Applications of Network Flow:

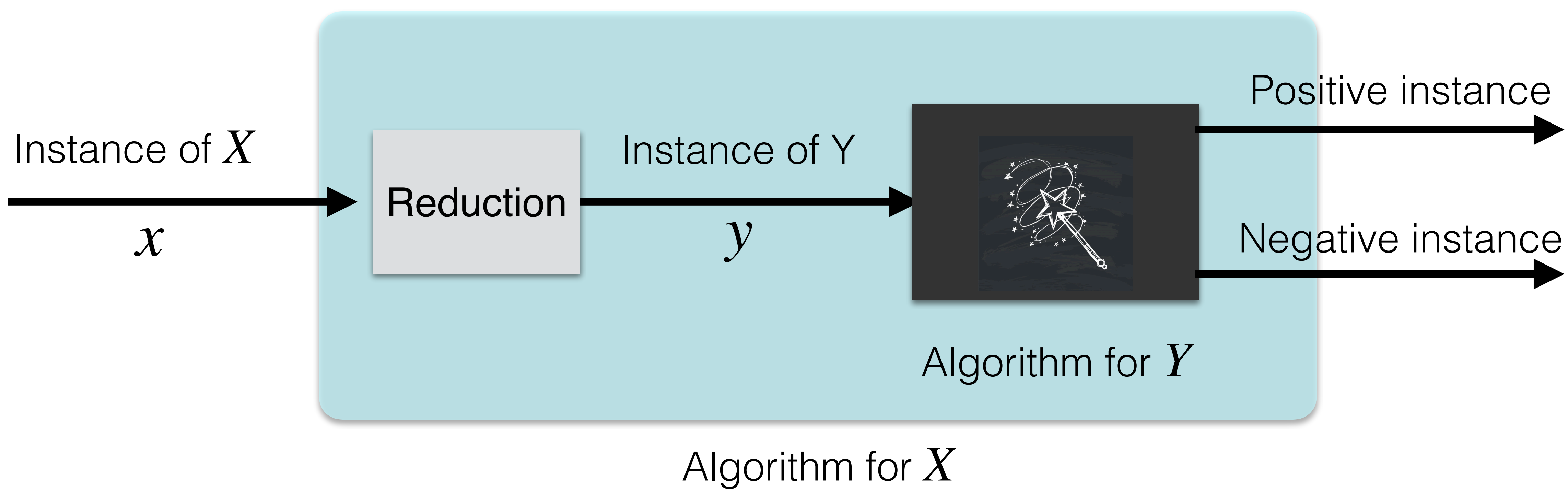
Solving Problems by Reduction to Network Flows

Max-Flow Min-Cut Applications

- Data mining
- Bipartite matching
- Network reliability
- Image segmentation
- Baseball elimination
- Network connectivity
- Markov random fields
- Distributed computing
- Network intrusion detection
- Many, many, more.

