Applications of Network Flow:

Solving Problems by Reduction to Network Flows

Today: Two (Fun) Max-Flow Min-Cut Applications

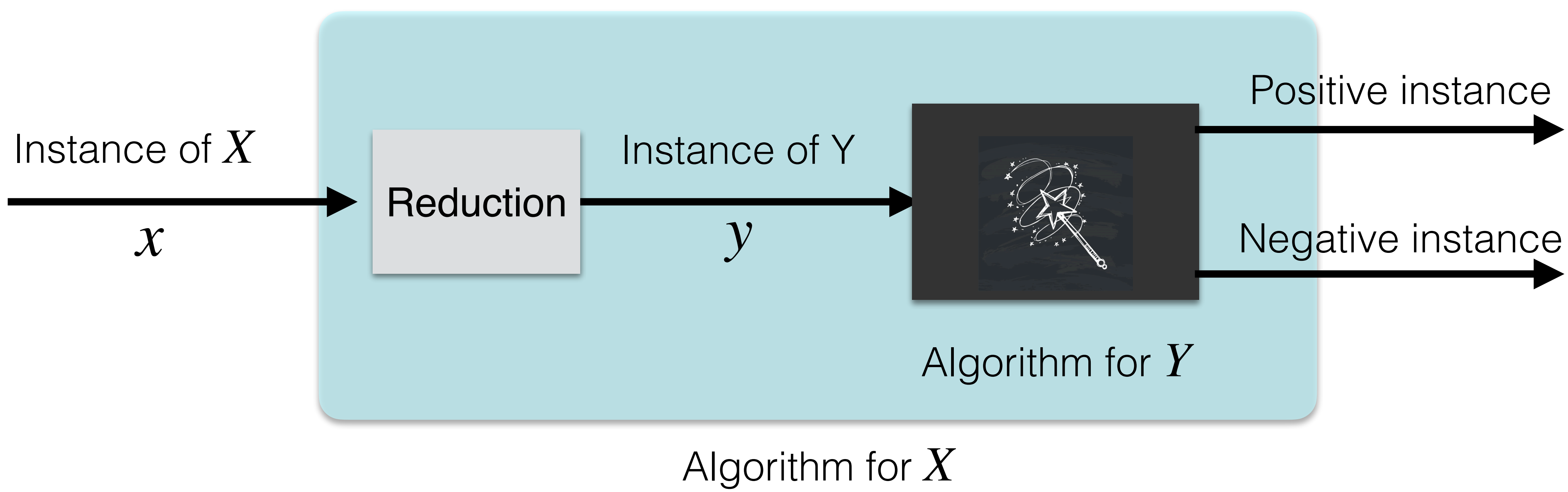
- Bipartite matching
- Baseball elimination

  - We will solve these problems by **reducing** them to a network flow problem
  - Our focus for this class will be on the concept of **problem reductions**

Anatomy of Problem Reductions

At a high level, a problem X reduces to a problem Y if an algorithm for Y can be used to solve X

- **Reduction.** Convert an arbitrary instance x of X to a special instance y of Y such that there is a 1-1 correspondence between them

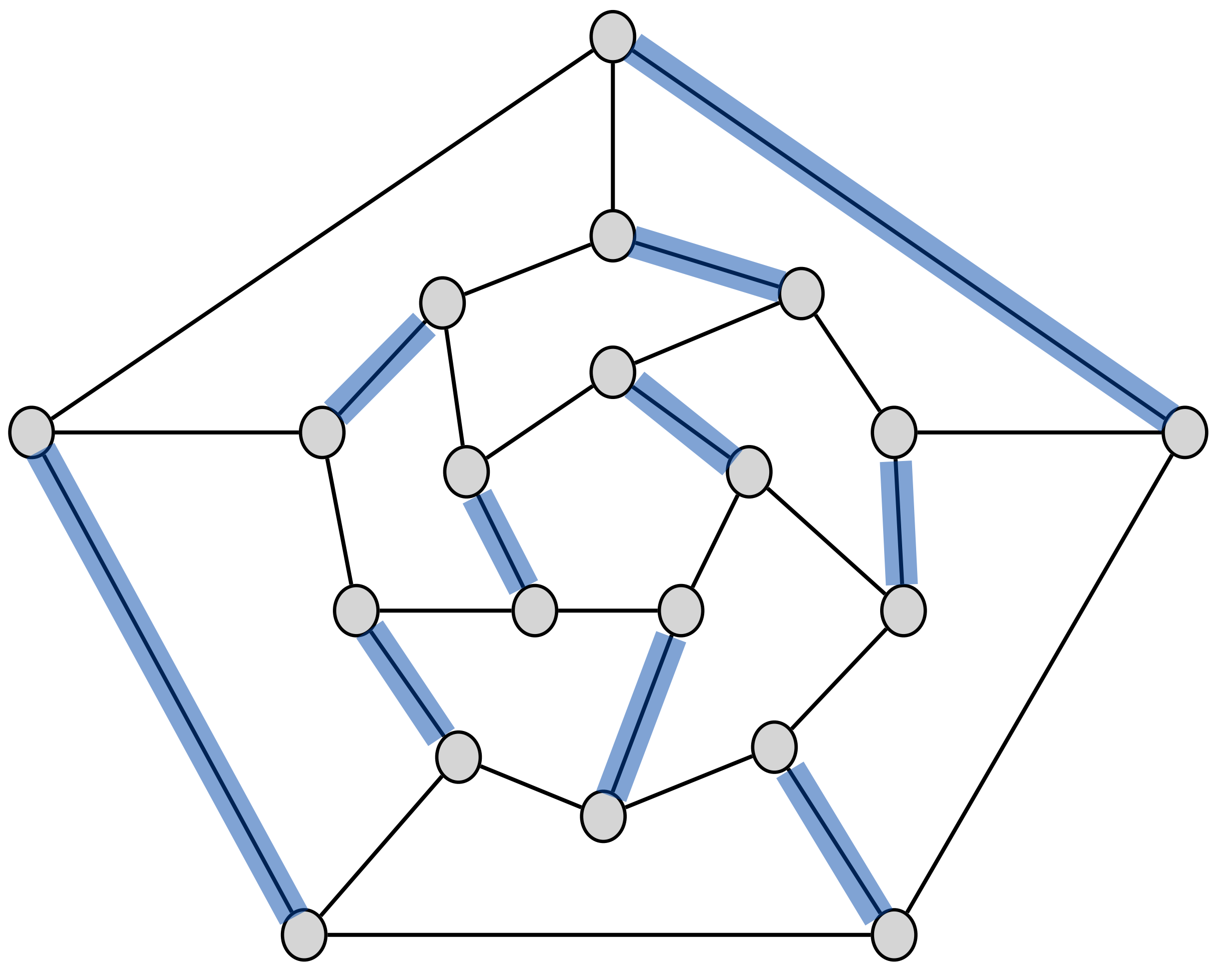


Bipartite Matching

Review: Matching in Graphs

Definition. Given an undirected graph $G = (V, E)$, a matching $M \subseteq E$ of G is a subset of edges such that no two edges in M are incident on the same vertex.

- Said differently, a node appears in at most one edge in M



Review: Matching in Graphs

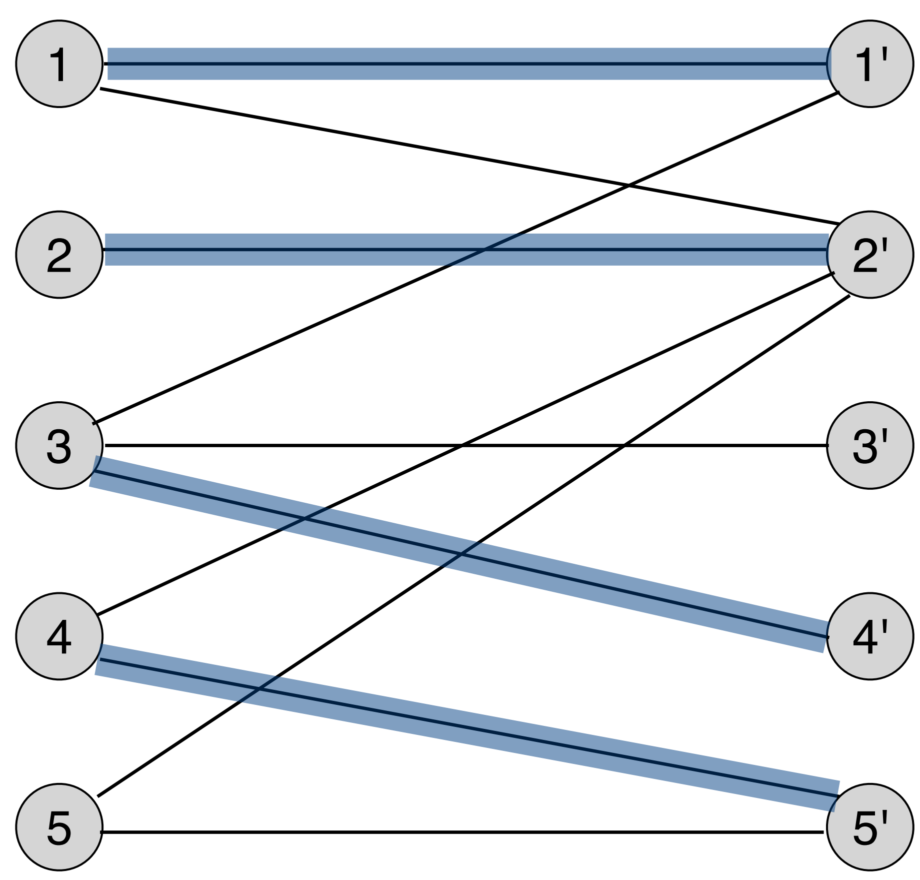
A **perfect matching** matches all nodes in G

