Admin:

→ **QUIZ I IS (STILL) COMING**
- Info handout will be up today on course website
  PLEASE MAKE SURE TO READ IT CAREFULLY!

→ PSet 4 due this Wed
PSet 5 out 3/16 (after the Quiz), due on 3/22

Today: Algorithms for computing max flow
→ Recap of the Max Flow Min Cut Thm
→ Recap of Ford-Fulkerson alg.
→ Flow integrality
→ Maximum bottleneck path & Edmonds-Karp algs.
→ Applications of max flow: Bipartite matching problem, (Baseball) Elimination

---

Last time: Network $G = (V, E, s, t, c)$ capacities: $c: E \to \mathbb{R}^{\geq 0}$

(Net) Flow: $f: V \times V \to \mathbb{R}$ (defined for $\forall u,v$; can be negative)

such that:
- → $f(u,v) \leq c(u,v)$ $\forall_{u,v}$ (feasibility)
- → $\sum_u f(u,v) = 0$ $\forall_{v \neq s,t}$ (flow conserv.)
- → $f(u,v) = -f(v,u)$ $\forall_{u,v}$ (skew symmetry)

Flow value: $|f| = \sum_v f(s,v)$

Maximum Flow problem: Given $G = (V, E, s, t, c)$,
find a flow in G of maximum value
(max flow might not be unique!)

Notation: F^* = value of max flow

s-t cut: A cut $(S, V \setminus S)$ s.t. $s \in S$ & $t \in V \setminus S$

→ **capacity of a cut:**

$$c(S) = c(S, V \setminus S) = \sum_{u \in S} \sum_{v \in V \setminus S} c(u,v)$$

(= capacity of edges leaving S)

→ **net flow f across the cut:**

$$f(S) = f(S, V \setminus S) = \sum_{u \in S} \sum_{v \in V \setminus S} f(u,v)$$

Note: $\boxed{f(S) \leq c(S)}$ (*) for any flow (by feasibility)

Minimum s-t cut problem: Given $G = (V, E, s, t, c)$,

find an s-t cut of minimum capacity

Claim: For any s-t cut S, and any flow f

(Proved last time) $\boxed{|f| = f(S)}$

⇒ If S^* is a minimum s-t cut and f^* is a max flow,

then $\boxed{F^* = |f^*| = f^*(S^*) \overset{(*)}{\leq} c(S^*)}$

↑ Weak duality of flows & s-t cuts

[More in Lecture 9]

How to find a max flow?

Residual network: $G_f = (V, E, s, t, c_f)$ of flow f in network G

→ **residual capacities**

$$c_f(u,v) = c(u,v) - f(u,v)$$

(= how much extra net $u \to v$ flow can be directly send)

Note: By feasibility of f, $0 \le c_f(u,v) \le c(u,v) + c(v,u)$

Captures the possibility of pushing some existing $v \to u$ flow "back".

→ edge $(u,v) \in E_f$ whenever $c_f(u,v) > 0$

(discards saturated edges)

Augmenting path = directed s-t path in G_f

→ Can be used to push additional flow on such path P up to (residual) bottleneck capacity

$$c_f(P) = \min_{(u,v) \in P} c_f(u,v)$$

(i.e., after the push $|f|$ increases by $c_f(P)$)

Note: $c_f(P) > 0$ since G_f contains only edges with positive c_f

---

Ford-Fulkerson alg. [1956]

Idea: Just keep increasing the flow with augmenting paths

Alg.: → $f(u,v) \leftarrow 0 \ \forall_{u,v}$ (Start with zero flow)

This graph changes in each iteration! (as f changes)

Each iteration takes $O(E)$ time (via DFS)

→ While an augmenting path P exists in G_f:

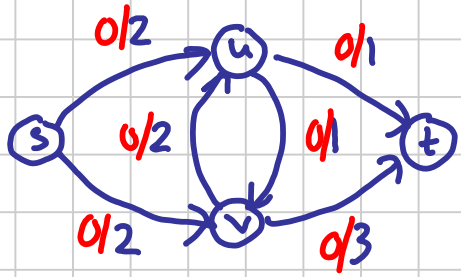
Augment f along P (increasing the value by $c_f(P)$) ✓ > 0