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



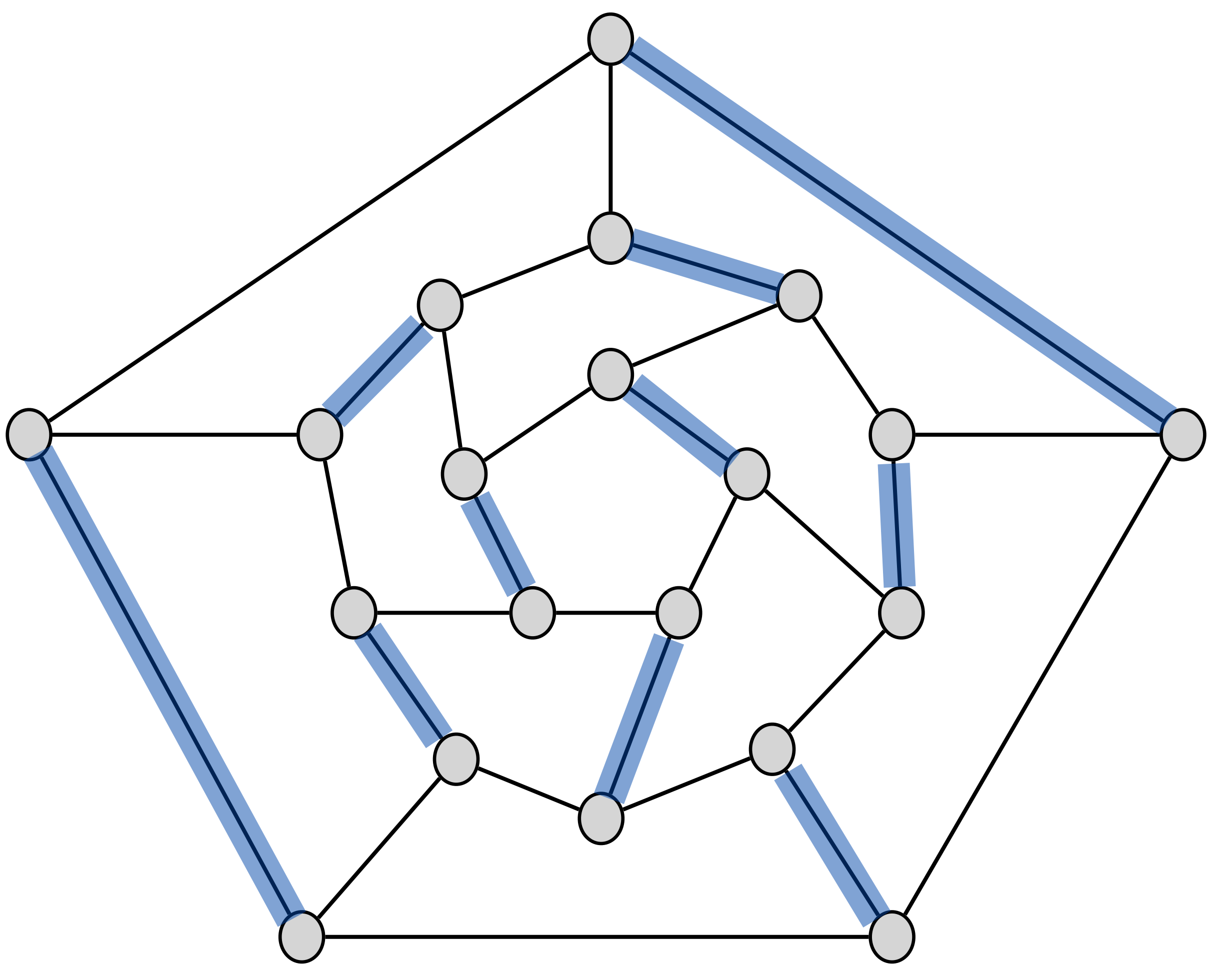
Anatomy of Problem Reductions

- **Claim.** x satisfies a property iff y satisfies a corresponding property
- Proving a reduction is correct: prove both directions
- x has a property (e.g. has matching of size k) $\implies y$ has a corresponding property (e.g. has a flow of value k)
- x does not have a property (e.g. does not have matching of size k) $\implies y$ does not have a corresponding property (e.g. does not have a flow of value k)
- Or equivalently (and this is often easier to prove):
  - y has a property (e.g. has flow of value k) $\implies x$ has a corresponding property (e.g. has a matching of value k)

Max-Cardinality 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.



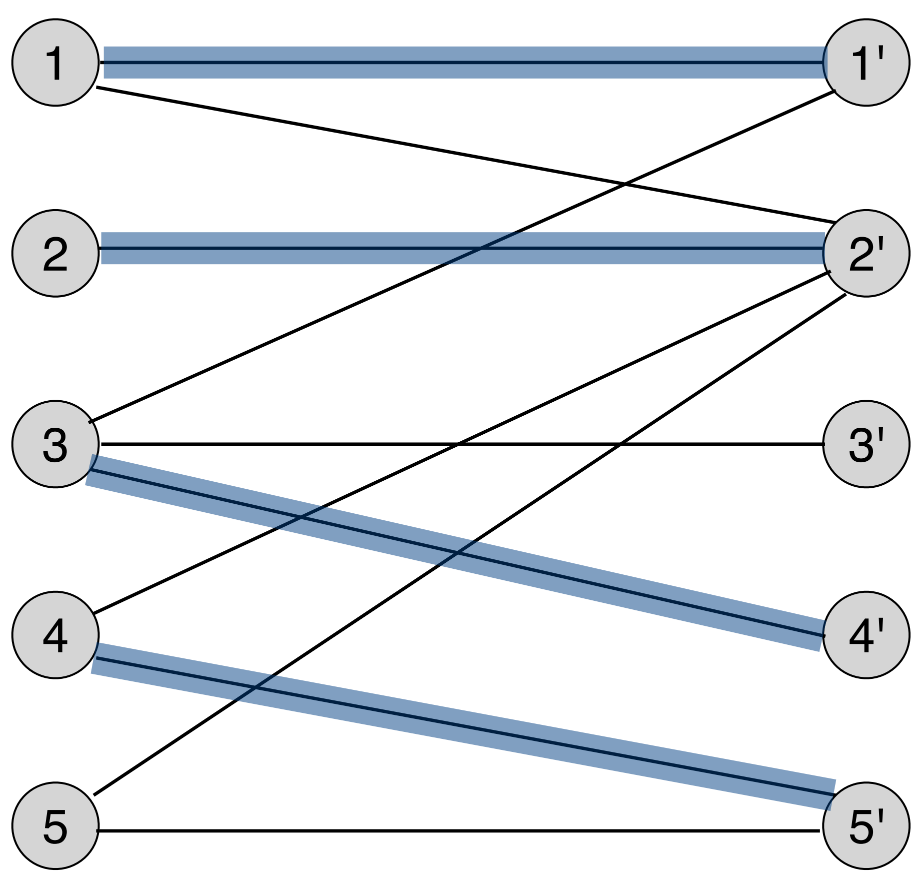
Review: Matching in Graphs

- **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.

