Lecture 17:
Max Flow → Bipartite Matching

Tim LaRock

larock.t@northeastern.edu

bit.ly/cs3000syllabus


THIS IS FINE.

Business

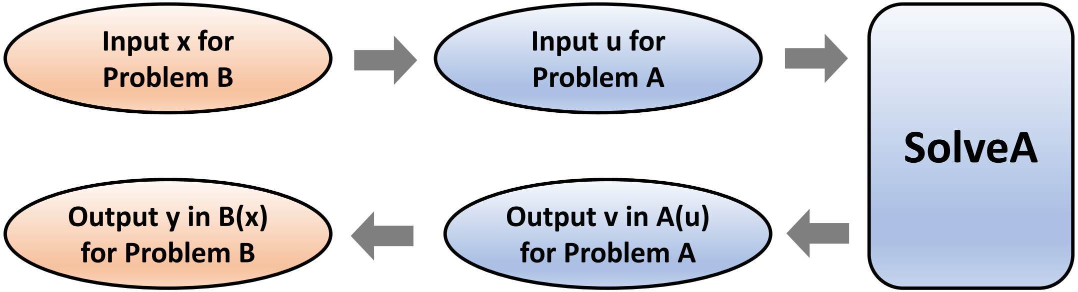
Homework 5 due Tuesday night 11:59 Boston time

No class Monday, per President Aoun's office

Class Tuesday and Wednesday next week, no class Thursday

Wednesday will include midterm review similar to last time

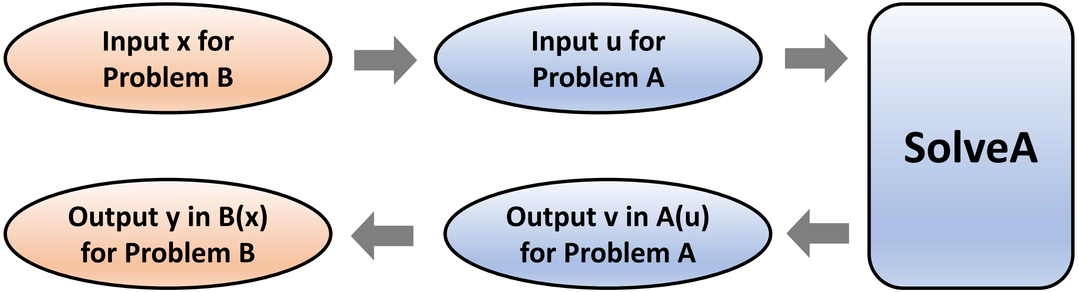
Last Time: Mechanics of Reductions



In the simplest case, we just call SolveA a single time. In fact we may use SolveA as a subroutine to a more complex reduction.

Last Time: Minimum Cut

A = MaxFlow
B = MinCut



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

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

Output $v \in A(u)$: $G = (V, E, s, t, \{c_e\})$

Output $y \in B(x)$: $G = (V, E, s, t, \{c_e\})$

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

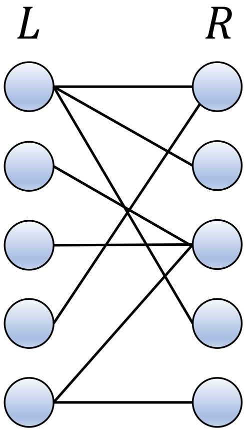
Bipartite Matching from Maximum Flow

Bipartite Matching

- **Input:** bipartite graph $G = (V, E)$ with $V = L \cup R$

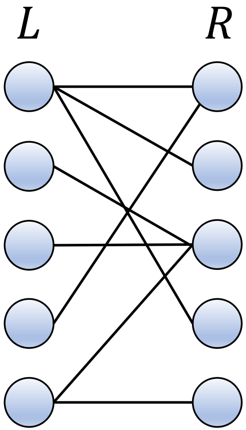
Models any problem where one type of object is assigned to another type:

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



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

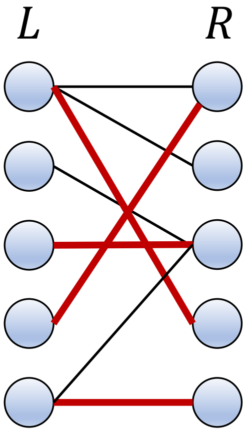


Models any problem where one type of object is assigned to another type:

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

Bipartite Matching

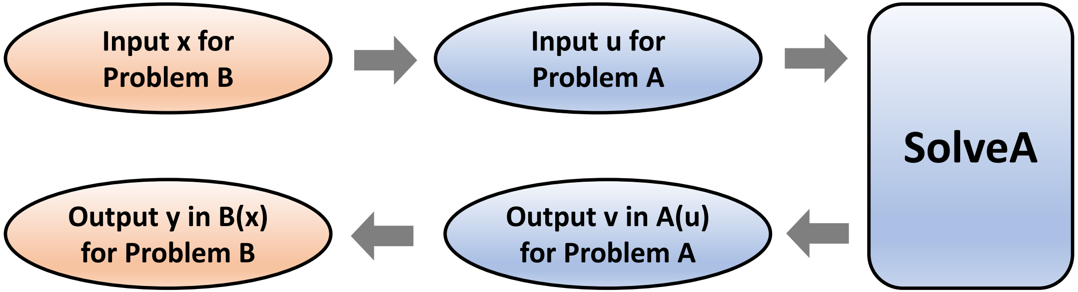
- **Input:** bipartite graph $G = (V, E)$ with $V = L \cup R$
- **Output:** a maximum cardinality matching
  - A **matching** $M \subseteq E$ is a set of edges such that every node v is an endpoint of at most one edge in M
  - Cardinality = $|M|$



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

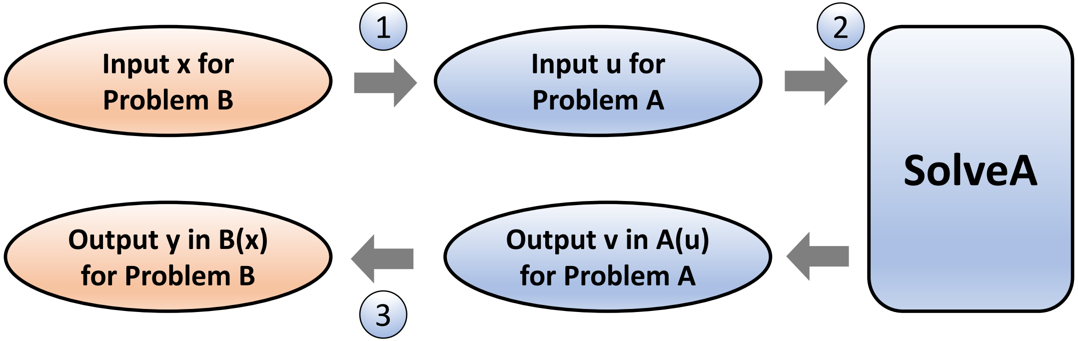
- doctors to hospitals
- jobs to processors
- advertisements to websites

Bipartite Matching



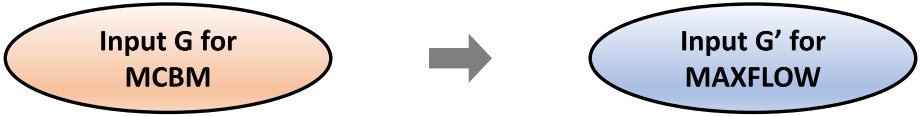
- There is a reduction that uses integer maximum s-t flow to solve maximum bipartite matching.
  - Problem B: maximum bipartite matching (MBM)
  - Problem A: integer maximum s-t flow