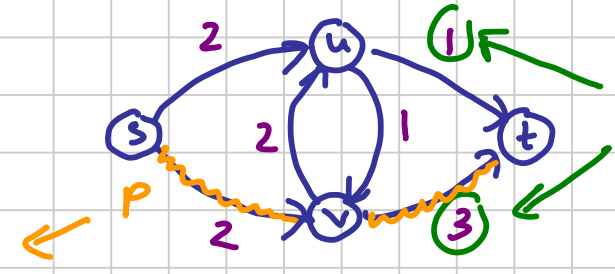
→ Output f

Example execution of F-F alg.:

Flow: | Residual network:

1)

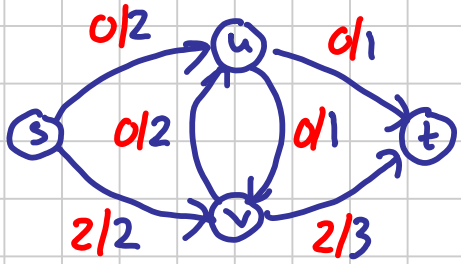


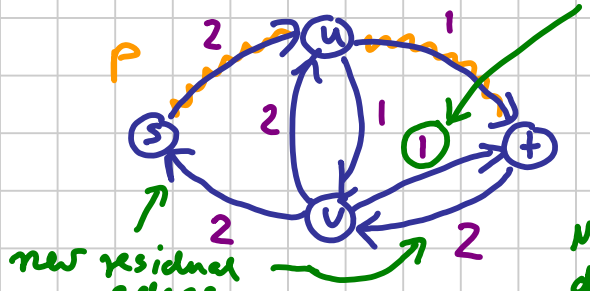


residual capacities (initially = org. capacity)

$P = (s, v, t)$ $c_f(P) = 2$

2)





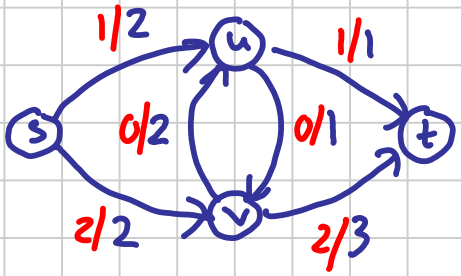
reduced original capacity

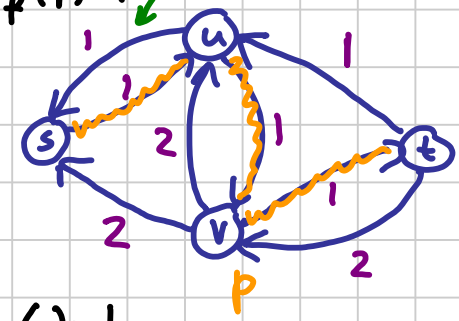
new residual edges

Note: (s,u) disappeared as it is saturated

3)

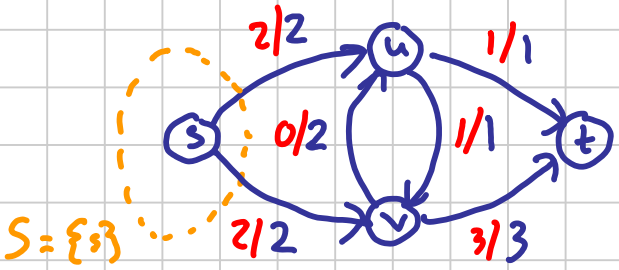
$P = (s, u, t)$ $c_f(P) = 1$

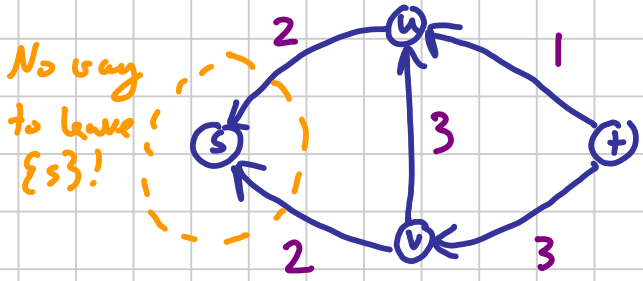




4)

$P = (s, u, v, t)$ $c_f(P) = 1$





No more augmenting paths!

→ Is this flow a maximum flow?

Yes, as the s-t cut $\{s\}$ has capacity 4

$\Rightarrow$ (by weak duality) $4 \geq F^* = |P| = 4$

F-F alg. works correctly here, but is it correct always?

