Flow Applications

Applications of Network Flow

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

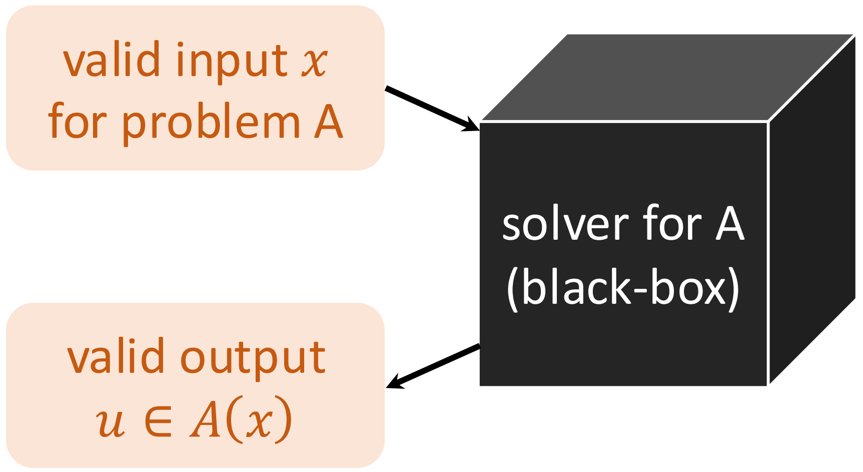
Mechanics of Reductions

- **Definition:** a computational **problem** is
  - a set of valid inputs $\boldsymbol{X}$ and
  - a set $\boldsymbol{A}(\boldsymbol{x})$ of valid outputs for each $\boldsymbol{x} \in \boldsymbol{X}$

- **Example:** integer maximum flow

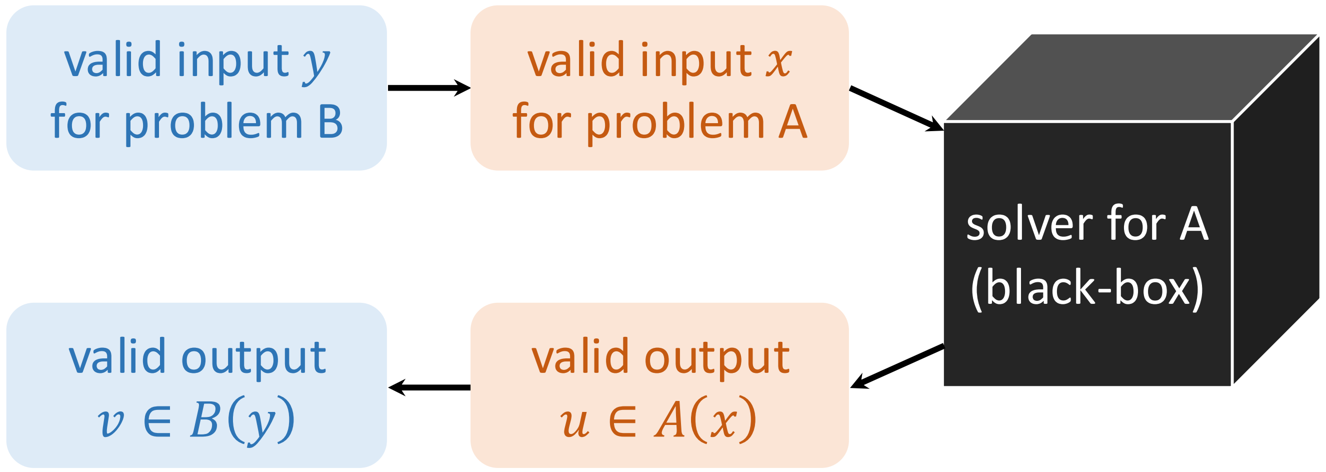
Mechanics of Reductions

- **Definition:** a **reduction** is an efficient algorithm that solves problem B using an algorithm that solves problem A as a **black-box**

