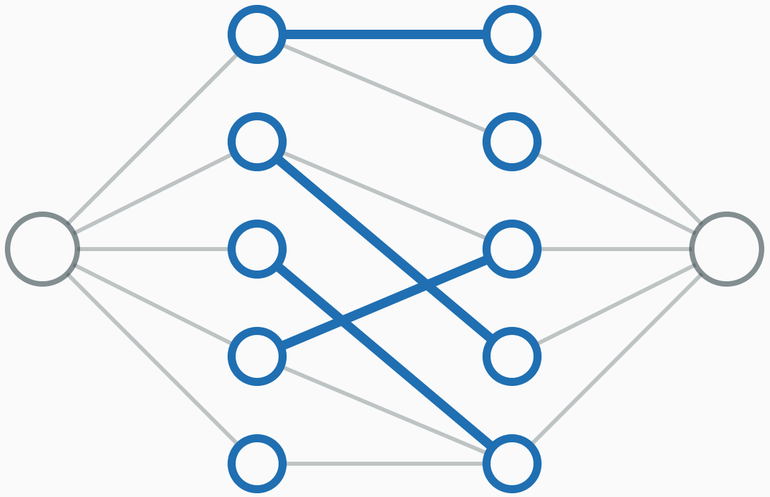
ALGORITHMS AND DATA STRUCTURES · LECTURE 12

Modelling with flows

Matching, baseball elimination, open-pit mining



Heng Guo

School of Informatics, University of Edinburgh

Thursday 29 October 2026 · 16:10–17:00

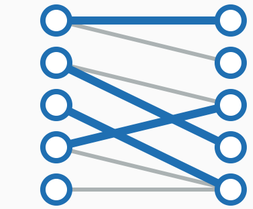
Today: what can we do with a maximum flow?

Lectures 9 to 11 gave us

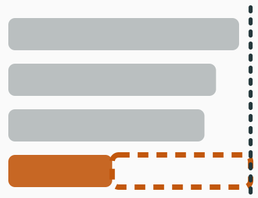
- maximum flow in polynomial time
- max flow = min cut
- a minimum cut from the last G_f
- integer capacities: an integral flow

? *Where do flow networks come from?*

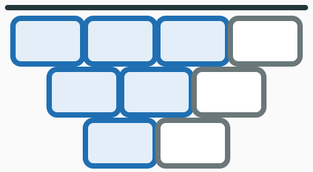
Most problems are **not** given as a network.
We **build** one and run the algorithm.



Bipartite matching
applicants to jobs



Baseball elimination
can our team still win?



Open-pit mining
which blocks to dig?

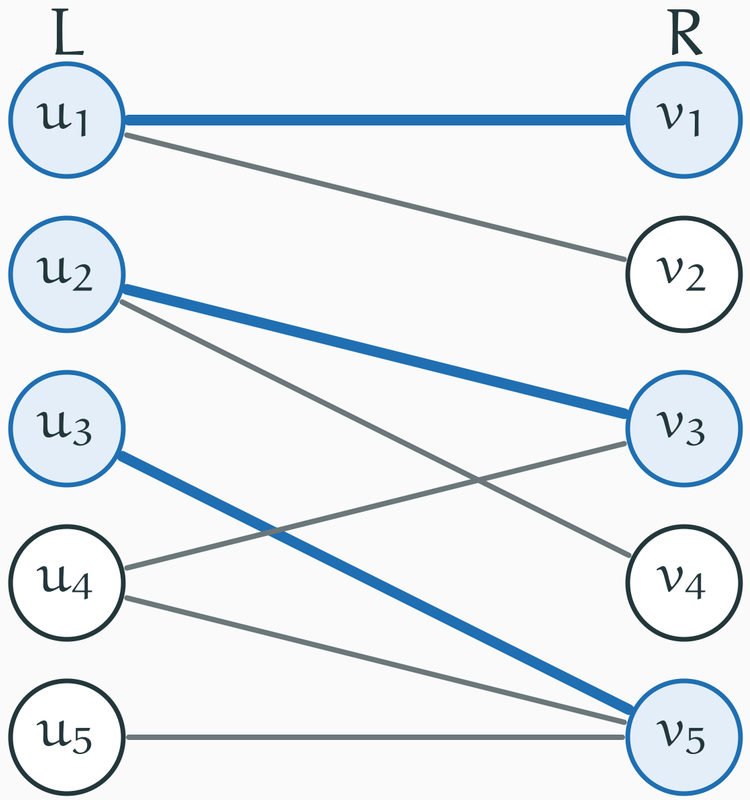
Bipartite matching

Matchings in bipartite graphs

Applicants L, jobs R.
An edge $\{u, v\}$ means: u can do v.

Bipartite graph.
$G = (L \cup R, E)$: every edge joins a node of L to a node of R.

Matching.
A set $M \subseteq E$ in which every node occurs **at most once**.



a matching M_0, in blue

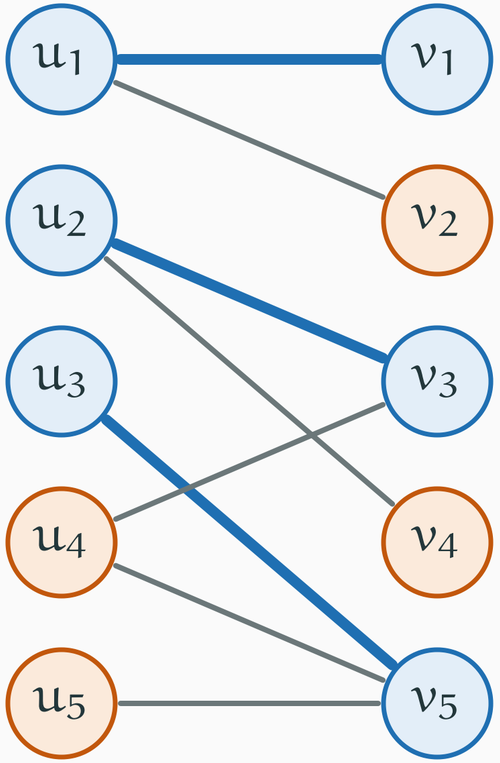
Match as many as possible?

Greedy: each of $u_1, \ldots, u_5$ in turn takes its first free neighbour.

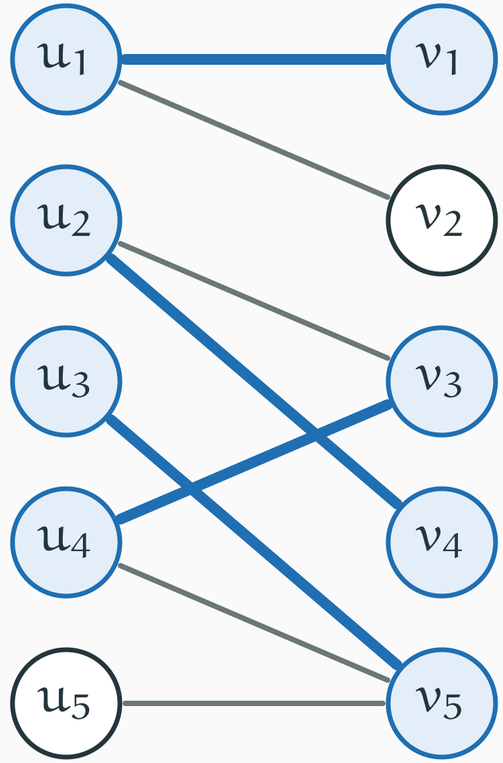
Maximal does not imply maximum: $|M_0| = 3 < 4 = |M^*|$.

> **Problem.**
> Given a bipartite graph,
> find a maximum matching.