Max Flow Min Cut Theorem

The following statements are equivalent:

1. $|f| = c(S)$ for some s-t cut S
2. f is a max flow (i.e., $F^* = |f|$)
3. f admits no augmenting path (i.e., there is no s-t path in G_f)

Note: → ③ ⇒ ② implies that F-F alg. is always correct

→ ① ⇔ ② implies that if S^* is the min s-t cut then

$$F^* = c(S^*)$$

Strong duality of flows & s-t cuts (implies weak duality)

⇒ s-t cuts are the only bottleneck for pushing flows in networks

---

Proof of the MFMC Thm:

Will show ① ⇒ ② ⇒ ③ ⇒ ①

① ⇒ ②: By weak duality, we know that

$$|f| \leq F^* \leq c(S^*) \leq c(S) = |f|$$

($|f| \leq F^*$: def. of F^*; $F^* \leq c(S^*)$: weak duality; $c(S^*) \leq c(S)$: def. of S^*; $c(S) = |f|$: ①)

$$\Rightarrow |f| = F^*$$

② ⇒ ③: Will show Not ③ ⇒ Not ②

But: Not ③ = ∃ augmenting path P in G_f

⇒ Augmenting f with P increases $|f|$

⇒ f could not be max. ⇒ Not ②

(3) ⇒ (1): No augmenting path

⇒ t not reachable from s in G_f

⇒ If $\hat{S} = \{v \mid \exists\ s \rightsquigarrow v \text{ path in } G_f\}$ then

→ $s \in \hat{S}$ & $t \notin \hat{S}$

⇒ $\hat{S}$ is an s-t cut & the residual capacity

$$c_f(\hat{S}) = \sum_{u \in \hat{S}} \sum_{v \in V \setminus \hat{S}} c_f(u,v) = 0$$

(otherwise $V \setminus \hat{S}$ reachable from $\hat{S}$ (and thus s) in G_f)

⇒ $c_f(\hat{S}) = 0$

⇒ $\forall_{u \in \hat{S}} \forall_{v \in V \setminus \hat{S}}$ $\quad 0 = c_f(u,v) = c(u,v) - f(u,v)$

⇒ $\forall_{u \in \hat{S}} \forall_{v \in V \setminus \hat{S}}$ $\quad c(u,v) = f(u,v)$

⇒ $c(\hat{S}) = f(\hat{S})$ ⇒ (1) ∎

Running time of Ford-Fulkerson alg.?

Recall: Each iteration/augmentation takes $O(E)$ time

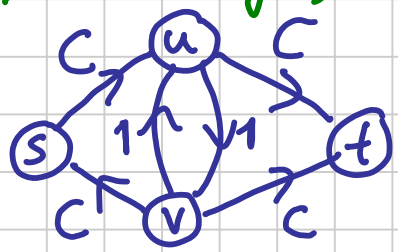
How many iterations/augmentations is there? (in the worst case)

(1) if capacities are integers $\in [0, C]$ then

# of augment. $\leq F^* \leq c(\{s\}) \leq |V| \cdot C$

(since $c_f(p) \geq 1$ always) ($\leq |V|$ edges leaving s)

→ Can happen!



Alternate paths: $s \to u \to v \to t$ & $s \to v \to u \to t$

⇒ Total runtime: $O(E \cdot V \cdot C)$

→ This running time makes F-F alg. be a pseudopolynomial algorithm = polynomial in sum of input numbers
i.e., polynomial in unary encoding of input
(NOT binary)

② If capacities rational
⇒ running time still finite (& pseudopoly.)

③ If capacities are real
⇒ potentially infinite runtime!

Important observations:

→ If residual capacities are all integral
⇒ $c_f(p)$ is integral as well

→ If $c_f(p)$ is integral & f is integral (i.e., $f(u,v) \in \mathbb{Z} \; \forall_{(u,v)}$)
⇒ residual capacities stay integral after augmentation & the flow

⇒ If network has all capacities integral then F-F alg. will find a max flow with all $f(u,v)$ being integral

⇒ Flow integrality thm: If $G=(V,E,s,t,c)$ has all capacities integral, then $\exists$ a flow f s.t. $|f| = F^*$ & both F^* and f are integral

Very useful when applying max flow ↓

Note: There still can $\exists$ max flow f' that is NOT integral (but $|f'| = F^*$ will still be integral)

From now on: Assume capacities integral

⇒ F-F runtime: $O(V \cdot E \cdot C)$

→ OK-ish when $C = \max_e c(e)$ small, e.g., $C = 1$ (unit-capacity case)

but not so much otherwise

Can we get a better algorithm for large C?