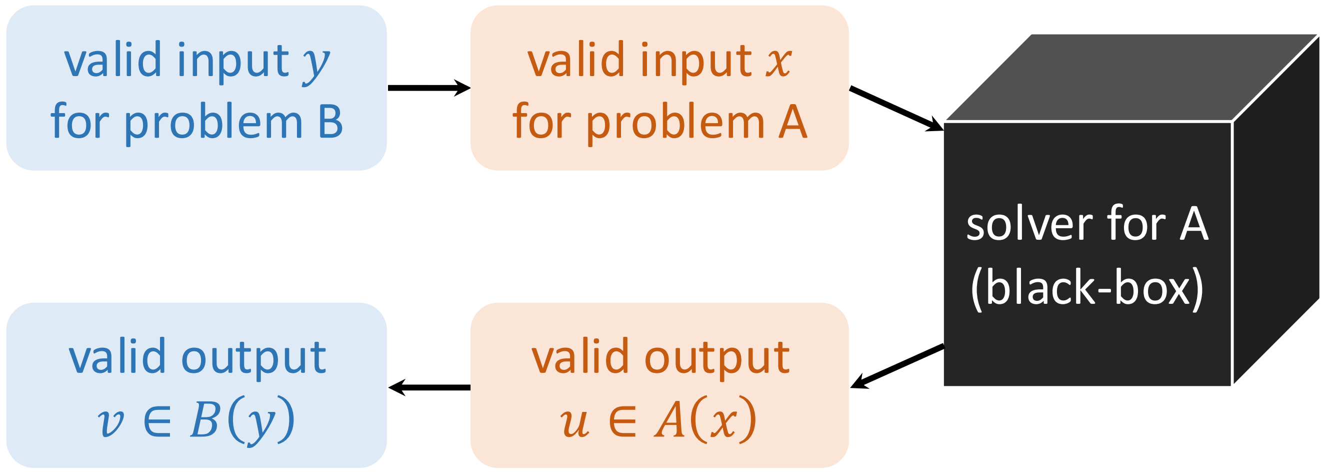


Mechanics of Reductions

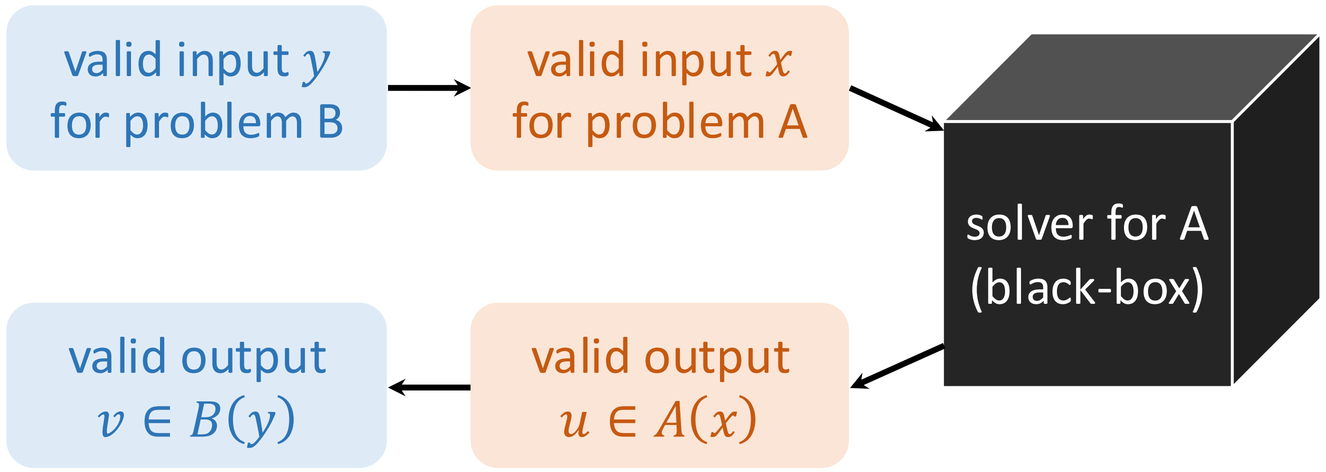
- **Definition:** a **reduction** is an efficient algorithm that solves problem B using an algorithm that solves problem A as a **black-box**


valid input y for problem B
valid input x for problem A
solver for A (black-box)
valid output $u \in A(x)$
valid output $v \in B(y)$

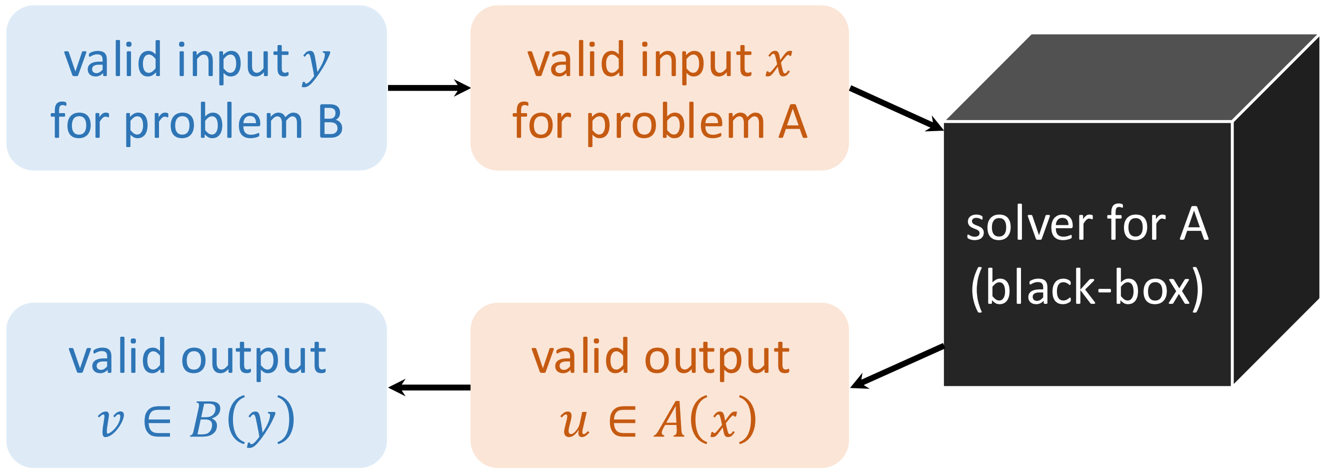
Correctness of Reductions


valid input y for problem B
valid input x for problem A
solver for A (black-box)
valid output $u \in A(x)$
valid output $v \in B(y)$

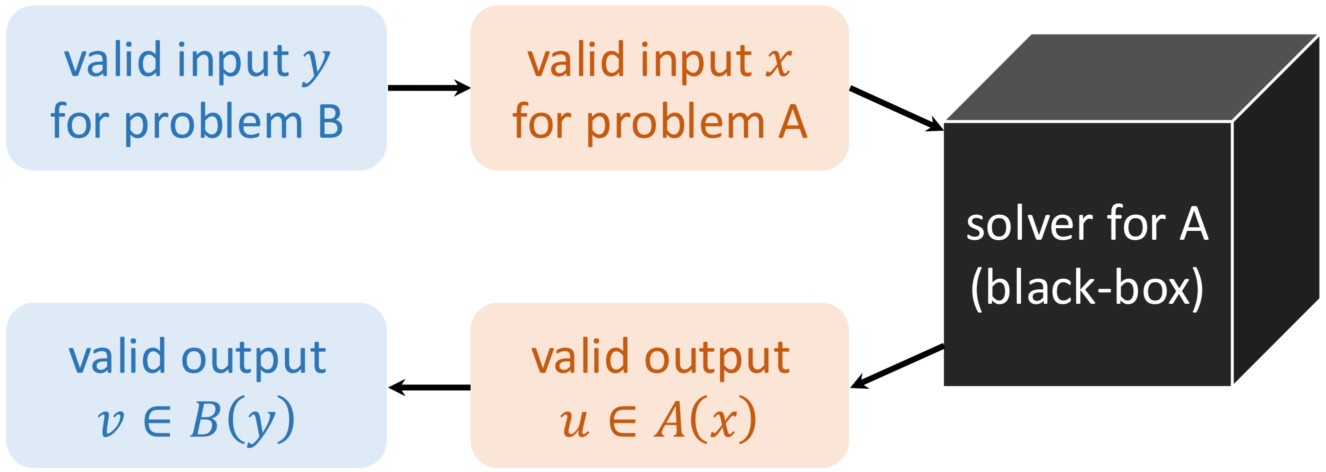
Running Time of Reductions


valid input y for problem B
valid input x for problem A
solver for A (black-box)
valid output $u \in A(x)$
valid output $v \in B(y)$