- **Max matching problem.** Find a matching of maximum cardinality for a given graph
  - That is, a matching with maximum number of edges
  - **Observation**: If it exists, a perfect matching is maximum!

Review: Bipartite Graphs

A graph is **bipartite** if its vertices can be partitioned into two subsets X, Y such that every edge $e = (u, v)$ connects $u \in X$ and $v \in Y$

- **Bipartite matching problem.** Given a bipartite graph $G = (X \cup Y, E)$ find a maximum matching.



Bipartite Matching Examples

Can be used to model many assignment problems, e.g.:

- A is a set of jobs, B as a set of machines
- Edge (a_i, b_j) indicates where machine b_j is able to process job a_i
- Perfect matching: a way to assign each job to a machine that can process it, such that each machine is assigned exactly one job

- Assigning customers to stores, students to dorms, etc.
- **Note.** This is a different problem than the one we studied for Gale-Shapely matching!

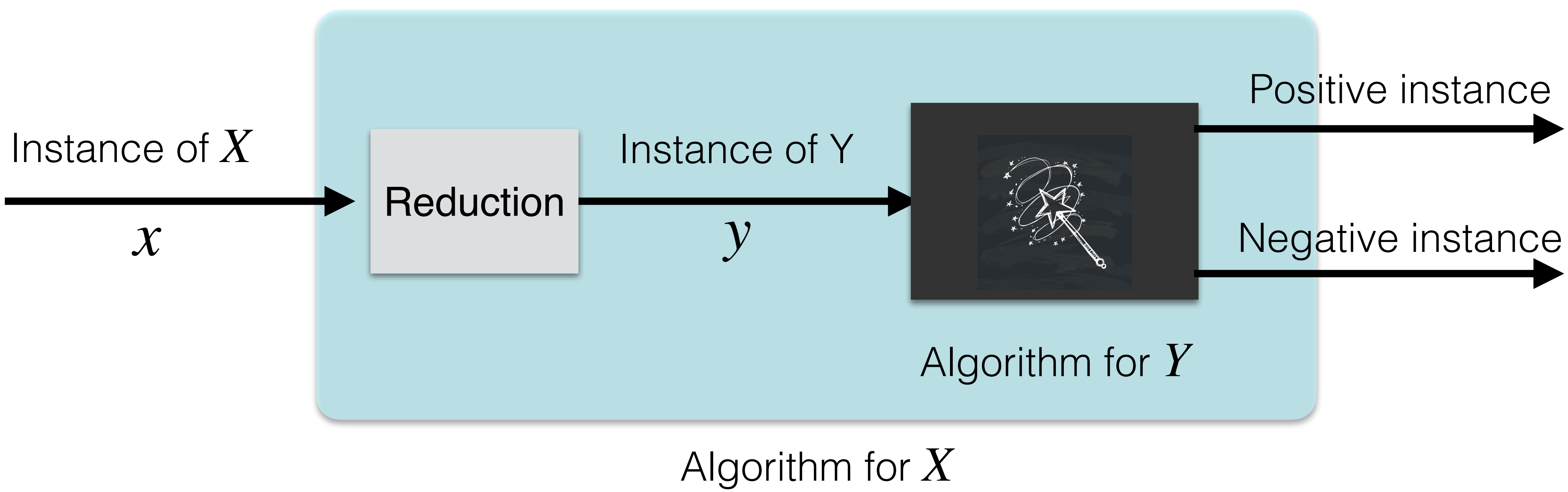
Maximum & Perfect Matchings

Finding the **largest matching on a bipartite graph** doesn't seem like a network flow problem: we must turn it into one!

- **Special case**: Find a perfect matching in G if it exists
    - What conditions do we need for perfect matching?
        - Certainly need $|A| = |B|$
        - What are the necessary and sufficient conditions?
    - Will use network flow to get us there!

Reduction to Max Flow

- **Given:** arbitrary instance x of bipartite matching problem (X): A, B and edges E between A and B
- **Goal:** Create a *special* instance y of a max-flow problem (Y): flow network: $G(V, E, c)$, source s, sink $t \in V$ s.t.
  - **1-1 correspondence.** There exists a matching of size k iff there is a flow of value k



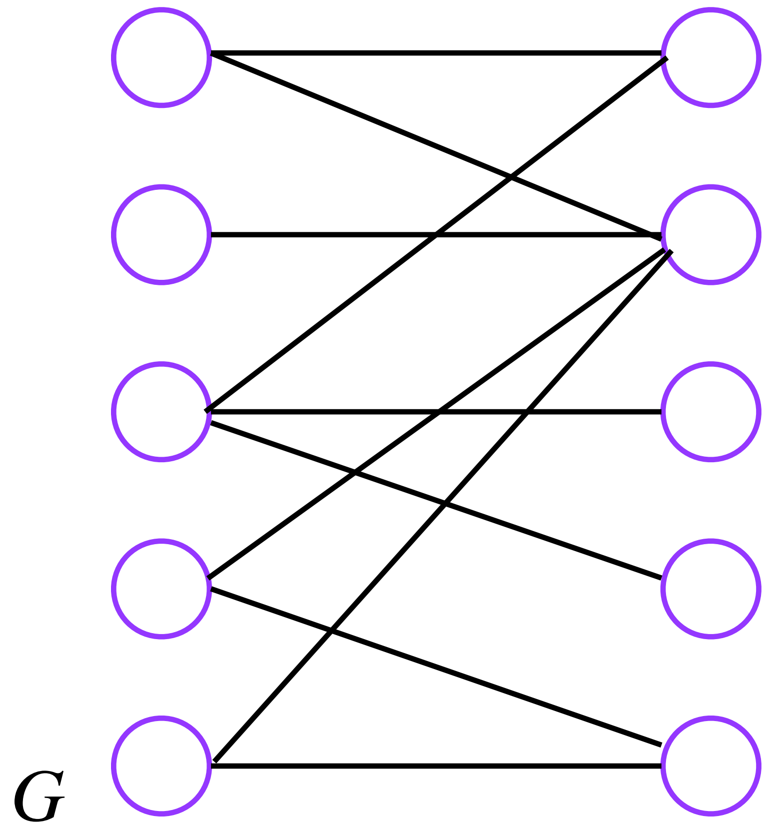
Reduction to Max Flow

Idea: Let's try to construct a flow network where $v(f) = k$ means we have a matching of size k.

- Problems abound! Our bipartite graph, G, is:
  - Sourceless and sinkless.
    - We'll need an s and a t
  - Undirected.
    - How should we fix this? Should we add edges? Convert existing edges to directed edges? Both?
  - Unweighted.
    - G has no edge capacities.
      - We need to add capacities s.t. $v(f) = k$ when we have a matching of size k.

