CS3000: Algorithms & Data
Jonathan Ullman

Lecture 20:

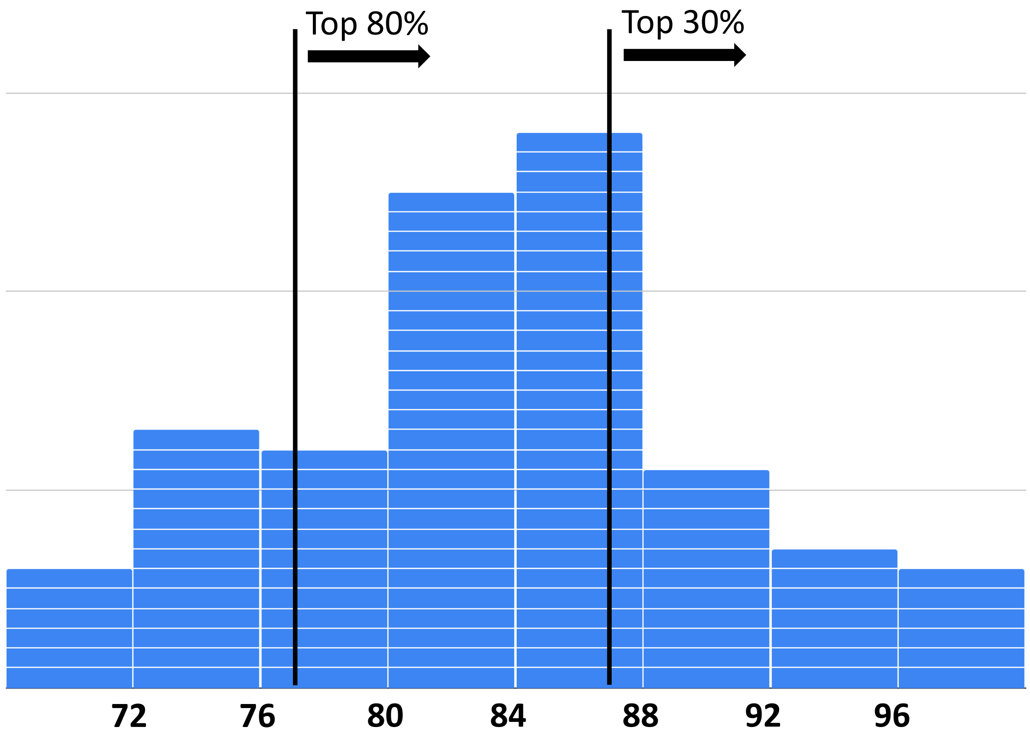
- Applications of Network Flow

Nov 27, 2018

Midterm II Stats

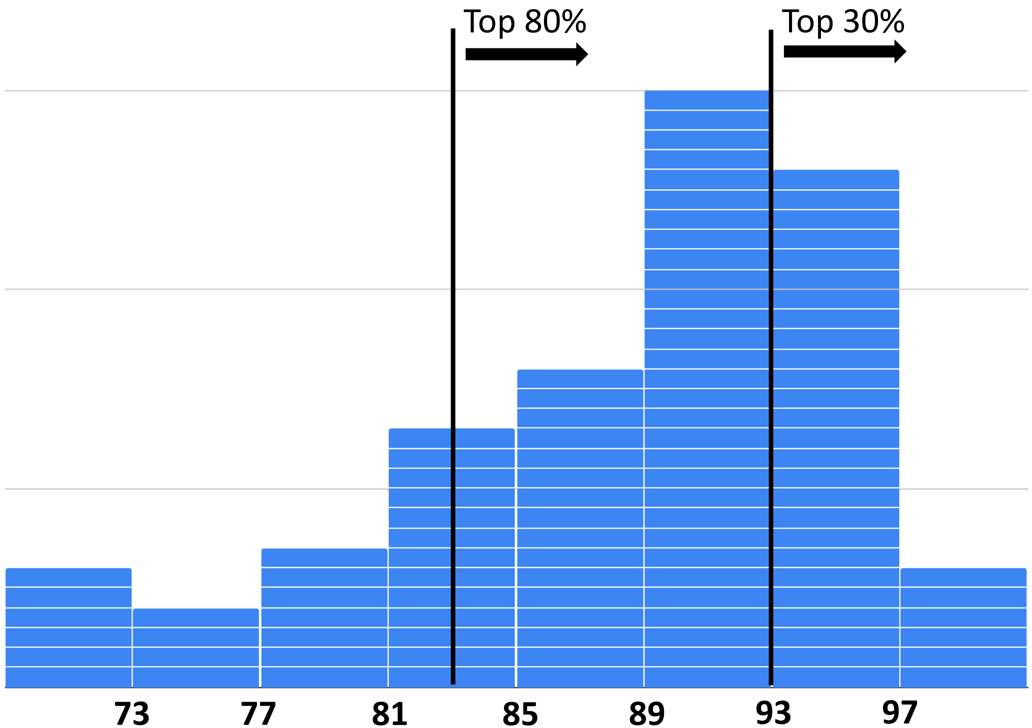
