L$.

Proof: Each node in S has to be matched to a different node in N(S).

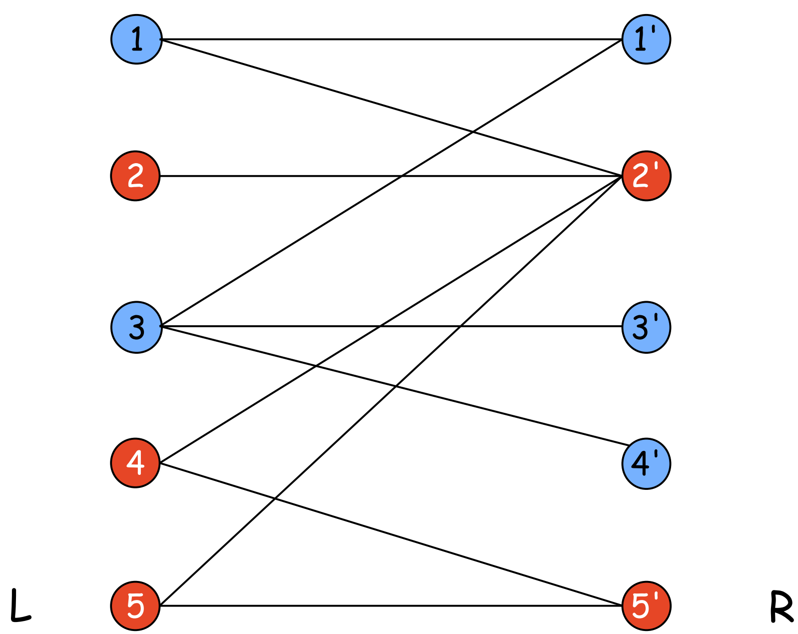


Perfect Matching

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 $S \subseteq L$.

Proof: Each node in S has to be matched to a different node in N(S).



No perfect matching:

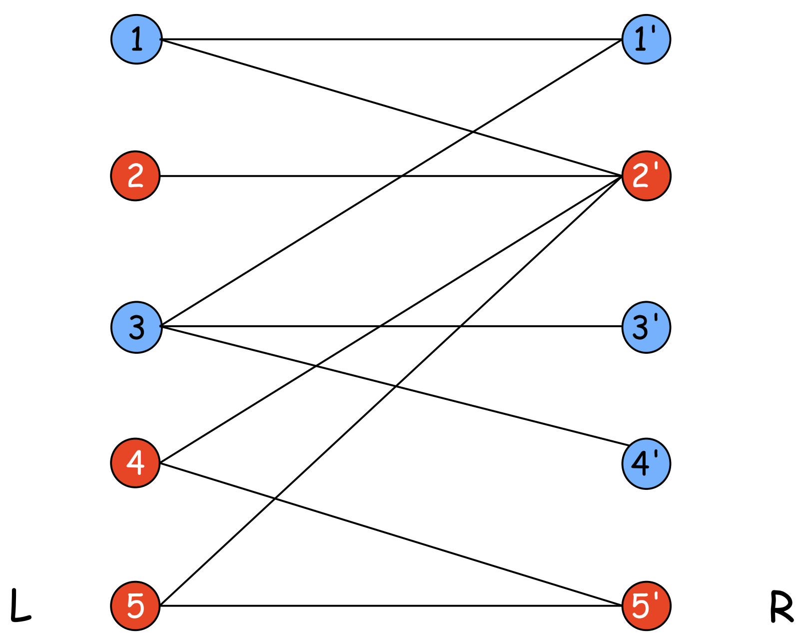
S = { 2, 4, 5 }

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

Marriage Theorem

Marriage Theorem. 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 $S \subseteq L$.

Proof: $\Rightarrow$ This was the previous observation.



No perfect matching:

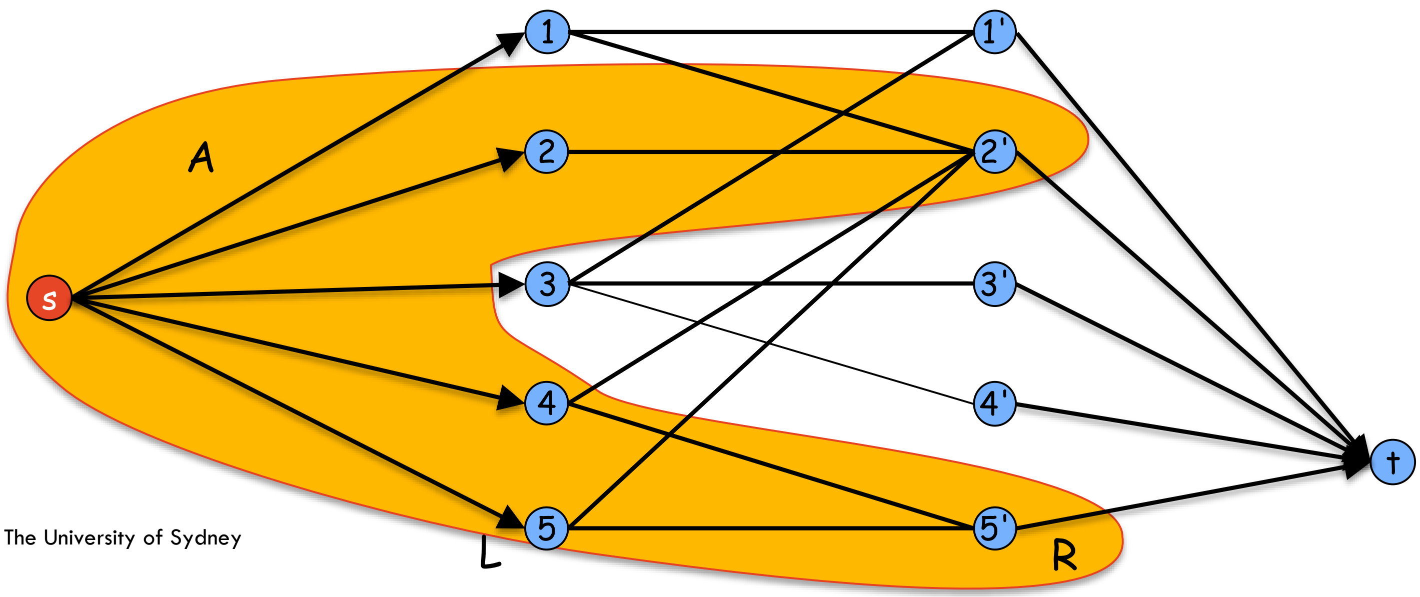
$S = \{ 2, 4, 5 \}$

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

Proof of Marriage Theorem

Proof: $\Leftarrow$ Suppose G does not have a perfect matching (flow<n).

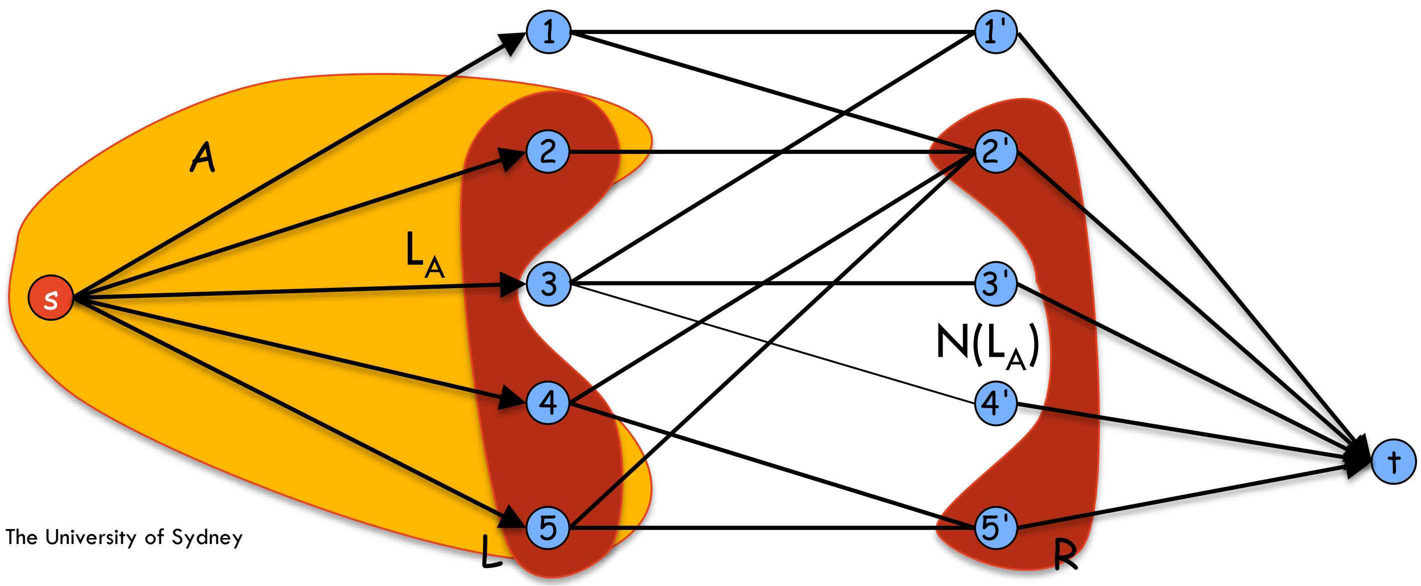
- Formulate as a max flow problem and let (A, B) be min cut in G'.
- By max-flow min-cut, cap(A, B) < n = | L | = |R|.
- Define $L_A = L \cap A$
- Idea of proof: Prove that $|N(L_A)| < |L_A|$



Proof of Marriage Theorem

Proof: $\Leftarrow$ Suppose G does not have a perfect matching (flow<n).