- **Max matching problem.** Find a matching of maximum cardinality for a given graph, that is, a matching with maximum number of edges

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.

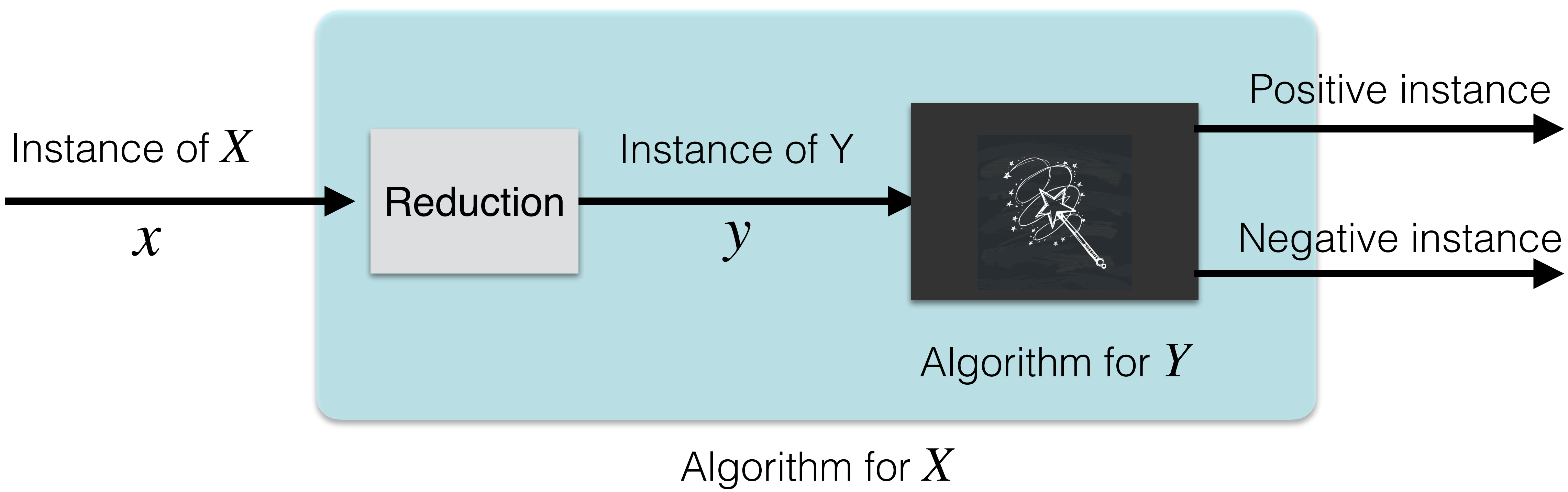


Bipartite Matching Example

- Suppose A is a set of students, B as a set of dorms
- Each student lists a set of dorms they'd like to live in, each dorm lists students it is willing to accommodate
- **Goal.** Find the largest matching (student, dorm) pairs that satisfies their requirements
- **Bipartite matching instance.** $V = (A, B)$ and $e \in E$ if student and dorm are mutually acceptable, goal is to find maximum matching
- **Note.** This is a different problem than the one we studied for Gale-Shapely matching!