Midterm II Grade Distribution

Mean ≈ 83



Midterm II Grades were really good!

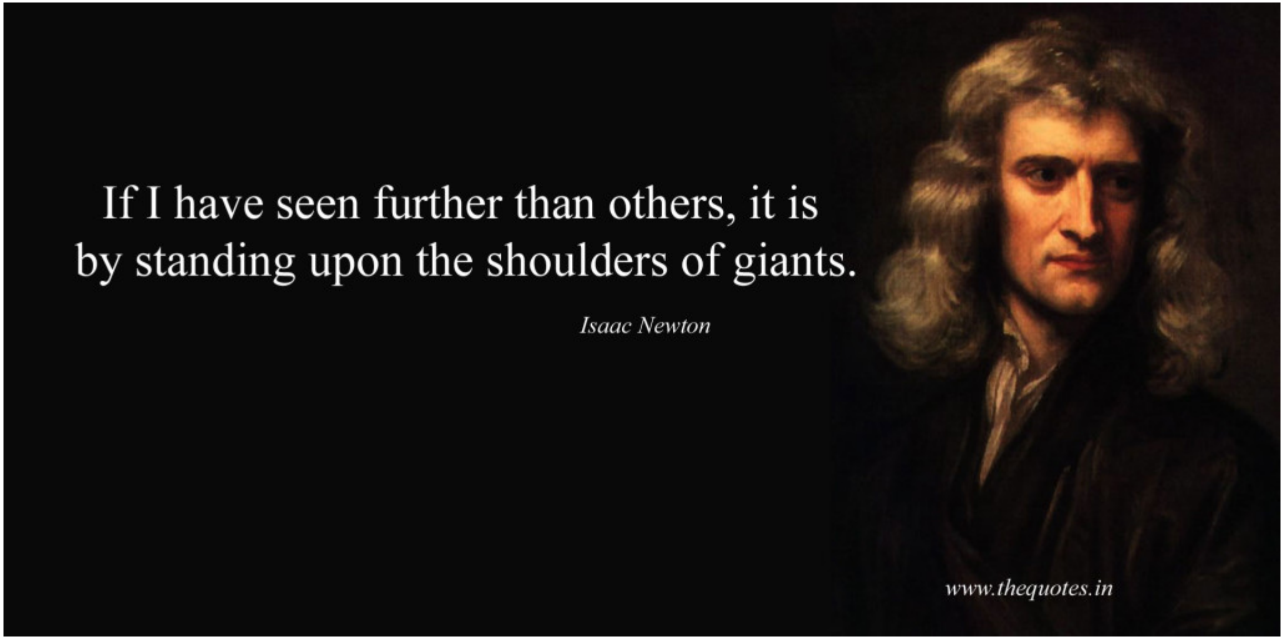
Homework Grade Distribution



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.