- Formulate as a max flow problem and let (A, B) be min cut in G'.
- By max-flow min-cut, cap(A, B) $< n = |L| = |R|$.
- Define $L_A = L \cap A$
- Idea of proof: Prove that $|N(L_A)| < |L_A|$



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

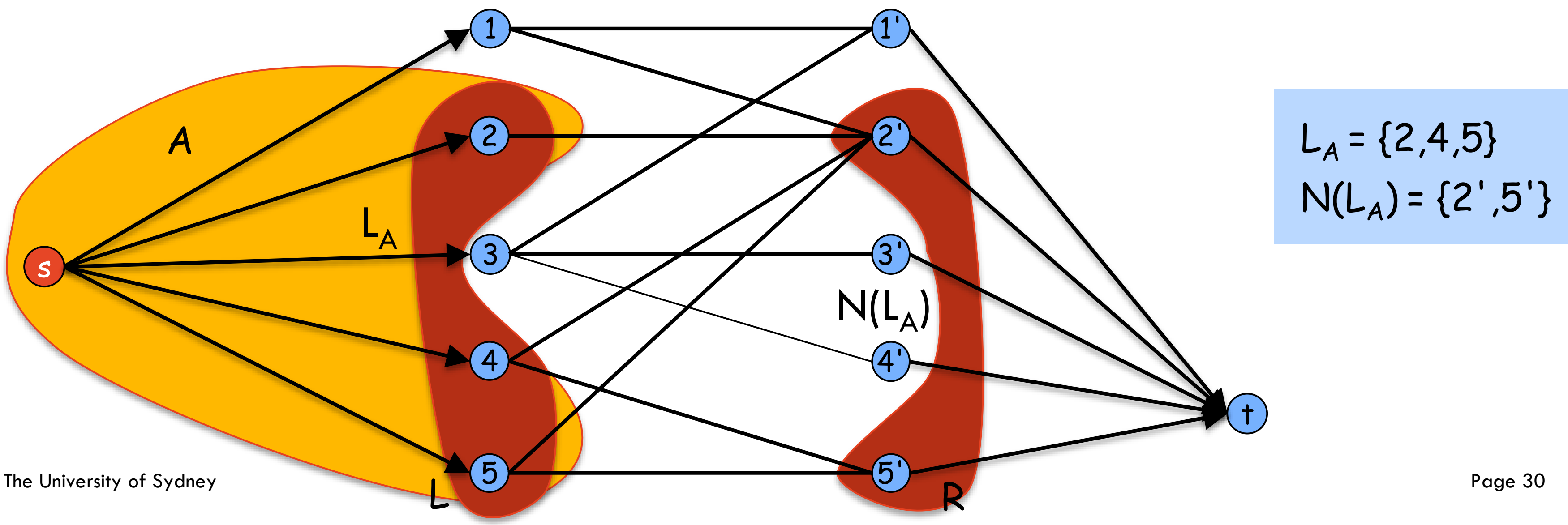
$N(L_A) = \{2',5'\}$

Proof of Marriage Theorem

Proof: $\Leftarrow$ Suppose G does not have a perfect matching (flow<n).

Idea of proof: Prove that $|N(L_A)| < |L_A|$

– Claim 1: One can modify the MinCut so that $N(L_A) \subseteq A$.

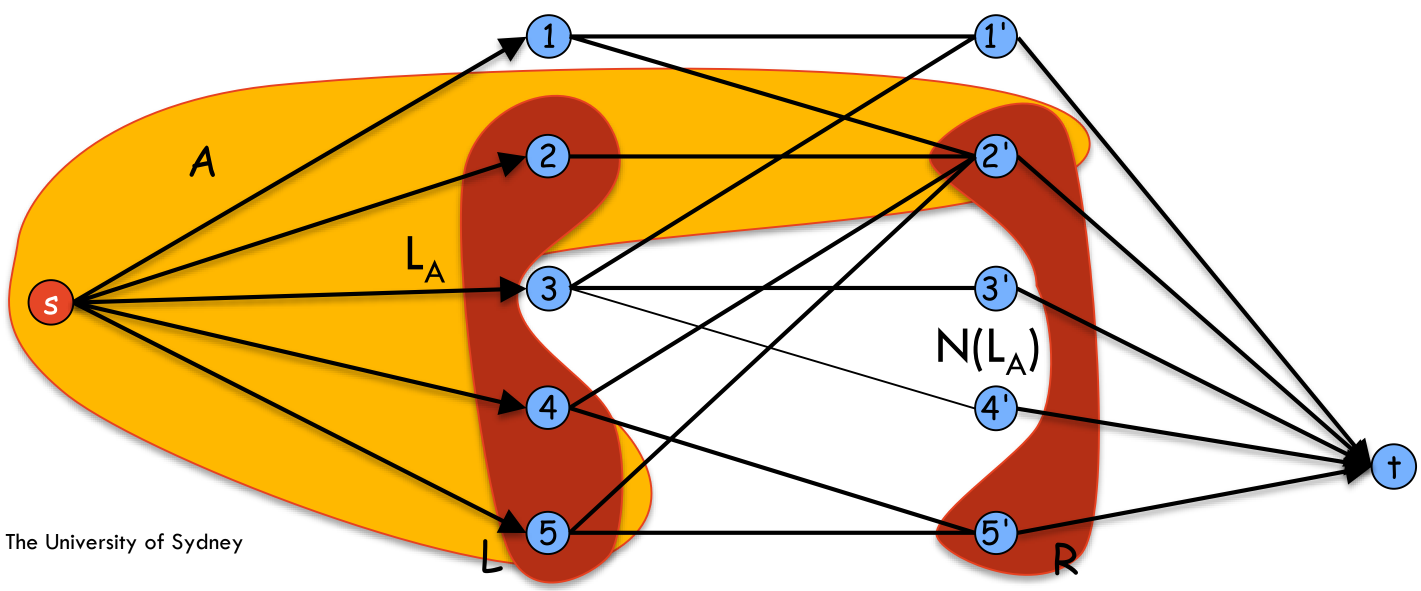


Proof of Marriage Theorem

Proof: $\Leftarrow$ Suppose G does not have a perfect matching (flow<n).

Idea of proof: Prove that $|N(L_A)| < |L_A|$

– Claim 1: One can modify the MinCut so that $N(L_A) \subseteq A$.



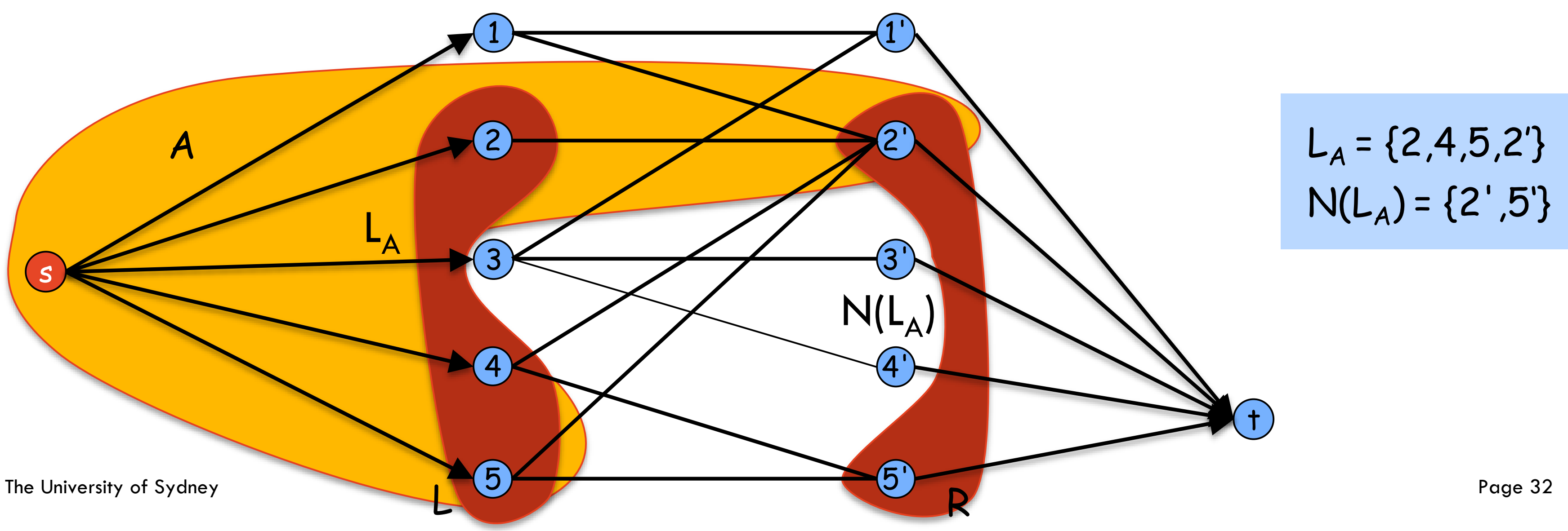
$L_A = \{2,4,5\}$
$N(L_A) = \{2',5'\}$

Proof of Marriage Theorem

Proof: $\Leftarrow$ Suppose G does not have a perfect matching (flow<n).

Idea of proof: Prove that $|N(L_A)| < |L_A|$

– Claim 1: One can modify the MinCut so that $N(L_A) \subseteq A$.
  – Edge (2',t) crosses the cut (capacity +1)
  – Edge (2,2') inside A (capacity -1) $\Rightarrow$ capacity does not increase!
    [(4,2') and (5,2') also ends up in A]

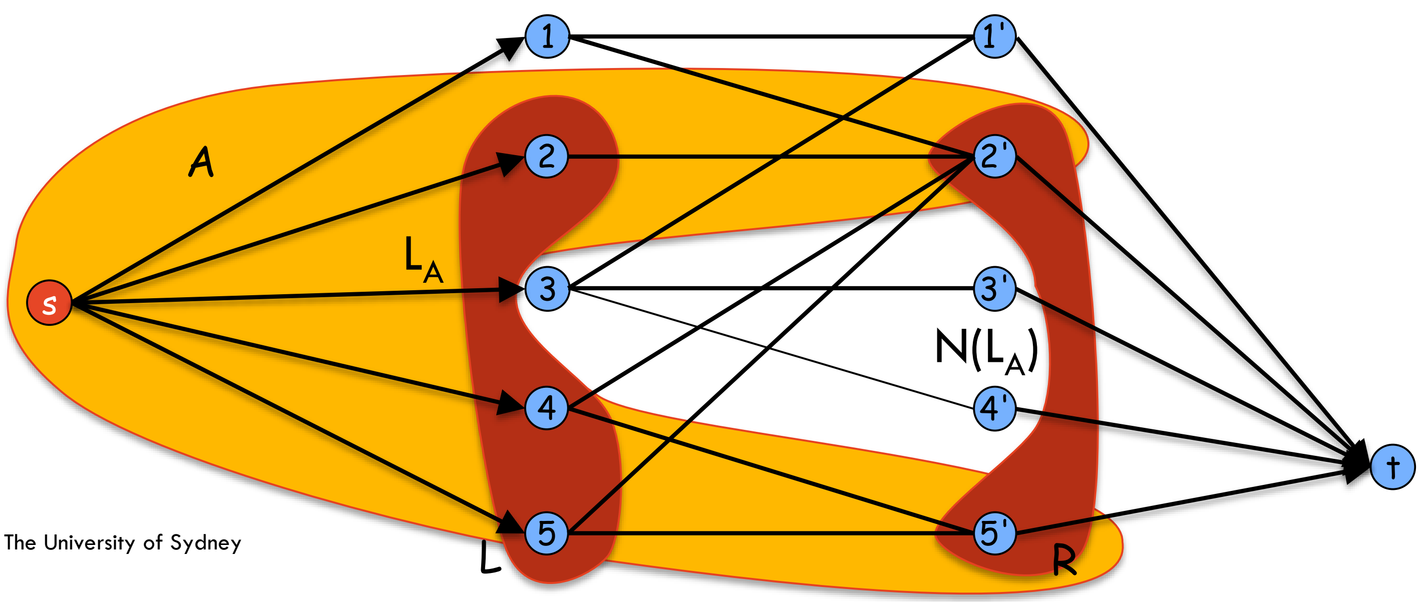


Proof of Marriage Theorem

Proof: $\Leftarrow$ Suppose G does not have a perfect matching (flow<n).