Example: Flows and Cuts


valid input y for problem B
valid input x for problem A
solver for A (black-box)
valid output $u \in A(x)$
valid output $v \in B(y)$

Example: Sorting and Median

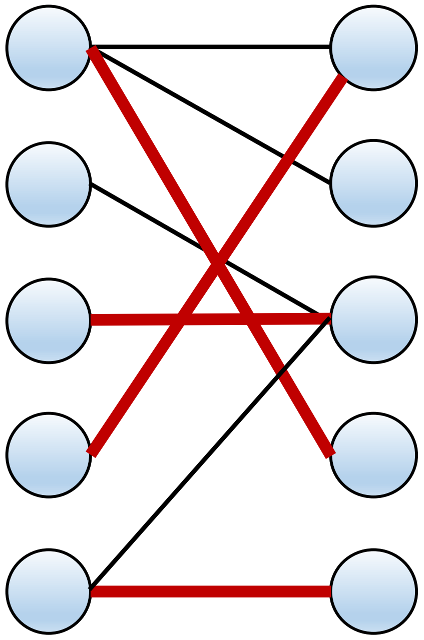

valid input y for problem B
valid input x for problem A
solver for A (black-box)
valid output $u \in A(x)$
valid output $v \in B(y)$

Maximum Bipartite Matching

- **Input:** bipartite graph $G = (V, E)$ with $V = L \cup R$
- **Output:** a matching of maximum size
  - 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
  - **Size** = $|M|$

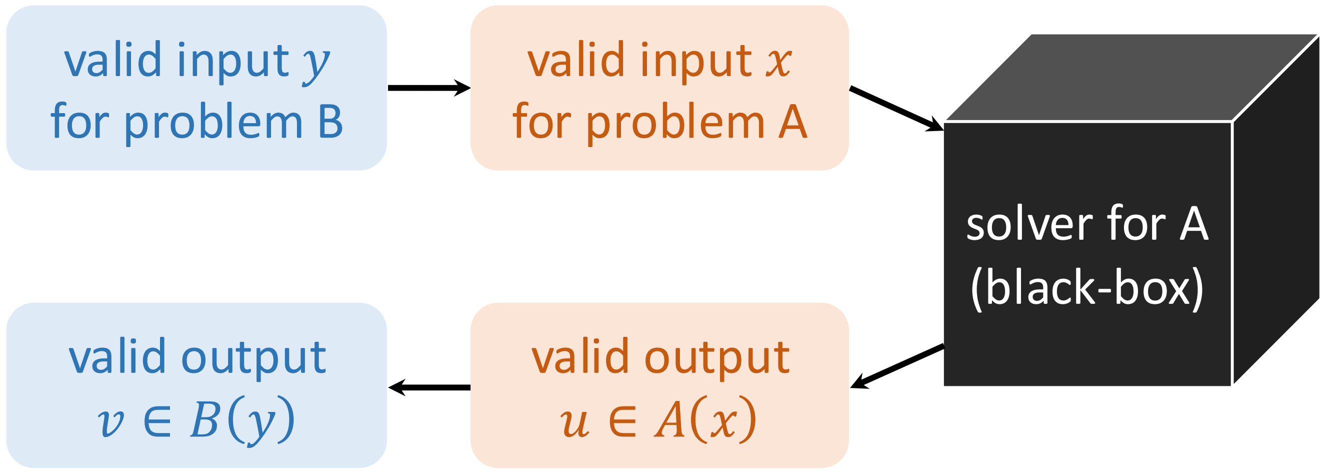
Models any problem where one type of object is assigned to another type:

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



Mechanics of Reductions

- **Theorem:** There is an efficient algorithm that solves maximum bipartite matching (MBM) using an algorithm that solves integer maximum s-t flow (MF)

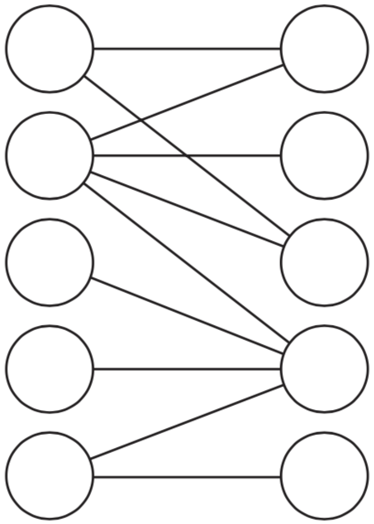

valid input y for problem B
valid input x for problem A
solver for A (black-box)
valid output $u \in A(x)$
valid output $v \in B(y)$