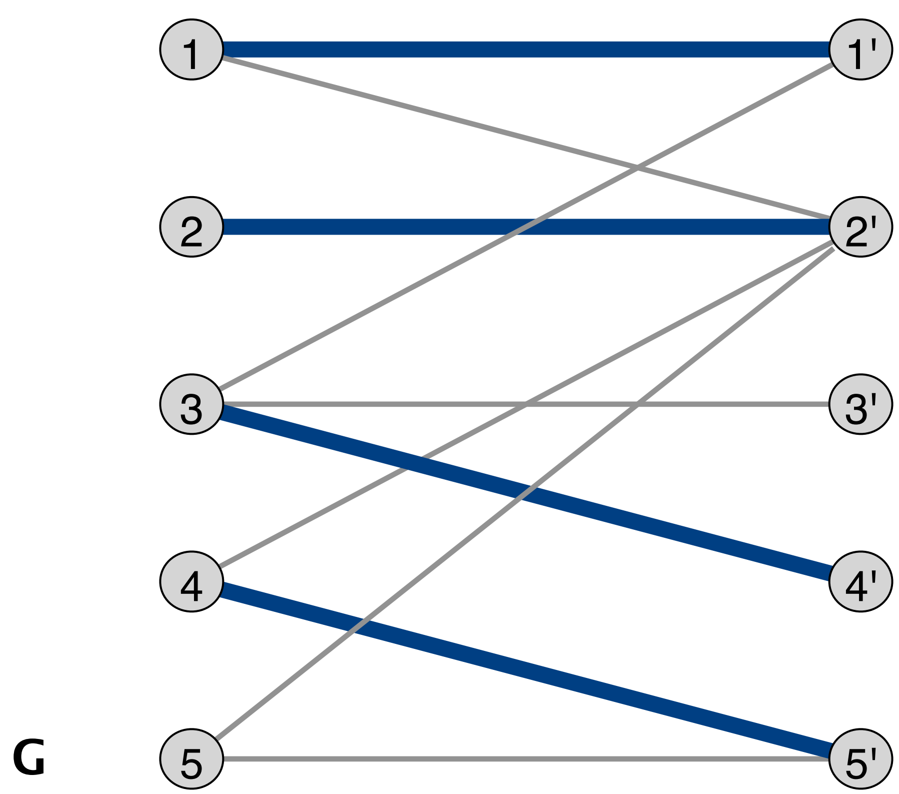
Reduction to Max Flow

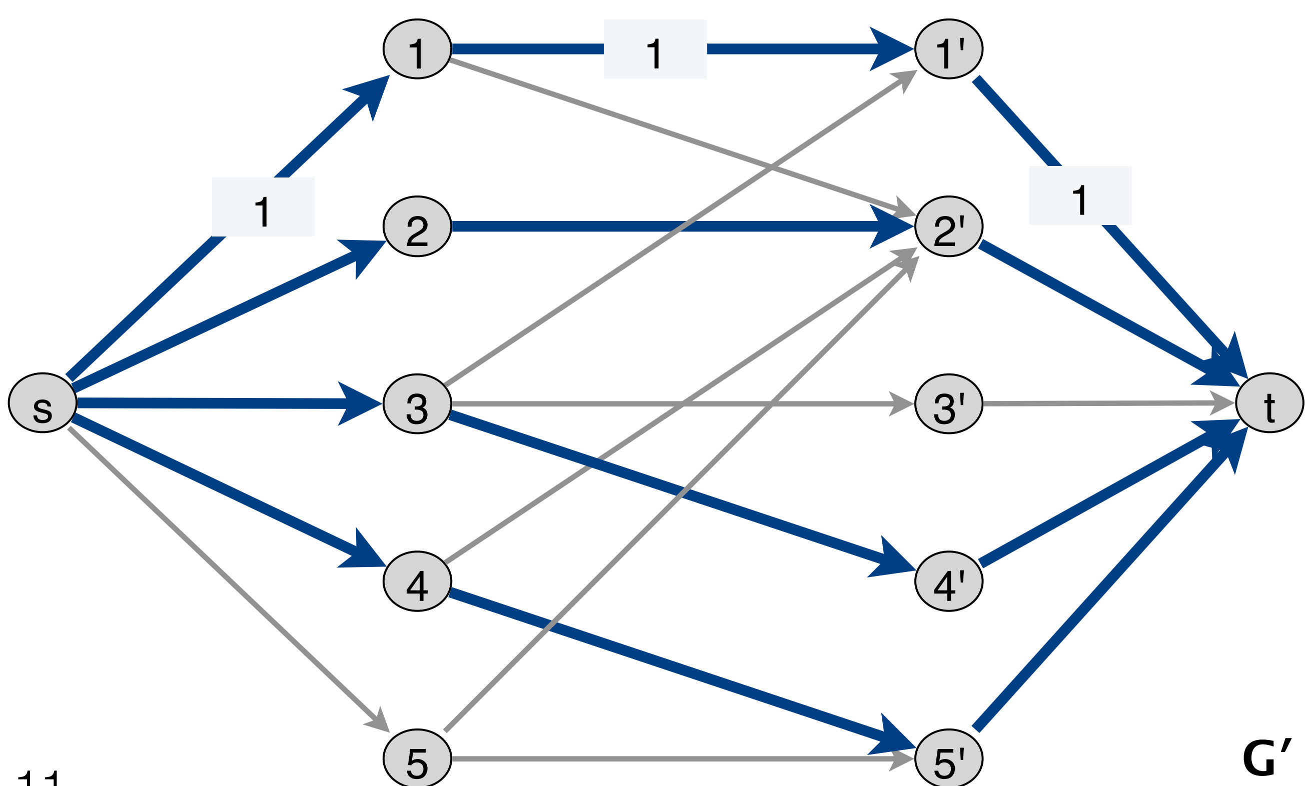
- Given arbitrary instance x of bipartite matching problem (X): A, B and edges E between A and B
- **Goal.** Create a special instance y of a max-flow problem (Y): flow network: $G(V, E, c)$, source s, sink $t \in V$ s.t.
- **1-1 correspondence.** There exists a matching of size k iff there is a flow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