Idea of proof: Prove that $|N(L_A)| < |L_A|$

- Claim 1: One can modify the MinCut so that $N(L_A) \subseteq A$.
  - Edge (5',t) crosses the cut (capacity +1)
  - Edges (4,5') and (5,5') inside A (capacity -2) $\Rightarrow$ capacity does not increase!



$L_A = \{2,4,5,2',5'\}$
$N(L_A) = \{2',5'\}$

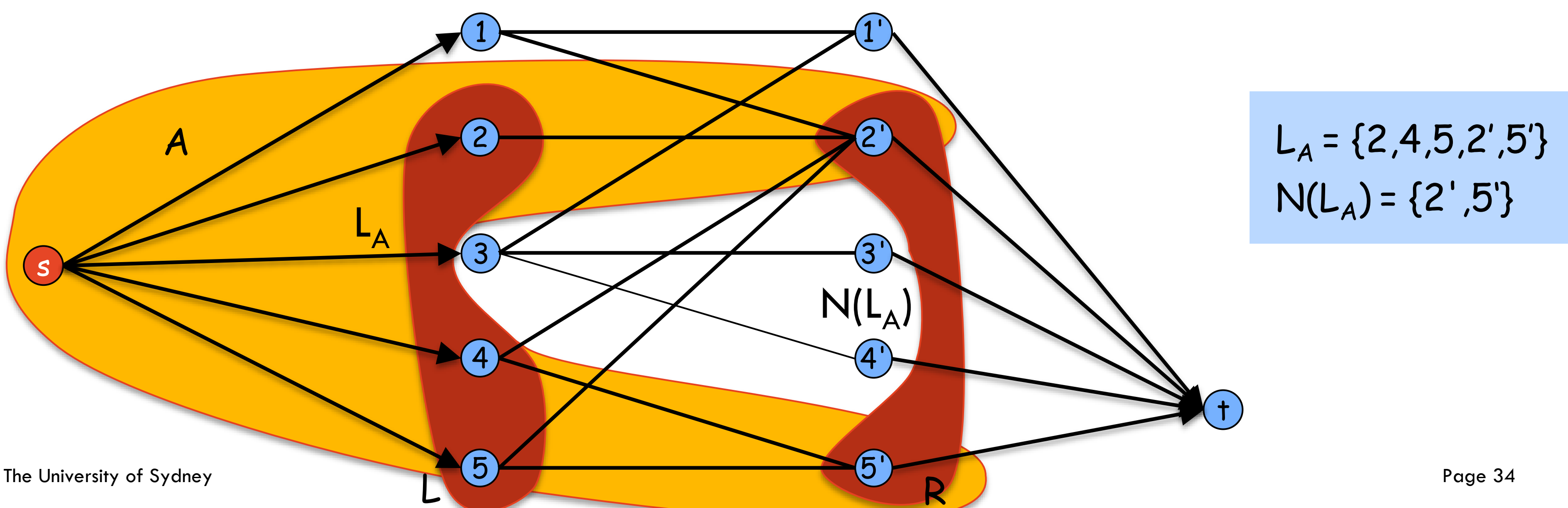
Proof of Marriage Theorem

Proof: $\Leftarrow$ Suppose G does not have a perfect matching (flow<n).

Idea of proof: Prove that $|N(L_A)| < |L_A|$

– **Claim 2:** Consider capacity of this cut (A,B).
  – All vertices in $N(L_A)$ are in A: $c(A,B) = |L \cap B| + |R \cap A|$
  – Since $|L \cap B| = n - |L_A|$ and $|R \cap A| \geq |N(L_A)|$ we have:

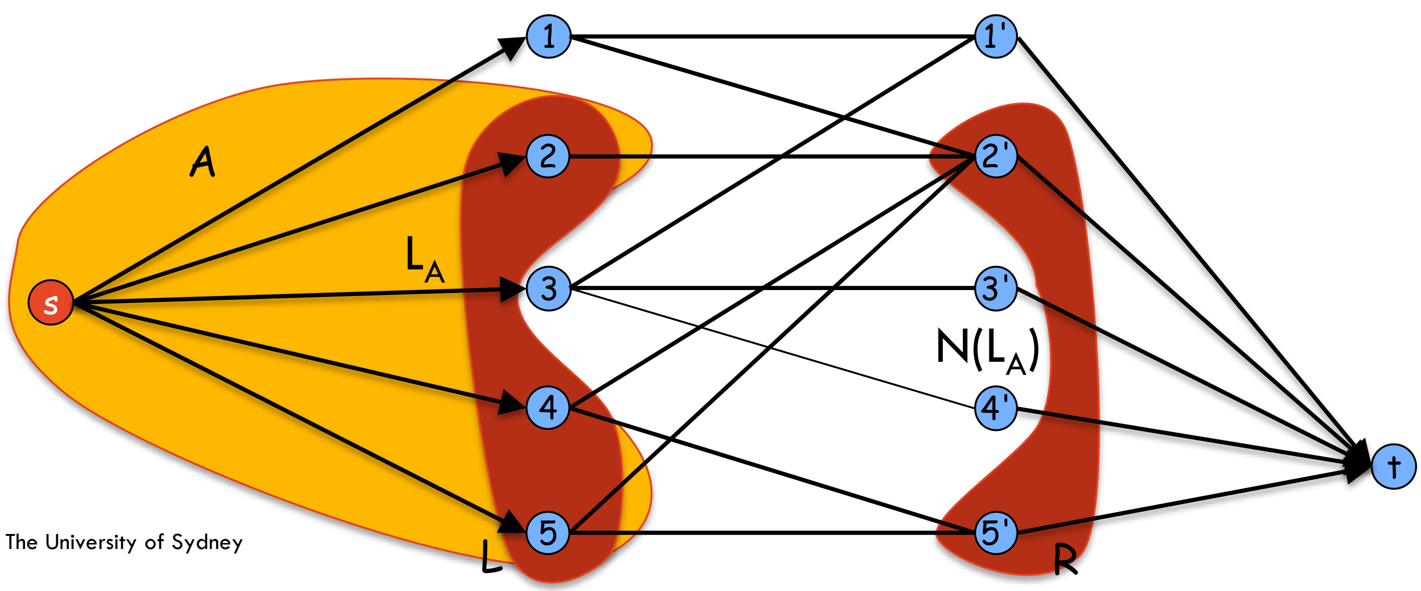
$$n > c(A,B) = |L \cap B| + |R \cap A| \geq n - |L_A| + |N(L_A)| \Rightarrow |L_A| > |N(L_A)| \quad \blacksquare$$



Proof of Marriage Theorem

Proof: $\Leftarrow$ Suppose G does not have a perfect matching (flow<n).

- Either G has a perfect matching, or
- there exists a subset $L_A \subseteq L$ such that $|N(L_A)| < |L_A|$



$L_A = \{2,4,5\}$
$N(L_A) = \{2',5'\}$

Bipartite Matching: Running Time

Which max flow algorithm to use for bipartite matching?

– Generic augmenting path: $O(mC) = O(mn)$.
– Capacity scaling: $O(m^2 \log C) = O(m^2 \log n)$.

– Best known: $O(mn^{1/2})$. [Micali-Vazirani 1980]

Non-bipartite matching:

– Structure of non-bipartite graphs is more complicated, but well-understood. [Tutte-Berge, Edmonds-Galai]
– Blossom algorithm: $O(mn^2)$. [Edmonds 1965]

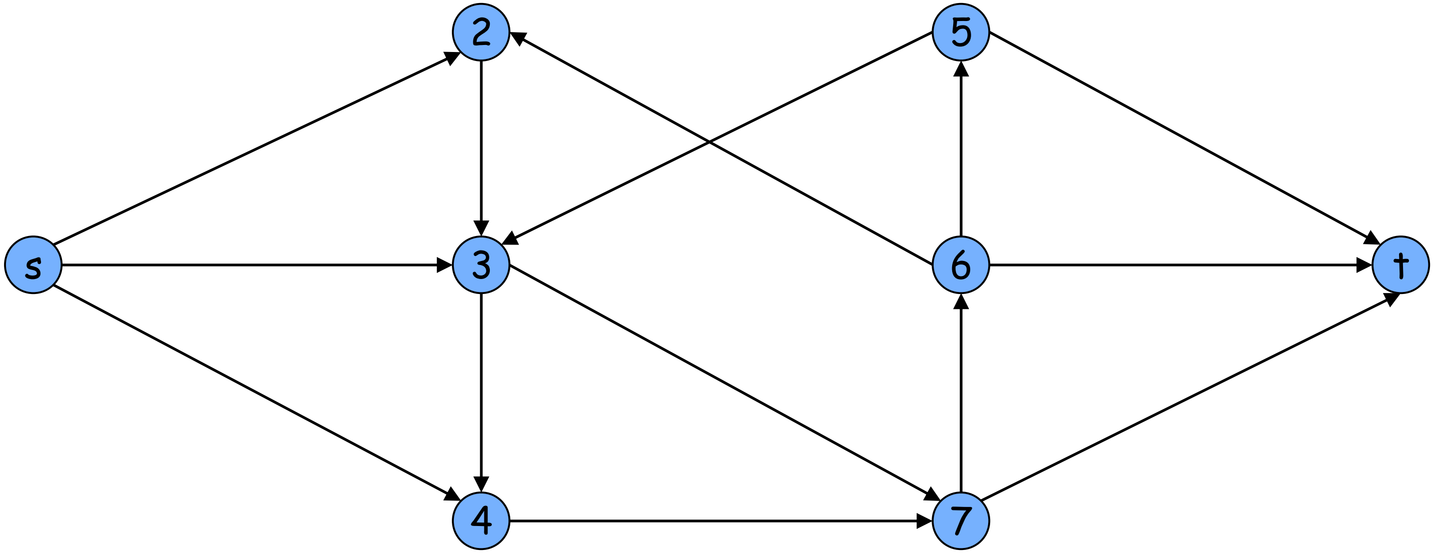
7.6 Disjoint Paths

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.

