Hello!

CS3000: Algorithms & Data
Jonathan Ullman

Lecture 16:

- Applications of Network Flow

March 23, 2020

Midterm II

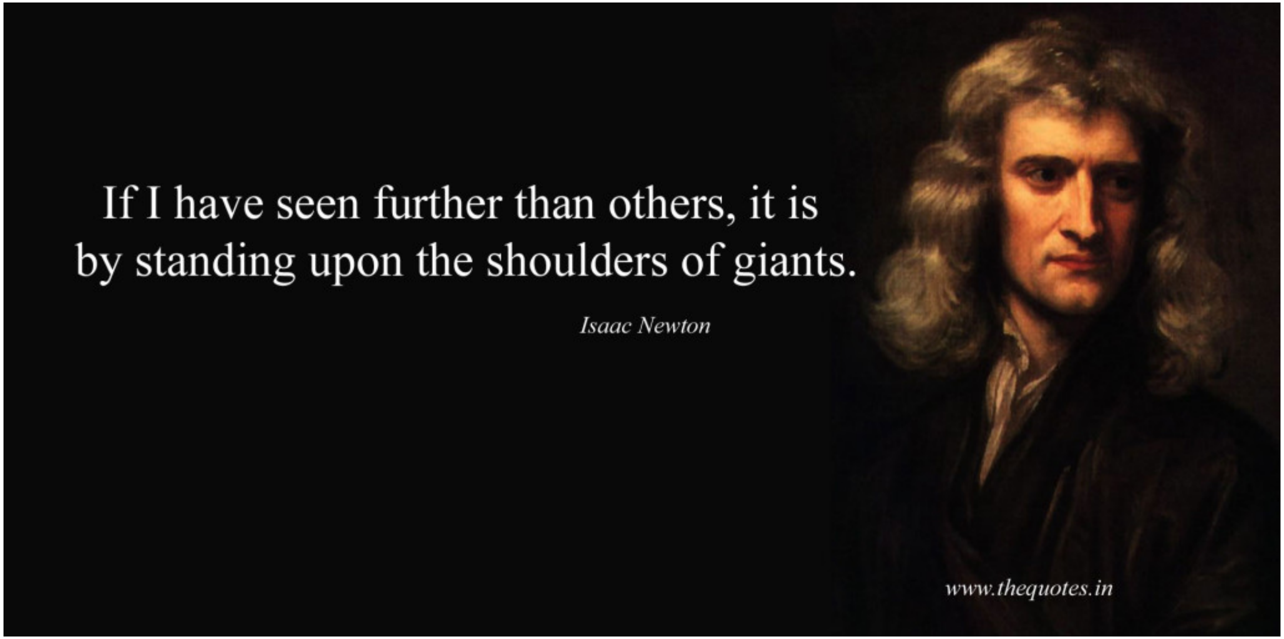
- **Logistics:**
  - Wednesday April 1st
  - Administratered through Gradescope
  - Exam will be available from 2:50pm until _____
  - You have 90 minutes from the time you start
- **Academic Honesty:**
  - You will have to take an honor pledge
  - Do not discuss the exam with **anyone** for 24 hours
  - Please be a good citizen

Midterm II

- **Format:**
  - Same as Midterm I, but condensed for Gradescope

- **Topics (graph algos):**
  - Key graph definitions and concepts
  - Adjacency list / matrix representations
  - DFS + topological sort
  - Minimum Spanning Tree
  - Shortest paths: BFS, Dijkstra, Bellman-Ford
  - Network flow: concepts, Ford-Fulkerson, choosing paths
  - Does not include this week

Applications of Network Flow



Applications of Network Flow

- Algorithms for maximum flow can be used to solve:
  - Bipartite Matching
  - Disjoint Paths
  - Survey Design
  - Matrix Rounding
  - Auction Design
  - Fair Division
  - Project Selection
  - Baseball Elimination
  - Airline Scheduling
  - …

If a problem can be solved in polynomial time, then it can be written as an application of max flow.

Reduction

- **Definition:** a **reduction** is an efficient algorithm that solves problem A using calls to function that solves problem B.

Mechanics of Reductions

- What exactly is a **problem**?
  - A set of legal inputs $\boldsymbol{X}$ (e.g. a list of numbers A[1] ... A[n])
  - A set $\boldsymbol{A(x)}$ of legal outputs for each $\boldsymbol{x} \in \boldsymbol{X}$

    (e.g. A in sorted order)