Step 1: Transform the Input

valid input G for MBM

$\rightarrow$

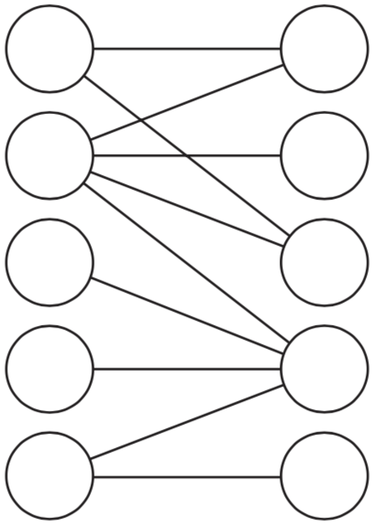
valid network G' for MF

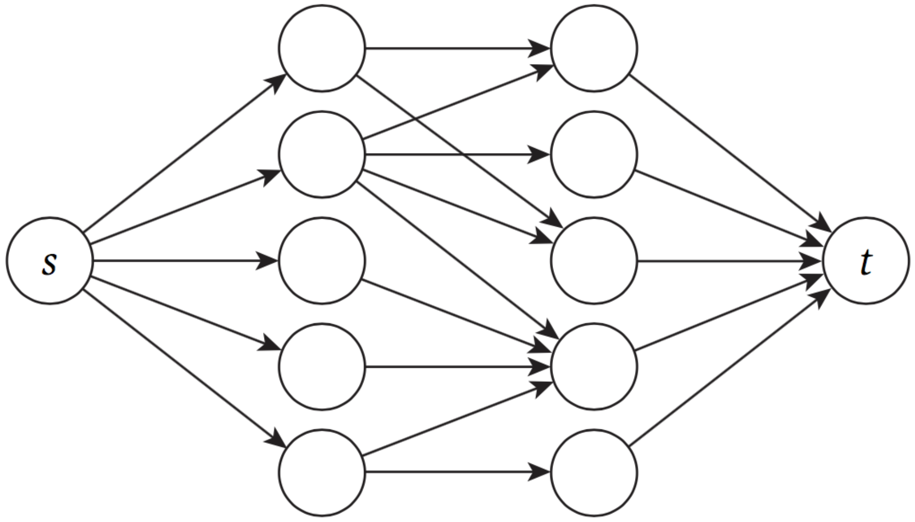


Step 1: Transform the Input

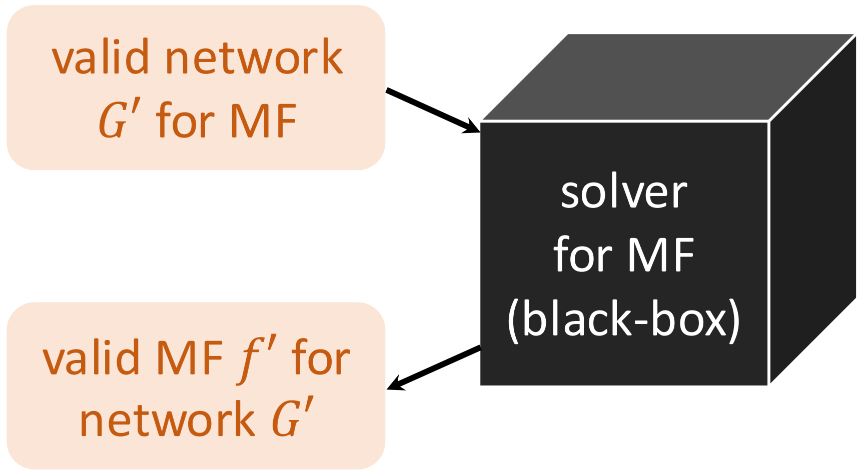
valid input G for MBM

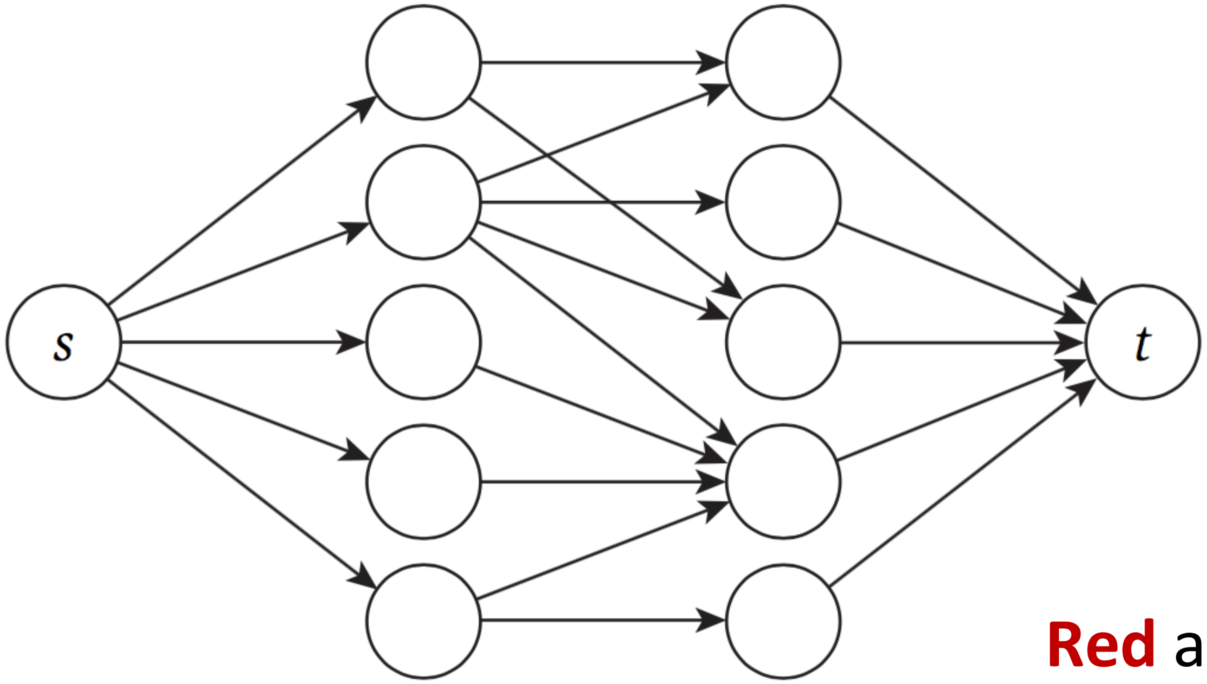
valid network G' for MF





Step 2: Receive the Output

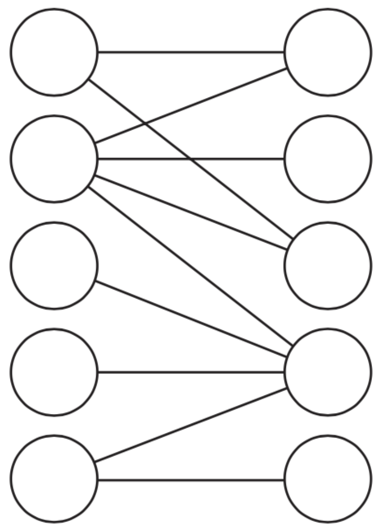

valid network G' for MF
solver for MF (black-box)
valid MF f' for network G'

