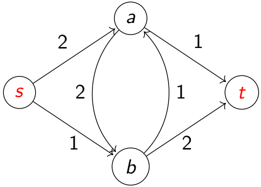
# Max-flow and min-cut problems



# Flow Network

A network $\mathcal{N} = (V, E, c, s, t)$ is formed by

- a digraph $G = (V, E)$,
- a source vertex $s \in V$
- a sink vertex $t \in V$,
- and edge capacities $c : E \to \mathbb{R}^+$



# A flow in a network

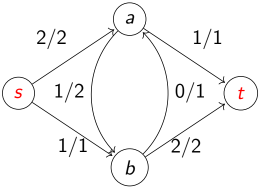


Given a network $\mathcal{N} = (V, E, c, s, t)$
A Flow is an assignment $f : E \to \mathbb{R}^+ \cup \{0\}$ that follows the Kirchoff's laws:

- $\forall (u, v) \in E$, $0 \leq f(u, v) \leq c(u, v)$,
- (Flow conservation) $\forall v \in V - \{s, t\}$, $\sum_{u \in V} f(u, v) = \sum_{z \in V} f(v, z)$

The value of a flow $f$ is

$$|f| = \sum_{v \in V} f(s, v) = f(s, V) = f(V, t).$$




$f(e)/c(e)$

with value 3.

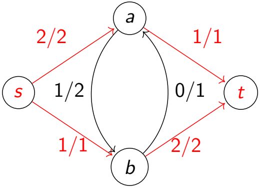
# A flow in a network

Given a network $\mathcal{N} = (V, E, c, s, t)$
A Flow is an assignment $f : E \to \mathbb{R}^+ \cup \{0\}$ that follows the Kirchoff's laws:

- $\forall (u, v) \in E$, $0 \leq f(u, v) \leq c(u, v)$,
- (Flow conservation) $\forall v \in V - \{s, t\}$, $\sum_{u \in V} f(u, v) = \sum_{z \in V} f(v, z)$

The value of a flow $f$ is

$$|f| = \sum_{v \in V} f(s, v) = f(s, V) = f(V, t).$$

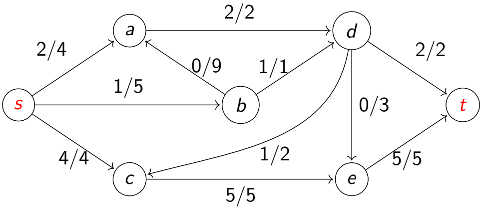



saturated

# The Maximum flow problem



INPUT: A network $\mathcal{N} = (V, E, c, s, t, )$
QUESTION: Find a flow of maximum value on $\mathcal{N}$.




The value of the flow is $7 = 4 + 1 + 2 = 5 + 2$.

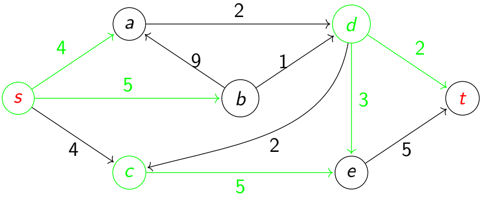
As $t$ cannot receive more flow, this flow is a maximum flow.

# The $(s,t)$-cuts



Given $\mathcal{N} = (V, E, c, s, t)$ a $(s,t)$-cut is a partition of $V = S \cup T$ ($S \cap T = \emptyset$), with $s \in S$ and $t \in T$.

The capacity of a cut $(S, T)$ is the sum of weights leaving $S$, i.e.,
$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$



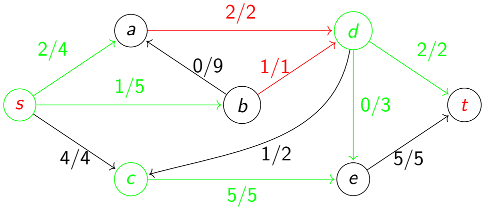

S={s,c,d}
$T = \{a, b, e, t\}$
$c(S,T) = 19$
$(4+5)+5+(3+2)$

# The $(s, t)$-cuts



Given $\mathcal{N} = (V, E, c, s, t)$ a $(s, t)$-cut is a partition of $V = S \cup T$ $(S \cap T = \emptyset)$, with $s \in S$ and $t \in T$.

The flow across the cut:
$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{v \in T} \sum_{u \in S} f(v, u)$.



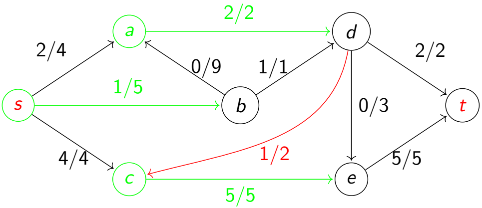

S={s,c,d}
$T = \{a, b, e, t\}$
$c(S, T) = 19$
$f(S, T) = 10 - 3 = 7$

Due to the capacity constrain: $f(S, T) \leq c(S, T)$

# Another $(s,t)$-cut

Given $\mathcal{N} = (V, E, c, s, t)$ a $(s,t)$-cut is a partition of $V = S \cup T$ $(S \cap T = \emptyset)$, with $s \in S$ and $t \in T$.

The flow across the cut:
$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{v \in T} \sum_{u \in S} f(v,u)$.

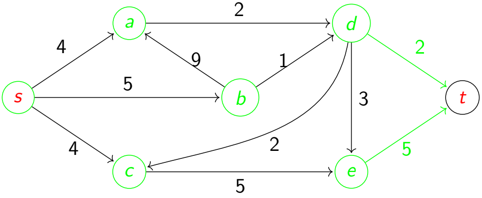



S={s,a,c}
$T = \{b, d, e, t\}$
$c(S,T) = 12$
$f(S,T) = 8 - 1 = 7$

# The Minimum Cut problem



INPUT: A network $\mathcal{N} = (V, E, c, s, t, )$
QUESTION: Find a $(s, t)$-cut of minimum capacity in $\mathcal{N}$.

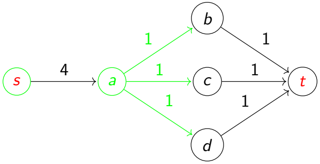


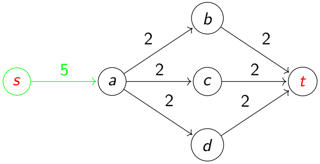

MinCut
S={s,a,b,c,d,e}
$T = \{t\}$
$c(S, T) = 7$

# Changing weights effect on min cuts

Given a network $\mathcal{N} = (V, E, s, t, c)$ assume that $(S, T)$ is a min $(s, t)$-cut.

If we change the input by adding $c > 0$ to the capacity of every edge, then it may happen that $(S, T)$ is not longer a min $(s, t)$-cut.

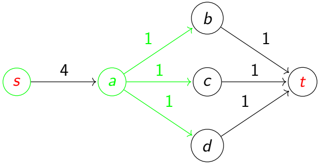


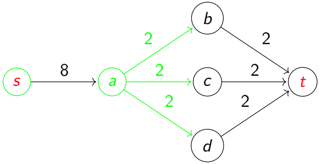



# Changing weights effect on Min-Cut and Max-Flow

Given a network $\mathcal{N} = (V, E, s, t, c)$.

If we change the network by multiplying by $c >$ the capacity of every edge, the capacity of any $(s, t)$-cut in the new network is $c$ times its capacity in the original network.



# Notation

Let $\mathcal{N} = (V, E, s, t, c)$ and $f$ a flow in $\mathcal{N}$

For $v \in V$, $U \subseteq V$ and $v \notin U$.

- $f(v, U)$ flow $v \to U$ i.e. $f(v, U) = \sum_{u \in U} f(v, u)$,
- $f(U, v)$ flow $U \to v$ i.e. $f(U, v) = \sum_{u \in U} f(u, v)$,

For a $(s, t)$-cut $(S, T)$ and $v \in S$

- $S' = S \setminus \{v\}$ and $T' = T \cup \{v\}$
- $f_{-v}(S, T) = \sum_{u \in S'} \sum_{w \in T} f(u, w) - \sum_{w \in T} \sum_{u \in S'} f(w, u)$ i.e, the contribution to $f(S, T)$ from edges not incident with $v$.

# Flow conservation on $(s, t)$-cuts

**Theorem**

*Let $\mathcal{N} = (V, E, s, t, c)$ and $f$ a flow in $\mathcal{N}$. For any $(s, t)$-cut $(S, T)$, $f(S, T) = |f|$.*

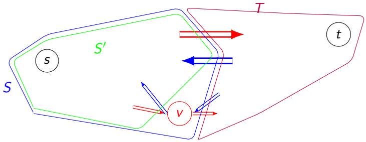
**Proof** (Induction on $|S|$)

- If $S = \{s\}$ then, by definition, $f(S, T) = |f|$.
- Assume it is true for $S' = S - \{v\}$ and $T' = T \cup \{v\}$, i.e. $f(S', T') = |f|$.

# Flow conservation on $(s,t)$-cuts



**Proof (cont.)** (Induction on $|S|$)



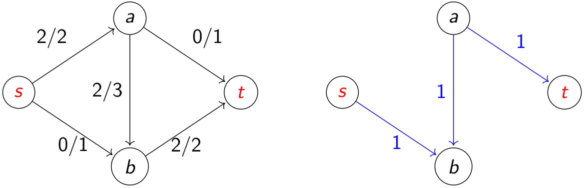

- IH: $f(S', T') = |f|$.
- Then, $f(S,T) = f_{-v}(S,T) + f(v,T) - f(T,v)$.
- But, $f(S', T') = f_{-v}(S,T) + f(S', v) - f(v, S')$ as $v \in T'$
- By flow conservation, $f(S', v) + f(T, v) = f(v, S') + f(v, T)$
- So, $f(S', v) - f(v, S') = f(v, T) - f(T, v)$
- Therefore, $f(S', T') = f(S,T) = |f|$ □

# Residual graph



Given a network $\mathcal{N} = (V, E, s, t, c)$ together with a flow $f$.
The residual graph, $(G_f = (V, E_f, c_f)$ is a weighted digraph on the same vertex set and with edge set:

- if $c(u, v) - f(u, v) > 0$, then $(u, v) \in E_f$ and $c_f(u, v) = c(u, v) - f(u, v) > 0$ (forward edges)




$\mathcal{N}, f$

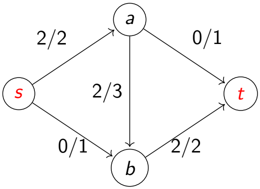
$G_f$ forward edges

# Residual graph

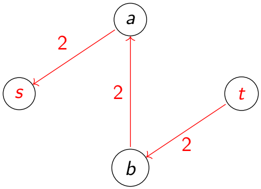


Given a network $\mathcal{N} = (V, E, s, t, c)$ together with a flow $f$ on it, the residual graph, $(G_f = (V, E_f, c_f)$ is a weighted digraph on the same vertex set and with edge set:

- if $f(u, v) > 0$, then $(v, u) \in E_f$ and $c_f(v, u) = f(u, v)$ (backward edges).




$\mathcal{N}, f$



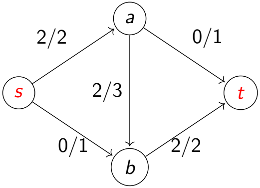

$G_f$ backward edges

# Residual graph

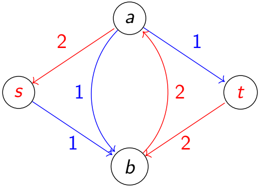


Given a network $\mathcal{N} = (V, E, s, t, c)$ together with a flow $f$ on it, the residual graph, $(G_f = (V, E_f, c_f)$ is a weighted digraph on the same vertex set and with edge set:

- if $c(u, v) - f(u, v) > 0$, then $(u, v) \in E_f$ and $c_f(u, v) = c(u, v) - f(u, v) > 0$ (forward edges)
- if $f(u, v) > 0$, then $(v, u) \in E_f$ and $c_f(v, u) = f(u, v)$ (backward edges).

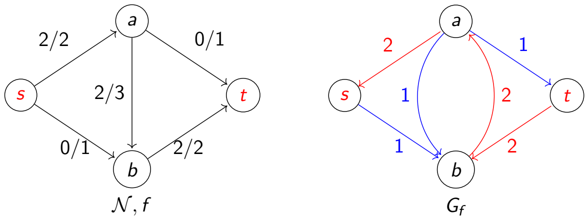



$\mathcal{N}, f$




$G_f$

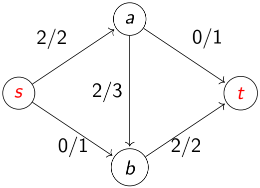
# Residual graph



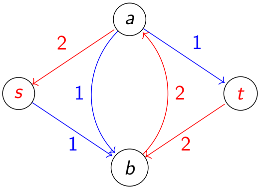

- Notice that, if $c(u, v) = f(u, v)$, then there is only a backward edge.
- $c_f$ are called the residual capacity.

# Residual graph






$\mathcal{N}, f$

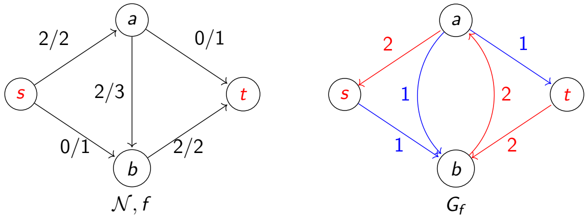



$G_f$

- forward edges: There remains capacity to push more flow through this edge.
- backward edges: there are units of flow that can be redirected through other links.

# Augmenting paths

Let $\mathcal{N} = (V, E, c, s, t)$ and let $f$ be a flow in $\mathcal{N}$,

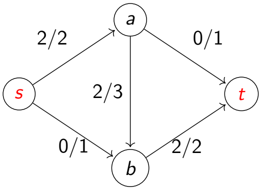



- An augmenting path $P$ is any simple path $P$ in $G_f$ from $s$ to $t$ $P$ might have forward and backward edges.
- For an augmenting path $P$ in $G_f$, the bottleneck, $b(P)$, is the minimum (residual) capacity of the edges in $P$.
  In the example, for $P = (s, b, a, t)$, $b(P) = 1$.

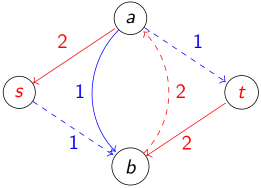
# Augmenting paths: increasing the flow



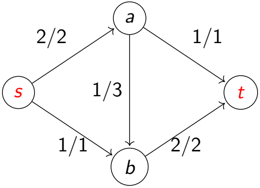
**Augment**$(P, f)$
b=bottleneck $(P)$
**for** each $(u, v) \in P$ **do**
  **if** $(u, v)$ is a forward edge **then**
    Increase $f(u, v)$ by $b$
  **else**
    Decrease $f(v, u)$ by $b$
**return** $f$




$\mathcal{N}, f$




$G_f,\ P = (s, a, t),\ b(P) = 1$




$\mathcal{N}, f'$

# Augmenting paths: increasing the flow



**Lemma**

*Let $f'$ =Augment$(P, f)$, then $f'$ is a flow in $G$.*

**Proof:** We have to prove the two flow properties.

- Capacity law
  - Forward edges $(u, v) \in P$, we increase $f(u, v)$ by $b$, as $b \leq c(u, v) - f(u, v)$ then $f'(u, v) = f(u, v) + b \leq c(u, v)$.
  - Backward edges $(u, v) \in P$ we decrease $f(v, u)$ by $b$, as $b \leq f(v, u), f'(v, u) = f(u, v) - b \geq 0$.

# Augmenting paths: increasing the flow



**Lemma**

*Let $f'$ =Augment$(P, f)$, then $f'$ is a flow in $G$.*

**Proof:** We have to prove the two flow properties.

- Conservation law, $\forall v \in P \setminus \{s, t\}$ let $u$ be the predecessor of $v$ in $P$ and let $w$ be its successor.
- As the path is simple only the alterations due to $(u, v)$ and $(v, w)$ can change the flow that goes trough $v$. We have four cases:
  - $(u, v)$ and $(v, w)$ are backward edges, the flow in $(v, u)$ and $(w, v)$ is decremented by $b$. As one is incoming and the other outgoing the total balance is 0.
  - $(u, v)$ and $(v, w)$ are forward edges, the flow in $(u, v)$ and $(v, w)$ is incremented by $b$. As one is incoming and the other outgoing the total balance is 0.

# Augmenting paths: increasing the flow



## Lemma

*Let $f'$ =Augment$(P, f)$, then $f'$ is a flow in $G$.*

**Proof:** We have to prove the two flow properties.

- Conservation law, $\forall v \in P \setminus \{s, t\}$ let $u$ be the predecessor of $v$ in $P$ and let $w$ be its successor.
- As the path is simple only the alterations due to $(u, v)$ and $(v, w)$ can change the flow that goes trough $v$. We have three cases:
  - $(u, v)$ is forward and $(v, w)$ is backward, the flow in $(u, v)$ is incremented by $b$ and the flow in $(w, v)$ is decremented by $b$. As both are incoming, the total balance is 0.
  - $(u, v)$ is backward and $(v, w)$ is forward, the flow in $(v, w)$ is incremented by $b$ and the flow in $(v, u)$ is decremented by $b$. As both are outgoing, the total balance is 0.

# Augmenting paths: incrementing the flow



**Lemma**

*Consider $f'$ =Augment$(P, f)$, then $|f'| > |f|$.*

Proof: Let $P$ be the augmenting path in $G_f$. The first edge $e \in P$ leaves $s$, and as $G$ has no incoming edges to $s$, $e$ is a forward edge. Moreover $P$ is simple $\Rightarrow$ never returns to $s$. Therefore, the value of the flow increases in edge $e$ by $b$ units. □

# Max-Flow Min-Cut theorem



Ford and Fulkerson (1954); Peter Elias, Amiel Feinstein and Claude Shannon (1956) (in framework of information-theory).

**Theorem**

*For any $\mathcal{N}(G, s, t, c)$, the maximum of the flow value is equal to the minimum of the $(S, T)$-cut capacities.*

$$\max_f \{|f|\} = \min_{(S,T)} \{c(S, T)\}.$$

# Max-Flow Min-Cut theorem:Proof

**Proof:**

- Let $f^*$ be a flow with maximum value, $|f^*| = \max_f\{|f|\}$
- For any $(s,t)$-cut $(S,T)$, $f^*(S,T) \leq c(S,T)$.
- $G_{f^*}$ has no augmenting path. So, if $S_s = \{v \in V | \exists s \rightsquigarrow v \text{ in } G_{f^*}\}$, then $(S_s, V - \{S_s\})$ is a $(s,t)$-cut.
- For $e = (u,v) \in E$ with $u \in S_s$ and $v \notin S_s$, $(u,v) \notin E(G_{f^*}$, therefore $f^*(u,v) = c(u,v)$,
- Then, $c(S_s, V - \{S_s\}) = f^*(S_s, V - \{S_s\}) = |f^*|$
- $(S_s, V - \{S_s\})$ is a minimum capacity $(s,t)$-cut in $G$.

□

1 Max Flow and Min Cut

2 Properties of flows and cuts

3 Residual graph

4 Augmenting path

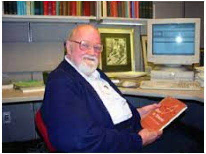
5 MaxFlow MinCut Thm

**6 Ford Fulkerson alg**

7 Maximum matching in Bip graphs

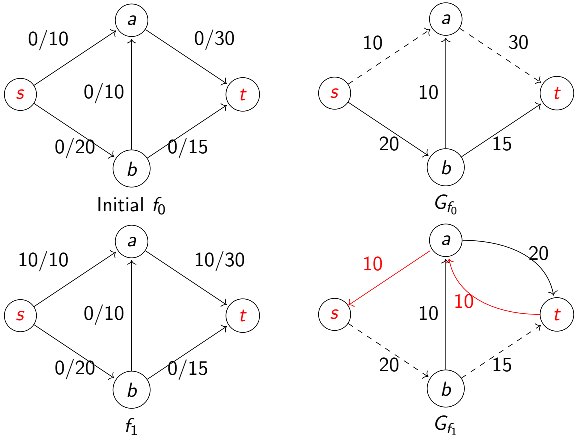
# Ford-Fulkerson algorithm

L.R. Ford, D.R. Fulkerson: *Maximal flow through a network*. Canadian J. of Math. 1956.

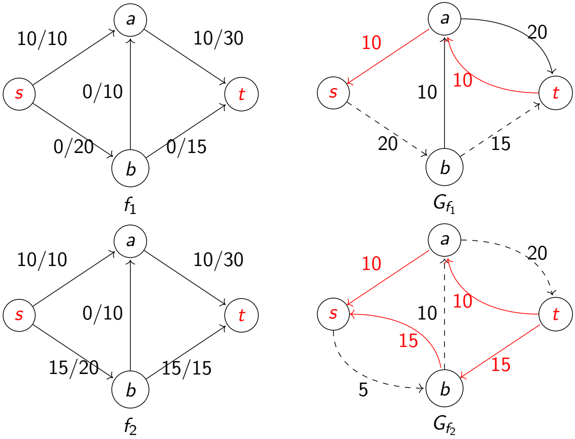


**Ford-Fulkerson**$(G, s, t, c)$
for all $(u, v) \in E$ set $f(u, v) = 0$
$G_f = G$
**while** there is an $(s, t)$ path $P$ in $G_f$ **do**
  $f = \text{Augment}(P, G_f)$
  Compute $G_f$
**return** $f$

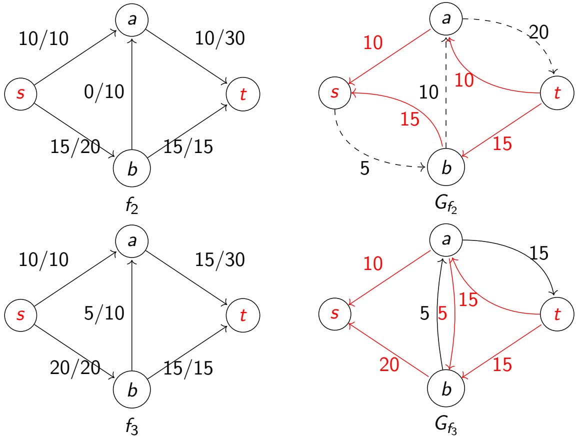
# FF algorithm example



# FF example

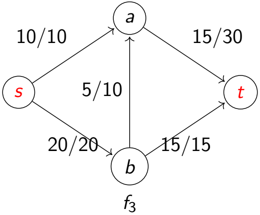

a
s
t
b
10/10
10/30
0/10
0/20
0/15
$f_1$
10
20
10
10
20
15
$G_{f_1}$
10/10
10/30
0/10
15/20
15/15
$f_2$
10
20
10
10
15
15
5
$G_{f_2}$

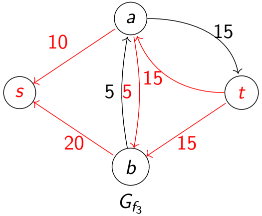
# FF example


a
10/10
10/30
s
0/10
t
15/20
15/15
b
$f_2$
a
10
20
s
10
10
t
15
15
5
b
$G_{f_2}$
a
10/10
15/30
s
5/10
t
20/20
15/15
b
$f_3$
a
15
10
s
5
5
15
t
20
15
b
$G_{f_3}$

# FF example






Flow with max value

$\{s\}, \{a, b, t\}$ is a min $(s, t)$-cut

# Correctness of Ford-Fulkerson



Consequence of the Max-flow min-cut theorem.

**Theorem**

*The flow returned by Ford-Fulkerson is the max-flow.*

# Networks with integer capacities

**Lemma (Integrality invariant)**

*Let $\mathcal{N} = (V, E, c, s, t)$ where $c : E \to \mathbb{Z}^+$. At every iteration of the Ford-Fulkerson algorithm, the flow values $f(e)$ are integers.*

Proof: (induction)

- The statement is true for the initial flow (all zeroes).
- Inductive Hypothesis: The statement is true after $j$ iterations.
- At iteration $j+1$: As all residual capacities in $G_f$ are integers, then bottleneck $(P, f) \in \mathbb{Z}$, for the augmenting path found in iteration $j+1$.
- Thus the augmented flow values are integers. □

# Networks with integer capacities

**Theorem (Integrality theorem)**

*Let $\mathcal{N} = (V, E, c, s, t)$ where $c : E \to \mathbb{Z}^+$. There exists a max-flow $f^*$ such that $f^*(e)$ is an integer, for any $e \in E$.*

Proof:
Since the algorithm terminates, the theorem follows from the integrality invariant lemma. □

# Networks with integer capacities: FF running time

**Lemma**

*Let $C$ be the min cut capacity (=max. flow value), Ford-Fulkerson terminates after finding at most $C$ augmenting paths.*

Proof: The value of the flow increases by $\geq 1$ after each augmentation. □

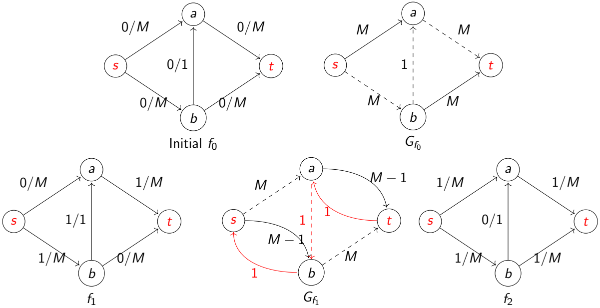
# Networks with integer capacities: FF running time

- The number of iterations is $\leq C$. At each iteration:
- Constructing $G_f$, with $E(G_f) \leq 2m$, takes $O(m)$ time.
- $O(n+m)$ time to find an augmenting path, or deciding that it does not exist.
- Total running time is $O(C(n+m)) = O(Cm)$
- Is that polynomic? No, only pseudopolynomic

# Networks with integer capacities: FF running time



The number of iterations of Ford-Fulkerson could be $\Theta(C)$



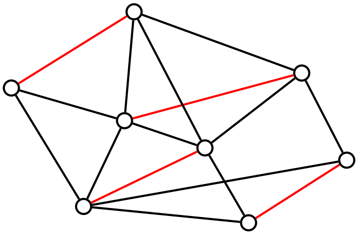

Ford-Fulkerson can alternate between the two long paths, and require $2M$ iterations. Taking $M = 10^{10}$, FF on a graph with 4 vertices can take time $210^{10}$.

# MAXIMUM MATCHING problem



Given an undirected graph $G = (V, E)$ a subset of edges $M \subseteq E$ is a matching if each node appears at most in one edge in $M$ (a node may not appear at all).

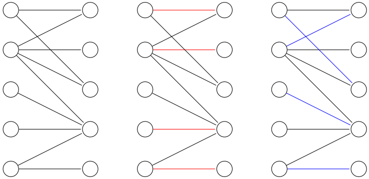
MAXIMUM MATCHING problem:
Given a graph $G$, find a matching with maximum cardinality.



# Maximum matching in bipartite graphs

A graph $G = (V, E)$ is bipartite if there is a partition of $V$ in $L$ and $R$, ($L \cup R = V$ and $L \cap R = \emptyset$), such that every $e \in E$ connects a vertex in $L$ with a vertex in $R$.

We want to solve the MAXIMUM MATCHING problem on bipartite graphs

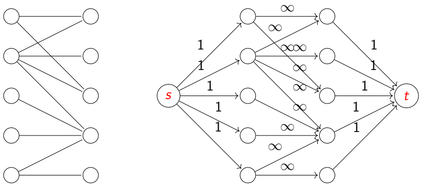


# Maximum Matching: Network formulation



From $G = (L \cup R, E)$ construct $\mathcal{N} = (\hat{V}, \hat{E}, c, s, t)$:

- Add vertices $s$ and $t$: $\hat{V} = L \cup R \cup \{s, t\}$.
- Add directed edges $s \to L$ with capacity 1. Add directed edges $R \to t$ with capacity 1.
- Direct the edges $E$ from $L$ to $R$, and give them capacity $\infty$.
- $\hat{E} = \{s \to L\} \cup E \cup \{R \to t\}$.



# Maximum matching algorithm: Analysis



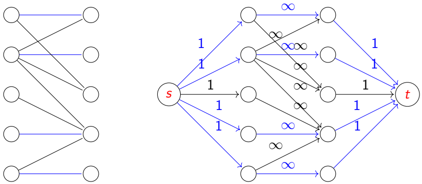
**Theorem**

*Max flow in $\mathcal{N}$=Max bipartite matching in G.*

**Proof** Matching as flows

Let $M$ be a matching in $G$ with $k$-edges, consider the flow $f$ that sends 1 unit along each one of the $k$ paths, $s \to u \to v \to t$, for $(u,v) \in M$.

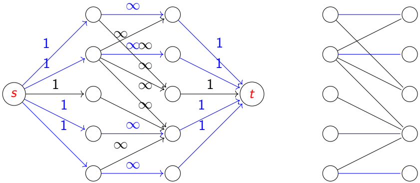
As $M$ is a matching all these paths are disjoint, so $f$ is a flow and has value $k$.



# Maximum matching algorithm: Analysis

## Flows as matchings

- Consider an integral flow $f$ in $\hat{G}$. Therefore, for any edge $e$, the flow is either 0 or 1.
- Consider the cut $C = (\{s\} \cup L, R \cup \{t\})$ in $\hat{G}$.
- Let $M$ be the set of edges in the cut $C$ with flow=1, then $|M| = |f|$.
- Each node in $L$ is in at most one $e \in M$ and every node in $R$ is in at most one head of an $e \in F$
- Therefore, $M$ is a matching in $G$ with $|M| \leq |f|$



# Maximum matching algorithm: Analysis



As $\mathcal{N}$ has integer capacities there is an integral maximum flow $f^*$, the associated matching is a maximum matching. □

# Maximum matching algorithm: Analysis



What is the cost of the algorithm?

- The bipartite graph, has $n$ vertices and $m$ edges. The capacities are integers. We need an integral solution.
- The algorithm: (1) constructs $\mathcal{N}$, (2) runs FF on $\mathcal{N}$ to obtain a maxflow $f$, (3) from $f$ obtain a maximum matching $M$.
- $\mathcal{N}$ has $n+2$ vertices and $m+2n$ edge, (1) takes $O(n+m)$
- The maximum value of a flow in $\mathcal{N}$ is at most $n$, (2) takes time $O(|f|(n+m)) = O(n(n+m))$
- (3) can be done in time $O(n+m)$.

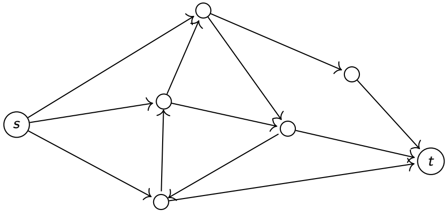
So, the cost is $O(n(n+m))$.

# Disjoint Path problem



Given a digraph $G = (V, E)$ and two vertices $s, t \in V$, a set of paths is edge-disjoint if their edges are disjoint (although they might share some vertex)

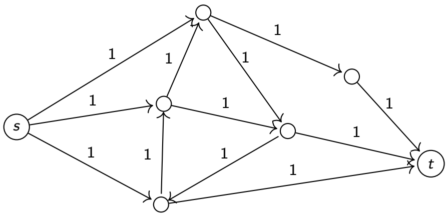
Disjoint Path problem: Given a digraph $G = (V, E)$ and two vertices $s, t \in V$, find a set of $s \rightsquigarrow t$ edge-disjoint paths of maximum cardinality



# Disjoint Path: Max flow formulation



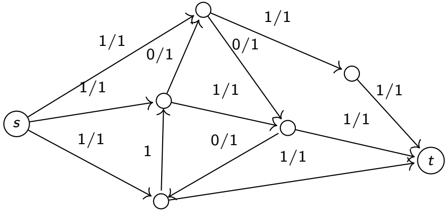
Thinking in terms of flow a path from $s$ to $t$ can be seen as a way of transporting a unit of flow.
We construct a network $\mathcal{N}$ assigning unit capacity to every edge.



# Disjoint Path: Max flow formulation



Thinking in terms of flow a path from $s$ to $t$ can be seen as a way of transporting a unit of flow.
We construct a network $\mathcal{N}$ assigning unit capacity to every edge
We solve MaxFlow for $\mathcal{N}$.



# Disjoint Path: Max flow formulation



Thinking in terms of flow a path from $s$ to $t$ can be seen as a way of transporting a unit of flow.
We construct a network $\mathcal{N}$ assigning unit capacity to every edge

**Theorem**

*The max number of edge disjoint paths $s \rightsquigarrow t$ is equal to the max flow value*

# Disjoint Path: Proof of the Theorem



**Proof.**

Number of disjoints paths $\leq$ max flow
If we have $k$ edge-disjoints paths $s \rightsquigarrow t$ in $G$ then making $f(e) = 1$ for each $e$ in a path, we get a flow with $|f| = k$

# Disjoint Path: Proof of the Theorem



Number of disjoints paths $\geq$ max flow

- If the max flow value is $k$, there exists a 0-1 flow $f^*$ with value $k$.
- Consider the graph $G^* = (V, E')$ where $E'$ is formed by all edges $e$ with $f(e) = 1$.
- We repeatedly compute a $s \rightsquigarrow t$ simple path in $G^*$, and remove its edges from $G^*$.
- Each time that we remove a path, the value of the flow in the network is reduced by one, so we can apply the process $k$ times.
- None of the paths share an edge, so we get $k$ disjoint paths. $\square$

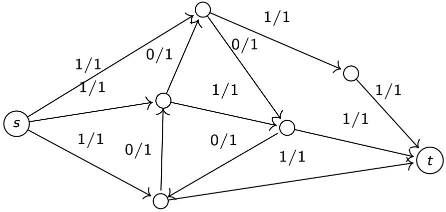
End Proof

# Disjoint Path: Max flow + path extraction algorithm



Algorithm

1. Construct the network $\mathcal{N}$ assigning unit capacity to every edge
2. Solve MaxFlow for $\mathcal{N}$
3. Extract the set of disjoint paths on the graph restricted to edges with flow $> 0$

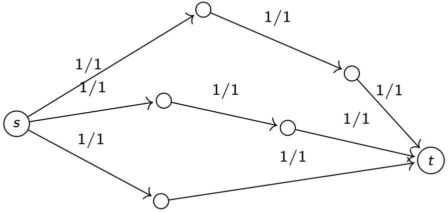


# Disjoint Path: Max flow + path extraction algorithm



Algorithm

1. Construct the network $\mathcal{N}$ assigning unit capacity to every edge
2. Solve MaxFlow for $\mathcal{N}$
3. Extract the set of disjoint paths on the graph restricted to edges with flow $> 0$



# Disjoint paths algorithm: Analysis



What is the cost of the algorithm?

- The graph, has $n$ vertices and $m$ edges. The capacities are integers. We need an integral solution.
- The algorithm: (1) constructs $\mathcal{N}$, (2) runs FF on $\mathcal{N}$ to obtain a max flow $f$, (3) from $f$ obtains $|f|$ edge disjoint paths.
- $\mathcal{N}$ has $n$ vertices and $m$ edges, (1) takes $O(n+m)$
- The maximum value of a flow in $\mathcal{N}$ is at most $n$, (2) takes time $O(|f|(n+m)) = O(n(n+m))$
- (3) can be done in time $O(n+m)$ per path, i.e., $O(|f|(n+m))$.

So the cost is $O(n(n+m))$.

# Vertex disjoint paths



Can we do something similar to get the maximum number of vertex disjoint paths?

# The case of undirected graphs



If we have an undirected graph, with two distinguised nodes $u, v$, how would you apply the max flow formulation to solve the problem of finding the max number of disjoint paths between $u$ and $v$?

# The case of undirected graphs