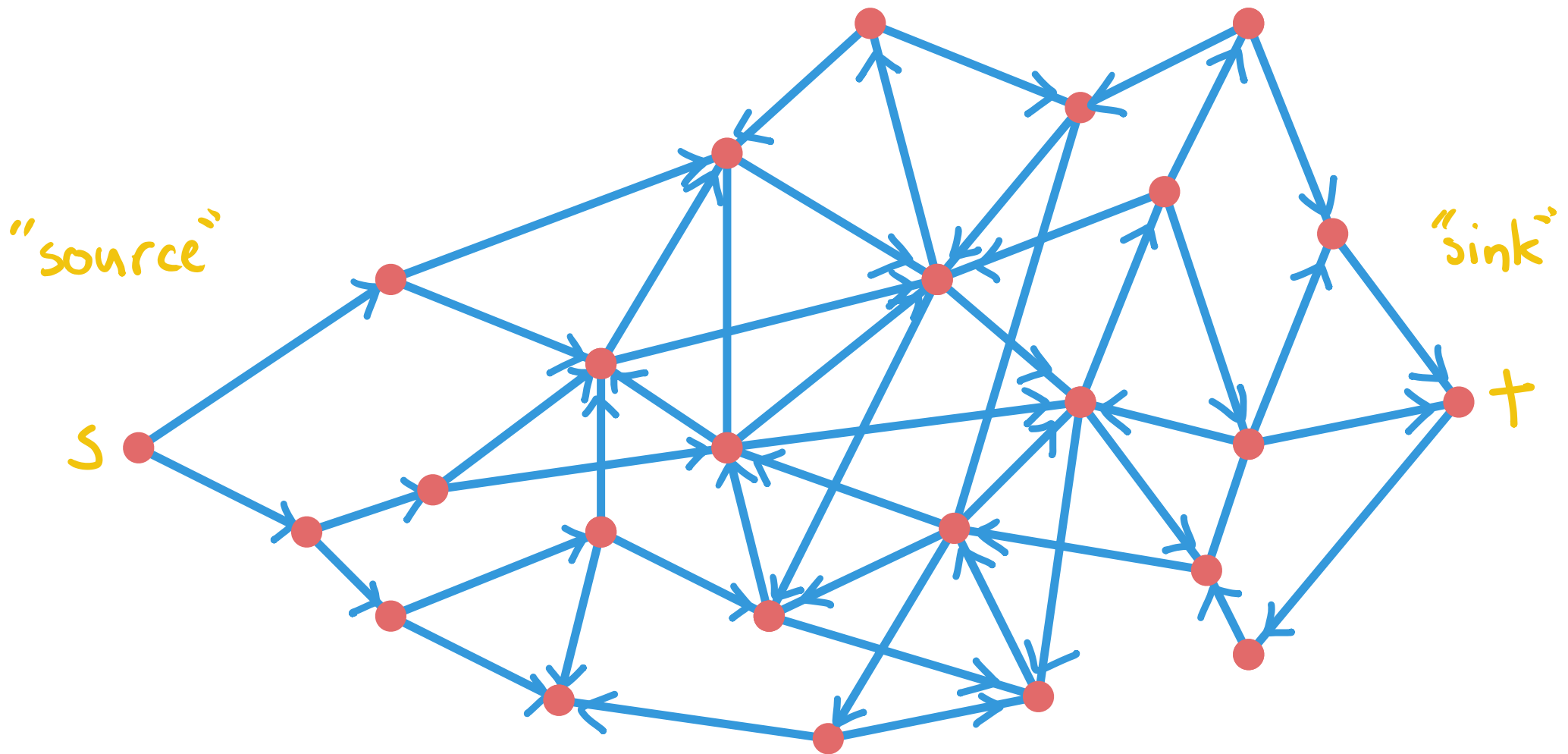
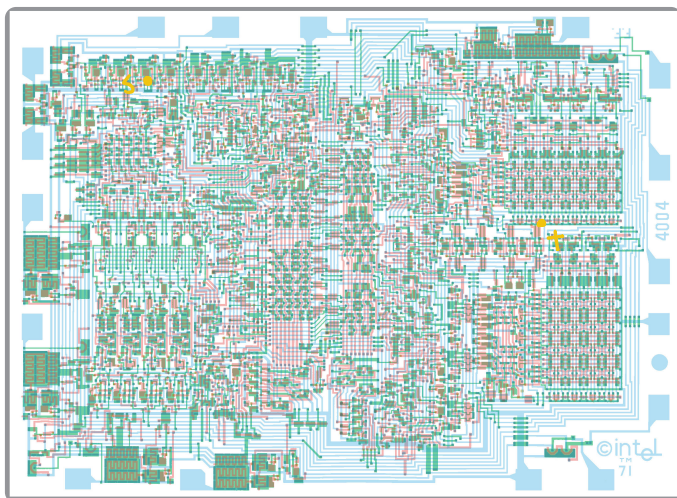
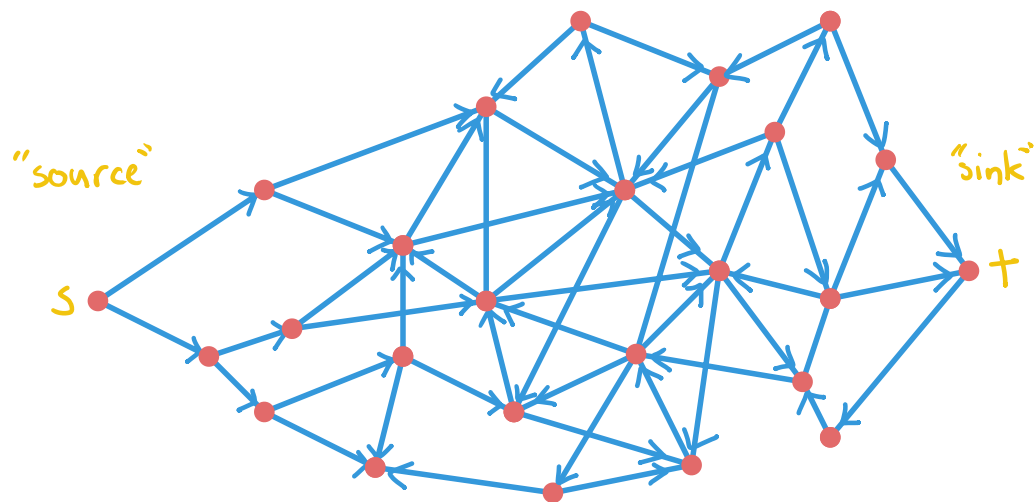


Network Flow

Measuring connectivity

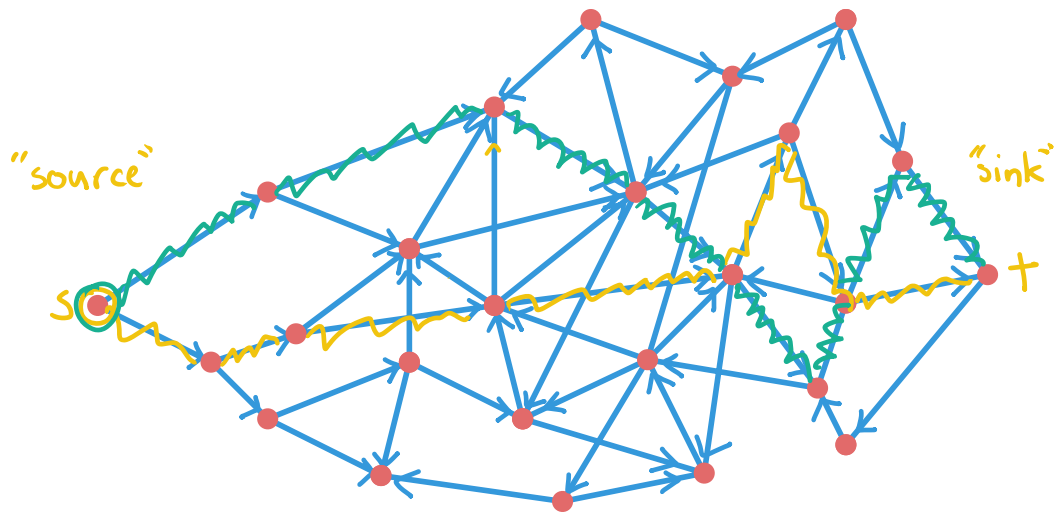


how well connected are s and t?

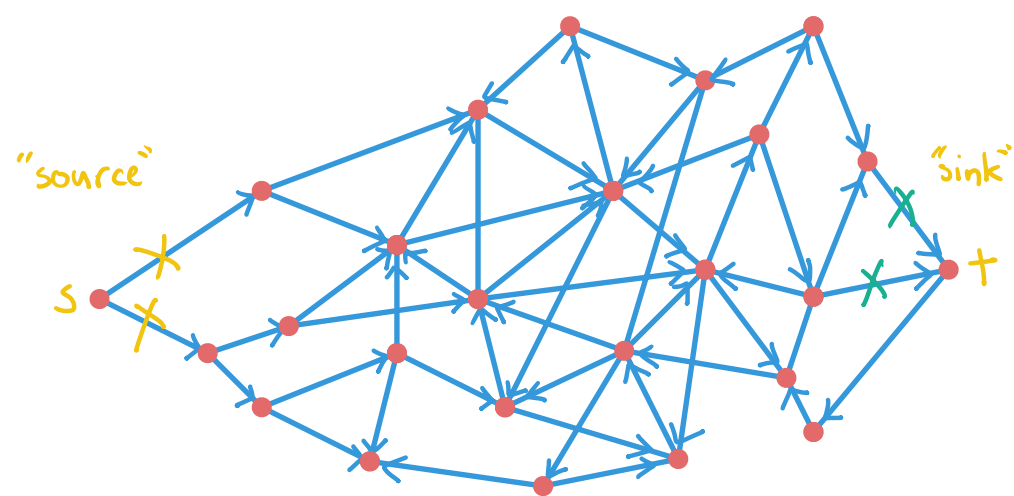


- ① Is there an $s \rightsquigarrow t$ path?
- ② How many distinct $s \rightsquigarrow t$ paths?
- ③ how many k -edge $s \rightsquigarrow t$ walks?
(k fixed)
- ④ length of shortest $s \rightsquigarrow t$ path?
- ⑤ how many shortest $s \rightsquigarrow t$ paths?
- ⑥ In a random walk starting from s , what is the expected # of steps until we reach t ?
- ⑦ max # of edge-disjoint $s \rightsquigarrow t$ paths?
- ⑧ min # of edge deletions to disconnect $s \rightsquigarrow t$?

Measuring connectivity

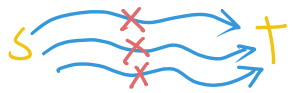


⑦ max # of edge-disjoint $s \rightarrow t$ paths?



