Today

Max Flow

Maximum flow

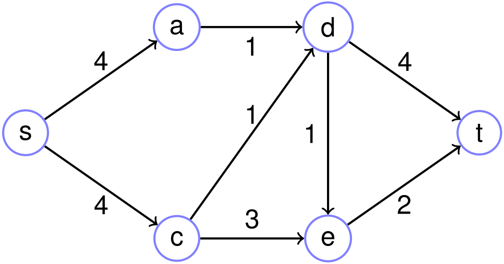
Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



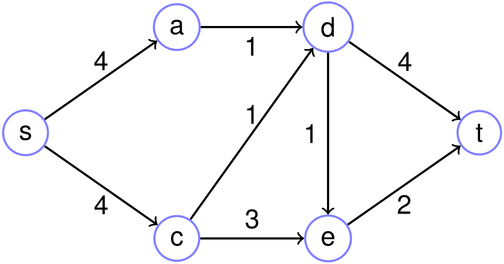
Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.

Maximize: $\text{size}(f) = \sum_{(s,u)\in E} f_{su}$.

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



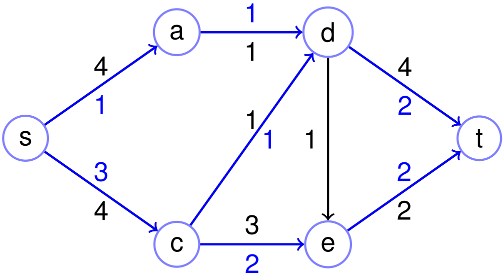
Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.

Maximize: size$(f) = \sum_{(s,u) \in E} f_{su}$.

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



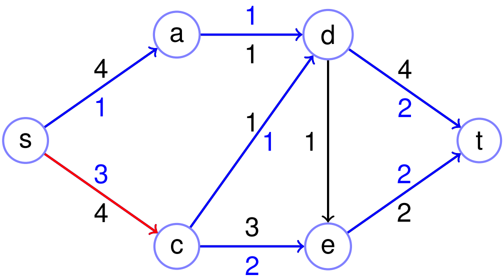
Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.

Maximize: size$(f) = \sum_{(s,u) \in E} f_{su}$.

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



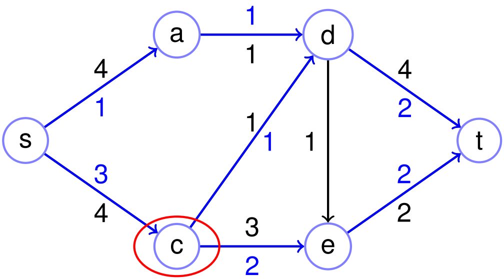
Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints." $3 = f_{s,c} \leq c_{s,c} = 4.$
2. If u is not s or t
   $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$.

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



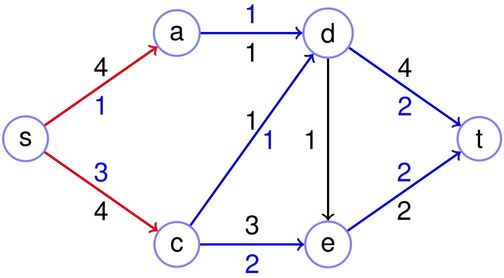
Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints." $3 = f_{s,c} \leq c_{s,c} = 4.$
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.$

Maximize: $\text{size}(f) = \sum_{(s,u)\in E} f_{su}$.

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



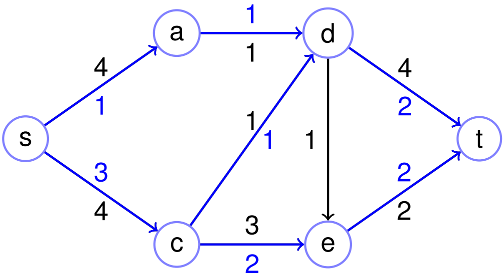
Find Flow: f_e

1. $0 \le f_e \le c_e$. "Capacity constraints." $3 = f_{s,c} \le c_{s,c} = 4.$
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.$

Maximize: size$(f) = \sum_{(s,u)\in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



Find Flow: f_e

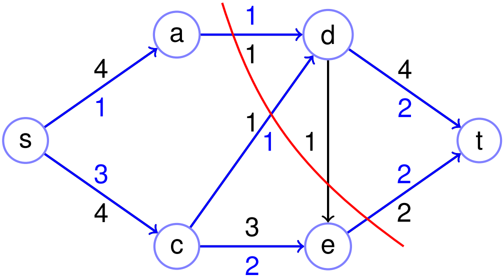
1. $0 \leq f_e \leq c_e$. "Capacity constraints." $3 = f_{s,c} \leq c_{s,c} = 4$.
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1$.

Maximize: $\text{size}(f) = \sum_{(s,u)\in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal?

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



Find Flow: f_e

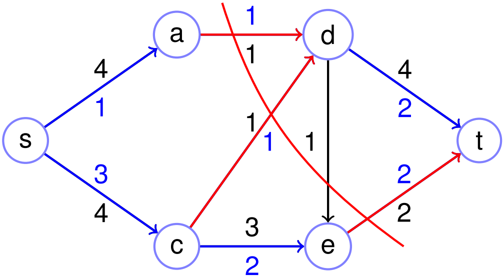
1. $0 \leq f_e \leq c_e$. "Capacity constraints." $3 = f_{s,c} \leq c_{s,c} = 4$.
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1$.

Maximize: $\text{size}(f) = \sum_{(s,u)\in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal?

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



Find Flow: f_e

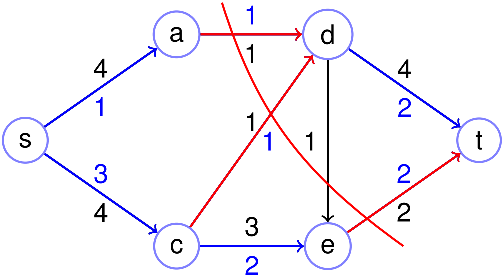
1. $0 \le f_e \le c_e$. "Capacity constraints." $3 = f_{s,c} \le c_{s,c} = 4$.
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1$.

Maximize: size$(f) = \sum_{(s,u)\in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal? $c_{ad} + c_{cd} + c_{et} = 1 + 1 + 2 = 4$.

Maximum flow

Flow network $G = (V, E)$, source s, sink $t \in V$, capacities $c_e > 0$.



Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints." $3 = f_{s,c} \leq c_{s,c} = 4.$
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.$

Maximize: $\text{size}(f) = \sum_{(s,u)\in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal? $c_{ad} + c_{cd} + c_{et} = 1 + 1 + 2 = 4.$

Any cut gives an upper bound.

Algorithms.

FindFlow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Algorithms.