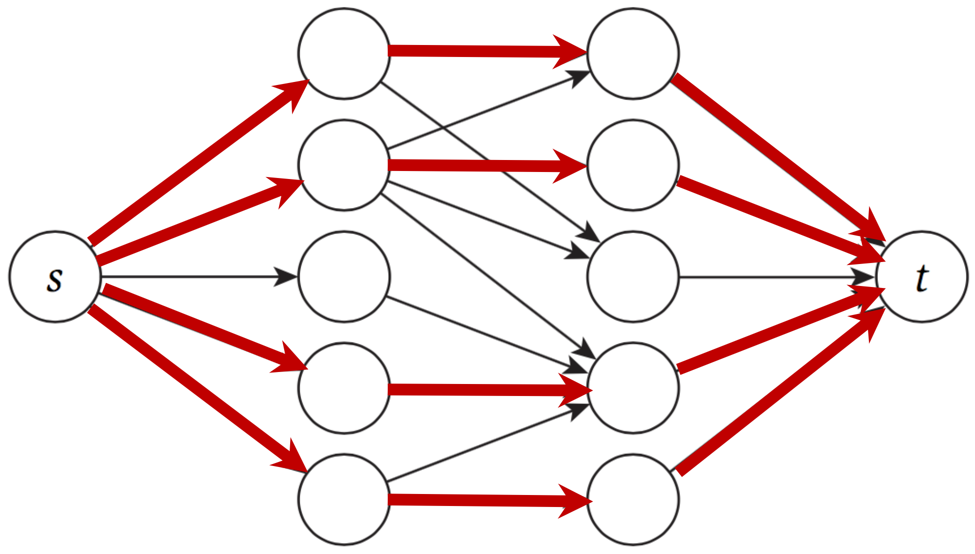


Red arrow means $f'(e) = 1$
Black arrow means $f'(e) = 0$

Step 3: Transform the Output

valid MBM M for graph G ⬅ valid MF f' for network G'





Reduction Recap

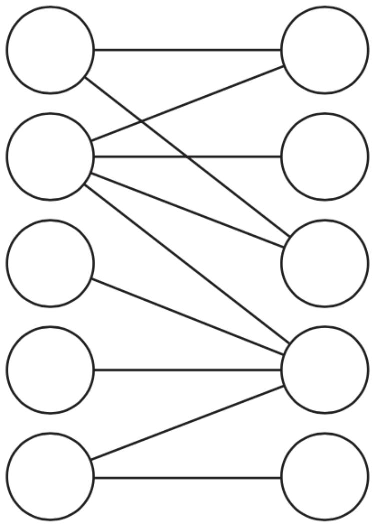
- Step 1: Transform the Input
  - Given bipartite graph $G = (L, R, E)$, produce flow network $G' = (V, E, \{c(e)\}, s, t)$ by:
    - orienting edges e from L to R
    - adding a node s with edges from s to every node in L
    - adding a node t with edges from every node in R to t
    - setting all capacities 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 $f'(e) = 1$