⑧ min # of edge deletions to disconnect $s \rightarrow t$?

path $p \xrightarrow{\text{packs into}} \text{edge } e \in P$
 $\xleftarrow{\text{Covers}}$



path $p \xrightarrow{\text{packs into}} \text{edge } e \in P$
 $\xleftarrow{\text{Covers}}$

Lemma

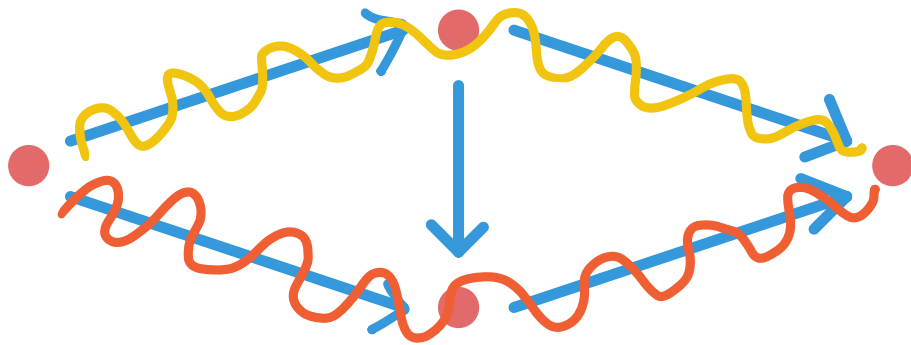
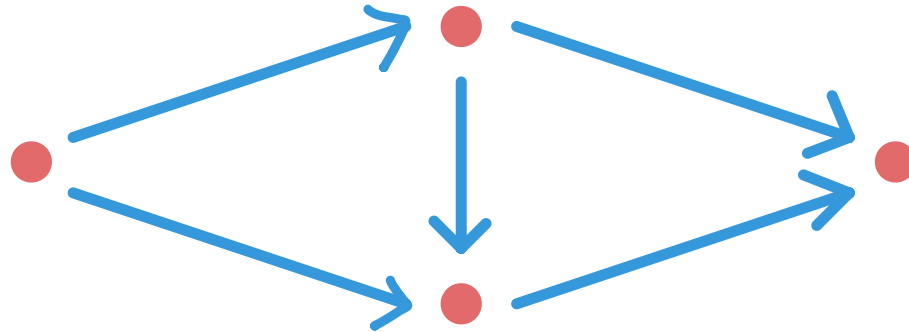
$$\left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightarrow t \text{ paths} \end{array} \right) \leq \left(\begin{array}{c} \text{min \# edges} \\ \text{in any } s \rightarrow t \text{ cut} \end{array} \right)$$

* 2nd example of
 packing $\leq$ covering

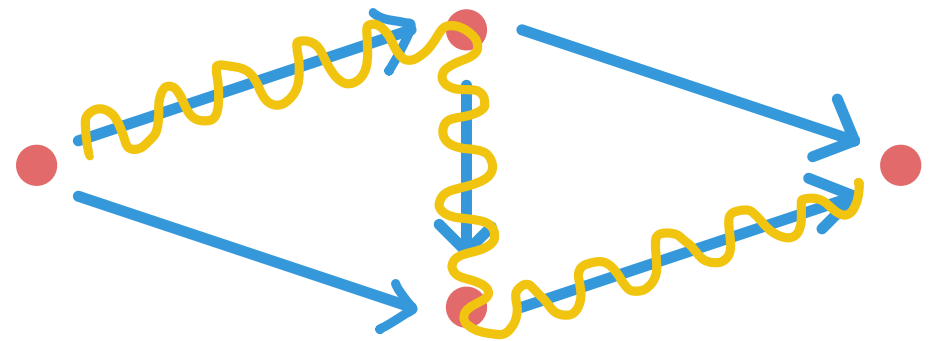
Iterative/greedy

① while there is a path $p: s \rightsquigarrow t$

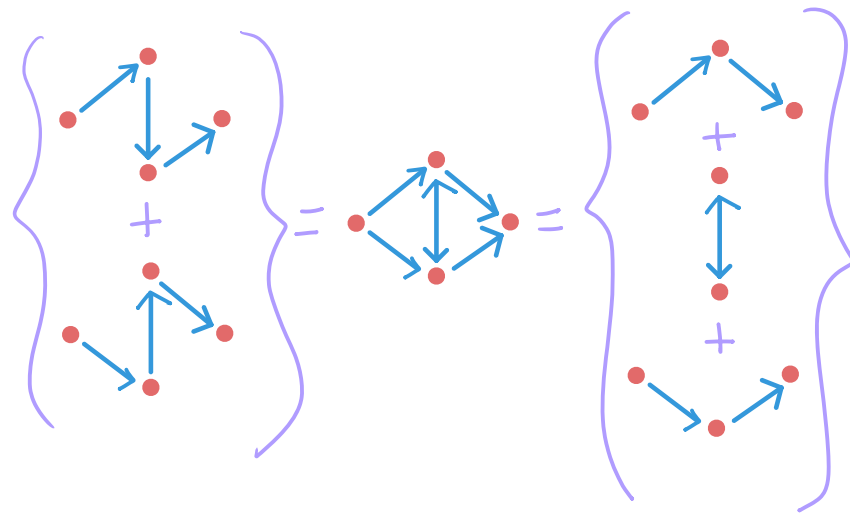
@ take p and remove all its edges from the graph



(good)



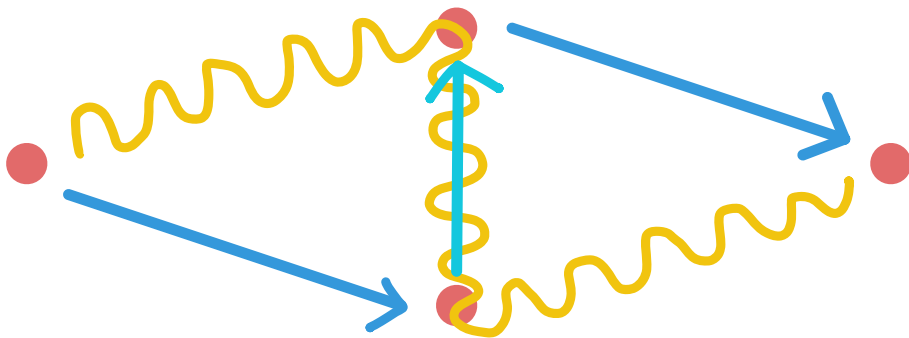
(bad)



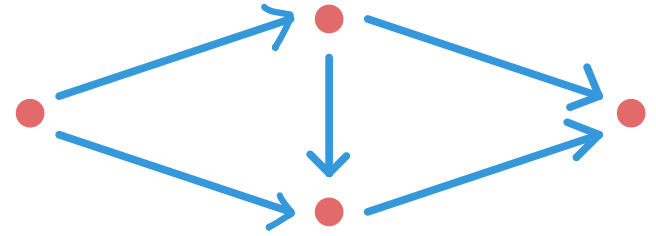
"opposite edges"



"cancel out"



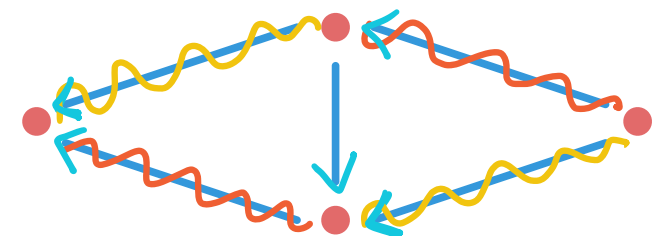
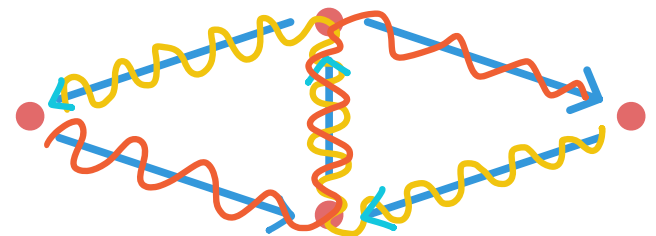
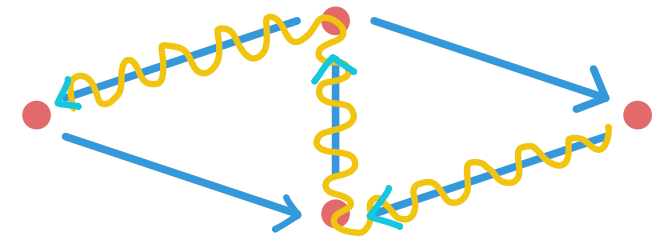
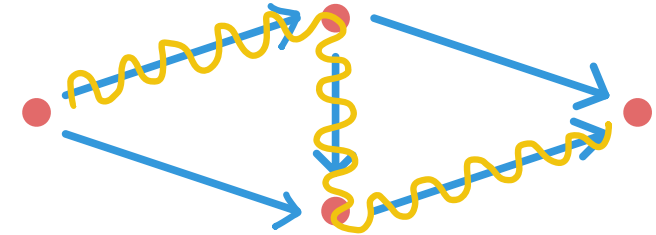
we should be allowed
to take reverse edges
in subsequent paths



augmenting-paths($G = (V, E), s, t$)

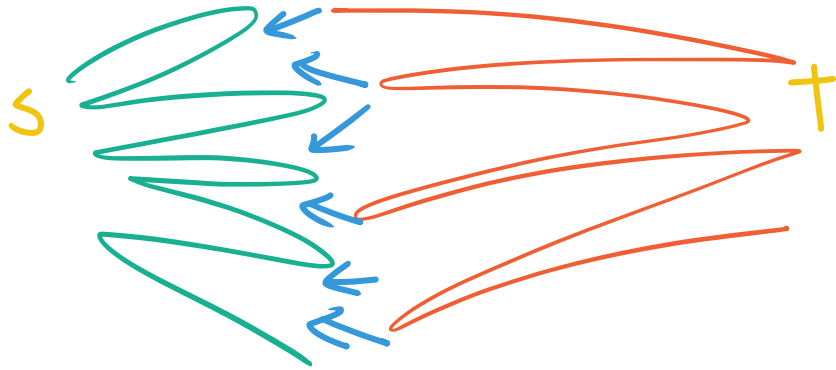
/ Ford and Fulkerson [FF56]*

1. $P \leftarrow \emptyset$
2. While there is an (s, t) -path $p : s \rightsquigarrow t$:
 - A. Remove all of p 's edges from the graph.
 - B. Add the reverse of all of p 's edges to the graph.
 - C. Uncross p with previously selected paths, and add p to P .



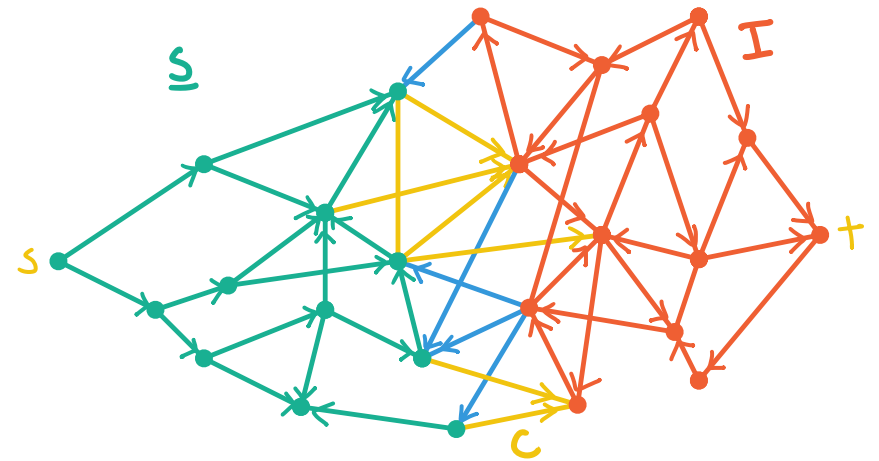
Proving optimality

algorithm terminates w/ $s \rightarrow t$ disconnected in residual graph



let S = vertices reachable from s in residual graph
 $I = V \setminus S$

$C = \delta^+(S) =$ edges from S to T



over the course of the algorithm:

- ① each $s \rightarrow t$ path has (exactly) 1 more edge from S to I than from I to S
- ② each $s \rightarrow t$ path decreases $|C|$ by 1
- ③ At the end, $|C| = 0$.

$$\Rightarrow (\text{\# paths packed}) = (\text{\# edges in } C)$$

$$(\text{\# paths packed}) = (\text{max \# disjoint } s \rightarrow t \text{ paths}) = (\text{min \# edges in any } s \rightarrow t \text{ cut}) = (\text{\# edges in } C)$$

equal

$$(\# \text{ paths packed}) \stackrel{=}{=} \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightarrow t \text{ paths} \end{array} \right) \stackrel{=}{=} \left(\begin{array}{c} \text{min \# edges} \\ \text{in any } s \rightarrow t \text{ cut} \end{array} \right) \stackrel{=}{=} \left(\begin{array}{c} \text{\# edges} \\ \text{in } C \end{array} \right)$$

Theorems

$$\textcircled{1} \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightarrow t \text{ paths} \end{array} \right) = \left(\begin{array}{c} \text{min \# edges} \\ \text{in any } s \rightarrow t \text{ cut} \end{array} \right)$$