FindFlow: f_e

1. $0 \le f_e \le c_e$. "Capacity constraints."
2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Algorithms.

FindFlow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables f_e, linear constraints, linear optimization function.

Algorithms.

FindFlow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables f_e, linear constraints, linear optimization function.

Cool!

S-T cut.

An $s - t$ cut is a partition of V into S and T where $s \in S$ and $t \in T$. Its capacity is the total capacity of edges from S to T.

Do you know the definition?

Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.

Do you know the definition?

Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.

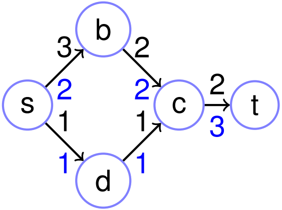
Valid or Invalid?

Do you know the definition?

Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.

Valid or Invalid?

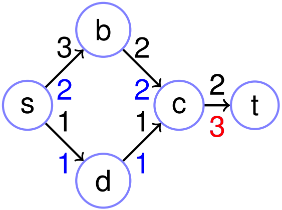


Do you know the definition?

Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.

Valid or Invalid?

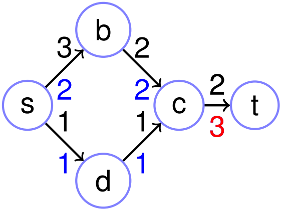


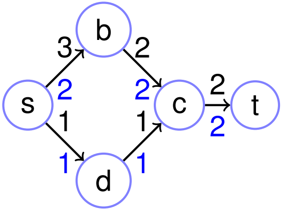
Do you know the definition?

Find Flow: f_e

1. $0 \le f_e \le c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.

Valid or Invalid?



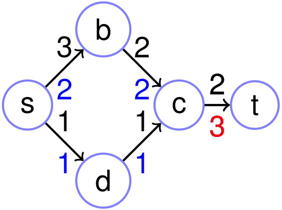


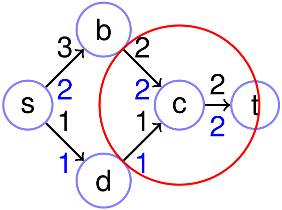
Do you know the definition?

Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.

Valid or Invalid?





$2 + 1 \neq 2$

Algorithms.

FindFlow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Algorithms.

FindFlow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Algorithms.

FindFlow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables f_e, linear constraints, linear optimization function.

Algorithms.

FindFlow: f_e

1. $0 \le f_e \le c_e$. "Capacity constraints."
2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables f_e, linear constraints, linear optimization function.

Cool!

Algorithms.

FindFlow: f_e

1. $0 \le f_e \le c_e$. "Capacity constraints."
2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables f_e, linear constraints, linear optimization function.

Cool!

Note...

Algorithms.

FindFlow: f_e

1. $0 \le f_e \le c_e$. "Capacity constraints."
2. $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables f_e, linear constraints, linear optimization function.

Cool!

Note...

Integer? (Given integer capacities.)

Algorithms.

FindFlow: f_e

1. $0 \leq f_e \leq c_e$. “Capacity constraints.”
2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables f_e, linear constraints, linear optimization function.

Cool!

Note...

Integer? (Given integer capacities.)

Yes. There is an integer vertex solution!

Algorithms.

FindFlow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables f_e, linear constraints, linear optimization function.

Cool!

Note...

Integer? (Given integer capacities.)

Yes. There is an integer vertex solution!

Constraint matrix has every subdeterminant being 1, 0, -1.

Algorithms.

FindFlow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.
3. maximize $\sum_{su} f_{su}$.

Linear program!

Variables f_e, linear constraints, linear optimization function.

Cool!

Note...

Integer? (Given integer capacities.)

Yes. There is an integer vertex solution!

Constraint matrix has every subdeterminant being 1, 0, -1.

Vertex solution to linear program (for this problem) must be integral!

Ford-Fulkerson.

"Simplex" method.

Ford-Fulkerson.

"Simplex" method.

Find s to t path with remaining capacity.

Ford-Fulkerson.

"Simplex" method.

Find s to t path with remaining capacity.

Add to flow variables along path.

Ford-Fulkerson.

"Simplex" method.

Find s to t path with remaining capacity.

Add to flow variables along path.

Update remaining capacity.

Ford-Fulkerson.

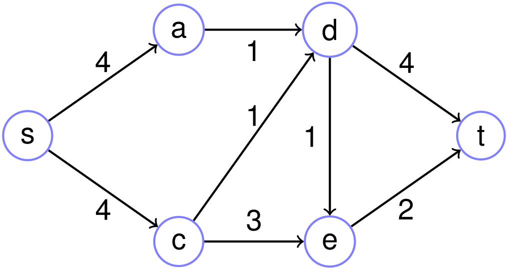
"Simplex" method.

Find s to t path with remaining capacity.

Add to flow variables along path.

Update remaining capacity.

Repeat.



Ford-Fulkerson.

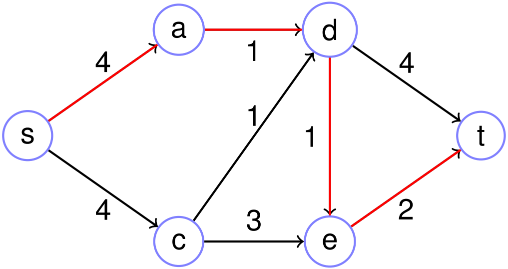
"Simplex" method.

Find s to t path with remaining capacity.

Add to flow variables along path.

Update remaining capacity.

Repeat.



Ford-Fulkerson.

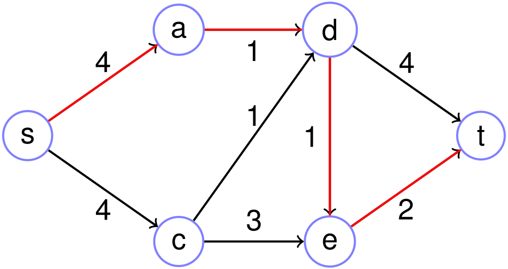
"Simplex" method.

Find s to t path with remaining capacity.

Add to flow variables along path.

Update remaining capacity.

Repeat.



Ford-Fulkerson.

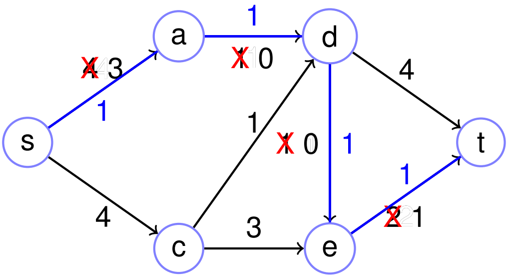
"Simplex" method.

Find s to t path with remaining capacity.

Add to flow variables along path.

Update remaining capacity.