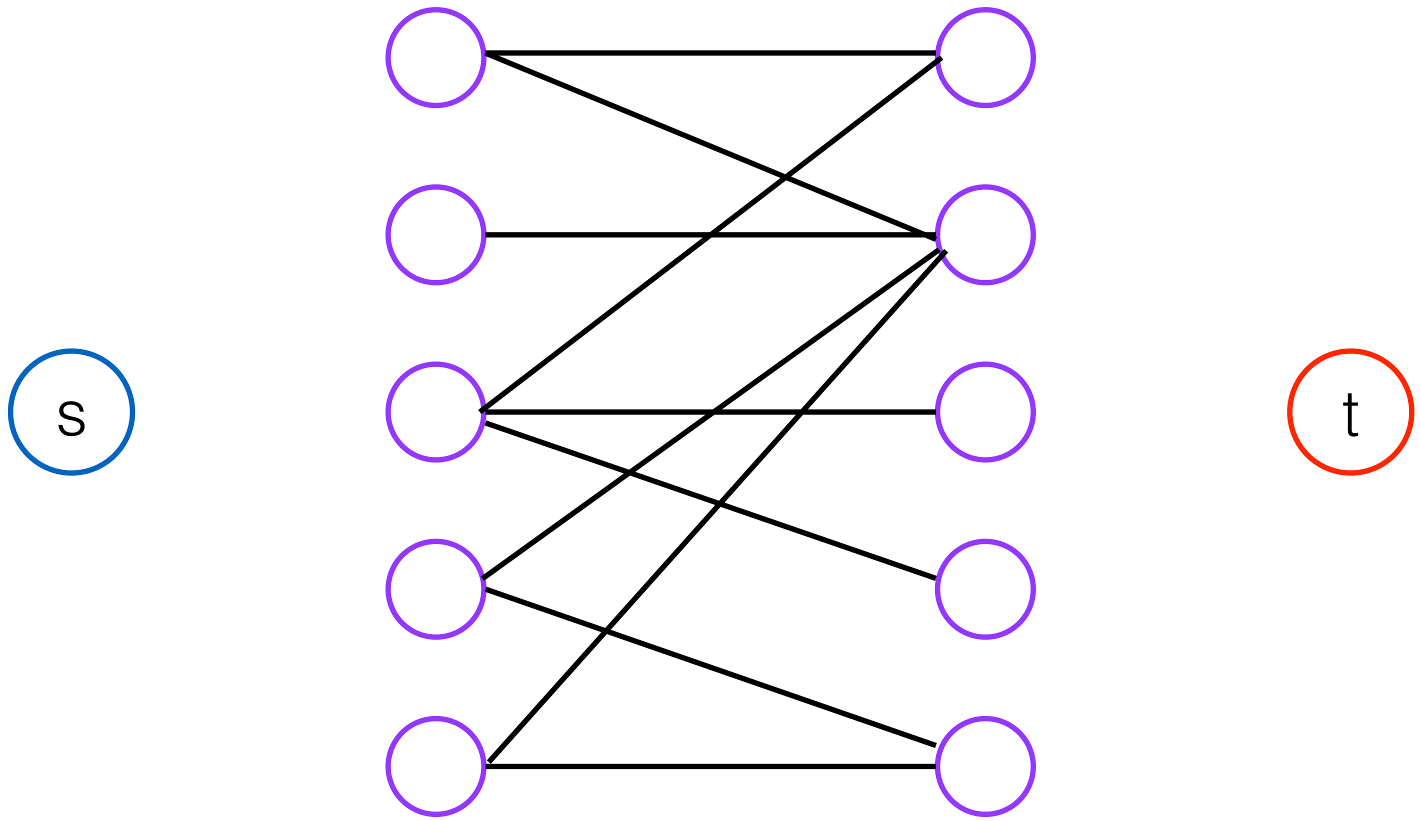
Reduction to Max Flow

- Our bipartite graph, G, is sourceless and sinkless.
  - It isn't clear how to "select" an s and a t among the nodes in G, so let's add new source/sink nodes



Reduction to Max Flow

- Our bipartite graph, G, is sourceless and sinkless.
    - It isn't clear how to "select" an s and a t among the nodes in G, so let's add new source/sink nodes



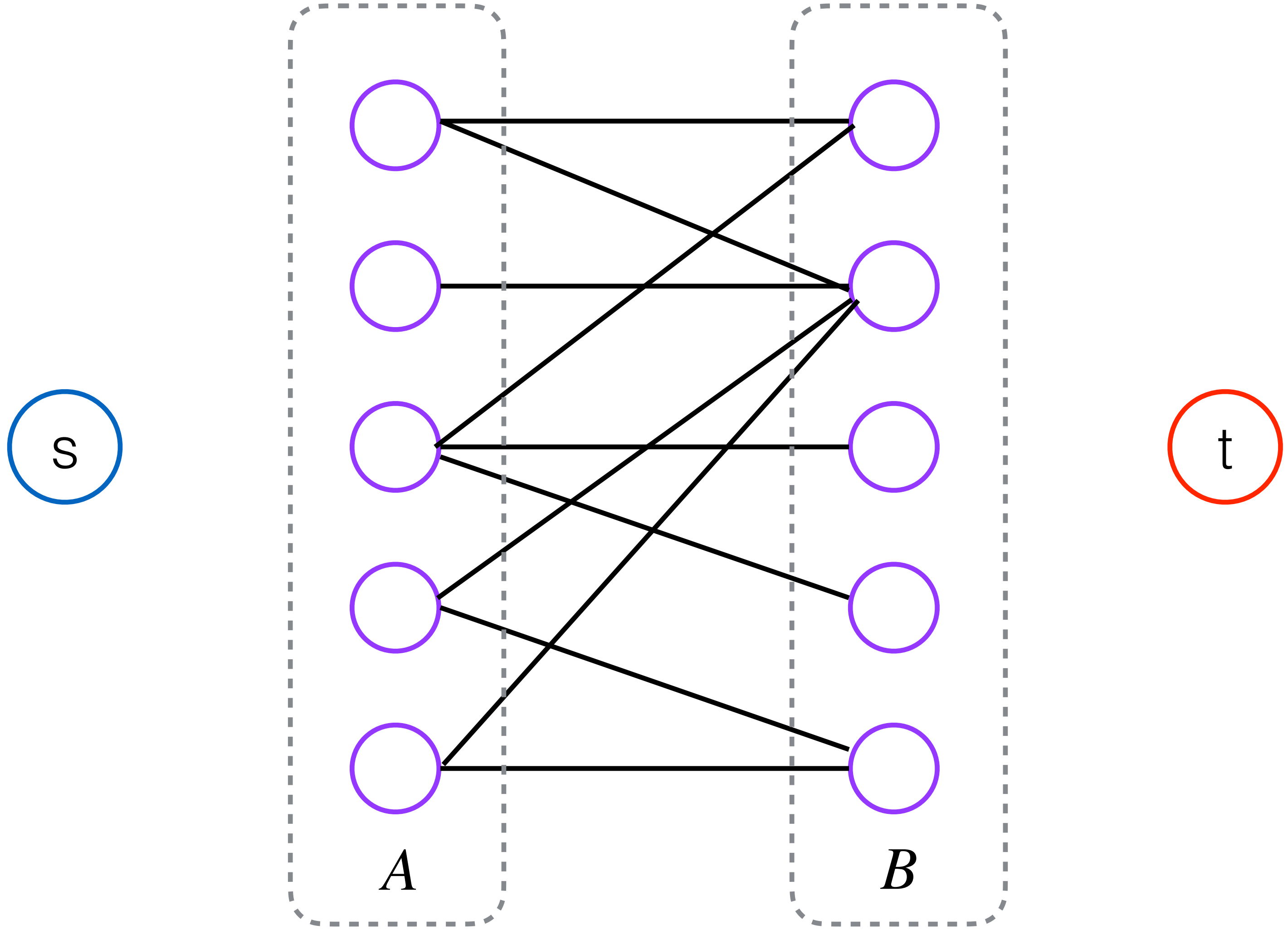
- How do we connect s and t to the nodes in G? Considerations:
    - Max flow = min cut
    - We want $v(f) = k$ when we have a matching of size k.

