Today: Algorithms for computing max flow
- → Recap of the last lecture
- → Ford-Fulkerson alg.
- → Max Flow Min Cut Theorem
- → 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: $c : V \times V \to \mathbb{R}^{\geq 0}$

$\boxed{\text{Convention: } c(u,v) = 0 \Rightarrow (u,v) \notin E}$

(Net) Flow: $f : \overbrace{V \times V}^{\text{defined for } \forall u,v} \to \mathbb{R}$ ← 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 $G = (V, E, s, t, c)$,

max flow might not be unique!

find a flow in G of maximum value

Notation: $F^* =$ value of max flow (i.e., $|f^*| = F^*$)

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) $\boxed{|f| = f(S)}$

$\Rightarrow$ 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: $G_f = (V, E_f, s, t, c_f)$ 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 we directly send)

Note: By feasibility of f, $0 \leq c_f(u,v) \leq c(u,v) + \underbrace{c(v,u)}$

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

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

(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

Also: If f is an s-t flow in G & f' is an s-t flow in G_f! ⇒ $f + f'$ is an s-t flow in G & $|f + f'| = |f| + |f'|$

Ford-Fulkerson alg. [1956]

Idea: Just keep increasing the flow with augmenting paths

Alg.: → $f(u,v) \leftarrow 0 \quad \forall u,v$ (start with zero flow)

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

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

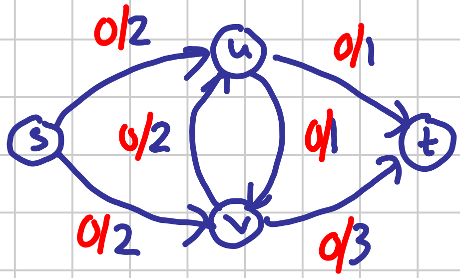
→ While an augmenting path P exists in G_f:
Augment f along P (increasing the value by $c_f(P)$) ✓

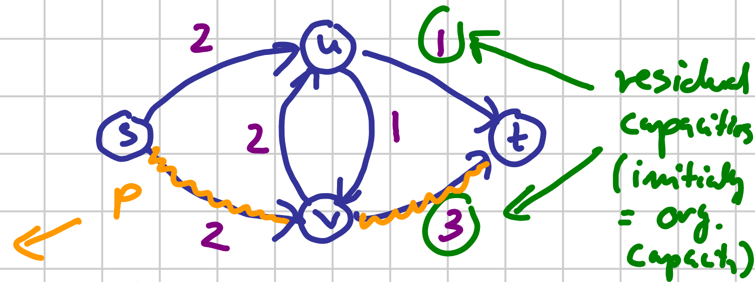
→ Output f

Example execution of F-F alg.:

Flow: | Residual network:

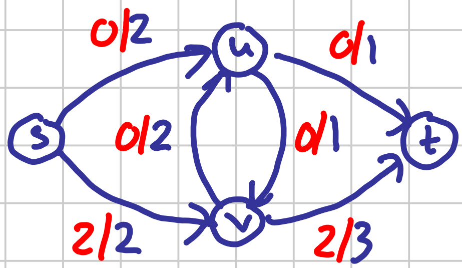
1)

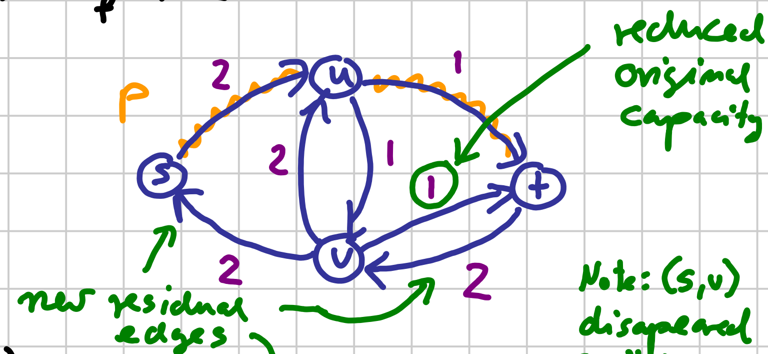




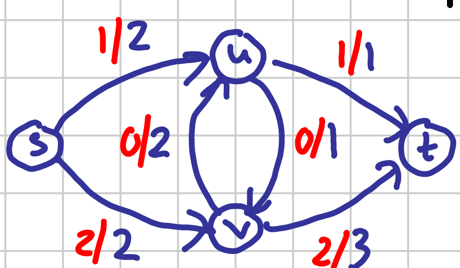
$P = (s, v, t)$ $\quad c_f(P) = 2$

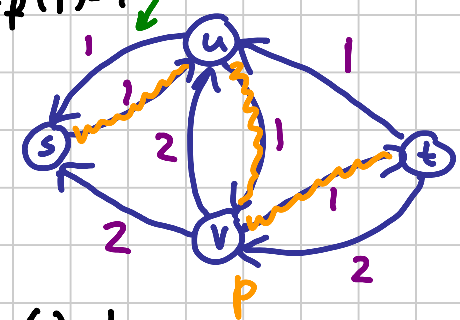
2)