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

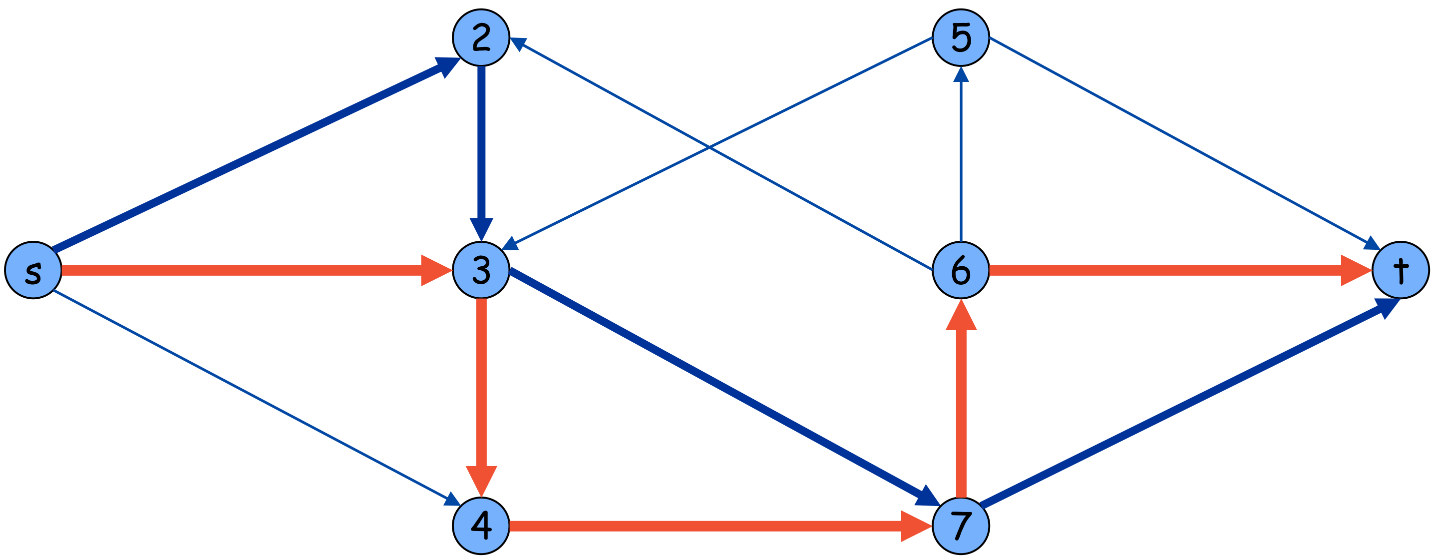


Edge Disjoint Paths

Disjoint path problem:

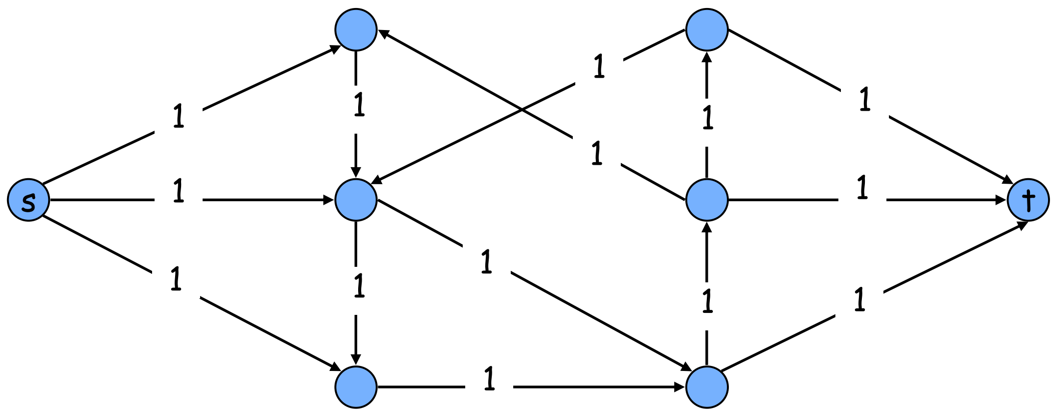
Given a digraph G = (V, E) and two nodes s and t, find the max number of edge-disjoint s-t paths.

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



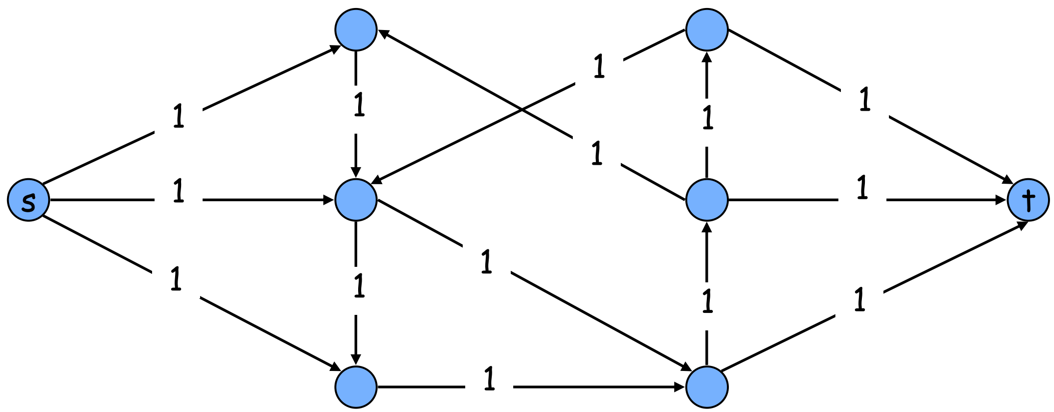
Edge Disjoint Paths

Max flow formulation: assign unit capacity to every edge.



Edge Disjoint Paths

Max flow formulation: assign unit capacity to every edge.



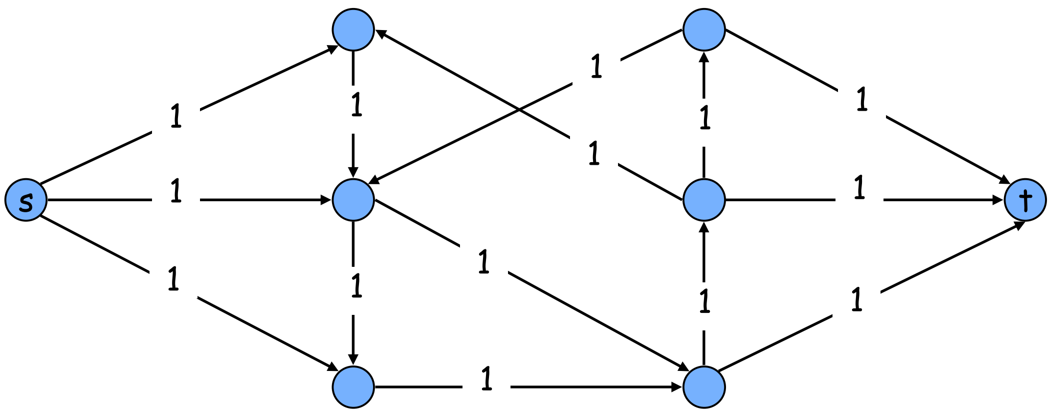
Theorem: Max number edge-disjoint s-t paths equals max flow value.

Proof: $\Rightarrow$

- 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.

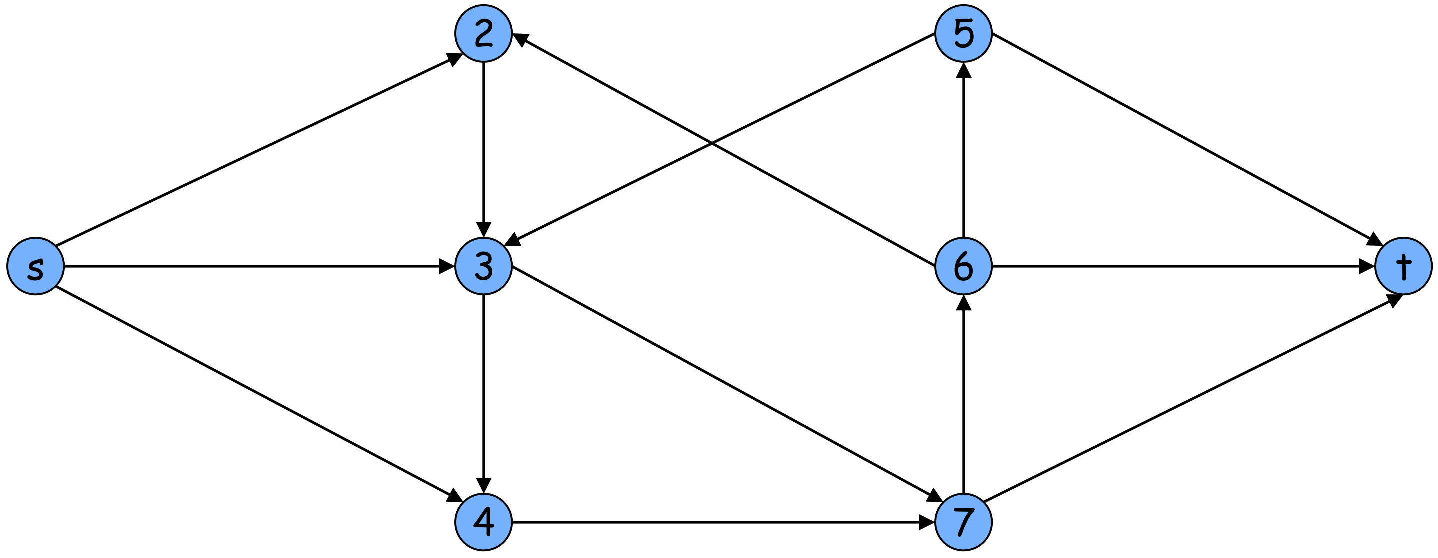
Proof: $\Leftarrow$

- 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. ▪

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.

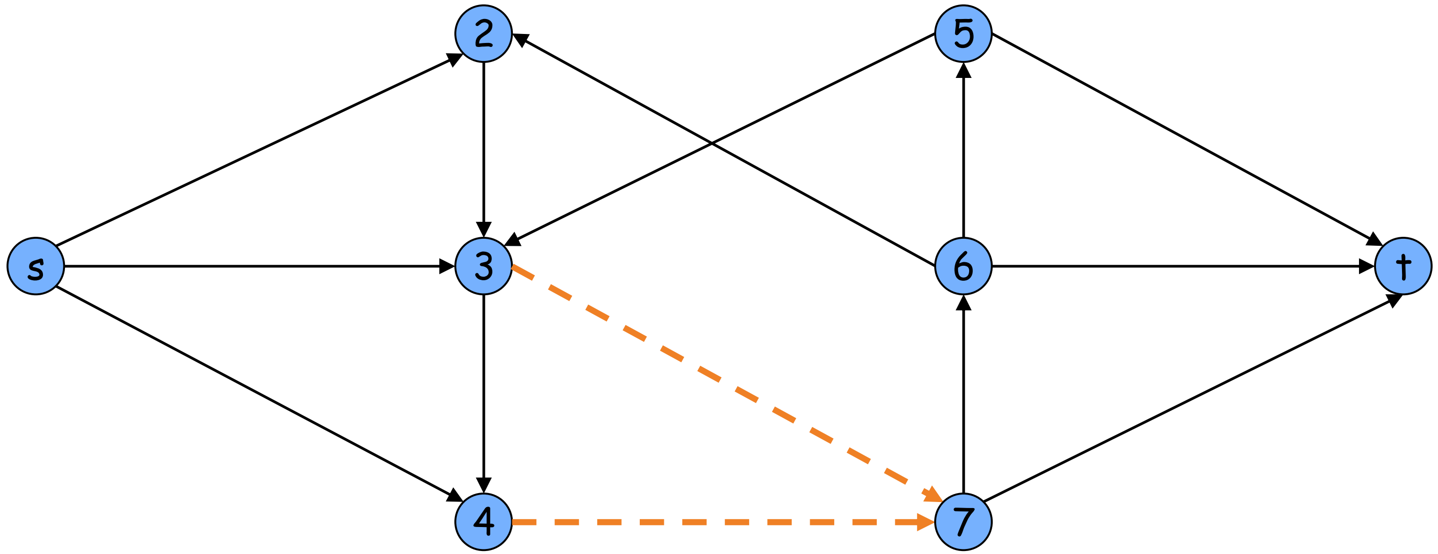
Definition: A set of edges $F \subseteq E$ disconnects t from s if all s-t paths uses at least one edge in F.