Reduction to Max Flow

Each vertex can be in at most one match

Observations:

- The size of a maximum matching is $min(|A|, |B|)$
- If max flow = min cut, two intuitive bottlenecks are $f_{out}(s)$ and $f_{in}(t)$

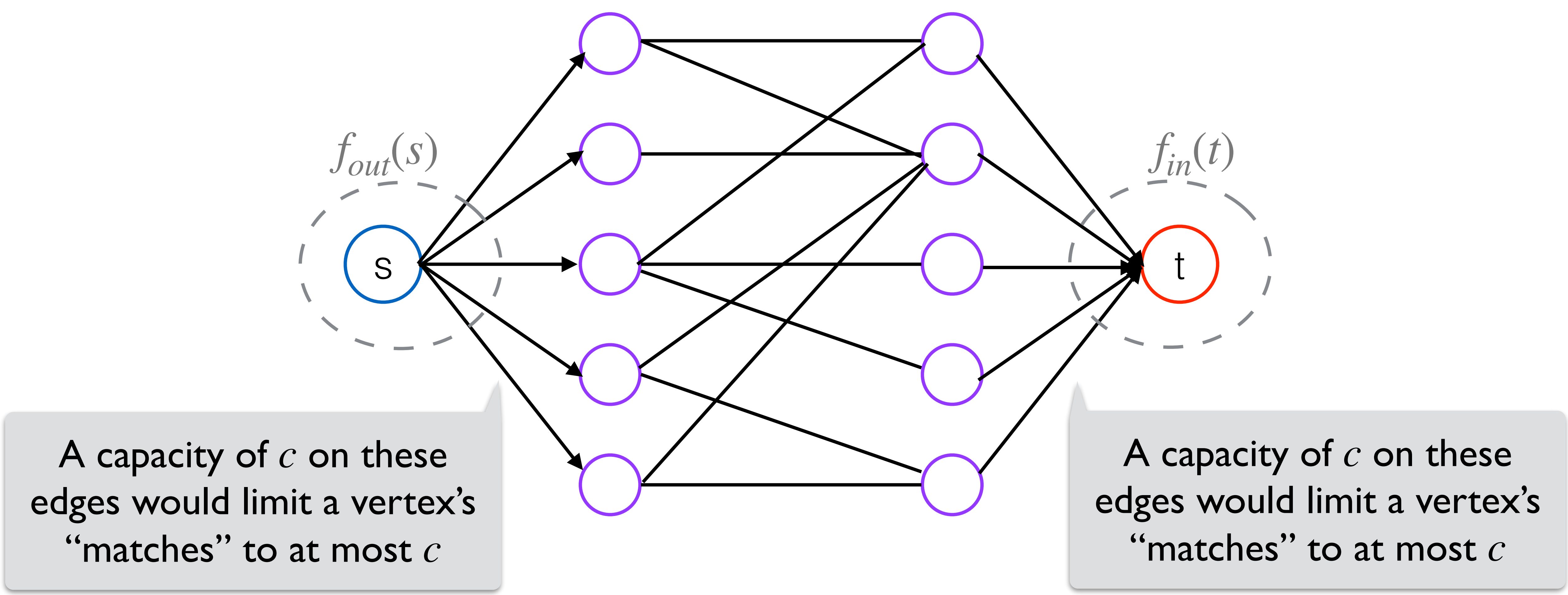


- If we add edges from s to each node in A, and from each node in B to t, flow across those edges could correspond to the vertex being matched

Reduction to Max Flow

Observations:

- The size of a maximum matching is $min(|A|, |B|)$
- If max flow = min cut, two intuitive bottlenecks are $f_{out}(s)$ and $f_{in}(t)$

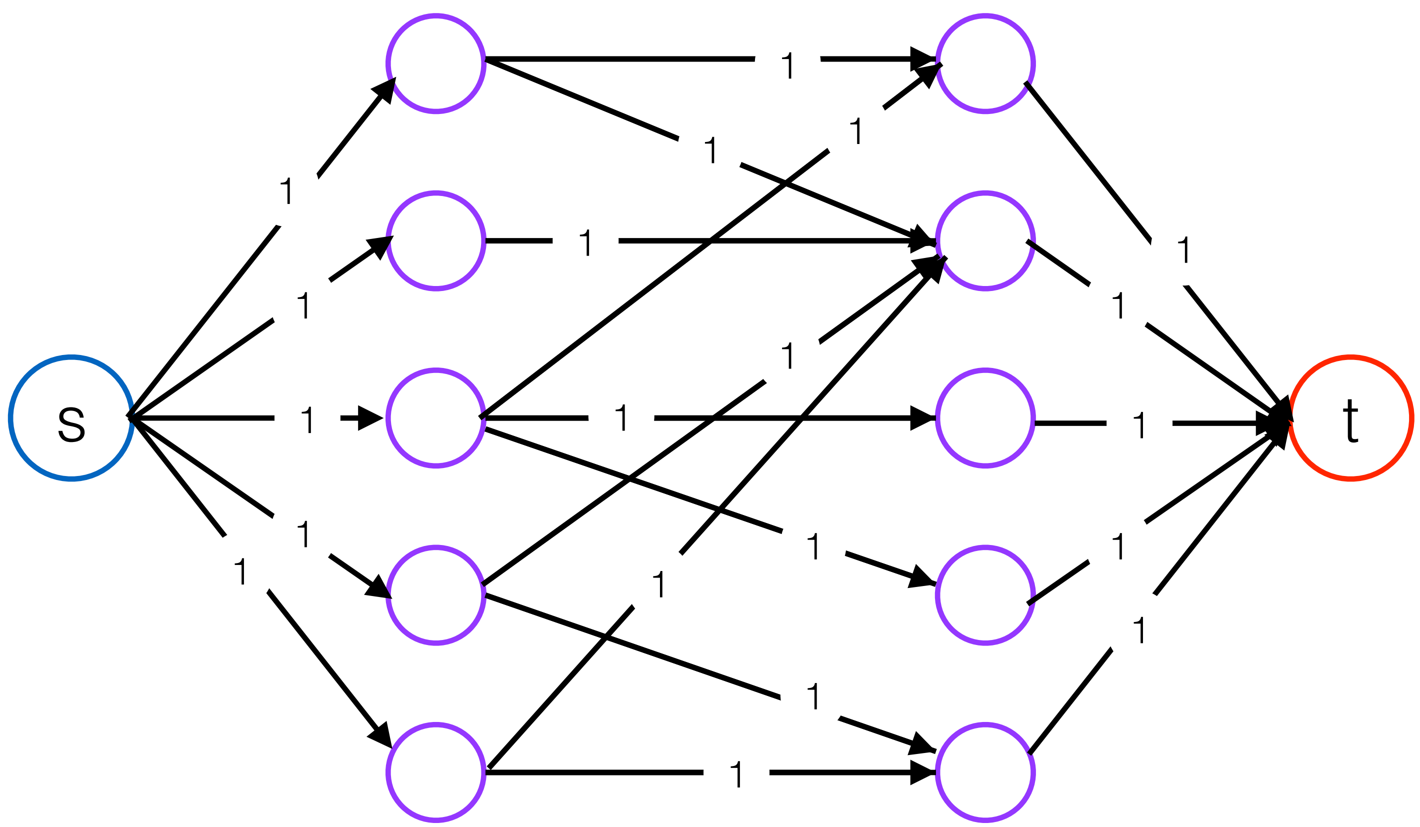


- If we add edges from s to each node in A, and from each node in B to t, flow across those edges could correspond to the vertex being matched

Reduction to Max Flow

Observations:

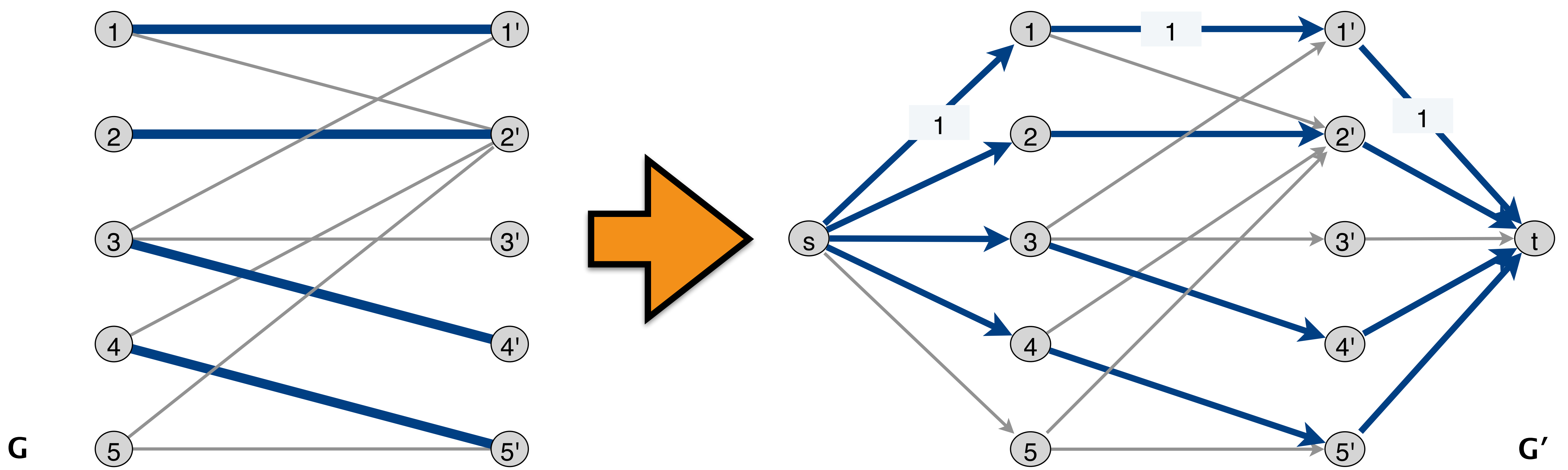
- If we orient the undirected edges to originate in "A" vertices and terminate in "B" vertices, flow can travel from source to sink



