Max Flow (cont'd)

s-t Max Flow

Input: Directed graph $G=(V, E)$ with capacity $c: E \to \mathbb{R}^{76}$, $s, t \in V$. ($n=|V|$, $m=|E|$)

Output: Find a s-t flow f feasible for G s.t. $\|f\|$ is maximized.

Theorem. For any input $G=(V,E)$, $c: E \to \mathbb{R}^{20}$, $s,t \in V$ of s-t max flow.

$$\max_{f:\text{ feasible s-t flow}} \|f\| = \min_{(A,B):\text{ s-t cut}} c(A,B).$$

Pf. Assume $\exists u, v \in V$ s.t. $(u,v), (v,u) \in E$.

Given a feasible s-t flow $f: E \to \mathbb{R}^{20}$,

Constructed "residual graph" $G'=(V, E')$

For each $u, v \in V$,

$$r(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{o.w.} \end{cases}$$

Then $r(u,v) + r(v,u) = \begin{cases} c(u,v) & \text{if } (u,v) \in E \\ c(v,u) & \text{if } (v,u) \in E \\ 0 & \text{o.w.} \end{cases}$

Let $E' := \{(u,v) : r(u,v) > 0\}$

① $\nexists$ s-t path: $A = \{v : v \text{ reachable from } s \text{ in } G'\}$ and $B = V \setminus A$ satisfies $\|f\| = c(A,B)$.

② $\exists$ s-t path $p = (s, v_1, \ldots, v_{k-1}, t)$ in $G' = (V, E')$

↖ "augmenting" path

"Update" f to f' using p

Also, let $F = \min_{(u,v) \in p} r(u,v) > 0$.

Consider a new flow $f' : E \to \mathbb{R}^{\geq 0}$ s.t. $\forall (u,v) \in E$

$$f'(u,v) = \begin{cases} f(u,v) + F & \text{if } (u,v) \in p \\ f(u,v) - F & \text{if } (v,u) \in p \\ f(u,v) & \text{o.w.} \end{cases}$$

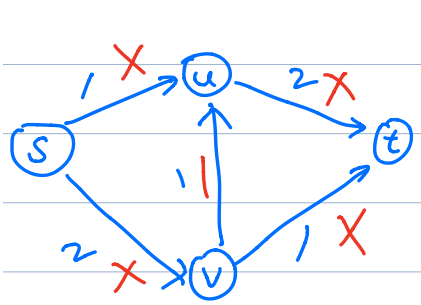
$\|f'\| = \|f\| + F$. □

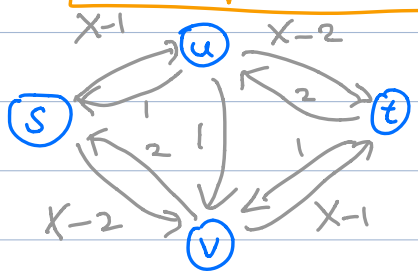
Ford-Fulkerson Algo: Start from $f = 0$, iterate the above proof! Runtime: $O(m \cdot OPT)$ where each capacity $c(e)$ is integer, and $OPT = \max_{\text{feasible } s\text{-}t \text{ flow } f} \|f\|$.

pseudo-poly-time algo:

n, m: upper bounds "number of input numbers"

OPT: depends on "magnitude of input numbers"





Edmonds-Karp

Edmonds-Karp

$f = 0$

While

  Compute r, $G' = (V, E')$

  If # s-t path, return f.

  Find "shortest s-t path" p in G'. ← smallest # edges

  Update f to f' using p.

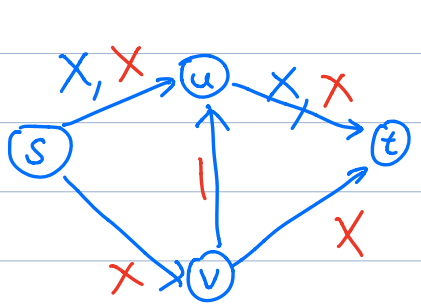
Will show (# while loops) $\leq O(mn)$

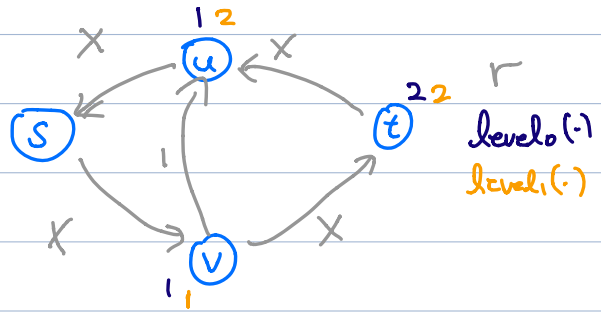
$\Rightarrow$ Running time $\leq O(nm^2)$.

