Lecture 17

The max flow and min cut problems

Chinmay Nirkhe | CSE 421 Winter 2026

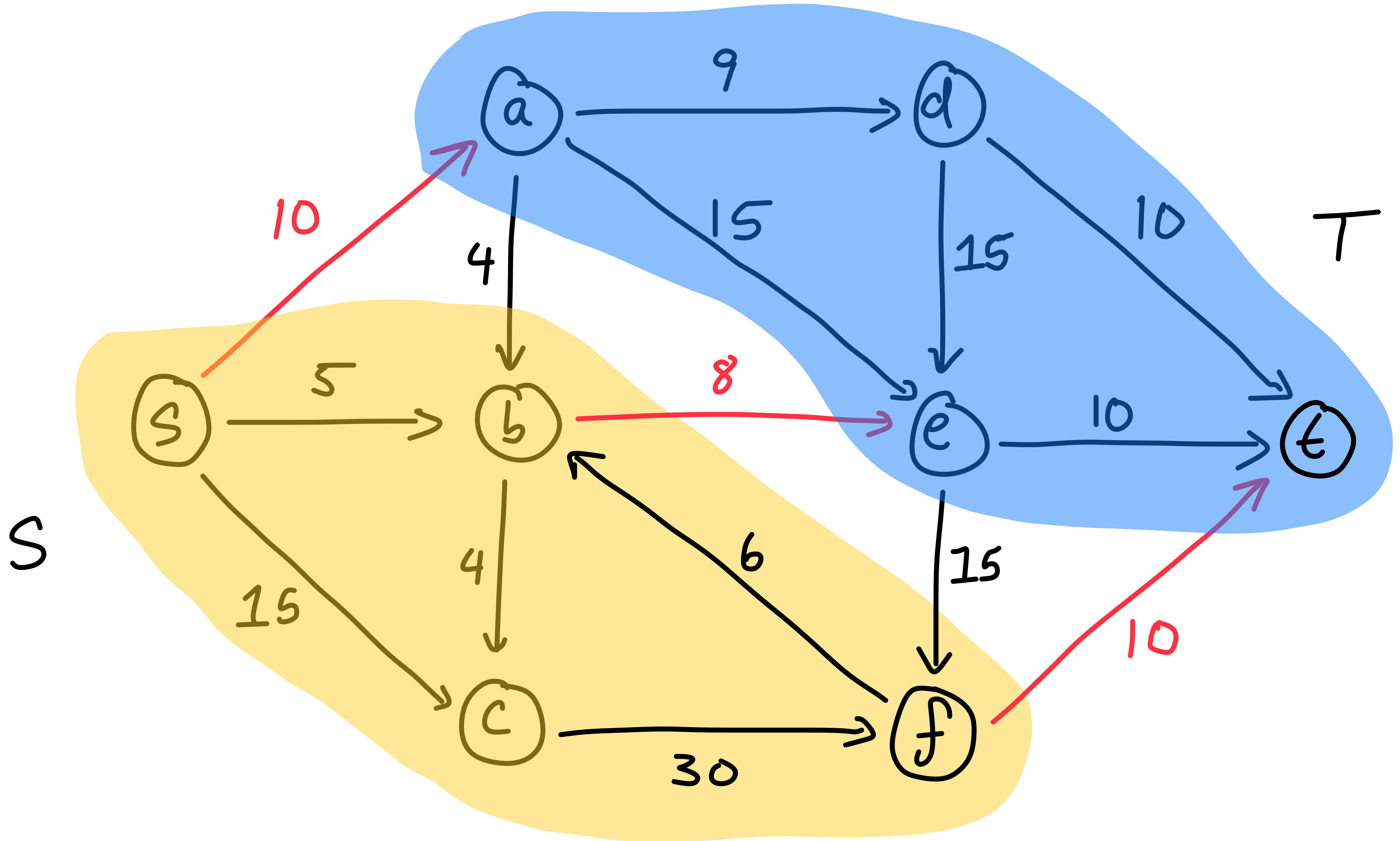
Previously in CSE 421…

The minimum cut problem