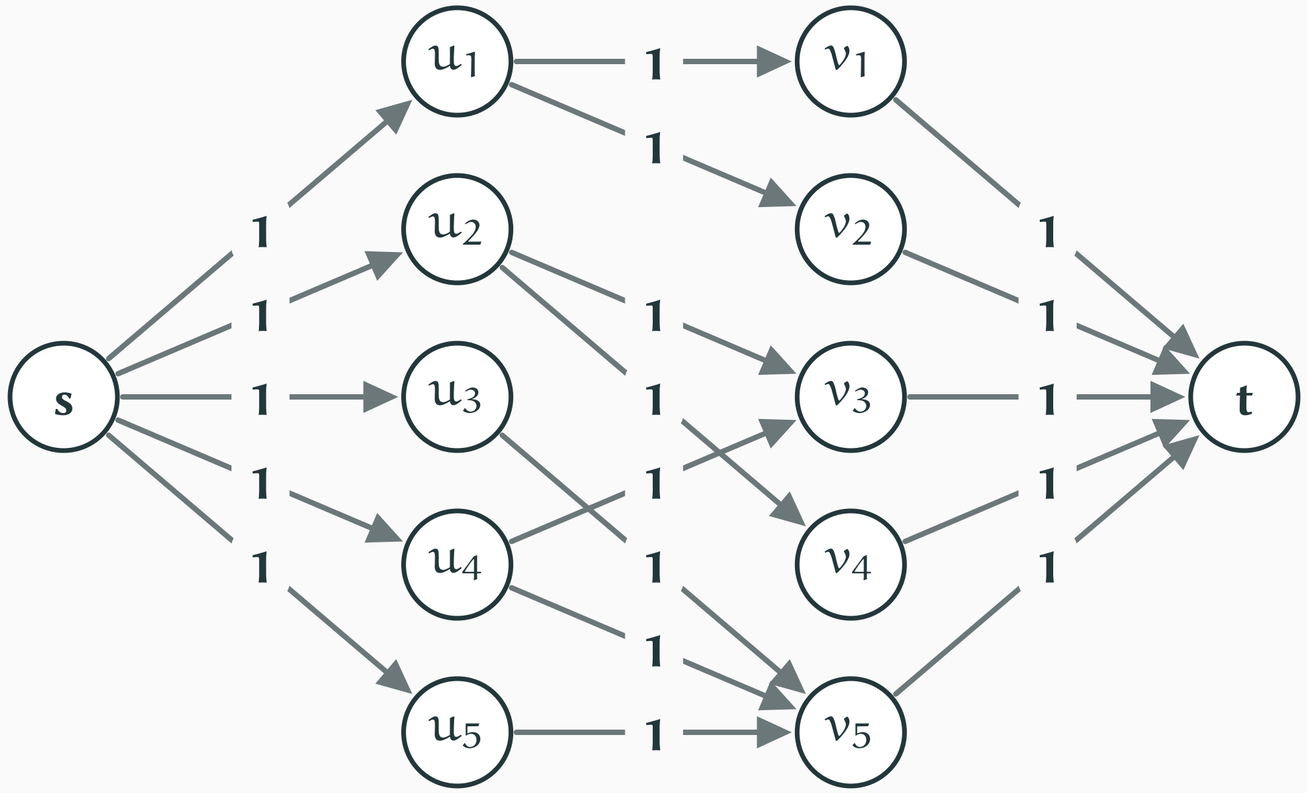
M_0 is **maximal**:
no edge can be added.



M^* is **maximum**:
no matching is larger.



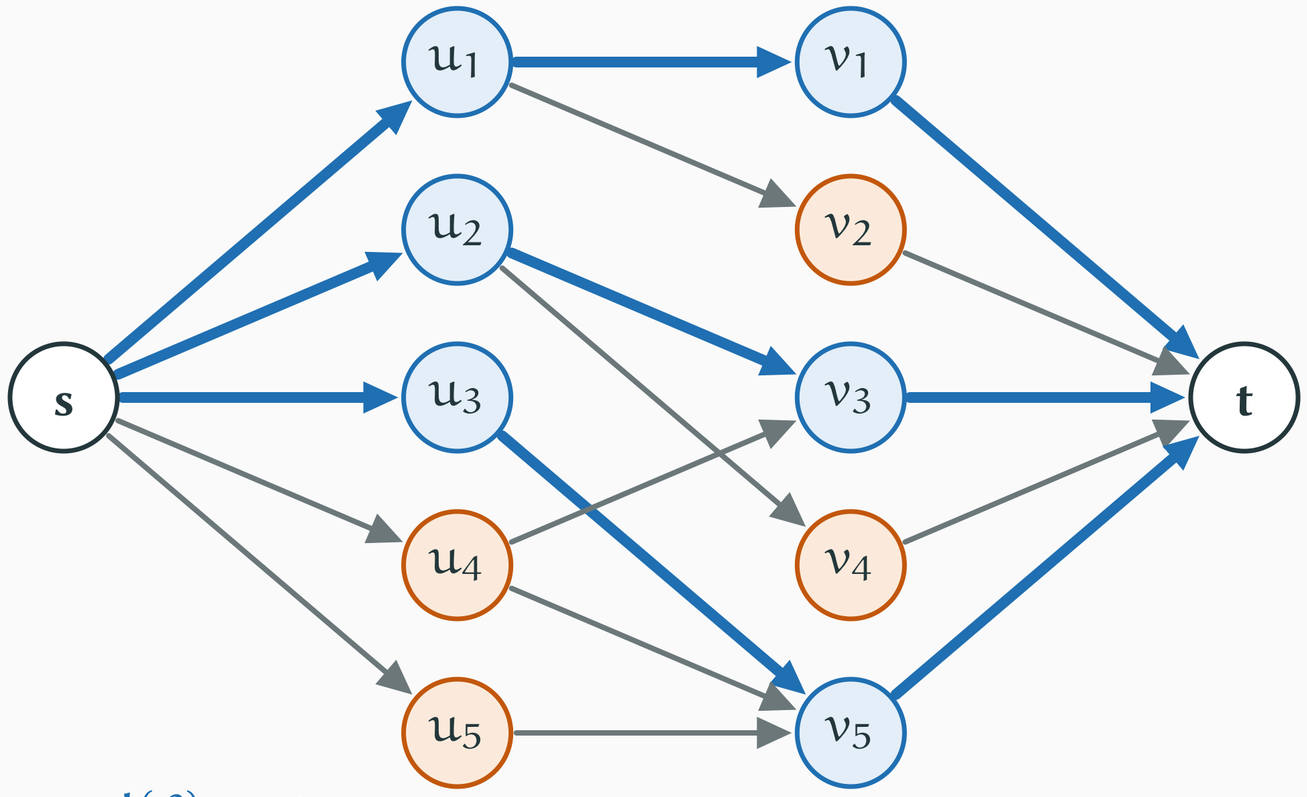
From a matching problem to a flow network



1. Add a source s and a sink t.
2. Place G between them.
3. Orient every edge from L to R.
4. Add edges (s, u) for all $u \in L$.
5. Add edges (v, t) for all $v \in R$.
6. Give every edge capacity 1.

The network G' has $n + 2$ nodes
and $m + n$ edges,
where $n = |L| + |R|$ and $m = |E|$.

From matchings to flows



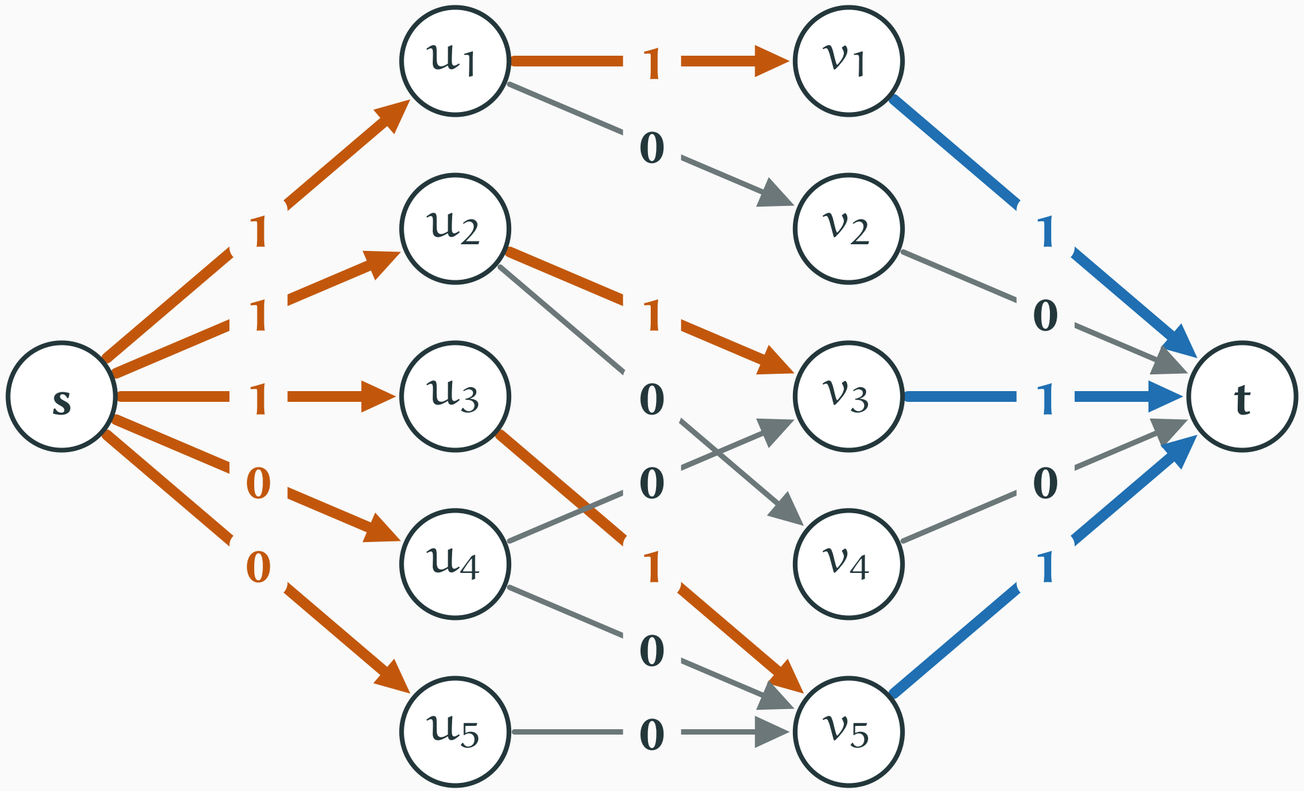
$\mathrm{val}(f) = 3$

Claim 1.
A matching M with k edges gives a flow of value k.

