Lecture 15

Network flow

Chinmay Nirkhe | CSE 421 Spring 2025

Midterm

- Midterm during class on May 5th in the usual lecture hall
- For with registered services with DRS for alternate testing
  - Glerum Room CSE2 345 starting at 3:30
  - Your responsibility to convey to the proctor your specific alterations
- You are allowed to bring XXXX resources with you. Pen and paper exam.
- Midterm will be 1 hour and starts promptly.

Midterm

- Covers subjects up through the dynamic programming except Bellman-Ford
- Sample midterm for practice problems and length is posted
- Section this week will review problems and strategy
- I’ll host a Q&A section about the subject on XXXX.

Communication disruption

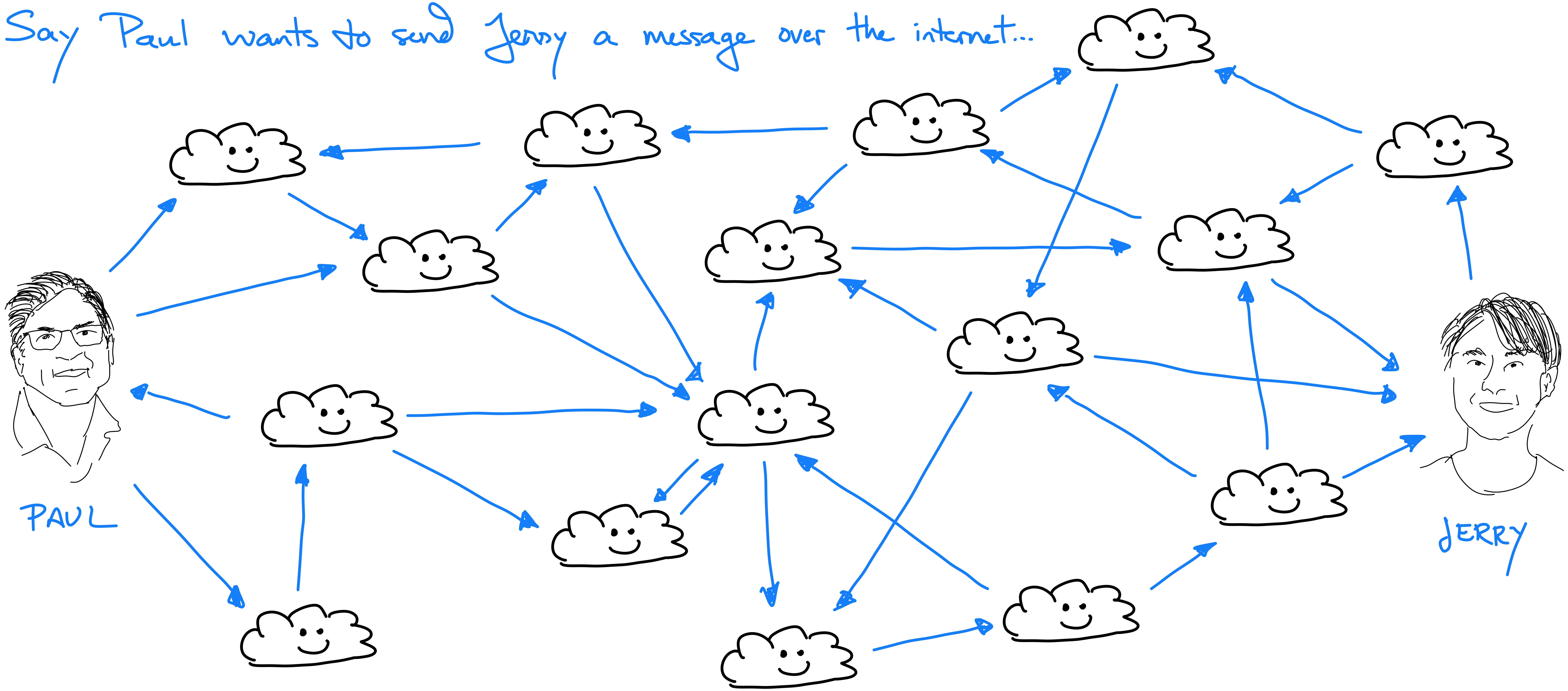
Say Paul wants to send Jerry a message over the internet...

PAUL

JERRY

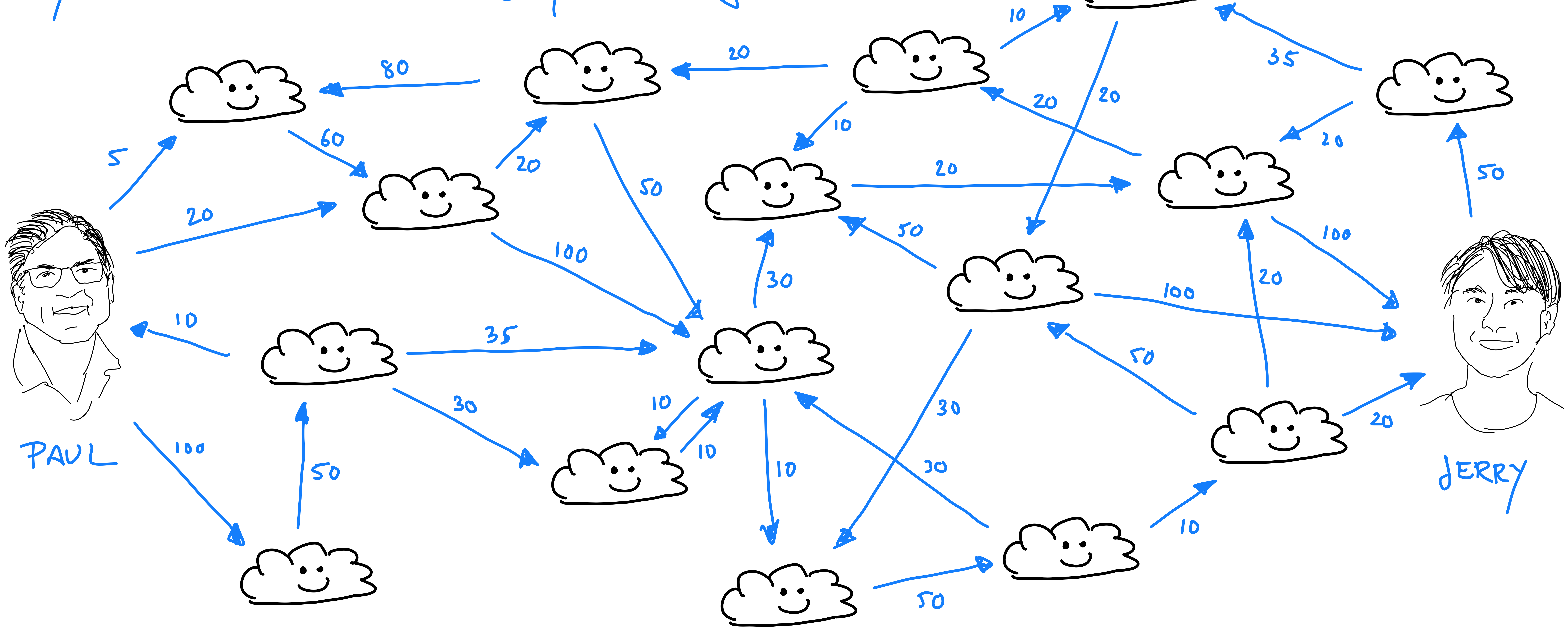
Communication disruption

Say Paul wants to send Jerry a message over the internet...



Communication disruption

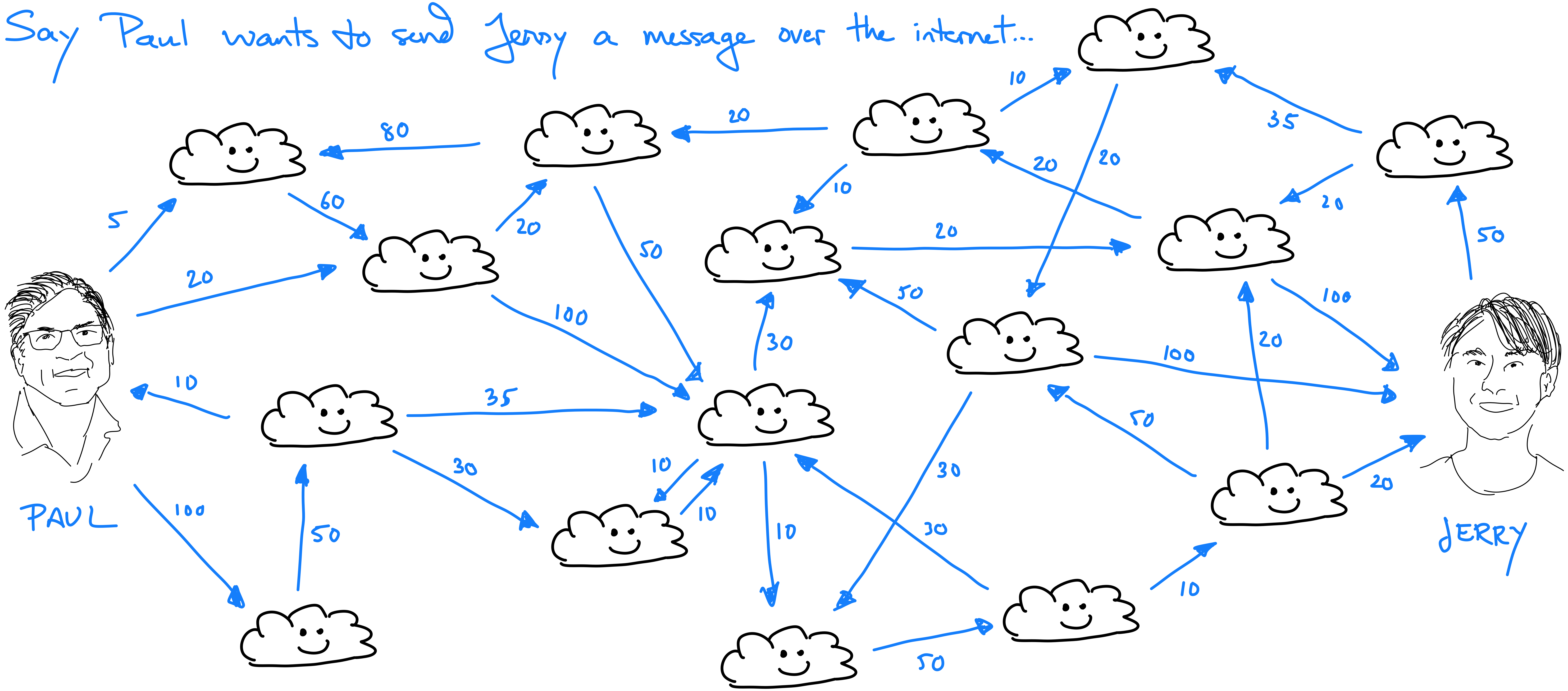
Say Paul wants to send Jerry a message over the internet...



Communication disruption

How many wires would an adversary have to cut to prevent transmission?

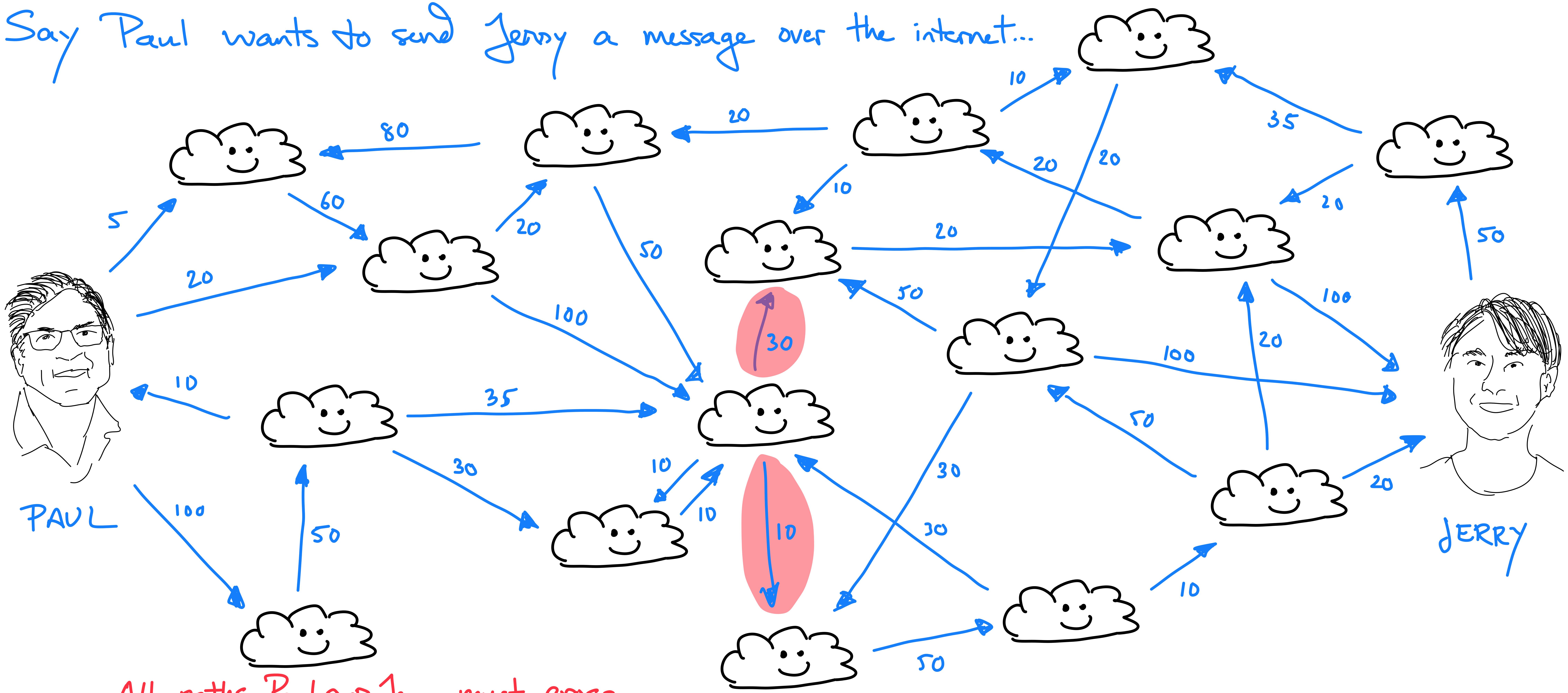
Say Paul wants to send Jerry a message over the internet...



Communication disruption

How many wires would an adversary have to cut to prevent transmission?

Say Paul wants to send Jerry a message over the internet...

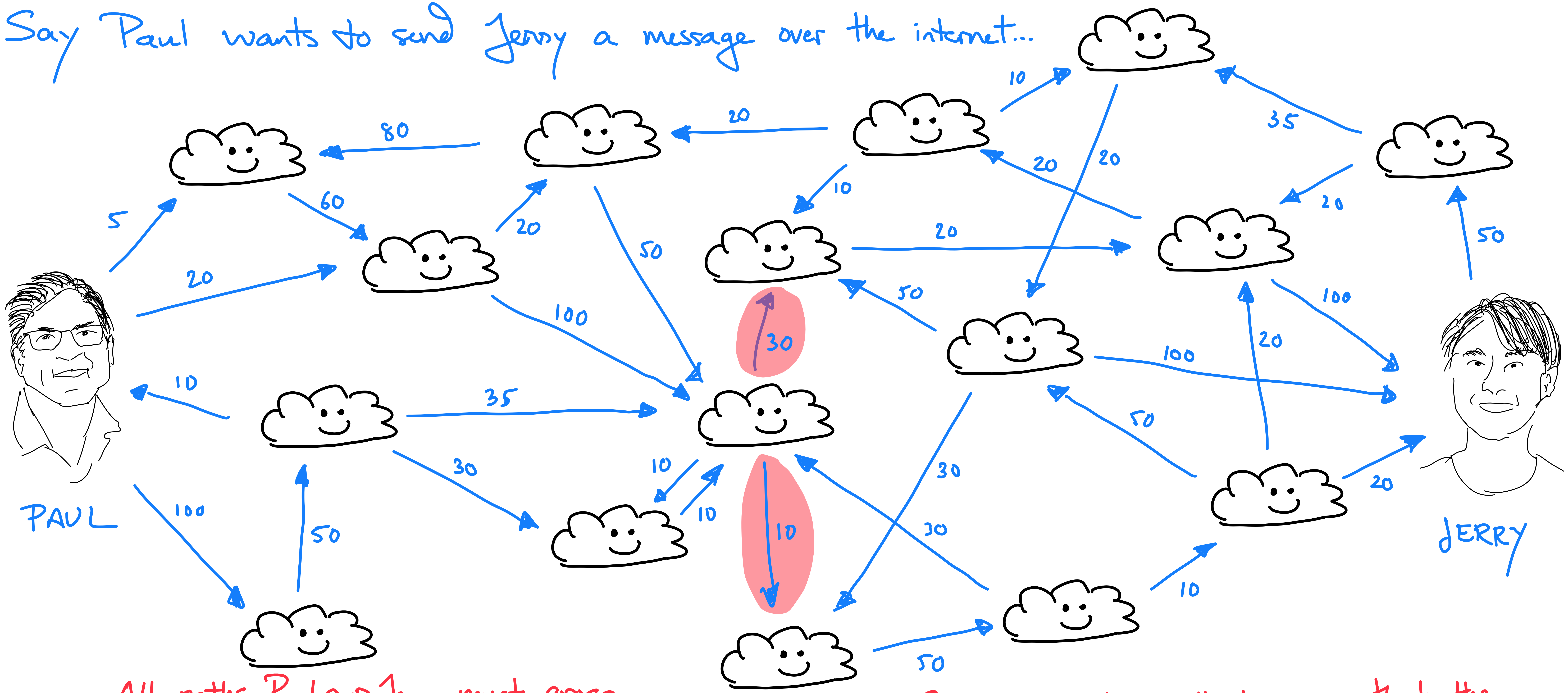


All paths Paul ⇝ Jerry must cross one of these two edges.

Communication disruption

How many wires would an adversary have to cut to prevent transmission?

Say Paul wants to send Jerry a message over the internet...



All paths Paul ⇝ Jerry must cross one of these two edges.

Consequently, this will also prove that the message rate from Paul ⇝ Jerry is at most 40.