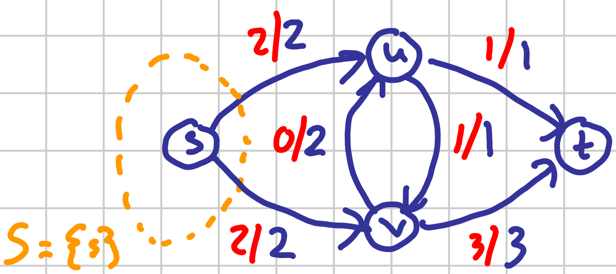


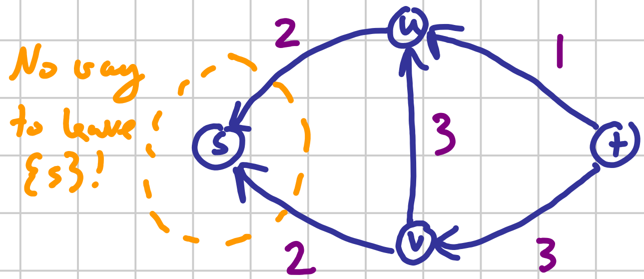
3) $P = (s, u, t)$ $\quad c_f(P) = 1$





4) $P = (s, u, v, t)$ $\quad c_f(p) = 1$





No more augmenting paths!

→ Is this flow a maximum flow?

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

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

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 $\boxed{F^* = c(S^*)}$

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

$$\underset{\text{def. of } F^*}{|f| \leq} F^* \underset{\text{weak duality}}{\leq} c(S^*) \underset{\text{def. of } S^*}{\leq} c(S) \underset{1}{=} |f|$$

$\Rightarrow |f| = F^*$

2 ⇒ 3: Will show Not 3 ⇒ Not 2

But: Not 3 = $\exists$ 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

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

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

⇒ $\boxed{c_f(\hat{S}) = 0}$

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

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

⇒ $c(\hat{S}) = f(\hat{S})$ ⇒ ① $\quad \boxed{\xi}$

Running time of Ford-Fulkerson alg.?

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

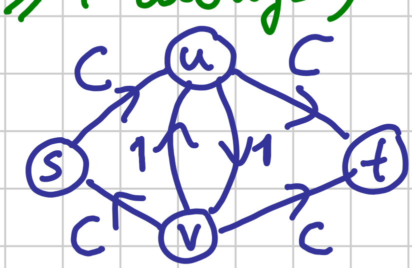
How many iterations/augmentations is there? (in the worst case)

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

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

(since $c_f(p) \geq 1$ always) $\quad$ see (+) for the reason why

→ Can happen!



Alternate paths:

$s \to u \to v \to t$

& $s \to v \to u \to t$

⇒ $\boxed{\text{Total runtime: } O(E \cdot V \cdot C)}$

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

② If capacities <u>rational</u>
⇒ running time still finite (& pseudopoly.)

③ If capacities are <u>real</u>
⇒ potentially <u>infinite</u> runtime!

<u>Important observations:</u> (+)

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

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

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

Very useful when applying max flow ⇓

⇒ <u>Flow integrality thm:</u> 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

---

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

From now on: Assume capacities integral (8)

$\Rightarrow$ F-F runtime: $O(V \cdot E \cdot C)$

$\rightarrow$ OK-ish when $\boxed{C = \max_e c(e)}$ small, e.g., $\underbrace{C=1}$

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: $c \le c^*$ 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:

$\rightarrow$ $f(u,v) = 0 \quad \forall_{u,v}$

$O(E \log V)$ time

$\rightarrow$ While $\exists$ augmenting path:

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

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

$\rightarrow$ 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

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

$$\boxed{\begin{matrix} m = |E| \\ n = |V| \end{matrix}}$$

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 $\boxed{F_j^* = F^* - F_j}$ 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

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

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

$\Rightarrow$ Set $\boxed{j^* \leftarrow m \ln(F^*) \overset{!}{=} O(m \log(nC))}$ (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 $\underbrace{F_{j^*+2}^* = 0}$

$\Rightarrow$ 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 represent. of numbers

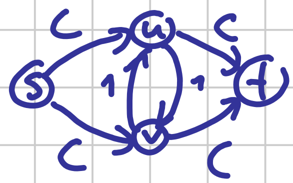
→ Runtime that is polynomial in $\log C$ is usually sufficient... but can we have NO dependency on C? (= strongly polynomial runtime)