For every $\{u, v\} \in M$ send one unit along $s \to u \to v \to t$:
$f = 1$ on these edges, $f = 0$ elsewhere.

- **Capacity:** every $f(e)$ is 0 or 1.
- **Conservation:** a matched node has one edge of M: one unit in, one out. An unmatched node: none, none.
- **Value:** $\mathrm{val}(f) = |M|$.

From flows to matchings

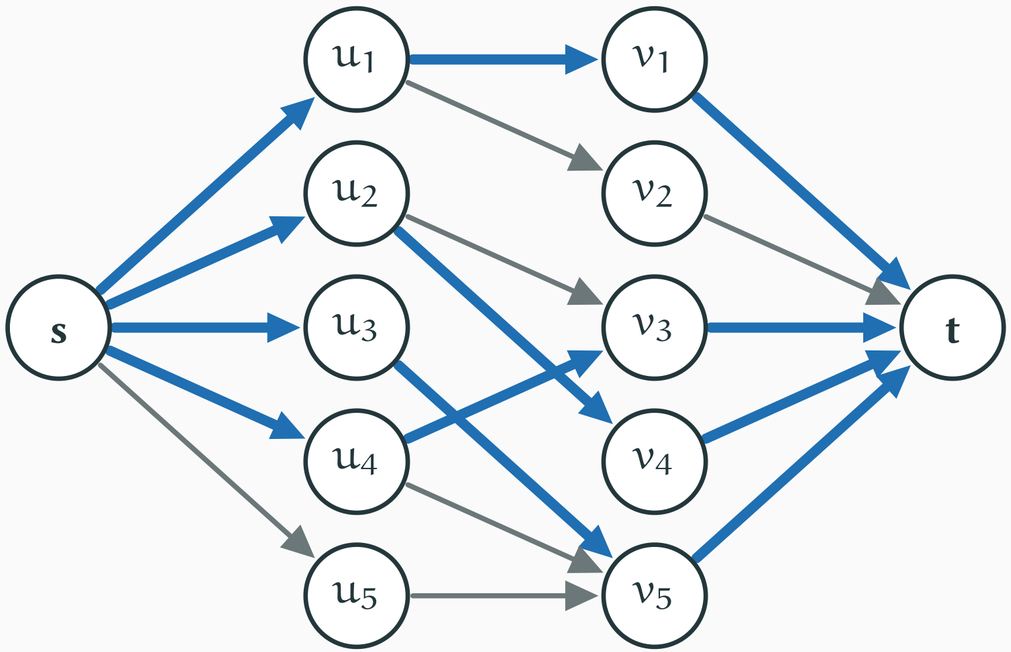


Claim 2.
An integral flow f of value k gives a matching with k edges.

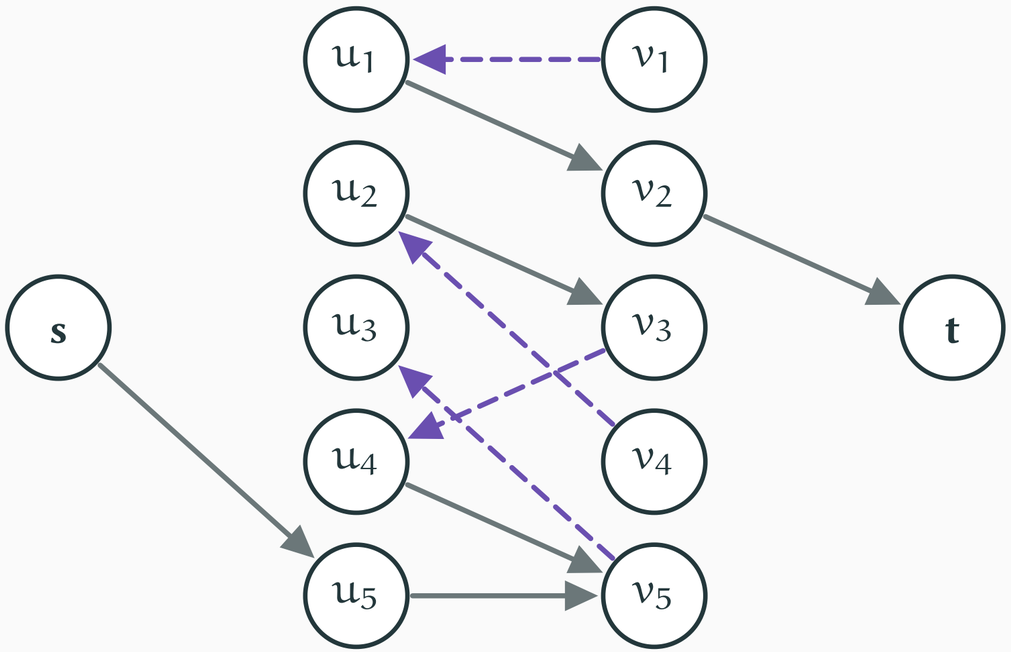
1. Every $f(e) \in \{0, 1\}$: capacities are 1 and f is integral.
2. $M_f = \{\{u, v\} : f(u, v) = 1\}$.
3. M_f **is a matching.**
   At u: in $\leqslant 1$, so out $\leqslant 1$.
   At v: out $\leqslant 1$, so in $\leqslant 1$.
4. **Size:** $k = \sum_u f(s, u)$
   $= \sum_{u,v} f(u, v) = |M_f|$.

Worked example: improving a matching

G' with the new flow f'



the new residual graph $G_{f'}$



$\mathrm{val}(f') = 4$, and t cannot be reached from s: no augmenting path.

The matching is maximum, but not perfect.

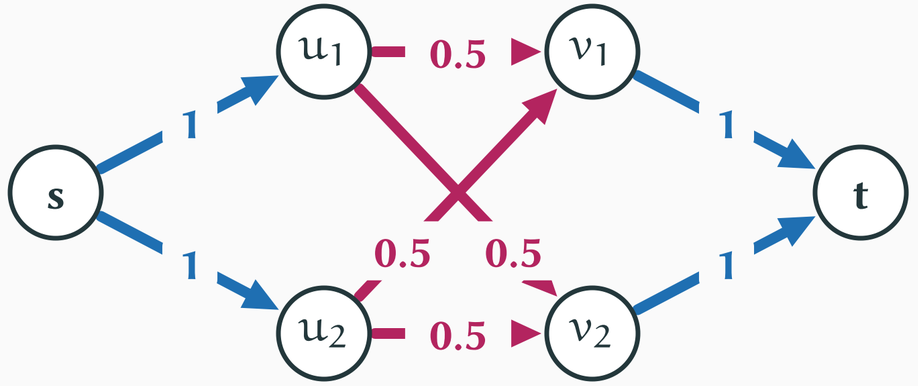
Maximum matching = maximum flow

Theorem.
The largest size of a matching of G
is the maximum value of a flow in G′.

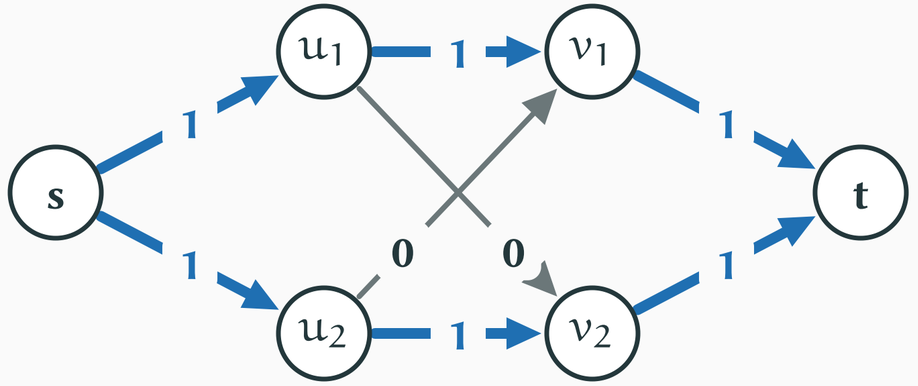
- ⩽ Claim 1: a matching gives a flow of the same value.
- ⩾ Claim 2: an **integral** maximum flow gives a matching of the same size.