Let f_i, r_i, G_i be f, r, G' at the start of i^{th} while loop.

(e.g., $f_0 = 0$, $r_0 = c$, $G_0 = G$)

For $v \in V$, $level_i(v)$ = (# edges) of shortest s-v path in G_i. (∞ if # s-v path.)





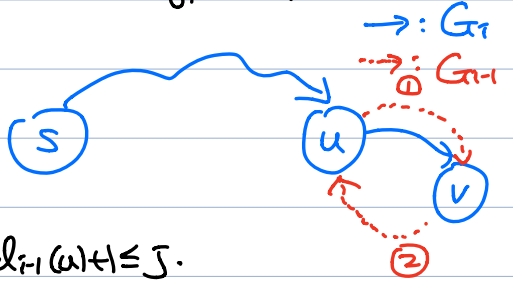
Lemma $\text{level}_i(v) \geq \text{level}_{i-1}(v)$ $\forall v \in V$ and $i \in \mathbb{N}$.

Pf. Induction on "$\text{level}_i(v)$" not i: want to prove

Claim: $\forall i \in \mathbb{N}$, $v \in V$, $j = \mathbb{N} \cup \{\infty\}$, if $j = \text{level}_{i-1}(v)$, then $j \leq \text{level}_i(v)$.

When $j = 0$, s is the only vertex s.t. $\text{level}_i(s) = 0$ for any i.

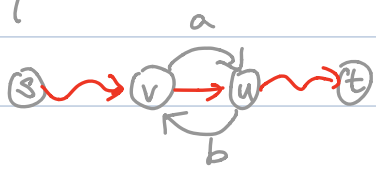
When Claim holds for $0, \dots, j-1$, consider v with $\text{level}_i(v) = j$. Let $(s, \dots, u, v)$ be a shortest s-v path. Then $\text{level}_i(u) = j-1$, so by induction hypothesis, $\text{level}_{i-1}(u) \leq \text{level}_i(u) = j-1$.



We are done if $\text{level}_{i-1}(v) \leq j$.

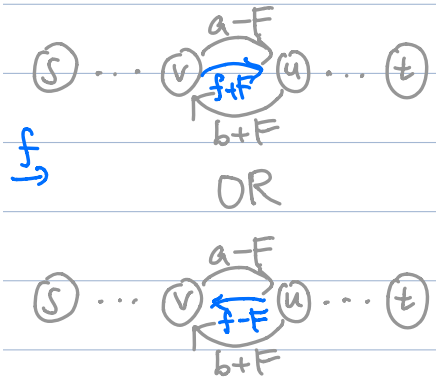
① If $(u,v) \in G_{i-1}$, then $\text{level}_{i-1}(v) \leq \text{level}_{i-1}(u) + 1 \leq j$.

② If $(u,v) \notin G_{i-1}$, since $(u,v) \in G_i$, by design of algo, (v,u) was in the shortest path p in the $(i-1)^{\text{th}}$ loop. So $\text{level}_{i-1}(v) = \text{level}_{i-1}(u) - 1 \leq j-2$ □

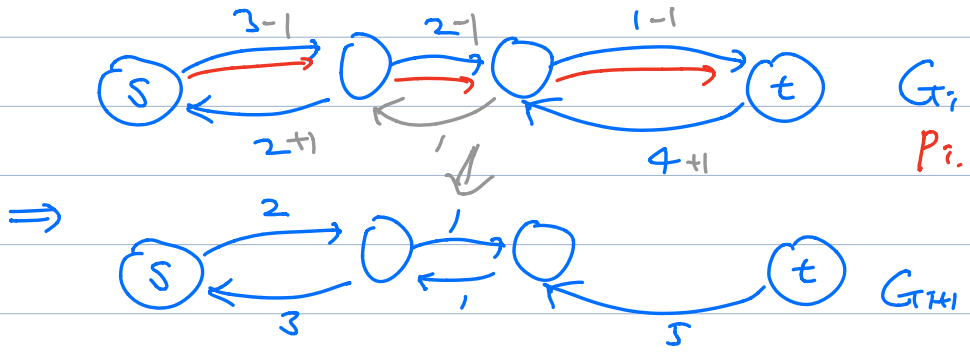


$$f'(u,v) = \begin{cases} f(u,v) + F & \text{if } (u,v) \in p \\ f(u,v) - F & \text{if } (v,u) \in p \\ f(u,v) & \text{o.w.} \end{cases} \quad (u,v) \in E \text{ or } (u,v) \in V^2$$

$$r(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{o.w.} \end{cases}$$



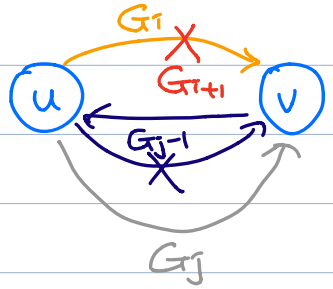
Note that $\forall i$, $\exists e \in G_i$ that "disappears" in G_{i+1}.