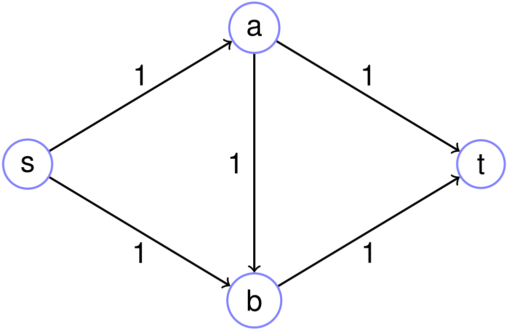
Repeat.



Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path.
Update remaining capacity.

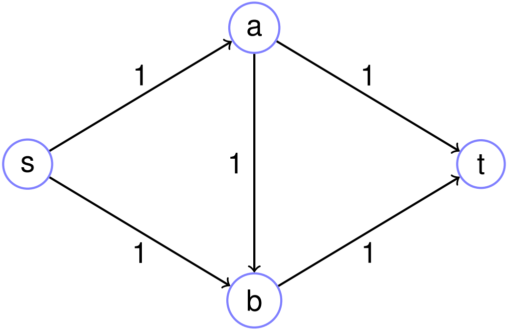
Repeat.



Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path.
Update remaining capacity.

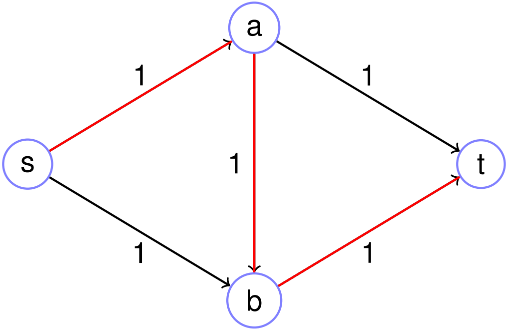
Repeat.



Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path.
Update remaining capacity.

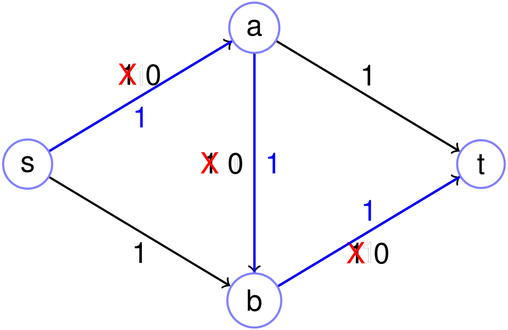
Repeat.



Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path.
Update remaining capacity.

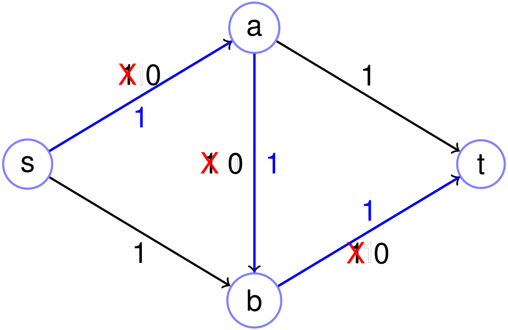
Repeat.



Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path.
Update remaining capacity.

Repeat.

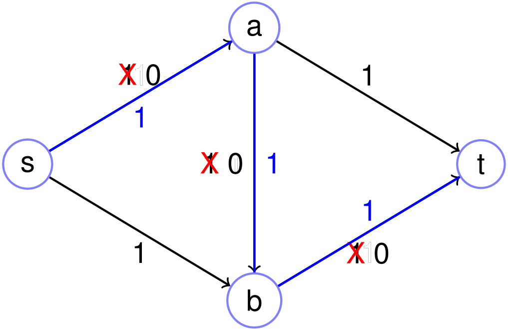


No remaining path.

Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path.
Update remaining capacity.

Repeat.

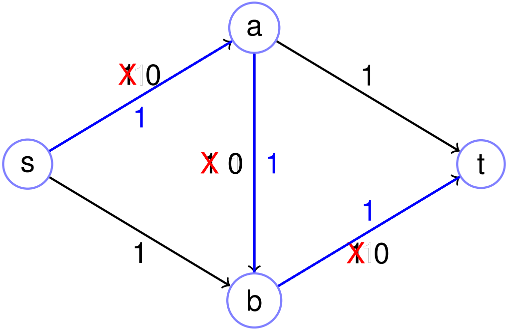


No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)

Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path.
Update remaining capacity.

Repeat.

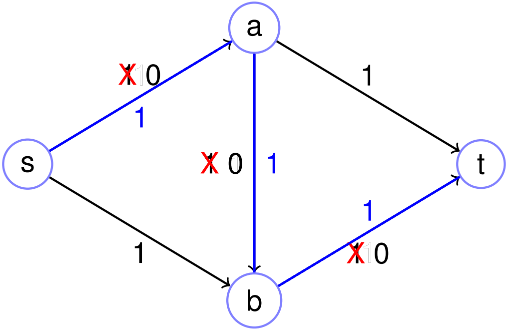


No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)

Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path. Or reduce flow on reverse edge.
Update remaining capacity.

Repeat.

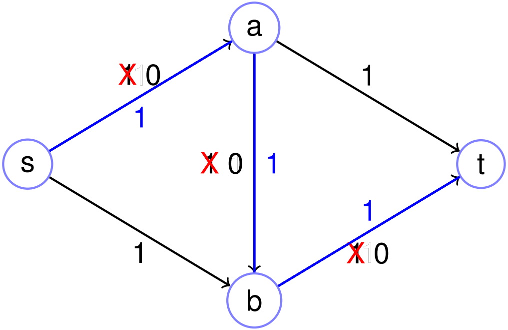


No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)

Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path. Or reduce flow on reverse edge.
Update remaining capacity.
Reduce $r_e = c_e - f_e$

Repeat.

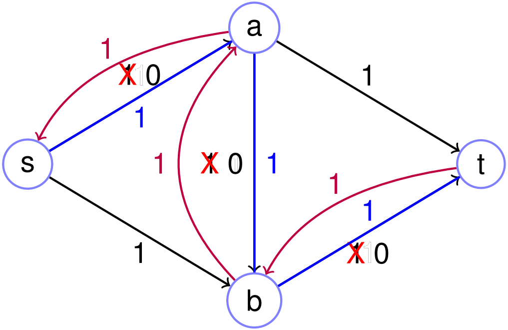


No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)

Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path. Or reduce flow on reverse edge.
Update remaining capacity.
Reduce $r_e = c_e - f_e$
and add reverse $r_{uv} = f_{vu}$

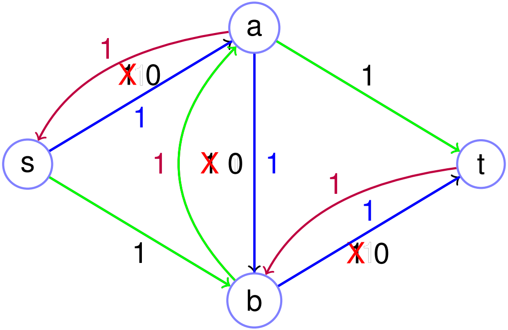
Repeat.