? *Why must the flow be integral?*

Ford–Fulkerson returns an integral flow (Lecture 11).

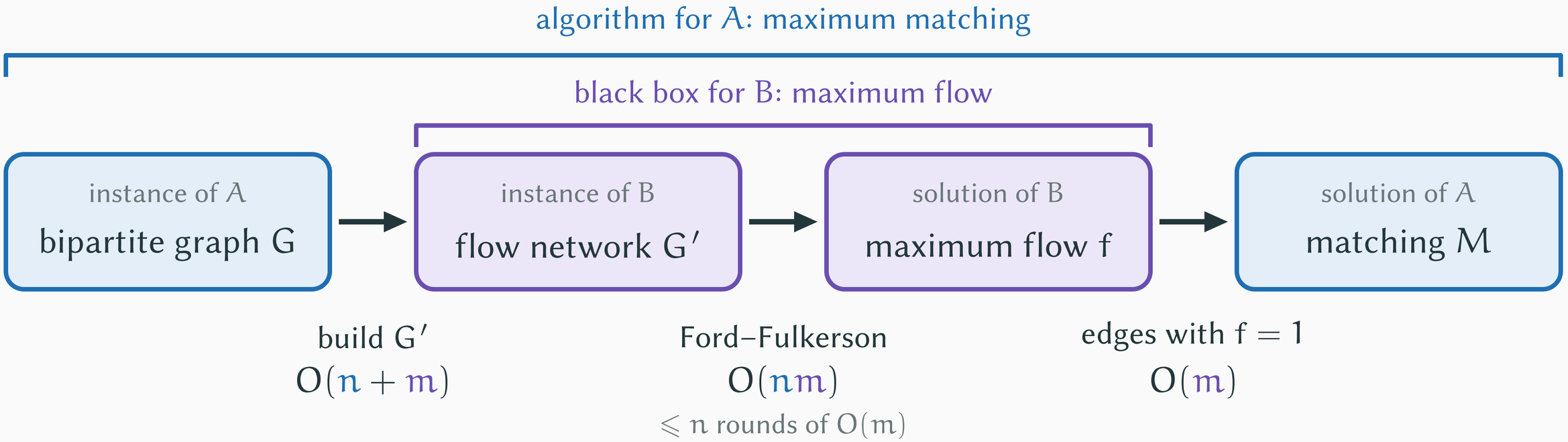


value 2, but not a matching



value 2, and a matching

Polynomial-time reductions



Polynomial-time reduction.
A **reduces** to B: to solve A, transform the instance, call a black box for B,
and transform the answer back. Both transformations take polynomial time.

Hence, if B can be solved in polynomial time, so can A.
Here: $O(nm)$ in total.

How fast can we find a maximum matching?

Bipartite graphs

a maximum flow in G'

1956	Ford–Fulkerson on G' (today)	$O(nm)$
1973	Hopcroft–Karp	$O(m\sqrt{n})$
2004	Mucha–Sankowski, randomised (not max flow)	$O(n^\omega)$
2022	Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva	$m^{1+o(1)}$

$\omega \leqslant 9/4$: the exponent of matrix multiplication (2026).

General graphs

no flow network: odd cycles

1965	Edmonds: the first P-time algorithm (blossoms)	$O(n^4)$
1980	Micali–Vazirani	$O(m\sqrt{n})$
2004	Mucha–Sankowski, randomised	$O(n^\omega)$
open	almost-linear time?	$m^{1+o(1)}$?

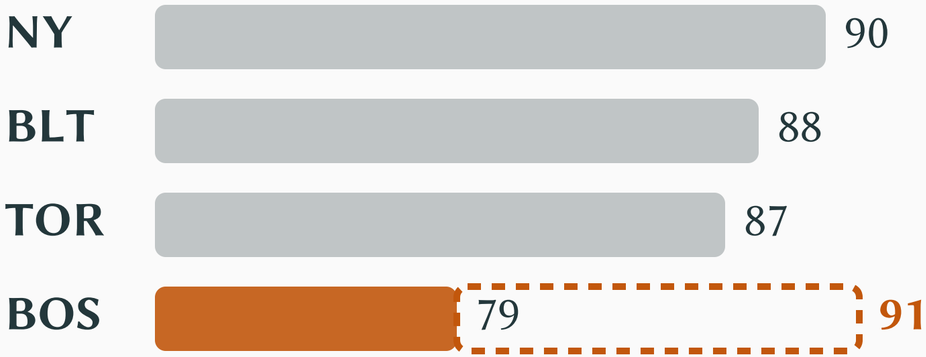
> Every faster maximum-flow algorithm is a faster bipartite matching algorithm: the reduction uses maximum flow as a black box.

Baseball elimination

Baseball elimination

	w_x		g_{xy}: games left against			
	wins	left	NY	BLT	TOR	BOS
New York	90	11	–	1	6	4
Baltimore	88	6	1	–	1	4
Toronto	87	11	6	1	–	4
Boston	79	12	4	4	4	–

No draws: every game has a winner.
Most wins finishes first. Ties count.



? *Can Boston still finish first?*

Problem.
Given all w_x, g_{xy} and a team z:
can z still finish first, possibly tied?

Football with 3–1–0 points (win, draw, loss): **NP-hard**.
with 2–1–0 points: still a flow problem.

Bernholt, Gülich, Hofmeister, Schmitt (1999)
Kern, Paulusma (2001)

Boston needs to win all its games

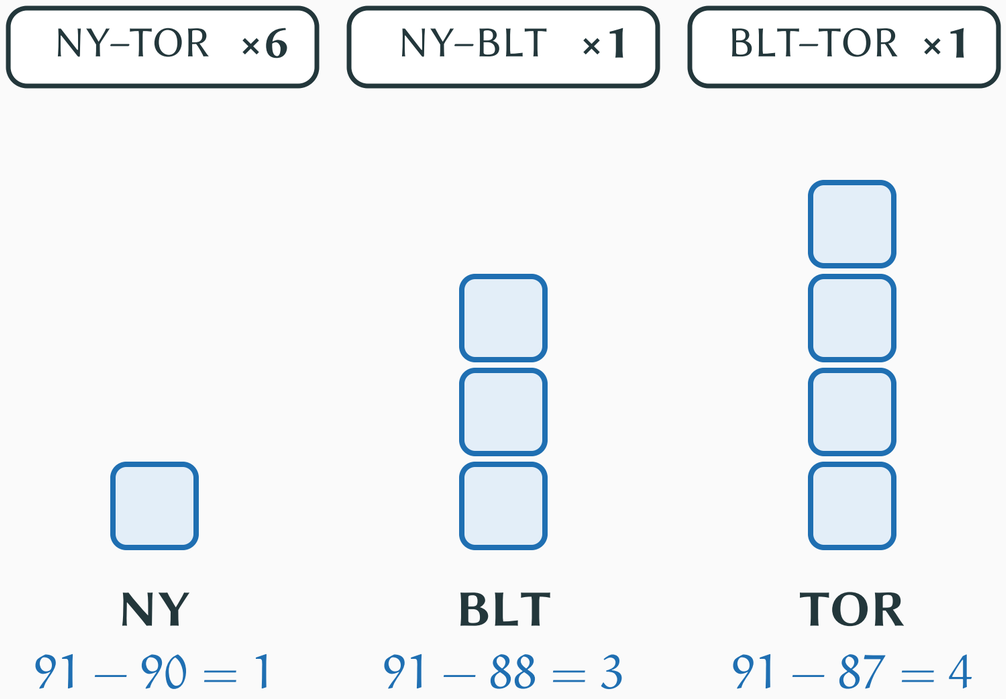
Observation. If Boston can finish first at all, it can after winning **all** its 12 games.

If it loses a game, let it win that game instead: it gains a win, someone else loses one.