Mechanics of Reductions

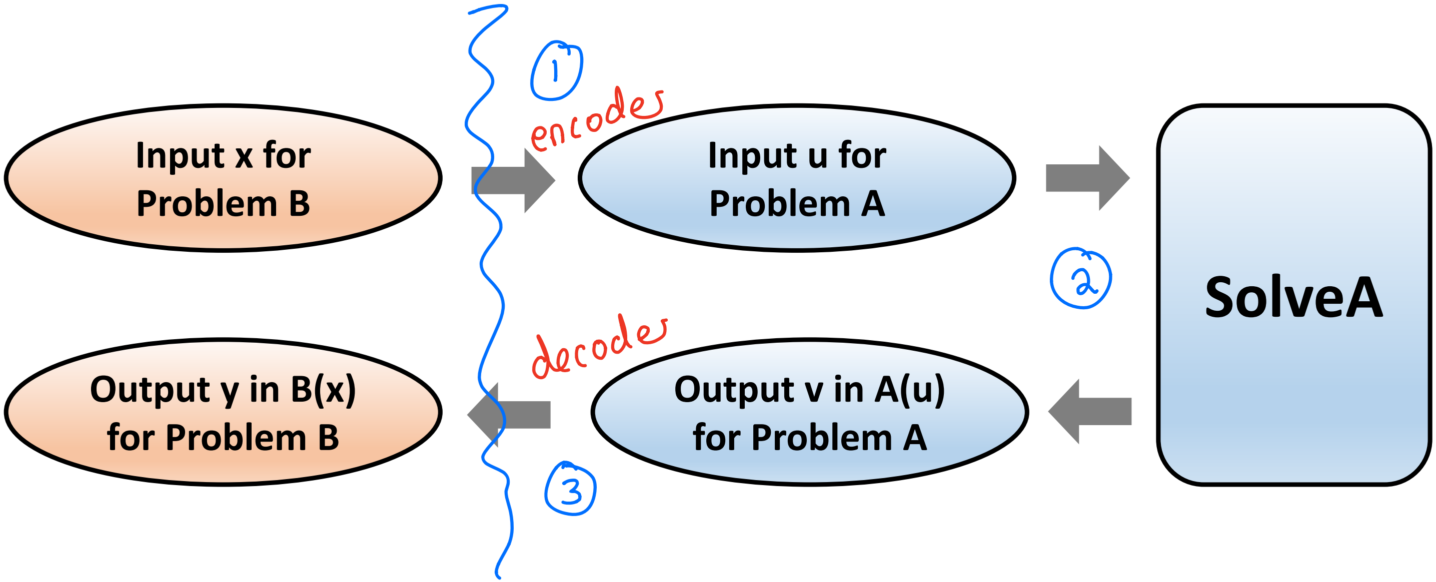
- What exactly is a **problem**?
  - A set of legal inputs $\boldsymbol{X}$
  - A set $\boldsymbol{A(x)}$ of legal outputs for each $\boldsymbol{x} \in \boldsymbol{X}$

- **Example:** integer maximum flow

$$X = \left\{ \begin{array}{l} \text{flow networks } G = (V, E, s, t, \{c(e)\}) \\ \text{such that } c(e) \in \mathbb{Z},\ c(e) \geq 0 \text{ for every } e \end{array} \right\}$$

$$A(G) = \left\{ \begin{array}{l} \text{maximum } s\text{-}t \text{ flows } f \text{ in } G \\ \text{such that } f(e) \in \mathbb{Z} \text{ for every } e \end{array} \right\}$$