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.

Definition: A set of edges $F \subseteq E$ disconnects t from s if all s-t paths uses at least one edge in F.

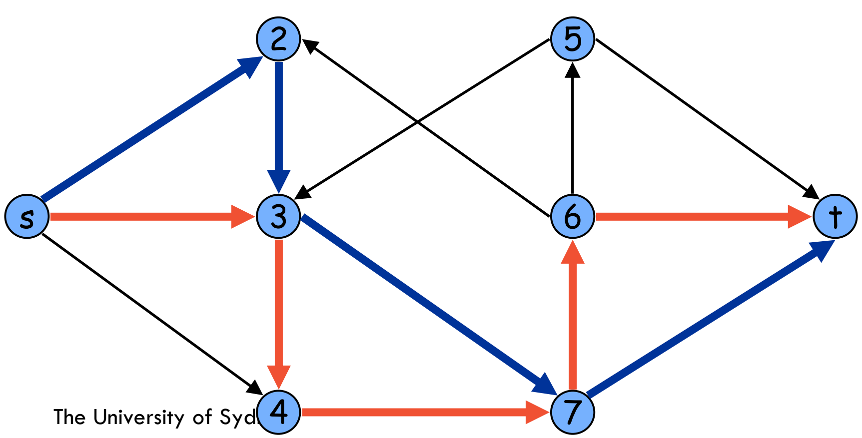


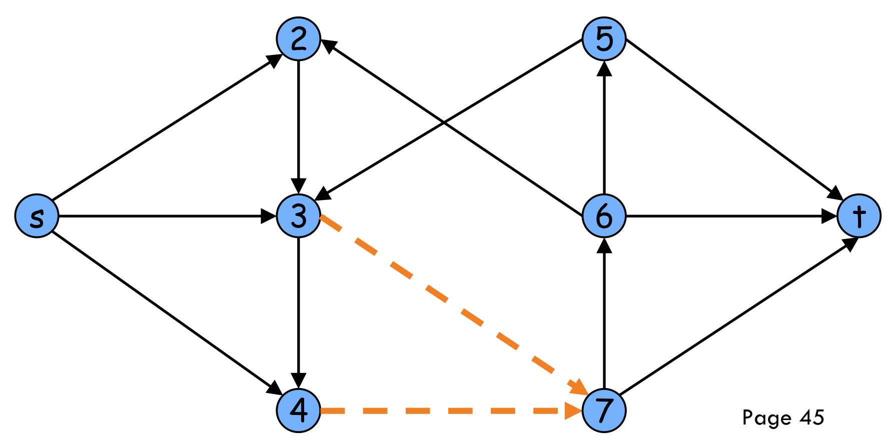
Edge Disjoint Paths and Network Connectivity

Theorem: The max number of edge-disjoint s-t paths is equal to the min number of edges whose removal disconnects t from s.

Proof: $\Leftarrow$

- Suppose the removal of $F \subseteq E$ disconnects t from s, and $|F| = k$.
- Every s-t path uses <u>at least</u> one edge of F. Hence, the number of edge-disjoint paths is at most k. ■



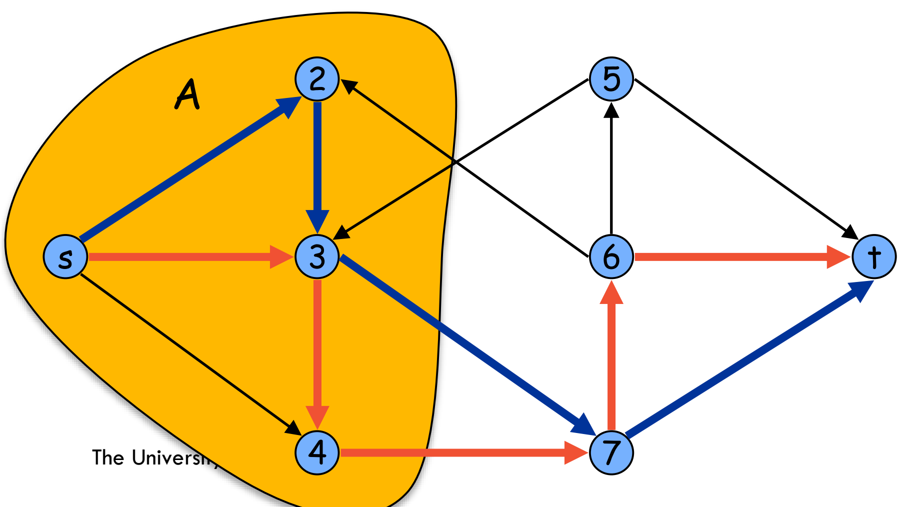


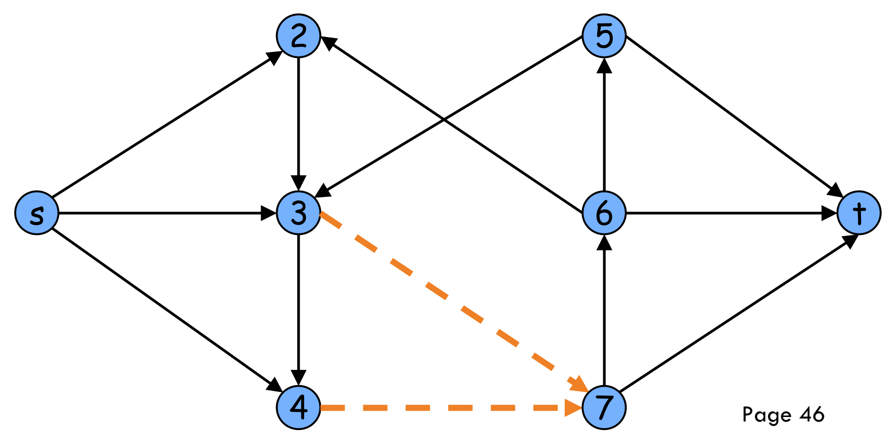
Edge Disjoint Paths and Network Connectivity

Theorem: The max number of edge-disjoint s-t paths is equal to the min number of edges whose removal disconnects t from s.

Proof: $\Rightarrow$

- 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 $d(v) = 0$

Definition: 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)

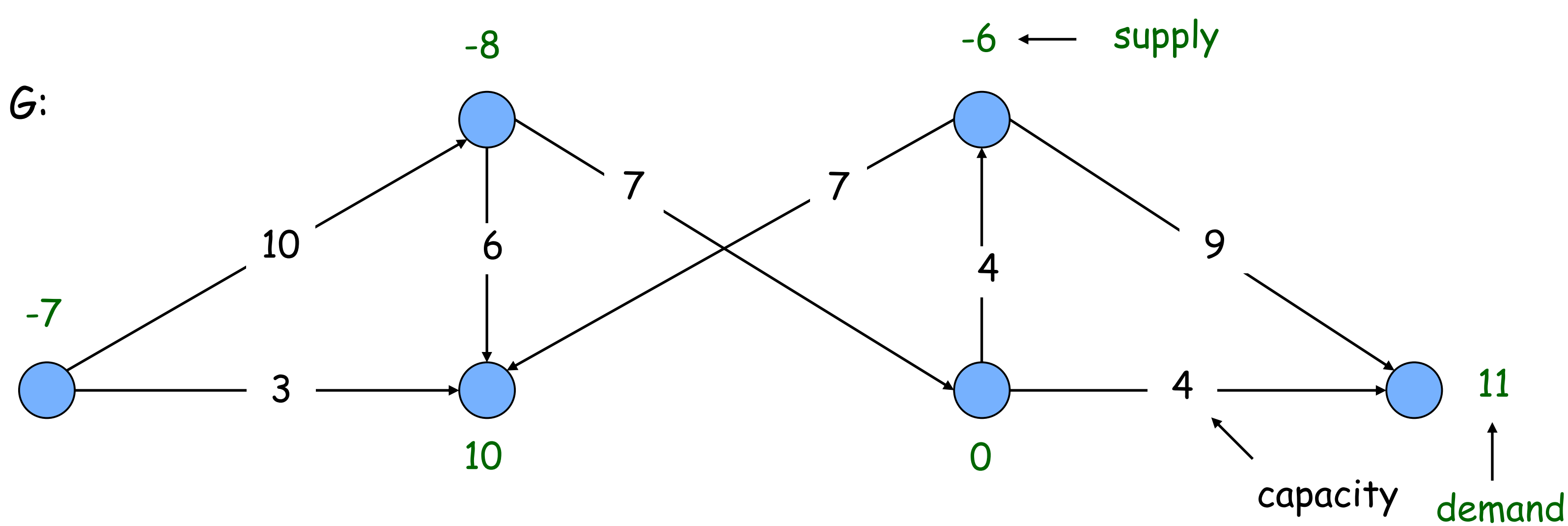
Circulation problem: Given (V, E, c, d), does there exist a circulation?

Circulation with Demands

Necessary condition: sum of supplies = sum of demands.

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

Proof: Sum conservation constraints for every demand node v.