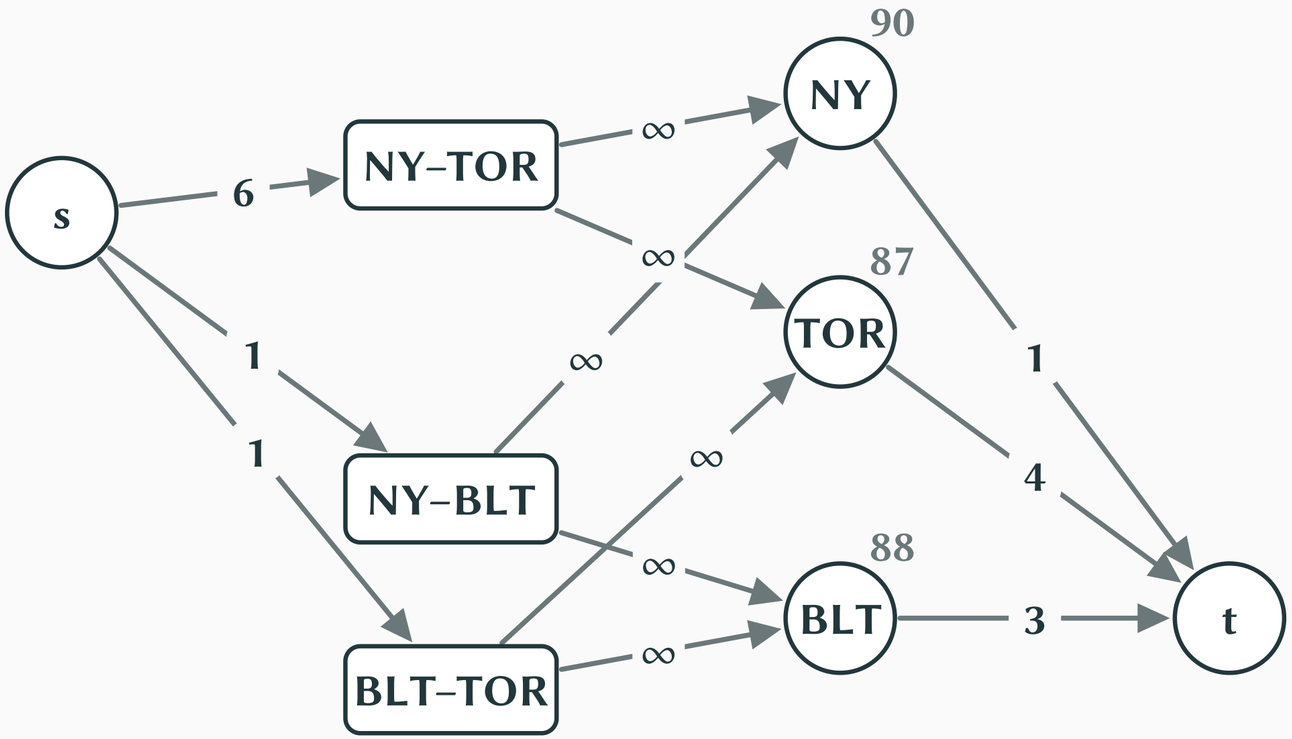
Thus Boston ends with $W = 79 + 12 = 91$ wins.

The other 8 games, between NY, BLT and TOR: can they be decided so that **nobody gets more than** 91?

? *8 games and room for* $1 + 3 + 4 = 8$ *wins. Is that enough?*



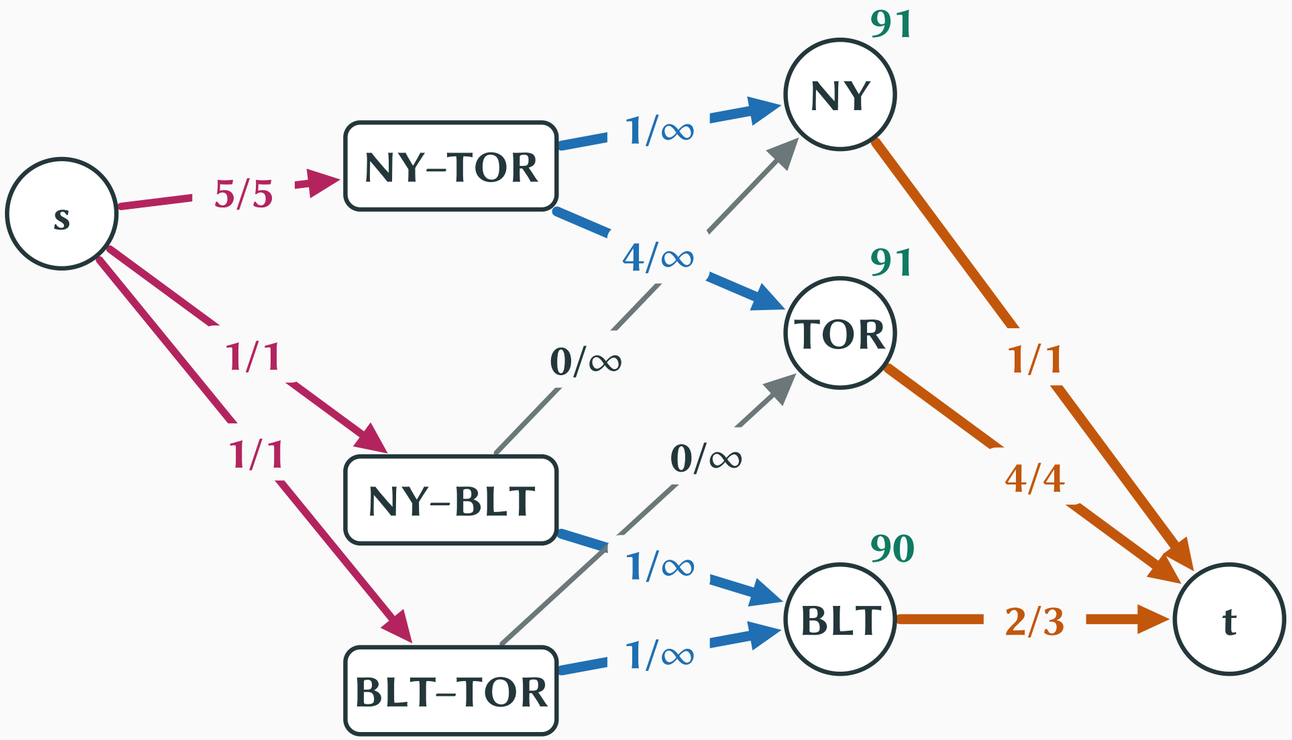
The flow network of a league



1. **Source and sink:** s and t.
2. **Teams:** a node x for every team except Boston, with its wins w_x.
3. **Games:** a node $\{x, y\}$ and an edge $(s, \{x, y\})$ of capacity g_{xy}.
4. **Winners:** edges from $\{x, y\}$ to x and to y, capacity ∞.
5. **Room:** an edge (x, t) of capacity $W - w_x$.

Claim.
Boston can finish first
$\iff$ the maximum flow value
is g^*, the number of games.

Proof of the claim



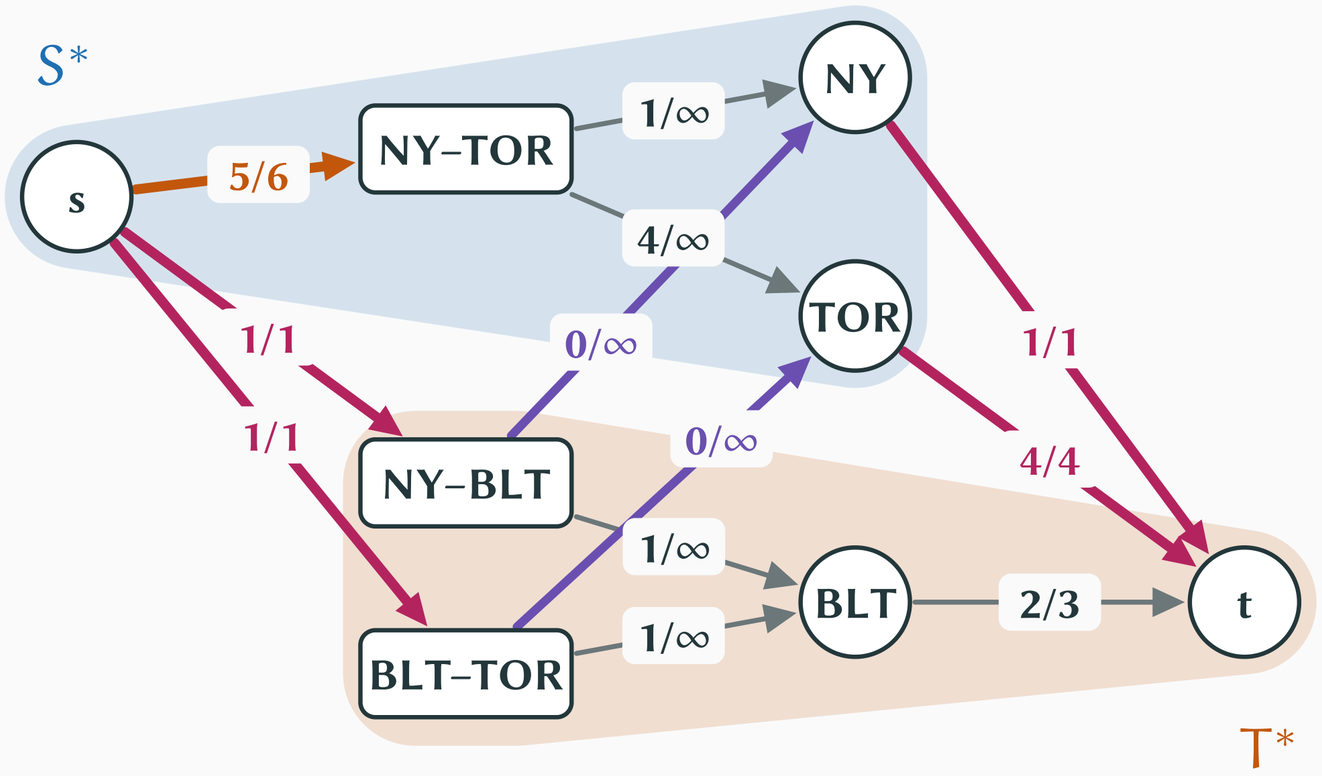
wins at the end: NY 91, BLT 90, TOR 91, BOS 91

Variant: **5** games NY–TOR.
Then $g^* = 7$,
and the maximum flow value is 7.