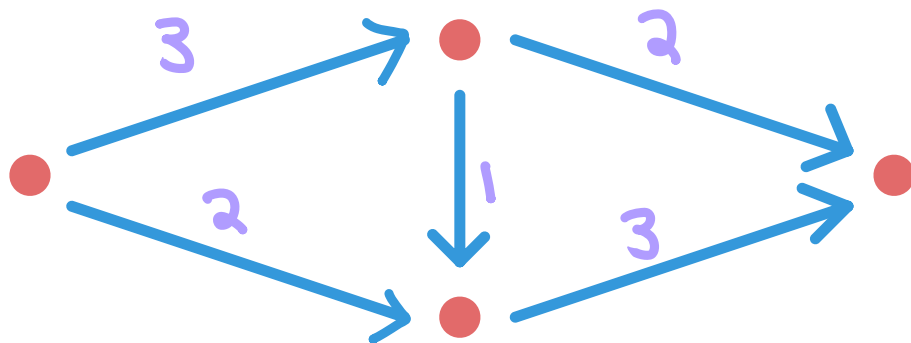
$$\textcircled{2} (\# \text{ paths packed}) = \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightarrow t \text{ paths} \end{array} \right)$$

$$\textcircled{3} \left(\begin{array}{c} \text{min \# edges} \\ \text{in any } s \rightarrow t \text{ cut} \end{array} \right) = \left(\begin{array}{c} \text{\# edges} \\ \text{in } C \end{array} \right)$$

1. $P \leftarrow \emptyset$
2. While there is an (s, t) -path $p: s \rightsquigarrow t$:
 - A. Remove all of p 's edges from the graph.
 - B. Add the reverse of all of p 's edges to the graph.
 - C. Uncross p with previously selected paths, and add p to P .³
3. $S \leftarrow \{\text{vertices still reachable from } s\}$.
4. Return the (s, t) -path packing P and the (s, t) -cut $\partial^+(S)$.

*/

Theorem 14.5 ([FF56]). Let $G = (V, E)$ be a directed graph with m edges, and let $s, t \in V$ be distinct vertices. Let λ be the size of the max-flow/min-cut from s to t . Then $\text{augmenting-paths}(G, s, t)$ produces both a maximum packing and a minimum cut in $O(m\lambda)$ time.



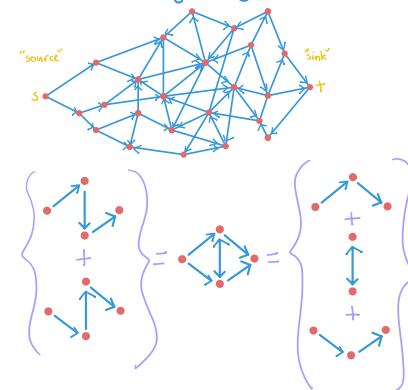
capacity $c(e) > 0$: reuse edge e up to $c(e)$ times

• if $c(e)$ large, Augmenting-Paths not polynomial time

• if $c(e)$ not integral (and just real-valued)
what about max fractional path packing

How to compute

⑦ max # of edge-disjoint $s \rightsquigarrow t$ paths?



Lemma let p, q be edge-disjoint $s \rightsquigarrow t$ paths



then there is $p', q': s \rightsquigarrow t$ disjoint without opposite pairs of edges



$$\textcircled{1} \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right) = \left(\begin{array}{c} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right)$$

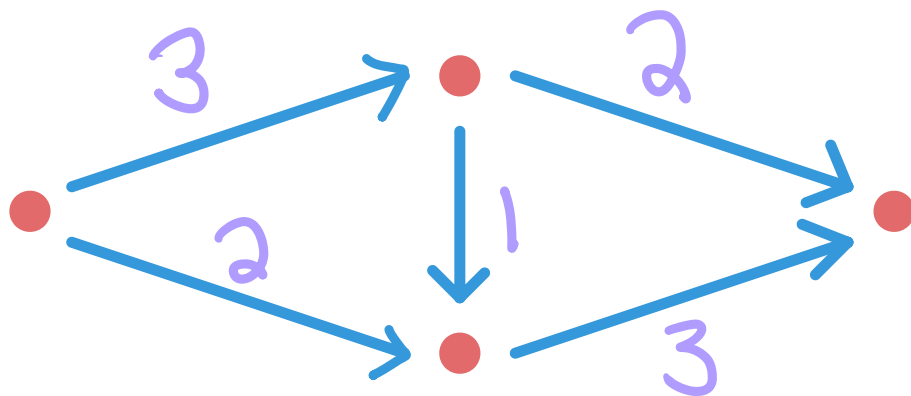
$$\textcircled{2} \left(\begin{array}{c} \# \text{ paths} \\ \text{packed} \end{array} \right) = \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right)$$

$$\textcircled{3} \left(\begin{array}{c} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right) = \left(\begin{array}{c} \# \text{ edges} \\ \text{initially in } C \end{array} \right)$$

1. $P \leftarrow \emptyset$
2. While there is an (s, t) -path $p: s \rightsquigarrow t$:
 - A. Remove all of p 's edges from the graph.
 - B. Add the reverse of all of p 's edges to the graph.
 - C. Uncross p with previously selected paths, and add p to P .³
3. $S \leftarrow \{\text{vertices still reachable from } s\}$.
4. Return the (s, t) -path packing P and the (s, t) -cut $\partial^+(S)$.

*/

Theorem 14.5 ([FF56]). Let $G = (V, E)$ be a directed graph with m edges, and let $s, t \in V$ be distinct vertices. Let λ be the size of the max-flow/min-cut from s to t . Then $\text{augmenting-paths}(G, s, t)$ produces both a maximum packing and a minimum cut in $O(m\lambda)$ time.



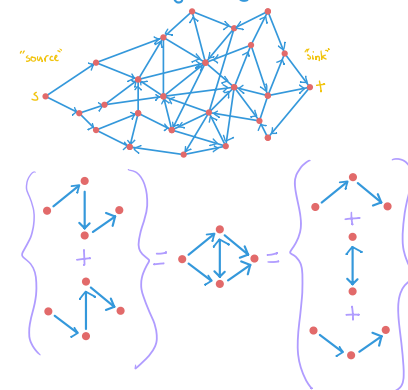
capacity $c(e) > 0$: reuse edge e up to $c(e)$ times

• if $c(e)$ large, Augmenting-Paths not polynomial time

• if $c(e)$ not integral (and just real-valued)
what about max fractional path packing

How to compute

⑦ max # of edge-disjoint $s \rightsquigarrow t$ paths?



Lemma let p, q be edge-disjoint $s \rightsquigarrow t$ paths



then there is p', q' : $s \rightsquigarrow t$ disjoint without opposite pairs of edges



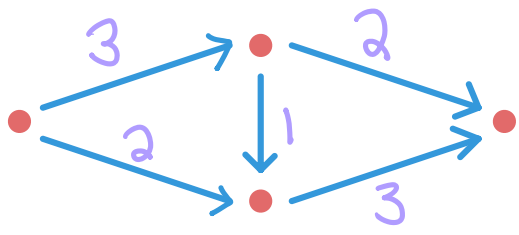
$$\textcircled{1} \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right) = \left(\begin{array}{c} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right)$$

$$\textcircled{2} \left(\begin{array}{c} \# \text{ paths} \\ \text{packed} \end{array} \right) = \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right)$$

$$\textcircled{3} \left(\begin{array}{c} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right) = \left(\begin{array}{c} \# \text{ edges} \\ \text{initially in } C \end{array} \right)$$

- 1. $P \leftarrow \emptyset$
- 2. While there is an (s, t) -path $p: s \rightsquigarrow t$:
 - A. Remove all of p 's edges from the graph.
 - B. Add the reverse of all of p 's edges to the graph.
 - C. Uncross p with previously selected paths, and add p to P .³
- 3. $S \leftarrow \{\text{vertices still reachable from } s\}$.
- 4. Return the (s, t) -path packing P and the (s, t) -cut $\partial^+(S)$.

Theorem 14.5 ([FF56]). Let $G = (V, E)$ be a directed graph with m edges, and let $s, t \in V$ be distinct vertices. Let λ be the size of the max-flow/min-cut from s to t . Then $\text{augmenting-paths}(G, s, t)$ produces both a maximum packing and a minimum cut in $O(m\lambda)$ time.



capacity $c(e) > 0$:
reuse edge e up to $c(e)$ times

(Weighted) path packings
 $P_{st} = \{\text{family of all } (s, t)\text{-paths}\}$

objective $\rightarrow \max \sum_{p \in P_{st}} x_p$

constraint $\rightarrow \text{s.t. } \sum_{\substack{p \in P_{st}: \\ e \in p}} x_p \leq c_e$

over $x: P_{st} \rightarrow \mathbb{R}_{\geq 0}$

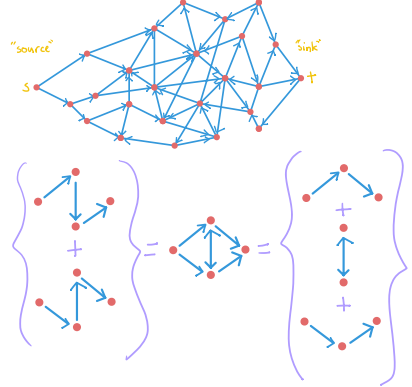
for all $e \in E$

$c_e = 3$

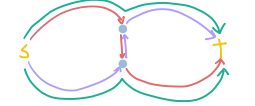
linear program

How to compute

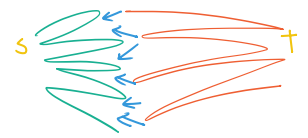
⑦ max # of edge-disjoint $s \rightsquigarrow t$ paths?



Lemma let p, q be edge-disjoint $s \rightsquigarrow t$ paths



then there is $p', q': s \rightsquigarrow t$ disjoint without opposite pairs of edges

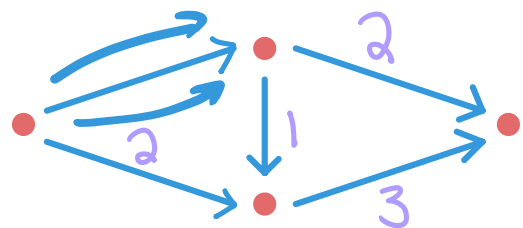


- ① $\left(\begin{smallmatrix} \text{max \#} \\ \text{disjoint } s \rightsquigarrow t \text{ paths} \end{smallmatrix} \right) = \left(\begin{smallmatrix} \text{min \#} \\ \text{edges in any } s \rightsquigarrow t \text{ cut} \end{smallmatrix} \right)$
- ② $\left(\begin{smallmatrix} \# \text{ paths} \\ \text{packed} \end{smallmatrix} \right) = \left(\begin{smallmatrix} \text{max \#} \\ \text{disjoint } s \rightsquigarrow t \text{ paths} \end{smallmatrix} \right)$
- ③ $\left(\begin{smallmatrix} \text{min \#} \\ \text{edges in any } s \rightsquigarrow t \text{ cut} \end{smallmatrix} \right) = \left(\begin{smallmatrix} \# \text{ edges} \\ \text{initially in } C \end{smallmatrix} \right)$

1. $P \leftarrow \emptyset$
2. While there is an (s, t) -path $p: s \rightsquigarrow t$:
 - A. Remove all of p 's edges from the graph.
 - B. Add the reverse of all of p 's edges to the graph.
 - C. Uncross p with previously selected paths, and add p to P .³
3. $S \leftarrow \{\text{vertices still reachable from } s\}$.
4. Return the (s, t) -path packing P and the (s, t) -cut $\partial^+(S)$.