Reduction to Max Flow

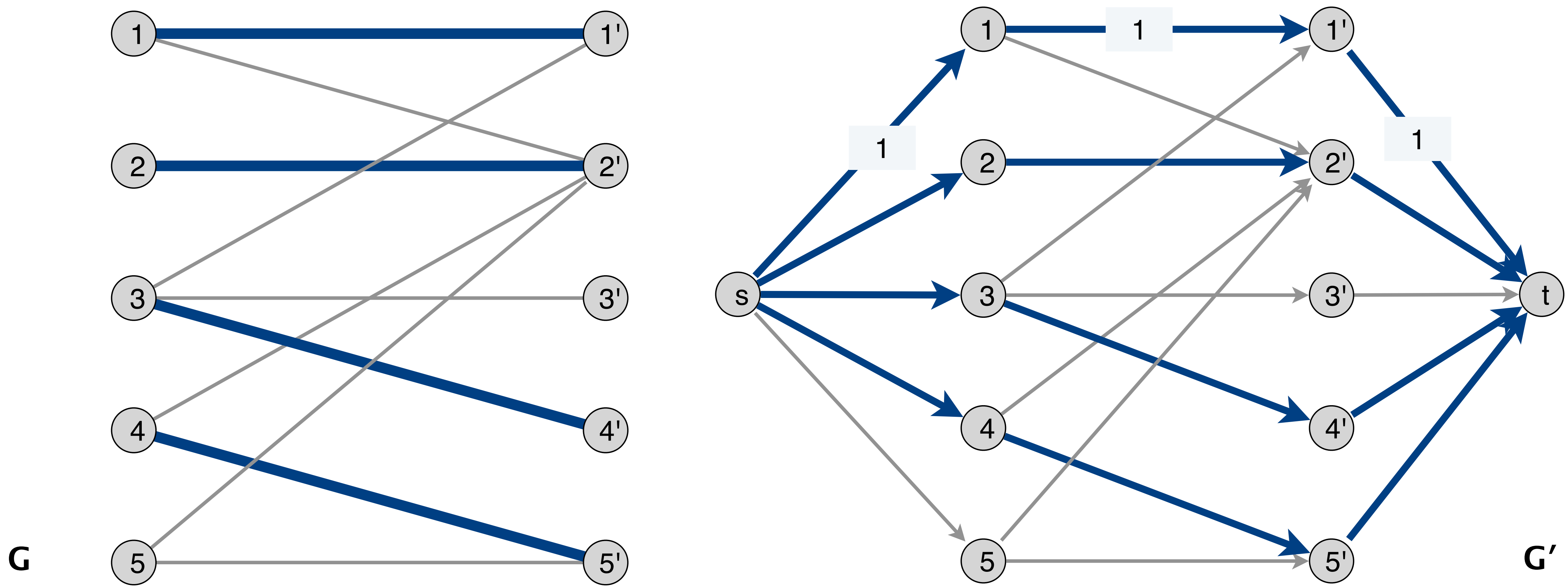
- Create a new directed graph $G' = (A \cup B \cup \{s, t\}, E', c)$
- Add edge $s \to a$ to E' for all nodes $a \in A$
- Add edge $b \to t$ to E' for all nodes $b \in B$
- Direct edge $a \to b$ in E' if $(a, b) \in E$
- Set capacity of all edges in E' to 1





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.



