HW9 is online

HW10 is the last HW

Midterm II back at the end of class.

CS4800: Algorithms & Data
Jonathan Ullman

Lecture 19:
Applications of Network Flow

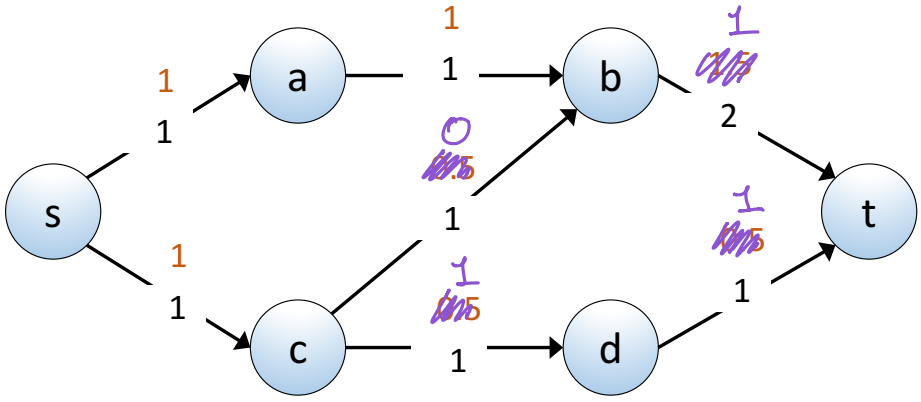
- Reductions
- Bipartite Matching

Mar 30, 2018

Ask the Audience

val(f) = 2

(a) Is this a maximum s-t flow? Yes! cap({s}, {s,a,c}) = 2



(b) Does this network have an integer max s-t flow? Yes!

(c) Does every graph have an integer max flow? No!

(d) Does every graph with integer capacities have an integer max flow?

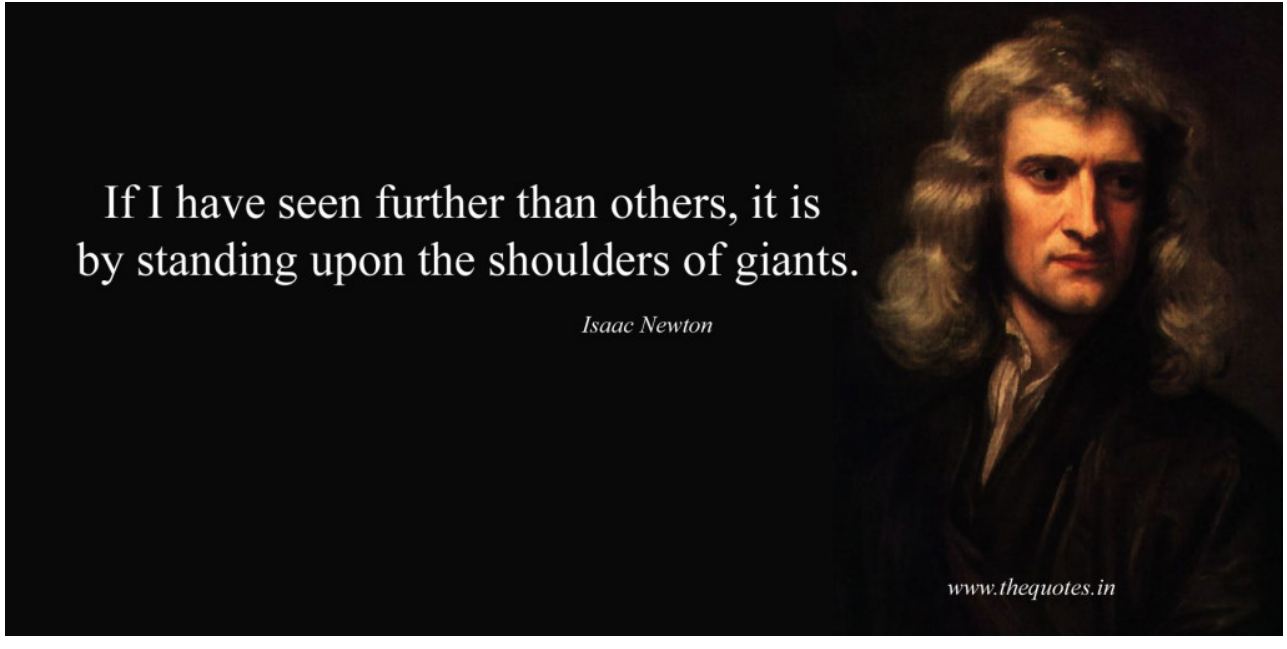
Claim: Given any flow network $G = (V, E, \{c(e)\}, s, t)$ with integer capacities. FF will find a maximum s-t flow s.t. the flow on every edge is an integer.