Maximum flow and minimum cut

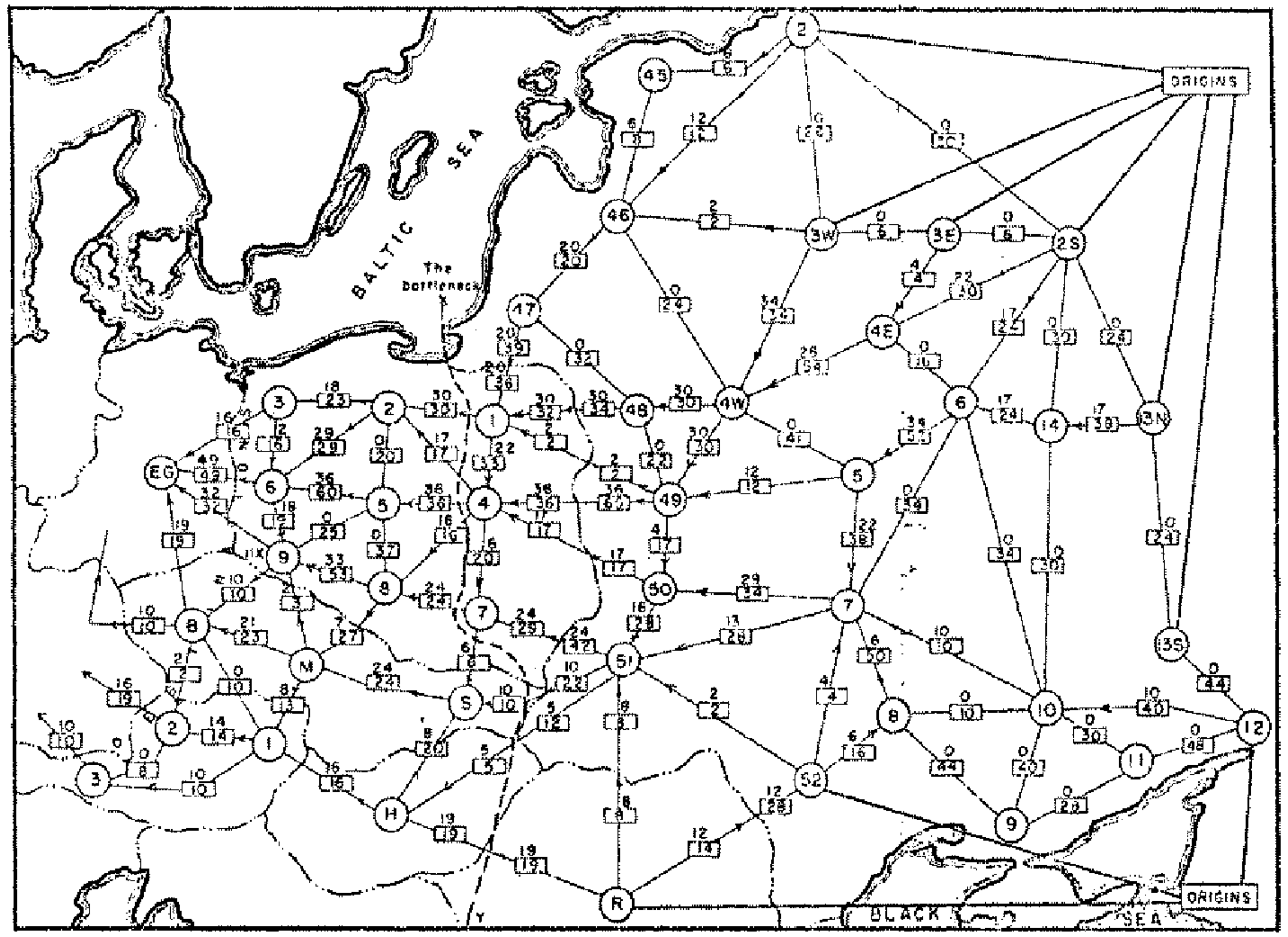
- Two very rich algorithmic problems.
- Cornerstone problems in combinatorial optimization.
- Beautiful mathematical duality.

Nontrivial applications / reductions:

- Data mining.
- Project selection.
- Airline scheduling.
- Bipartite matching.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Strip mining.
- Network reliability.
- Distributed computing.
- Egalitarian stable matching.
- Security of statistical data.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- many many more …

The origin of the max flow/min cut problems

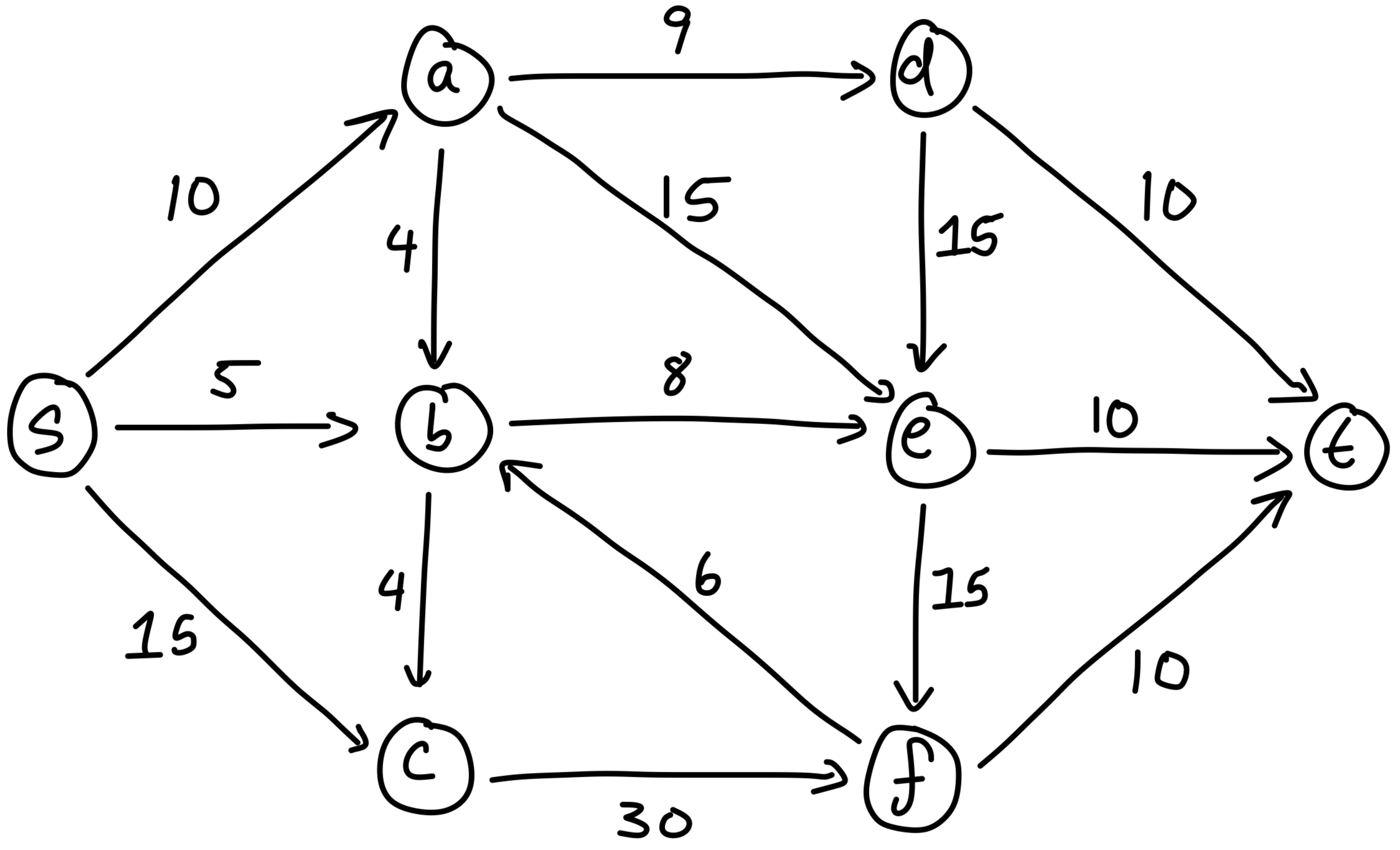
- **Max Flow problem:** Rail transportation for the Soviet Union
- **Min Cut problem:** Cold War attempts to cripple Soviet supply routes
- Ford & Fullerton prove (1955) that problems are equivalent



Reference: *On the history of the transportation and maximum flow problems*.
Alexander Schrijver in Math Programming, 91: 3, 2002.

Flow network definition

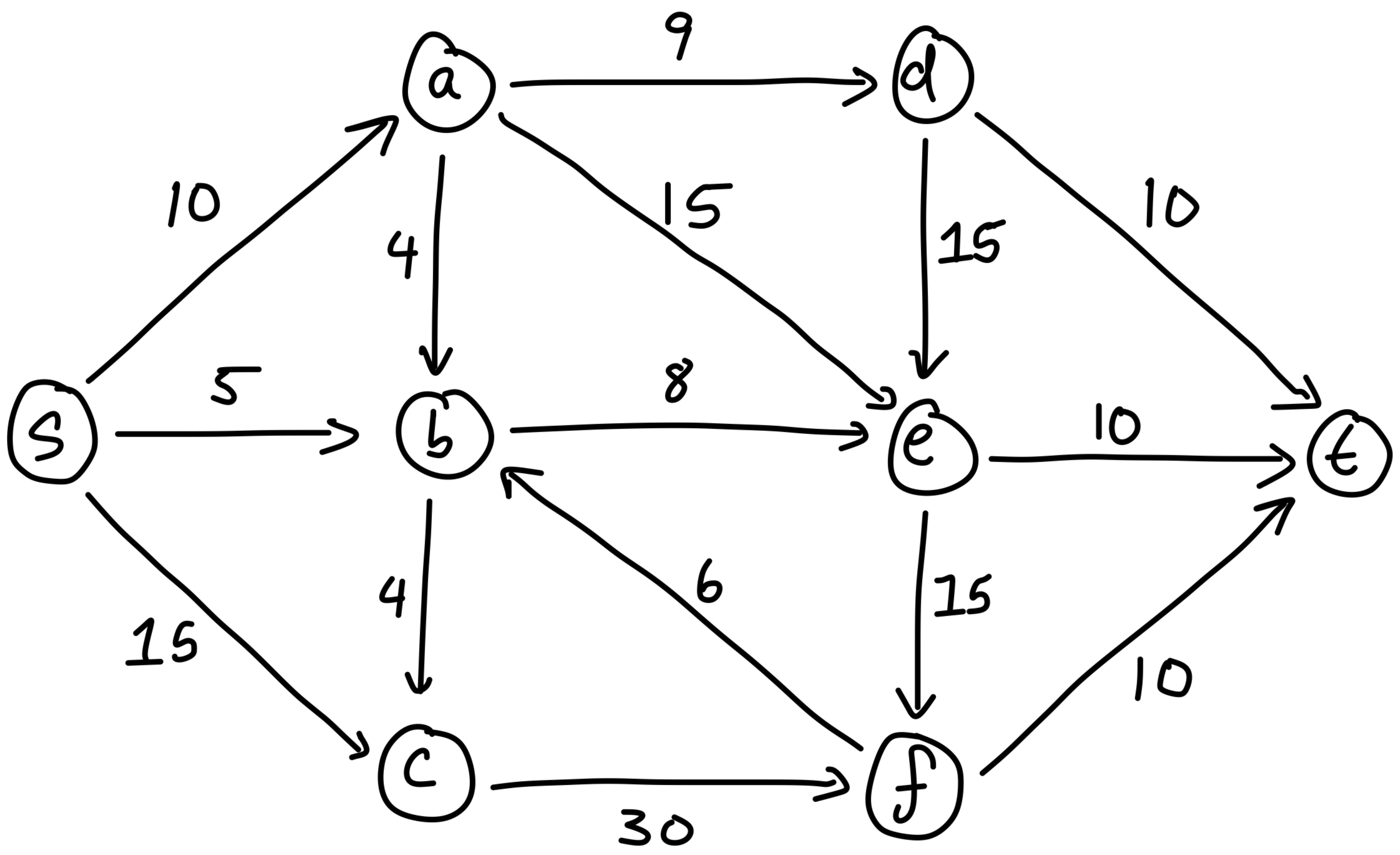
- Imagine each edge in a graph is a directional water pipe
- Each edge has a capacity $c(e)$ for $c : E \to \mathbb{R}_{\geq 0}$
- There are two specified vertices s, t for source and sink
- $G = (V, E)$ graph with no parallel edges
- The tuple (G, c, s, t) define a **flow network**



Graph cuts

- An **s-t cut** in a graph is a partition of the vertices into $V = S \sqcup T$ such that $s \in S$ and $t \in T$. The capacity of a s-t cut (S, T) is

$$c(S,T) := \sum_{\substack{\text{edges } e \text{ leaving} \\ S}} c(e)$$

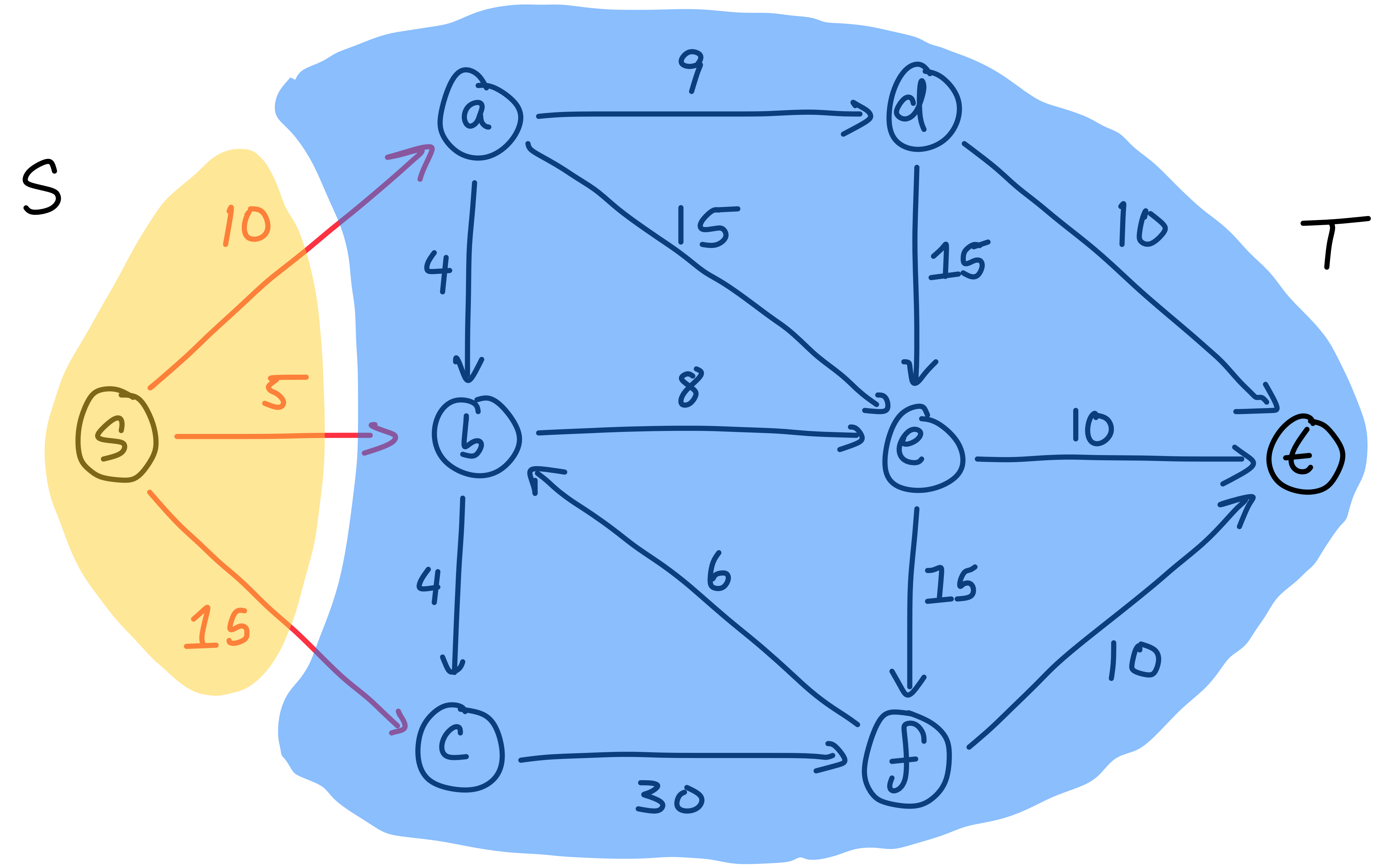


Graph cuts

- An **s-t cut** in a graph is a partition of the vertices into $V = S \sqcup T$ such that $s \in S$ and $t \in T$. The capacity of a s-t cut (S, T) is

$$c(S,T) := \sum_{\substack{\text{edges } e \text{ leaving} \\ S}} c(e)$$

$$c(S,T) = 10 + 5 + 15 = 30$$

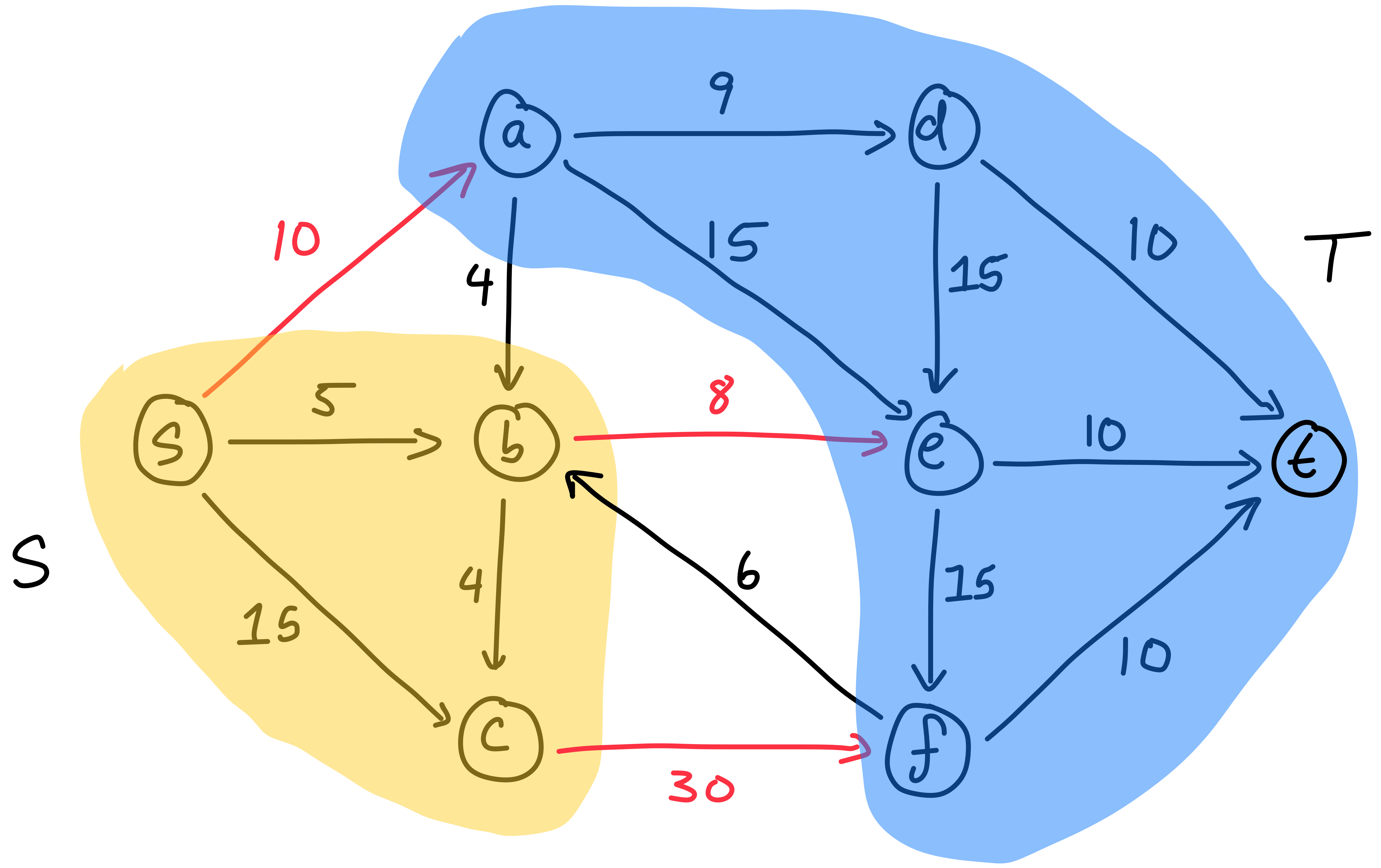


Graph cuts

- An **s-t cut** in a graph is a partition of the vertices into $V = S \sqcup T$ such that $s \in S$ and $t \in T$. The capacity of a s-t cut (S, T) is