1. Source edges full:
   every game is decided.
2. A unit from $\{x, y\}$ to x is
   a game won by x.
3. Team x wins $f(x, t) \leqslant W - w_x$:
   at most 91 **wins**.

Conversely.
An outcome with at most W wins
for everybody gives a flow of value g^*.

Worked example: Boston is eliminated

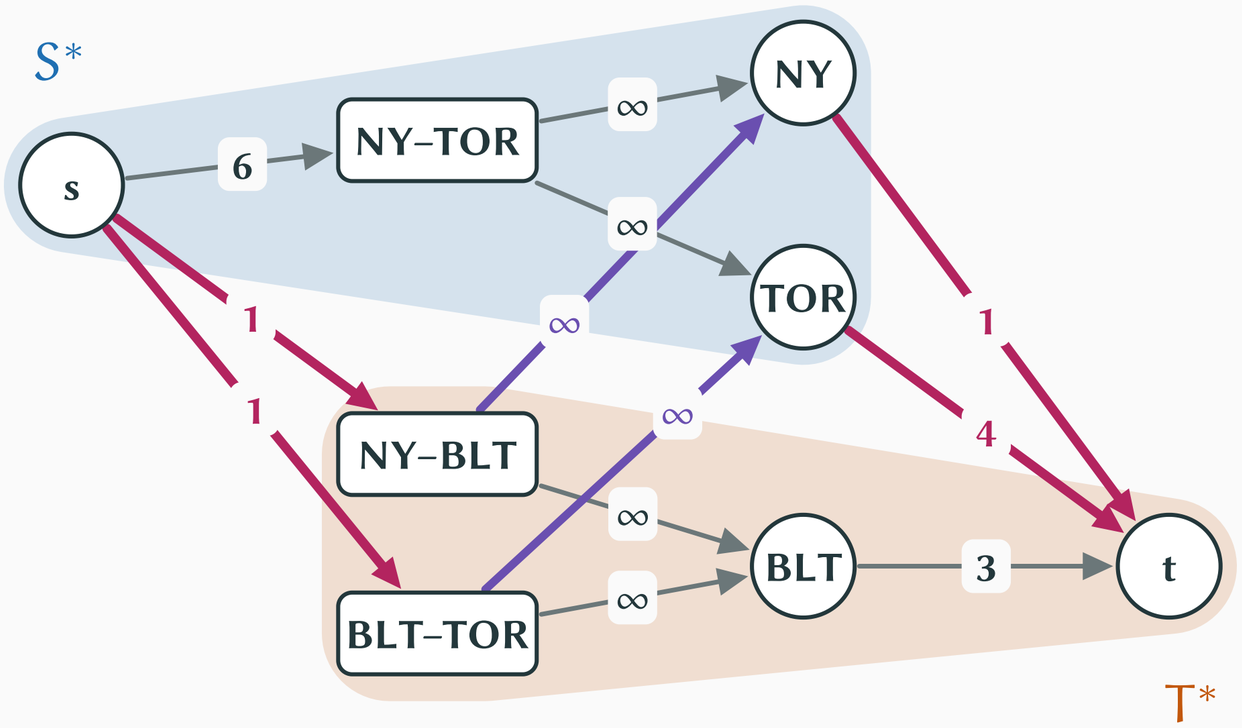


1. s ⟶ NY–BLT ⟶ NY ⟶ t, +**1**
2. s ⟶ NY–TOR ⟶ TOR ⟶ t, +**4**
3. s ⟶ BLT–TOR ⟶ BLT ⟶ t, +**1**
4. s ⟶ NY–TOR ⟶ NY ⟵ NY–BLT ⟶ BLT ⟶ t, +**1**

value 7

> $7 < 8 = g^*$: one NY–TOR game is left over. Boston is eliminated.

Another way to think: the minimum cut



$c(S^*, T^*) = 1 + 1 + 1 + 4 = 7 < 8$

1. S^*: the nodes reached from s in the last G_f.
2. $R = \{NY, TOR\}$: the teams in S^*.
3. NY and TOR end with at least $w(R) + g(R) = 90 + 87 + 6 = 183$ wins, but at most $W|R| = 182$.

> With $S = \{s\} \cup R \cup \{\text{games inside } R\}$:
> $c(S, T) = g^* - g(R) + W|R| - w(R)$,
> so $c(S, T) < g^*$ iff $w(R) + g(R) > W|R|$.

Then, on average, the teams in R win more than W games:
at least one of them wins more than W.

Open-pit mining

Open-pit mining

Blocks of earth: mining z **costs** c_z and **yields** ore worth p_z.

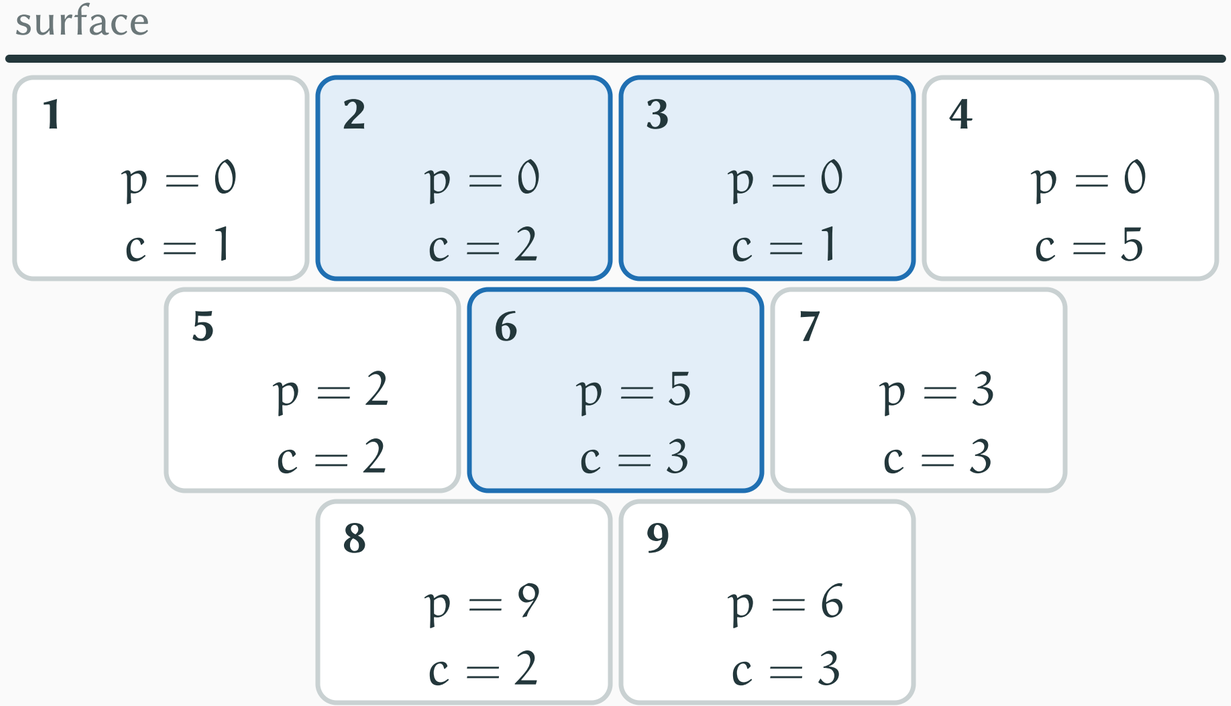
A block can be mined only **after the two blocks above it**.

> **Feasible set.**
> S contains, with each block, the blocks above it.