*/

Theorem 14.5 ([FF56]). Let $G = (V, E)$ be a directed graph with m edges, and let $s, t \in V$ be distinct vertices. Let λ be the size of the max-flow/min-cut from s to t . Then $\text{augmenting-paths}(G, s, t)$ produces both a maximum packing and a minimum cut in $O(m\lambda)$ time.



capacity $c(e) > 0$:
reuse edge e up to $c(e)$ times

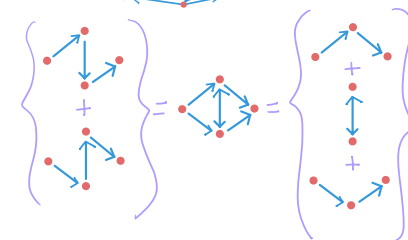
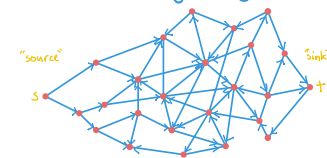
Integer capacities

duplicated edges to
reduce to edge-disjoint

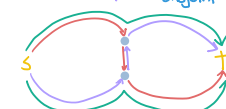
$O(m\lambda)$

How to compute

⑦ max # of edge-disjoint $s \rightsquigarrow t$ paths?



Lemma let p, q be edge-disjoint $s \rightsquigarrow t$ paths



then there is $p', q': s \rightsquigarrow t$ disjoint
without opposite pairs of edges



$$\textcircled{1} \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right) = \left(\begin{array}{c} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right)$$

$$\textcircled{2} \left(\begin{array}{c} \# \text{ paths} \\ \text{packed} \end{array} \right) = \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right)$$

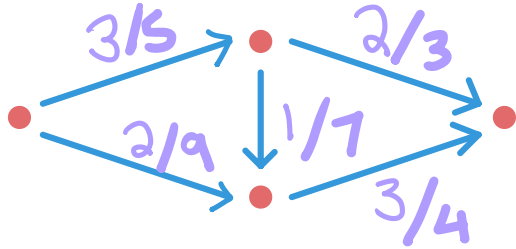
$$\textcircled{3} \left(\begin{array}{c} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right) = \left(\begin{array}{c} \# \text{ edges} \\ \text{initially in } C \end{array} \right)$$

1. $P \leftarrow \emptyset$
2. While there is an (s, t) -path $p: s \rightsquigarrow t$:
 - A. Remove all of p 's edges from the graph.
 - B. Add the reverse of all of p 's edges to the graph.
 - C. Uncross p with previously selected paths, and add p to P .³
3. $S \leftarrow \{\text{vertices still reachable from } s\}$.
4. Return the (s, t) -path packing P and the (s, t) -cut $\partial^+(S)$.

*/

Theorem 14.5 ([FF56]). Let $G = (V, E)$ be a directed graph with m edges, and let $s, t \in V$ be distinct vertices. Let λ be the size of the max-flow/min-cut from s to t . Then $\text{augmenting-paths}(G, s, t)$ produces both a maximum packing and a minimum cut in $O(m\lambda)$ time.

x1260



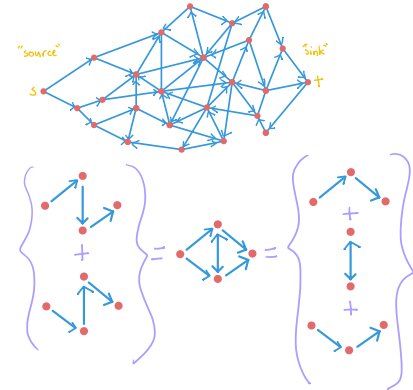
capacity $c(e) > 0$:
reuse edge e up to $c(e)$ times

Rational capacities $c(e) = 5/3$

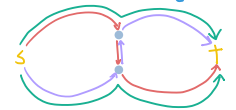
clear denominators $\rightarrow$ integer capacitated

How to compute

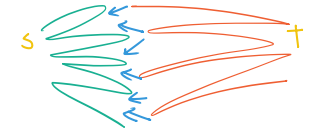
⑦ max # of edge-disjoint $s \rightsquigarrow t$ paths?



Lemma let p, q be edge-disjoint $s \rightsquigarrow t$ paths



then there is p', q' : $s \rightsquigarrow t$ disjoint without opposite pairs of edges



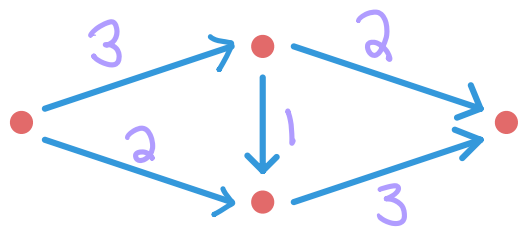
$$\textcircled{1} \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right) = \left(\begin{array}{c} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right)$$

$$\textcircled{2} \left(\begin{array}{c} \# \text{ paths} \\ \text{packed} \end{array} \right) = \left(\begin{array}{c} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right)$$

$$\textcircled{3} \left(\begin{array}{c} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right) = \left(\begin{array}{c} \# \text{ edges} \\ \text{initially in } C \end{array} \right)$$

1. $P \leftarrow \emptyset$
2. While there is an (s, t) -path $p: s \rightsquigarrow t$:
 - A. Remove all of p 's edges from the graph.
 - B. Add the reverse of all of p 's edges to the graph.
 - C. Uncross p with previously selected paths, and add p to P .³
3. $S \leftarrow \{ \text{vertices still reachable from } s \}$.
4. Return the (s, t) -path packing P and the (s, t) -cut $\partial^+(S)$.

Theorem 14.5 ([FF56]). Let $G = (V, E)$ be a directed graph with m edges, and let $s, t \in V$ be distinct vertices. Let λ be the size of the max-flow/min-cut from s to t . Then $\text{augmenting-paths}(G, s, t)$ produces both a maximum packing and a minimum cut in $O(m\lambda)$ time.



capacity $c(e) > 0$:
reuse edge e up to $c(e)$ times

Real capacities fix $\epsilon > 0$.

define $\tilde{c}(e)$ rational s.t.

$$|\tilde{c}(e) - c(e)| \leq \epsilon \quad \forall e \in E$$

$$\left| \left(\text{max paths} \right)_{\text{w/r/t } \tilde{c}} - \left(\text{max paths} \right)_{\text{w/r/t } c} \right| \leq m\epsilon$$

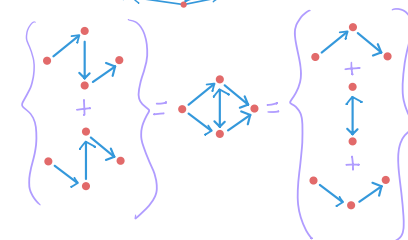
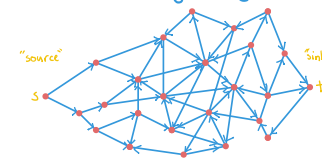
$$\left| \left(\text{min cut} \right)_{\text{w/r/t } \tilde{c}} - \left(\text{min cut} \right)_{\text{w/r/t } c} \right| \leq m\epsilon$$

equal

$$\left| \left(\text{max path} \right)_{\text{w/r/t } \tilde{c}} - \left(\text{min cut} \right)_{\text{w/r/t } \tilde{c}} \right| \leq \left| \left(\text{max path} \right)_{\text{w/r/t } \tilde{c}} - \left(\text{max path} \right)_{\text{w/r/t } c} \right| + \left| \left(\text{min cut} \right)_{\text{w/r/t } \tilde{c}} - \left(\text{min cut} \right)_{\text{w/r/t } c} \right| \leq 2m\epsilon$$

How to compute

⑦ max # of edge-disjoint $s \rightsquigarrow t$ paths?



Lemma let p, q be edge-disjoint $s \rightsquigarrow t$ paths



then there is $p', q': s \rightsquigarrow t$ disjoint without opposite pairs of edges



$$\textcircled{1} \quad \left(\text{max \# disjoint } s \rightsquigarrow t \text{ paths} \right) = \left(\text{min \# edges in any } s \rightsquigarrow t \text{ cut} \right)$$

$$\textcircled{2} \quad \left(\text{\# paths packed} \right) = \left(\text{max \# disjoint } s \rightsquigarrow t \text{ paths} \right)$$

$$\textcircled{3} \quad \left(\text{min \# edges in any } s \rightsquigarrow t \text{ cut} \right) = \left(\text{\# edges initially in } C \right)$$

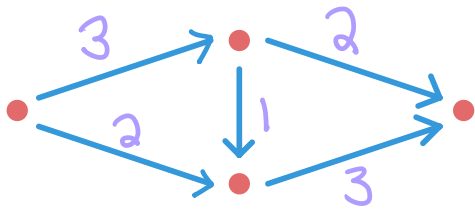
augmenting-paths($G = (V, E), s, t$)

/* Ford and Fulkerson [FF56]

1. $P \leftarrow \emptyset$
2. While there is an (s, t) -path $p : s \rightsquigarrow t$:
 - A. Remove all of p 's edges from the graph.
 - B. Add the reverse of all of p 's edges to the graph.
 - C. Uncross p with previously selected paths, and add p to P .³
3. $S \leftarrow \{\text{vertices still reachable from } s\}$.
4. Return the (s, t) -path packing P and the (s, t) -cut $\partial^+(S)$.

*/

Theorem 14.5 ([FF56]). Let $G = (V, E)$ be a directed graph with m edges, and let $s, t \in V$ be distinct vertices. Let λ be the size of the max-flow/min-cut from s to t . Then $\text{augmenting-paths}(G, s, t)$ produces both a maximum packing and a minimum cut in $O(m\lambda)$ time.



capacity $c(e) > 0$: reuse edge e
up to $c(e)$ times

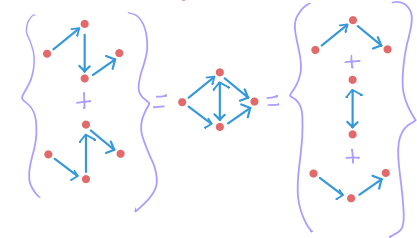
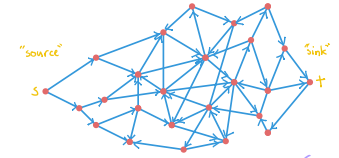
- if $c(e)$ large, Augmenting-Paths not polynomial time
- if $c(e)$ not integral (and just real-valued) what about max fractional path packing

Good news:

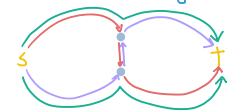
our duality theorems still hold

How to compute

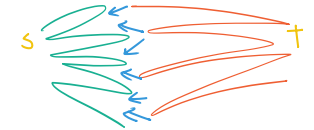
⑦ max # of edge-disjoint $s \rightsquigarrow t$ paths?



Lemma let p, q be edge-disjoint $s \rightsquigarrow t$ paths



then there is $p', q' : s \rightsquigarrow t$ disjoint
without opposite pairs of edges



$$\textcircled{1} \left(\begin{array}{l} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right) = \left(\begin{array}{l} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right)$$

$$\textcircled{2} \left(\begin{array}{l} \text{\# paths} \\ \text{packed} \end{array} \right) = \left(\begin{array}{l} \text{max \#} \\ \text{disjoint} \\ s \rightsquigarrow t \text{ paths} \end{array} \right)$$

$$\textcircled{3} \left(\begin{array}{l} \text{min \#} \\ \text{edges in any} \\ s \rightsquigarrow t \text{ cut} \end{array} \right) = \left(\begin{array}{l} \text{\# edges} \\ \text{initially in } C \end{array} \right)$$

remaining issues algorithmic

Max Flow "liquid" version of path packing

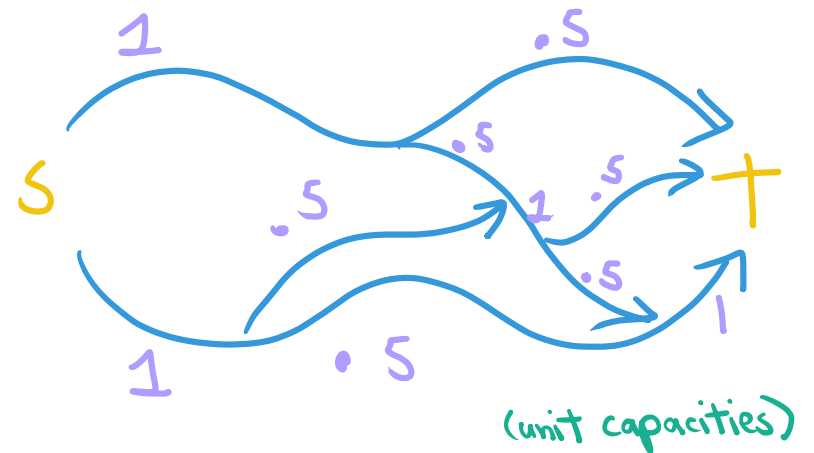
Input $G=(V,E)$ directed capacities $c:E \rightarrow \mathbb{R}_{>0}$

source $s \in V$, sink $t \in V$

"s-t flow" $f:E \rightarrow \mathbb{R}_{\geq 0}$ s.t.