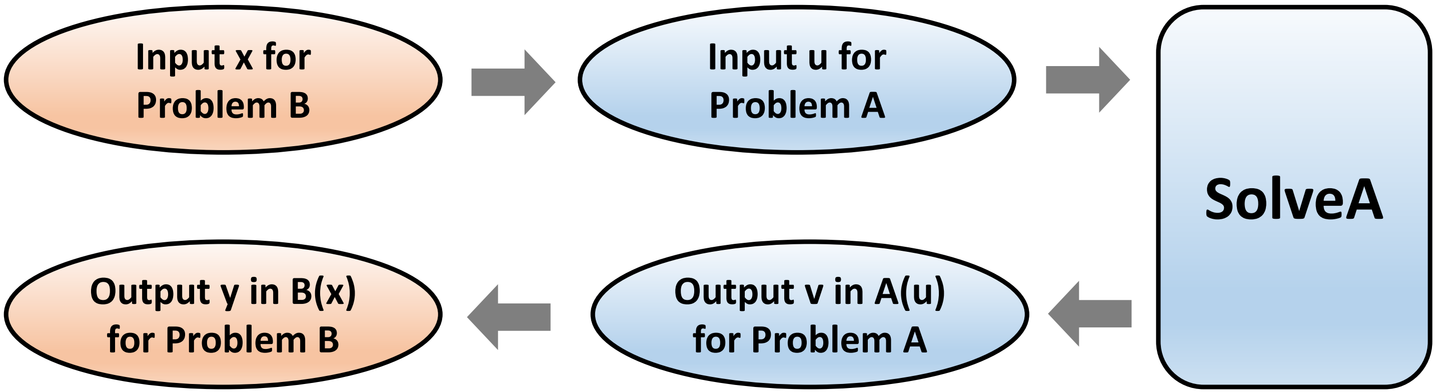
Mechanics of Reductions



1. Transform the input
2. Solve problem A
3. Transform the output

You design these

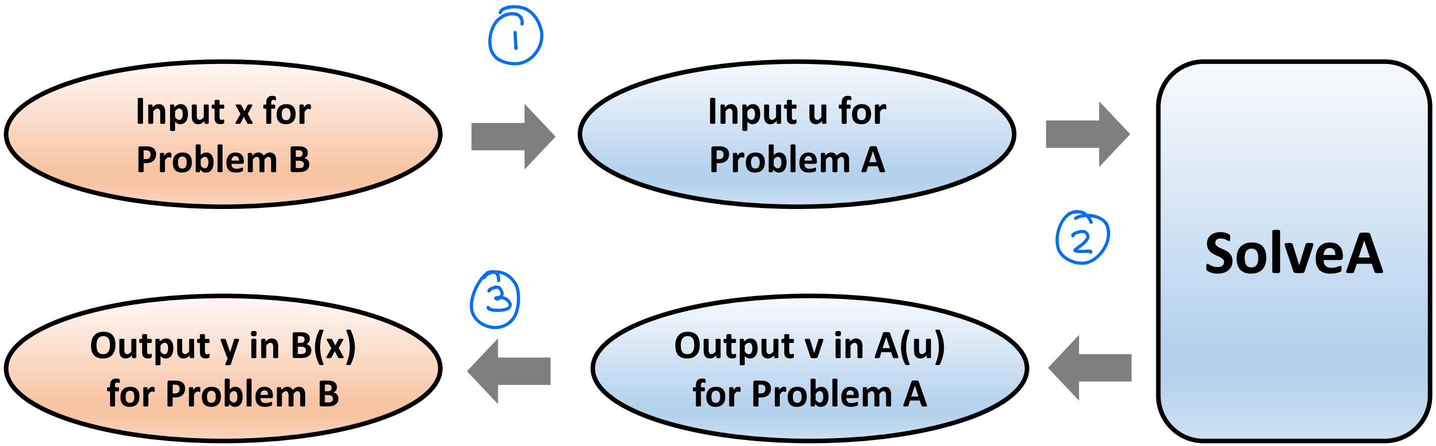
When is a Reduction Correct?



If SolveA is correct for A, then the algorithm is correct for B

1. Assume SolveA is correct (but nothing else)
2. Argue that for every legal x for B, $u(x)$ is legal for A
3. For all x, and all $v \in A(u(x))$, $y \in B(x)$

What is the Running Time?