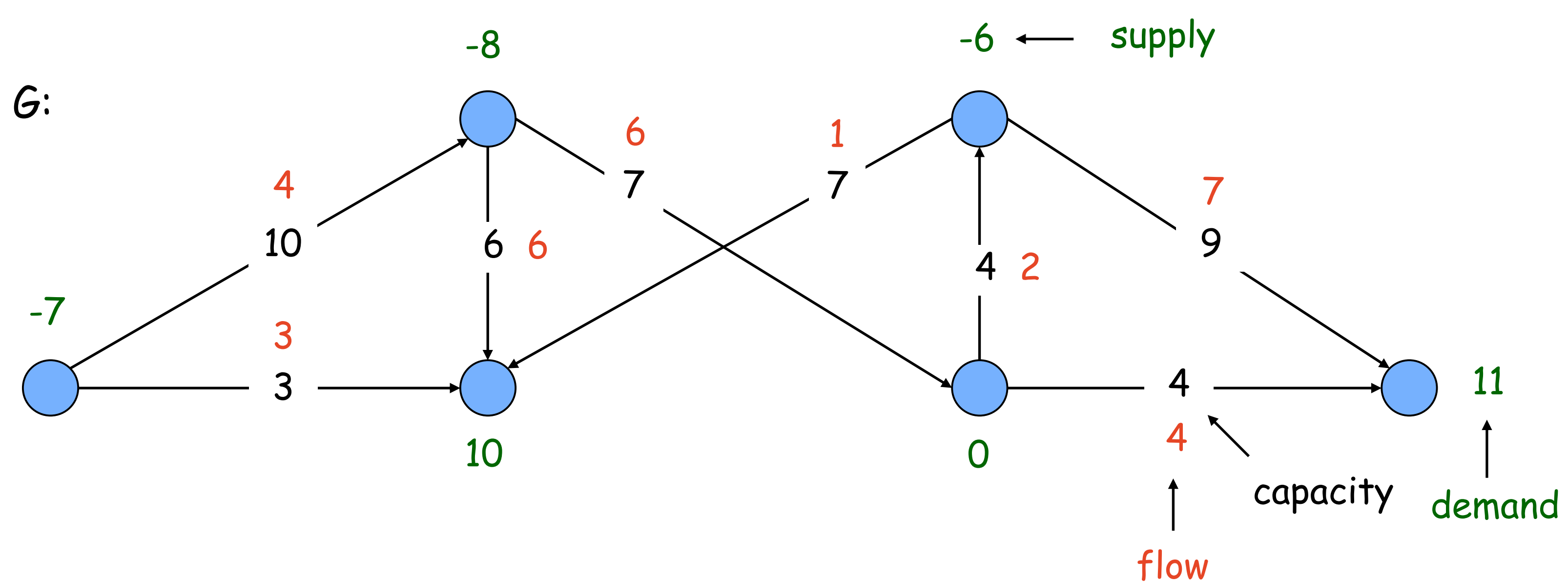


Circulation with Demands

Necessary condition: sum of supplies = sum of demands.

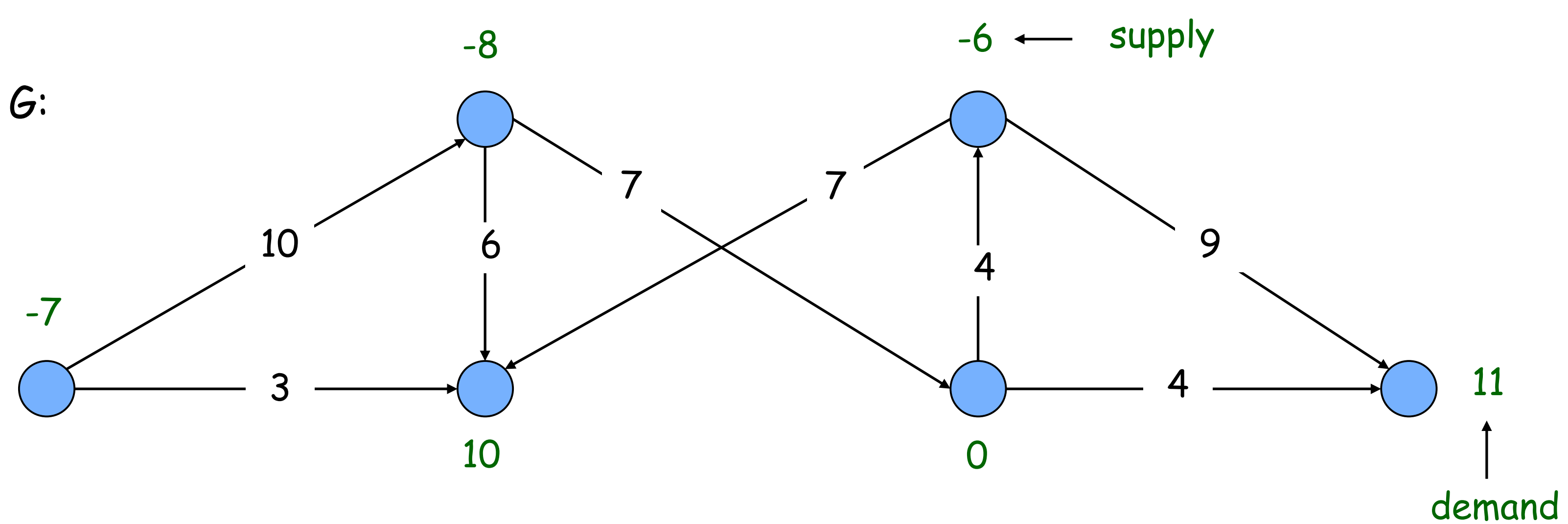
$$\sum_{v:d(v)>0} d(v) = \sum_{v:d(v)<0} -d(v) =: D$$

Proof: Sum conservation constraints for every demand node v.



Circulation with Demands

Max flow formulation.

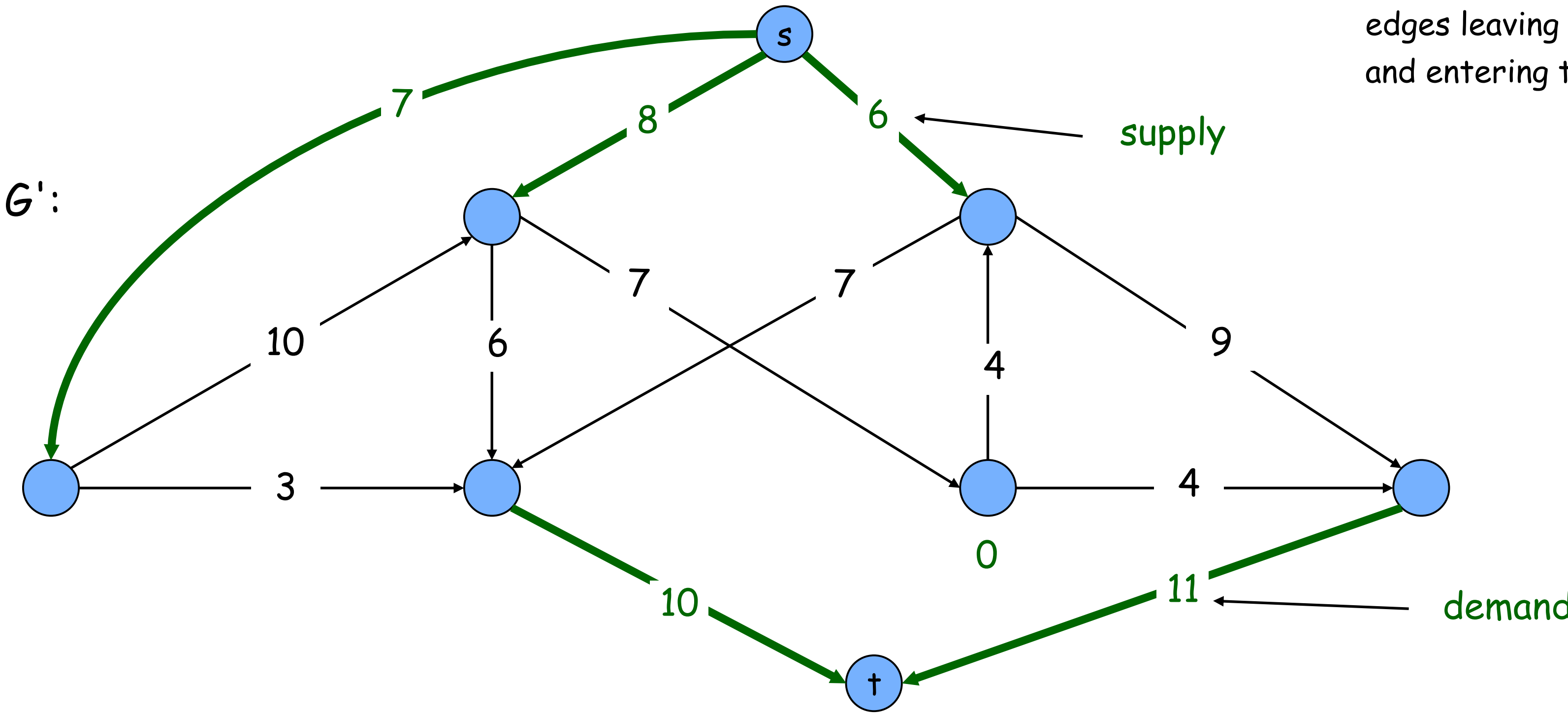


Circulation with Demands

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.

Proof: Follows from max flow formulation and integrality theorem for max flow.

Characterization.

Given (V, E, c, d), there is a feasible circulation with demand d_v iff for all cuts (A, B) ,

$$\Sigma_{v \in B} d_v \leq cap(A, B).$$

Proof idea: Look at min cut in G'. Similar proof as the proof for the Marriage Theorem.

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 $d(v)$, $v \in V$.

Definition: 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)

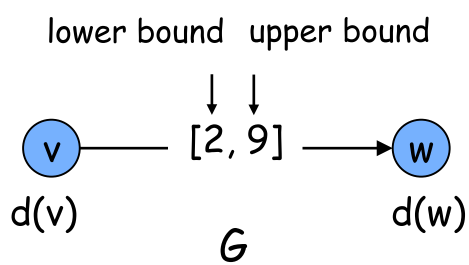
Circulation problem with lower bounds.

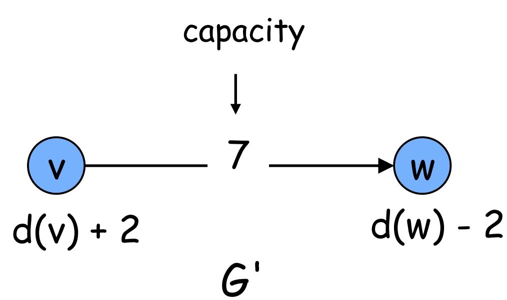
Given (V, E, ℓ, c, d), does there exists a circulation?

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.

Proof 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: Problem

- 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.