No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)
Add reverse arcs to indicate "reverse" capacity.

Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path. Or reduce flow on reverse edge.
Update remaining capacity.
Reduce $r_e = c_e - f_e$
and add reverse $r_{uv} = f_{vu}$
Repeat.

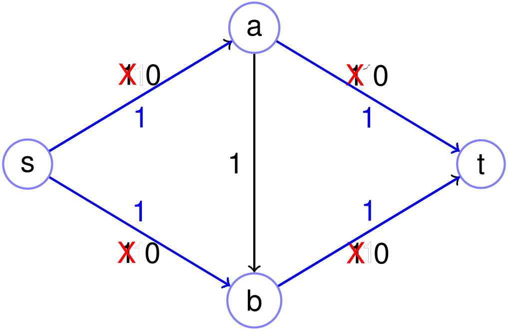


No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)
Add reverse arcs to indicate "reverse" capacity.

Residual Capacity.

Find s to t path with remaining capacity.
Add to flow along path. Or reduce flow on reverse edge.
Update remaining capacity.
Reduce $r_e = c_e - f_e$
and add reverse $r_{uv} = f_{vu}$

Repeat.



No remaining path. Uh oh! Optimal is 2! (At most 2 due to cut.)
Add reverse arcs to indicate “reverse” capacity.

Bigger Example.

Bigger Example.

Find s to t path with remaining capacity.

Bigger Example.

Find s to t path with remaining capacity.

Add to flow along path.

Bigger Example.

Find s to t path with remaining capacity.

Add to flow along path.

Update residual capacities: : $r_e = c_e - f_e; r_{uv} = f_{vu}$.

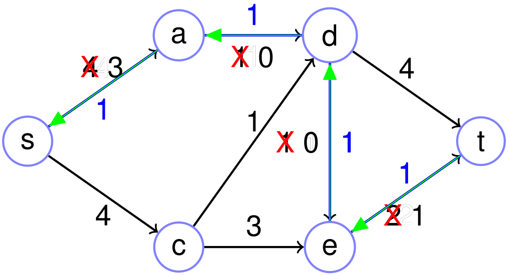
Bigger Example.

Find s to t path with remaining capacity.

Add to flow along path.

Update residual capacities: : $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.



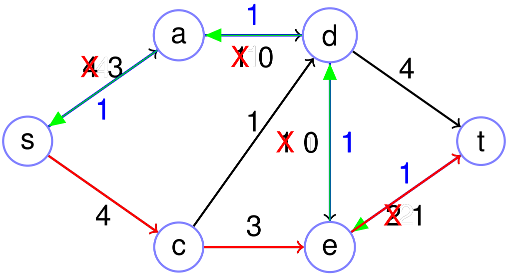
Bigger Example.

Find s to t path with remaining capacity.

Add to flow along path.

Update residual capacities: : $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.



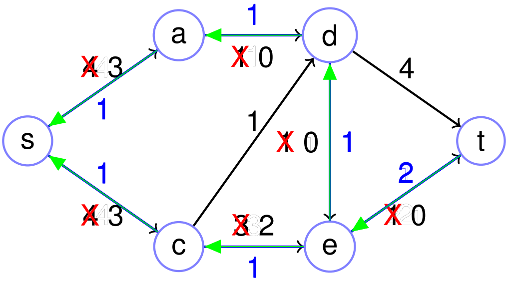
Bigger Example.

Find s to t path with remaining capacity.

Add to flow along path.

Update residual capacities: : $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.



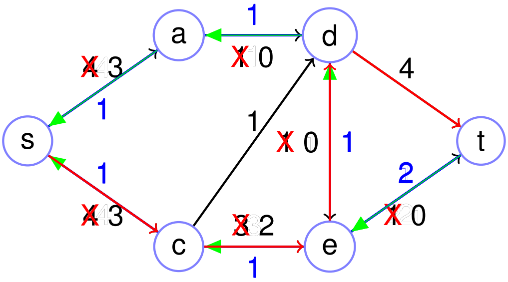
Bigger Example.

Find s to t path with remaining capacity.

Add to flow along path.

Update residual capacities: : $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.



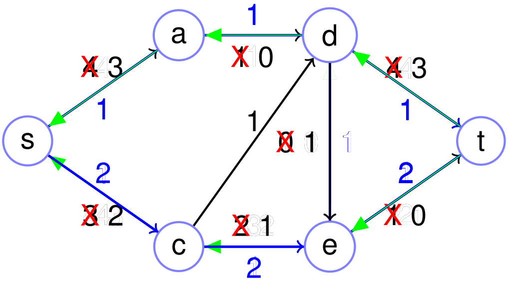
Bigger Example.

Find s to t path with remaining capacity.

Add to flow along path.

Update residual capacities: : $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.



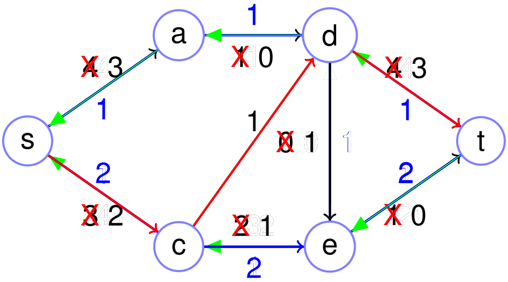
Bigger Example.

Find s to t path with remaining capacity.

Add to flow along path.

Update residual capacities: : $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.



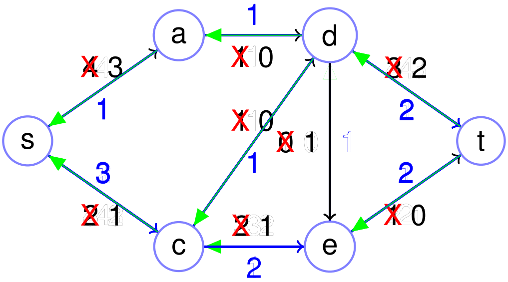
Bigger Example.

Find s to t path with remaining capacity.

Add to flow along path.

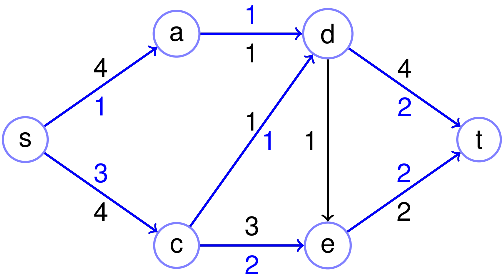
Update residual capacities: : $r_e = c_e - f_e; r_{uv} = f_{vu}$.

Repeat.



Check Result...

Check Result...

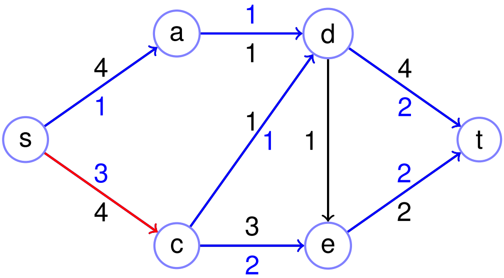


Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints."
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$.

Maximize: size$(f) = \sum_{(s,u)\in E} f_{su}$.

Check Result...

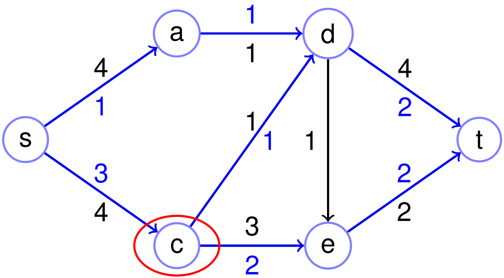


Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints." $3 = f_{s,c} \leq c_{s,c} = 4$.
2. If u is not s or t
   $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$.

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$.

Check Result...

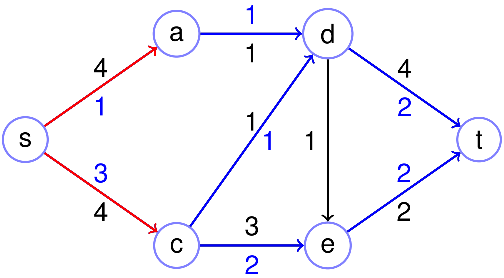


Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints." $3 = f_{s,c} \leq c_{s,c} = 4.$
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.$

Maximize: size$(f) = \sum_{(s,u)\in E} f_{su}$.

Check Result...

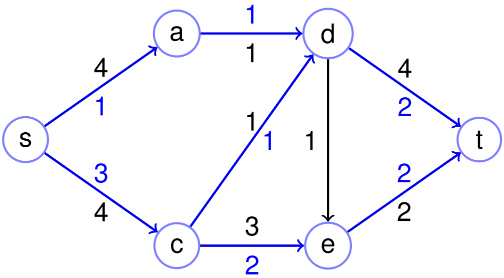


Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints." $3 = f_{s,c} \leq c_{s,c} = 4$.
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1$.

Maximize: $\text{size}(f) = \sum_{(s,u)\in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Check Result...



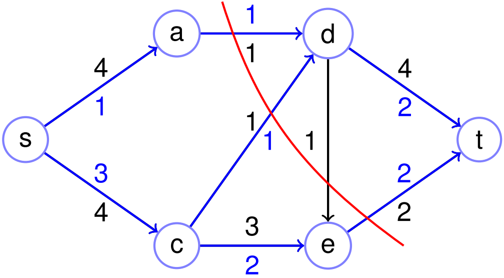
Find Flow: f_e

1. $0 \le f_e \le c_e$. "Capacity constraints." $3 = f_{s,c} \le c_{s,c} = 4$.
2. If u is not s or t
   $\sum_{(w,u)\in E} f_{wu} = \sum_{(u,w)\in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1$.

Maximize: $\text{size}(f) = \sum_{(s,u)\in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal?

Check Result...



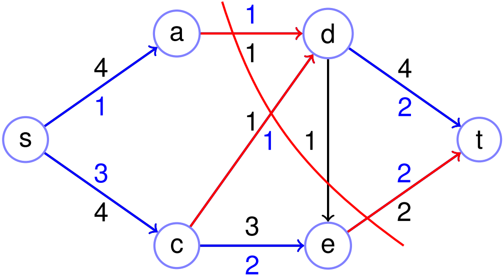
Find Flow: f_e

1. $0 \leq f_e \leq c_e$. "Capacity constraints." $3 = f_{s,c} \leq c_{s,c} = 4.$
2. If u is not s or t
   $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.$

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal?

Check Result...



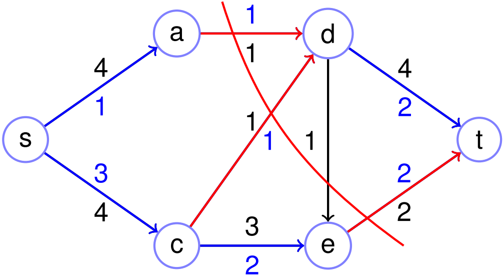
Find Flow: f_e

1. $0 \le f_e \le c_e$. "Capacity constraints." $3 = f_{s,c} \le c_{s,c} = 4.$
2. If u is not s or t
   $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1.$

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal? $c_{ad} + c_{cd} + c_{et} = 1 + 1 + 2 = 4.$

Check Result...



Find Flow: f_e

1. $0 \le f_e \le c_e$. "Capacity constraints." $3 = f_{s,c} \le c_{s,c} = 4$.
2. If u is not s or t
   $\sum_{(w,u) \in E} f_{wu} = \sum_{(u,w) \in E} f_{uw}$. $3 = f_{s,c} = f_{c,d} + f_{c,e} = 2 + 1$.

Maximize: $\text{size}(f) = \sum_{(s,u) \in E} f_{su}$. $f_{sa} + f_{sc} = 1 + 3 = 4$

Optimal? $c_{ad} + c_{cd} + c_{et} = 1 + 1 + 2 = 4$.

Any $s - t$ cut gives an upper bound.

Correctness.

Correctness.

1. Capacity Constraints: $0 \le f_e \le c_e$.

Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
Only increase flow to c_e.

Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
Only increase flow to c_e.
Or use reverse arcs decrease to 0.

Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
Only increase flow to c_e.
Or use reverse arcs decrease to 0.
Flow values to be between 0 and c_e.

Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
Only increase flow to c_e.
Or use reverse arcs decrease to 0.
Flow values to be between 0 and c_e.

2. Conservation Constraints:

Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
Only increase flow to c_e.
Or use reverse arcs decrease to 0.
Flow values to be between 0 and c_e.

2. Conservation Constraints:
"flow into v" = "flow out of v" (if not s or t.)

Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
Only increase flow to c_e.
Or use reverse arcs decrease to 0.
Flow values to be between 0 and c_e.

2. Conservation Constraints:
"flow into v" = "flow out of v" (if not s or t.)
Algorithm adds flow, say f, to path from s to t.

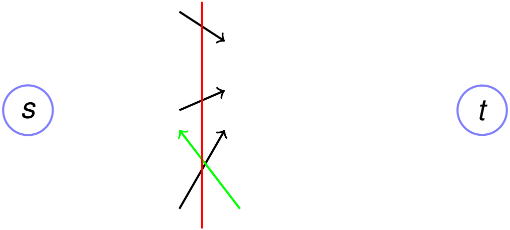
Correctness.

1. Capacity Constraints: $0 \leq f_e \leq c_e$.
Only increase flow to c_e.
Or use reverse arcs decrease to 0.
Flow values to be between 0 and c_e.

2. Conservation Constraints:
"flow into v" = "flow out of v" (if not s or t.)
Algorithm adds flow, say f, to path from s to t.
Each internal node has f in, and f out.

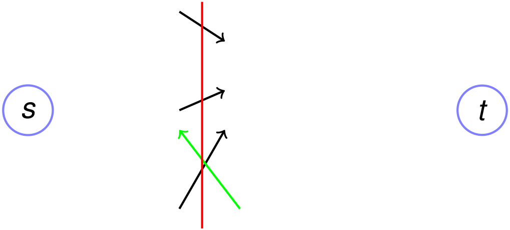
Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Optimality: upper bound.

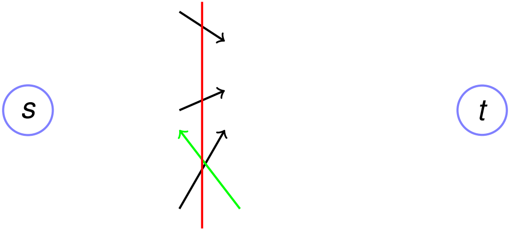
s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

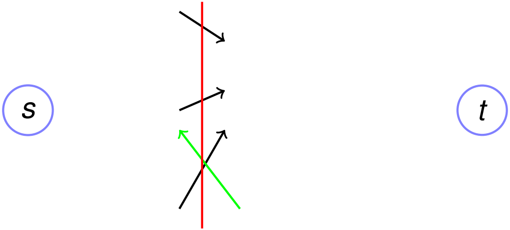


Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T

Optimality: upper bound.

s-*t* Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.

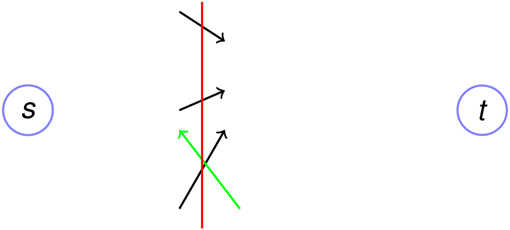


Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u\in S,V\in T} c_e$

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



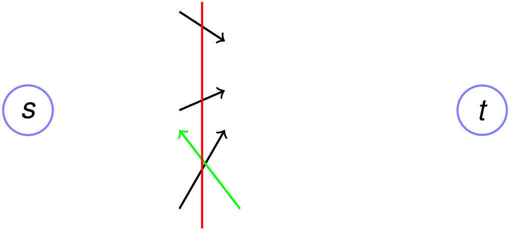
Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



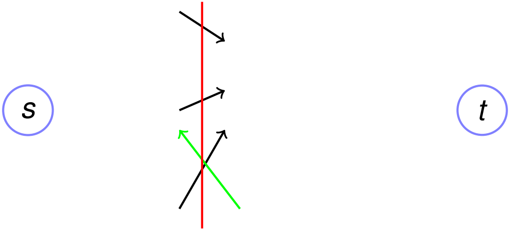
Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.

Optimality: upper bound.

s-*t* Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



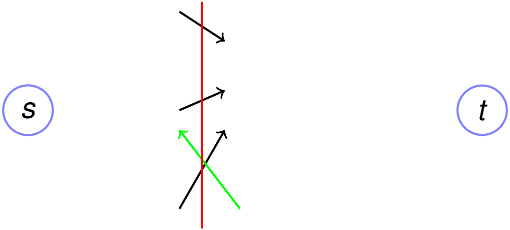
Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



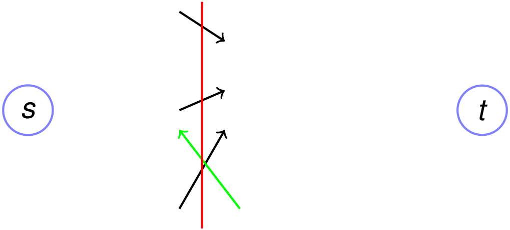
Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

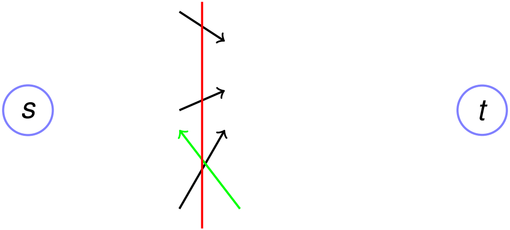
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

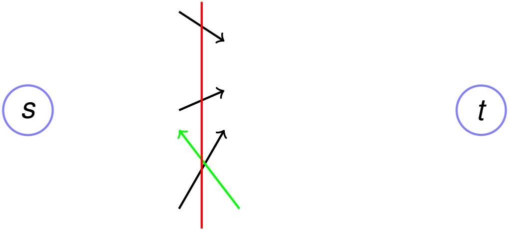
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

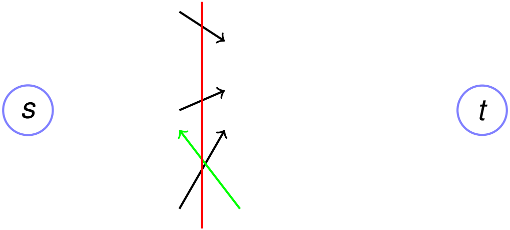
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v): u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e$

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

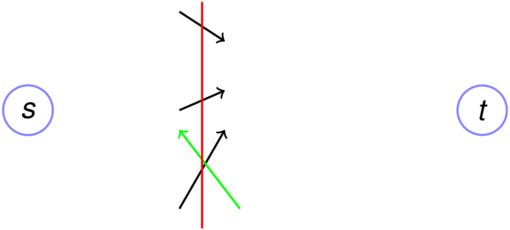
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e$

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

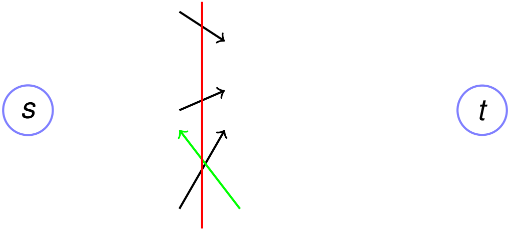
$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0$

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

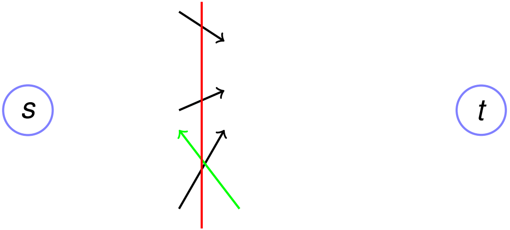
$C(S, T)$ - sum of capacities of all arcs from S to T
$C(S, T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \rightarrow Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S, T).$

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

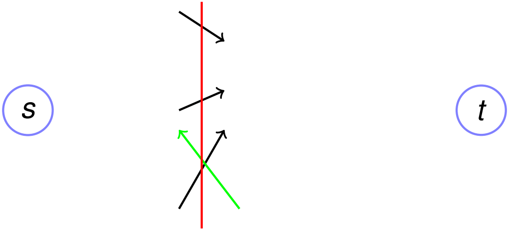
For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S,T)$.

$\rightarrow$ The value of any valid flow is at most $C(S,T)$!

Optimality: upper bound.

s-t Cut: $V = S \cup T$ and $s \in S$ and $t \in T$.



Lemma: Capacity of any $s - t$ cut is an upper bound on the flow.

$C(S,T)$ - sum of capacities of all arcs from S to T
$C(S,T) = \sum_{e=(u,v):u \in S, V \in T} c_e$

For valid flow:
Flow out of (S) = Flow out of s.
Flow into (T) = Flow into t.

For any valid flow, $f : E \to Z+$, the flow out of S (into T)
$\sum_{e \in S \times T} f_e - \sum_{e \in T \times S} f_e \leq \sum_{e \in S \times T} c_e - \sum_{e \in T \times S} 0 = C(S,T)$.

$\to$ The value of any valid flow is at most $C(S,T)$! $\square$

Optimality: max flow = min cut.

At termination of augmenting path algorithm.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.

- No path with residual capacity!

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.

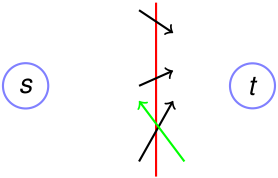
S be reachable nodes.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.

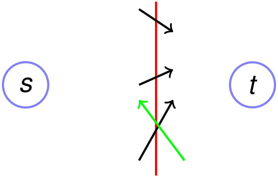
S be reachable nodes.



Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



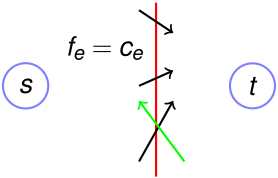
S be reachable nodes.

No arc with positive residual capacity leaving S

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

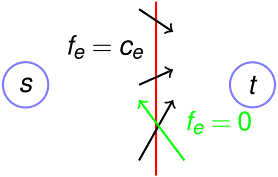
No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

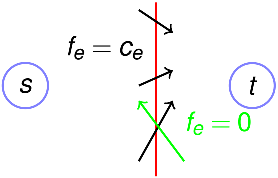
$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

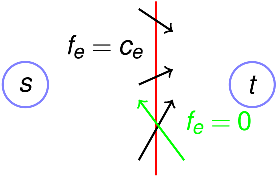
$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

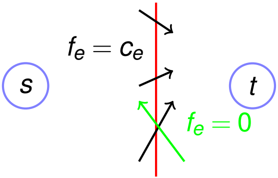
Total flow leaving S is $C(S,T)$.

Valid flow $\implies$ all that flow from source.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

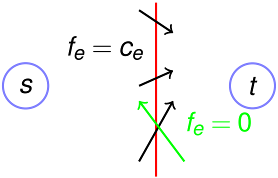
Value of flow equals value of $C(S, T)$.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.

No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S,T)$.

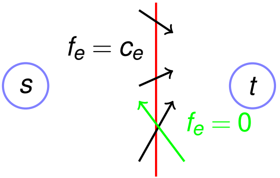
Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S,T)$. and

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S,T)$.

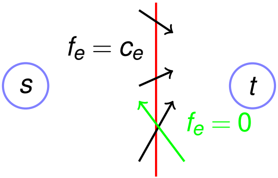
Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S,T)$. and Optimal is $\leq C(S,T)$.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

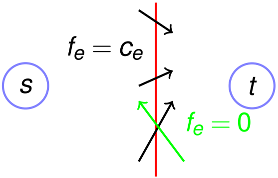
Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

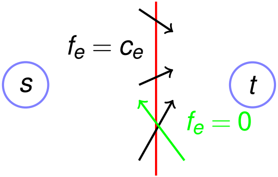
Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S,T)$.

Valid flow $\implies$ all that flow from source.

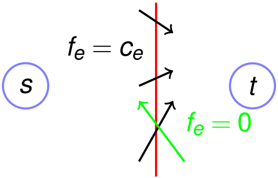
Value of flow equals value of $C(S,T)$. and Optimal is $\leq C(S,T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s-t$ cut too!

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
 No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

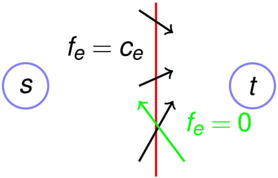
Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s - t$ cut too!
"any flow" $\leq$ "any cut"

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S,T)$.

Valid flow $\implies$ all that flow from source.

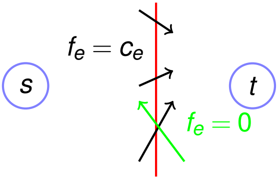
Value of flow equals value of $C(S,T)$. and Optimal is $\leq C(S,T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s-t$ cut too!
"any flow" $\leq$ "any cut" and

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S,T)$.

Valid flow $\implies$ all that flow from source.

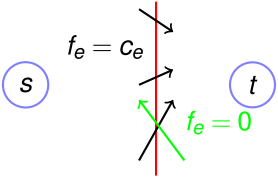
Value of flow equals value of $C(S,T)$. and Optimal is $\leq C(S,T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s-t$ cut too!
"any flow" $\leq$ "any cut" and this flow $=$ this cut.

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

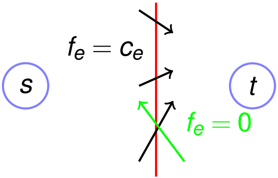
Cut is minimum $s - t$ cut too!
"any flow" $\leq$ "any cut" and this flow $=$ this cut.

$\rightarrow$

Optimality: max flow = min cut.

At termination of augmenting path algorithm.
  No path with residual capacity!

Depth first search only starting at s does not reach t.



S be reachable nodes.

No arc with positive residual capacity leaving S

$\implies$ All arcs leaving S are full.

$\implies$ No arcs into S have flow.

Total flow leaving S is $C(S, T)$.

Valid flow $\implies$ all that flow from source.

Value of flow equals value of $C(S, T)$. and Optimal is $\leq C(S, T)$.
$\rightarrow$ Flow is maximum!!

Cut is minimum $s - t$ cut too!
"any flow" $\leq$ "any cut" and this flow $=$ this cut.
$\rightarrow$ Maximum flow and minimum $s - t$ cut!

Celebrated max flow -minimum cut theorem.

Theorem: In any flow network, the maximum s-t flow is equal to the minimum cut.

Back to business: Algorithm Terminates?

Back to business: Algorithm Terminates?

It will!!

Back to business: Algorithm Terminates?

It will!!

Flow keeps increasing.

Back to business: Algorithm Terminates?

It will!!

Flow keeps increasing.

How long?

Back to business: Algorithm Terminates?

It will!!

Flow keeps increasing.

How long?

One more unit every step!

Back to business: Algorithm Terminates?

It will!!

Flow keeps increasing.

How long?

One more unit every step!

$O(mF)$ time

Back to business: Algorithm Terminates?

It will!!

Flow keeps increasing.

How long?

One more unit every step!

$O(mF)$ time where F is size of flow.

Back to business: Algorithm Terminates?

It will!!

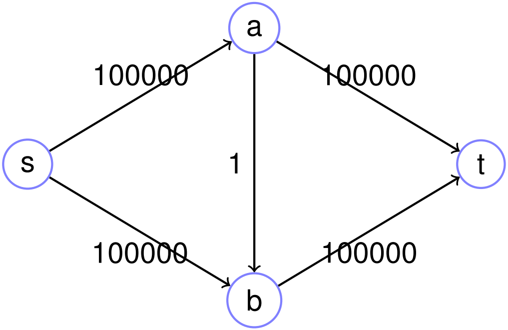
Flow keeps increasing.

How long?

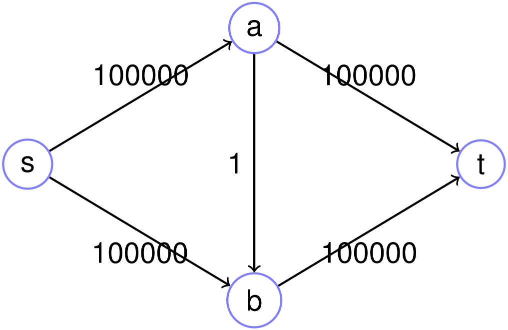
One more unit every step!

$O(mF)$ time where F is size of flow.

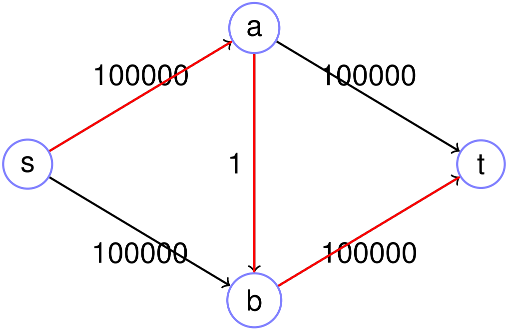
Efficiency.



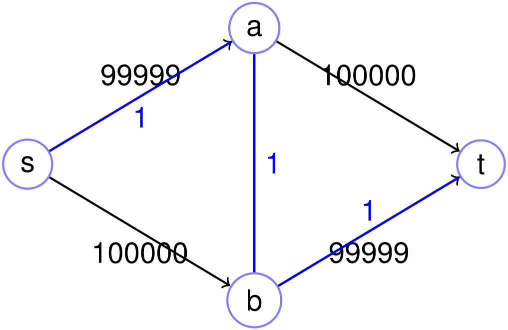
Efficiency.



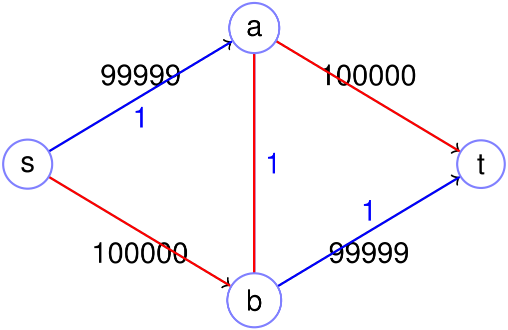
Efficiency.



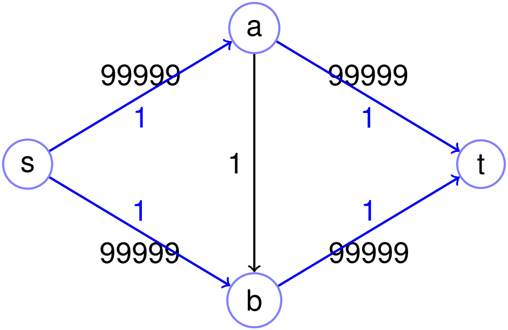
Efficiency.



Efficiency.



Efficiency.



Edmonds-Karp

Augment along shortest path.

Edmonds-Karp

Augment along shortest path.

Breadth first search!

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Analysis idea.

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Analysis idea.

$d(v)$ is distance to sink.

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Analysis idea.

$d(v)$ is distance to sink.

Only route flow on (u, v) if $d(u) \geq d(v)$.

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Analysis idea.

$d(v)$ is distance to sink.

Only route flow on (u,v) if $d(u) \geq d(v)$.

Only reverse flow on (u,v) if $d(v) \leq d(u)$.

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Analysis idea.

$d(v)$ is distance to sink.

Only route flow on (u,v) if $d(u) \geq d(v)$.

Only reverse flow on (u,v) if $d(v) \leq d(u)$.

Maximum $d(v)$ is $|V|$.

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Analysis idea.

$d(v)$ is distance to sink.

Only route flow on (u,v) if $d(u) \geq d(v)$.

Only reverse flow on (u,v) if $d(v) \leq d(u)$.

Maximum $d(v)$ is $|V|$.

Distances only go up. (To prove!)

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Analysis idea.

$d(v)$ is distance to sink.

Only route flow on (u,v) if $d(u) \geq d(v)$.

Only reverse flow on (u,v) if $d(v) \leq d(u)$.

Maximum $d(v)$ is $|V|$.

Distances only go up. (To prove!)

Every augment removes edge "at" a distance.

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Analysis idea.

$d(v)$ is distance to sink.

Only route flow on (u, v) if $d(u) \geq d(v)$.

Only reverse flow on (u, v) if $d(v) \leq d(u)$.

Maximum $d(v)$ is $|V|$.

Distances only go up. (To prove!)

Every augment removes edge "at" a distance.

$O(|V||E|)$ removals.

Edmonds-Karp

Augment along shortest path.

Breadth first search!

$O(|V||E|)$ augmentations.

Analysis idea.

$d(v)$ is distance to sink.

Only route flow on (u,v) if $d(u) \geq d(v)$.

Only reverse flow on (u,v) if $d(v) \leq d(u)$.

Maximum $d(v)$ is $|V|$.

Distances only go up. (To prove!)

Every augment removes edge "at" a distance.

$O(|V||E|)$ removals.

$O(|V||E|^2)$ time.