$$c(S,T) := \sum_{\substack{\text{edges } e \text{ leaving} \\ S}} c(e)$$

$$c(S,T) = 10 + 8 + 30 = 48$$

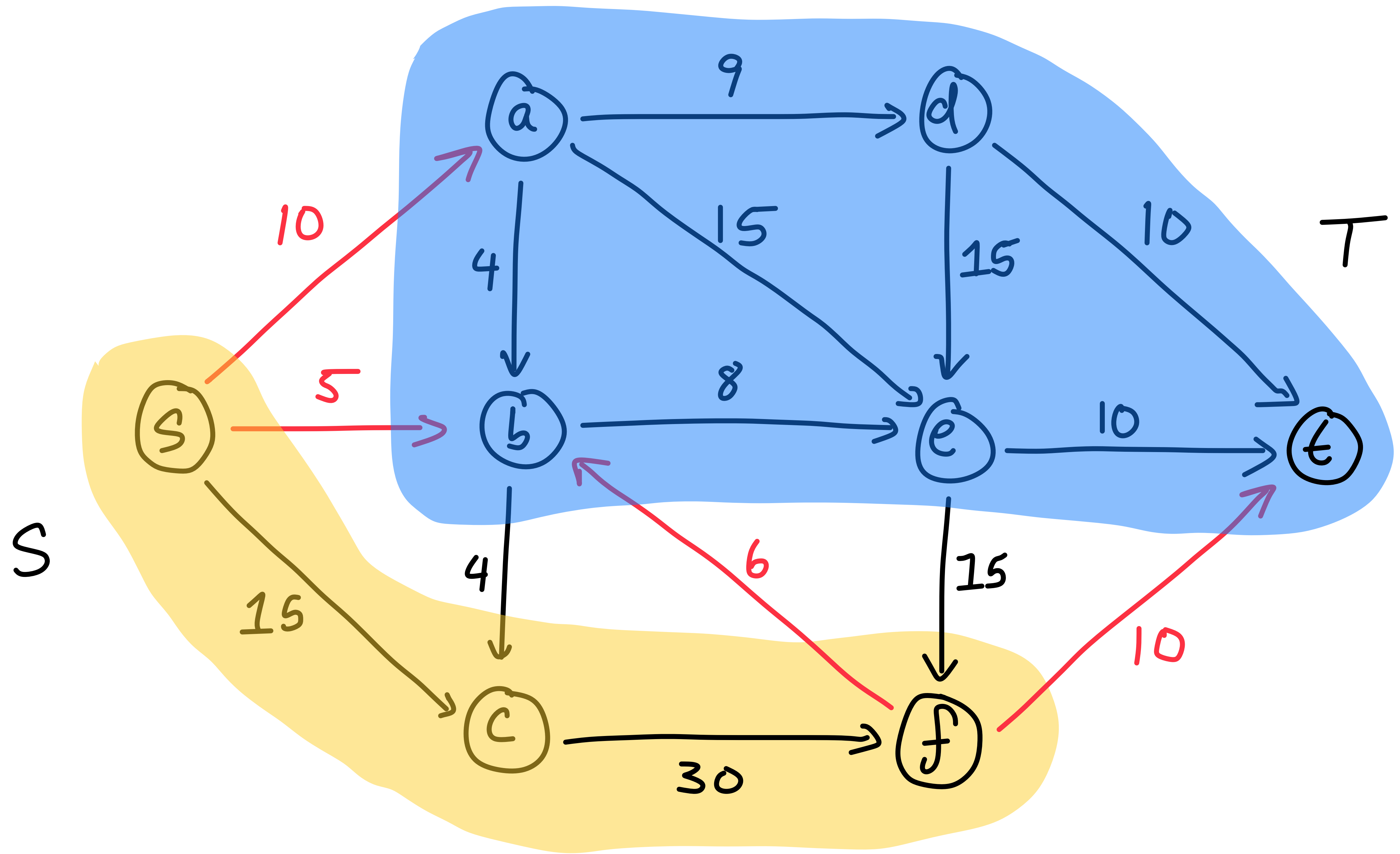


Graph cuts

- An **s-t cut** in a graph is a partition of the vertices into $V = S \sqcup T$ such that $s \in S$ and $t \in T$. The capacity of a s-t cut (S, T) is

$$c(S,T) := \sum_{\substack{\text{edges } e \text{ leaving} \\ S}} c(e)$$

$$c(S,T) = 10 + 5 + 6 + 10 = 21$$

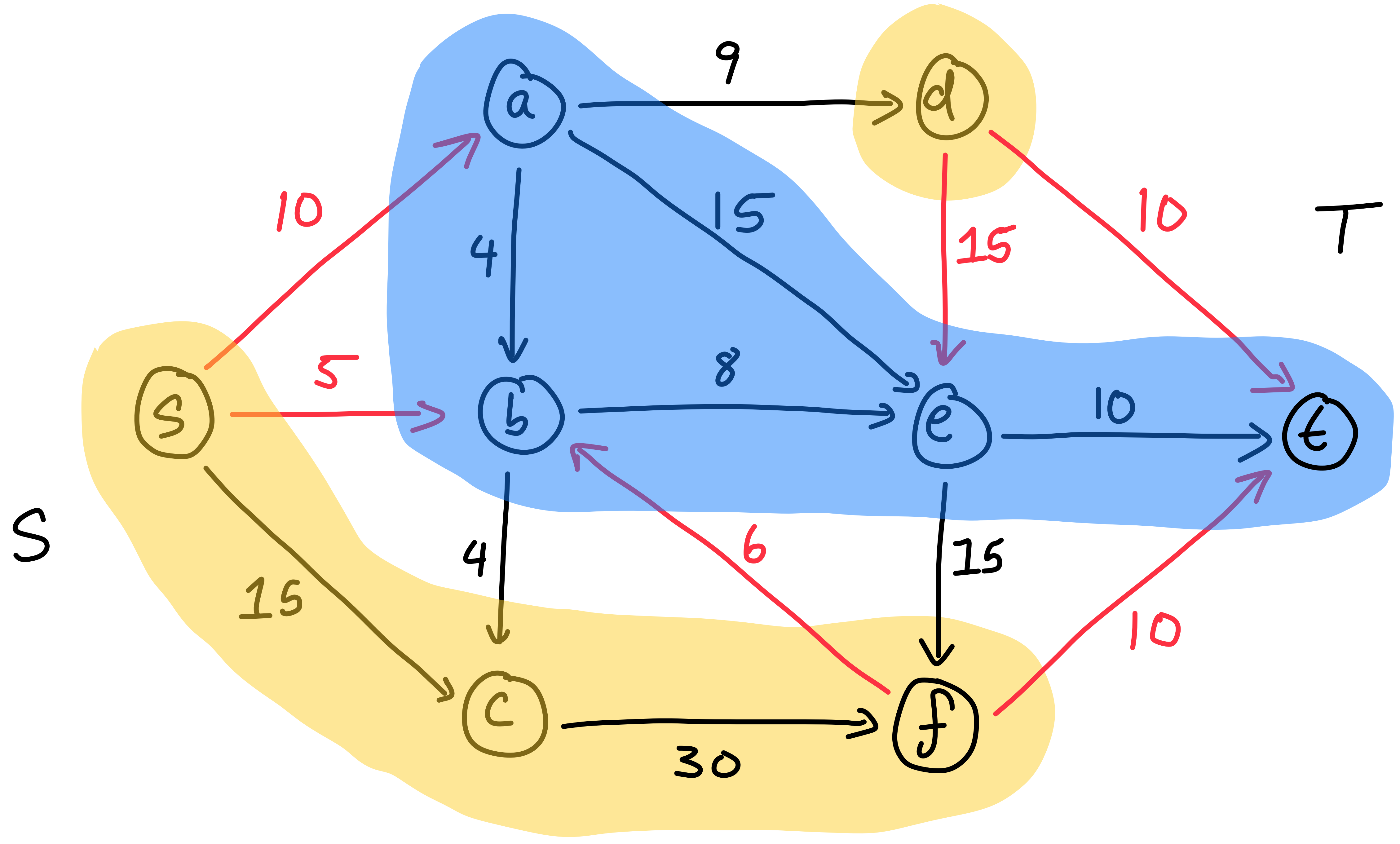


Graph cuts

- An **s-t cut** in a graph is a partition of the vertices into $V = S \sqcup T$ such that $s \in S$ and $t \in T$. The capacity of a s-t cut (S, T) is

$$c(S,T) := \sum_{\substack{\text{edges } e \text{ leaving} \\ S}} c(e)$$

$$c(S,T) = 10 + 5 + 6 + 10 + 10 + 15 = 46$$

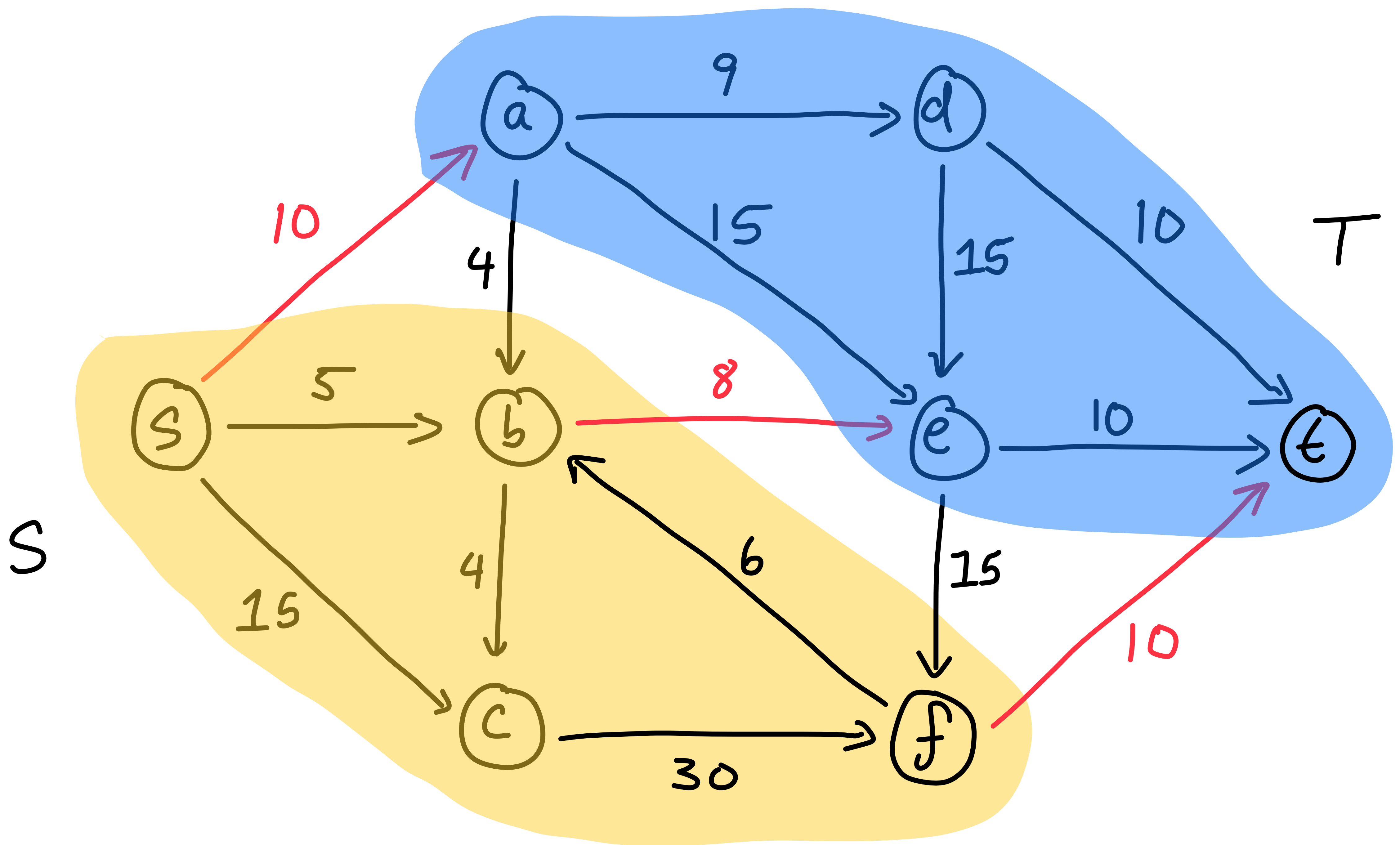


The minimum cut problem

- **Input:** a flow network (G, c, s, t)
- **Output:** a s-t cut of minimum capacity

$$\mathrm{mincut}(G, c, s, t) = \min_{\substack{\text{s-t cut} \\ (S,T)}} \left\{ c(S,T) \right\}$$

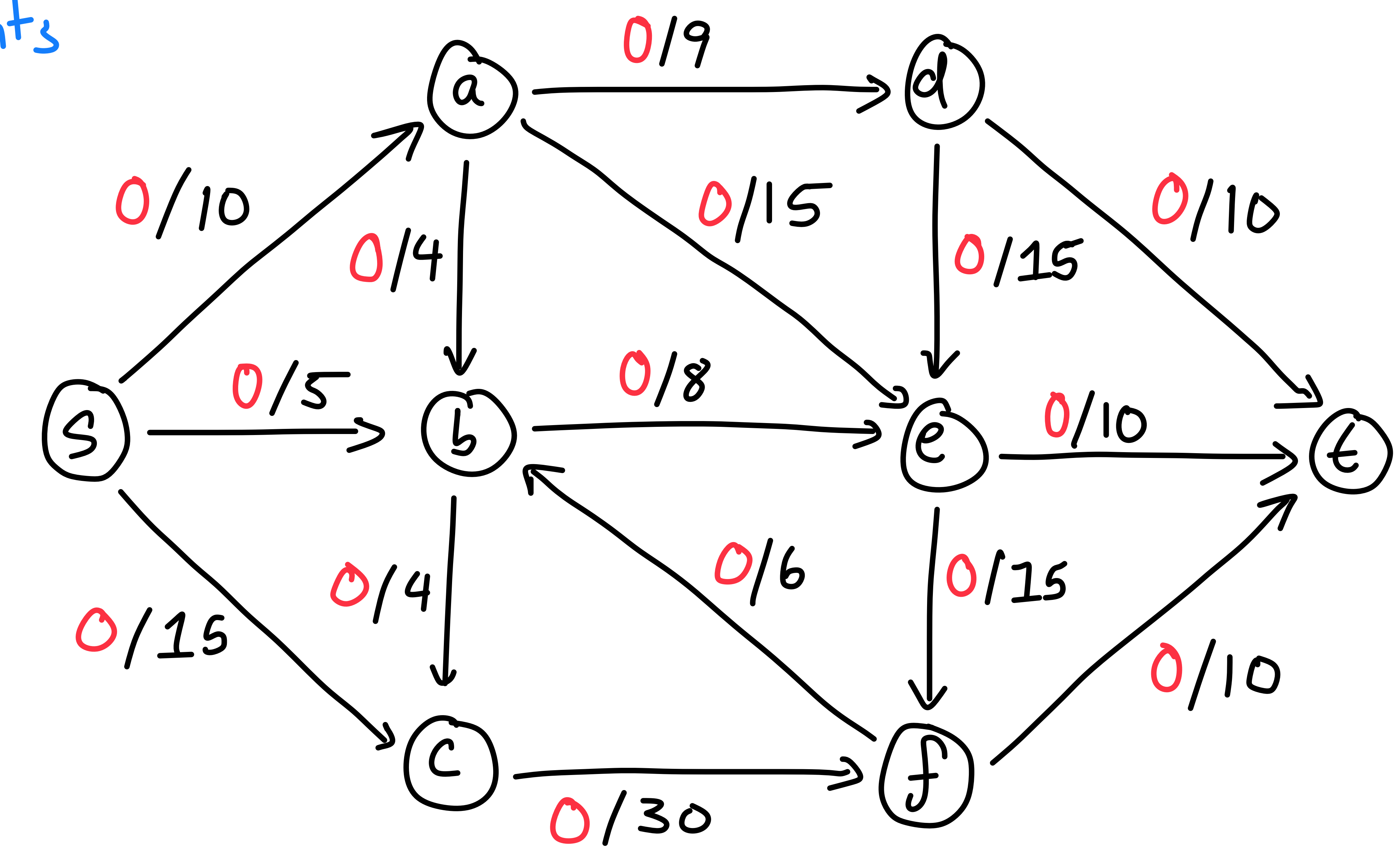
in this case, mincut = 28



s-t flow

- A **s-t flow** in a flow network is a fn. $f : E \to \mathbb{R}_{\geq 0}$ that satisfies:
  - For each edge $e \in E$, $0 \leq f(e) \leq c(e)$ capacity constraints
  - For every $v \in V \backslash \{s, t\}$, conservation of flow
$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ outof } v} f(e).$$
- The **value** of a flow f is $v(f) := \sum_{e \text{ outof} s} f(e)$

trivial flow $f \equiv 0$.



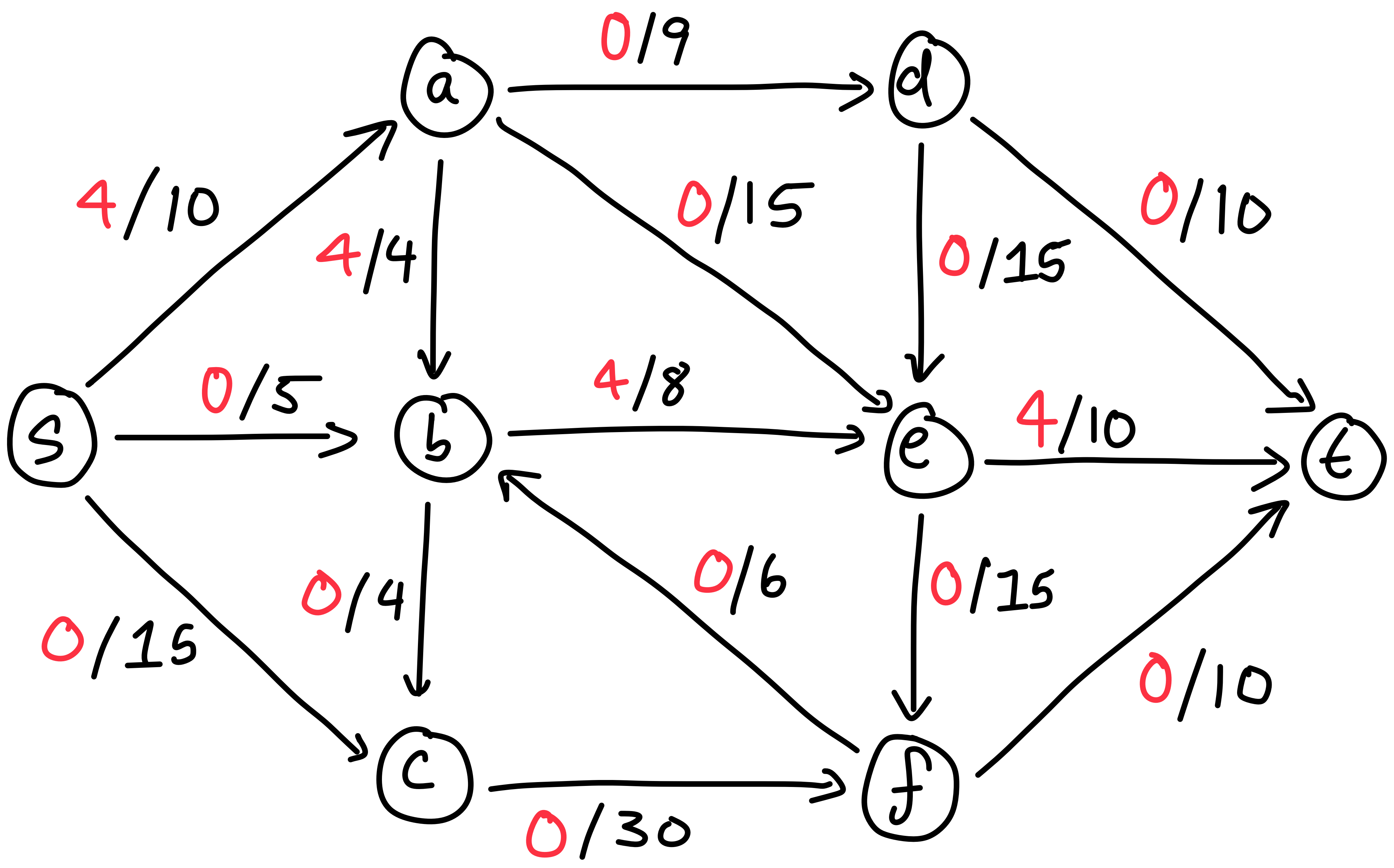
s-t flow

- A **s-t flow** in a flow network is a fn. $f : E \to \mathbb{R}_{\geq 0}$ that satisfies:
  - For each edge $e \in E$, $0 \leq f(e) \leq c(e)$
  - For every $v \in V \backslash \{s, t\}$,

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ outof } v} f(e).$$

- The **value** of a flow f is $v(f) := \sum_{e \text{ outof} s} f(e)$

flow of value 4.

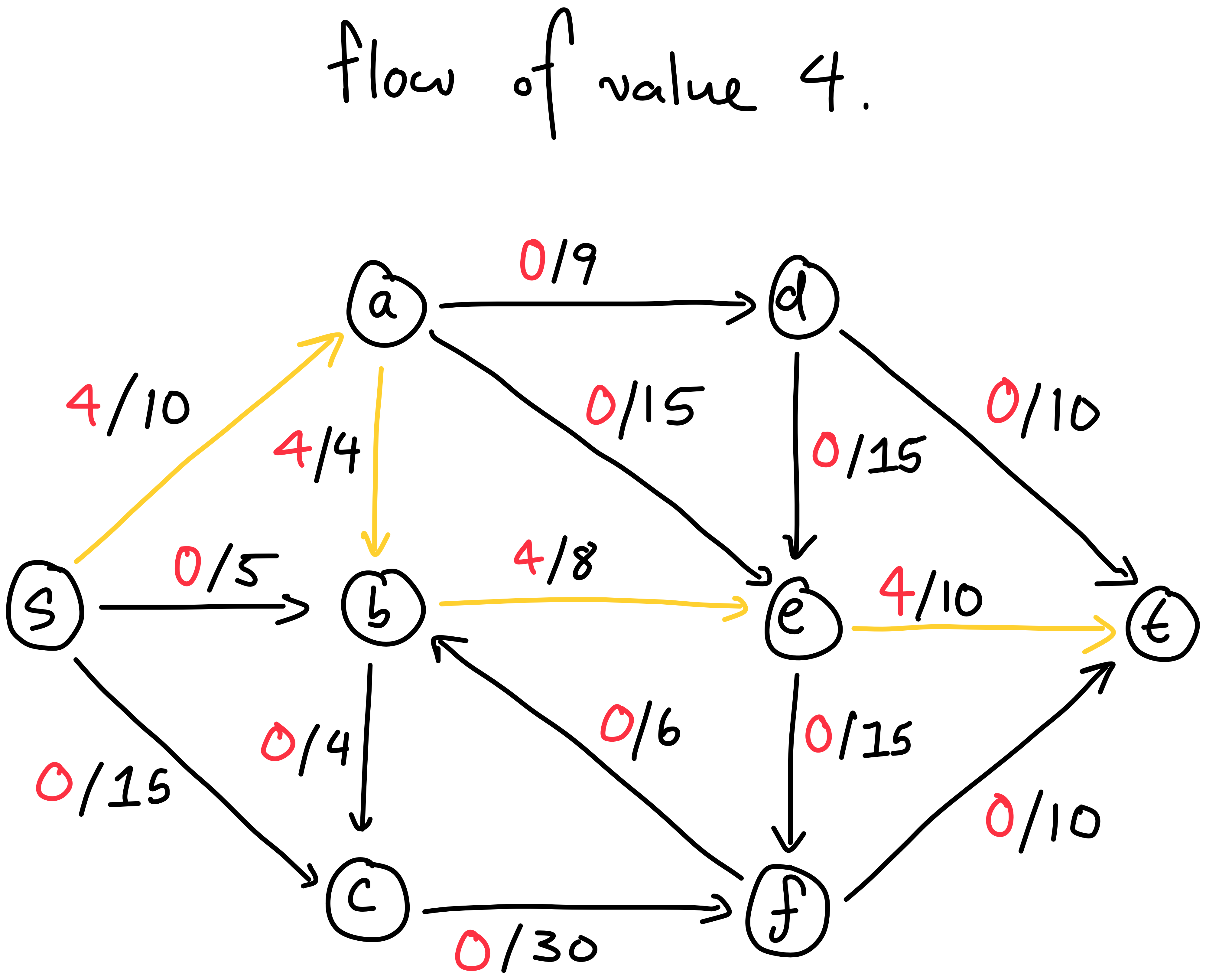


s-t flow

- A **s-t flow** in a flow network is a fn. $f : E \to \mathbb{R}_{\geq 0}$ that satisfies:
  - For each edge $e \in E$, $0 \leq f(e) \leq c(e)$
  - For every $v \in V \backslash \{s, t\}$,

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ outof } v} f(e).$$

- The **value** of a flow f is $v(f) := \sum_{e \text{ outof} s} f(e)$



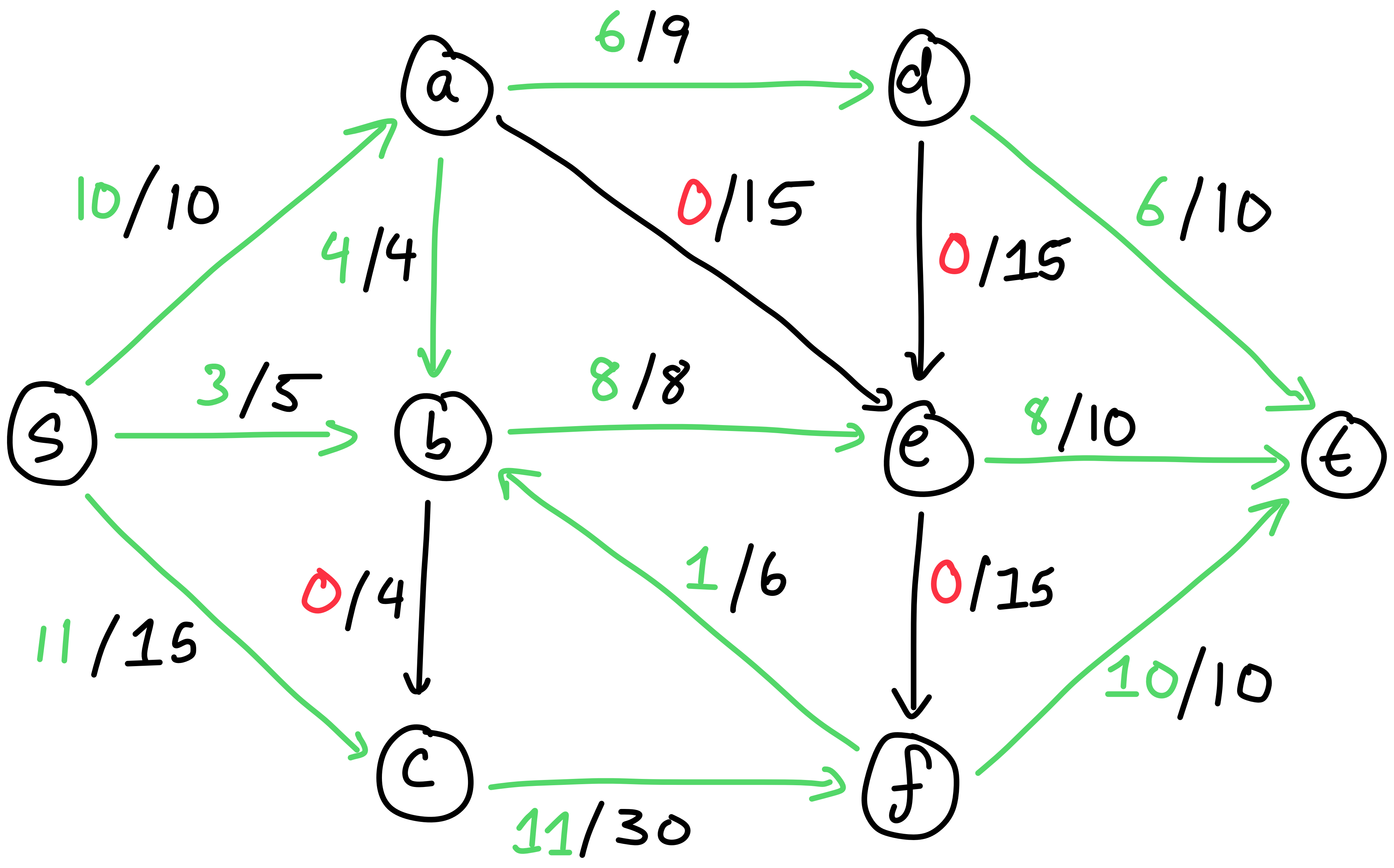
s-t flow

- A **s-t flow** in a flow network is a fn. $f : E \to \mathbb{R}_{\geq 0}$ that satisfies:
  - For each edge $e \in E$, $0 \leq f(e) \leq c(e)$
  - For every $v \in V \backslash \{s, t\}$,

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ outof } v} f(e).$$

- The **value** of a flow f is $v(f) := \sum_{e \text{ outof} s} f(e)$

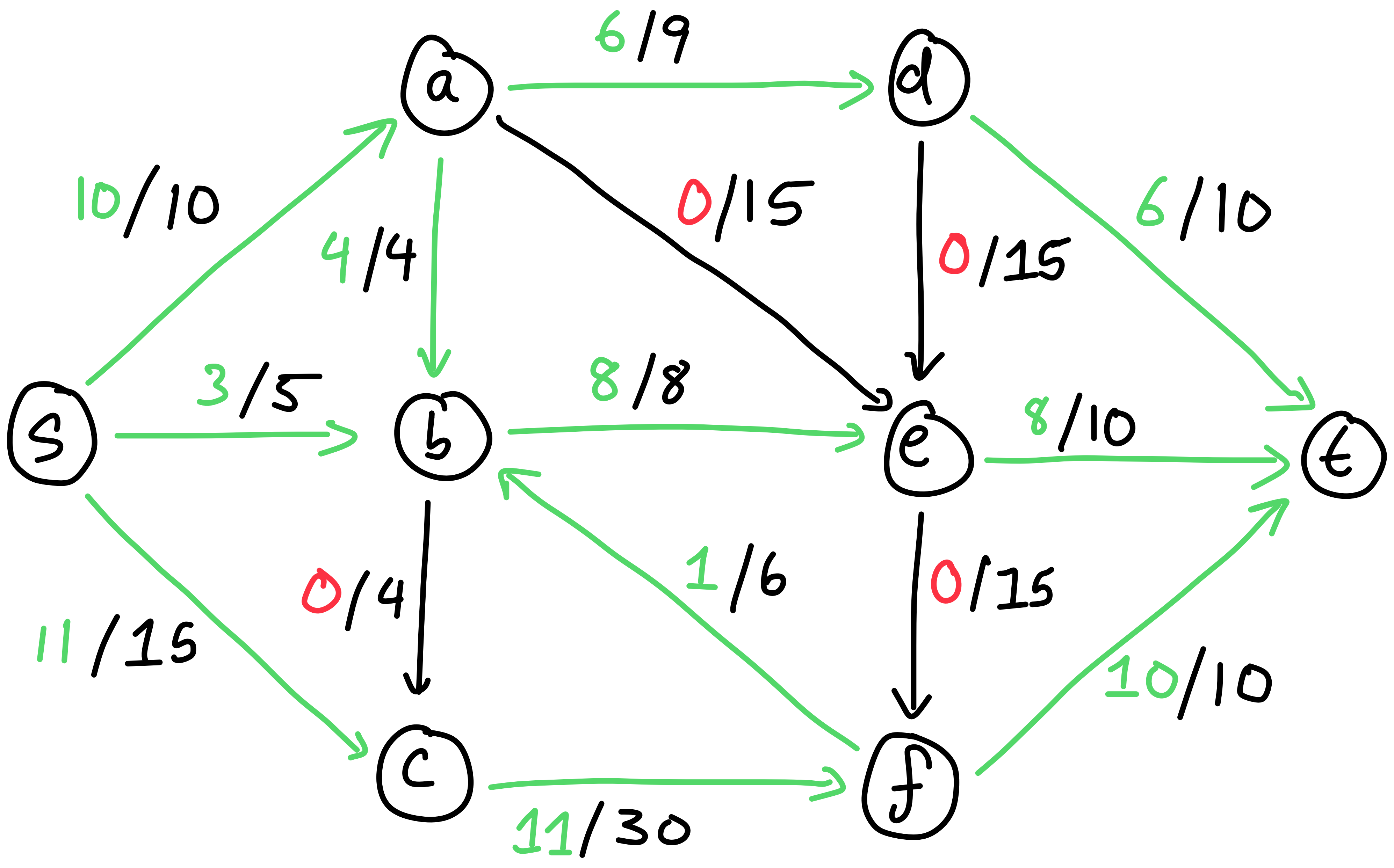
flow of value 24.



The maximum flow problem

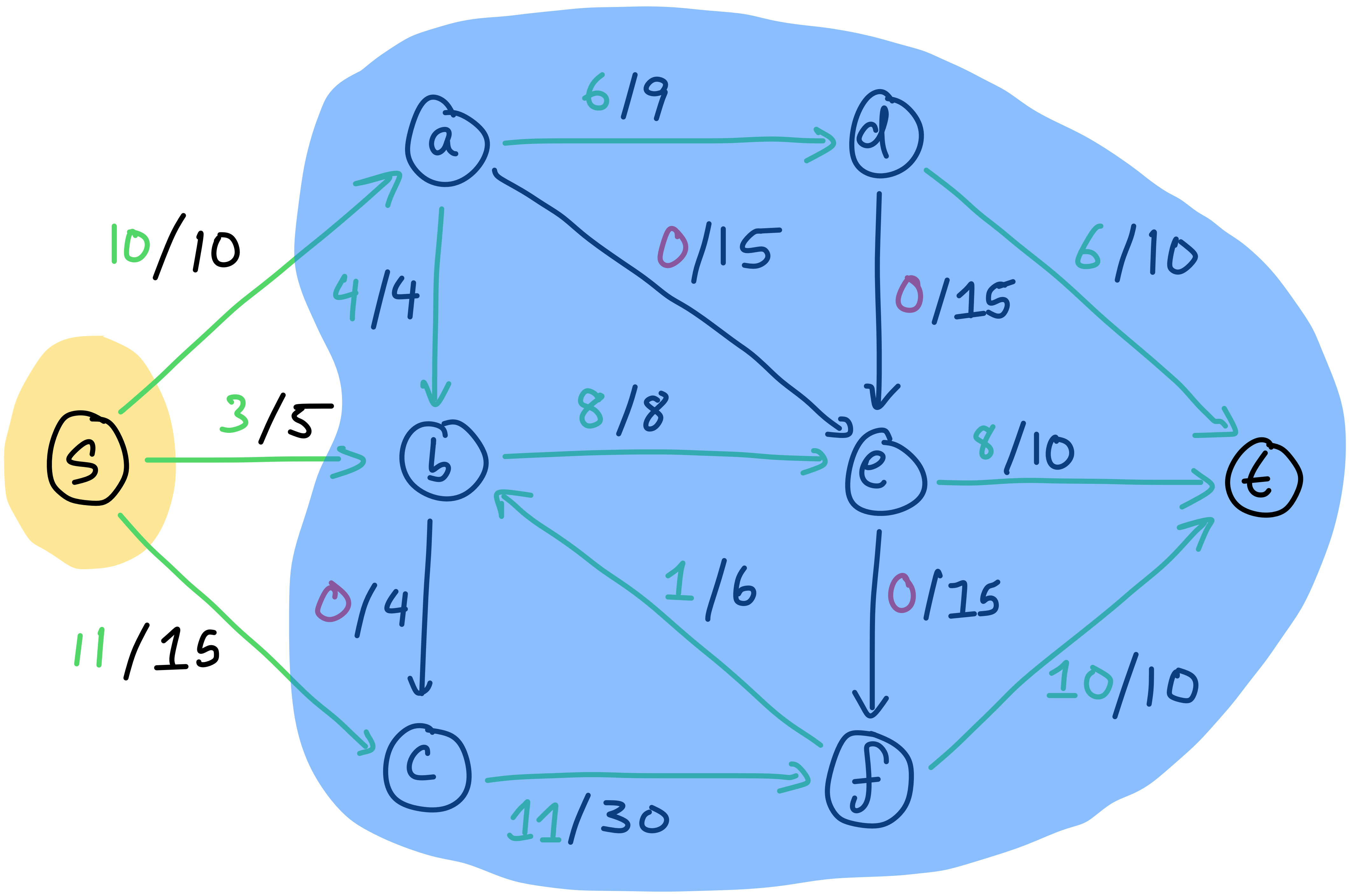
- **Input:** a flow network (G, c, s, t)
- **Output:** a s-t flow of maximum value

flow of value 24.



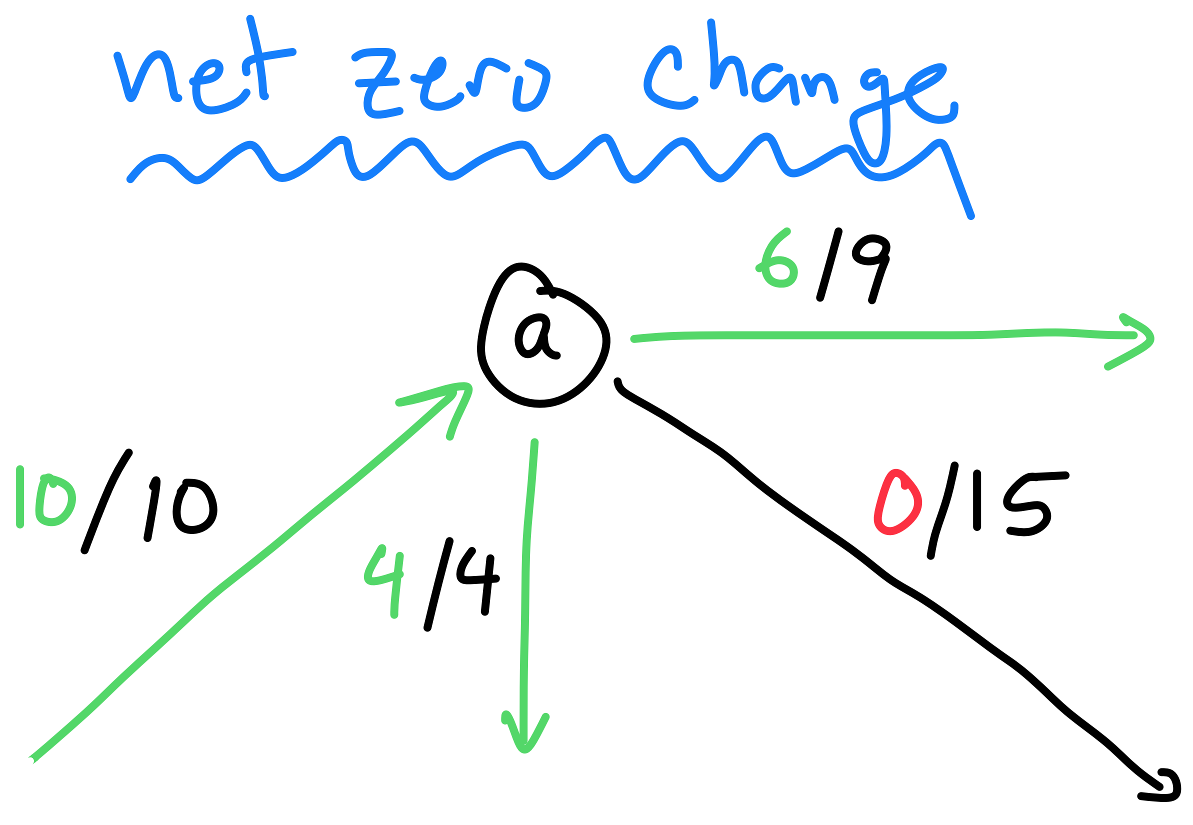
Conservation of flow

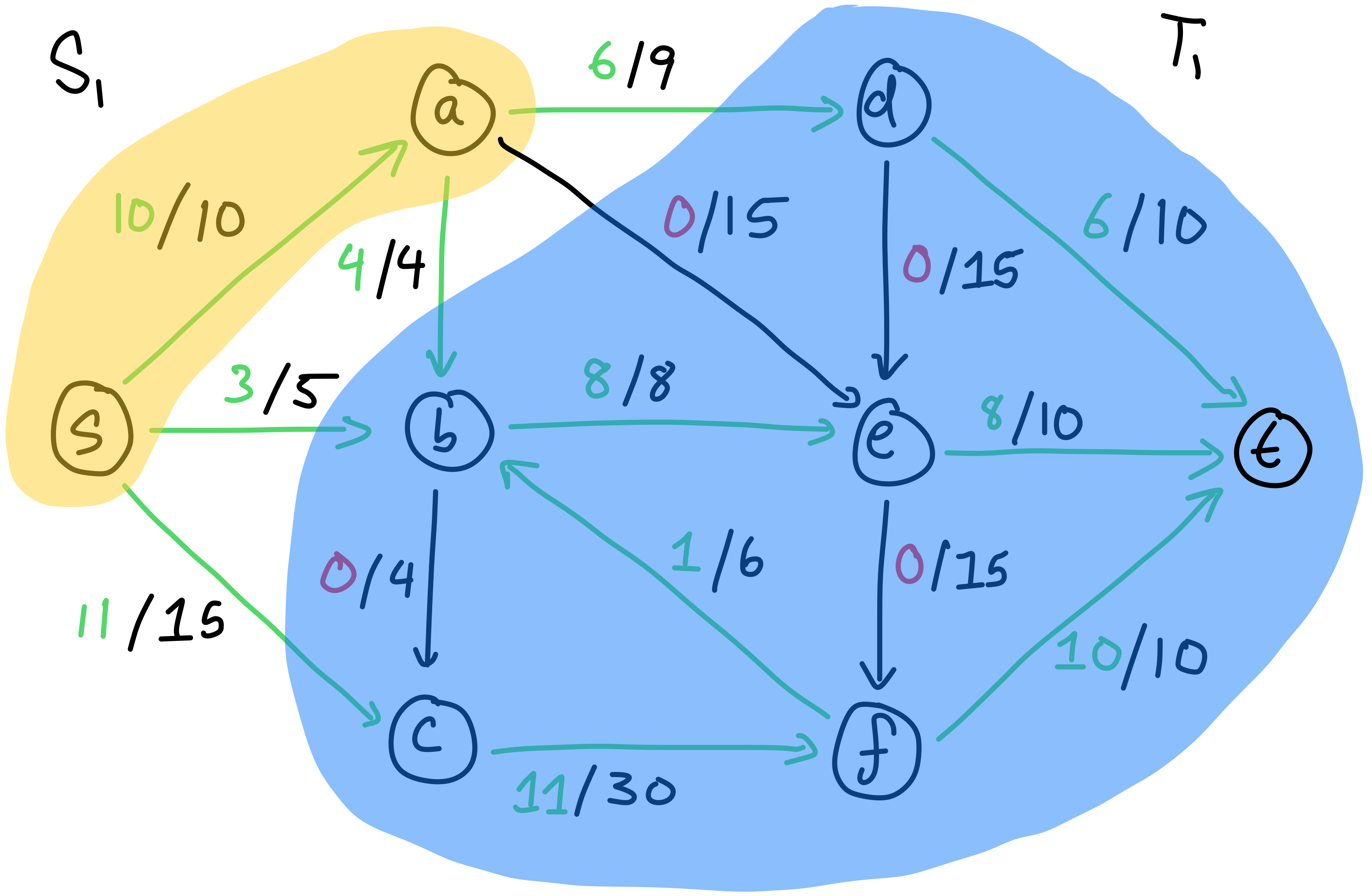
- Let $S_0 = \{s\}, T_0 = V \backslash \{s\}$.
- Then, $v(f) = \sum_{e \text{ from } S_0 \text{ to } T_0} f(e)$.



Conservation of flow

- Let $S_0 = \{s\}, T_0 = V \backslash \{s\}$.
- Then, $v(f) = \sum_{e \text{ from } S_0 \text{ to } T_0} f(e)$.
- Define $S_1 \leftarrow S_0 \cup \{a\}, T_1 \leftarrow T_0 \backslash \{a\}$.
- **Claim:** $v(f) = \sum_{e \text{ from } S_1 \text{ to } T_1} f(e)$.
- **Proof:** Switching between sums requires
  - subtracting the flow $f(s \to a)$ and
  - adding the flows $f(a \to b), f(a \to e), f(a \to d)$.
  - by flow conservation, these changes are net zero.



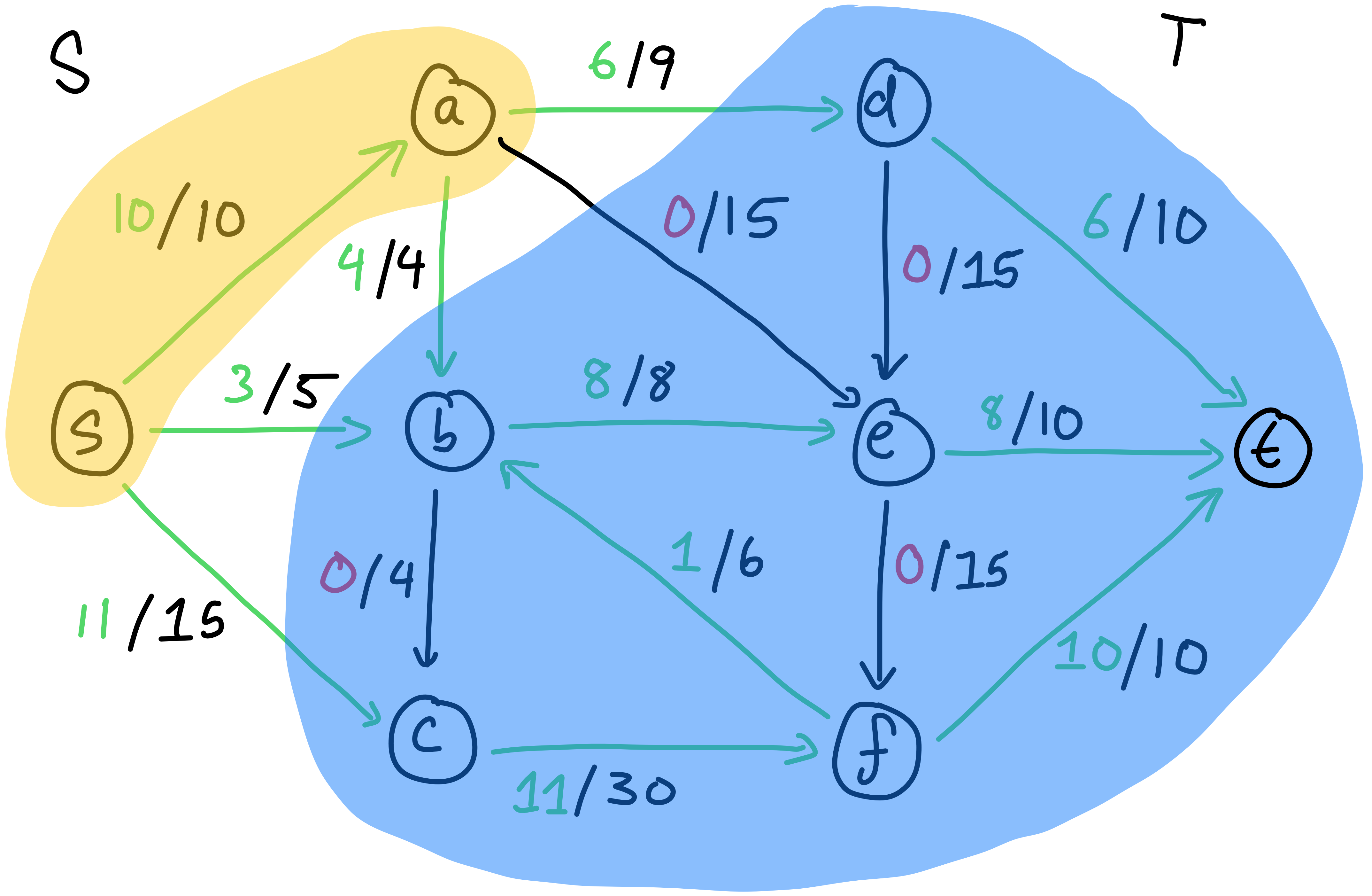


Flow value lemma

- **Flow value lemma:** Let f be a s-t flow and *any* s-t cut (S, T). Then

$$v(f) = \sum_{e \text{ from } S \text{ to } T} f(e) - \sum_{e \text{ from } T \text{ to } S} f(e)$$

- **Proof (intuition):**
  - Add the vertices of S one by one until the set is generated.

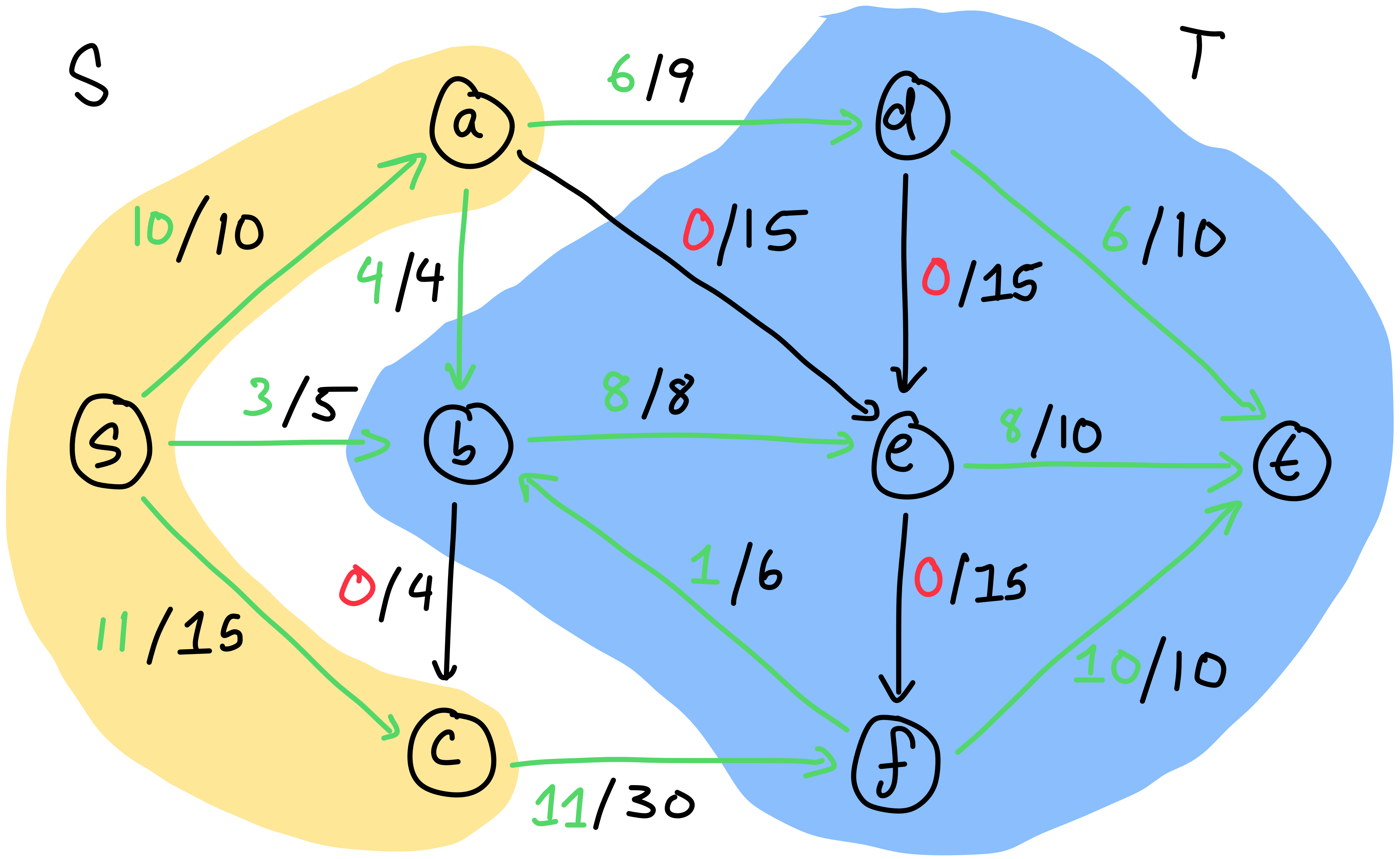


Flow value lemma

- **Flow value lemma:** Let f be a s-t flow and *any* s-t cut (S, T). Then

$$v(f) = \sum_{e \text{ from } S \text{ to } T} f(e) - \sum_{e \text{ from } T \text{ to } S} f(e)$$

- **Proof (intuition):**
    - Add the vertices of S one by one until the set is generated.

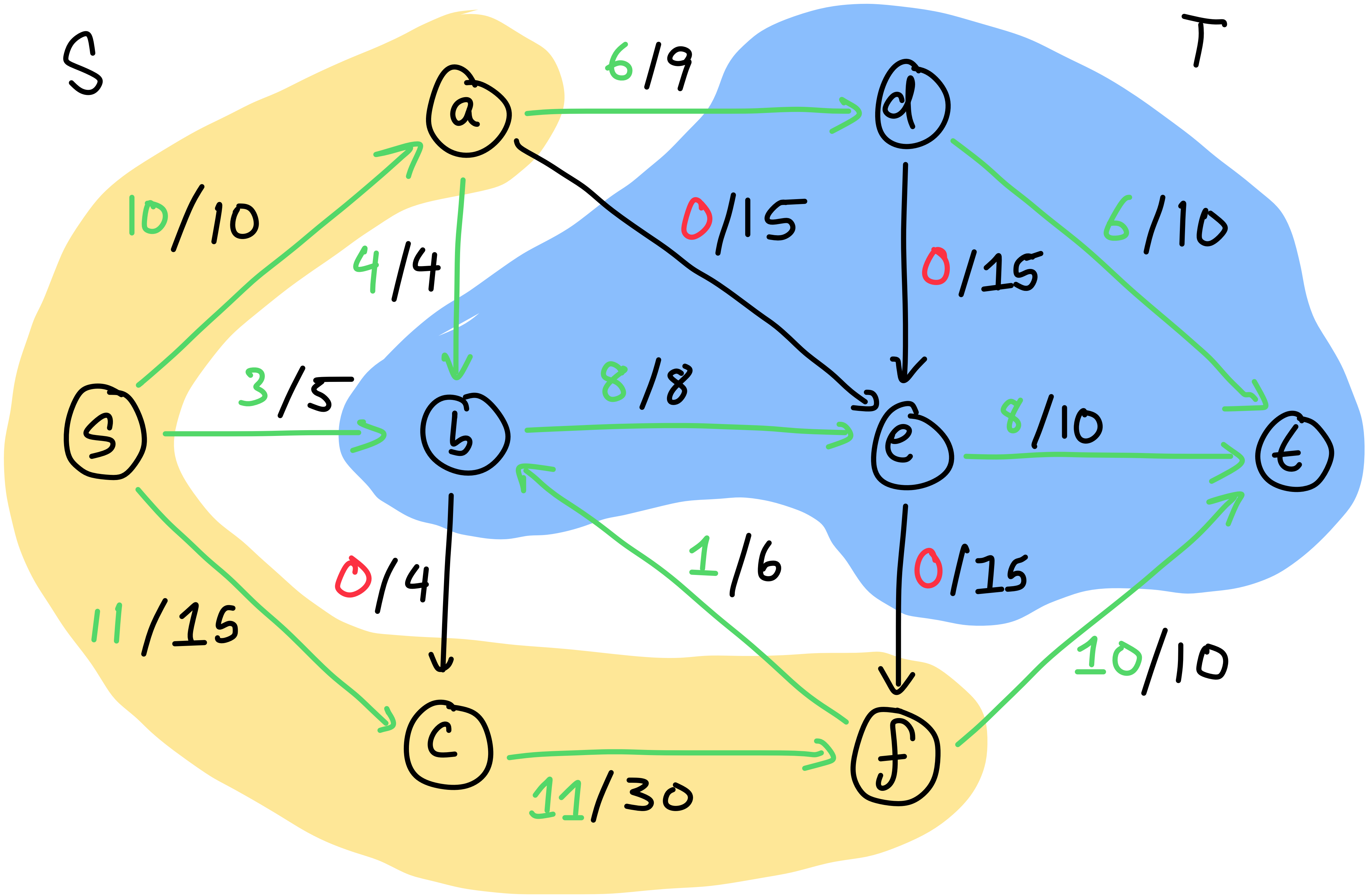


Flow value lemma

- **Flow value lemma:** Let f be a s-t flow and *any* s-t cut (S, T). Then

$$v(f) = \sum_{e \text{ from } S \text{ to } T} f(e) - \sum_{e \text{ from } T \text{ to } S} f(e)$$

- **Proof (intuition):**
    - Add the vertices of S one by one until the set is generated.



Flow value proof (formal)

- Let $a \in T\backslash\{t\}$ for a s-t cut (S, T) . Then,

$$\sum_{e \text{ from } S \text{ to } T} f(e) - \sum_{e \text{ from } T \text{ to } S} f(e) = \sum_{e \text{ from } S \text{ to } T} f(e) \overbrace{- \sum_{e \text{ from } S \text{ to } a} f(e) + \sum_{e \text{ from } a \text{ to } T} f(e) + \sum_{e \text{ from } a \text{ to } S} f(e) - \sum_{e \text{ from } T \text{ to } a} f(e)}^{=0 \text{ by conservation of flow}} - \sum_{e \text{ from } T \text{ to } S} f(e)$$

$$= \sum_{e \text{ from } S \text{ to } T\backslash\{a\}} f(e) + \sum_{e \text{ from } a \text{ to } T} f(e) + \sum_{e \text{ from } a \text{ to } S} f(e) - \sum_{e \text{ from } T \text{ to } S\cup\{a\}} f(e)$$

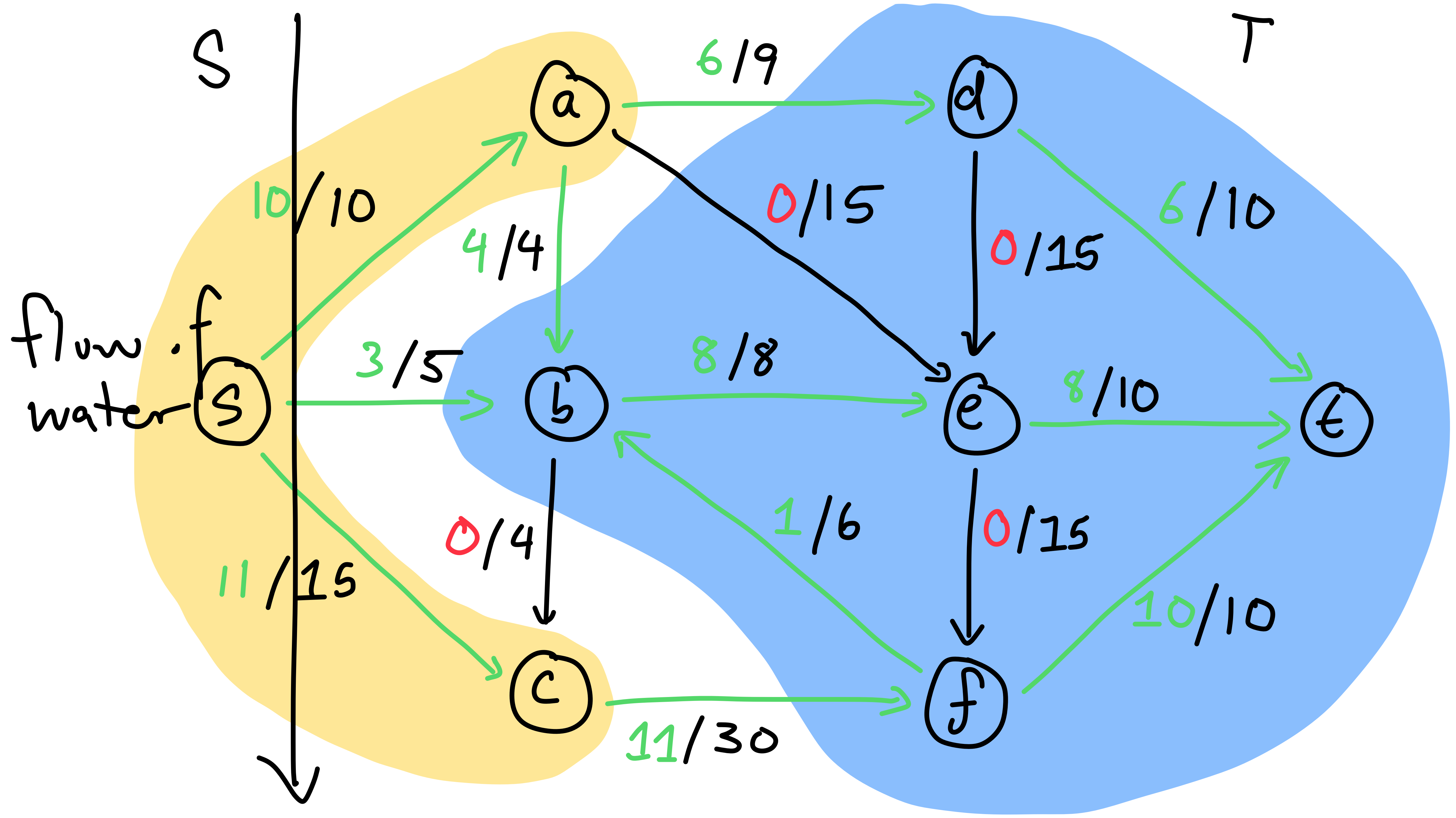
$$= \sum_{e \text{ from } S\cup\{a\} \text{ to } T\backslash\{a\}} f(e) - \sum_{e \text{ from } T\backslash\{a\} \text{ to } S\cup\{a\}} f(e).$$

Proves inductive step.

So the value of the flow is equal for every s-t cut.

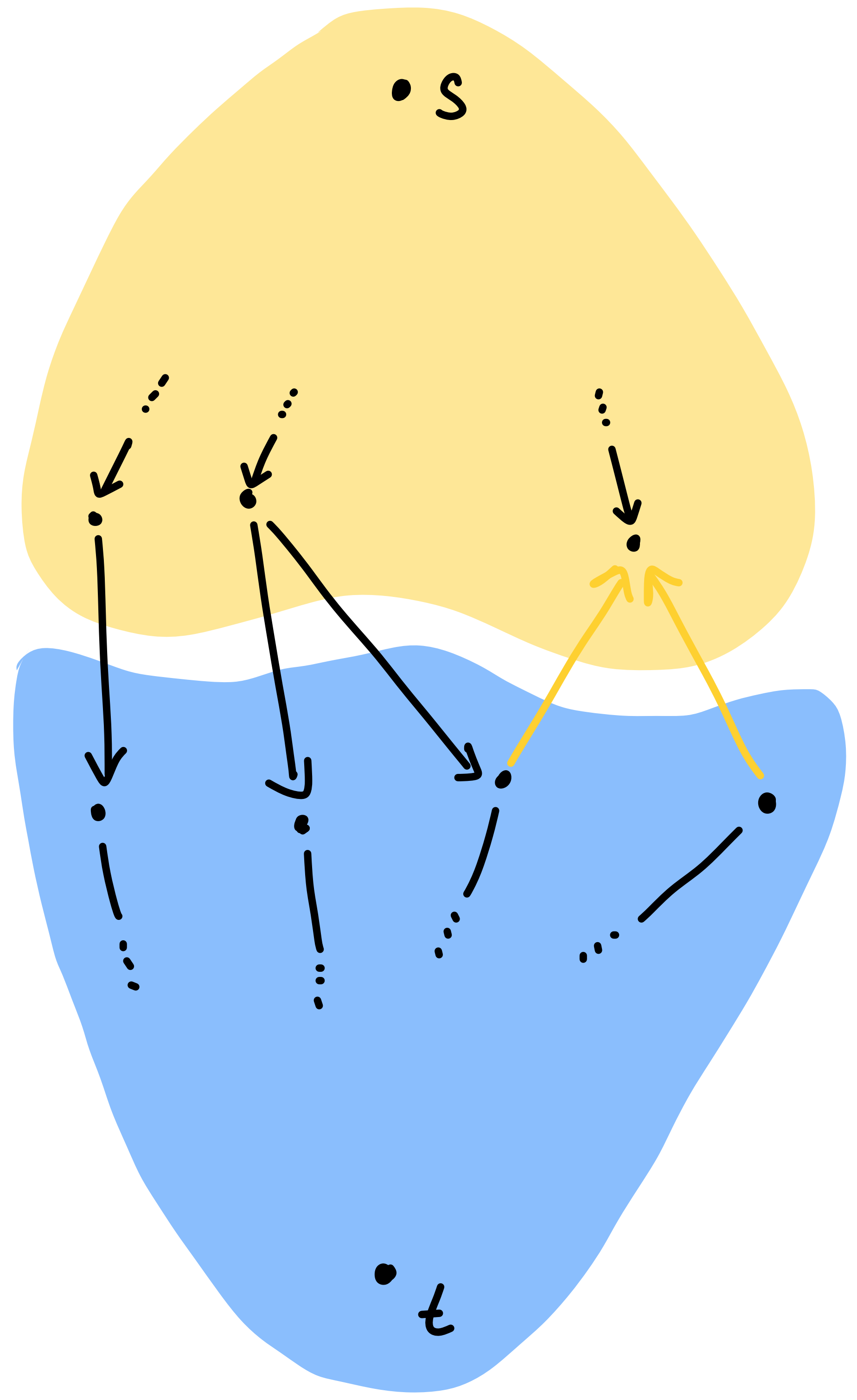
The water intuition

- Imagine the edges as pipes and water is flowing from s at a *steady* rate of $v(f)$.
- The flow of water leaving s must equal the flow of water leaving S.
- Water moving within S or T is inconsequential to the total flow



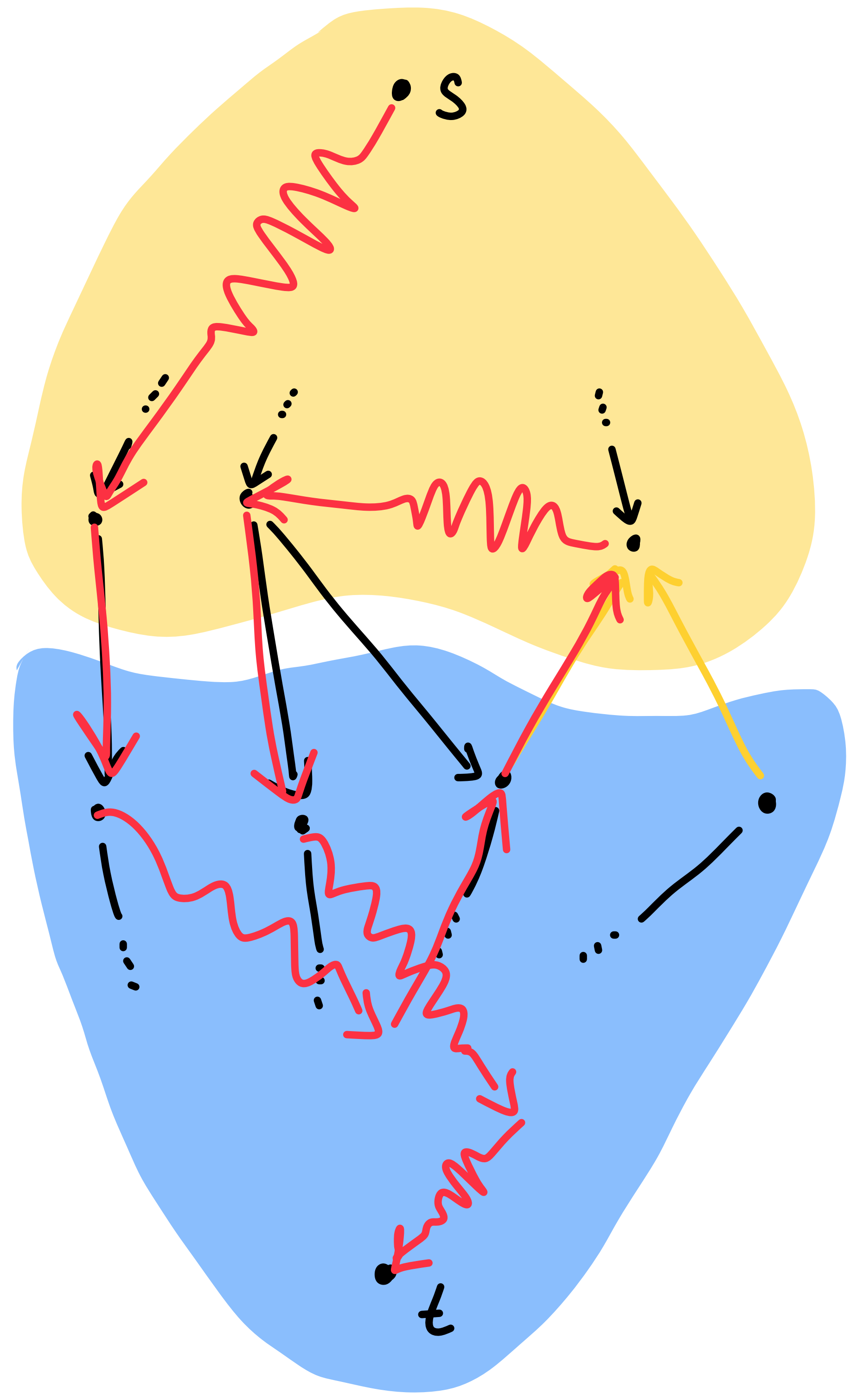
The relationship between flows and cuts

- **Weak duality:** For any s-t cut (S, T), $v(f) \leq C(S, T)$.
- **Proof intuition:**
    - In order for water to flow (positively) from S to T it has to use one of the edges from S to T.
    - The total capacity of which is $C(S, T)$.
    - And the value of the flow is $\leq$ the sum of the flow out of S.



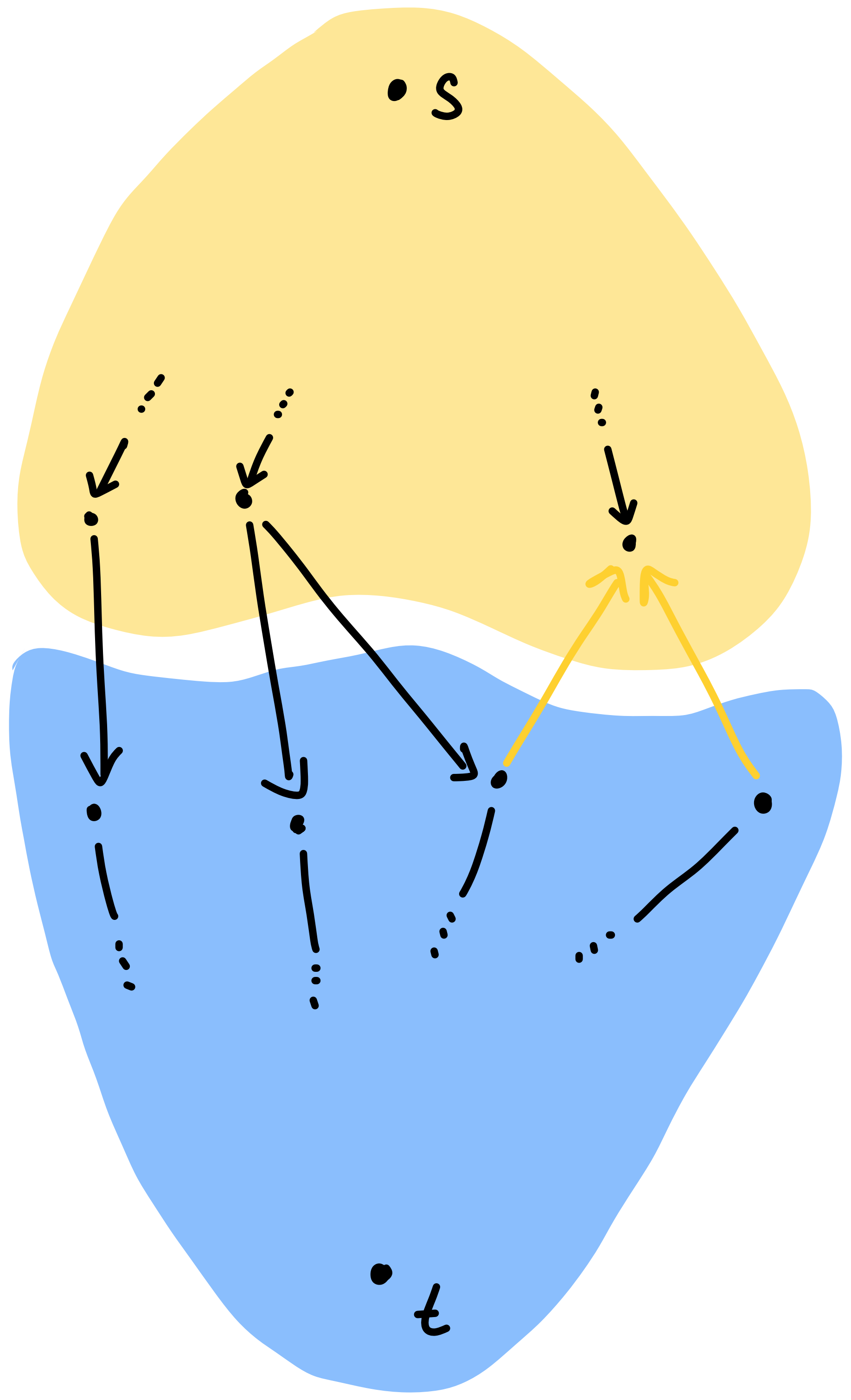
The relationship between flows and cuts

- **Weak duality:** For any s-t cut (S, T), $v(f) \leq C(S, T)$.
- **Proof intuition:**
    - In order for water to flow (positively) from S to T it has to use one of the edges from S to T.
    - The total capacity of which is $C(S, T)$.
    - And the value of the flow is $\leq$ the sum of the flow out of S.



The relationship between flows and cuts

- **Weak duality:** For any s-t cut (S, T), $v(f) \leq C(S, T)$.
- **Proof:**
- $$v(f) = \sum_{e \text{ from } S \text{ to } T} f(e) - \underbrace{\sum_{e \text{ from } T \text{ to } S} f(e)}_{\geq 0 \text{ since } f(e) \geq 0 \text{ for all edges}}$$
- $$\leq \sum_{e \text{ from } S \text{ to } T} f(e)$$
- $$\leq \sum_{e \text{ from } S \text{ to } T} c(e) \quad \leftarrow \text{since } f(e) \leq c(e) \text{ for all edges}$$
- $$= C(S, T)$$

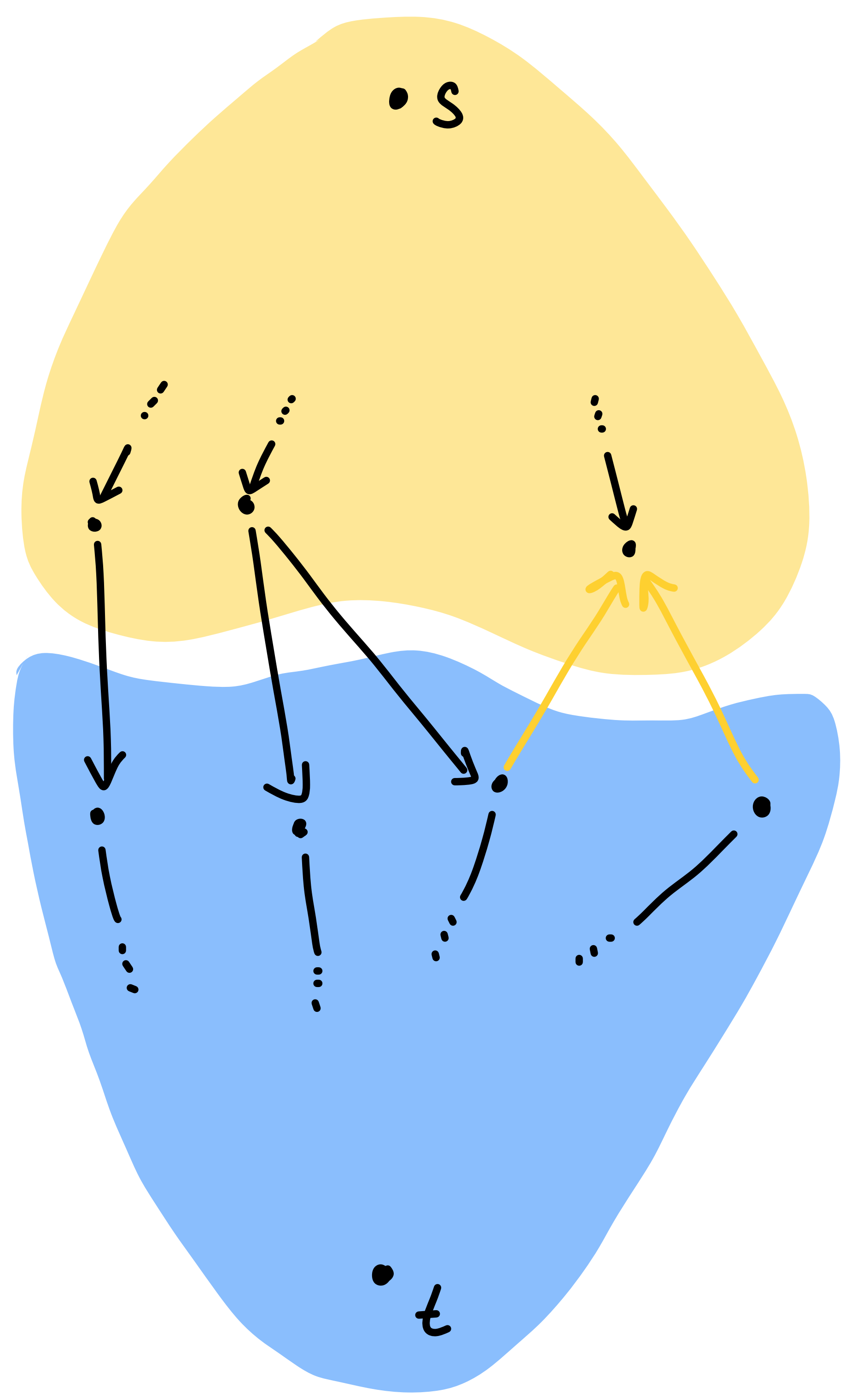


The relationship between flows and cuts

- **Weak duality:** For any s-t cut (S, T), $v(f) \leq C(S, T)$.
- **Corollary:** As this is true for all s-t cuts and all s-t flows, for any flow network,

  The max flow is always $\leq$ the min cut.

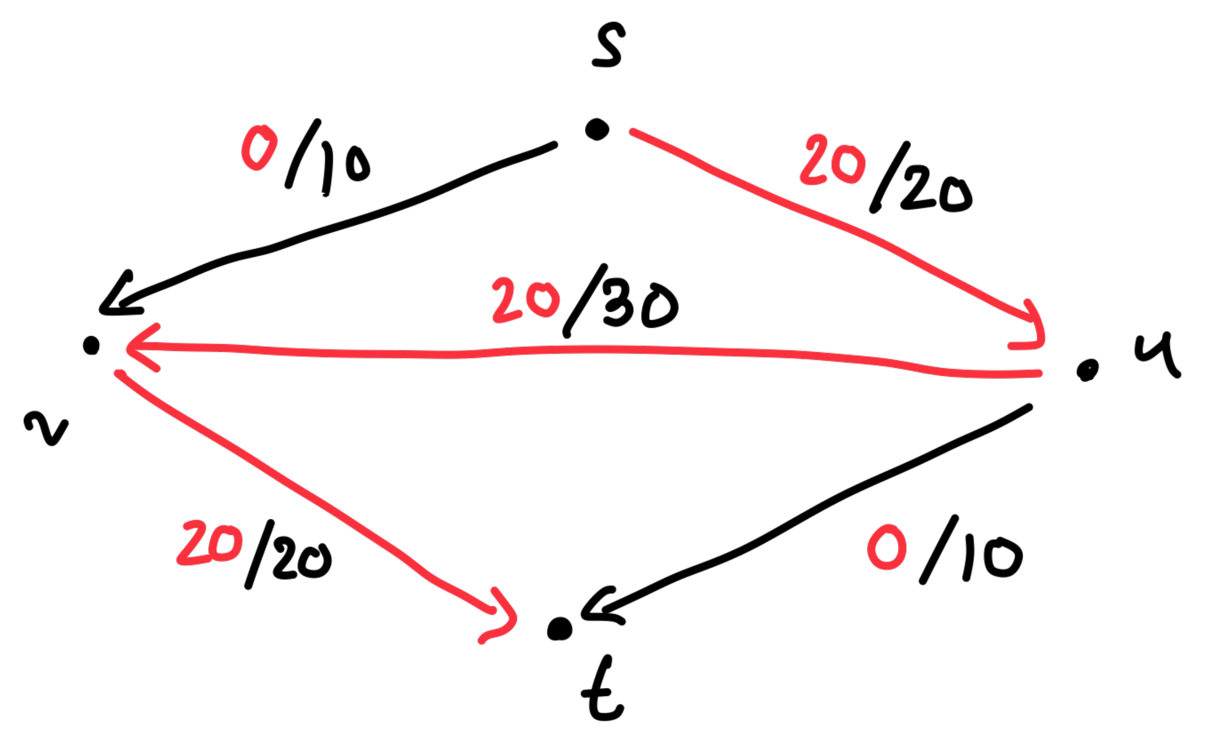
- **Theorem:** If there exists a flow f and a cut (S, T) such that $v(f) = c(S, T)$ then f must be a maximal flow and (S, T) must be a minimizing cut.



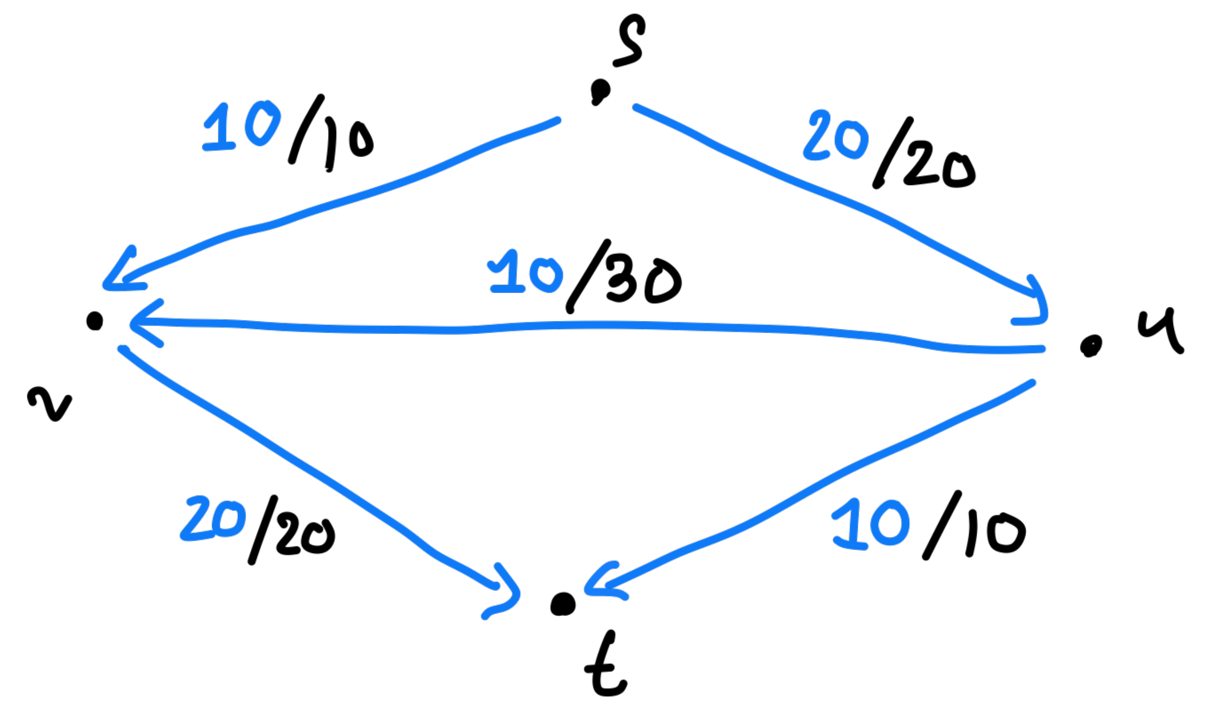
Algorithms for max flow

- **Greedy algorithm attempt:**
  - Start with $f(e) = 0$.
  - While there is a s-t path $p : s \rightsquigarrow t$ where each edge $e \in p$ has $f(e) < c(e)$,
    - "Augment" the flow along p by adding α flow on each edge $e \in p$
    - Where $\alpha = \min_{e \in p} \left[c(e) - f(e) \right]$
  - Each augmentation increases $v(f)$ by α and preserves a valid flow (capacity and conservation of flow constraints).

Greedy algorithm can get stuck…



Even if a larger flow exists:

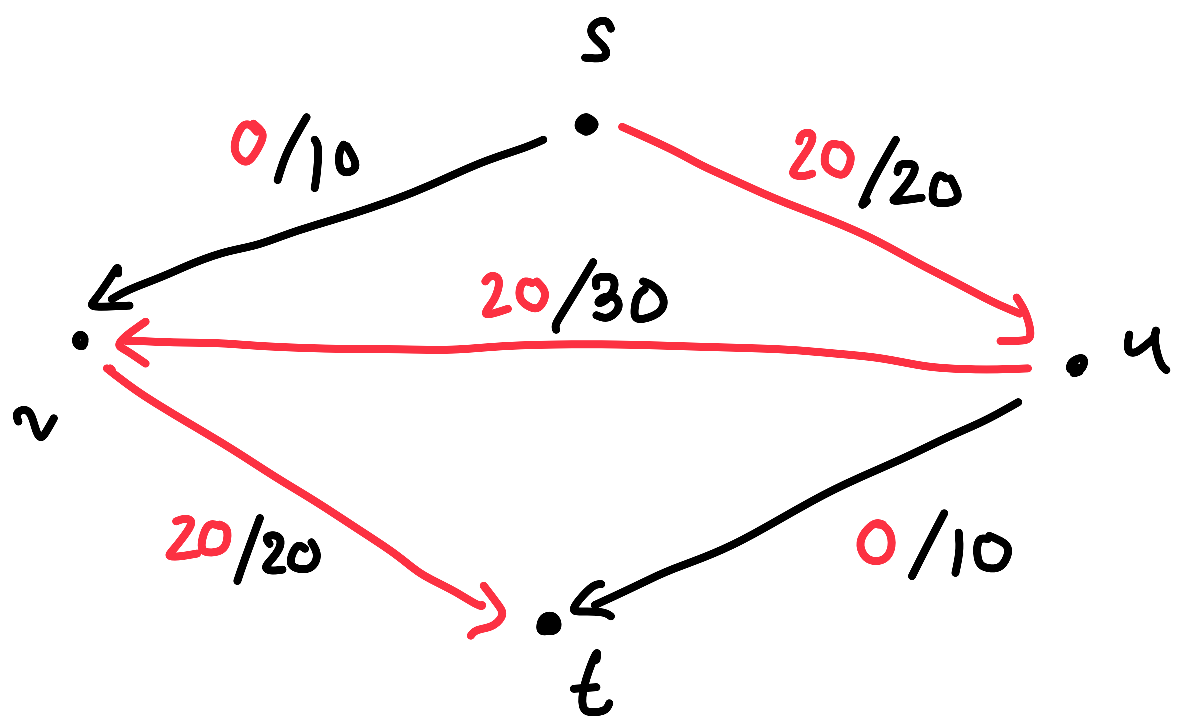


Greedy algorithms get stuck

- What if there was a way to "undo" a choice made by a greedy algorithm and keep going?
- Residual graphs
  - A graph that represents how much we can change for any edge
  - If an edge has a capacity of $c(e)$ and is currently flow assigns it $f(e) \leq c(e)$
    - Then we can either add up to $f(e) - c(e)$ additional flow
    - Or remove up to $c(e)$ flow from this edge.