Idea: Ford-Fulkerson alg. just picks any augmenting path. Is there a smart choice of augmenting path?

Maximum bottleneck path = augmenting path P that maximizes bottleneck capacity $c_f(P)$

Exercise: Max bottleneck path can be found in $O(E \log V)$ time

(Do binary search to find the max capacity c^*
Note: $C \leq c^*$ iff $\exists$ s-t path in G_f after removing all edges (u,v) with $c_f(u,v) < C$)

Maximum bottleneck path (MBP) algorithm:

→ $f(u,v) = 0 \quad \forall_{u,v}$

→ While $\exists$ augmenting path: ($O(E \log V)$ time)

→ Find augmenting path P with maximum bottleneck capacity $c_f^* = c_f(P)$

→ Augment the flow with P
(so, $|f|$ increases by c_f^*)

→ Output f

What is the running time?

Claim: In any graph $G=(V, E, s, t, c)$, there exists an s-t path P in G with

$$c(P) = \min_{e \in P} c(e) \geq \frac{F^*}{m}$$

$m = |E|$
$n = |V|$

Proof: Let f^* be the max flow in G

$\Rightarrow$ By flow decomposition lemma, f^* decomposes into $\leq m$ flow paths & cycles

$\Rightarrow$ The whole value $|f^*| = F^*$ of flow pushed by f^* is supported on $\leq m$ flow paths
(cycles do not push any flow from s to t)

$\Rightarrow$ By averaging argument, at least one, P, of these $\leq m$ flow paths has to carry $\geq \frac{F^*}{m}$ flow

$\Rightarrow$ $c(P) \geq \frac{F^*}{m}$ ∎

Corollary: MBP alg. runs in $O(m^2 \log n \log nC)$ time

Proof: → Let F^* be the max flow value of the input network G

→ Let F_j be the amount of flow pushed by the MBP alg. up until iteration j

→ Let $F_j^* = F^* - F_j$ be the amount of flow "missing" at the beginning of iteration j
(= flow still to be pushed to get max flow)

$\Rightarrow$ Max flow value of the residual graph G_f in j-th iteration is F_j^*

$\Rightarrow$ By applying the Claim to G_f we get (NOT to G!) that the (residual) bottleneck capacity of the max bottleneck capacity augmenting path P we found has to be

$$c_f(P) \geq \frac{F_j^*}{m}$$

$$\Rightarrow \boxed{\forall_j \; F_{j+1}^* \leq \left(1-\frac{1}{m}\right) F_j^* \leq \left(1-\frac{1}{m}\right)^j F_1^* = \left(1-\frac{1}{m}\right)^j F^* }$$

$\Rightarrow$ Set $\boxed{j^* \leftarrow m \ln(F^*) \in O(m \log(nC))}$ (since $F^* \leq nC$)

After j^*+1 iterations, we have

$$F_{j^*+2}^* \leq \left(1-\frac{1}{m}\right)^{j^*+1} F^* < \left(1-\frac{1}{m}\right)^{j^*} F^* \leq 1$$

Recall: $\left(1-\frac{1}{k}\right)^k \leq e^{-1}$

$\Rightarrow$ But, by Flow integrality thm,

if $F_{j^*+2}^* < 1$ then $\boxed{F_{j^*+2}^* = 0}$

$\Rightarrow$ MBP alg. terminates after $j^*+1 = O(m \log nC)$ iterations

$\Rightarrow$ Total running time: $O(m^2 \log n \cdot \log nC)$ ∎

What did we gain?

→ When C small, e.g., $C = O(1)$, vanilla F-F alg. still better

$$O(mn) \quad \text{vs.} \quad O(m^2 \log^2 n)$$

Note: For connected graphs, $m \geq n-1$

→ But, the runtime dependence only logarithmic in C

$\Rightarrow$ MBP is a (weakly) polynomial time alg.!