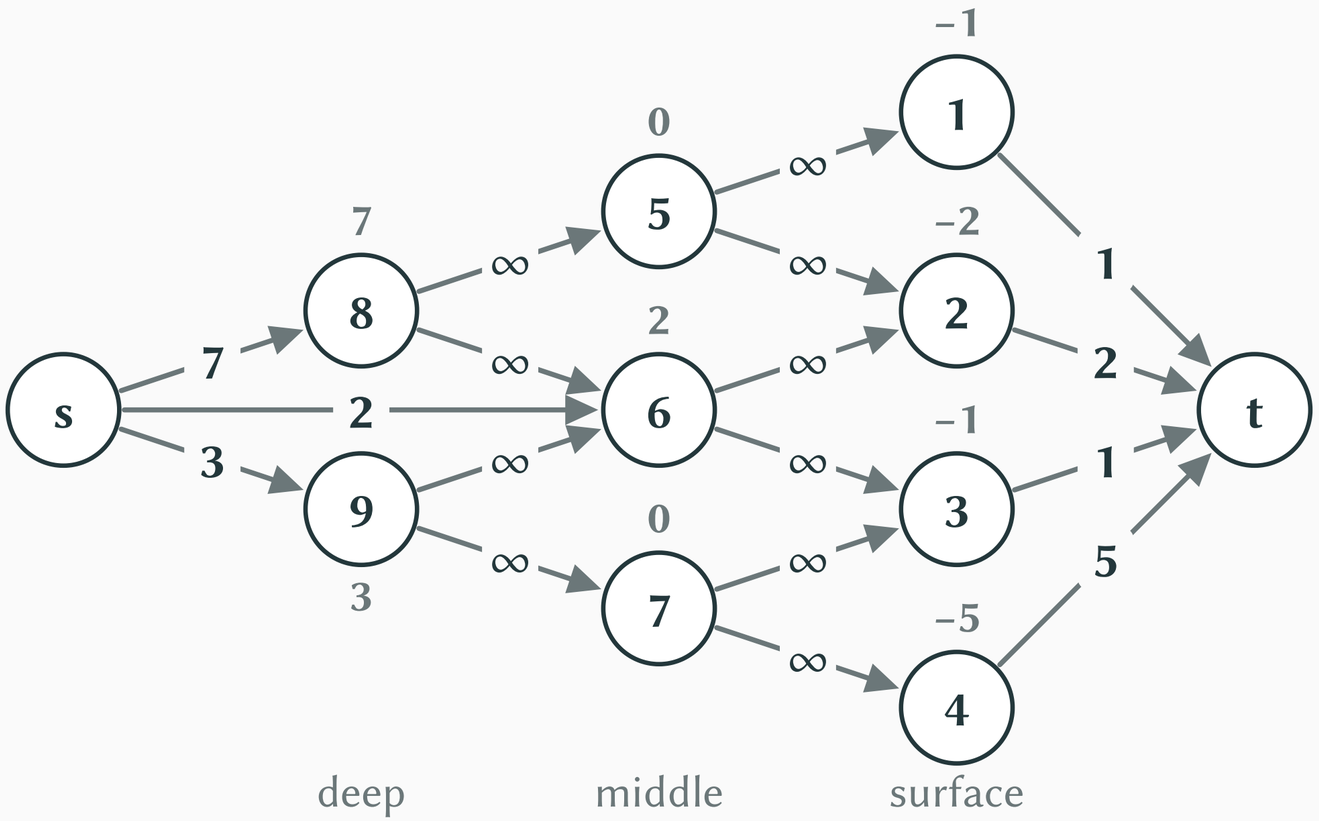
$\{6\}$ is not feasible.

$\{2, 3, 6\}$ is, with **profit** $\sum_{z \in S}(p_z - c_z) = -2 - 1 + 2 = -1$.

? *Which feasible set is best?*



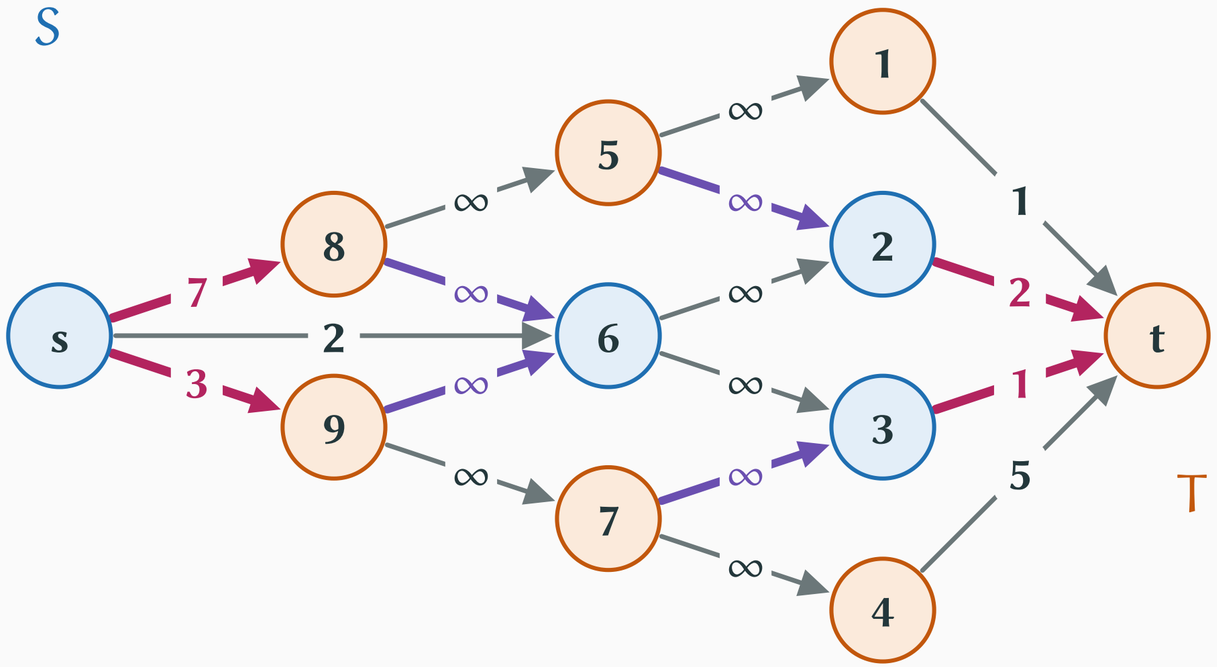
The flow network of a pit



1. **Source and sink:** s and t.
2. **Blocks:** a node z with its net value $b_z = p_z - c_z$.
3. $b_z > 0$: edge (s, z), capacity b_z.
4. $b_z < 0$: edge (z, t), capacity $-b_z$.
5. z needs x: edge (z, x), capacity ∞.

Claim.
A minimum cut $(S \cup \{s\},\ T \cup \{t\})$ gives a best set S to mine.

A finite cut is a feasible set



$$c(S, T) = 7 + 3 + 2 + 1 = 13$$

Mine the blocks in S,
the source side of the cut.

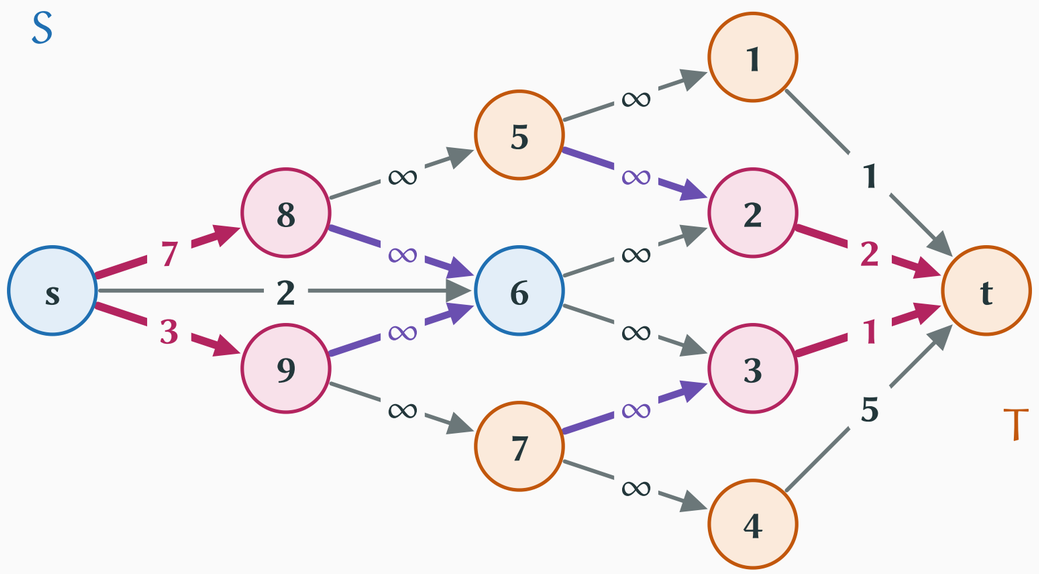
$S = \{6\}$: the edges $(6, 2)$ and $(6, 3)$ leave S.
Capacity ∞.