Yes! Again: use a "smart" choice of augmenting path in FF 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 $\boxed{O(m^2 n)}$ 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:

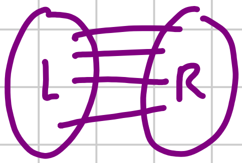
- [King Rao Tarjan '94]: $O(mn \log_{\frac{m}{n \log n}} n)$
  ($= O(mn)$ if $m = n^{1+\varepsilon}$ $\varepsilon > 0$)
- [Orlin '13]: $O(mn)$

→ (Weakly) polynomial time:

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

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. $V = L \uplus R$ & $E \subseteq L \times R$

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

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

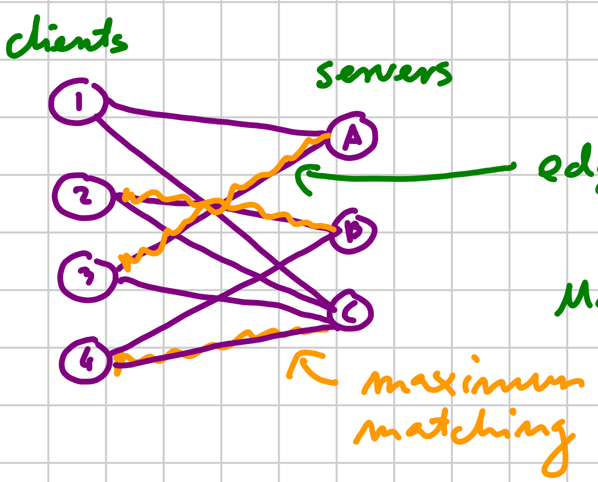
Maximum bipartite matching problem: Given bipartite $G=(L \uplus R, E)$ find a matching of maximum cardinality

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?

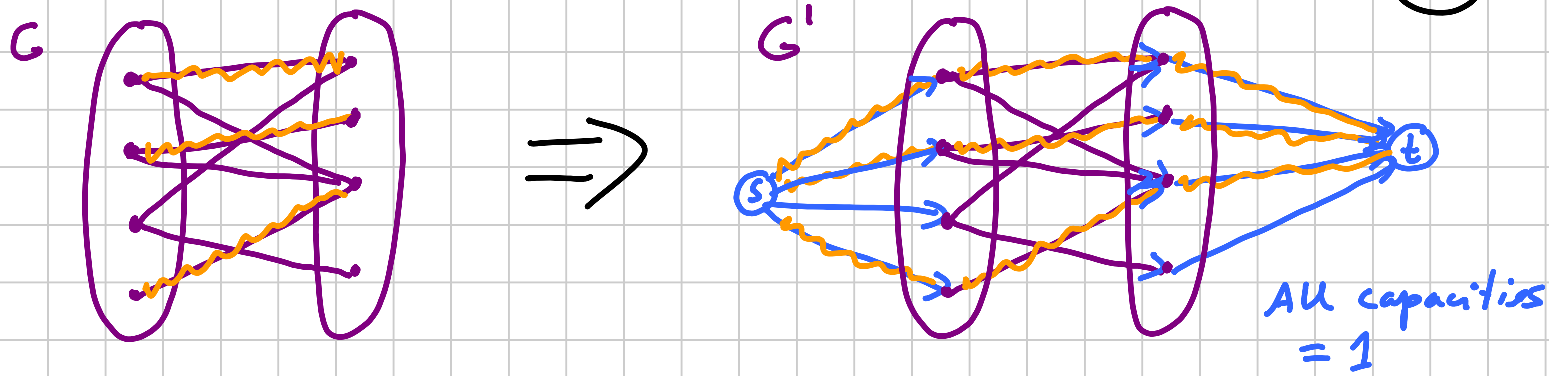
$\boxed{\text{Reduce}}$ it to max flow problem!

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) ← $C=1$

$\Rightarrow$ By flow integrality thm, integral max flow f^* exists

$\Rightarrow$ Ford-Fulkerson alg. will find such f^* in $\boxed{O(mn)}$ time

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

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

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

$\{s\} \cup L$ is an s-t cut containing only edges in $L \times R$

$f^*(\{s\} \cup L) = |f^*|$ →

$|f^*| \leq |M^*|$: 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

$\Rightarrow$ they form a matching of size $|f^*|$

$\Rightarrow$ $|f^*| \leq |M^*|$ ∎

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

[Madry '13] $O(m^{10/7} \log^{O(1)} n)$

good in dense graphs $(m = \Omega(n^2))$

good in sparse graphs $(m = O(n))$

Another application of max flow:

Baseball elimination → see next page