- We need to limit vertex matches to at most 1 match
- Adding a capacity of 1 to all directed edges completes our reduction

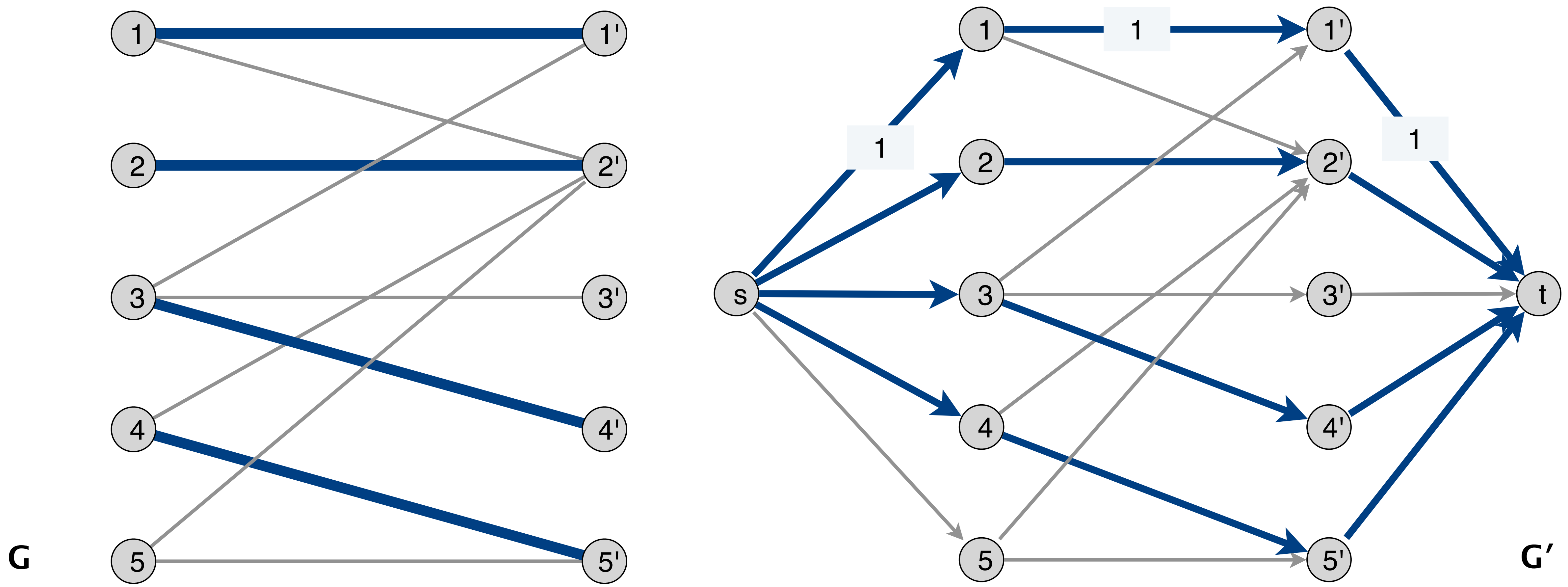
Reduction Summary

- Create a new directed graph $G' = (A \cup B \cup \{s, t\}, E', c)$
- Add edge $s \rightarrow a$ to E' for all nodes $a \in A$
- Add edge $b \rightarrow t$ to E' for all nodes $b \in B$
- Direct edge $a \rightarrow b$ in E' if $(a, b) \in E$
- Set capacity of all edges in E' to 1



Correctness of Reduction

- **Claim ($\Rightarrow$).**
  If the bipartite graph (A, B, E) has matching M of size k then flow-network G' has an integral flow of value k.