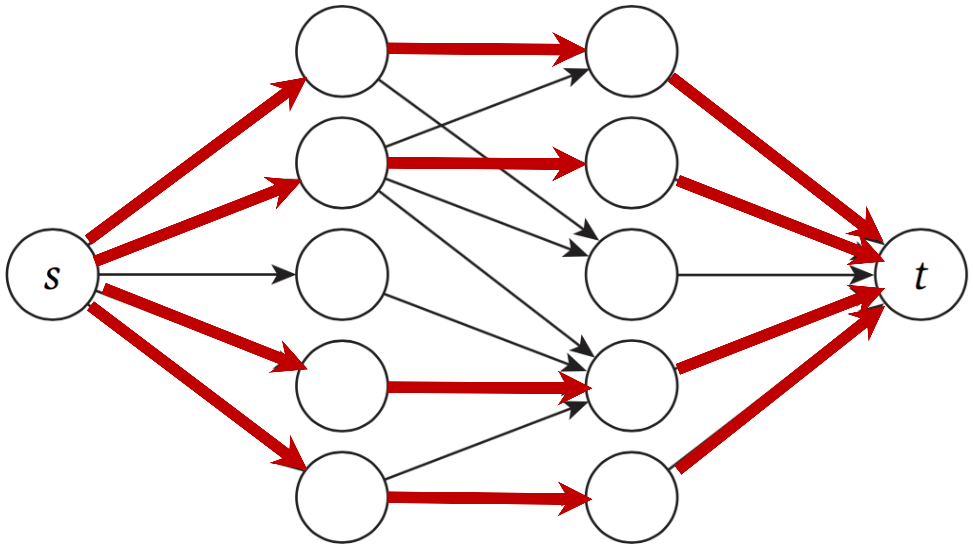
Correctness

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

Correctness

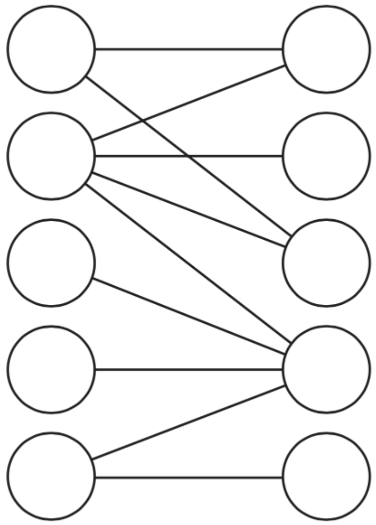
- This algorithm returns a matching

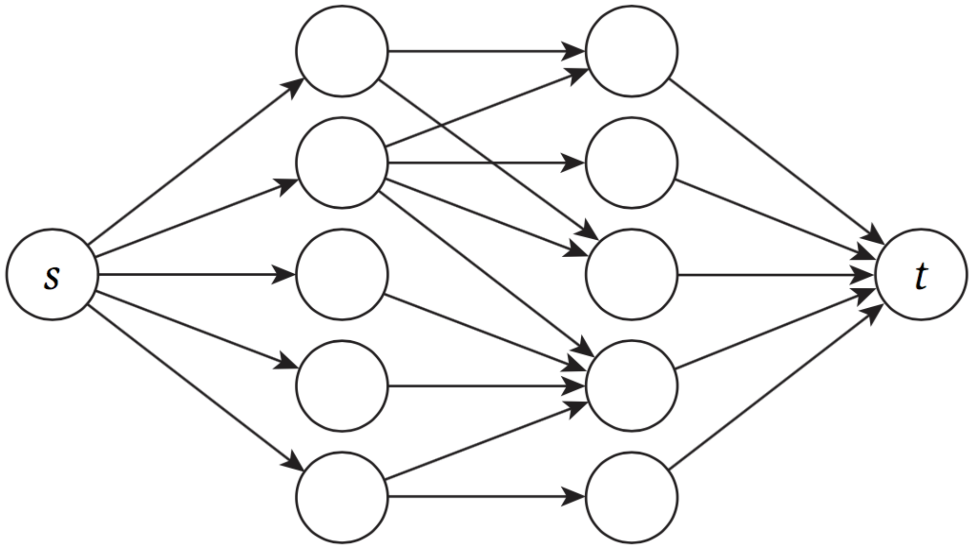




Correctness

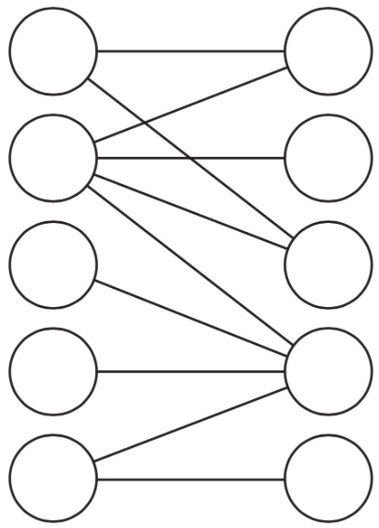
- **Claim:** G has a matching of cardinality k if and only if G' has an s-t flow of value k

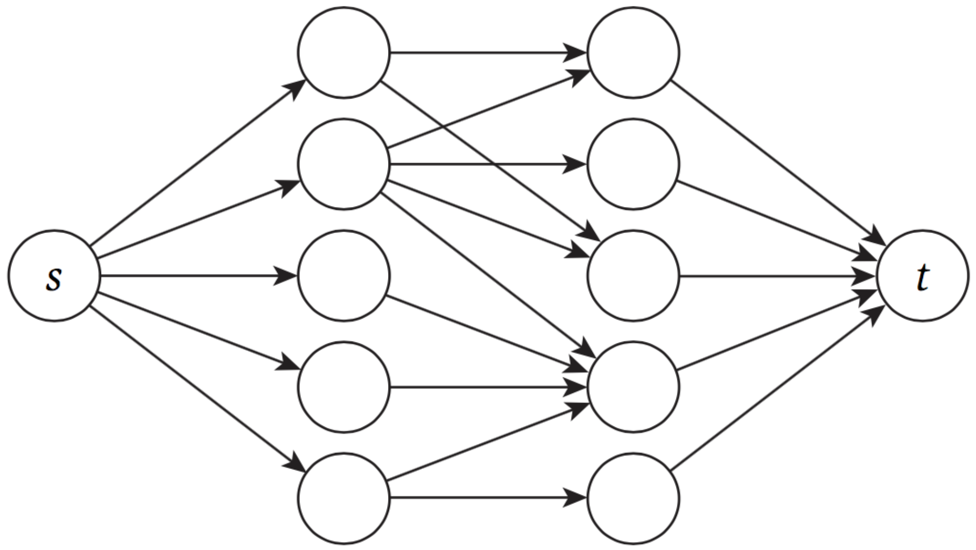




Correctness

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

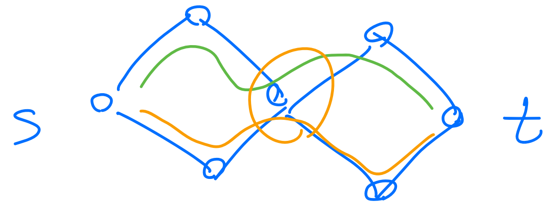




Edge-Disjoint Paths

- **Input:** directed graph $G = (V, E)$ and vertices s, t.
- **Output:** maximum number of edge disjoint paths between s and t.
  - Each edge must appear in at most one path.
  - A vertex may belong to multiple paths.

Set all capacities to ~~be~~ 1. Solve max flow, return the answer as the ans to EDP problem.



Thm: with this reduction, the sol to EDP is K iff the sol to the flow instance is K.

Proof: If EDP is K, pass a flow of one through each path, so flow sol $\geq K$.

For the other direction we apply the flow decomposition thm (the integral version)

This gives K flow paths with integer flows, which must have flow exactly one. Additionally these paths must be edge disjoint because each edge has capacity 1.

Vertex-Disjoint Paths

- **Input:** directed graph $G = (V, E)$ and vertices s, t.
- **Output:** maximum number of **vertex** disjoint paths between s and t.
    - Each edge must appear in at most one path.
    - A vertex $\wedge$ may belong to ~~multiple paths.~~ at most one path.

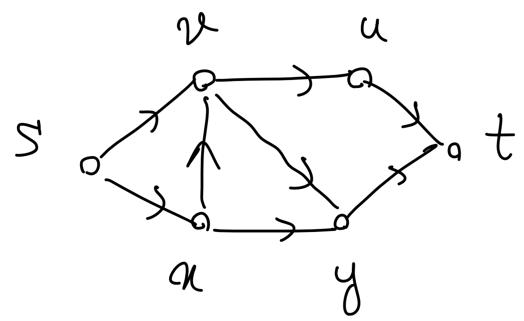
other than s,t.