"Proof": By induction. After each augmenting path the flow is an integer flow.

Recap

- Can solve maximum s-t flow in a flow network $G=(V,E,\{c(e)\},s,t)$ in $O(mn)$ time
  - If all capacities are integers, we get an integer flow
- Can find a minimum s-t cut in $O(mn)$ time

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
  - …

Reductions

There is an efficient algorithm that solves Problem B using calls to the library function that solves Problem A.

MAXFLOW

Mechanics of Reductions

- What exactly is a computational problem?
  - A set of legal inputs x (An array A[1], A[2], ...)
  - A set A(x) of legal outputs for each x (A in sorted order)
- Example Integer Maximum Flow

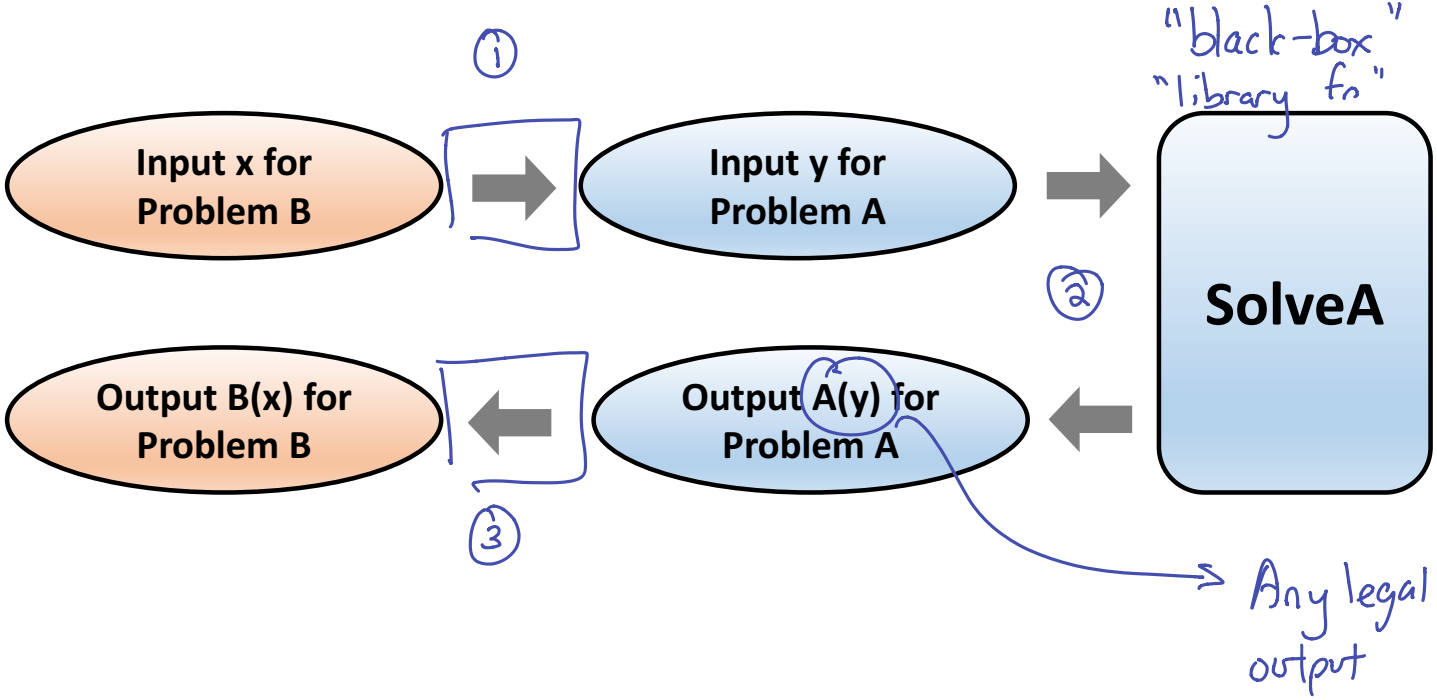
$$x : \{ G = (V, E), \{c(e)\}, s, t \}$$