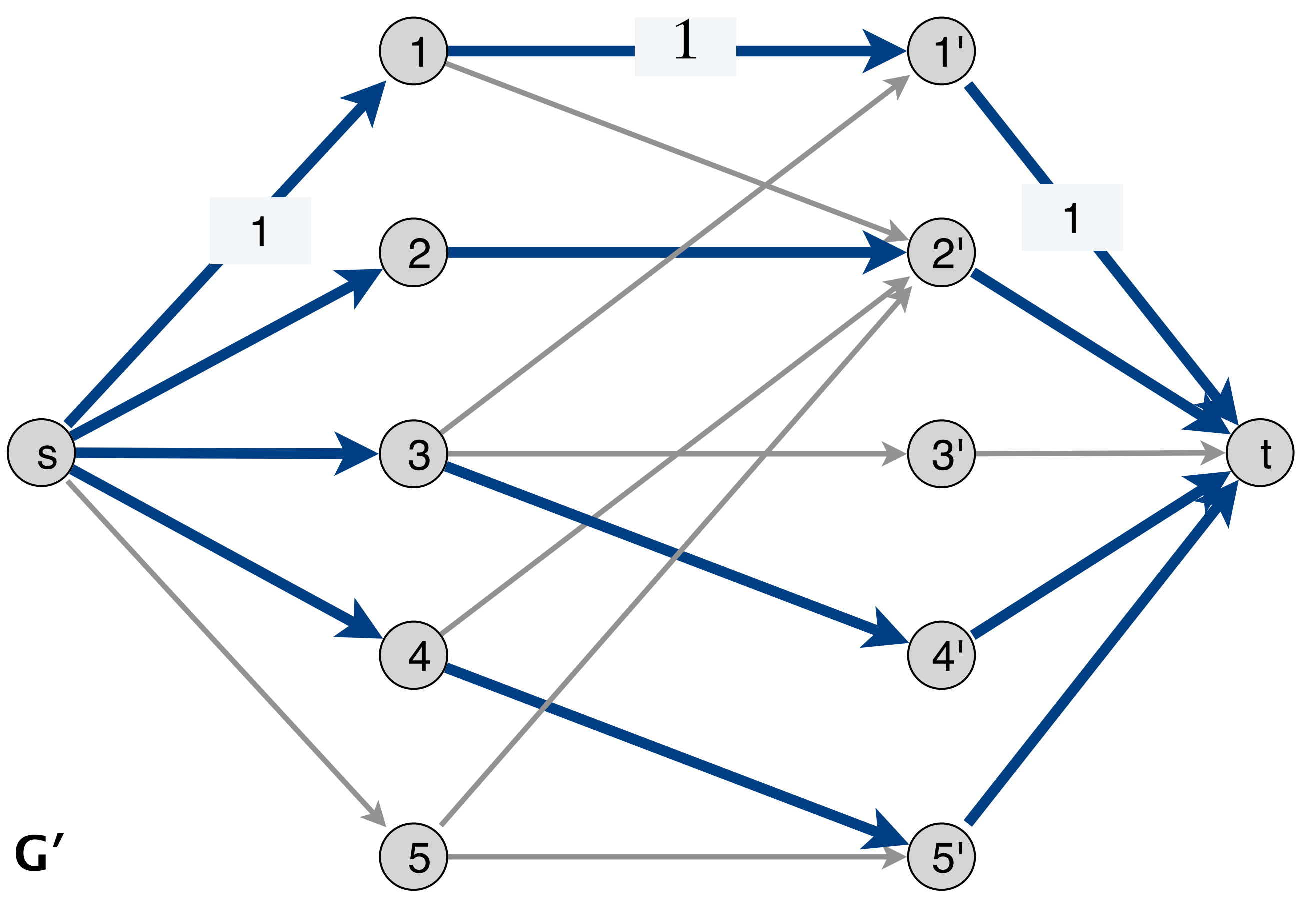
Correctness of Reduction

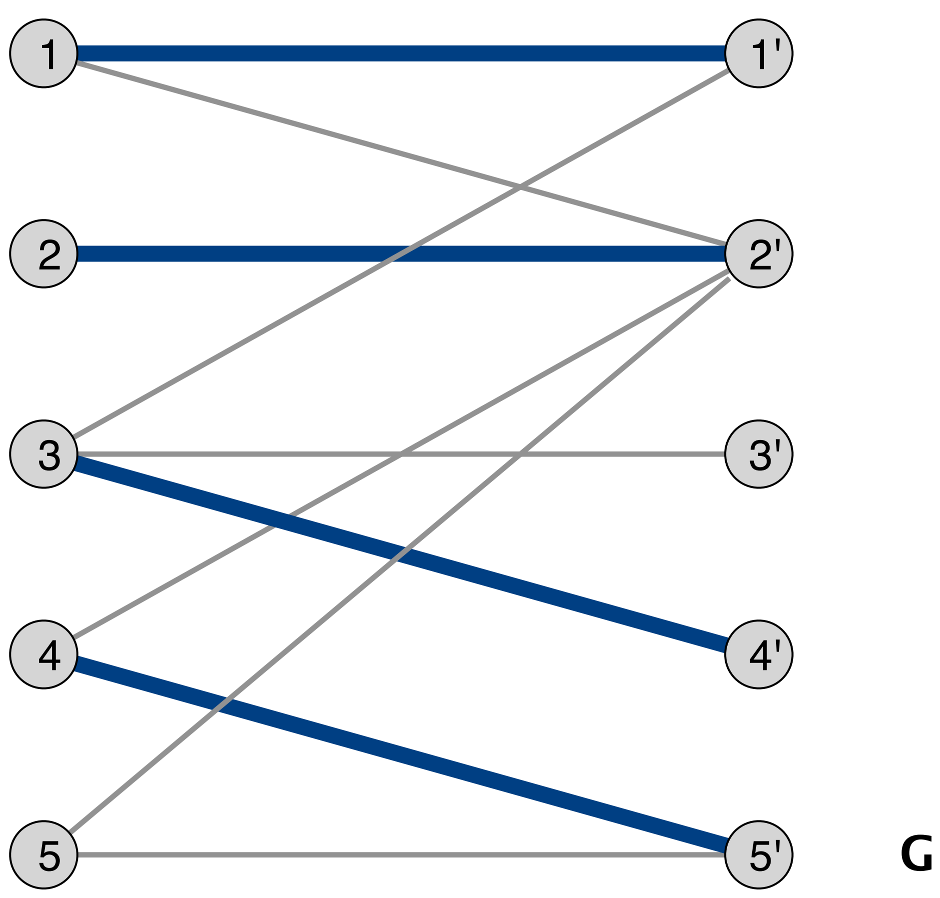
- **Claim** ($\Rightarrow$).
  If the bipartite graph (A, B, E) has matching M of size k then flow-network G' has an integral flow of value k.

- **Proof scketch** (Complete proof in textbook).
  - For every edge $e = (a, b) \in M$, let f be the flow resulting from sending 1 unit of flow along the path $s \to a \to b \to t$
  - f is a feasible flow (satisfies capacity and conservation) and integral
  - $v(f) = k$

Correctness of Reduction

- **Claim ($\Leftarrow$).**
  If flow-network G' has an integral flow of value k, then the bipartite graph (A, B, E) has matching M of size k.





Correctness of Reduction

- **Claim** $(\Leftarrow)$**.**
  If flow-network G' has an integral flow of value k, then the bipartite graph (A, B, E) has matching M of size k.

- **Proof**.
  - Let $M =$ set of edges from A to B with $f(e) = 1$.
  - No two edges in M share a vertex, why?
  - $|M| = k$
    - $v(f) = f_{out}(S) - f_{in}(S)$ for any $(S, V - S)$ cut
    - Let $S = A \cup \{s\}$

Summary & Running Time

- Proved matching of size k iff flow of value k
- Thus, max-flow iff max matching
- Running time of algorithm overall:
    - Running time of reduction + running time of solving the flow problem (flow alg. dominates)
- What is running time of Ford–Fulkerson algorithm for a flow network with all unit capacities?
    - $O(nm)$
- Overall running time of finding max-cardinality bipartite matching: $O(nm)$

Baseball Elimination

The Baseball Elimination Problem

You are given the wins, and losses, and remaining schedule for all teams in a league/division. Which teams have been mathematically eliminated from contention (i.e., they cannot possibly come in first or tie for first place)?

Team	Wins	Losses	Games Left	Angels	Athletics	Mariners	Rangers
Angels	83	71	8	-	1	6	1
Athletics	80	79	3	1	-	0	2
Mariners	78	78	6	6	0	-	0
Rangers	77	82	3	1	2	0	-

The Baseball Elimination Problem

You are given the wins, and losses, and remaining schedule for all teams in a league/division. Which teams have been mathematically eliminated from contention (i.e., they cannot possibly come in first or tie for first place)?

Team	Wins	Losses	Games Left	Angels	Athletics	Mariners	Rangers
Angels	83	71	8	-	1	6	1
Athletics	80	79	3	1	-	0	2
Mariners	78	78	6	6	0	-	0
Rangers	77	82	3	1	2	0	-

No! Even if the Rangers' win all remaining games, their win total won't surpass the Angels' current win total.

Can the Rangers possibly come in first place? Why or why not?

The Baseball Elimination Problem

You are given the wins, and losses, and remaining schedule for all teams in a league/division. Which teams have been mathematically eliminated from contention (i.e., they cannot possibly come in first or tie for first place)?

Team	Wins	Losses	Games Left	Angels	Athletics	Mariners	Rangers
Angels	83	71	8	-	1	6	1
Athletics	80	79	3	1	-	0	2
Mariners	78	78	6	6	0	-	0
Rangers	77	82	3	1	2	0	-

Can the Athetlics possibly come in first place? Why or why not?

No! If the Angels lose all remaining games (as needed), 6 of them are losses to the Mariners. The Mariners will leapfrog the Athletics and come in first!

A More Principled Approach

What we need is a way to prove that a team is eliminated. Let's try to reduce this problem to a max flow problem…

Team	Wins	Losses	Games Left	Angels	Athletics	Mariners	Rangers
Angels	83	71	8	-	1	6	1
Athletics	80	79	3	1	-	0	2
Mariners	78	78	6	6	0	-	0
Rangers	77	82	3	1	2	0	-

Let’s Leverage the Average

Team	Wins	Losses	Games Left	Angels	Athletics	Mariners	Rangers
Angels	83	71	8	-	1	6	1
Athletics	80	79	3	1	-	0	2
Mariners	78	78	6	6	0	-	0
Rangers	77	82	3	1	2	0	-

What is the maximum number of games the Athletics can win this season?

83

The Angels and Mariners have how many wins between them?

83+78 = 161

How many remaining games will the Angels/Mariners play against each other?

6

Then how many wins **must** the Angles and Mariners have between them?

161+6 = 167

If two teams have 167 wins between them, then one team **must** have at least how many wins?

$\lceil 167/2 \rceil = 84$

Let’s Leverage the Average

Team	Wins	Losses	Games Left	Angels	Athletics	Mariners	Rangers
Angels	83	71	8	-	1	6	1
Athletics	80	79	3	1	-	0	2
Mariners	78	78	6	6	0	-	0
Rangers	77	82	3	1	2	0	-

What is the maximum number of games the Athletics can win this season?

83

The Angels and Mariners have how many wins between them?

83+78 = 161

How many remaining games will the Angels/Mariners play against each other?

6

Then how many wins **must** the Angles and Marine

161+6 = 167

If two teams have 167 wins between them, then on

how many wins?

$\lceil 167/2 \rceil = 84$

This is a short proof showing that the Athletics cannot possibly come in first

Let's Leverage the Average

In general, if we have a set of k integers that sum to N, the value of *some* integer in that set must be $\geq \lceil N/k \rceil$

- So, for any team $t \in T$, if there is *some* subset of teams $S \subseteq T - \{t\}$ where their (total number of current wins + total number of remaining head-to-head games) $\div$ $|S|$ is more than t's maximum possible wins, t is mathematically eliminated from contention.
  - For the Rangers, one subset is {Angels}
    - 77+3=80 < 83+0=83
  - For the Athletics, {Mariners, Rangers} does not ensure elimination, but {Angels, Mariners} does

Finding the right subset S of teams can serve as a proof that some team t is eliminated from contention

Notation

We want to be able to talk about our problem/constraints, so we'll define some terms we use in our reduction.