Runtime polynomial in binary representation of numbers

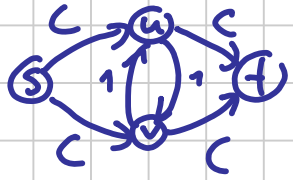
→ Runtime that is polynomial in $\log C$ is usually sufficient... but can we have NO dependency on C? (= strongly polynomial runtime)

Yes! Again: use a "smart" choice of augmenting path in F-F alg.

Edmonds-Karp alg.: Variant of F-F alg. in which we always choose the augmenting path with fewest number of edges (i.e., shortest path if each edge in G_f has length 1) [1972]

Note: Can find such path in $O(m)$ time via BFS exploration

This choice avoids the "bad" instance for F-F alg. →



E-K alg. (& MBP alg.!) need only two iterations to find max flow here

Claim: Edmonds-Karp alg. runs in $O(m^2 n)$ time

↖ Indeed, no dependency on C!

Proof: See appendix of recitations notes
(You are NOT responsible for this proof)

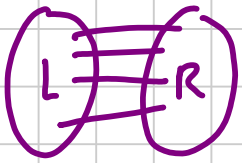
Highlights of later work on max flow algorithms:

→ Strongly polynomial time:
- [King Rao Tarjan '94]: $O(mn \log_{\frac{m}{n \log n}} n)$
  (= $O(mn)$ if $m = n^{1+\epsilon}$ $\epsilon > 0$)
- [Orlin '13]: $O(mn)$

→ (Weakly) polynomial time:
- [Goldberg Rao '98]: $O(\min\{m^{\frac{3}{2}}, mn^{\frac{2}{3}}\} \log \frac{n^2}{m} \log C)$
- [Lee Sidford '14]: $O(m\sqrt{n} \log^{O(1)} n)$
- unit-capacity: [Madry '13]: $O(m^{\frac{10}{7}} \log^{O(1)} n)$

Applications of Max Flow

Bipartite graph = graph $G=(V,E)$ with two sides L and R s.t. all edges go between L and R



In other words: $G=(V,E)$ s.t. $V = L \cup R$ & $E \subseteq L \times R$

($\cup$: disjoint union; $\times$: Cartesian product)

Matching = subset of edges s.t. no two of them share a vertex

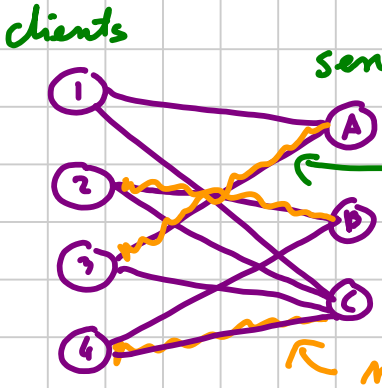
Maximum bipartite matching problem: Given bipartite $G=(L \cup R, E)$ find a matching of maximum cardinality

Think: L = clients / jobs / groups / ...
R = servers / machines / slots / ...

(Here: each client/job/... needs exactly one exclusive server/machine/...)

Matching = assignment of clients to servers, jobs to machines, groups to slots...

Maximum matching = assignment that satisfies the most clients / jobs / groups



edge (u,v) iff client u can be served by server v

Matching ensures that:
- each client uses ≤ 1 server
- each server serves ≤ 1 clients

How to compute maximum bipartite matching?

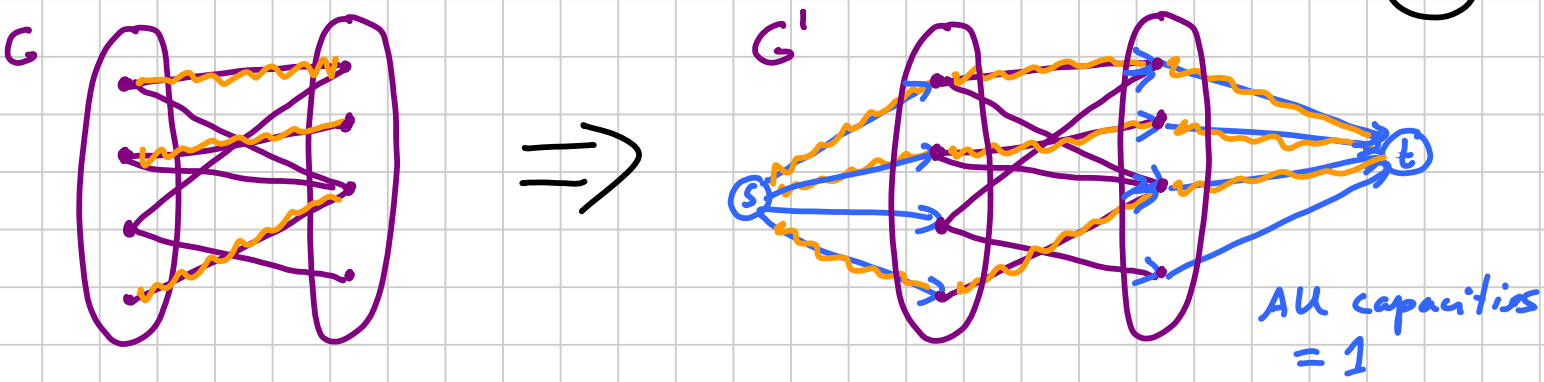
Reduce it to max flow problem!