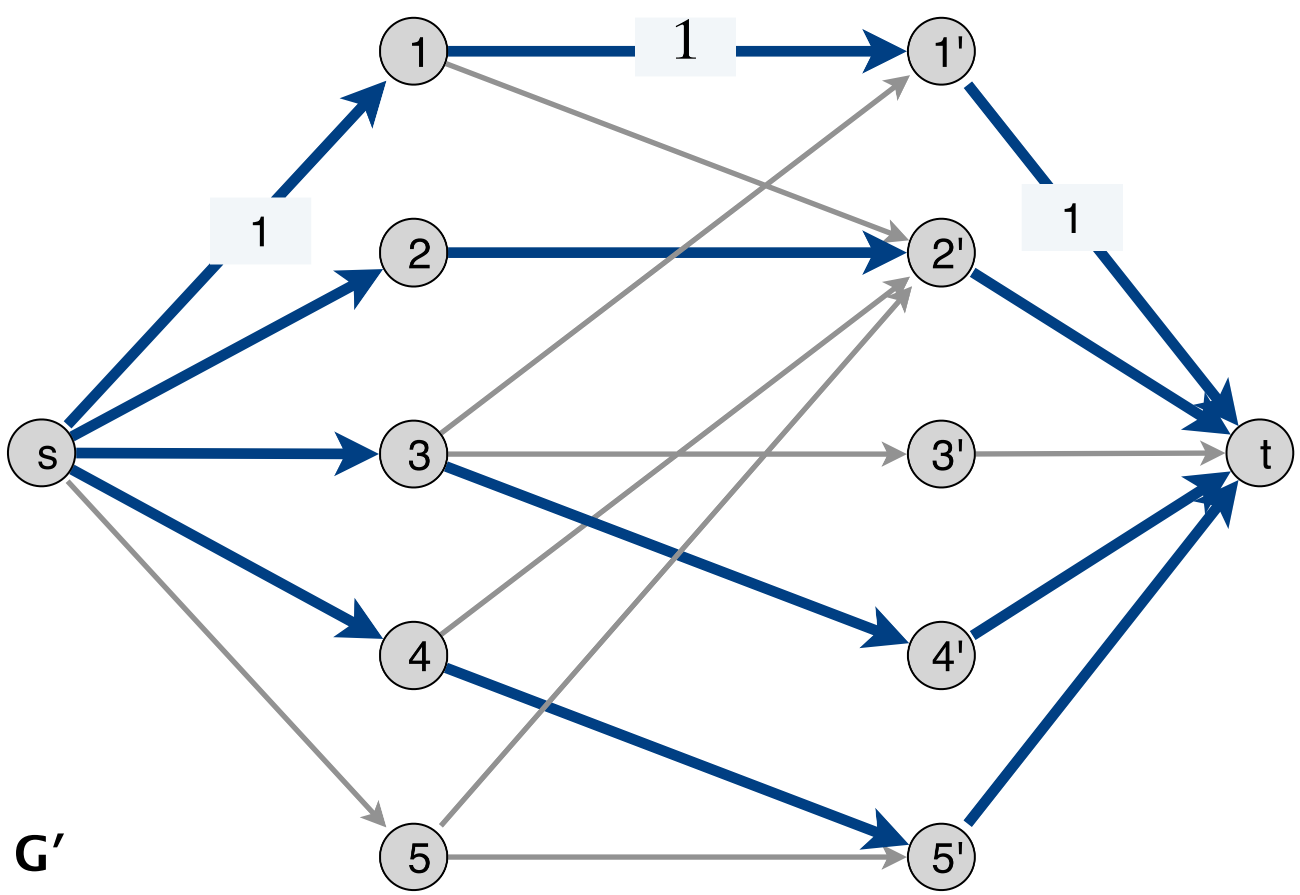
Correctness of Reduction

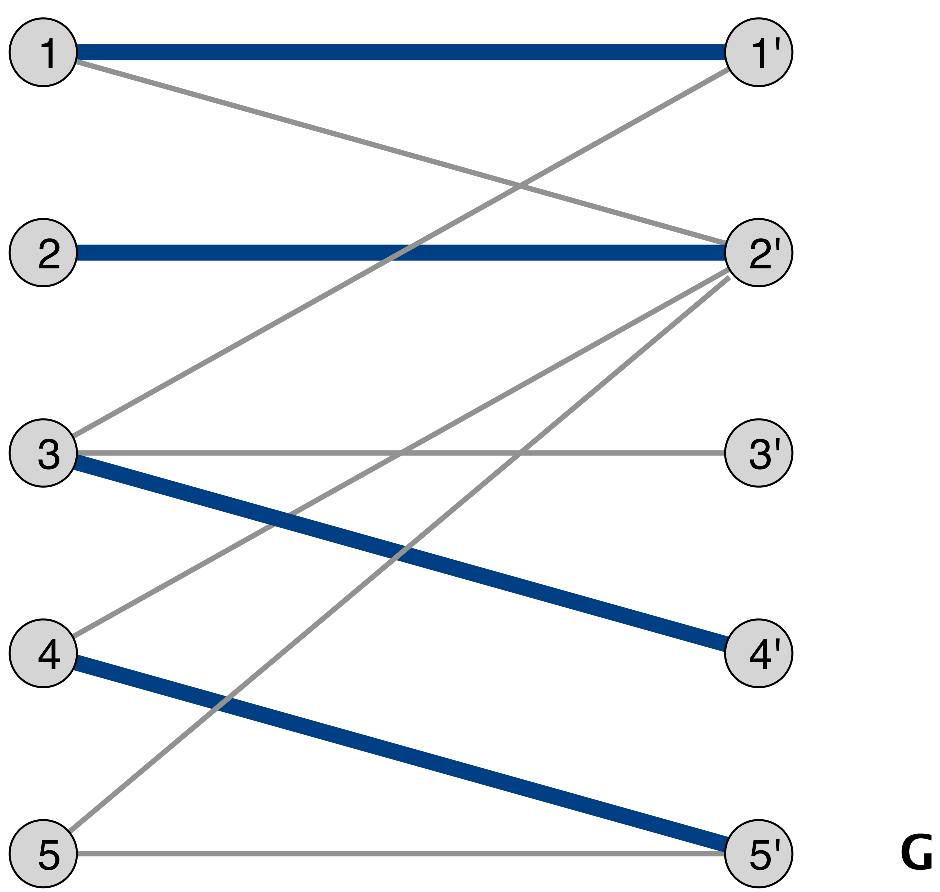
- **Claim ($\Rightarrow$).**
  If the bipartite graph (A, B, E) has matching M of size k then flow-network G' has an integral flow of value k.