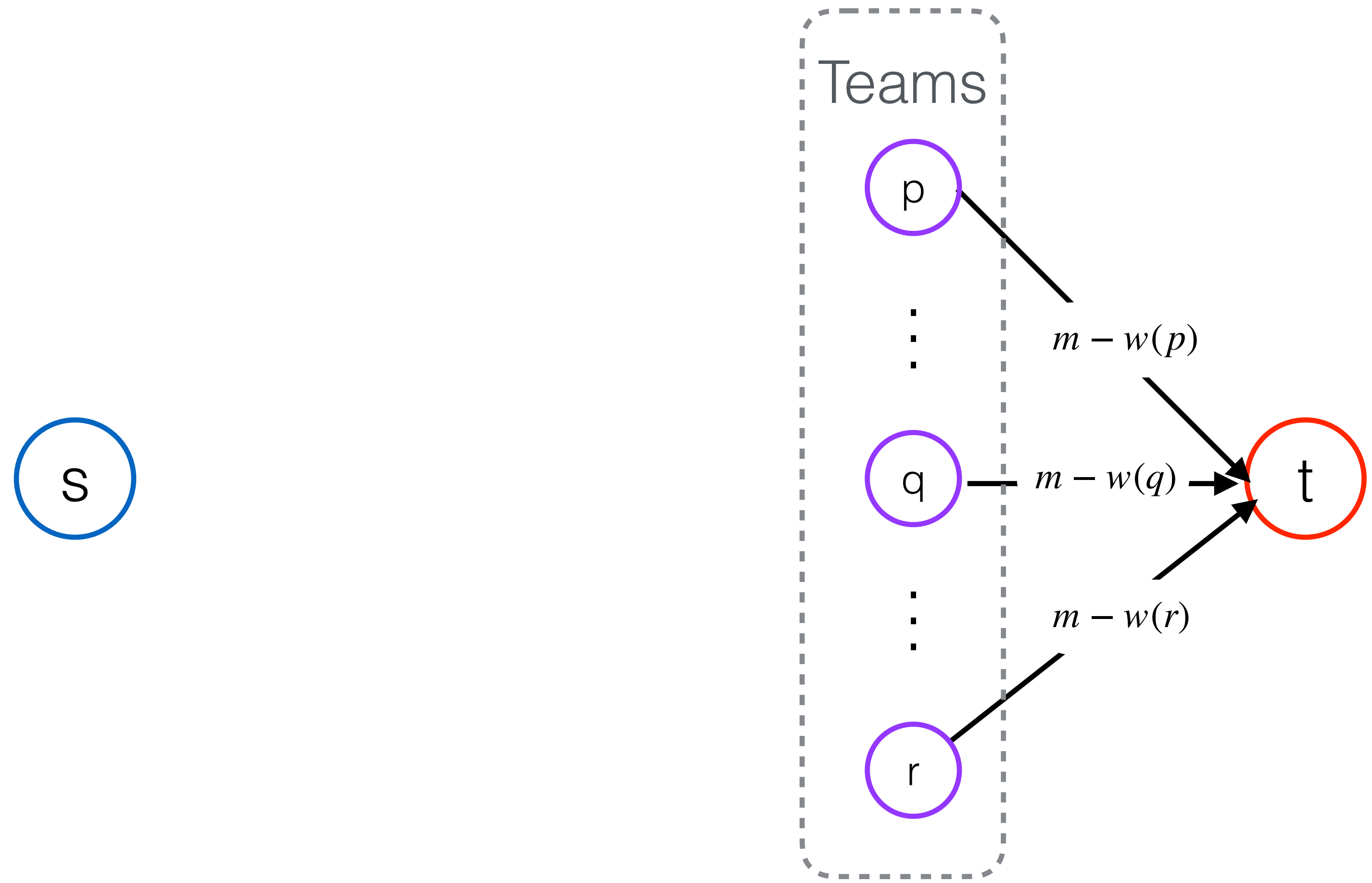
- Let S be a set of teams in some division
  - E.g., $S =$ { Angels, Athletics, Mariners, Rangers }
- If $x \in S$, then let $w(x)$ be the number of wins by team x
  - E.g., if $x \leftarrow$ Angels, then $w(x) = 83$
- If x and y are teams in S, then let $g(x, y)$ denote the number of games remaining between x and y
  - E.g., if $x \leftarrow$ Angels, $y \leftarrow$ Rangers, then $g(x, y) = 1$

Defining the Reduction

We will next build a flow network that is unique to a single team, $x \in S$. It will allow us to answer, for $x \in S$, "Has x been eliminated from contention?"

- Idea: Assume x wins **all** of its remaining games. Denote this number as m. We want to construct a flow network that includes all other teams (i.e., $S' = S - \{x\}$), but each team's victories are constrained by m (no team can win more than m games).
  - For every team $i \in S'$, create a node in the network, and connect it to t.
  - We want to make sure no team i can have more than m wins. A team i already has $w(i)$ wins. What should the capacity be for the edge connecting team i to t?
    - $m - w(i)$

Abstract Flow Network

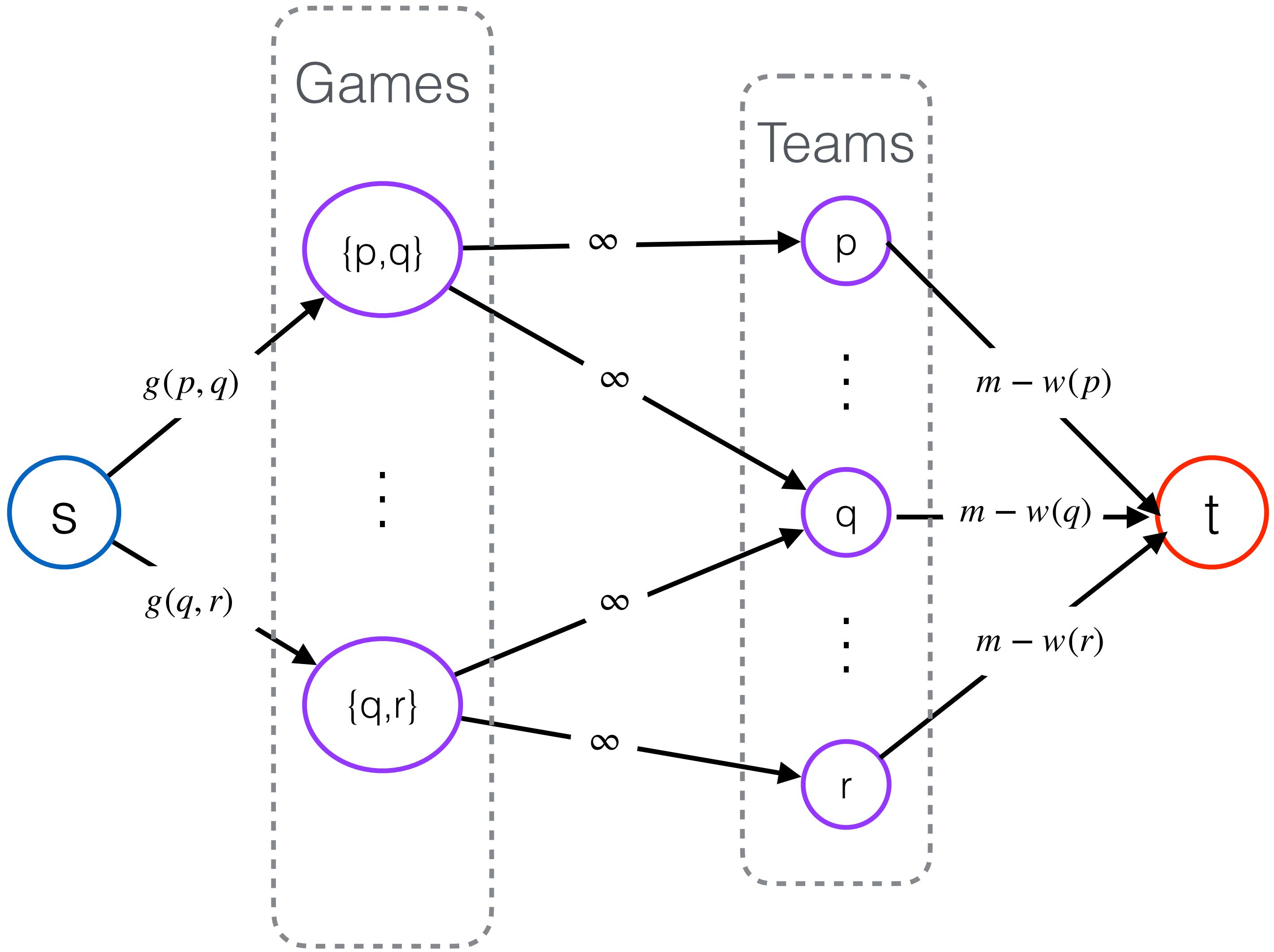


Defining the Reduction

If one unit of flow represents one win, then we need a way to model the genesis of wins.