We already know how to do that

→ = transform (quickly) an instance of max matching problem to an instance of max flow problem so as solving max flow instance gives you a solution to the max matching instance

Reduction:

(Note: Can be done in $O(m+n)$ time)



→ Direct all the edges from L to R
→ Add source s & sink t
→ Add edges from s to each $u \in L$ & from each $v \in R$ to t
→ Set all capacities to 1

Note: All capacities in G' are = 1 (and thus integral) ← C=1
⇒ By flow integrality thm, integral max flow f^* exists
⇒ Ford-Fulkerson alg. will find such f^* in $O(mn)$ time

Claim: $|f^*| = |M^*|$, where M^* = max matching in G

Proof: $|f^*| \geq |M^*|$: Let $M^* = \{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$

A flow f that pushes unit flow on each path $\langle s, u_i, v_i, t \rangle$ has a value $|f| = k = |M^*|$

But, obviously, $|f| \leq |f^*| = F^*$

$|f^*| \leq |M^*|$: f^* is integral, so is supported on exactly $|f^*|$ edges in $L \times R$. By feasibility & flow conserv. none of these edges share an endpoint

$\{s\} \cup L$ is an s-t cut containing only edges in $L \times R$
$f^*(\{s\} \cup L) = |f^*|$

⇒ they form a matching of size $|f^*|$
⇒ $|f^*| \leq |M^*|$

Max bipartite matching alg.

→ Compute G' — $O(m+n)$ time

→ Run F-F alg. on G' to compute f^* — $O(mn)$ time

→ Return a matching of edges in $L \times R$ that support (integral) f^* — $O(m+n)$ time

Runtime: $O(mn)$

Can we do better?

[Hopcroft Karp '73]	$O(m\sqrt{n})$
[Mucha Sankowski '04]	$O(n^\omega)$ ← exponent of fast matrix mult. ≤ 2.376 ← goal in dense graphs ($m = \Omega(n^2)$)
[Madry '13]	$O(m^{10/7} \log^{O(1)} m)$ ← goal in sparse graphs ($m = O(n)$)

Another application of max flow:

Baseball elimination → see next page

Baseball Elimination

Till 1993, two divisions in each league with only 4 playoff teams.
In 1995, a wildcard was added.

AL East standings, August 30, 1996.

roughly speaking!

	Team	w_i Wins	l_i Losses	r_i To Play	r_{ij} Against each other				
1	NY	75	59	28	–	5	7	4	3
2	Baltimore	71	63	28	5	–	2	4	4
3	Boston	69	65	28	7	2	–	4	0
4	Toronto	63	71	28	4	4	4	–	0
5	Detroit	◯	◯	28	3	4	0	0	–
					NY	Ba	Bo	T	D

All that most sports writers know is: (strictly less)

Team i is eliminated if $w_i + r_i < w_j$

If Detroit $w_5 = 46, l_5 = 88$, $46 + 28 = 74 < 75$.

Sufficient, but not necessary!

What if $w_5 = 47$? Still eliminated because $w_5 + r_5 = 75$ and either NY or Baltimore will win 76 games since they play each other 5 times!

Is Detroit Eliminated?