- **Proof**.
  - For every edge $e = (a, b) \in M$, let f be the flow resulting from sending 1 unit of flow along the path $s \rightarrow a \rightarrow b \rightarrow t$
  - 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

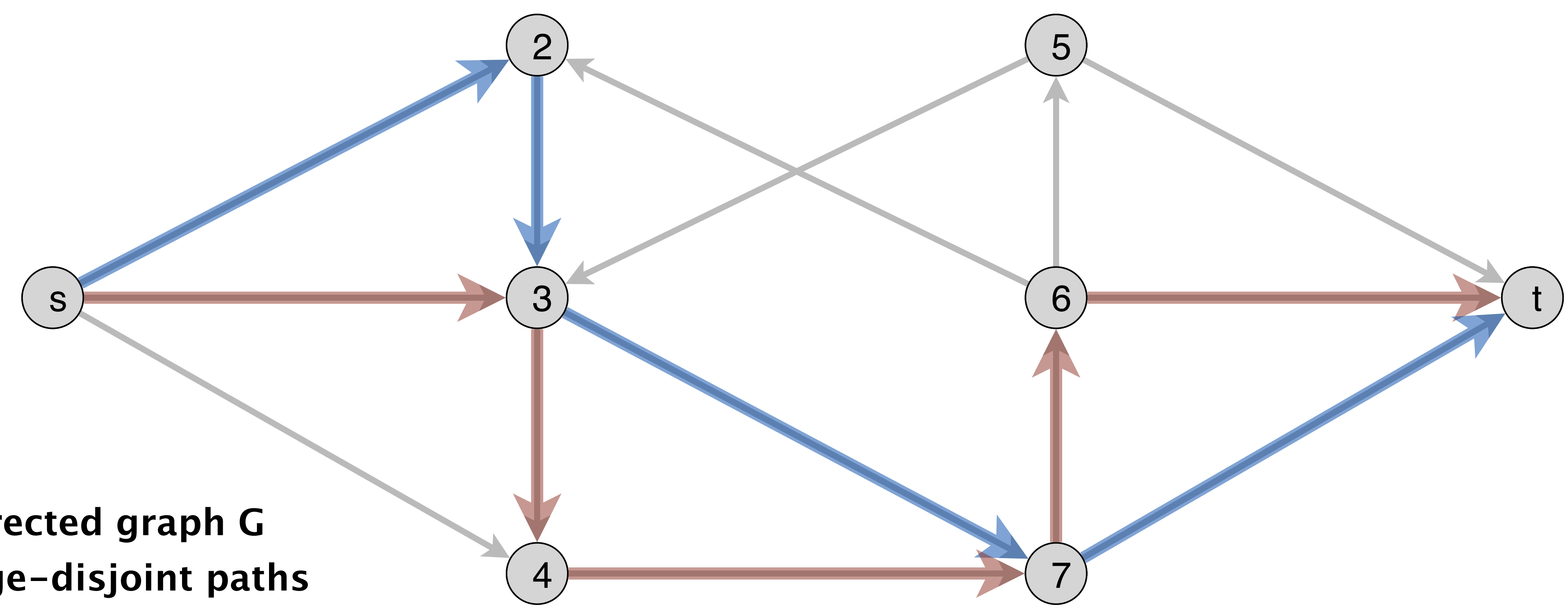
- Proved matching of size k iff flow of value k
- Thus, max-flow iff max matching
- Running time of algorithm overall:
    - Running time of reduction + running time of solving the flow problem (dominates)
- What is running time of Ford–Fulkerson algorithm for a flow network with all unit capacities?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

Disjoint Paths Problem

Disjoint Paths Problem

- **Definition.** Two paths are edge-disjoint if they do not have an edge in common.