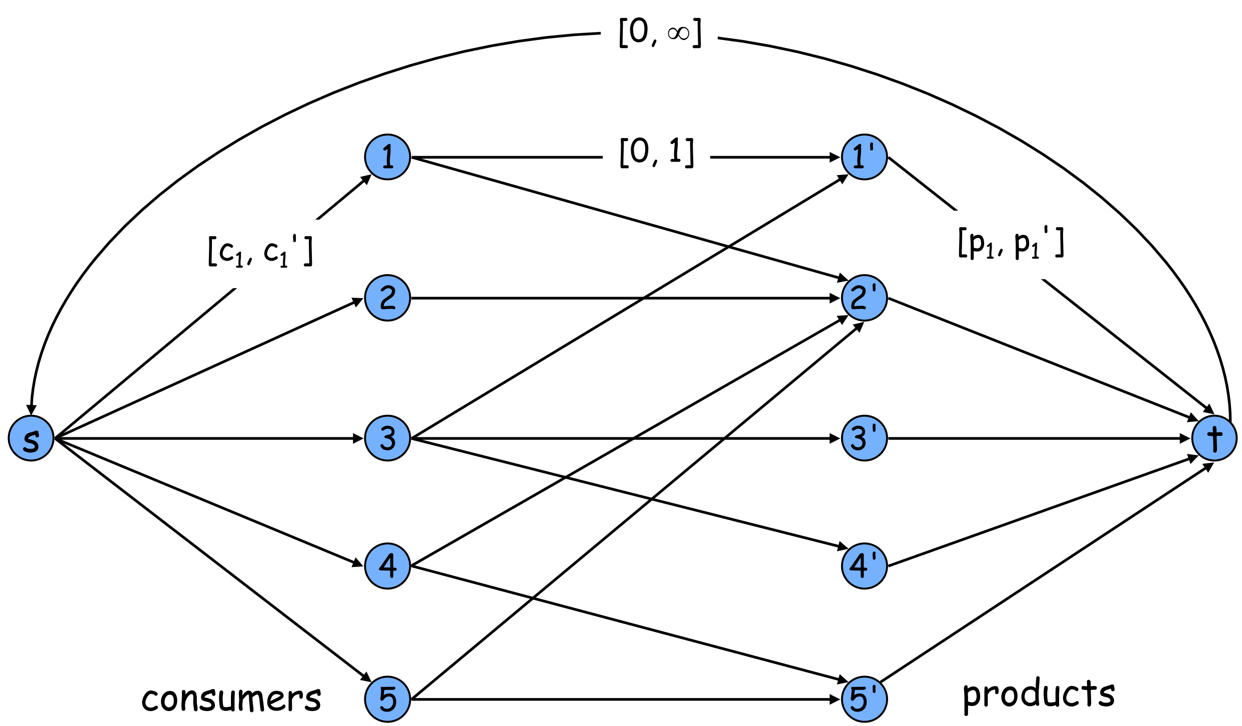
Goal: Design a survey that meets these specs, if possible.

Survey Design: Algorithm

Formulate as a circulation problem with lower bounds.

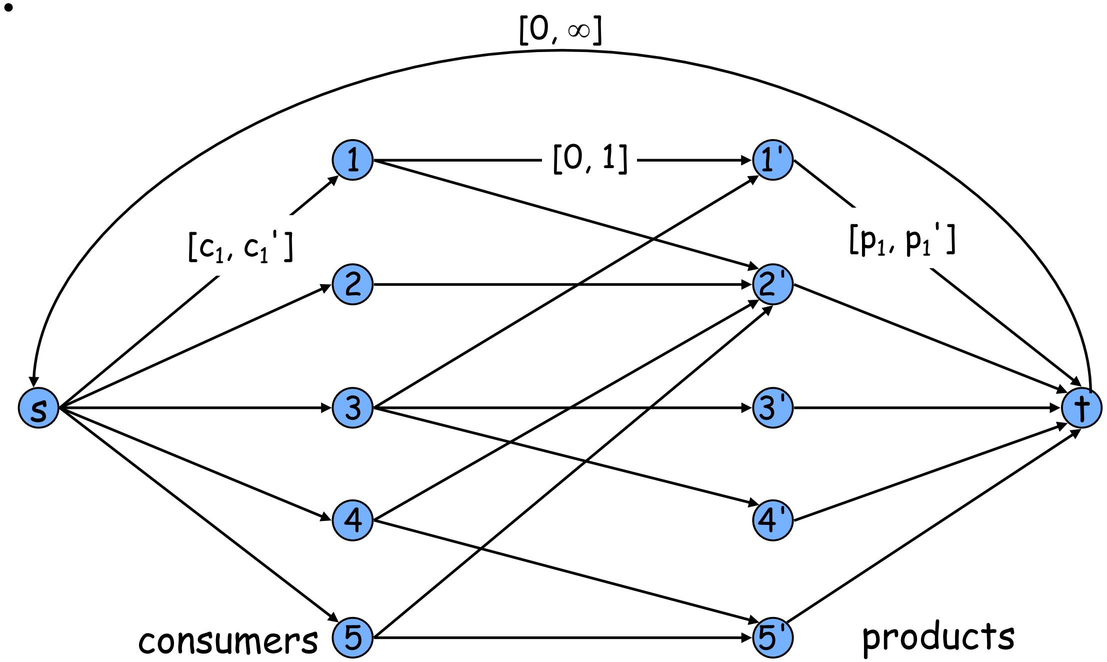
– Include an edge (i, j) if customer own product i.

– Integer circulation $\Leftrightarrow$ feasible survey design.



Survey Design: Correctness

1. If the Circulation problem (with lower bounds) is feasible then the Survey Design problem is feasible.
2. If the Survey Design problem is feasible then the Circulation problem is feasible.



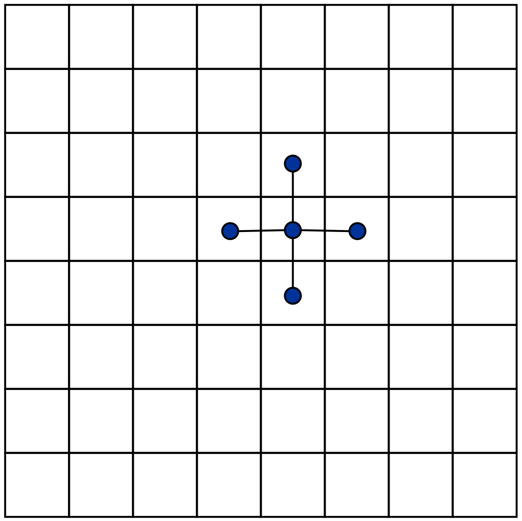
7.10 Image Segmentation

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: Problem



Foreground / background segmentation.

- V = set of pixels, E = pairs of neighboring pixels.
- Label each pixel in picture as belonging to foreground or background.
- $a_i \geq 0$ 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 (for (i,j) in E).

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.

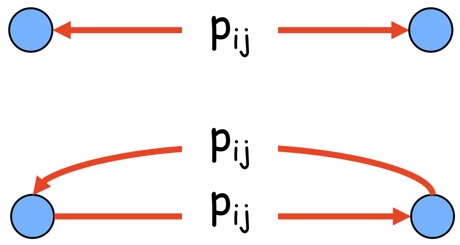
Turn into minimization problem.

- 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.

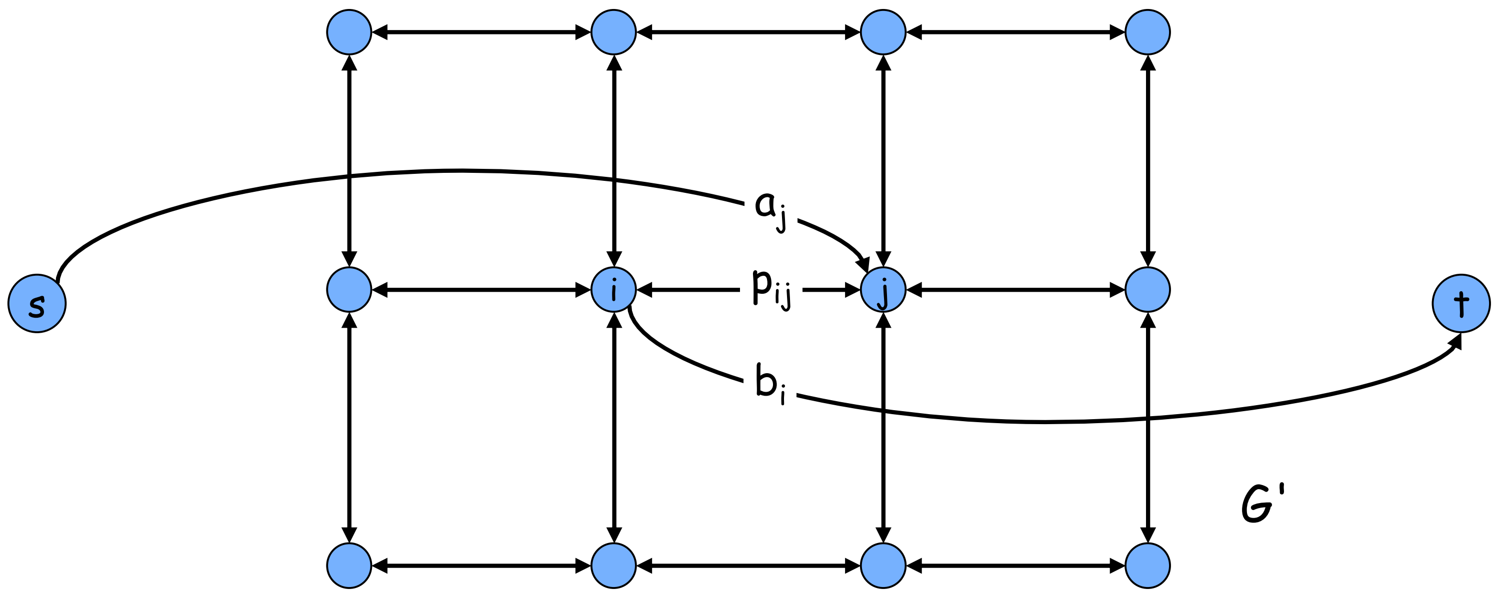


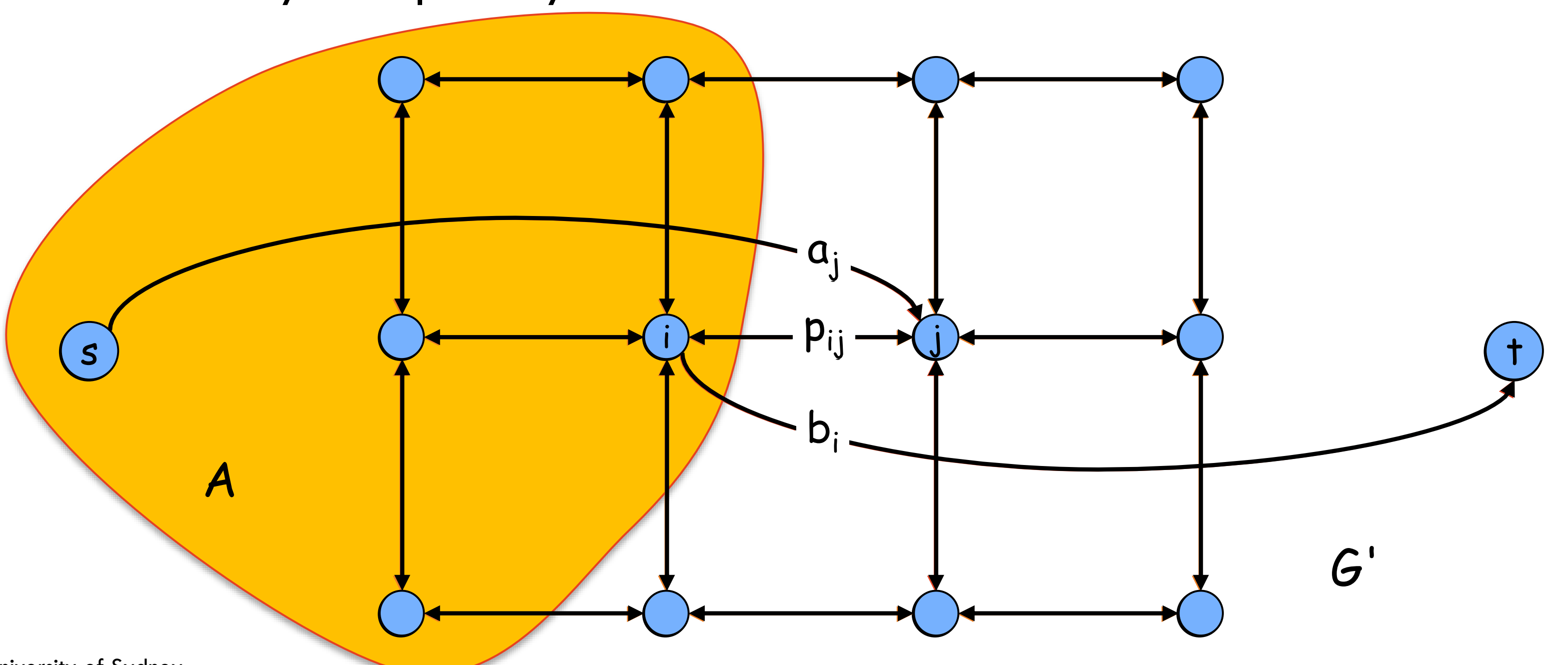
Image Segmentation

- 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.