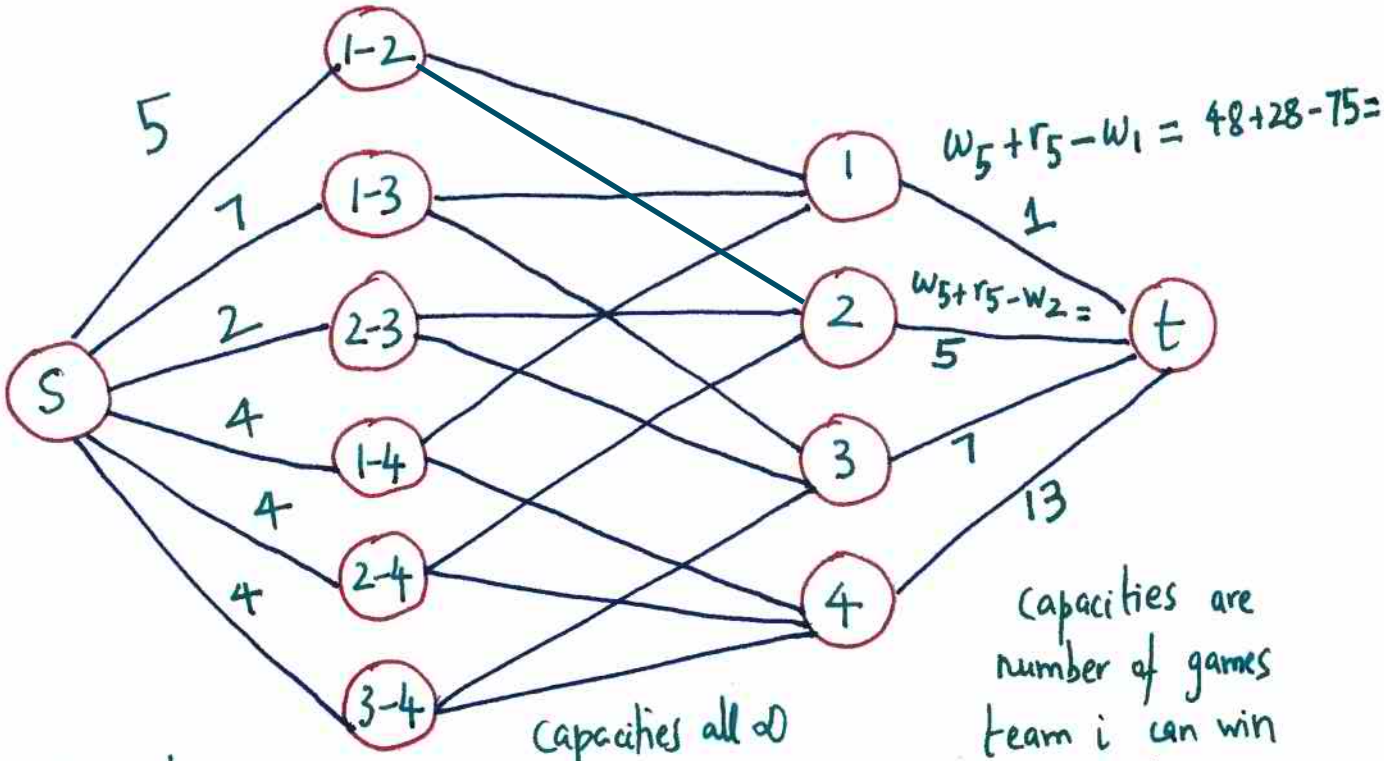
Eliminated if $w_5 = 46$ or 47. } $r_5 = 28$ all cases

What about $w_5 = 48$ $(l_5 = 86)$?

More difficult analysis but can show elimination.

Use max-flow!

Flow network to determine if team 5 is eliminated.



Capacities are games to be played between i and j

Capacities all ∞

Capacities are number of games team i can win and not have more wins than team 5

Intuition: Assume team 5 wins all remaining games. Divvy up remaining games so all teams have $\leq w_5 + r_5$ wins.