Bipartite Matching

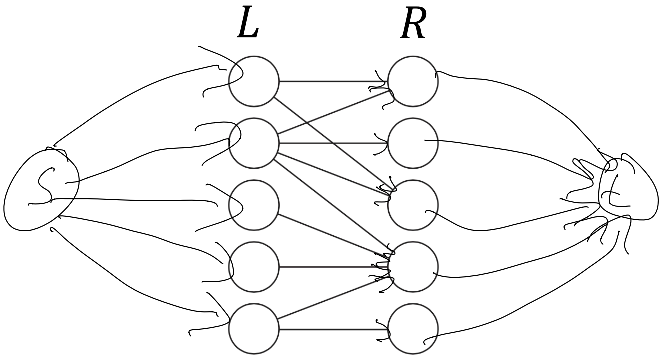


- There is a reduction that uses integer maximum s-t flow to solve maximum bipartite matching.
  - Problem B: maximum bipartite matching (MBM)
  - Problem A: integer maximum s-t flow

Step 1: Transform the Input

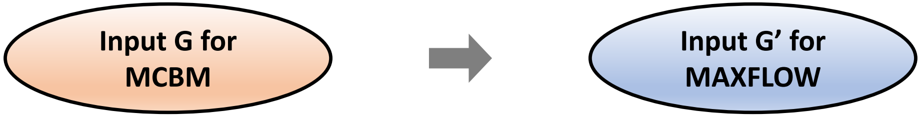


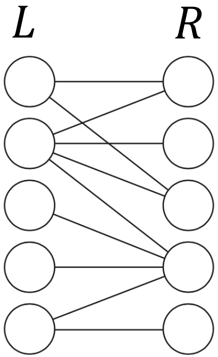
$G = (V, E, s, t, \{c_e\})$

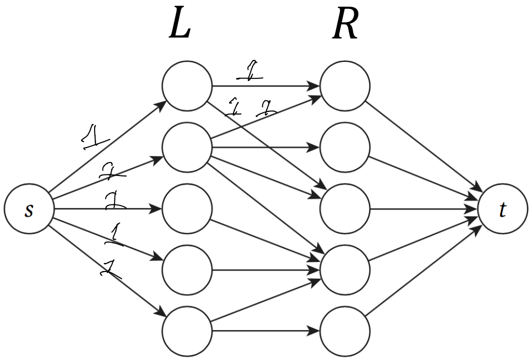


$\forall_{e \in E} \; c(e) = 1$

Step 1: Transform the Input



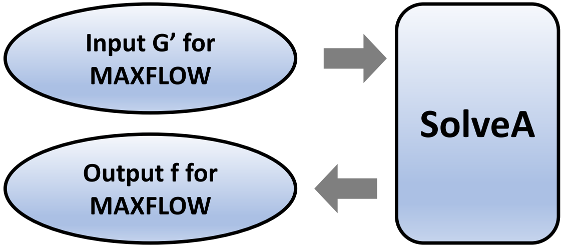


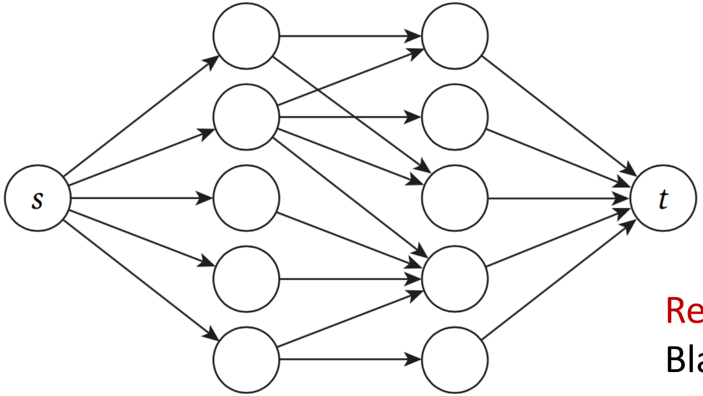


Set all edge capacities to $c(e) = 1$

Step 2: Receive the Output

Ford-Fulkerson

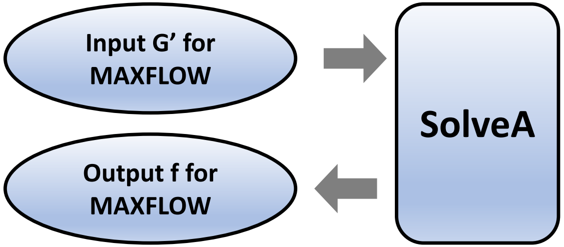


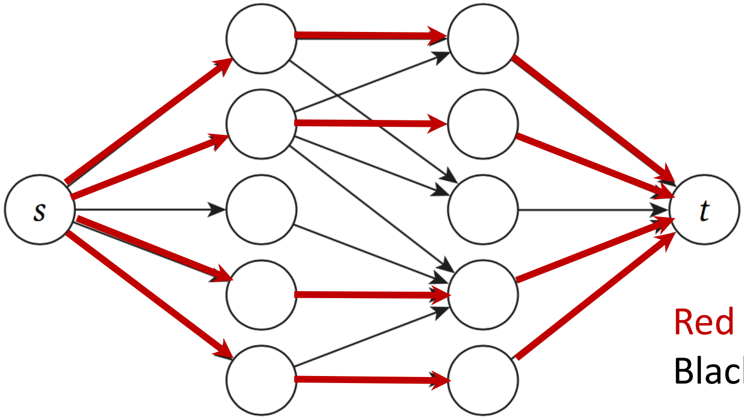


Red arrow means f(e)=1