But edges may "reappear".

Lemma, $\forall (u,v) \in E$, it "disappears" at most $n/2$ times.

Proof, Suppose (u,v) disappeared in G_{i+1} and reappeared in G_j.



$\text{level}_i(u)+1 = \text{level}_i(v)$

$\wedge 1$

$\text{level}_{j-1}(u)-1 = \text{level}_{j-1}(v)$

$\therefore$ level (u) increases by ≥ 2 in each disappear/reappearance

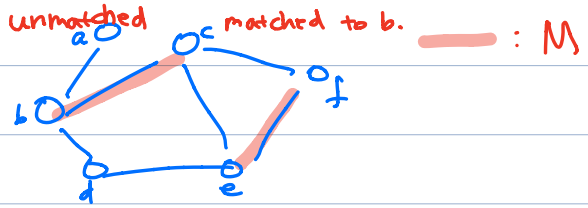
□

Since $\text{level}_i(v) \in [0, n-1] \cup \{\infty\}$,

(# while loops) $\leq nm/2$.

Bipartite Matching

Let $G=(V,E)$ be undirected graph. $M \subseteq E$ is called a **matching** if every $v \in V$ is incident on at most one edge in M.



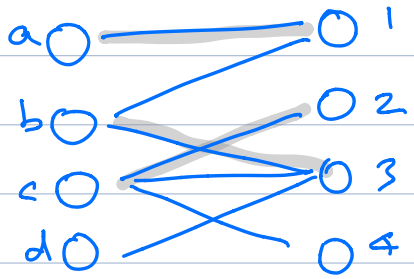
If v is incident on one $e=(u,v) \in M$, then v is "matched to u" (vice versa). Otherwise, v is "unmatched".

Maximum (Unweighted) Bipartite Matching

Input: Undirected bipartite graph $G=(A \cup B, E)$.

Output: Matching $M \subseteq E$ with largest $|M|$.

Applications
- Jobs/Applicants
- Donors/Recipients
- ⋮



Generalizations

- Weighted: $w: E \to \mathbb{R}$ and maximize $w(M)=\sum_{e \in M} w(e)$.
- General graph.
- Even weighted general graph matching can be solved in poly-time!

ALGO: Use "Reduction" from Matching to Max-Flow.

input for matching — input for Max Flow.

Given $G=(A \cup B, E)$, produce $G'=(A \cup B \cup \{s,t\}, E')$ (directed)

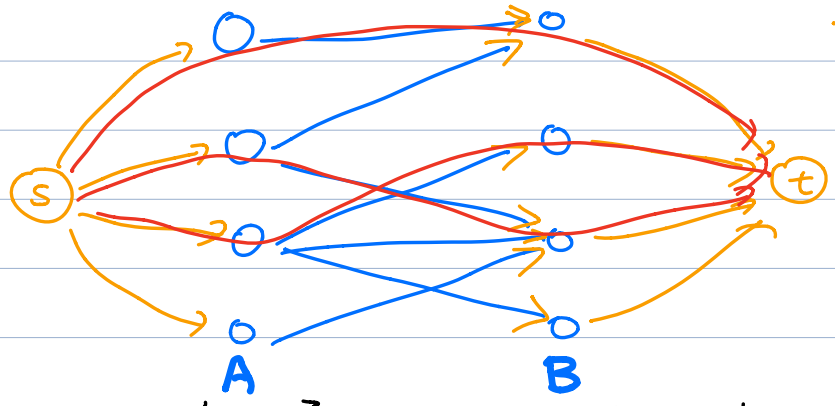
s.t. (1) $\forall a \in A$, add (s,a) to E'

(2) $\forall b \in B$, add (b,t) to E'

(3) $\forall (a,b) \in E$ $(a \in A, b \in B)$, add (a,b) to E'.

— : G

→ : G'



and let $c: E' \to \mathbb{R}^{\geq 0}$ s.t. $c(e)=1$ $\forall e \in E'$.

Then G', c, s, t form a Max Flow instance!

Claim: (Size of max matching in G) = (value of max flow in G')

Proof: 1-to-1 correspondence between matching M and "integral" feasible s-t flow $f: E' \to \{0,1\}$. But by previous lecture, since all capacities are 1, max flow value can be achieved by an integral flow. □

Running time: $O(|E'| \cdot OPT_{flow}) \leq O((|V|+|E|) \cdot |V|)$.