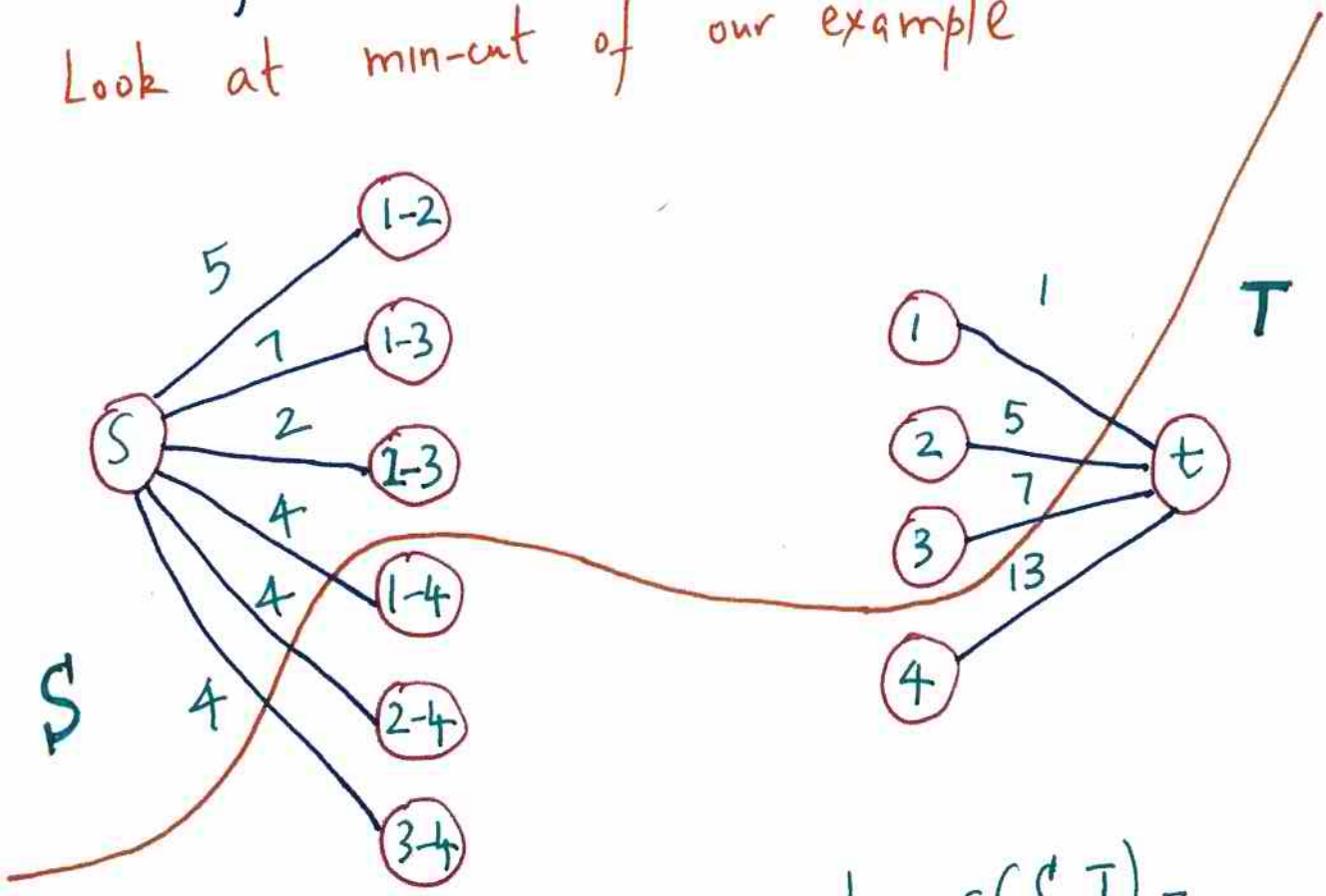
CLAIM

Theorem: Team 5 is eliminated iff max-flow does not saturate all edges leaving source. (i.e., max flow value < 26).

(Saturation corresponds to playing all the remaining games.)

Argument: If you can't play all the remaining games without exceeding the capacities of $i \to t$ edges, team 5 is eliminated.

Look at min-cut of our example



(S, T) above is min-cut. $c(S,T) =$

$4+4+4+1+5+7 = 25 \Rightarrow$ ELIMINATION.

EXPLANATION FOR SPORTS WRITERS

Look at subset of teams, namely, 1, 2 & 3.

$w_1 + w_2 + w_3 = 75 + 71 + 69 = 215$ wins

Teams play each other $5 + 7 + 2 = 14$ times.

$\therefore$ Total number of guaranteed wins

$= 215 + 14 = 229$

Observe $\frac{229}{3} = 76.33$

At least one of those teams will win 77 games.
(Detroit can win $48 + 28 = 76$)

Max-flow min-cut finds this subset if it exists.