$S = \{2, 3, 6\}$: no infinite edge leaves S.
Capacity 13.

Fact.
The cut of S has finite capacity
iff S is feasible.

The cut of $S = \emptyset$ has capacity
$C = 2 + 7 + 3 = 12$: so a minimum cut
is finite, hence feasible.

Capacity of a cut = C − profit



$L = \{6, 8, 9\}$: the blocks with an edge from s

$R = \{1, 2, 3, 4\}$: the blocks with an edge to t

$C = \sum_{z \in L} b_z = 2 + 7 + 3 = 12$

the cut of $S = \{2, 3, 6\}$: what does each block contribute?

block z	1	2	3	4	5	6	7	8	9
net value b_z	−1	−2	−1	−5	0	2	0	7	3
side of the cut	**T**	**S**	**S**	**T**	**T**	**S**	**T**	**T**	**T**
contributes	0	2	1	0	0	0	0	7	3

$c(S, T) = 2 + 1 + 7 + 3 = 13 \;=\; C - \text{profit}(S) = 12 - (-1)$

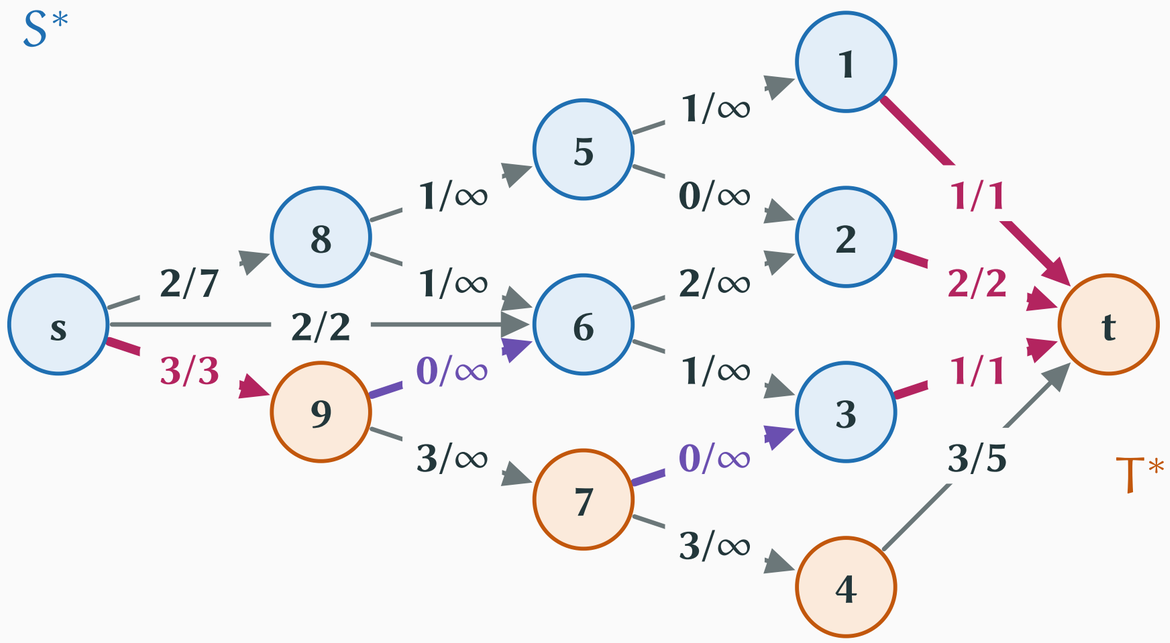
Waste we dig ($z \in S \cap R$) contributes $-b_z$.

Ore left in the ground ($z \in T \cap L$) contributes b_z.

$$\begin{aligned} c(S, T) &= \textstyle\sum_{z \in S \cap R} (-b_z) + \sum_{z \in T \cap L} b_z \\ &= \textstyle\sum_{z \in S \cap R} (-b_z) + C - \sum_{z \in L \setminus T} b_z \\ &= \textstyle\sum_{z \in S \cap R} (-b_z) + C - \sum_{z \in S \cap L} b_z \\ &= C - \textstyle\sum_{z \in S} b_z. \end{aligned}$$

Fact. $c(S, T) = C - \text{profit}(S)$ for every feasible S: a minimum cut is a best set.

Worked example: the best pit

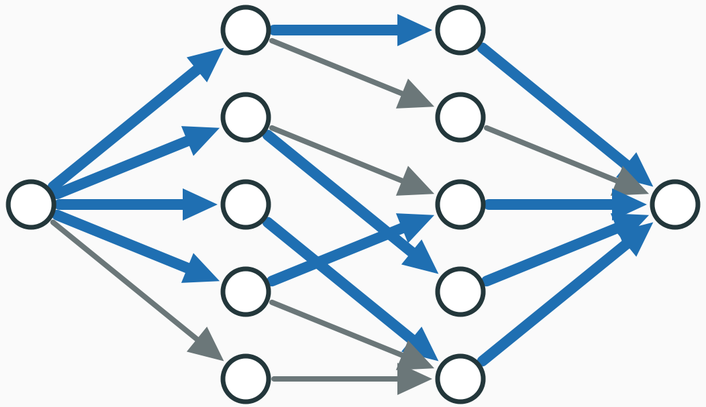


1. Build the network.
2. Maximum flow, path by path:
   1. s ⟶ 6 ⟶ 2 ⟶ t, +**2**
   2. s ⟶ 8 ⟶ 5 ⟶ 1 ⟶ t, +**1**
   3. s ⟶ 8 ⟶ 6 ⟶ 3 ⟶ t, +**1**
   4. s ⟶ 9 ⟶ 7 ⟶ 4 ⟶ t, +**3**
   value 7
3. S^*: the nodes reached from s in the last G_f.
4. Mine the blocks in S^*.
   Profit $= C - \mathrm{val}(f) = 12 - 7 = 5$.



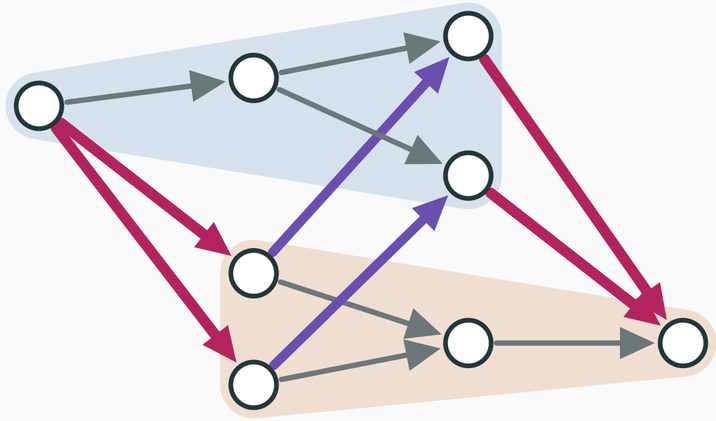
Three models, one recipe

Matching



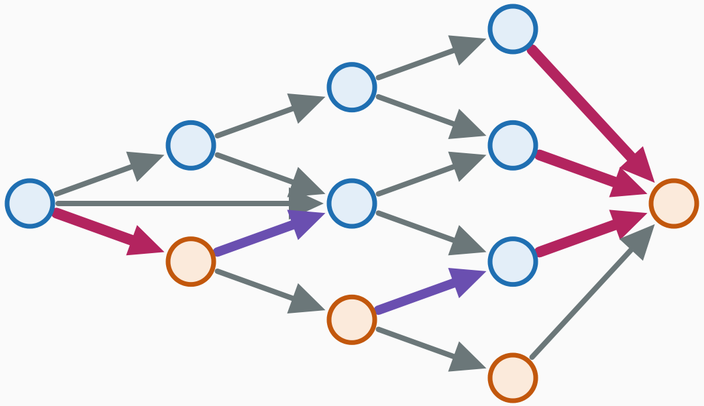
a flow is a set of choices:
$|M| = \mathrm{val}(f)$

Baseball elimination



a flow fills all games,
or a cut explains “no”

Open-pit mining



a cut is a set of blocks:
$\text{profit} = C - c(S, T)$

instance $\rightarrow$ network $\rightarrow$ maximum flow or minimum cut $\rightarrow$ answer

Summary

1. **Build a network, call a black box.**
   A polynomial-time reduction: transform the instance,
   solve a flow or cut problem, transform back.

2. **Flows make choices.** Integral flows are matchings.
   A flow filling every game decides them all.
   A minimum cut explains an elimination.

3. **Cuts select sets.** $c(S, T) = C - \text{profit}(S)$:
   a minimum cut is a best set to mine.

> Matching in $O(nm)$. Baseball elimination
> and open-pit mining in polynomial time.

Reference: KT, Chapters 7.5, 7.11, 7.12; CLRS, Chapter 24.3