Black means f(e) = 0

Step 2: Receive the Output

The matching will be all of the edges from L to R with nonzero flow!

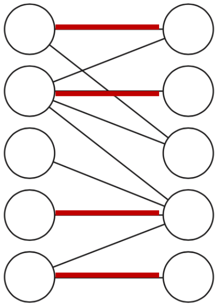


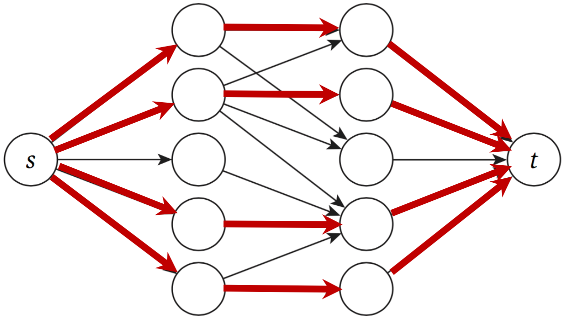


Red arrow means f(e)=1
Black means f(e) = 0

Step 3: Transform the Output

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





Reduction Recap

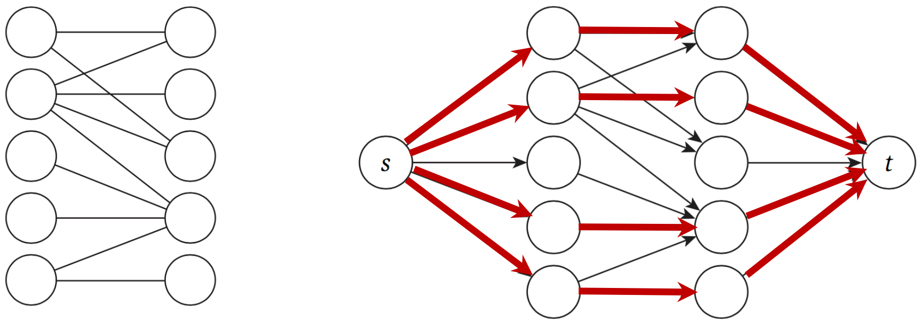
- Step 1: Transform the Input
  - Given G = (L,R,E), produce G' = (V,E,{c(e)},s,t) by...
    - ... orienting edges e from L to R
    - ... adding a node s with edges from s to every node in L
    - ... adding a node t with edges from every ~~not~~ node in R to t
    - ... seting all capacities to 1
-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