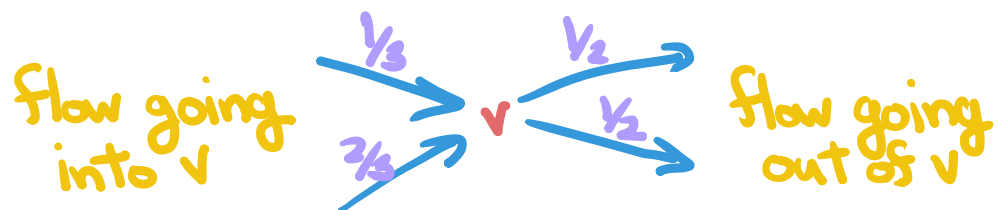
(a) "capacity constraints"

$$0 \leq f(e) \leq c(e) \quad \forall e \in E$$



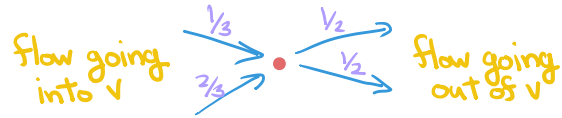
(b) "conservation of flow"

$$\sum_{(u,v) \in E} f(u,v) = \sum_{(v,w) \in E} f(v,w) \quad \forall v \notin \{s,t\}$$

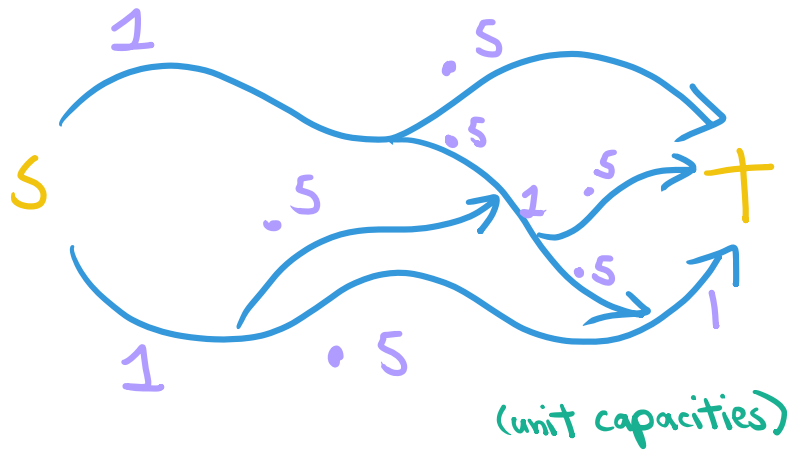


(6) Conservation of flow ~

$$\sum_{(u,v) \in E} f(u,v) = \sum_{(v,w) \in E} f(v,w) \quad \forall v \notin \{s,t\}$$



net flow out of s = net flow into t



"net flow from s to t",
"size of flow"

Max flow problem

find $s \rightarrow t$ flow of maximum size

"linear program"

size of flow

$$\max \sum_{(s,w) \in E} f(s,w) - \sum_{(u,s) \in E} f(u,s) \quad \text{over } f \in \mathbb{R}^E$$

"capacity constraints"

$$\text{s.t. } 0 \leq f(e) \leq c(e)$$

$$\forall e \in E$$

"conservation of flow"

$$\sum_{(u,v) \in E} f(u,v) = \sum_{(v,w) \in E} f(v,w)$$

$$\forall v \notin \{s, t\}$$

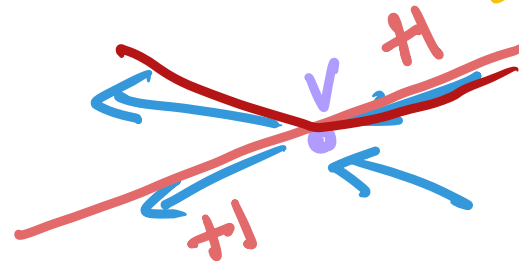
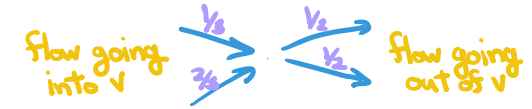
"s-t flow" $f: E \rightarrow \mathbb{R}_{\geq 0}$ s.t.

(a) "capacity constraints"

$$0 \leq f(e) \leq c(e) \quad \forall e \in E$$

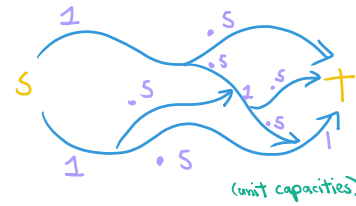
(b) "conservation of flow"

$$\sum_{(u,v) \in E} f(u,v) = \sum_{(v,w) \in E} f(v,w) \quad \forall v \notin \{s, t\}$$



"linear program" because objective, constraints, all linear functions of f

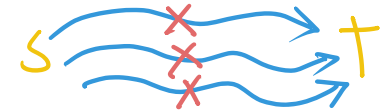
How does Max Flow

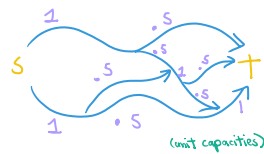


relate to

max path packings,

min cut?

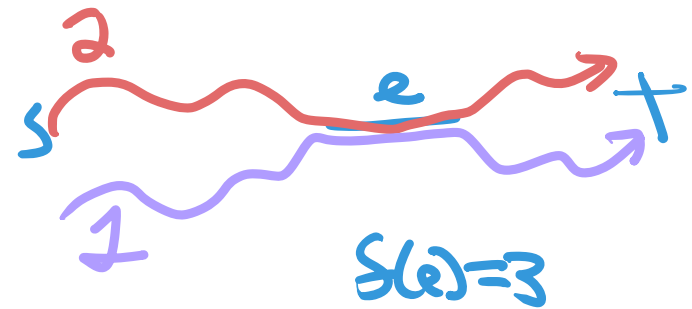



 $\leq$

 max path packings vs Max Flow

let $x: P_{s,t} \rightarrow \mathbb{R}_{\geq 0}$ encode a path packing

define $f: E \rightarrow \mathbb{R}_{\geq 0}$ as:

$$f(e) = \sum_{p: e \in p} x_p.$$



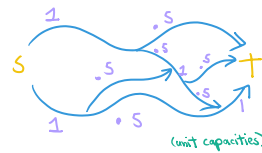
then f is feasible flow of same size

(PSO discusses converse)



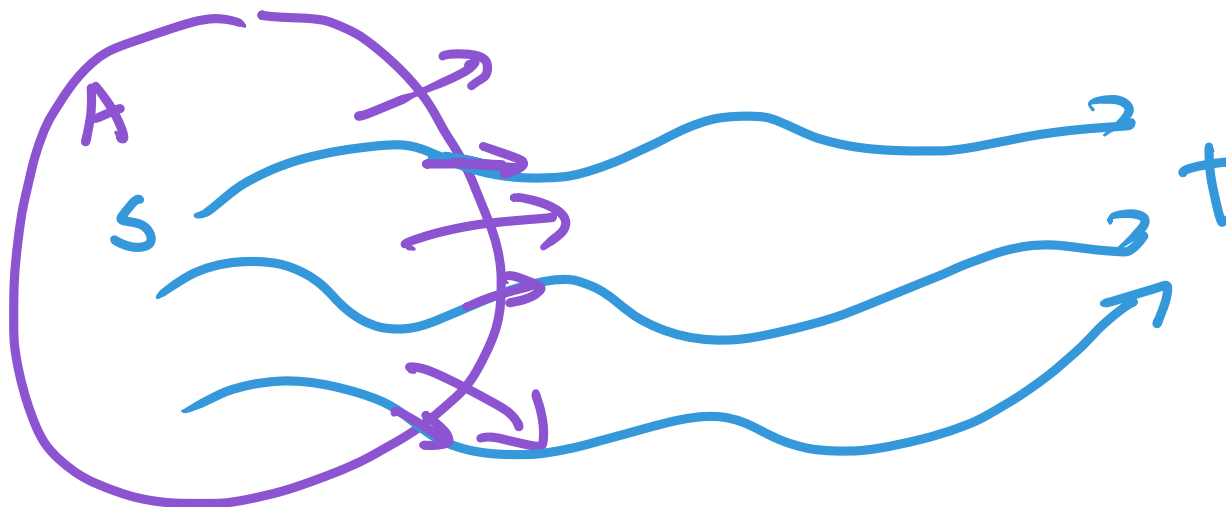
min cut

vs



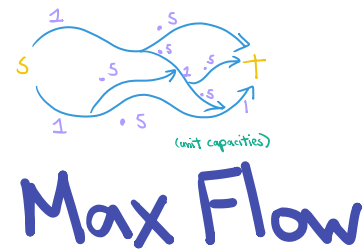
Max Flow

let $C = 2^+(A)$ min-cut, $f: E \rightarrow \mathbb{R}_{\geq 0}$ max flow
 (ACV w/ $s \in A$, $t \notin A$)

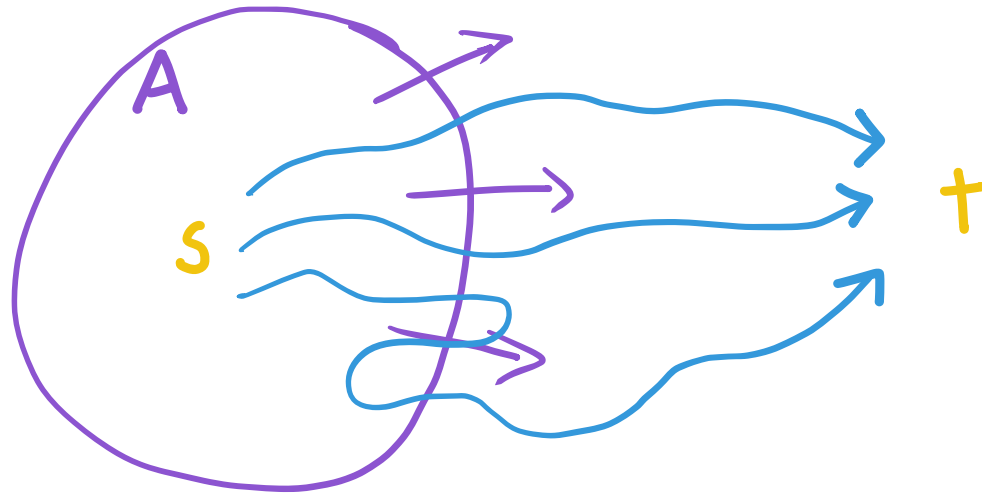




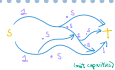
$\geq$
vs



let $C = 2^+(A)$ min-cut, $f: E \rightarrow \mathbb{R}_{\geq 0}$ max flow
 (ACV w/ $s \in A, t \notin A$)



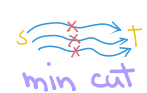
$$(\text{size of } f) = (\text{net flow out of } A) \leq (\text{capacity of } 2^+(A))$$



define $f: E \rightarrow \mathbb{R}_{\geq 0}$ as:

$$f(e) = \sum_{p: e \in p} x_p.$$

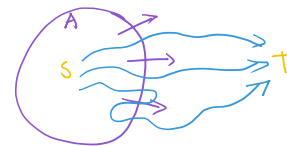
then f is feasible flow of same size



vs

Max Flow

let $C = \mathcal{J}^+(A)$ min-cut, $f: E \rightarrow \mathbb{R}_{\geq 0}$ max flow
(ACV w/ $seA \nmid \#A$)

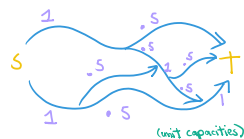


$$(\text{size of } f) = (\text{net flow out of } A) \leq (\text{capacity of } J^+(A))$$

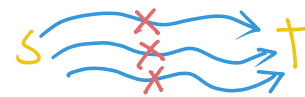
we showed



max path packing



Max Flow



min cut

But

equal

max path packings vs Max Flow

let $x: P_{s \rightarrow t} \rightarrow \mathbb{R}_{\geq 0}$ encode a path packing

define $f: E \rightarrow \mathbb{R}_{\geq 0}$ as:

$$f(e) = \sum_{p \ni e} x_p$$

then f is feasible flow of same size

min cut vs Max Flow

let $C = \mathcal{J}^c(A)$ min-cut, $f: E \rightarrow \mathbb{R}_{\geq 0}$ max flow

(MFC of unit flow)

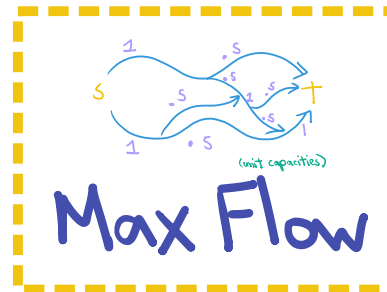


(size of f) = (net flow out of A) $\leq$ (capacity of $\mathcal{J}^c(A)$)

Theorem



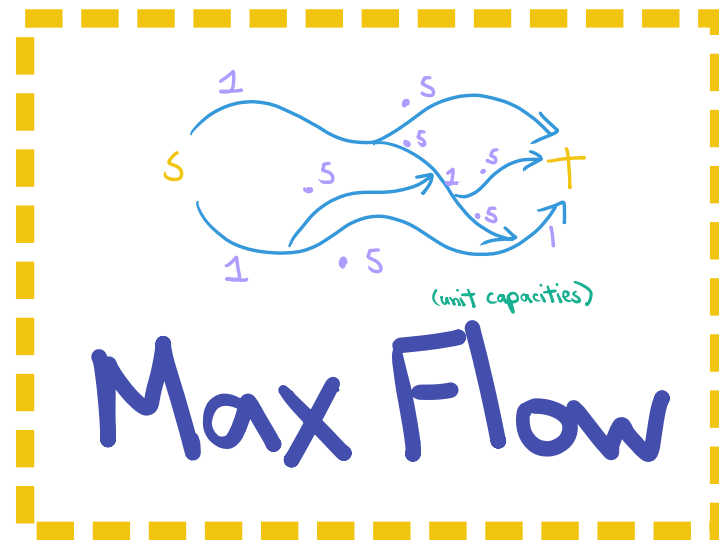
=



=



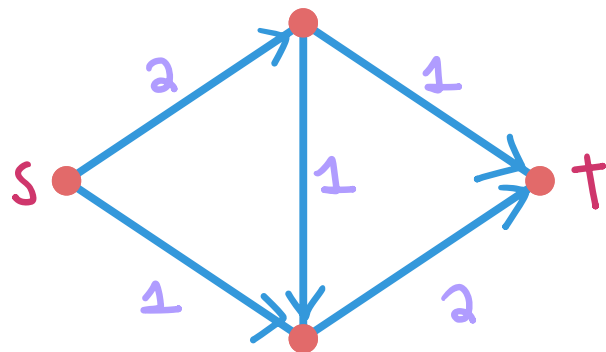
Next: strongly polynomial times



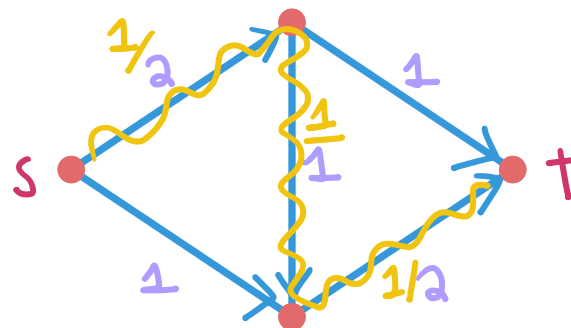
Residual graphs w/r/t flows

(exactly what you would expect)

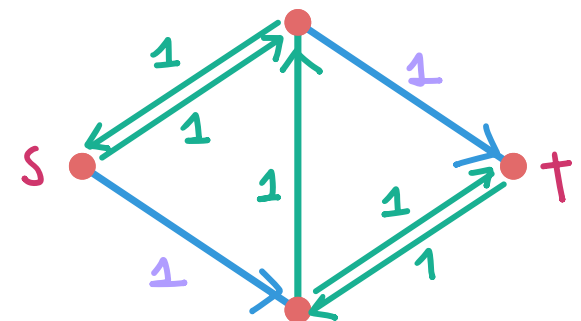
(capacities)



(1 unit flow)

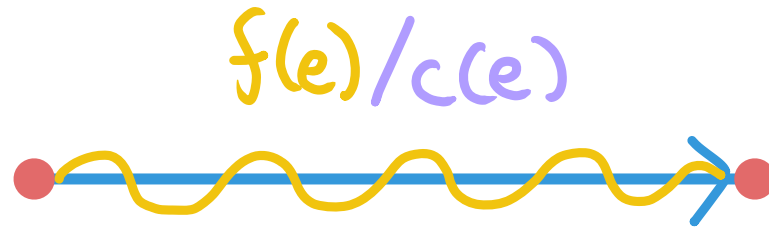


(residual edges)



let $f: E \rightarrow \mathbb{R}_{\geq 0}$ be (s,t) -flow

$e \in E$ w/ capacity $c(e)$

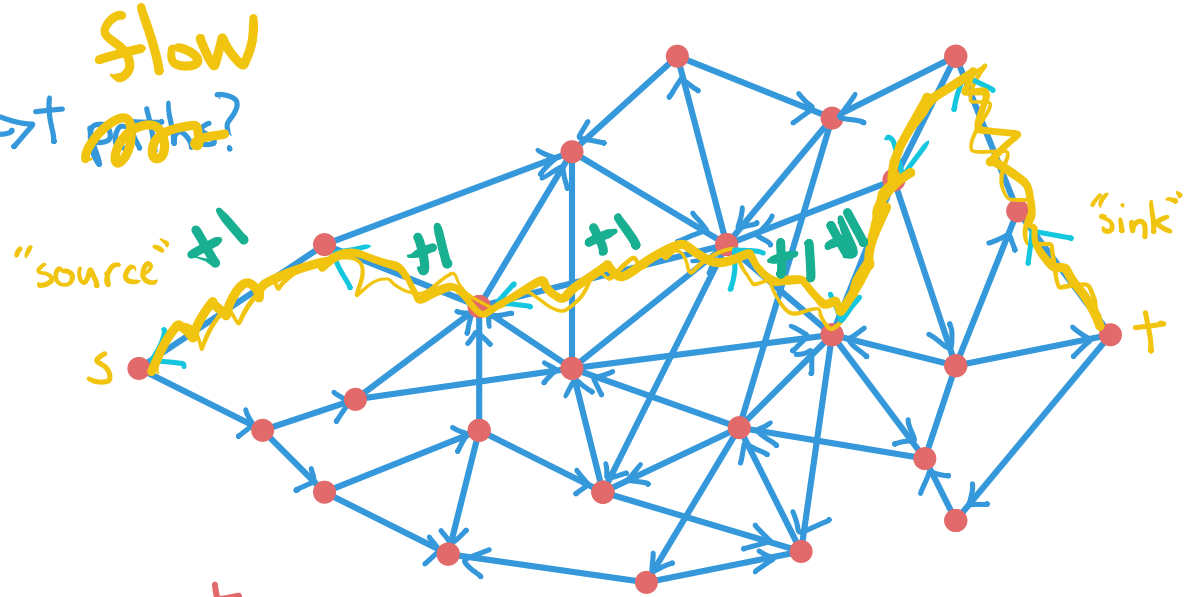


this leaves $f(e) - c(e)$ capacity in forwards direction
and $f(e)$ capacity backwards



How to compute

⑦ max # of ~~edge-disjoint~~ $s \rightsquigarrow t$ ~~paths?~~ ^{flow}



① while there is a path $p: s \rightsquigarrow t$

① ~~take p and remove all its edges from the graph~~

route as much flow as possible along p

② ~~add the reverse of all of p 's edges in the graph~~

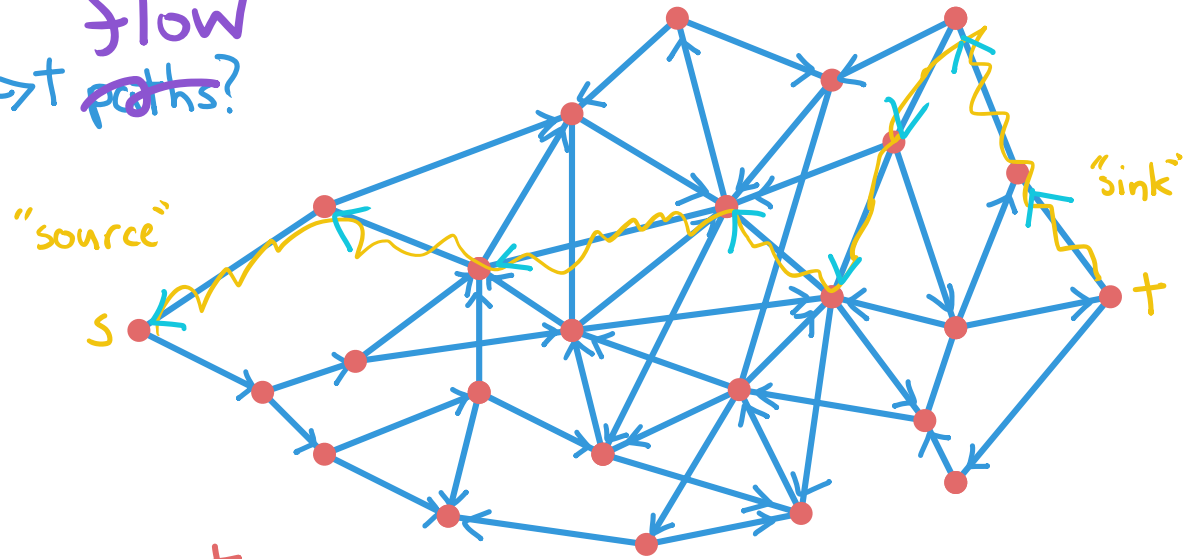
③ ~~uncross p w/ previously selected paths~~

update residual graph

↪ reasonable, but not quite polynomial time

How to compute

⑦ max # of ~~edge-disjoint~~ $s \rightsquigarrow t$ ~~paths?~~ ^{flow}



① while there is a path $p: s \rightsquigarrow t$

Ⓐ take p and ~~remove all its edges from the graph~~

route as much flow as possible

Ⓑ add the reverse of all of p 's edges in the graph

Ⓒ uncross p w/ previously selected paths

update residual graph

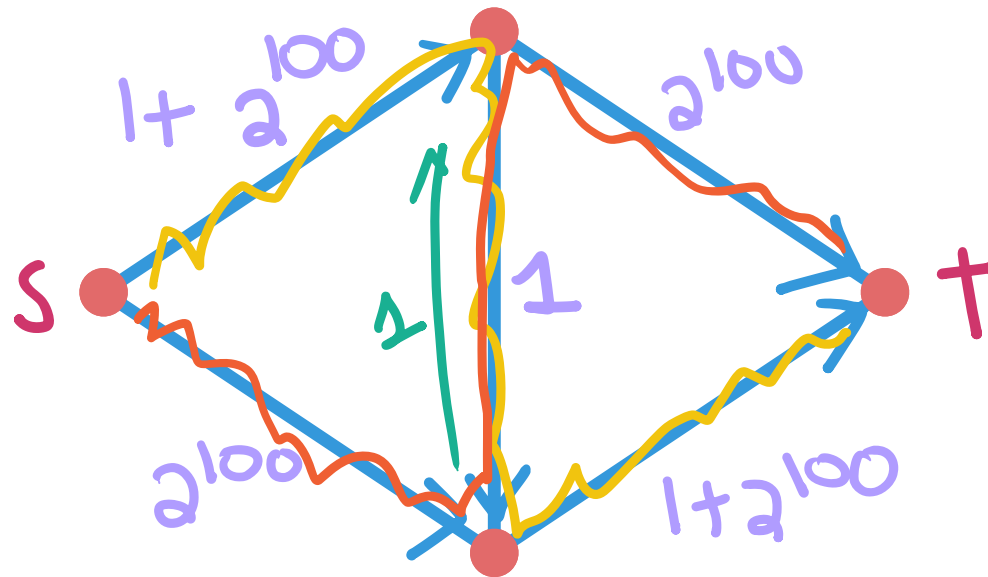
↪ reasonable, but not quite polynomial time

① while there is a path $p: s \rightsquigarrow t$:

② take p and route as much flow as possible

③ update residual graph

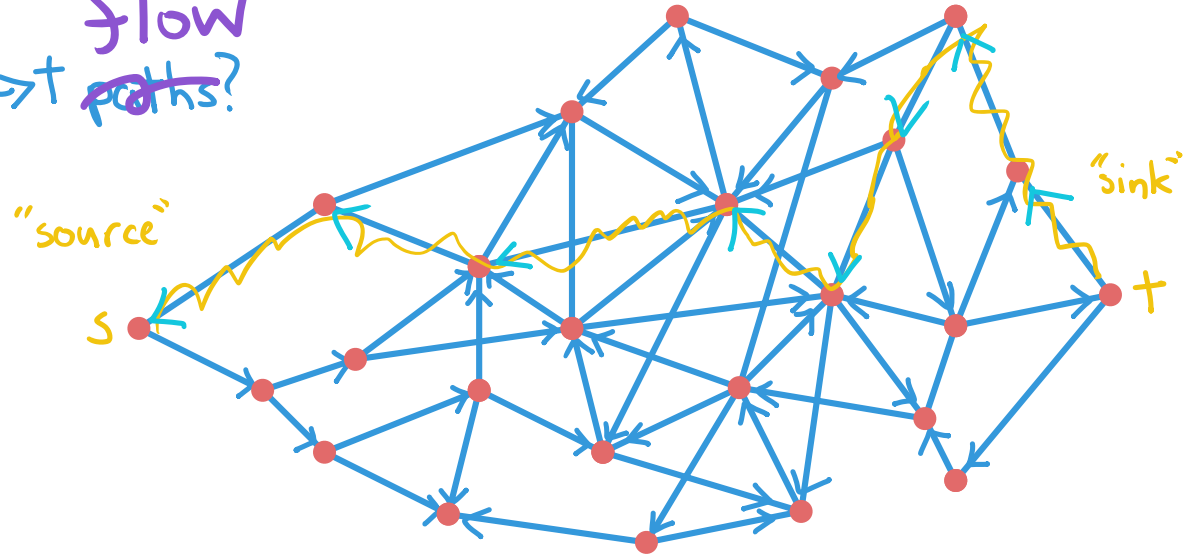
$$\underline{O(\text{poly}(n) \log(\lambda))}$$



How to compute

⑦ max ~~# of edge-disjoint~~ $s \rightsquigarrow t$ ~~paths?~~ ^{flow}

shortest path
(Edmonds, Karp)



① while there is a path $p: s \rightsquigarrow t$

Ⓐ take p and ~~remove all its~~
~~edges from the graph~~

route as much
flow as possible

Ⓑ add the reverse of all of p 's
edges in the graph

Ⓒ uncross p w/ ~~previously selected~~
~~paths~~

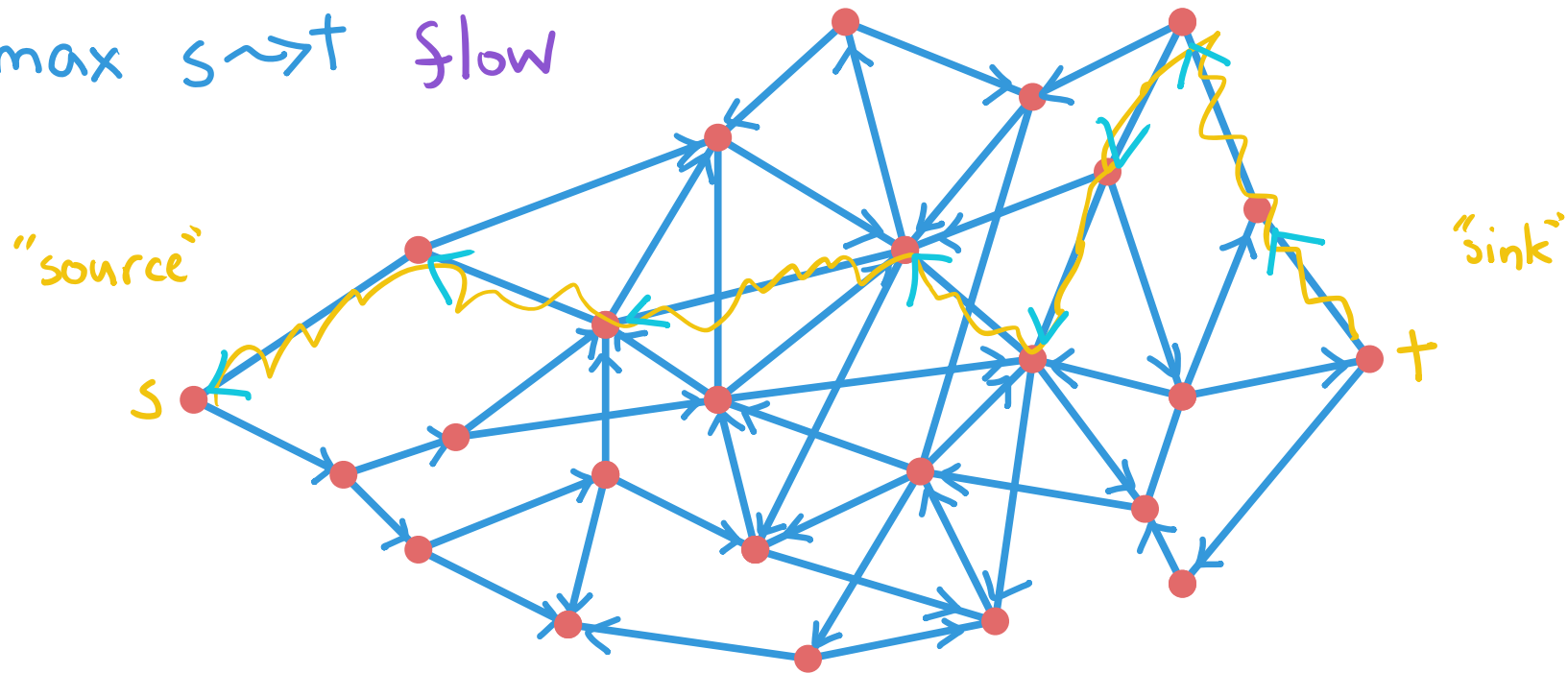
update residual graph

(cleaned up)

(Edmonds, Karp)

How to compute

⑦ max $s \rightsquigarrow t$ flow



① while there is a path $p: s \rightsquigarrow t$:

Ⓐ take shortest path p and route as much flow as possible

Ⓑ update residual graph

Analysis

- ① while there is a path $p: s \rightsquigarrow t$:
 - Ⓐ take shortest path p and route as much flow as possible
 - Ⓑ update residual graph

Key
Lemma

$\forall v \in V$, distance from s to v in residual graph never decreases

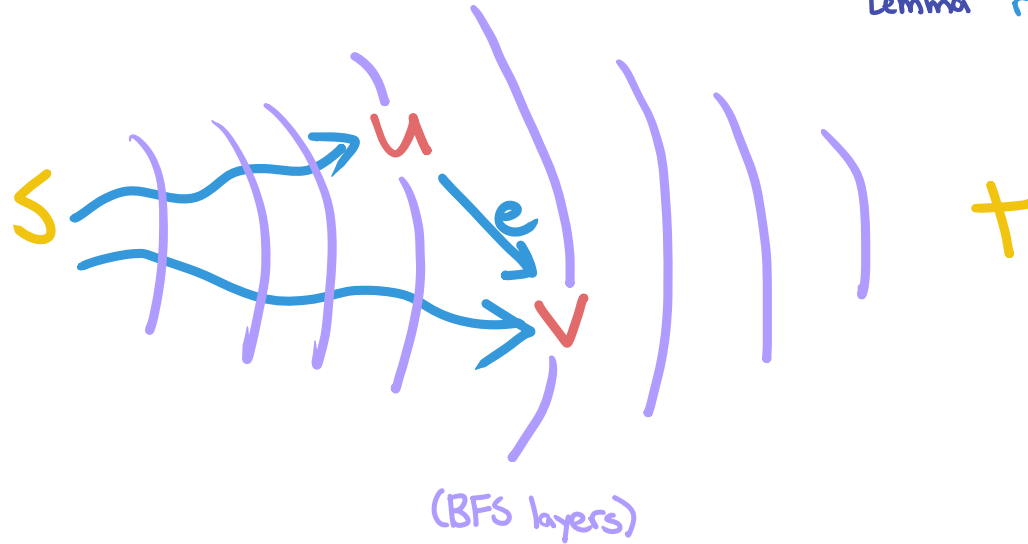
let us first assume lemma is true

fix $e = (u \rightarrow v) \in E$.

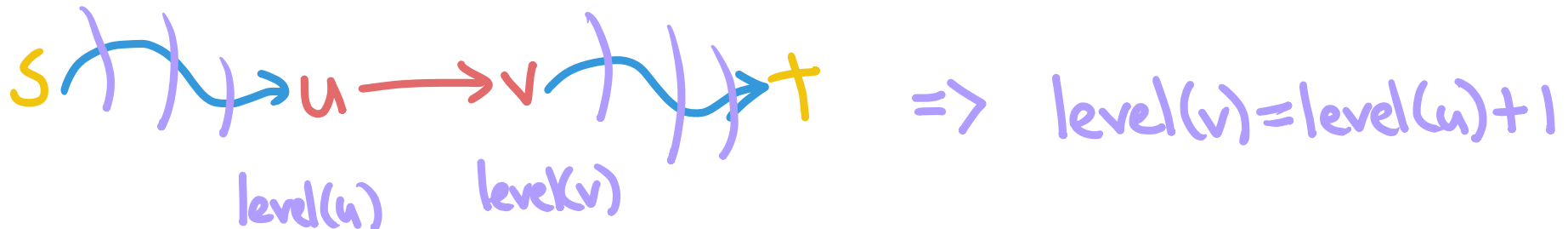
consider distances from s to u & v

- ① while there is a path $p: s \rightsquigarrow t$:
 - Ⓐ take shortest path p and route as much flow as possible
 - Ⓑ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases



① e removed from ^(residual graph) RG , $\Rightarrow e$ in shortest $s \rightsquigarrow t$ path



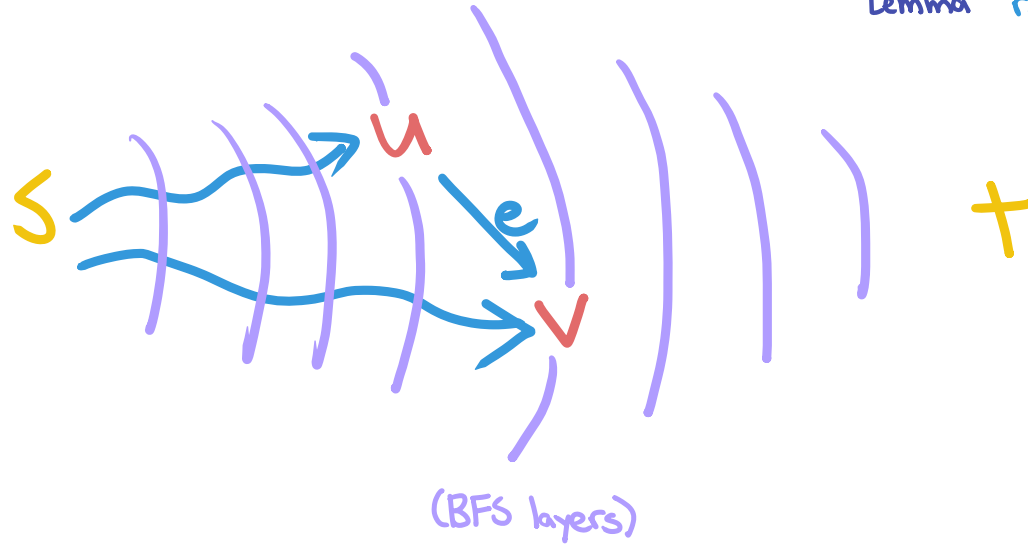
let us first assume lemma is true

fix $e = (u \rightarrow v) \in E$.

consider distances from s to u & v

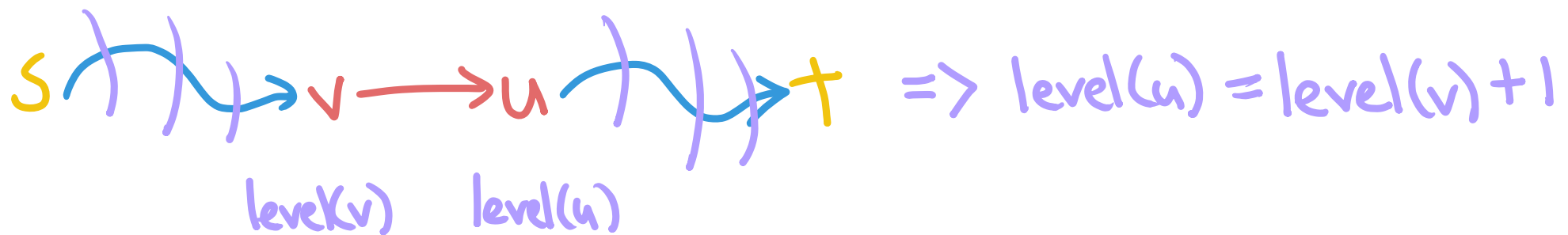
- ① while there is a path $p: s \rightsquigarrow t$:
- ② take shortest path p and route as much flow as possible
- ③ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases



② $e = (u \rightarrow v)$ added back in to RG

$\Rightarrow (v \rightarrow u)$ was in shortest $s \rightsquigarrow t$ path



let us first assume lemma is true

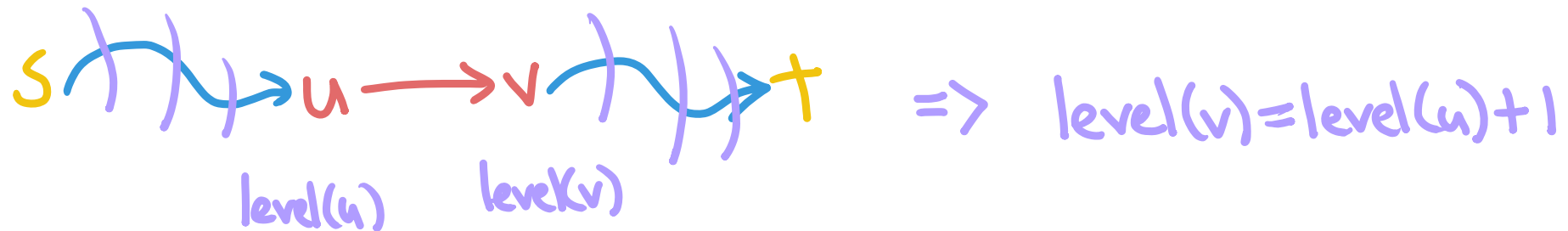
fix $e = (u \rightarrow v) \in E$.

consider distances from s to u & v

- ① while there is a path $p: s \rightsquigarrow t$:
 - @ take shortest path p and route as much flow as possible
 - ⑥ update residual graph

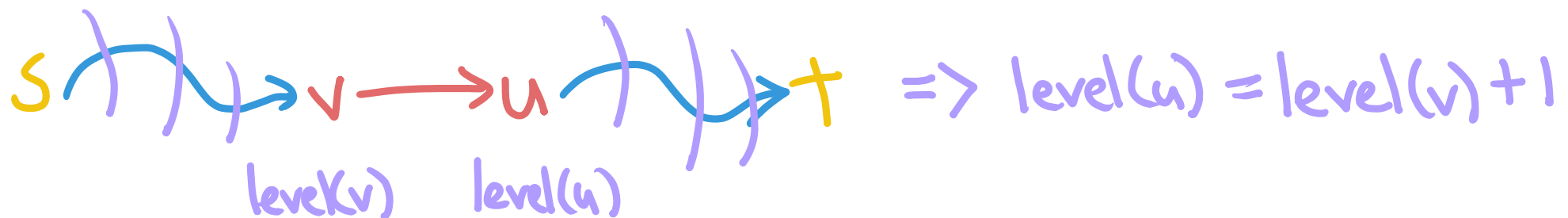
Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

① e removed from ^(residual graph) RG , $\Rightarrow e$ in shortest $s \rightsquigarrow t$ path



② $e = (u \rightarrow v)$ added back in to RG

$\Rightarrow (v \rightarrow u)$ was in shortest $s \rightsquigarrow t$ path



let us first assume lemma is true

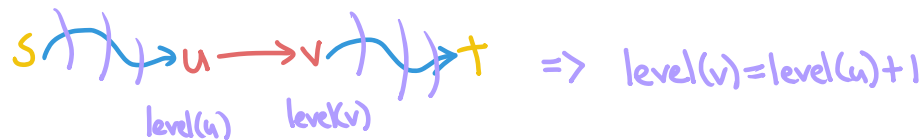
fix $e = (u \rightarrow v) \in E$.

consider distances from s to u & v

- ① while there is a path $p: s \rightsquigarrow t$:
- ② take shortest path p and route as much flow as possible
- ③ update residual graph

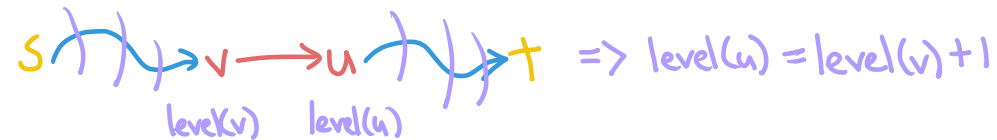
Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

① removed from ^(residual graph) RG , $\Rightarrow e$ in shortest $s \rightsquigarrow t$ path



② $e = (u \rightarrow v)$ added back in to RG

$\Rightarrow (v \rightarrow u)$ was in shortest $s \rightsquigarrow t$ path



each event ①/② $\Rightarrow \text{level}(u), \text{level}(v)$ have switched order.

level's nondecreasing by key lemma

max finite level is $n-1$

$\Rightarrow e$ gets deleted/added at most n times

let us first assume lemma is true

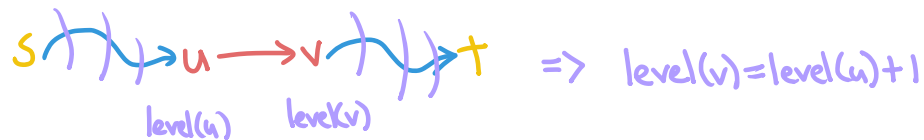
fix $e = (u \rightarrow v) \in E$.

consider distances from s to u & v

- ① while there is a path $p: s \rightsquigarrow t$:
 - @ take shortest path p and route as much flow as possible
 - ⑥ update residual graph

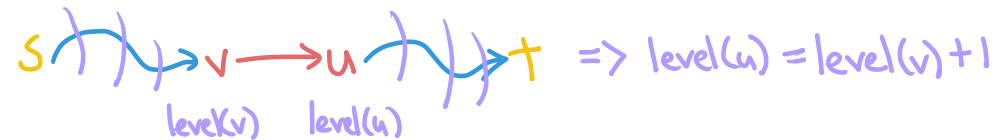
Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

① e removed from ^(residual graph) RG , $\Rightarrow e$ in shortest $s \rightsquigarrow t$ path



② $e = (u \rightarrow v)$ added back in to RG

$\Rightarrow (v \rightarrow u)$ was in shortest $s \rightsquigarrow t$ path



✓ e gets deleted/added at most n times

Last observation each iteration removes (at least)
one edge from residual graph. why?

b/c we route as much as possible

= $O(mn)$ iterations

- ① while there is a path $p: s \rightsquigarrow t$:
 - Ⓐ take shortest path p and route as much flow as possible
 - Ⓑ update residual graph

Running time

$O(m)$ per iteration

$O(mn)$ iterations

$O(m^2n)$ time

Key $\forall v \in V$, distance from s to v (" $l(v)$ ")
Lemma in residual graph never decreases

- ① while there is a path $p: s \rightsquigarrow t$:
- ② take shortest path p and route as much flow as possible
- ③ update residual graph

proof

Suppose we augment along $p: s \rightsquigarrow t$
need to show no levels decrease

proof by induction on level after updating residual graph.

Base case: level 0

s is the only vertex at level 0, which never change

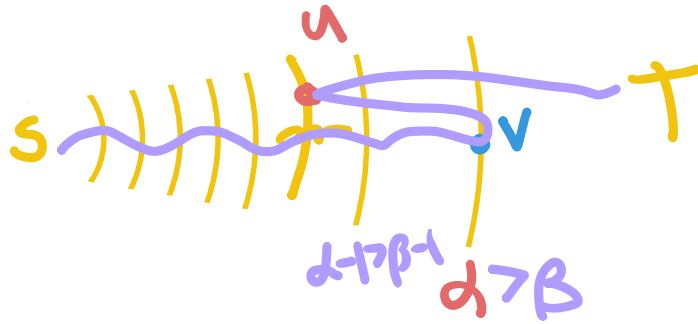
General case: level $\beta > 0$

let $l(v) = d$ before updating
 $l(v) = \beta > 0$ after updating

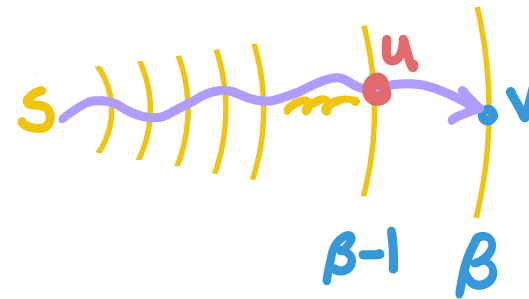
- ① while there is a path $p: s \rightarrow t$:
- ② take shortest path p and route as much flow as possible
- ③ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

Before



After



$u =$ 2nd to last vertex on shortest (s, v) path after

level(u) = $\beta-1$ after augmenting

level(u) $\leq \beta-1$ before aug. by induction

suppose by contrad. $d > \beta$. level(u) $\leq d-2$ before aug

$\Rightarrow (v, u)$ was on shortest (s, t) -path
before

- ① while there is a path $p: s \rightsquigarrow t$:
 - Ⓐ take shortest path p and
route as much flow as possible
 - Ⓑ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

Edmonds, Karp

- ① while there is a path $p: s \rightsquigarrow t$:
 - Ⓐ take shortest path p and route as much flow as possible
 - Ⓑ update residual graph

computes max (s,t) -flow, min (s,t) -cut in $O(m^2n)$ time

Key Lemma $\forall v \in V$, distance from s to v in residual graph never decreases

① while there is a path $p: s \rightarrow t$:

② take shortest path p and
route as much flow as possible

③ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

① while there is a path $p: s \rightarrow t$:

② take shortest path p and
route as much flow as possible

③ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

① while there is a path $p: s \rightarrow t$:

② take shortest path p and
route as much flow as possible

③ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

① while there is a path $p: s \rightarrow t$:

② take shortest path p and
route as much flow as possible

③ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

① while there is a path $p: s \rightarrow t$:

② take shortest path p and
route as much flow as possible

③ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

① while there is a path $p: s \rightarrow t$:

② take shortest path p and
route as much flow as possible

③ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases

① while there is a path $p: s \rightarrow t$:

② take shortest path p and
route as much flow as possible

③ update residual graph

Key $\forall v \in V$, distance from s to v in
lemma residual graph never decreases