(directed graph; integer cap's; source; sink)

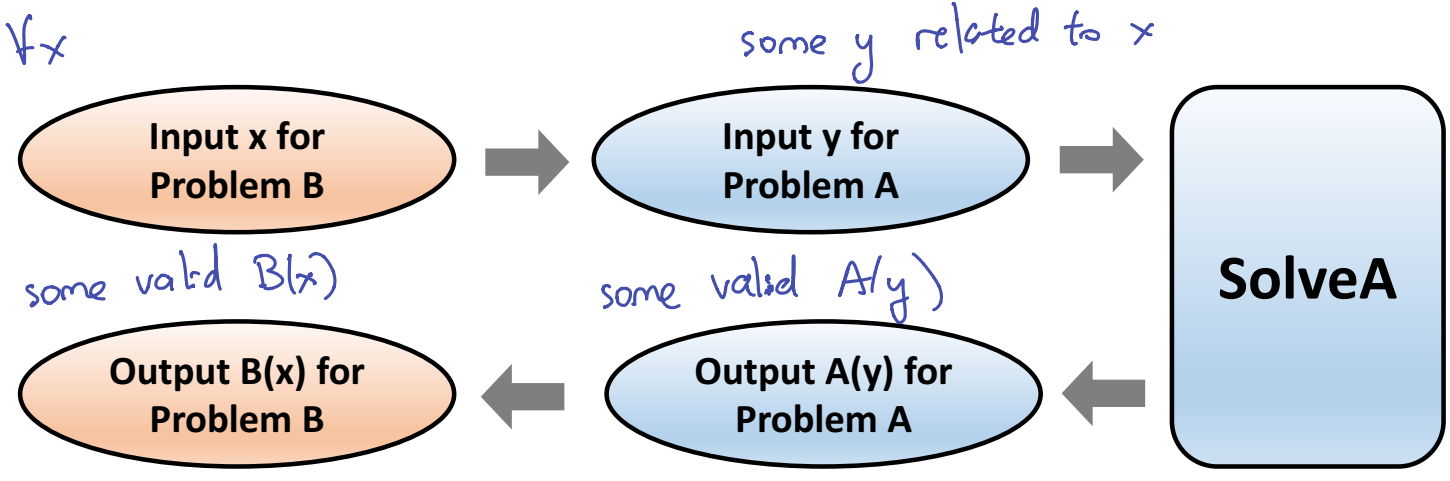
$$A(x) = \{ \{f(e)\} : f \text{ satisfies non-negativity, conservation, capacity; } val(f) \text{ is maximum} \}$$

integer flow

Mechanics of Reductions

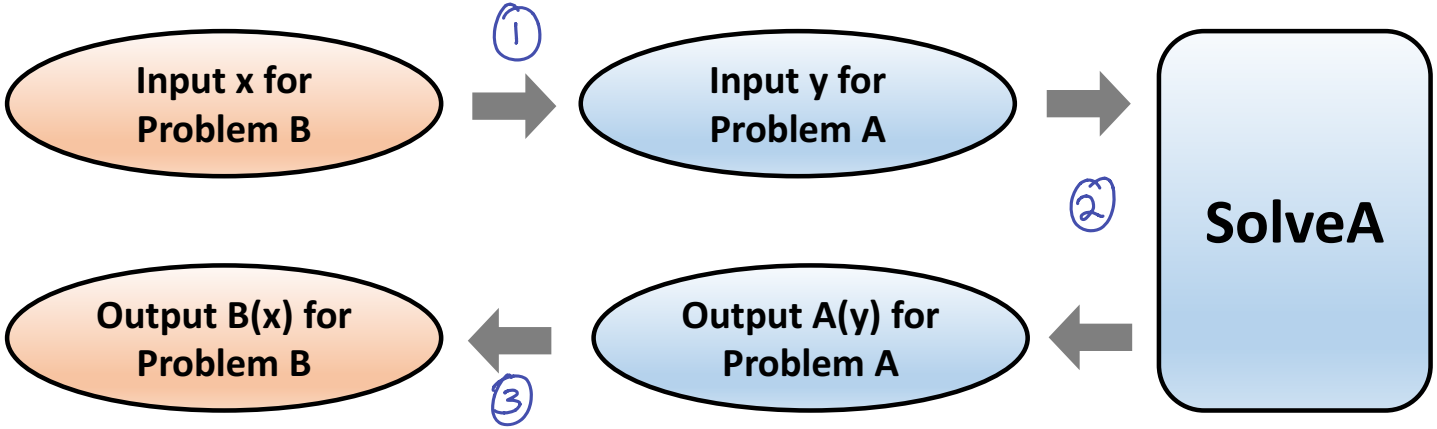


When is a Reduction Correct?