Total Running Time: 1 + 2 + 3

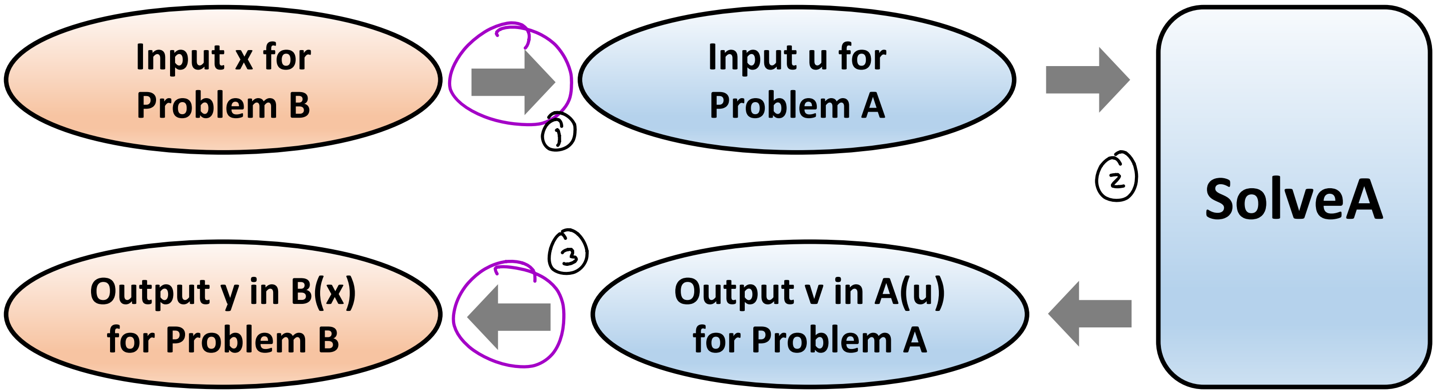
caveat: time for 2 depends on $|u|$, but we need to state the running time as a function of $|x|$.

Example: Minimum Cut

Problem B is mincut
Problem A is maxflow

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

$G' = (V, E, s, t, \{c(e)\})$



a maximum s-t flow f for G'

1. compute G_f

2. let A be all nodes reachable from s in G_f

3. let $B = V \setminus A$

Running Time $O(mn)$

1. $O(m+n)$

3. $O(m+n)$

2. Time to solve maxflow w/ n nodes, m edges $O(mn)$

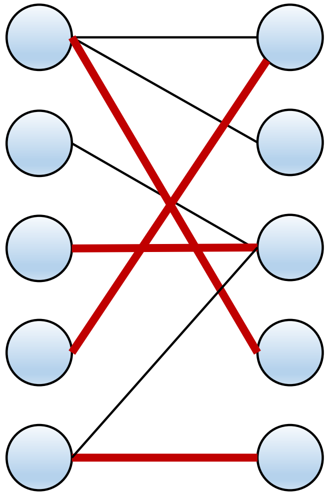
Bipartite Matching

- **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

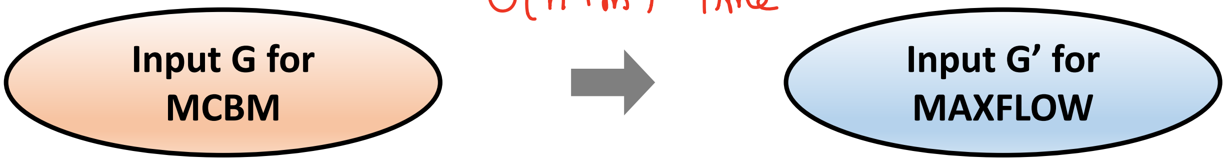


Bipartite Matching

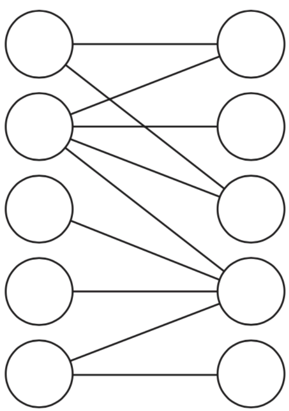
- There is a reduction that uses integer maximum s-t flow to solve maximum bipartite matching.