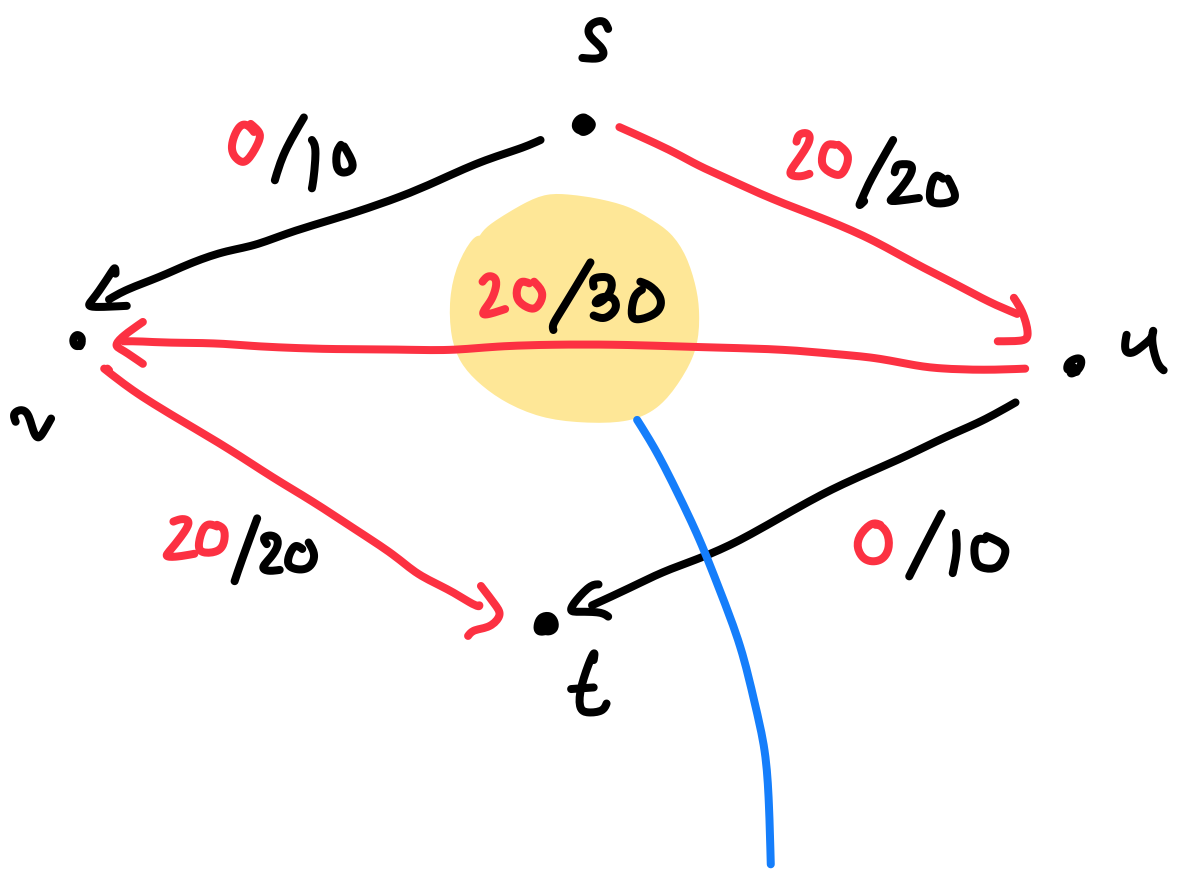
Augmenting paths through residual flow

Current choice of flow:

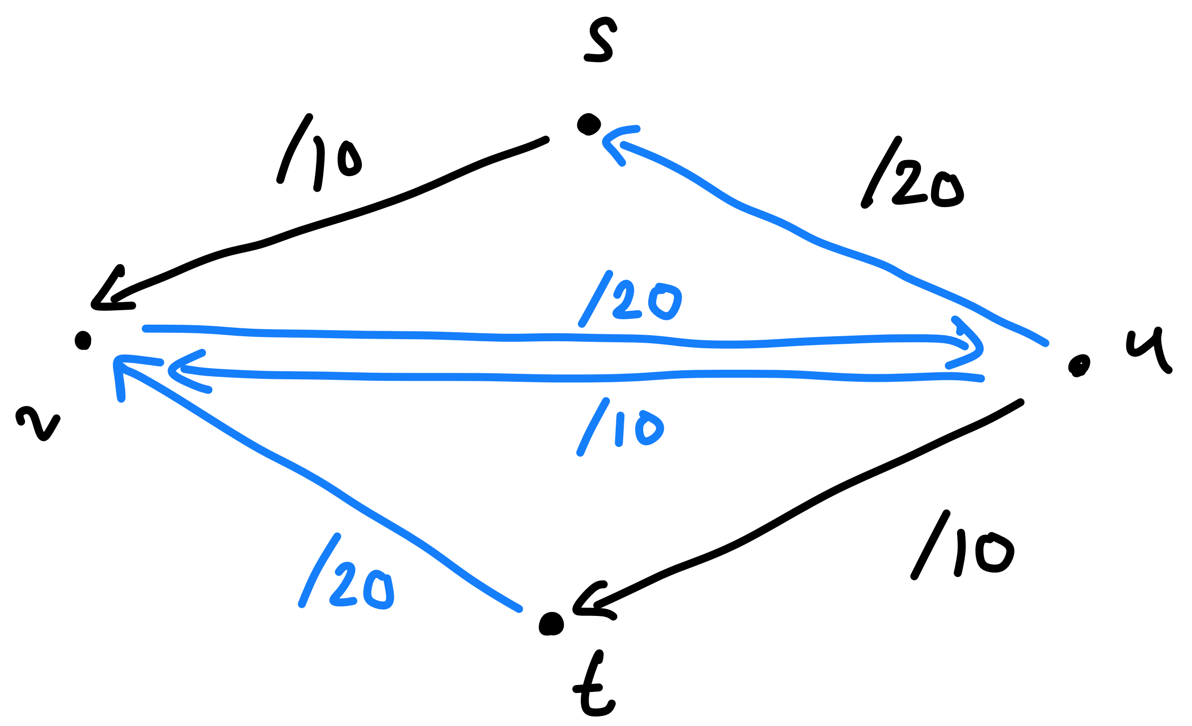


Augmenting paths through residual flow

Current choice of flow:



Residual flow network



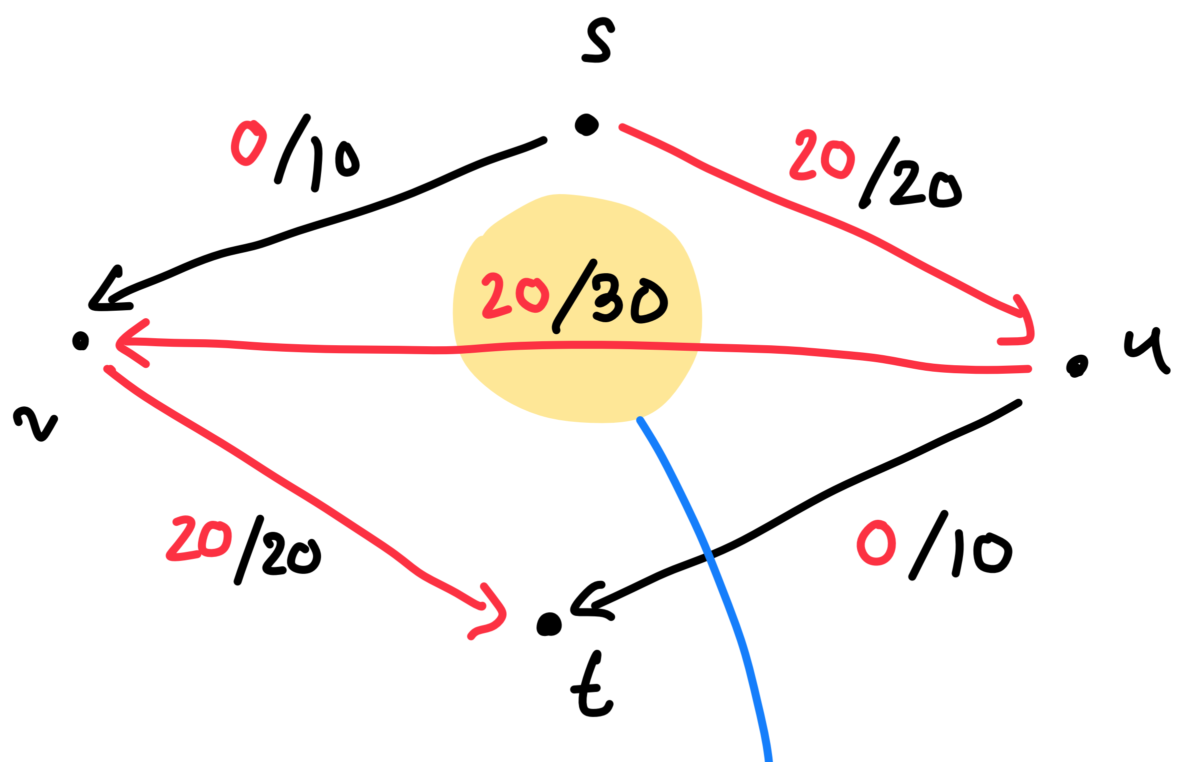
Can either add 10 flow to the left ←

or

remove 20 flow, i.e. add 20 flow to the right ⟶

Augmenting paths through residual flow

Current choice of flow:

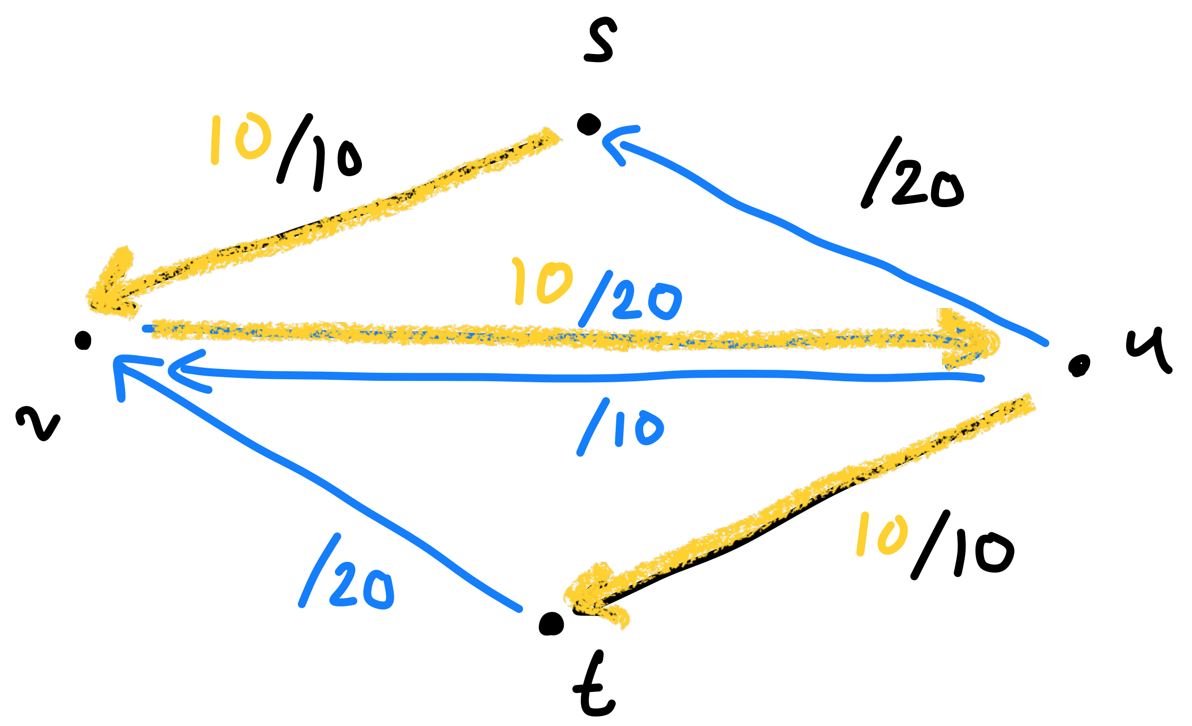


Can either add 10 flow to the left ←

or

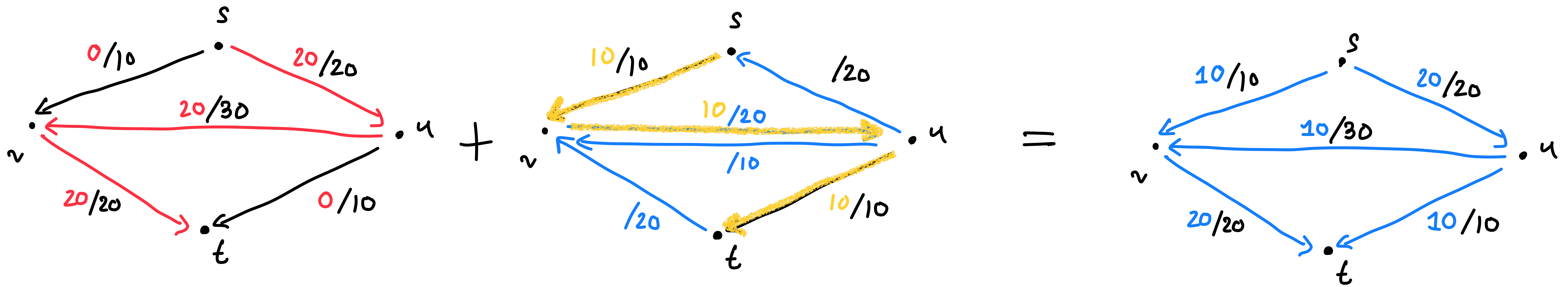
remove 20 flow, i.e. add 20 flow to the right ⟶

Residual flow network



We can find an augmenting path in the residual flow network.

Augmenting paths through residual flow



original flow

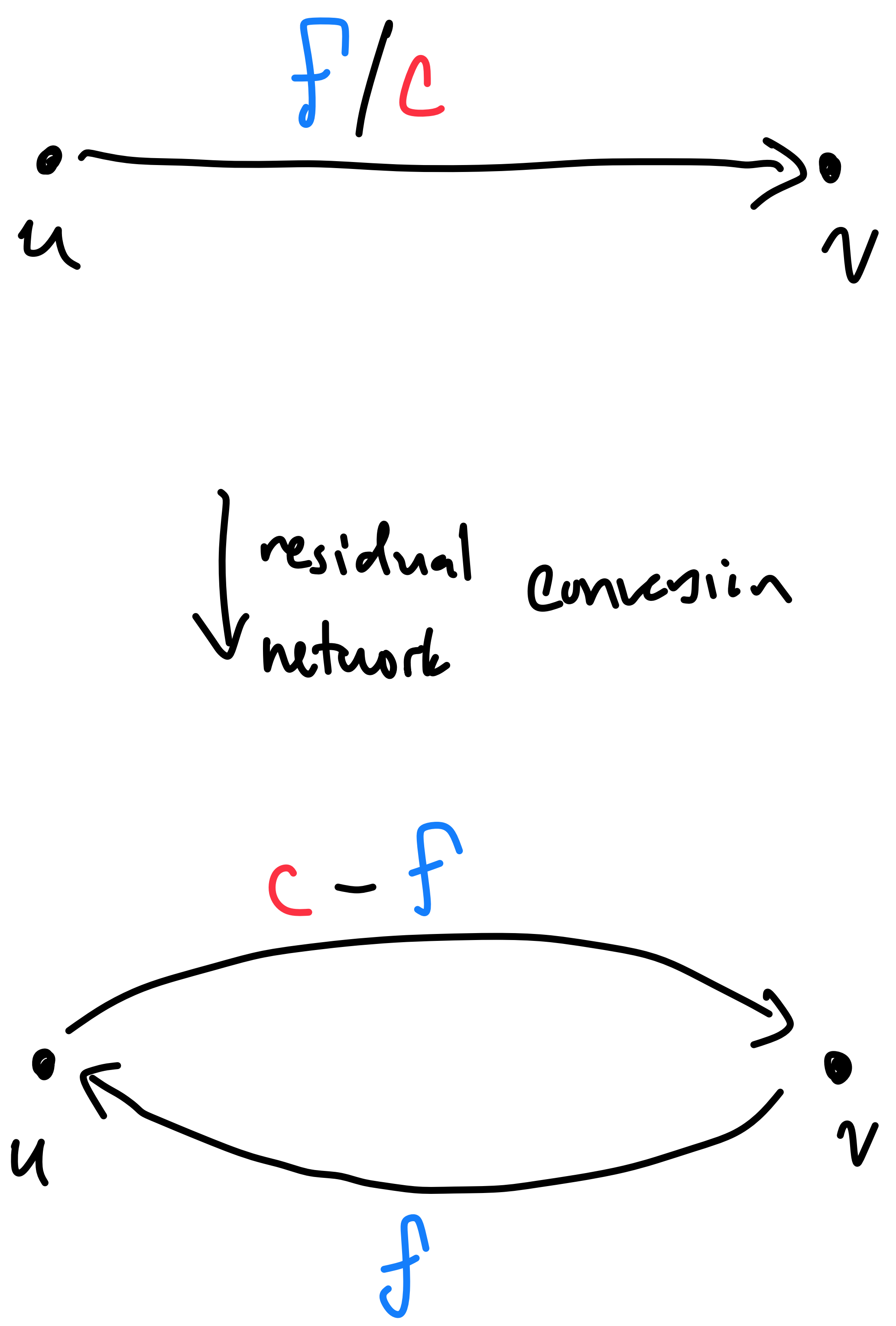
augmenting path in residual network

a larger flow in the original network

in this case optimal since flow = $c(S,T)$ for all S,T.

Residual network definition

- For (G, c, s, t) and flow f, define G_f as the **residual network** with the same vertices, source s and sink t
  - For every edge $e = (u \to v)$,
    - (Forward edge): Add an edge $u \to v$ of capacity $c(e) - f(e)$
    - (Backward edge): Add an edge $v \to u$ of capacity $f(e)$



Notation

- For a flow f, let $f^{\text{out}}(v) = \sum_{e \text{ outof } v} f(e), \quad f^{\text{in}}(v) = \sum_{e \text{ into } v} f(e)$.
- Conservation of flow: $f^{\text{in}}(v) = f^{\text{out}}(v)$.
- Positivity of flow: $0 \leq f(e) \leq c(e)$.

Augmenting path

- An alternative (and mathematically equivalent) way to think about an augment flow f_{aug} in the residual network G_f is that
  - Capacity constraints: $-f(e) \leq f_{\text{aug}} \leq c(e) - f(e)$
  - Conservation of augmenting flow: $(f_{\text{aug}})^{\text{in}}(v) = (f_{\text{aug}})^{\text{out}}(v)$
- **Claim:** For flow f in G and augmenting flow f_{aug} in G_f, $f + f_{\text{aug}}$ is a flow in G.
- **Proof:** Adding up capacity constraints and conservation equations proves that $f + f_{\text{aug}}$ is a valid flow. ∎
- $v(f + f_{\text{aug}}) = v(f) + v(f_{\text{aug}})$ so a positive augmenting flow increases the flow in the graph.

New greedy algorithm (Ford-Fulkerson)

- Initialize a flow of $f(e) \leftarrow 0$ for all edges. Set residual network $G_f \leftarrow G$
- While there is a simple path $p : s \rightsquigarrow t$ in G_f

  no repeat vertices

  - Let f_{aug} be the flow along p of weight $\min_{e \in p} c_{G_f}(e)$
  - Augment $f \leftarrow f + p$
  - Update G_f along the edges of p

O(n) time.

How do we find such a path?

One option is run Graph Traversal from s to t using the edges of positive capacity.

O(n+m) time.

How many times will the "while loop" repeat?

Ford Fulkerson algorithm

- **Lemma:** Let (G, c, s, t) be a flow network with integer capacities: $c : E \to \mathbb{Z}_{\geq 0}$ and $C = c^{\text{out}}(s)$.
- Then the previous greedy algorithm terminates in time $O(Cm)$.
- **Proof:**
  - Each iteration of the while loop must increase $v(f)$ by at least 1.
  - C is a trivial bound on the max flow in the network.
  - Therefore, at most C iterations each taking $O(m)$ time.

Ford Fulkerson algorithm correctness

- **Lemma:** Let (G, c, s, t) be a flow network with integer capacities: $c : E \to \mathbb{Z}_{\geq 0}$ and $C = c^{\text{out}}(s)$.
- Then the previous greedy algorithm computes the max flow.
- **Proof:**
  - In next lecture!