Correctness

- This algorithm returns a matching

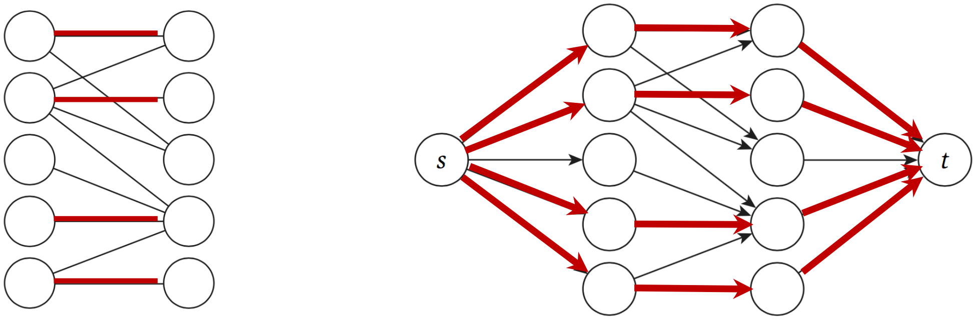


Since the capacity on every edge is 1, by conservation of flow we have:

- For any node in L, exactly one outgoing edge can have flow
- For any node in R, exactly one incoming edge can have flow

Correctness

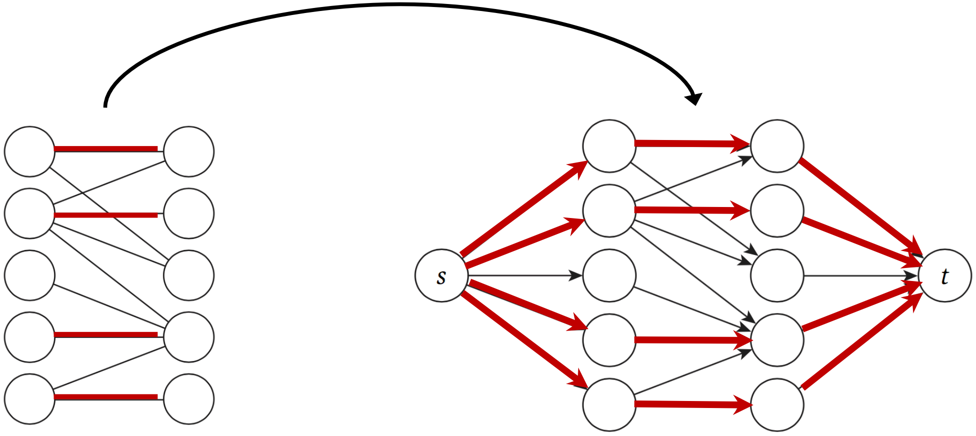
- **Claim:** G has a matching of cardinality at least k if and only if G' has an s-t flow of value at least k



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

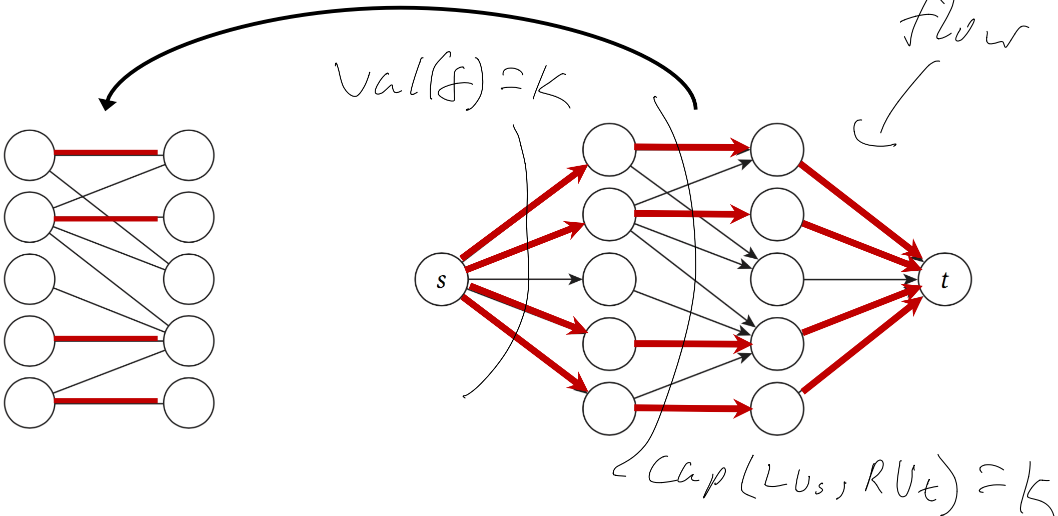
A matching of size k immediately implies a valid flow of value k