- **Example:** integer maximum flow

Inputs: $G = (V, E, \{c(e)\}, s, t)$ where

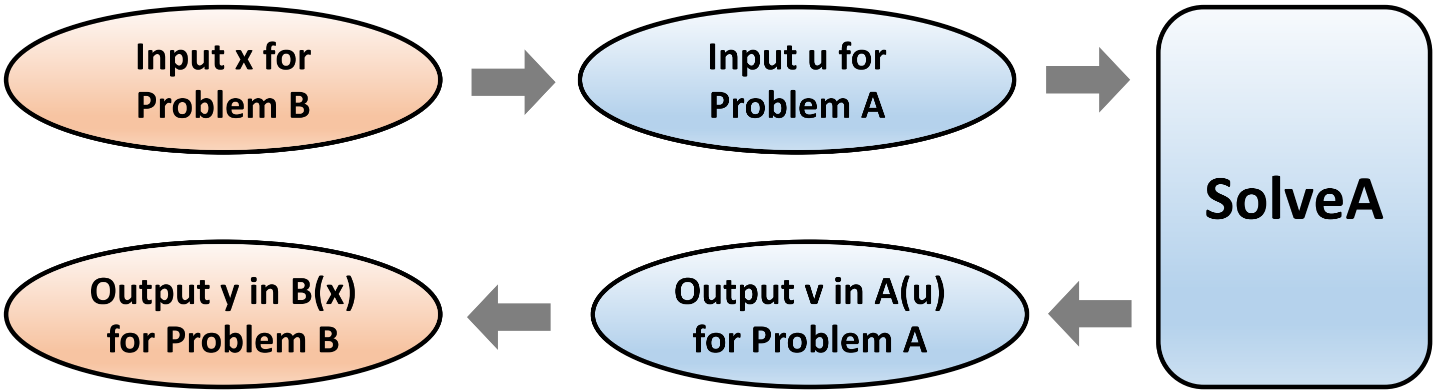
$c(e)$ is an integer for every $e \in E$.

Outputs: A maximum flow $\{f(e)\}$ for G

where $f(e)$ is an integer for every $e \in E$.

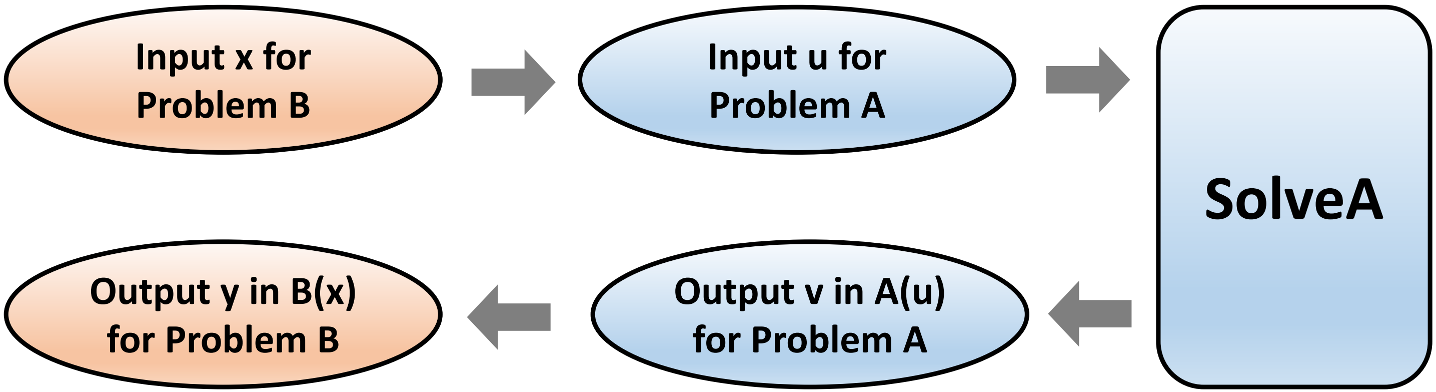
Mechanics of Reductions

Use an alg for Problem A to solve Problem B



Simplified Version: we only make one call to SolveA

When is a Reduction Correct?

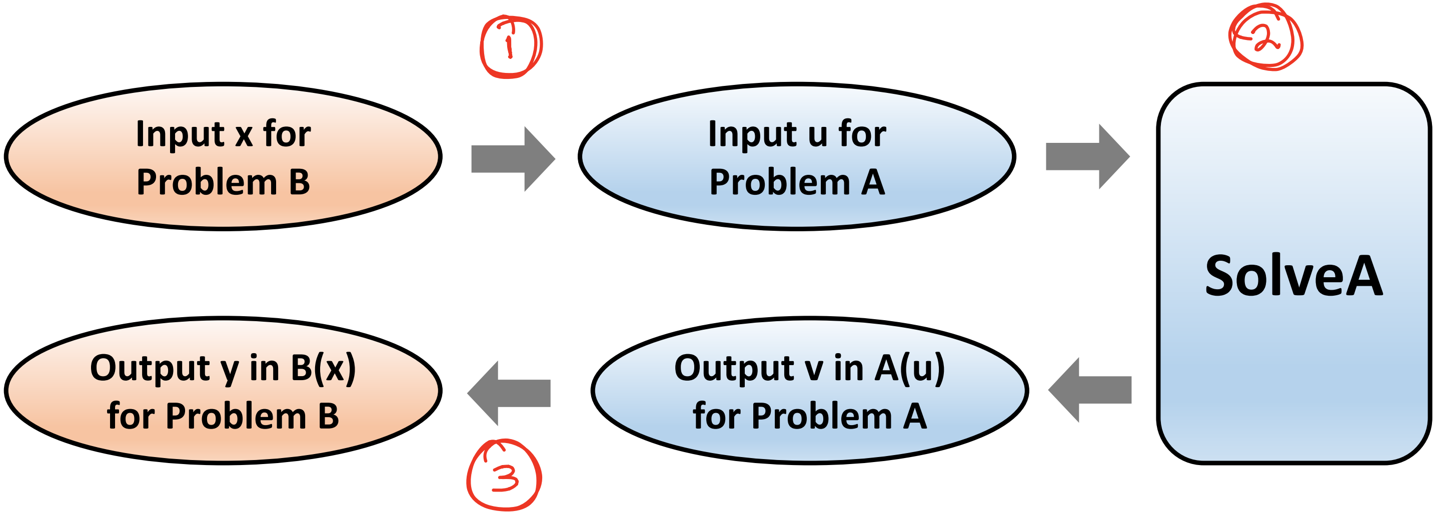


For every valid input x:
If v is any valid output in $A(u)$,
then y is a valid output in $B(x)$

Assume for valid input u for ProbA,
SolveA(u) returns a valid output $v \in A(u)$

Can be any $v \in A(u)$

What is the Running Time?



Total Time = 1 + 2 + 3

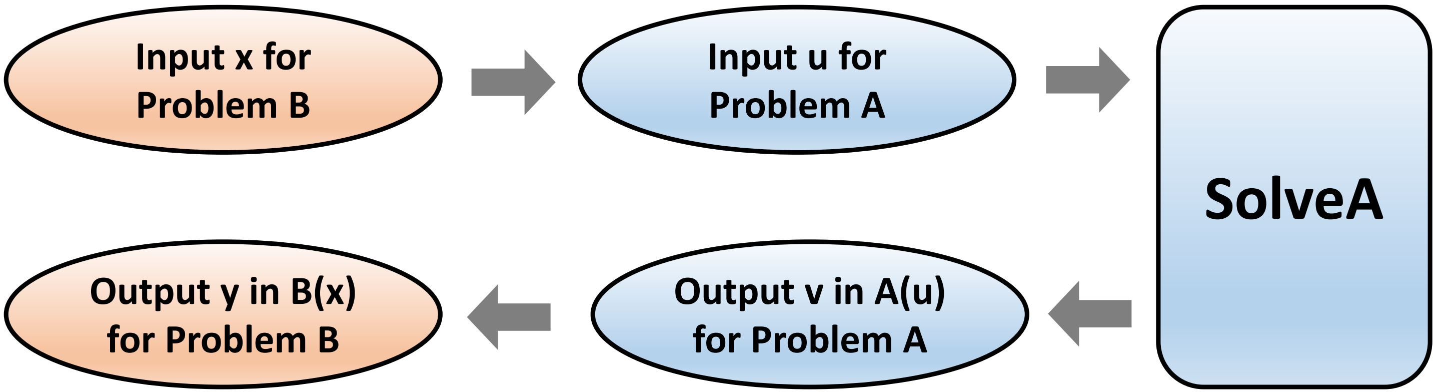
2 depends on the running time of Solve A on the input u.

(The size of u may differ from that of x)

Example: Minimum Cut

A = max flow

B = min cut



Input x for B: $G = (V, E, \{c(e)\}, s, t)$

$\Downarrow$

Input u for A: $G = (V, E, \{c(e)\}, s, t)$

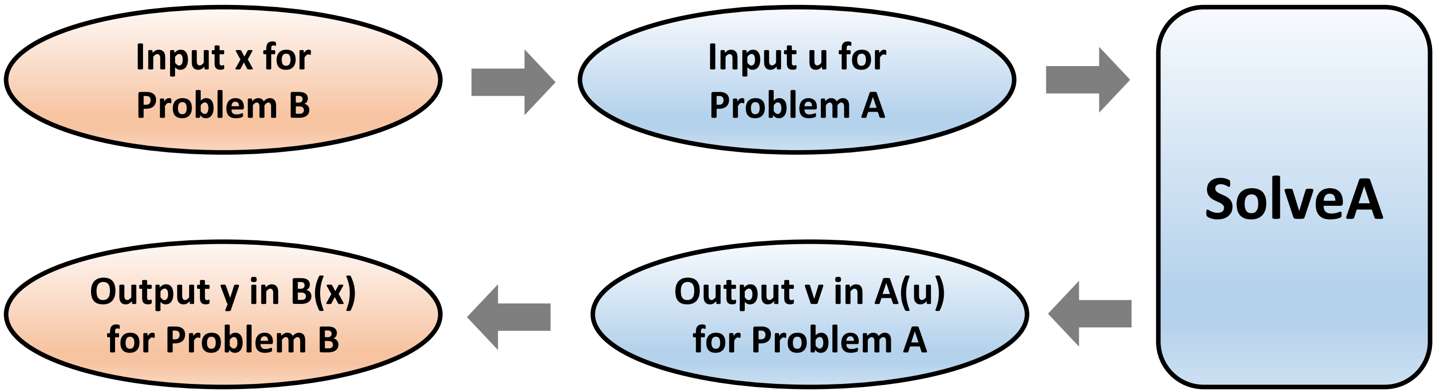
$\Downarrow$

Output $v \in A(u)$: A max flow f in G

Example: Minimum Cut

A = max flow

B = min cut



$\Downarrow$

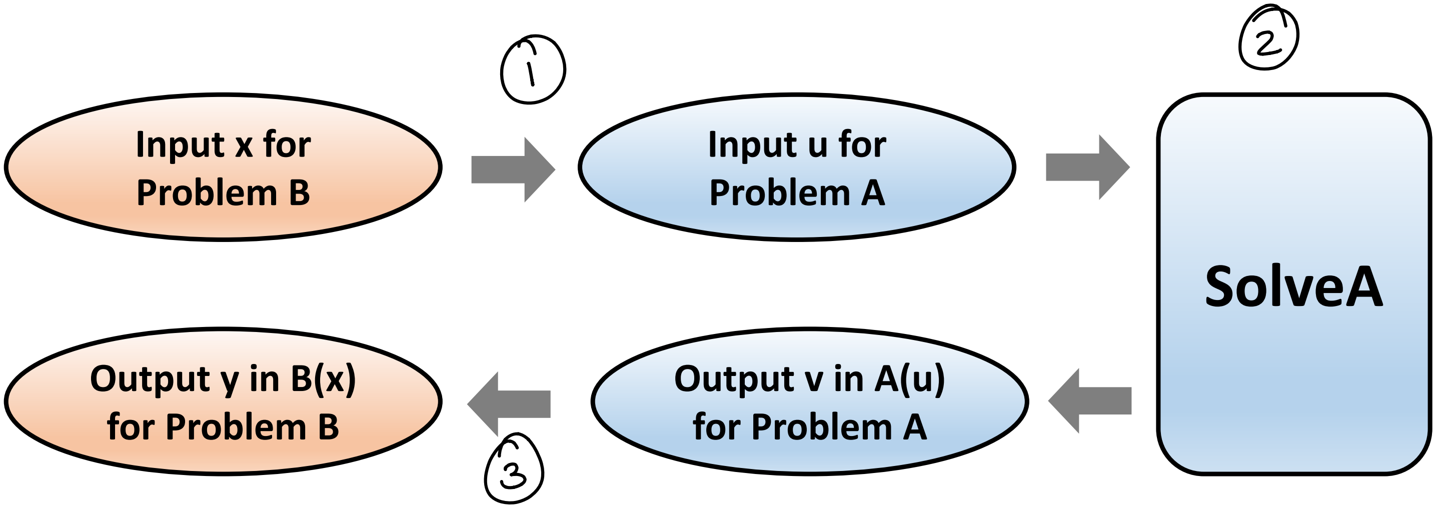
Output $y \in B(x)$:

1. Take f, compute residual graph G_f
2. Find nodes reachable from s in G_f
3. Output those nodes as the cut.

Example: Minimum Cut

A = max flow

B = min cut



Running Time:

solve max flow in $O(nm)$ time

1 Do nothing

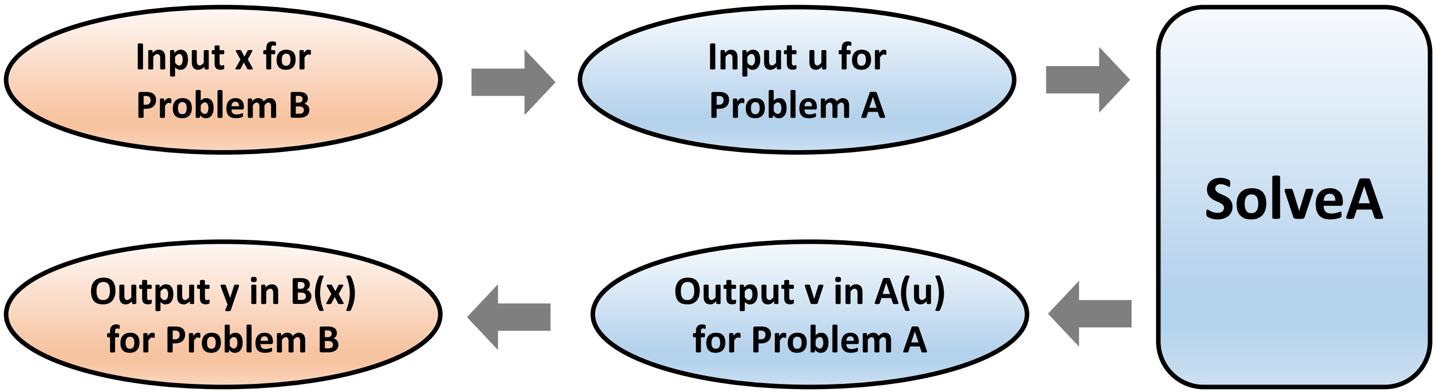
$\boxed{2\quad O(\text{(nodes in } u)(\text{edges in } u)) = O(nm)}$

3 BFS, $O(n+m)$ time

Example: Median

A = sorting

B = median



Input x for B: An array A of length n

Input u for A: The same array

Output $v \in A(u)$: A in sorted order

Output $y \in B(x)$: $A[\lfloor \frac{n}{2} \rfloor]$

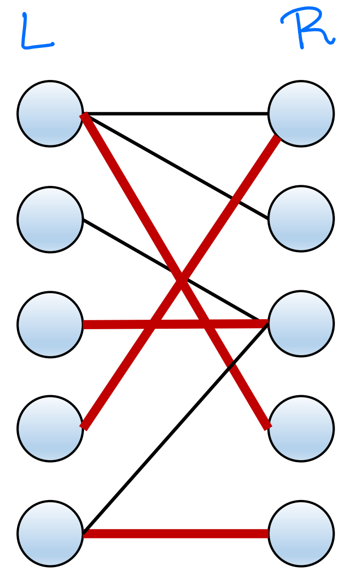
$O(n \log n)$

Bipartite Matching

$\forall e=(u,v)\in E$

$u\in L$, $v\in R$ or vice versa

- **Input:** bipartite graph $G = (V, E)$ with $V = L \cup R$
- **Output:** a maximum cardinality matching
  - A **matching** $M \subseteq E$ is a set of edges such that every node v is an endpoint of at most one edge in M
  - Cardinality = $|M|$



red edges are a matching

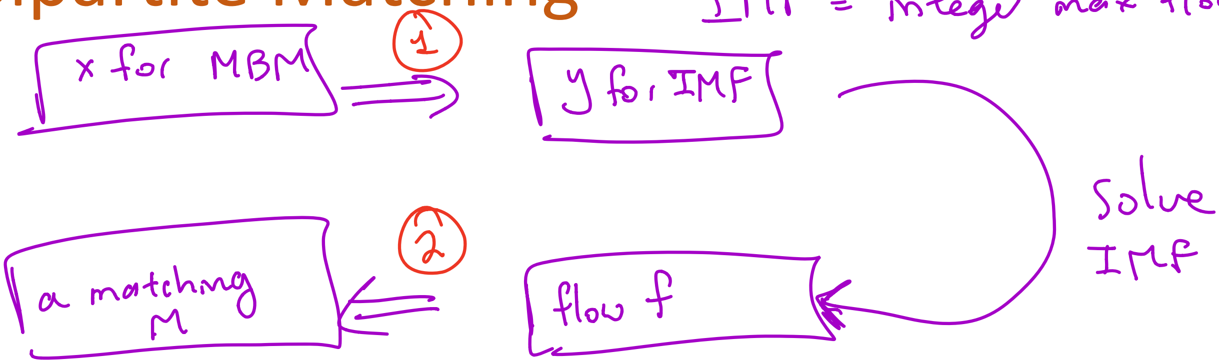
Models any problem where one type of object is assigned to another type:

- doctors to hospitals
- jobs to processors
- advertisements to websites

MBM = max bipartite matching

IMF = integer max flow

Bipartite Matching

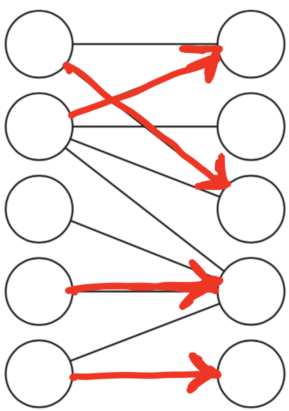


- There is a reduction that uses integer maximum s-t flow to solve maximum bipartite matching.

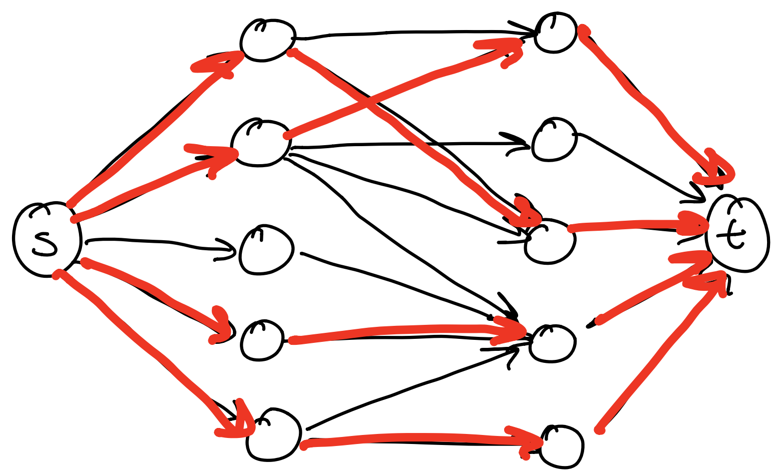
Step 1: Transform the Input

Input G for MCBM → **Input G' for MAXFLOW**

A matching gives a set of non-overlapping pipes from L to R



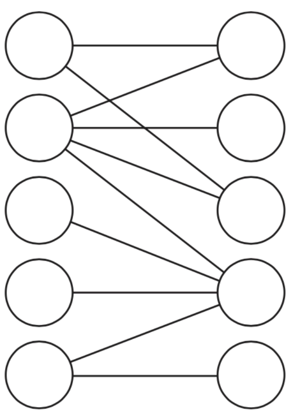
Set all capacities to $c(e) = 1$

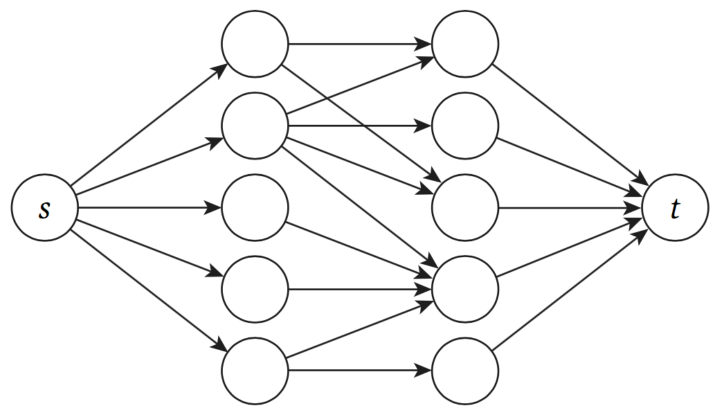


Step 1: Transform the Input

Input G for MCBM

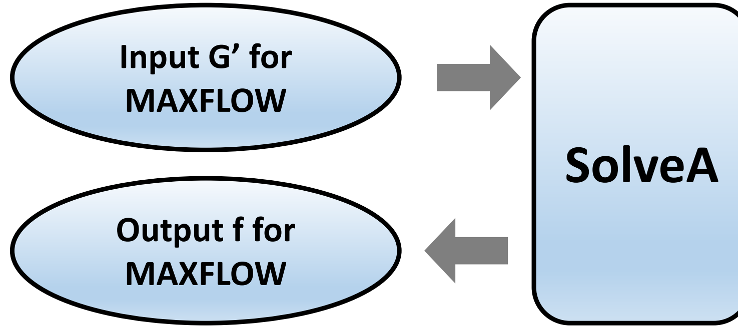
Input G' for MAXFLOW

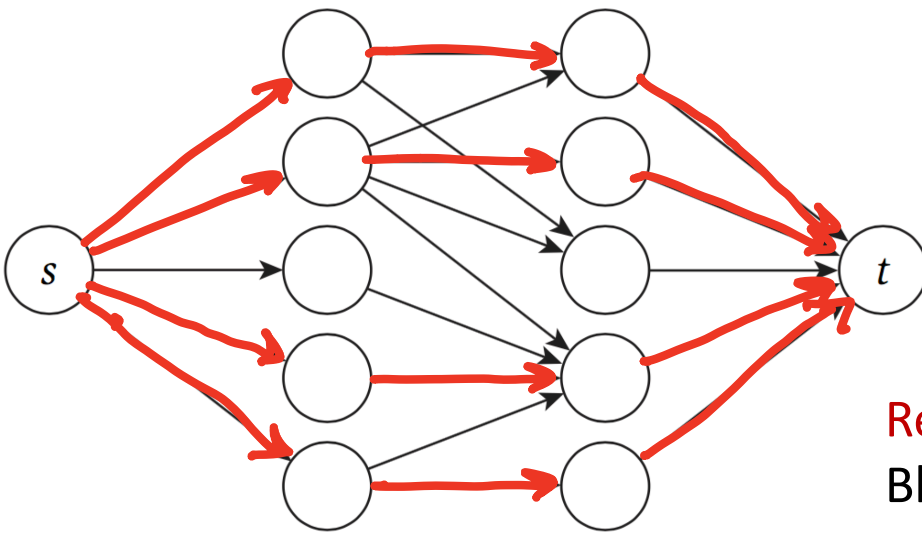




Step 2: Receive the Output

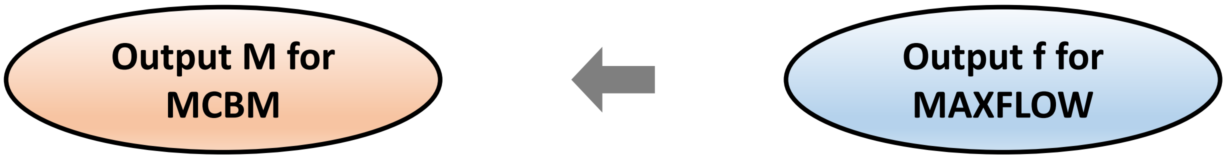
Matching will be all of the L → R edges that carry flow



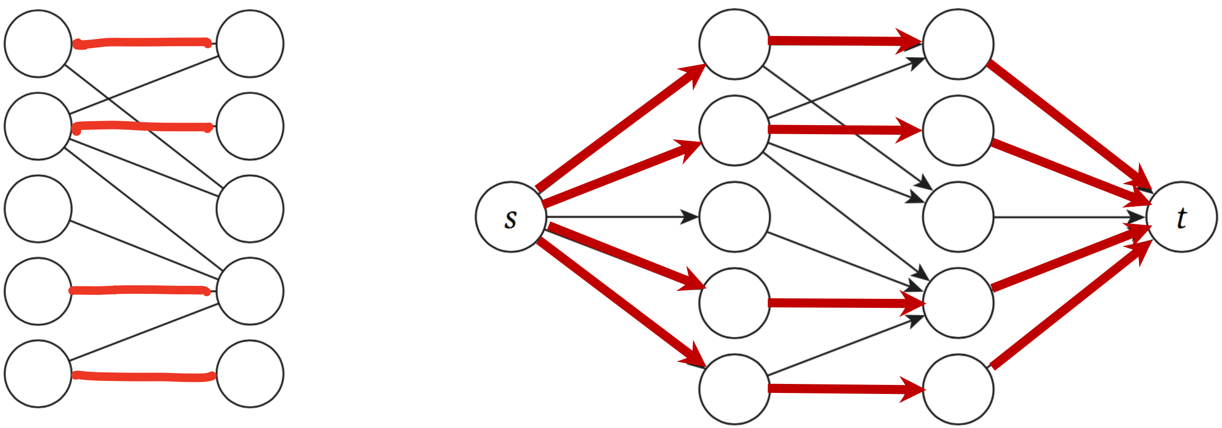


Red arrow means f(e)=1
Black means f(e) = 0

Step 3: Transform the Output



M = all edges from L to R that carry flow



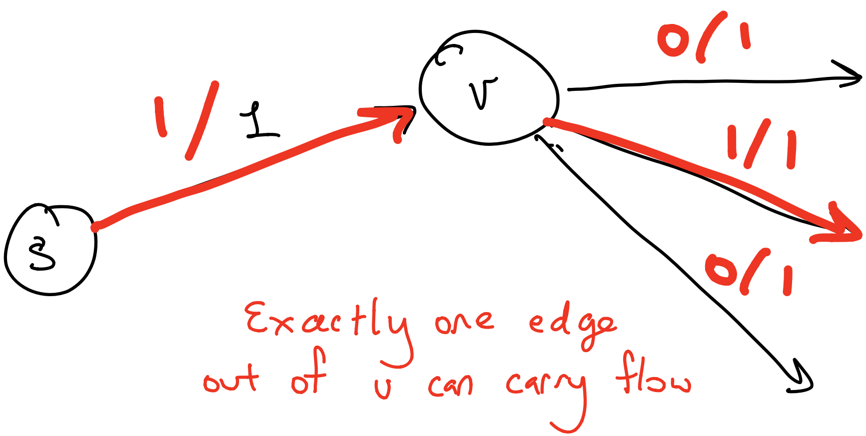
Reduction Recap

- Step 1: Transform the Input
    - Given G = (L,R,E), produce G' = (V,E,{c(e)},s,t) by...
        - ... orienting edges e from L to R
        - ... adding a node s with edges from s to every node in L
        - ... adding a node t with edges from every not in R to t
        - ... seting all capacities to 1
- Step 2: Receive the Output
    - Find an integer maximum s-t flow f in G'
- Step 3: Transform the Output
    - Given an integer s-t flow f(e)...
        - Let M be the set of edges e going from L to R that have f(e)=1

Correctness

- Need to show:
  - (1) This algorithm returns a matching
  - (2) This matching is a maximum cardinality matching

① Matching: Every node v is the endpoint of at most 1 edge in M.

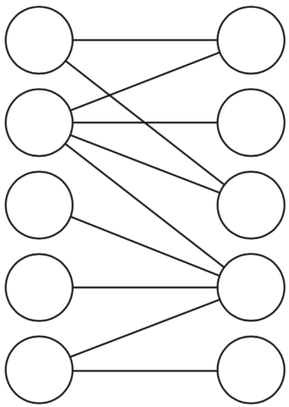


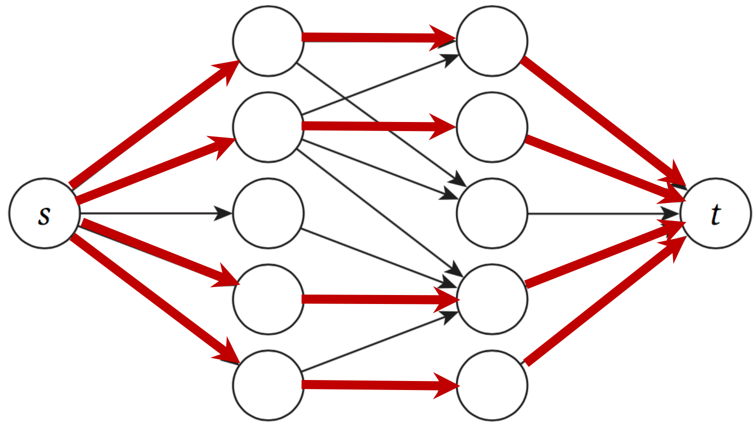
Correctness

- Need to show:
  - (1) This algorithm returns a matching
  - (2) This matching is a maximum cardinality matching

Correctness

- This algorithm returns a matching

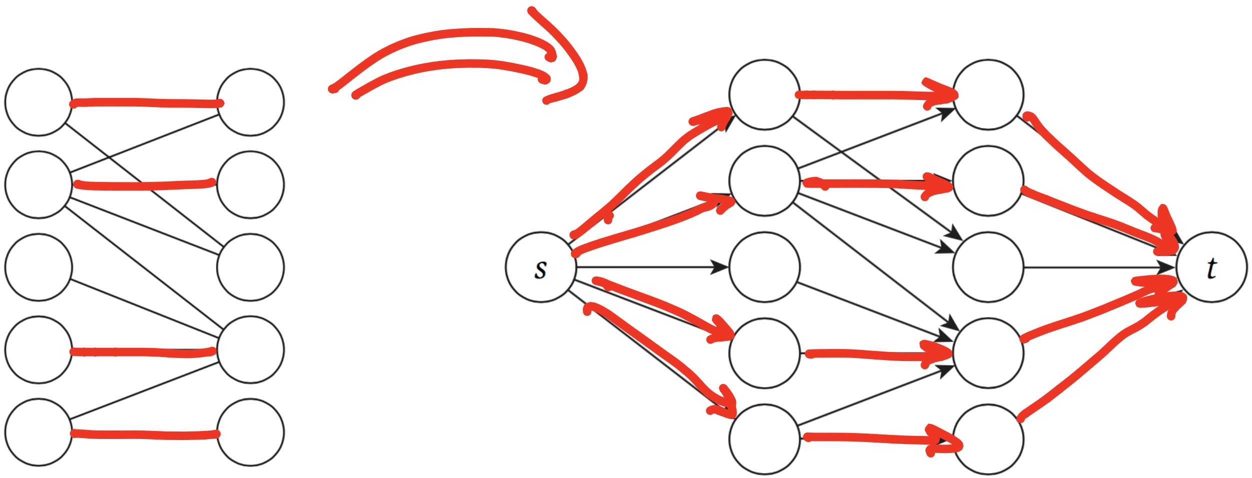




Correctness

- **Claim:** G has a matching of cardinality at least k if and only if G' has an s-t flow of value at least k

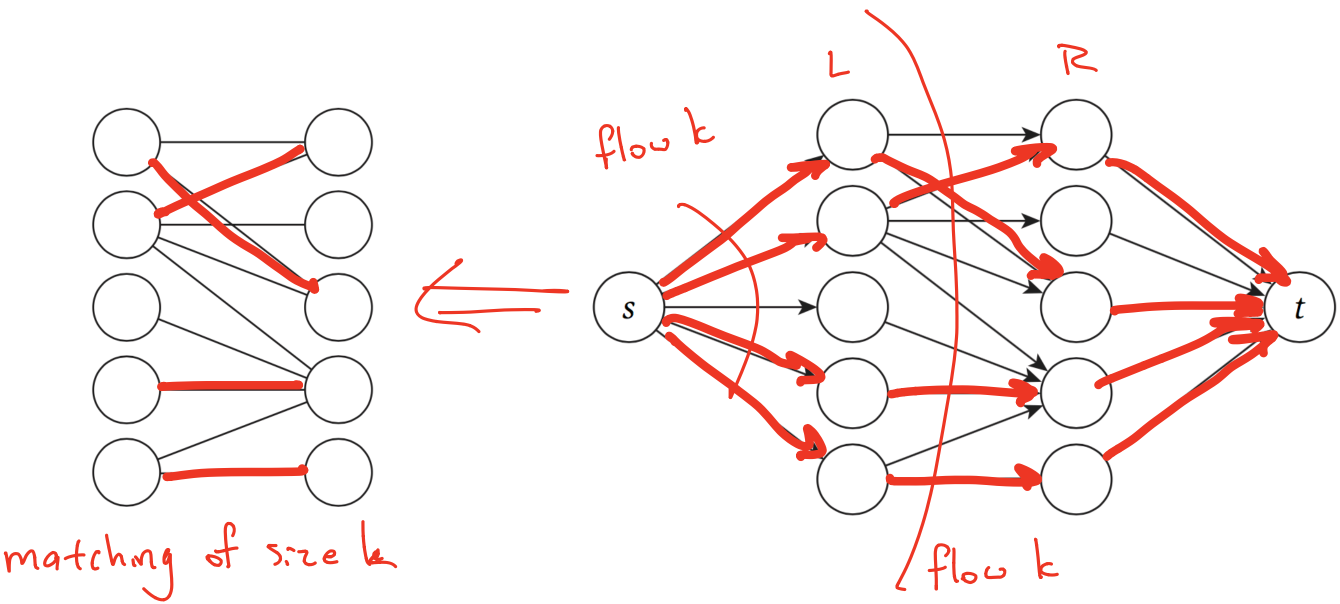
A matching of size k gives a flow of value k



Correctness

- **Claim:** G has a matching of cardinality at least k if and only if G' has an s-t flow of value at least k

A flow of value k must have k edges carrying flow from L to R



Running Time

G has n nodes, m edges

- Need to analyze the time for:
  - (1) Producing G' given G
  - (2) Finding a maximum flow in G'
  - (3) Producing M given G'

O(nm)

① G' has n+2 nodes, n+m edges
Can write G' in time O(n+m)

② T(n+2, n+m) where T(n', m') is the time to solve max flow in a graph with n' nodes m' edges : O(nm)

③ Scan the edges of G' and see which have flow
O(n+m) time

Summary

Integer

Solving maximum s-t flow in a graph with n+2 nodes and m+n edges and $c(e) = 1$ in time T

↓

Solving maximum bipartite matching in a graph with n nodes and m edges in time $T + O(m+n)$

- Can solve max bipartite matching in time $O(nm)$ using Ford-Fulkerson
  - Improvement for maximum flow gives improvement for maximum bipartite matching

Hall's Theorem:

every nodes gets matched

G has a perfect matching iff there does not exist a set of nodes $S \subseteq L$ such that $|\Gamma(S)| \leq |S|$

$\Gamma(S)$ is the set of all neighbors of nodes in S

obstacle