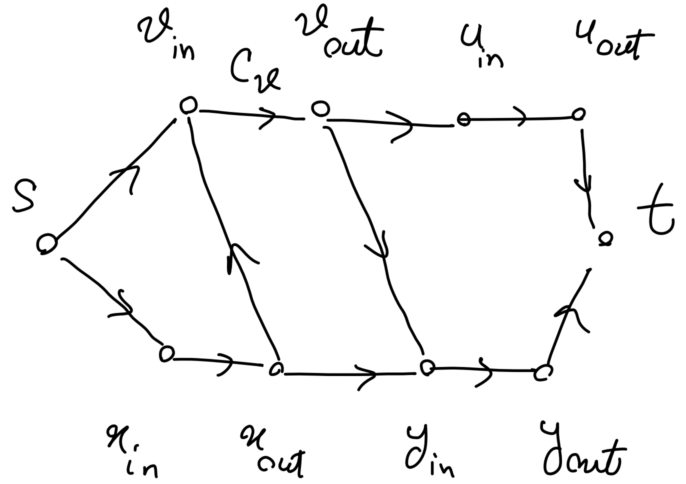
Vertex Capacitated Flows: In addition to edge capacities, suppose we are also given a capacity C_v for every vertex v, restricting the amount of flow into v.

Lemma: The vertex cap $\overset{\text{max}}{\text{flow}}$ can be solved in $O(m+T)$ time where T is the time need to solve max flow on a graph with $2n$

vertices and $O(m+n)$ edges.

Pf:





For every edge (u,v), add edge (u_{out}, v_{in}).

It can be shown that the resulting graph has max flow k iff the max vertex-cap flow in G has val k. $\square$

We use the same reduction as before, except we put a capacity of one on every vertex.

Baseball Elimination

- Every year millions of American baseball fans eagerly watch their favorite team, hoping they will win a spot in the playoffs, and ultimately World Series.
- Sadly, many teams are “mathematically eliminated” days or weeks before the regular season ends. E.g. if a team cannot win enough games to catch up to the current leader, they are eliminated.
- But the situation is not always this easy. Consider the following standing from American League East on Aug 30, 1996.
- While Detroit is clearly behind, if they win all their 27 remaining games they will end up with 76 wins, more than any team has now.
- But does this mean Detroit can end up being the leader?

Team	**Won–Lost**	**Left**	NYY	BAL	BOS	TOR	DET
New York Yankees	76 ~~75~~–59	28		3	~~8~~ 1 7	7	3
Baltimore Orioles	76 71–63	28	~~3~~ 3 0		~~2~~ 2 0	7	4
Boston Red Sox	76 ~~69~~–66	27	~~8~~ 7 1	2		0	0
Toronto Blue Jays	77 ~~63~~–72	27	7	7	0		0
Detroit Tigers	~~49~~–86 76	27	3	4	0	0	

Team	**Won–Lost**	**Left**	NYY	BAL	BOS	TOR	DET
New York Yankees	75–59	28		3	8	7	3
Baltimore Orioles	71–63	28	3		2	7	4
Boston Red Sox	69–66	27	8	2		0	0
Toronto Blue Jays	63–72	27	7	7	0		0
Detroit Tigers	49–86	27	3	4	0	0	

By winning all of their remaining games, Detroit can finish the season with a record of 76 and 86. If the Yankees win just 2 more games, then they will finish the season with a 77 and 85 record which would put them ahead of Detroit. So, let's suppose the Tigers go undefeated for the rest of the season and the Yankees fail to win another game.

The problem with this scenario is that New York still has 8 games left with Boston. If the Red Sox win all of these games, they will end the season with at least 77 wins putting them ahead of the Tigers. Thus, the only way for Detroit to even have a chance of finishing in first place, is for New York to win exactly one of the 8 games with Boston and lose all their other games. Meanwhile, the Sox must lose all the games they play against teams other than New York. This puts them in a 3-way tie for first place. . . .

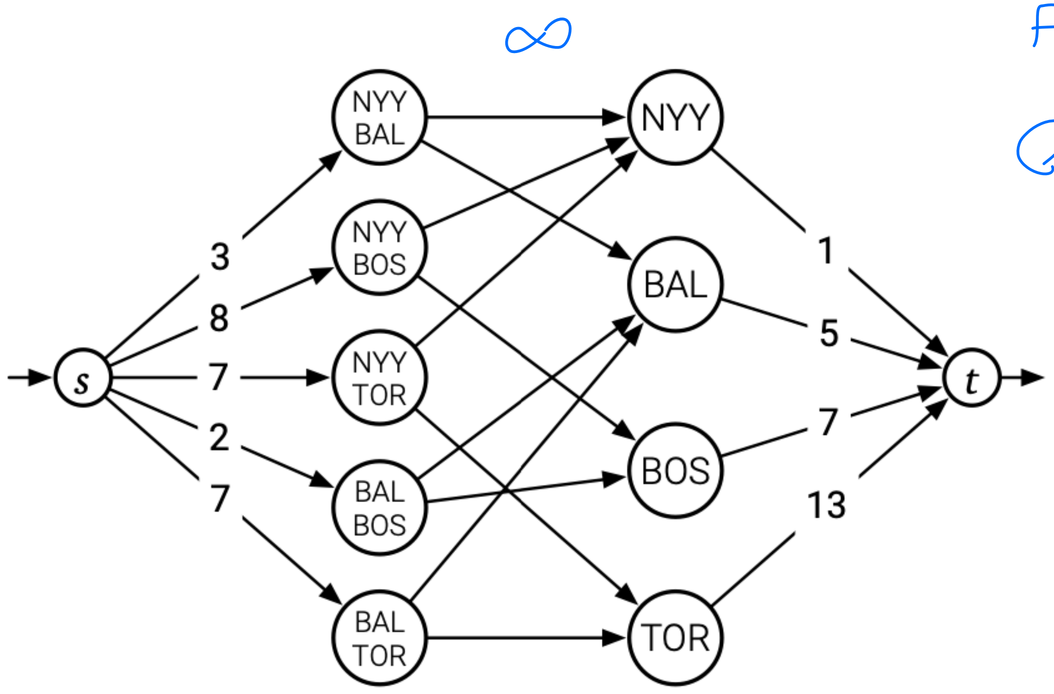
Now let's look at what happens to the Orioles and Blue Jays in our scenario. Baltimore has 2 games left with with Boston and 3 with New York. So, if everything happens as described above, the Orioles will finish with at least 76 wins. So, Detroit can catch Baltimore only if the Orioles lose all their games to teams other than New York and Boston. In particular, this means that Baltimore must lose all 7 of its remaining games with Toronto. The Blue Jays also have 7 games left with the Yankees and we have already seen that for Detroit to finish in first place, Toronto must will all of these games. But if that happens, the Blue Jays will win at least 14 more games giving them at final record of 77 and 85 or better which means they will finish ahead of the Tigers. So, no matter what happens from this point in the season on, Detroit can not finish in first place in the American League East.