For every x, if the output of SolveA is a valid output for y, then B(x) is a valid output for B.

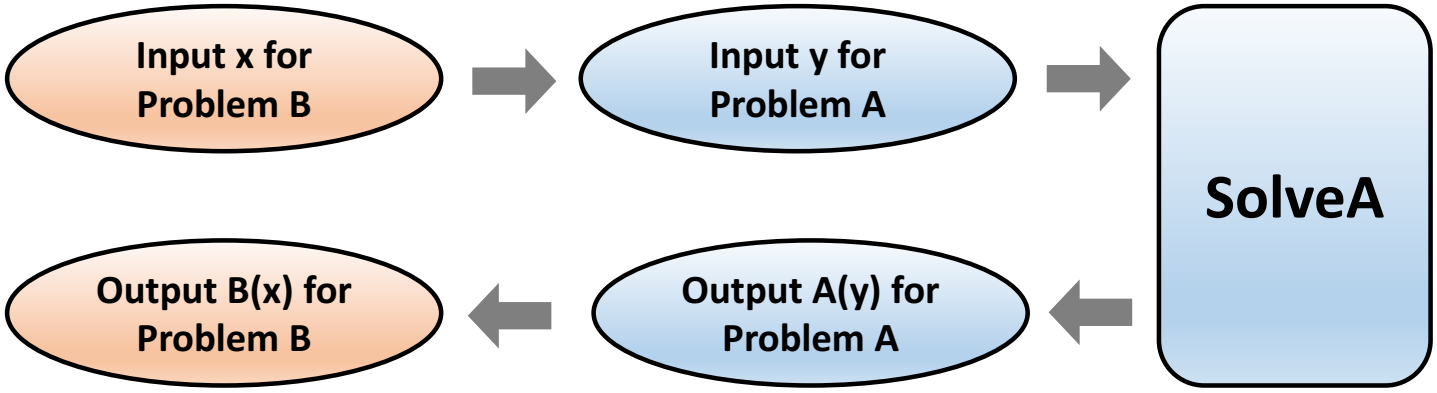
What is the Running Time?



Running Time : ① + ② + ③

- How long does it take to compute y given x?
- How long does SolveA take to solve y?
- How long does it take to compute B(x) given A(y)?

Retconned Example: Minimum Cut



B = MIN S-T CUT

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

A = MAX S-T FLOW

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

B(x): ① compute G_{f^*}

② find $A = \{ v \text{ reachable from } s \text{ in } G_{f^*} \}$

③ output A

$A(y) =$ a max flow f^*

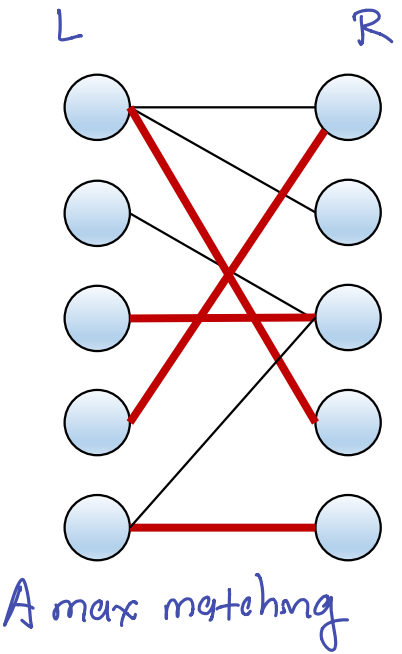
Bipartite Matching

undirected, unweighted

- Input: bipartite graph $G = (V, E)$ with $V = L \cup R$
- Output: a maximum cardinality matching
  - A matching M is a subset of E 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:

- employees to employers
- jobs to processors
- advertisements to websites



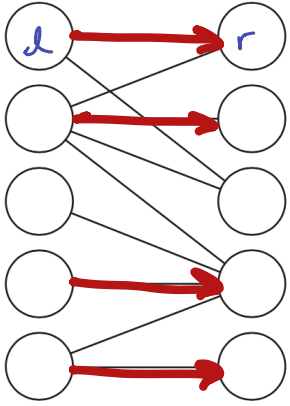
Bipartite Matching

- Theorem: can solve maximum bipartite matching using an algorithm that solves integer max flow

Step 1: Transform the Input

Input G for MCBM

$G = (V, E)$

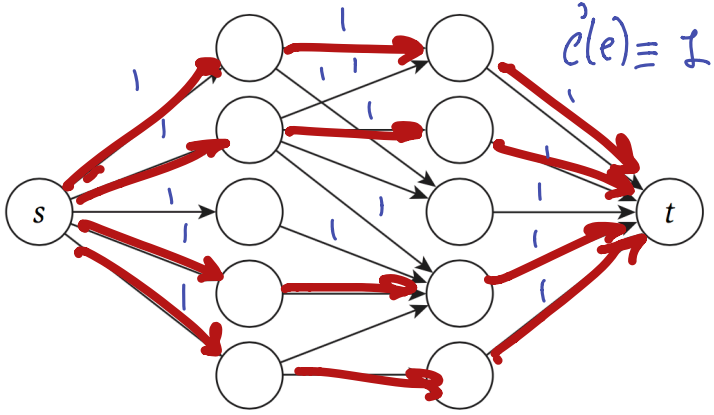


Input G' for MAXFLOW

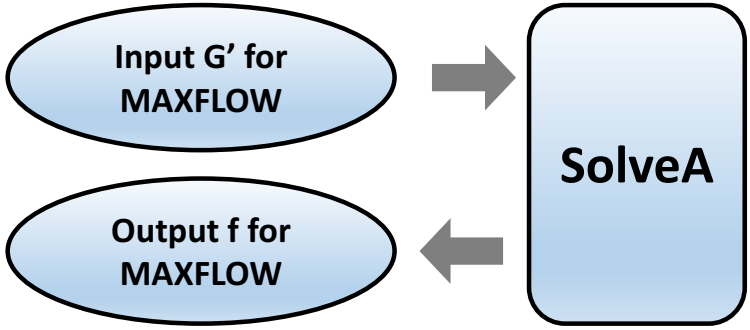
G' $V' = V + \{s, t\}$

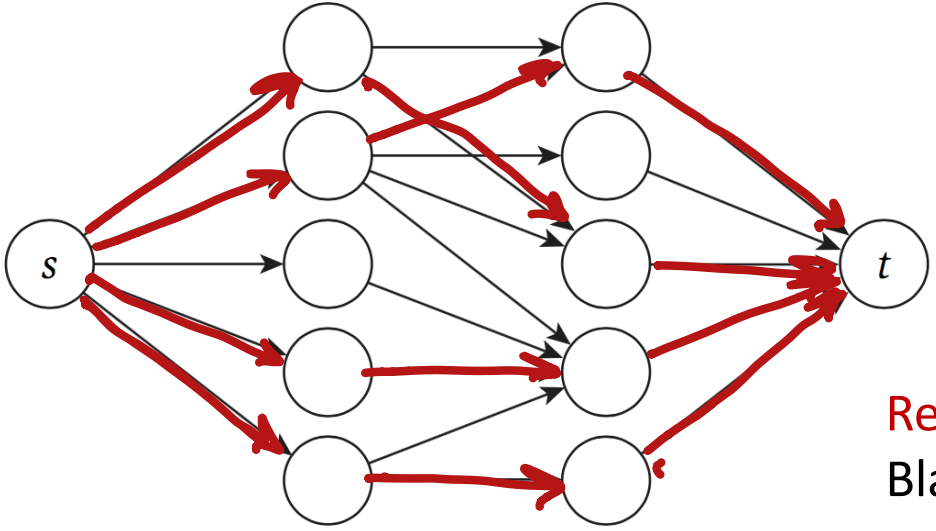
E' = for every $(l, r) \in V$
add $l \to r$
add $s \to l$, $r \to t$