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

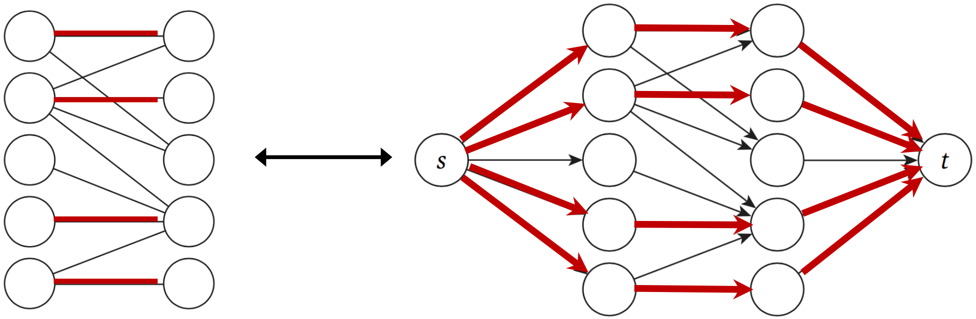
A flow of value k must have k edges carrying flow from L to R



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 immediately implies a valid flow of value k ⟷ A flow of value k must have k edges carrying flow from L to R



When k is the maximum cardinality matching, there must be a flow, and vice versa!

Running Time

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

Running Time

- Need to analyze the time for:
  - (1) Producing G' given G
    - G' has $n + 2$ nodes and $n + m$ edges, so we can construct it in $O(n + m)$ time
  - (2) Finding a maximum flow in G'
  - (3) Producing M given G'

Running Time

- Need to analyze the time for:
  - (1) Producing G’ given G
    - G’ has $n + 2$ nodes and $n + m$ edges, so we can construct it in $O(n + m)$ time
  - (2) Finding a maximum flow in G’
    - MaxFlow with all capacities 1 can be solved in $O(nm)$
  - (3) Producing M given G’

Running Time

- Need to analyze the time for:
  - (1) Producing G' given G
    - G' has $n + 2$ nodes and $n + m$ edges, so we can construct it in $O(n + m)$ time
  - (2) Finding a maximum flow in G'
    - MaxFlow with all capacities 1 can be solved in $O(nm)$
  - (3) Producing M given G'
    - We can scan the edges of G' to find the max flow in $O(n + m)$ time

Running Time

- Need to analyze the time for:
  - (1) Producing G’ given G
    - G’ has $n + 2$ nodes and $n + m$ edges, so we can construct it in $O(n + m)$ time
  - (2) Finding a maximum flow in G’
    - MaxFlow with all capacities 1 can be solved in $O(nm)$
  - (3) Producing M given G’
    - We can scan the edges of G’ to find the max flow in $O(n + m)$ time

(1) + (2) + (3)

- Adding the three together, we have

$$O(2 \cdot (n + m) + nm) \sim O(nm)$$

Summary

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

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

Wrap-up

No class Monday, per President Aoun’s office

Homework 5 due Tuesday night 11:59 Boston time

Class Tuesday and Wednesday next week, no class Thursday

Wednesday will include midterm review of some kind

Stay safe and enjoy your weekend