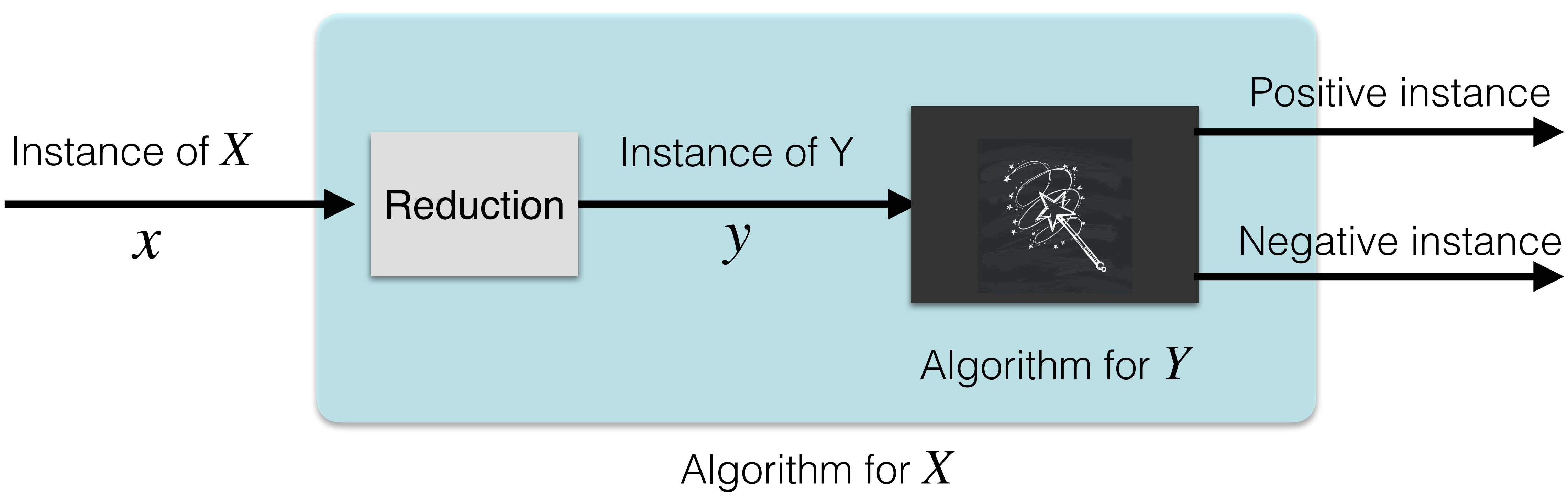
- **Edge-disjoint paths problem.** Given a directed graph with two nodes s and t, find the max number of edge-disjoint $s \rightsquigarrow t$ paths.



Directed graph G
2 edge-disjoint paths

Towards Reduction

- Given: arbitrary instance x of disjoint paths problem (X): directed graph G, with source s and sink t
- **Goal**. create a special instance y of a max-flow problem (Y): flow network $G'(V', E', c)$ with s', t' s.t.
- **1-1 correspondence.** Input graph has k edge-disjoint paths iff flow network has a flow of value k



Reduction to Max Flow

- **Reduction.** G' : same as G with unit capacity assigned to every edge
- **Claim** [Correctness of reduction]. G has k edge disjoint $s \rightsquigarrow t$ paths iff G' has an integral flow of value k.
- Proof. $(\Rightarrow)$
- Set $f(e) = 1$ if e in some disjoint $s \rightsquigarrow t$, $f(e) = 0$ otherwise.
- We have $v(f) = k$ since paths are edge disjoint.
- $(\Leftarrow)$ Need to show: If G' has a flow of value k then there are k edge-disjoint $s \rightsquigarrow t$ paths in G

Correction of Reduction

- **Claim**. $(\Leftarrow)$ If f is a 0-1 flow of value k in G', then the set of edges where $f(e) = 1$ contains a set of k edge-disjoint $s \rightsquigarrow t$ paths in G.
- **Proof** [By induction on the # of edges k' with $f(e) = 1$]
- If $k' = 0$, no edges carry flow, nothing to prove
- IH: Assume claim holds for all flows that use $< k'$ edges
- Consider an edge $s \to u$ with $f(s \to u) = 1$
- By flow conservation, there exists an edge $u \to v$ with $f(u \to v) = 1$, continue “tracing out the path" until
- Case (a) reach t, Case (b) visit a vertex v for a 2nd time

Correction of Reduction

- **Case (a)** We reach t, then we found a $s \rightsquigarrow t$ path P
  - f' : Decrease the flow on edges of P by 1
  - $v(f') = v(f) - 1 = k - 1$
  - Number of edges that carry flow now $< k'$: can apply IH and find $k - 1$ other $s \rightsquigarrow t$ disjoint paths
- **Case (b)** visit a vertex v for a 2nd time: consider cycle C of edges visited btw 1st and 2nd visit to v
  - f' : decrease flow values on edges in C to zero
  - $v(f') = v(f)$ but # of edges in f' that carry flow $< k'$, can now apply IH to get k edge disjoint paths

$\blacksquare$

Summary & Running Time

- Proved k edge-disjoint paths iff flow of value k
- Thus, max-flow iff max # of edge-disjoint $s \leadsto t$ paths
- Running time of algorithm overall:
    - Running time of reduction + running time of solving the max-flow problem (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 # of edge-disjoint $s \leadsto t$ paths: $O(nm)$

[Take-home Exercise]
Reduction to Think About

Room Scheduling

- Williams College is holding a big gala and has hired you to write an algorithm to schedule rooms for all the different parties happening as part of it.
- There are n parties and the ith party has p_i invitees.
- There are r different rooms and the jth room can fit r_j people in it.
- Thus, party i can be held in room j iff $p_i \leq r_j$.
- Describe and analyze an efficient algorithm to assign a room to each party (or report correctly that no such assignment is possible).

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)