$c'(e) = 1$



Step 2: Receive the Output

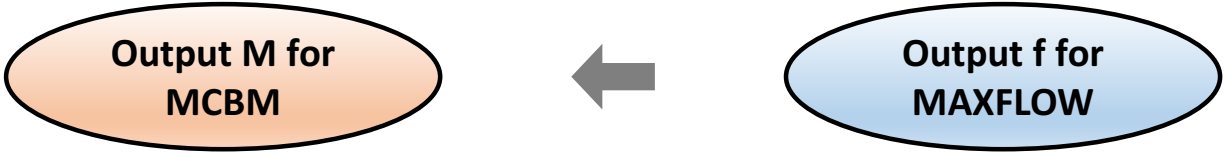




Red arrow means f(e)=1
Black means f(e) = 0

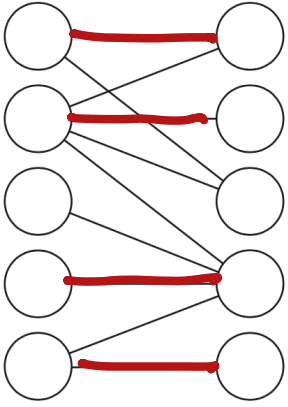
all c(e)=1

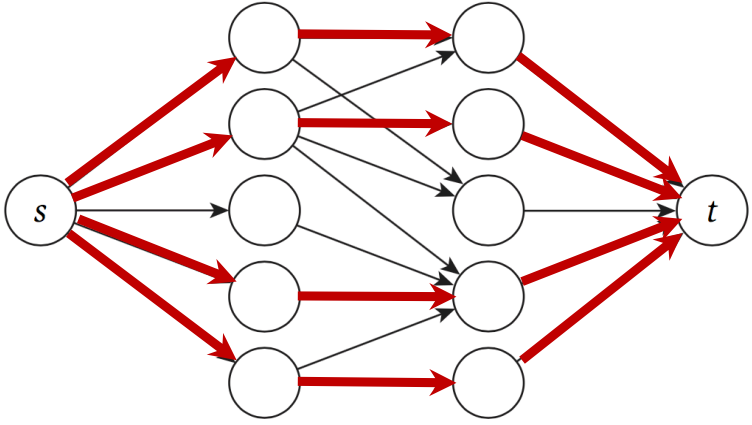
Step 3: Transform the Output



$\forall\ e=(l \to r)$, if $f(e)=1$, add (l,r) to M

Output M.





Reduction Recap

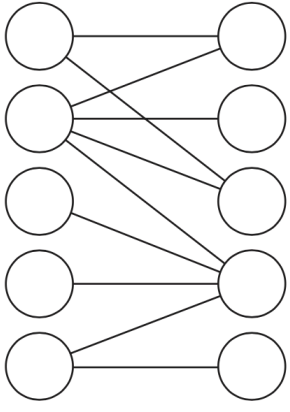
- Step 1: Transform the Input
  - Given G = (L,R,E), produce G' = (V,E,{c(e)},s,t) by...
    - ... orient edges e from L to R
    - ... add a node s with edges from s to every node in L
    - ... add a node t with edges from every not in R to t
    - ... set all capacities to 1
- Step 2: Receive the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

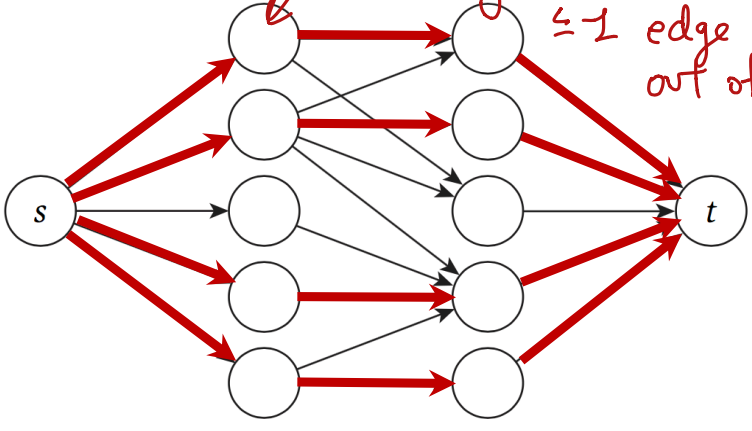
Correctness

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

Correctness

- This algorithm returns a matching





∀ nodes in L, flow in = flow out. ≤ 1 edge coming into v ⇒ ≤ 1 edge going out of v.