Step 1: Transform the Input

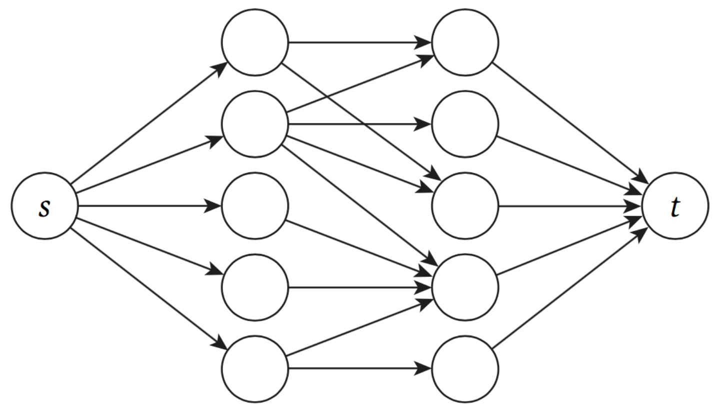
$O(n+m)$ time



all edges have capacity 1



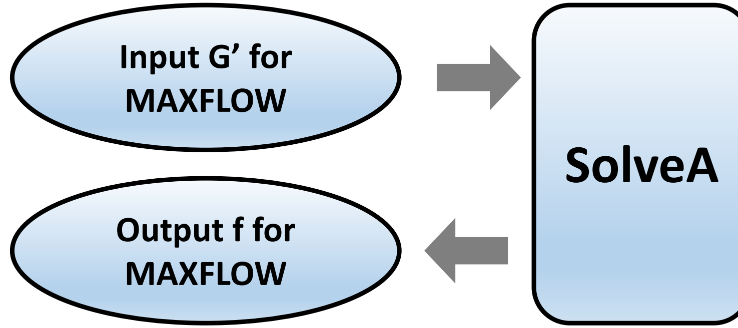
n nodes, m edges

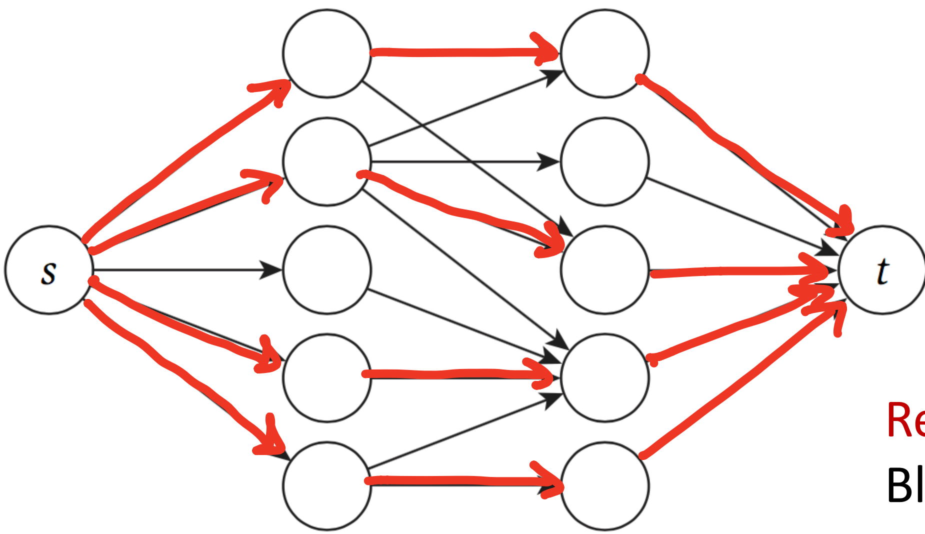


n+2 nodes, n+m edges

Step 2: Receive the Output

Can be any maximum flow in G'





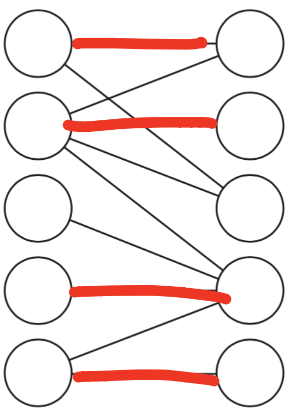
Red arrow means f(e)=1

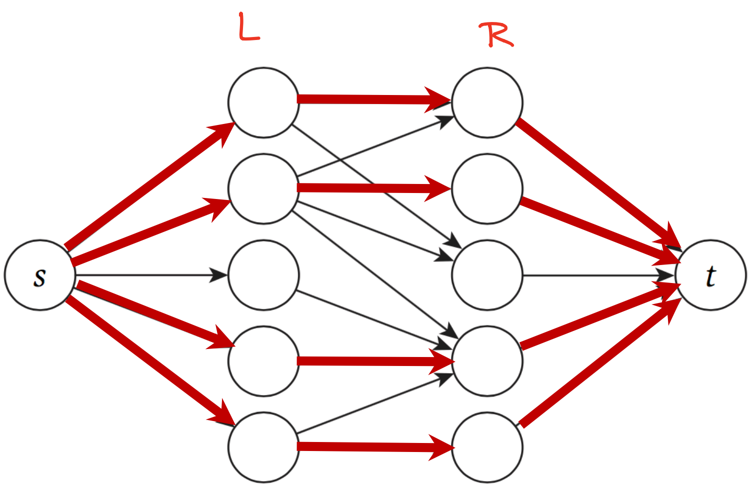
Black means f(e) = 0

Step 3: Transform the Output

Output M for MCBM ← $O(m+n)$ **Output f for MAXFLOW**

$M = \{ (u,v) \text{ s.t. } f(u,v) = 1 \text{ and } u \in L, v \in R \}$





Observation $|M| = val(f)$

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

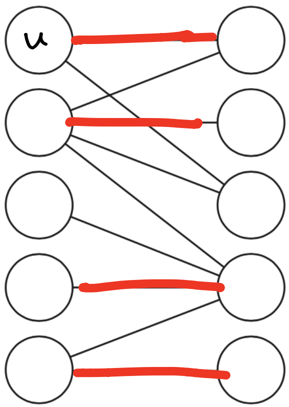
Correctness

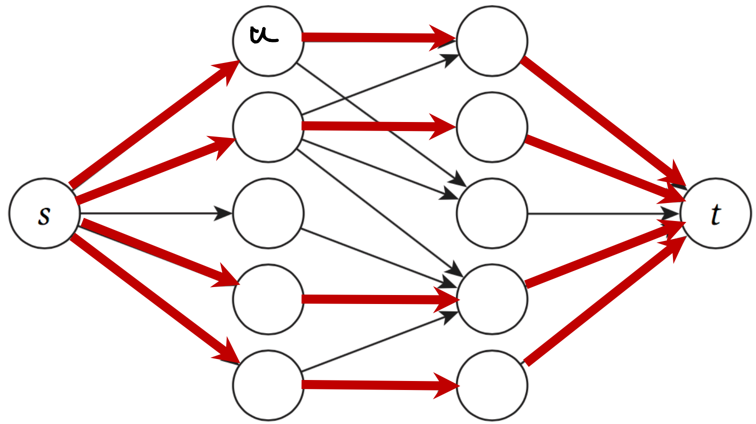
- Need to show:
  - (1) This algorithm returns a matching
  - (2) This matching is a maximum cardinality matching

(Assuming f is a maximum flow)