- Idea: In addition to team nodes, create a games node for the head-to-head games between each pair of teams.
  - The number of games between teams i and j is denoted by $g(i, j)$
    - $g(i, j)$ should be the capacity entering the game node from s
  - Flow leaving a game node represents a win, so each game node must connect to the two teams that are playing
    - The most wins a team could get is $g(i, j)$, but the conservation of flow self-limits these edges (i.e., game node to team node)
    - Using ∞ simplifies later analysis, so let's use that as the capacity

Abstract Flow Network



Interpreting the Network

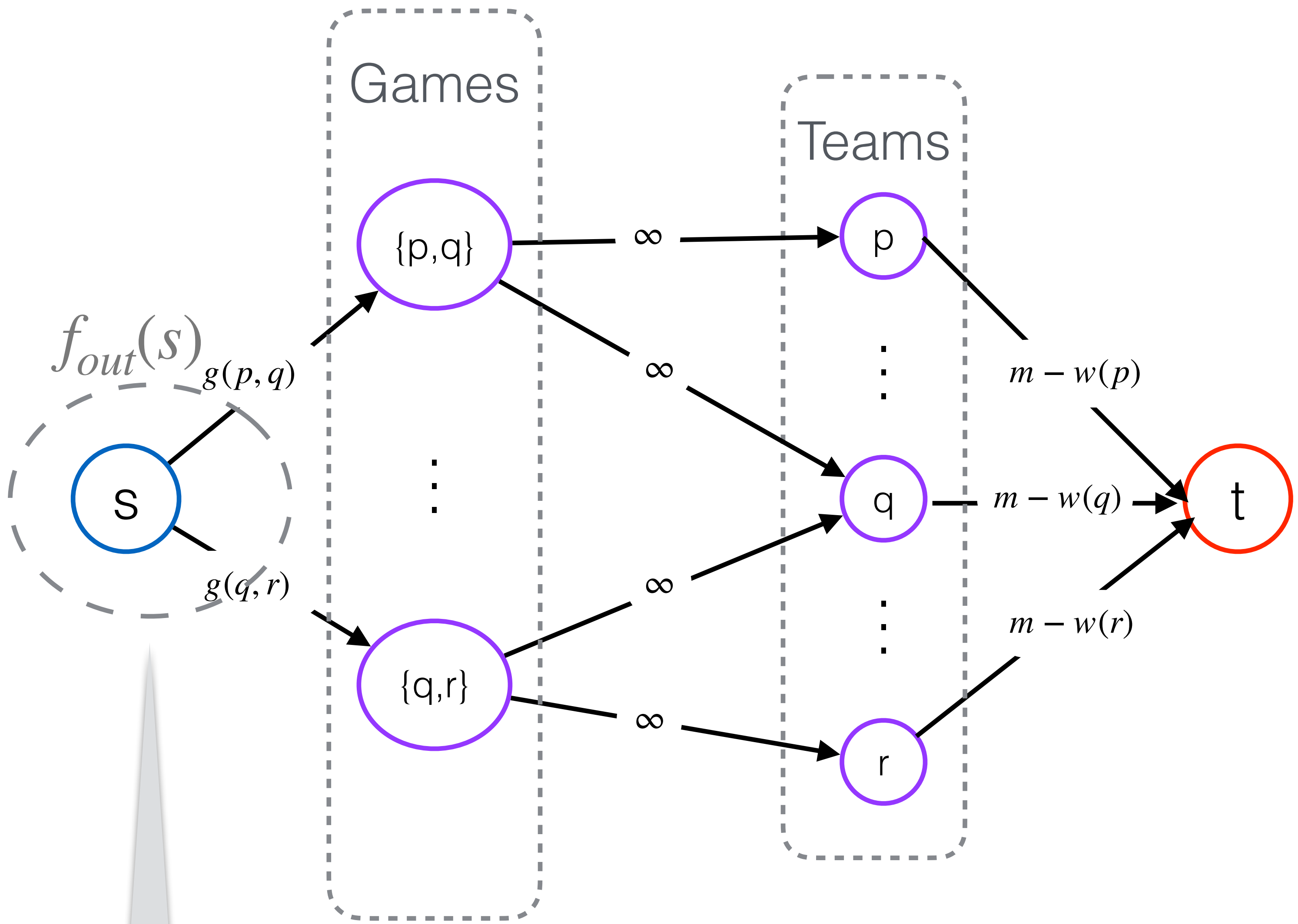
Now that we've built our flow network, how can we use it to solve the problem? Let's think about what we've constructed…

- We've constrained the number of games that each team can win by assigning capacities $m - w(i)$ to the edges leaving each team.
- We've represented each game yet to be played (by a team other than x). These games must be won by someone.
  - Let g denote the number of games yet to be played
  - Let g^* denote the max flow on our network.
    - What does it mean when $g = g^*$?
    - What does it mean when $g^* < g$?

Each game is assigned as a win to some team—and it falls within the x-constrained limit!

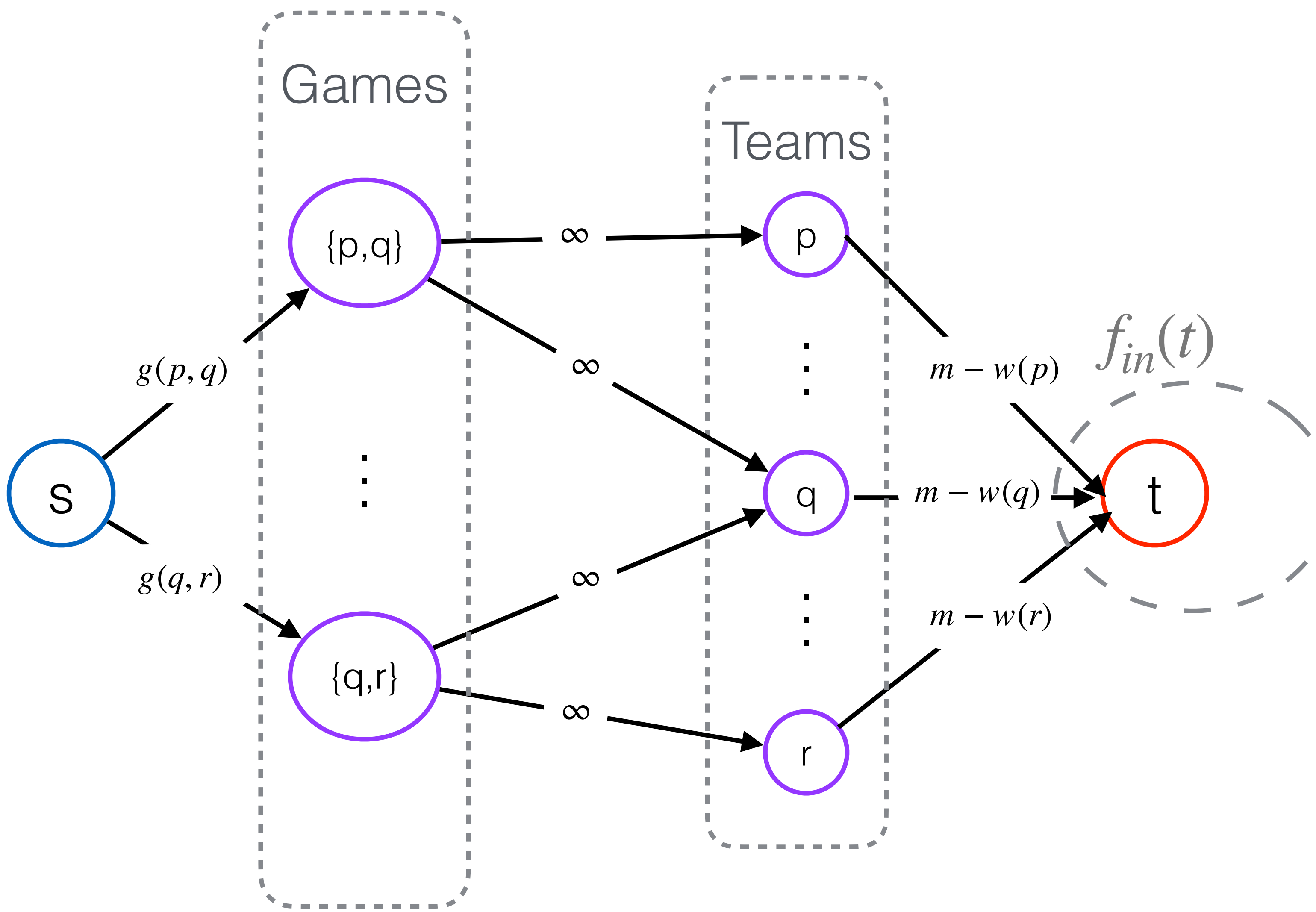
There weren't enough "allowed" wins to meet the remaining games played. Thus, team x is eliminated!

Abstract Flow Network



If $f_{out}(s) = c(\{s\}, V - \{s\})$, then all games were successfully assigned as wins to some team, subject to the constraints needed for team x to have a chance.

Abstract Flow Network



If $f_{in}(t) < c(\{s\}, V - \{s\})$, then there were games that could not be successfully assigned as wins to some team, subject to the constraints needed for team x to have a chance. In other words, there were more games played than teams were allowed to win.

Acknowledgments

- Some of the material in these slides are taken from
  - Kleinberg Tardos Slides by Kevin Wayne (https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/04GreedyAlgorithmsI.pdf)
  - Jeff Erickson's Algorithms Book (http://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf)