CS3000: Algorithms & Data
Jonathan Ullman

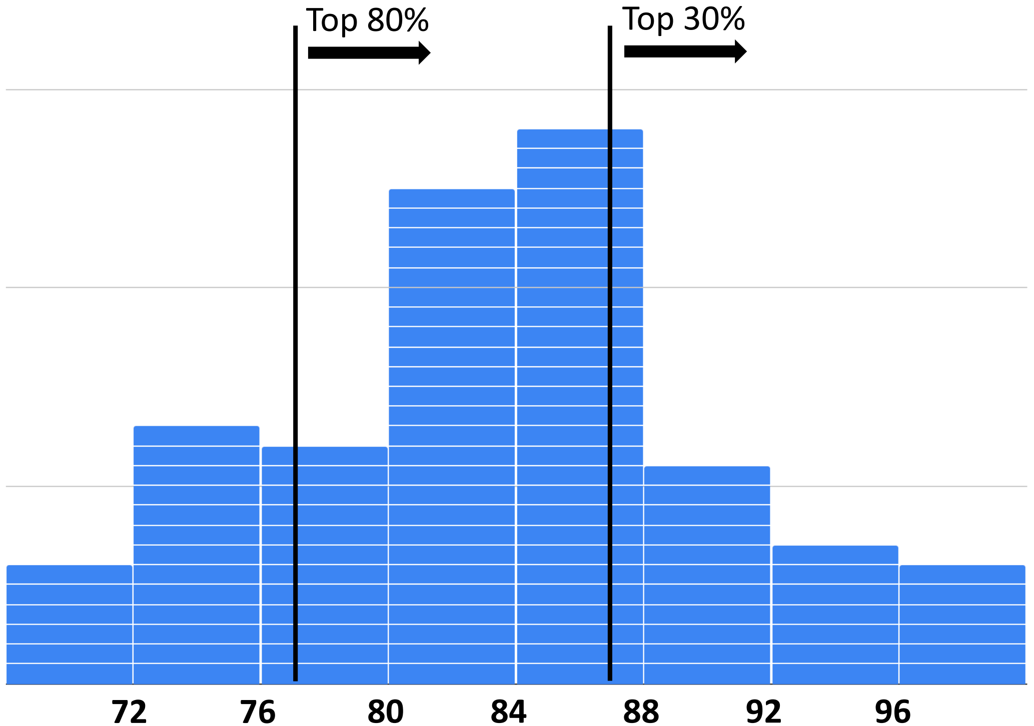
Lecture 20:
- Applications of Network Flow

Nov 27, 2018

Midterm II Stats

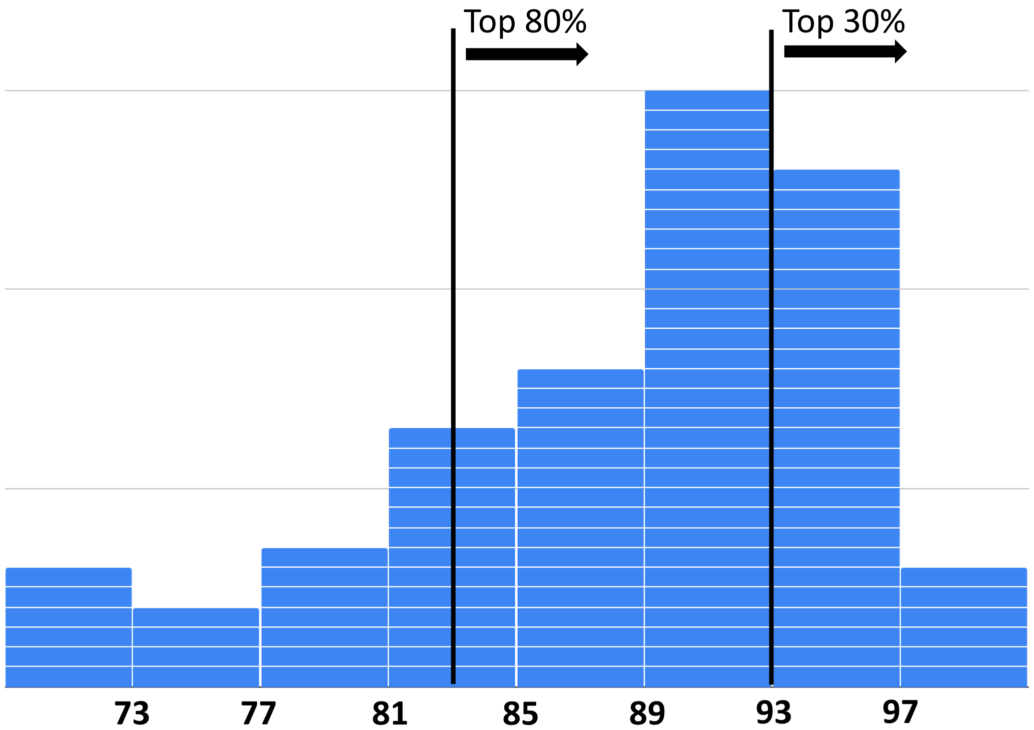
Midterm II Grade Distribution

Mean ≈ 83



Midterm II Grades were really good!

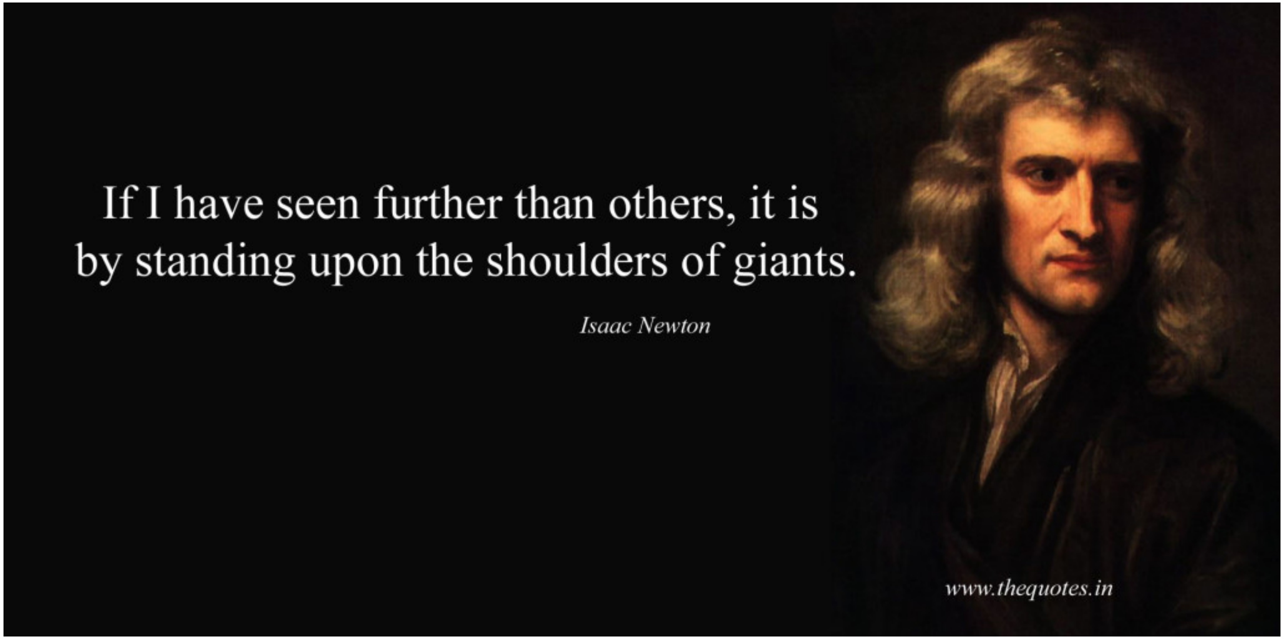
Homework Grade Distribution



I have dropped your lowest grade (so far)

Applications of Network Flow

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
  - …

Reduction

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

Given

$$
\begin{tikzcd}
\text{input} \quad x \text{ for problem } A \arrow[r, squiggly, "\text{input}"] & u \text{ for problem } B \arrow[d, "\text{(library call)}"] \\
\text{solution} \quad y \text{ for problem } A & \text{solution} \quad v \text{ for problem } B \arrow[l]
\end{tikzcd}
$$

Mechanics of Reductions

- What exactly is a **problem**?
  - A set of legal inputs $\boldsymbol{X}$
  - A set $\boldsymbol{A}(\boldsymbol{x})$ of legal outputs for each $\boldsymbol{x} \in \boldsymbol{X}$

- **Example:** sorting ~~integer maximum flow~~

$$X = \{ \text{all arrays of numbers} \}$$

$$A\left(\begin{array}{|c|c|c|c|}\hline 3 & 8 & 5 & 7 \\ \hline\end{array}\right) = \left\{ \begin{array}{|c|c|c|c|}\hline 3 & 5 & 7 & 8 \\ \hline\end{array} \right\}$$

Mechanics of Reductions

- What exactly is a **problem**?
  - A set of legal inputs $\boldsymbol{X}$
  - A set $\boldsymbol{A}(\boldsymbol{x})$ of legal outputs for each $\boldsymbol{x} \in \boldsymbol{X}$

- **Example:** integer maximum flow

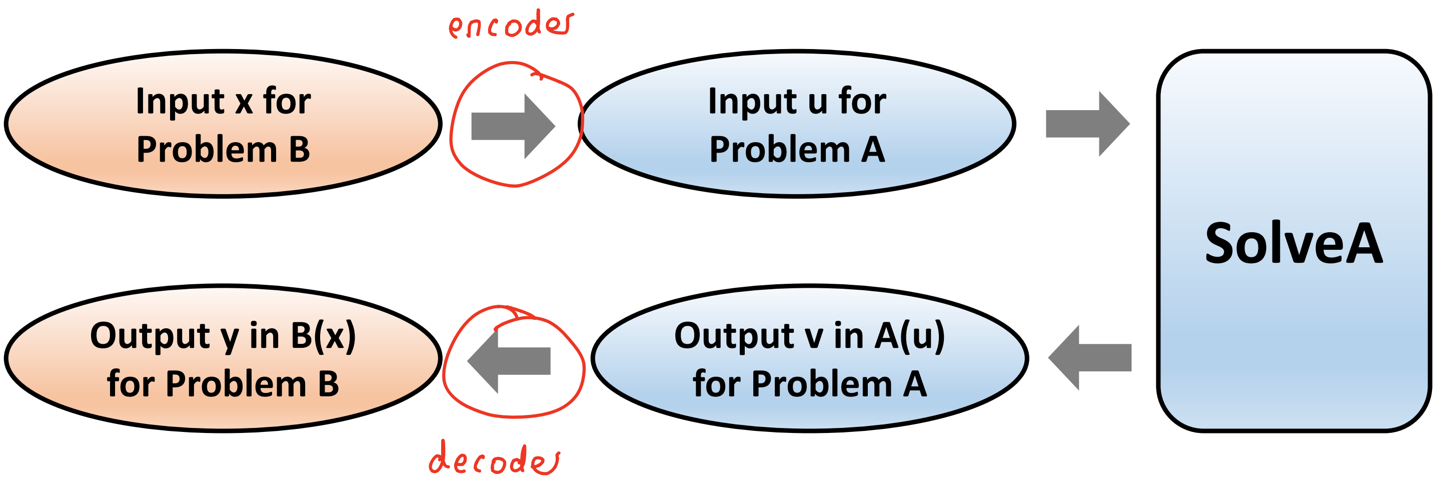
legal inputs: $x = (V, E, s, t, \{c(e)\})$

- $E \subseteq V \times V$
- $s, t \in V$
- $c(e) \in \mathbb{N}$ for every $e \in E$

legal outputs for x: Any flow f that maximizes $val(f)$

- satisfies capacity, conservation
- $f(e) \in \mathbb{Z}$

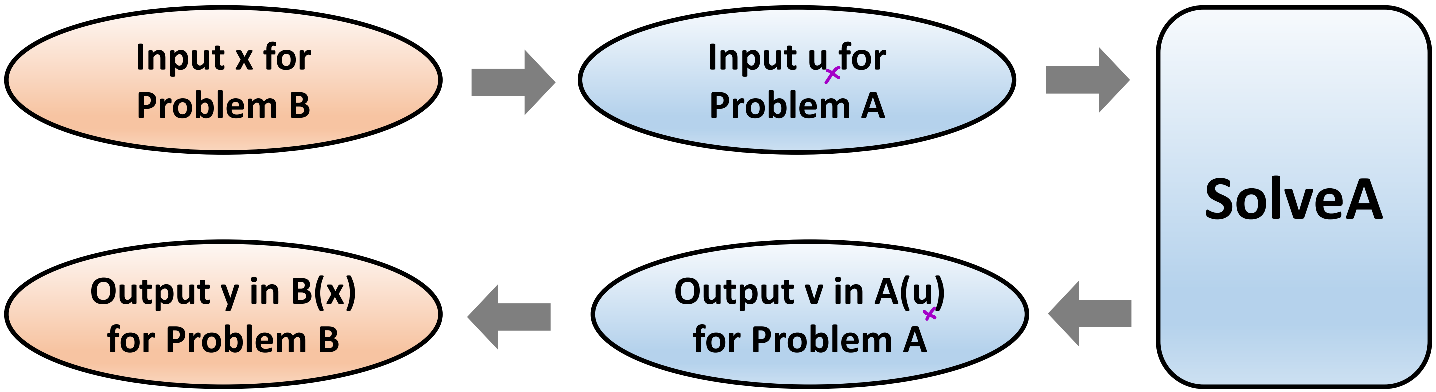
Mechanics of Reductions



- Your job is to design the two pieces in red.

If u is a legal input for A, then v is in the set of legal outputs A(u).

When is a Reduction Correct?



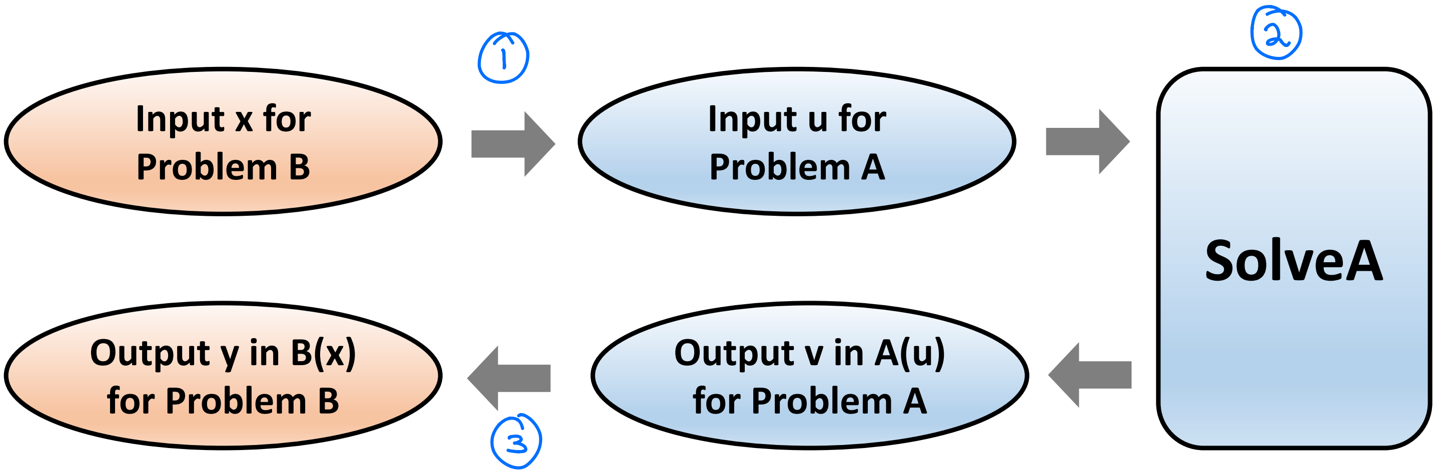
① Assume that SolveA is a correct algorithm for A
(do not assume anything else)

② Argue that for every legal x for B, u is legal for A

③ Argue that for every x, and every $v \in A(u)$, $y \in B(x)$

u_x
depends on x

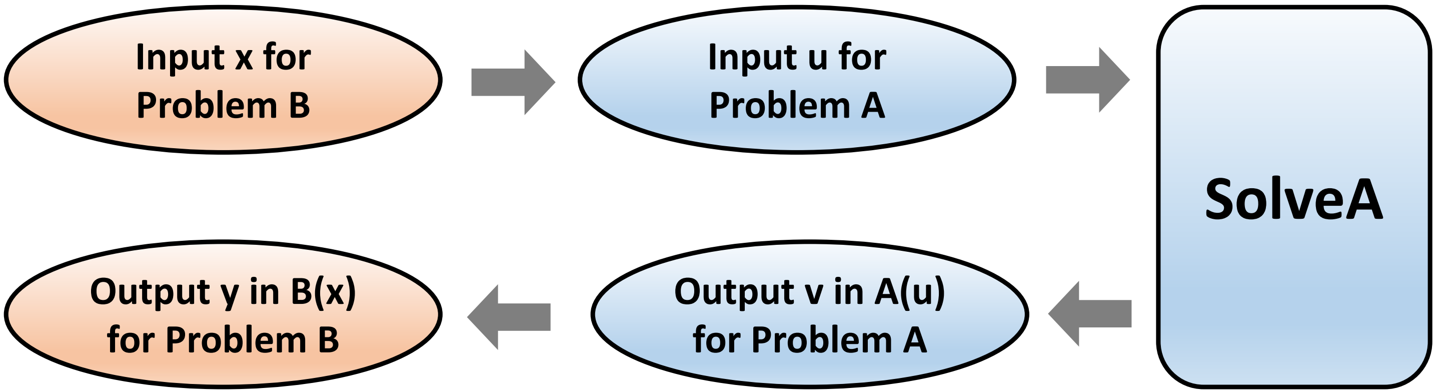
What is the Running Time?



Total time: 1 + 2 + 3

- 2 is the time to solve A on an input of size $|u|$

Example: Minimum Cut

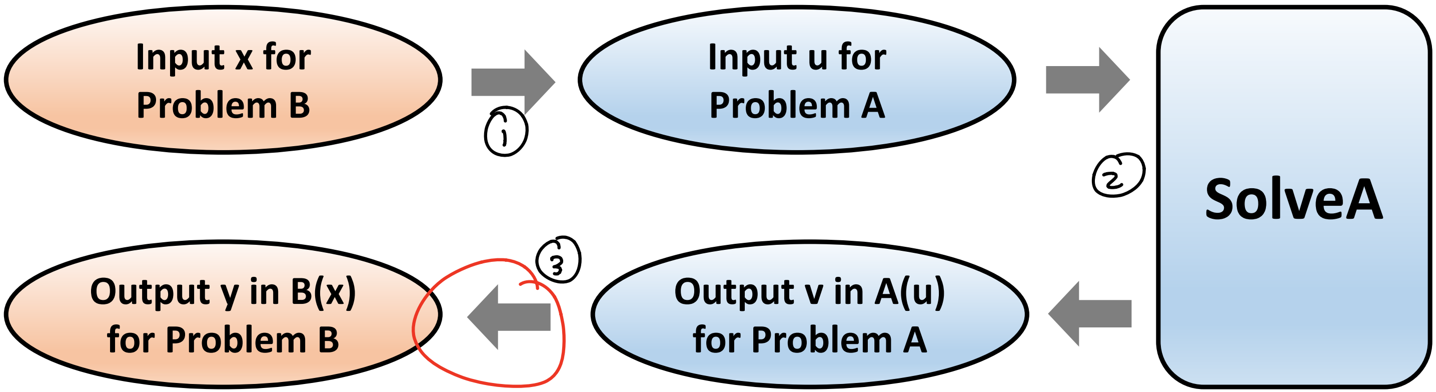


Problem B is min cut → Given $G=(V,E,s,t,\{c(e)\})$
Find a min s-t cut (A,B)

Problem A is max flow → Given $G=(V,E,s,t,\{c(e)\})$
Find a max s-t flow f

Example: Minimum Cut

$G = (V, E, s, t, \{c(e)\})$ $\qquad$ $G = (V, E, s, t, \{c(e)\})$



max flow f for G

① Find residual graph G_f

② Run BFS from s to find nodes reachable in G_f, call it A

③ $B = V \setminus A$

Running Time:

① $O(m+n)$ to write down the graph

③ $\underbrace{O(m)}_{\text{find } G_f} + \underbrace{O(n+m)}_{\text{BFS in } G_f} + \underbrace{O(n)}_{\text{write } A, B} = O(m+n)$

② Time to find max flow on a graph w/ n nodes and m edges. $O(mn)$ time.

$O(mn)$

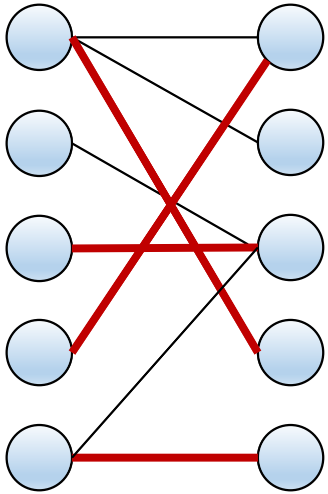
Bipartite Matching

- **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 = $|M|$

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

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

Similar to stable matching



Bipartite Matching

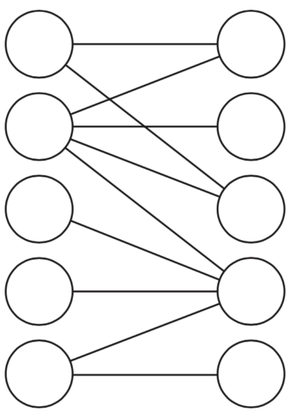
- There is a reduction that uses integer maximum s-t flow to solve maximum bipartite matching.

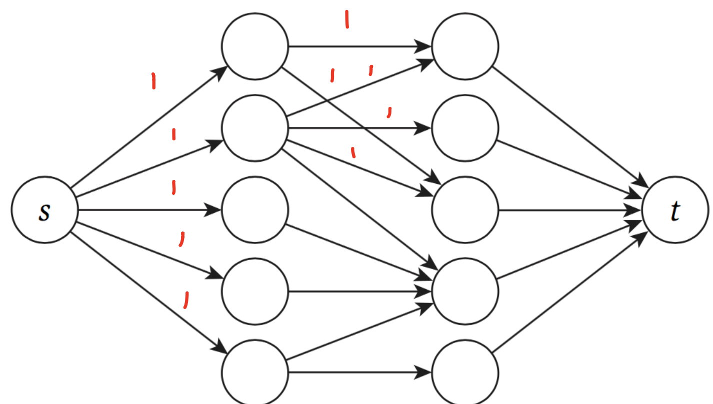
Step 1: Transform the Input

time $O(m+n)$

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

$c(e) = 1$ for every $e \in E$

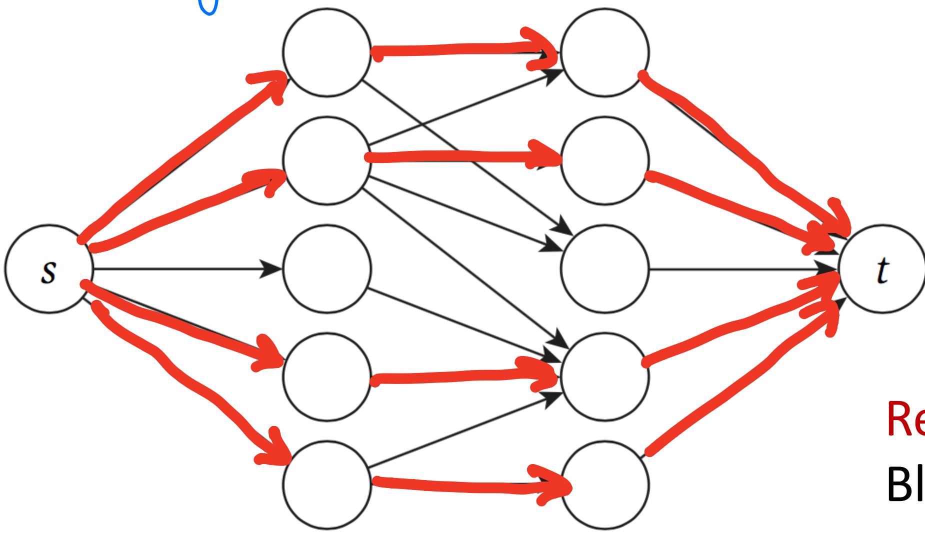


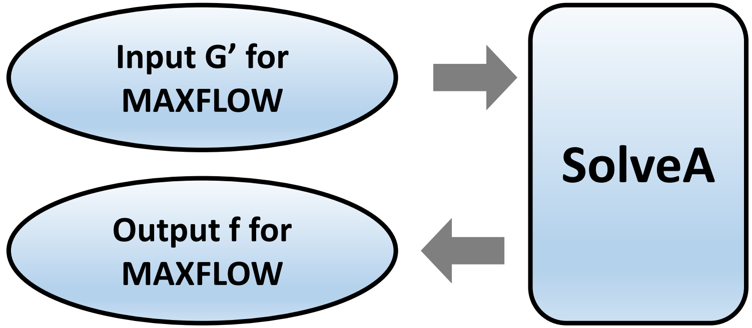


$n' = n+2$ $\quad$ $m' = m+n$

Step 2: Receive the Output

Every node on the left has ≤ 1 unit leaving. (Similar for the right)





Integer Max Flow

Red arrow means f(e)=1

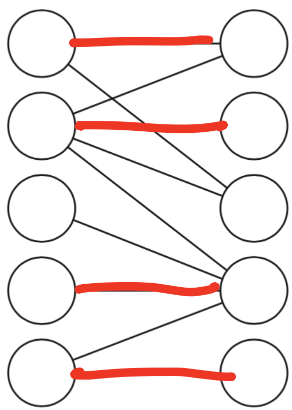
Black means f(e) = 0

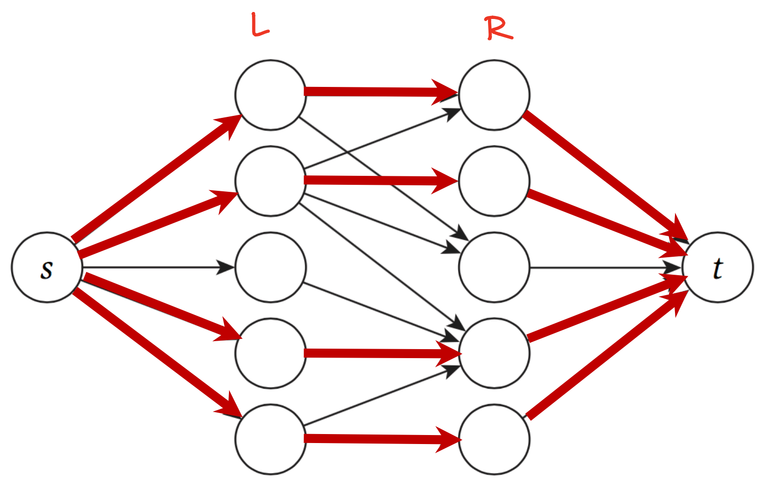
Step 3: Transform the Output

$O(m+n)$

Output M for MCBM ← **Output f for MAXFLOW**

M is all edges from L to R s.t. $f(e)=1$





Reduction Recap

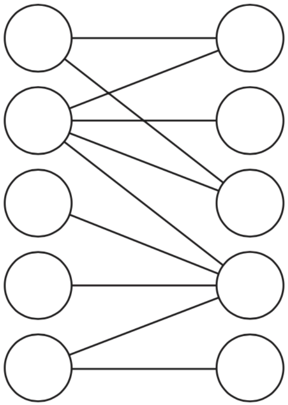
- Step 1: Transform the Input
  - Given G = (L,R,E), produce G’ = (V,E,{c(e)},s,t) by...
    - ... orient edges e from L to R
    - ... add a node s with edges from s to every node in L
    - ... add a node t with edges from every not in R to t
    - ... set all capacities to 1
- Step 2: Receive the Output
  - Find an integer maximum s-t flow 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

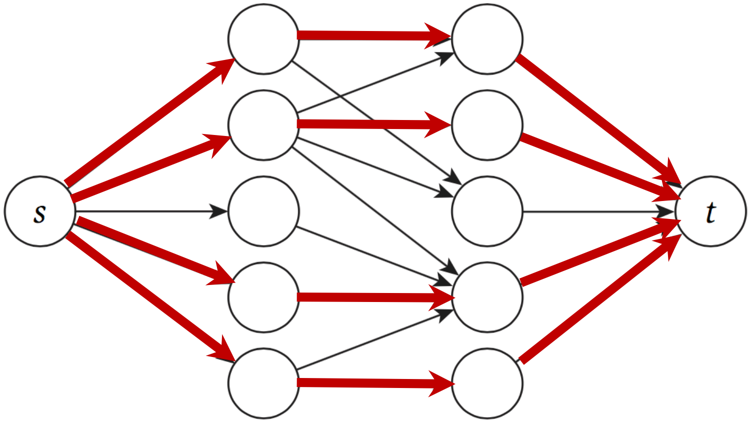
Correctness

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

Correctness

- This algorithm returns a matching

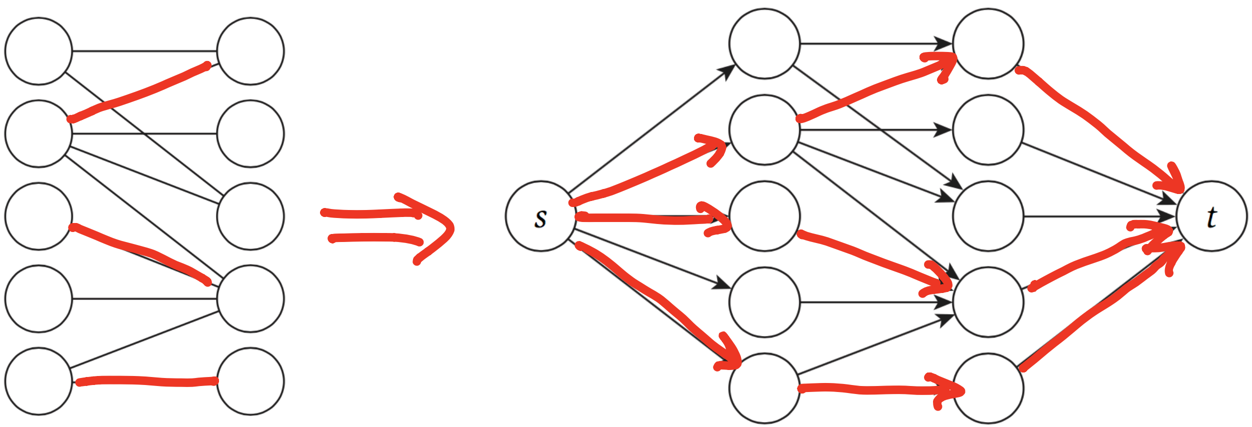




Correctness

- **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
- Proof ($\Longrightarrow$):

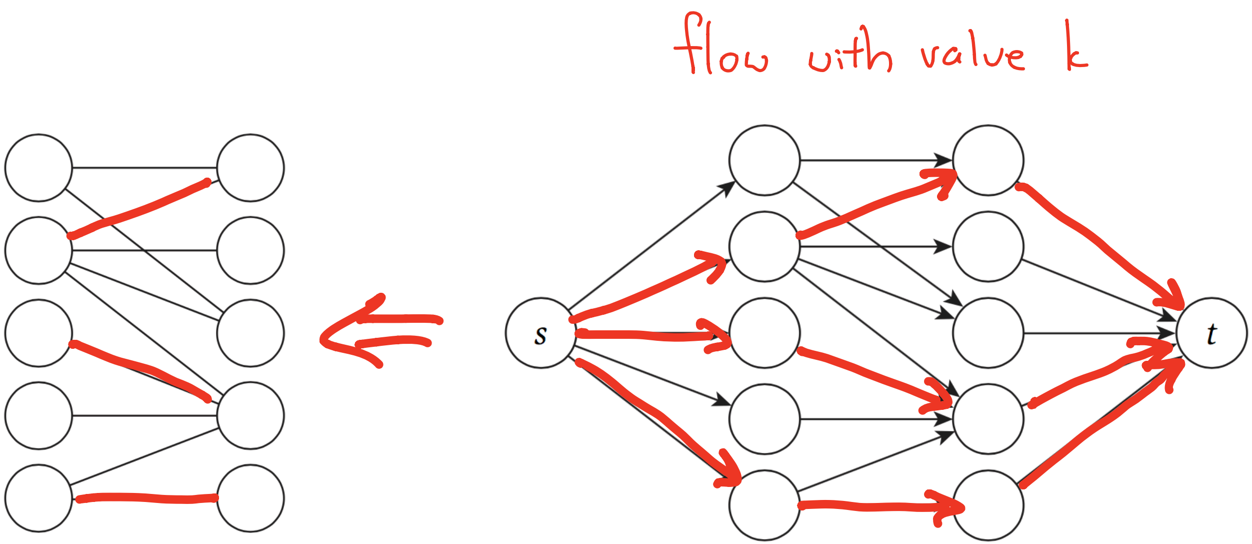
matching w/ k edges



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

- Proof ($\Leftarrow$):



Running Time

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

$T(n', m') = T(n+2, m+n)$

T is time to solve max flow in graph with n' nodes, m' edges

$T(n', m') = O(n'm') = O((n+2)(m+n))$

$= O(mn + 2m + 2n + n^2) = O(mn)$

Summary

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 maximum bipartite matching in time O(nm) using Ford-Fulkerson
  - Improvement for maximum flow gives improvement for maximum bipartite matching!