Correctness

- This algorithm returns a matching





Suppose u is the endpoint of > 1 red edge

⇒ two red edges leave u in f

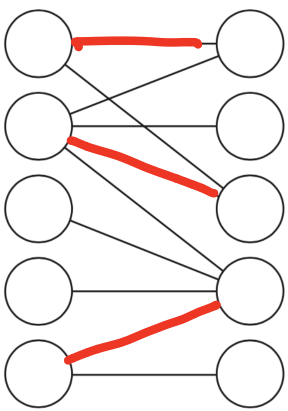
⇒ two red edges enter u in f

⇒ contradiction

Correctness

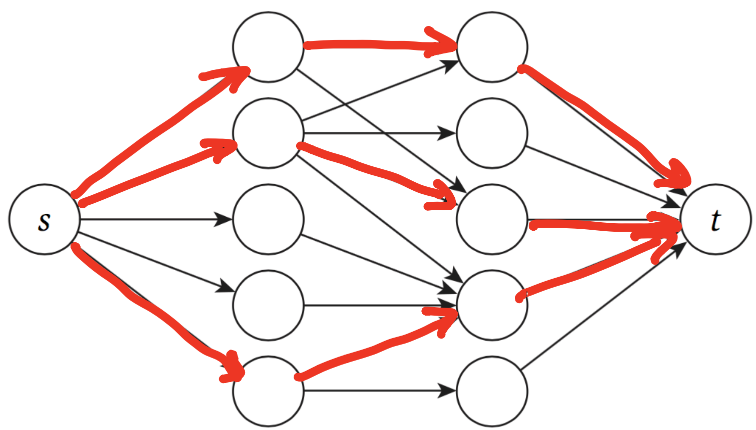
- **Claim:** G has a matching of cardinality at least k if and only if G’ has an s-t flow of value at least k
- Proof ($\Rightarrow$)

matching M with k edges



$\Longrightarrow$

flow of value k

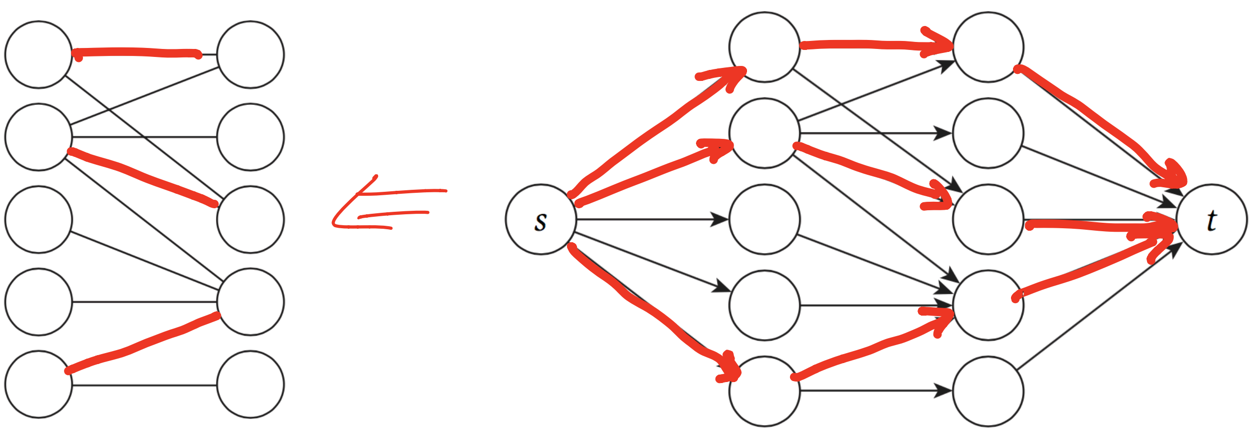


Correctness

- **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
- Proof ($\Leftarrow$)

matching M with k edges

flow of value k



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' $O(m+n)$

<u>Fact</u>: Can find a max flow in G' in time $O(n'm')$

n' = nodes in G'

m' = edges in G'

$$\Rightarrow O((n+2)(m+n)) = O(nm + \cancel{2m} + \cancel{2n} + n^2)$$

$$= O(nm)$$

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!