Baseball Elimination

- The baseball elimination problem can be abstracted as follows.
- **Input:**
    - $W[1..n]$: Number of current wins by team i
    - $G[1..n, 1..n]$: Number of game left between teams i and j.
- **Goal:** Can team n end up with maximum number of wins possibly tied with other teams?

Baseball Elimination

- Let $R[i] = \sum_j G[i,j]$ be the remaining games for team i.
- Let's assume team n wins all of its $R[n]$ remaining games.
- **Observation:** Team n ends up in first place iff every team i wins at most $W[n] + R[n] - W[i]$ of its remaining games.

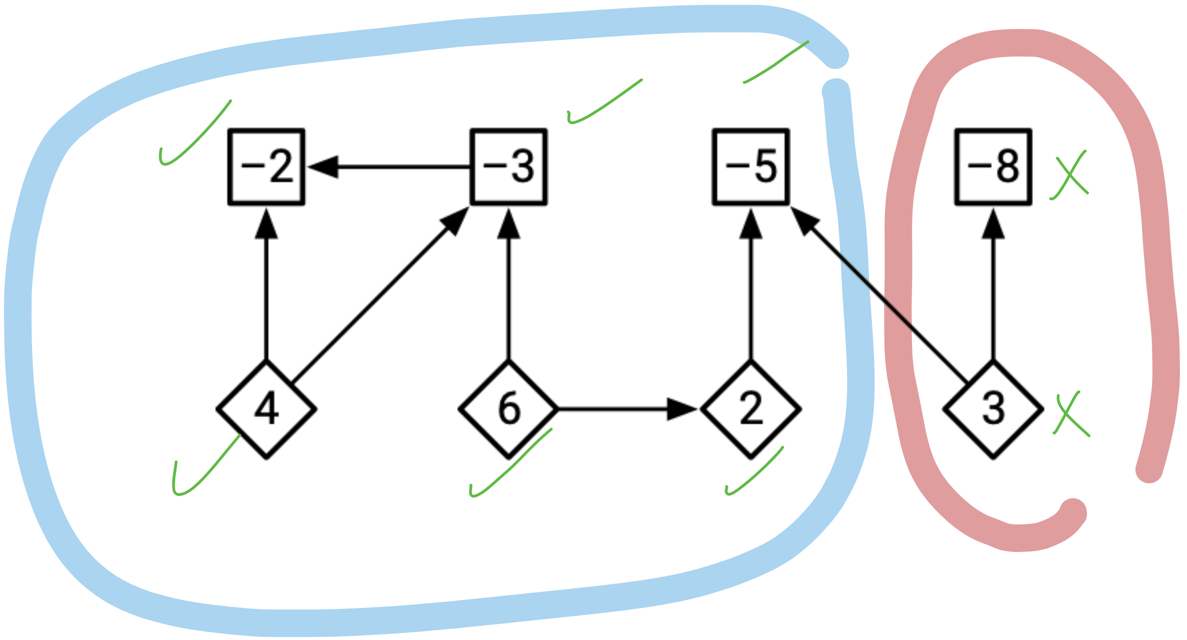


FF solves in

$O(m\, v^*) = O(n^4)$.

Project Selection

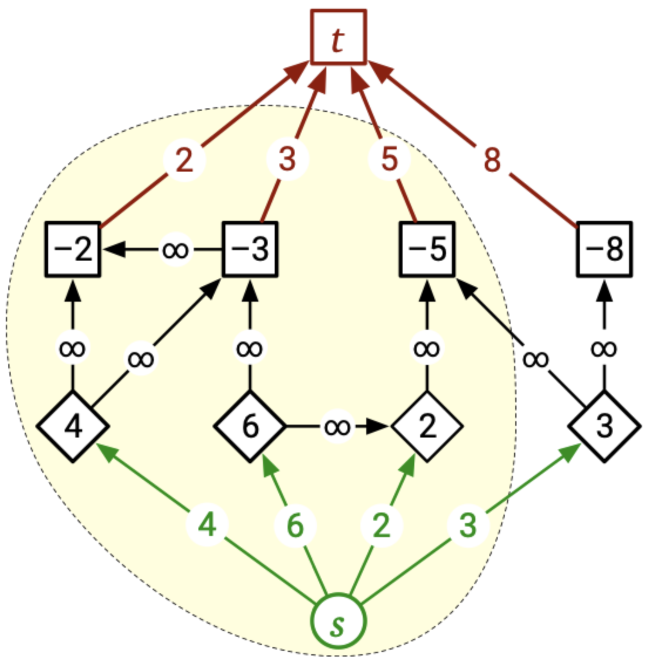
- A set of n projects. Some projects cannot be started until certain other projects are completed. The projects and their dependencies are described by a directed acyclic graph (DAG).
- Each project v has a profit $\$(v)$ which can be positive or negative (in that case doing the project has a *cost*).
- Goal: Finish a subset of valid projects to maximize profit.



P = 15

W = 13

$\textcircled{2}$



Costly jobs

Profitable jobs

- **Key Claim:** Take any S-T cut of finite weight W. Then selecting the jobs in S guarantees a profit of P-W where P is total profit of all profitable jobs.

It therefore suffices to minimize W to maximize profit.