7.12 Baseball Elimination

Team i	Wins w_i	Losses l_i	To play r_i	Against = r_{ij}			
				Atl	Phi	NY	Mon
Atlanta	83	71	8	-	1	6	1
Philly	80	79	3	1	-	0	2
New York	78	78	6	6	0	-	0
Montreal	77	82	3	1	2	0	-

Which teams have a chance of finishing the season with most wins?

- Montreal eliminated since it can finish with at most 80 wins, but Atlanta already has 83.
- $w_i + r_i < w_j \Rightarrow$ team i eliminated.
- Sufficient, but not necessary!

Baseball Elimination

Team i	Wins w_i	Losses l_i	To play r_i	Against = r_{ij}			
				Atl	Phi	NY	Mon
Atlanta	83	71	8	-	1	6	1
Philly	80	79	3	1	-	0	2
New York	78	78	6	6	0	-	0
Montreal	77	82	3	1	2	0	-

- Which teams have a chance of finishing the season with most wins?
  - Philly can win 83, but still eliminated . . .
  - If Atlanta loses a game, then some other team wins one.

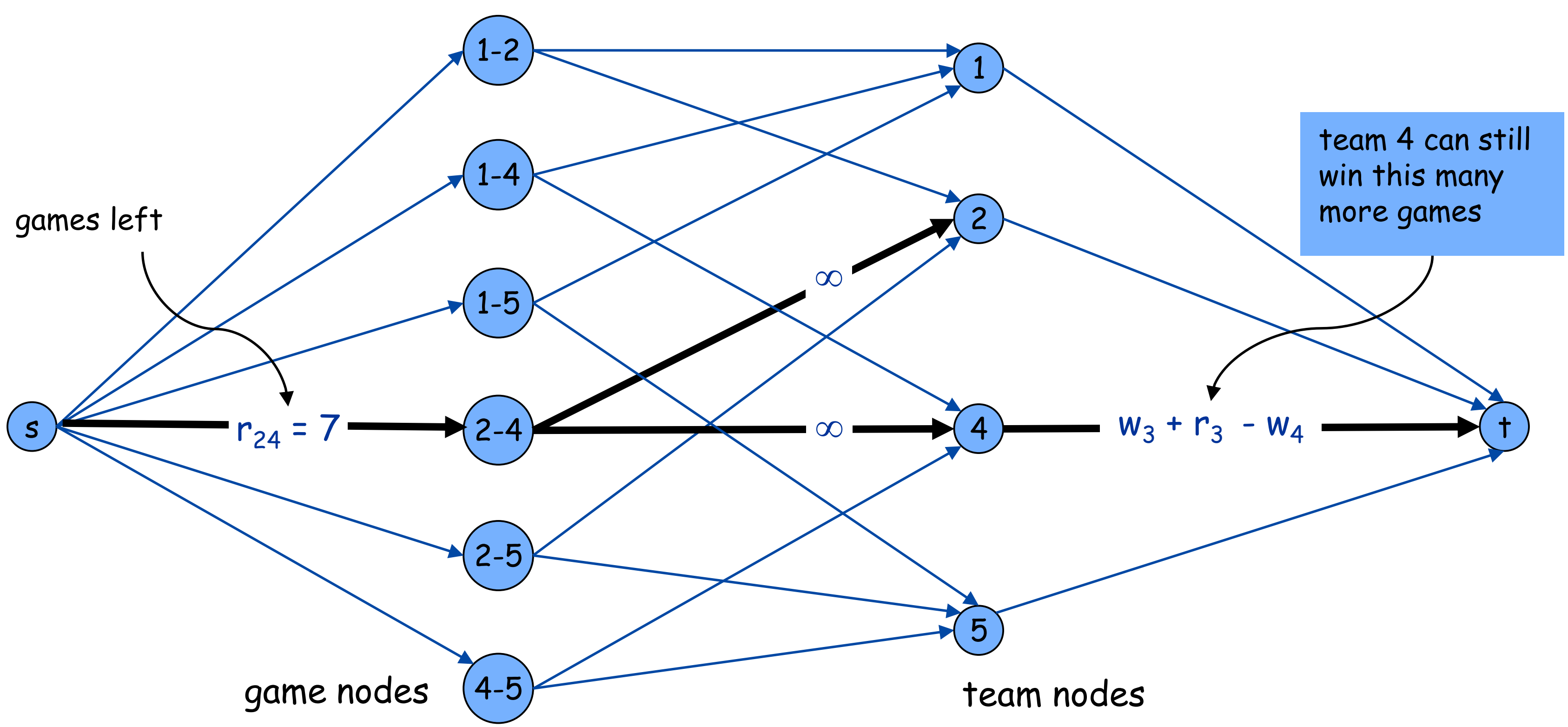
- Remark: Answer depends not just on how many games already won and left to play, but also on whom they're against.

Baseball Elimination

- Baseball elimination problem.
  - Set of teams S.
  - Distinguished team $s \in S$.
  - Team x has won w_x games already.
  - Teams x and y play each other r_{xy} additional times.
  - Is there any outcome of the remaining games in which team s finishes with the most (or tied for the most) wins?

Baseball Elimination: Max Flow Formulation

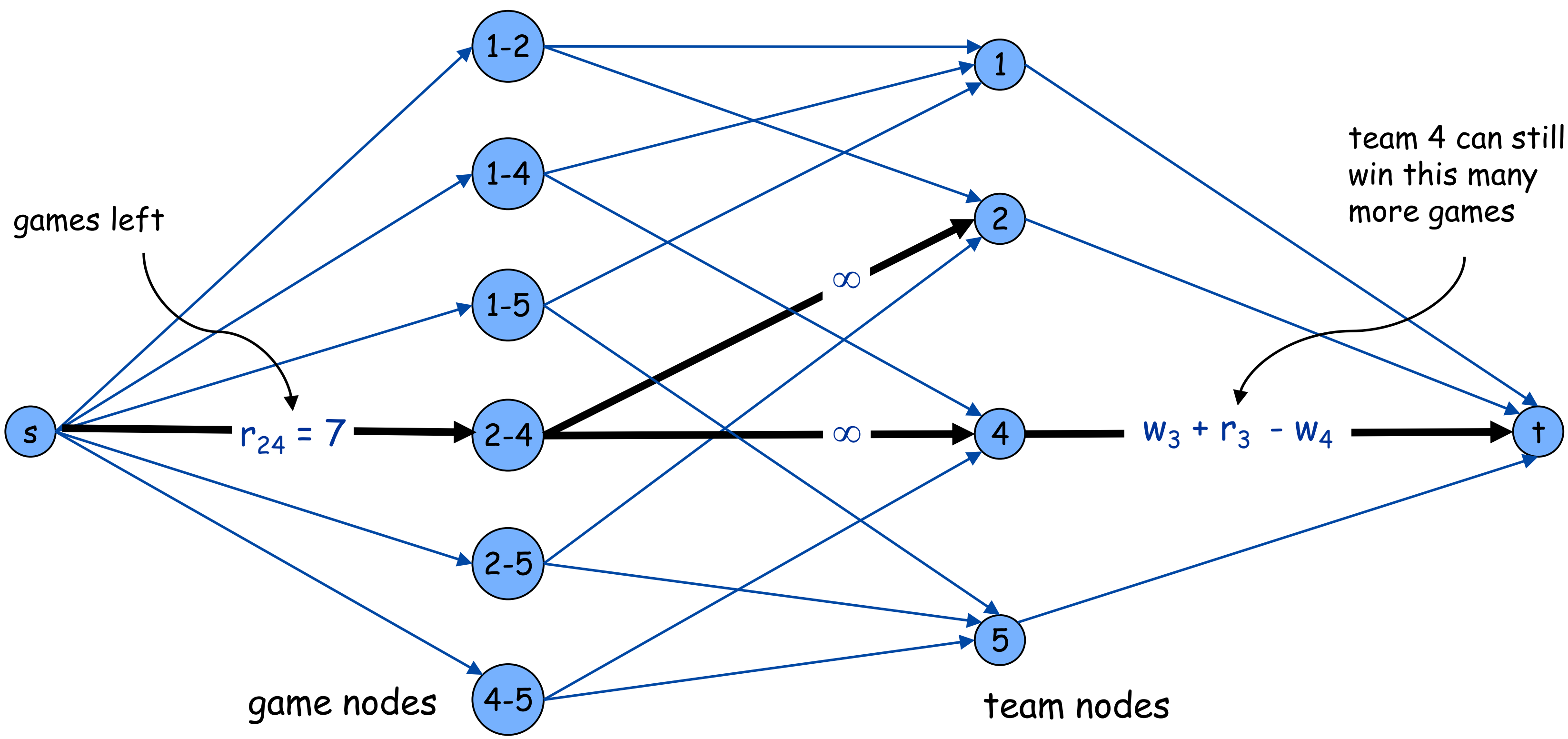
- Can team 3 finish with most wins?
  - Assume team 3 wins all remaining games $\Rightarrow$ $w_3 + r_3$ wins.
  - Distribute remaining games so that all teams have $\leq$ $w_3 + r_3$ wins.



Baseball Elimination: Max Flow Formulation

Theorem. Team 3 is eliminated iff max flow strictly less than the total number of games left.

- Integrality theorem $\Rightarrow$ each remaining game between x and y added to number of wins for team x or team y.
- Capacity on (x, t) edges ensure no team wins too many games.



Applications

- Bipartite matching
- Perfect matching
- Disjoint paths
- Network connectivity
- Circulation problems
- Image segmentation
- Baseball elimination
- Project selection