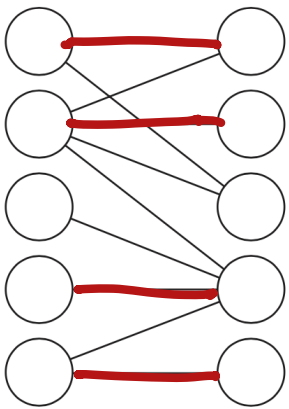
Clm: If M is the matching from our reduction, then no vertex touches two edges.

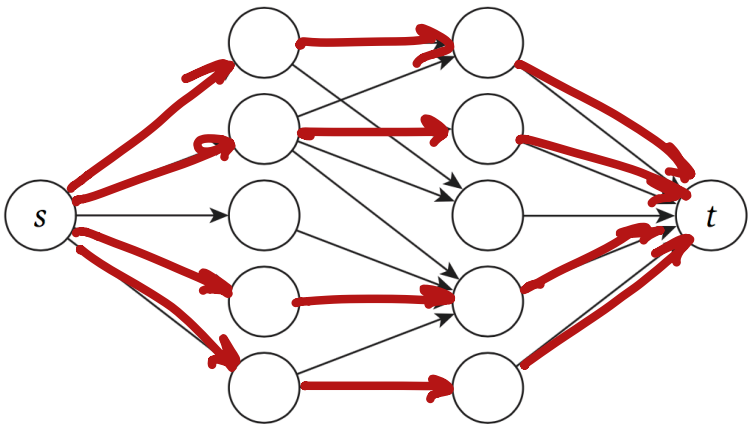
Correctness

- 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

matching $\Rightarrow$ flow

MAXFLOW $\geq$ MAXMATCHING





Correctness

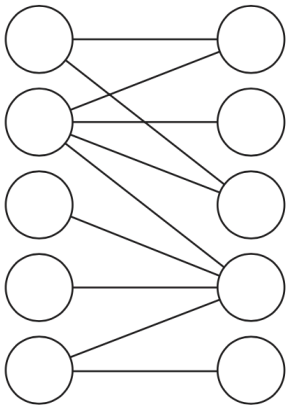
- Claim: G has a matching of cardinality at least k if and only if G' has an s-t flow of value at least k

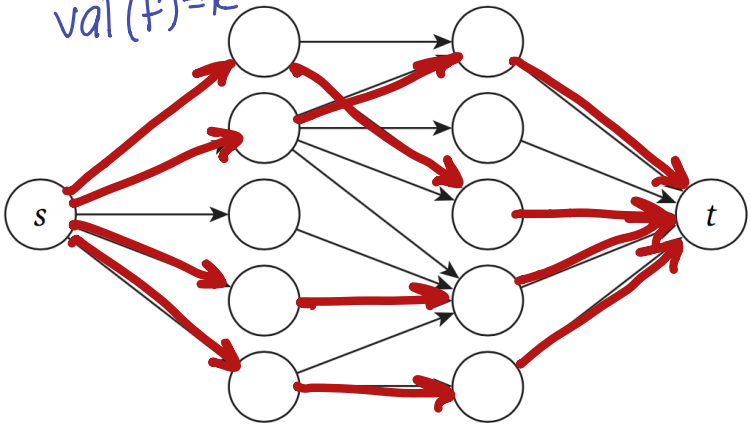
flow ⟹ matching

MAXMATCH ≥ MAXFLOW

- there are k flow carrying edges leaving s, each carrying one unit of flow
- therefore the are k L→R edges w/ f(e) = 1
- therefore M has size k

val(f) = k





Running Time

Total Time = $O(mn)$

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

① $O(n)$ to create nodes of G', $O(m+n)$ to create edges of G',
$O(m+n)$ to create capacities, $O(1)$ to designate source/sink

② G' has n' nodes, m' edges $n' = n+2$ $m' = m+n$

Time is $O(m'n') = O((m+n)(n+2)) = O(mn+n^2+m+n)$
$= O(mn+n^2)$
$= O(mn)$

③ $O(m)$ 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 maximum bipartite matching in time O(nm) using Ford-Fulkerson
  - Improvement for maximum flow gives improvement for maximum bipartite matching!

State-of-the-art is $O(m^{10/7})$?

Midterm II Grades

Mean ≈ 71/100