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

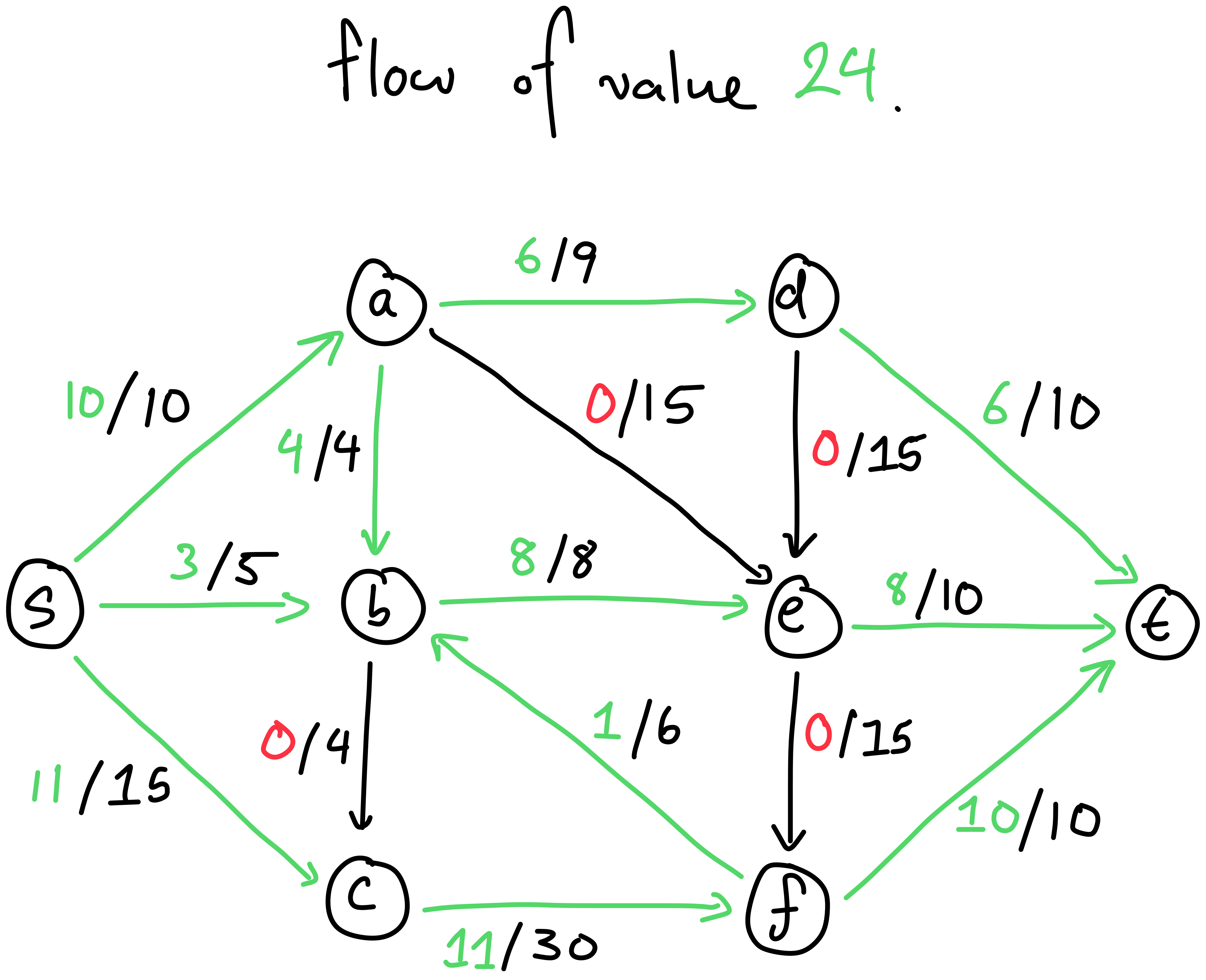
$$\mathrm{mincut}(G, c, s, t) = \min_{\substack{\text{s-t cut} \\ (S,T)}} \{ c(S,T) \}$$

in this case, mincut = 28



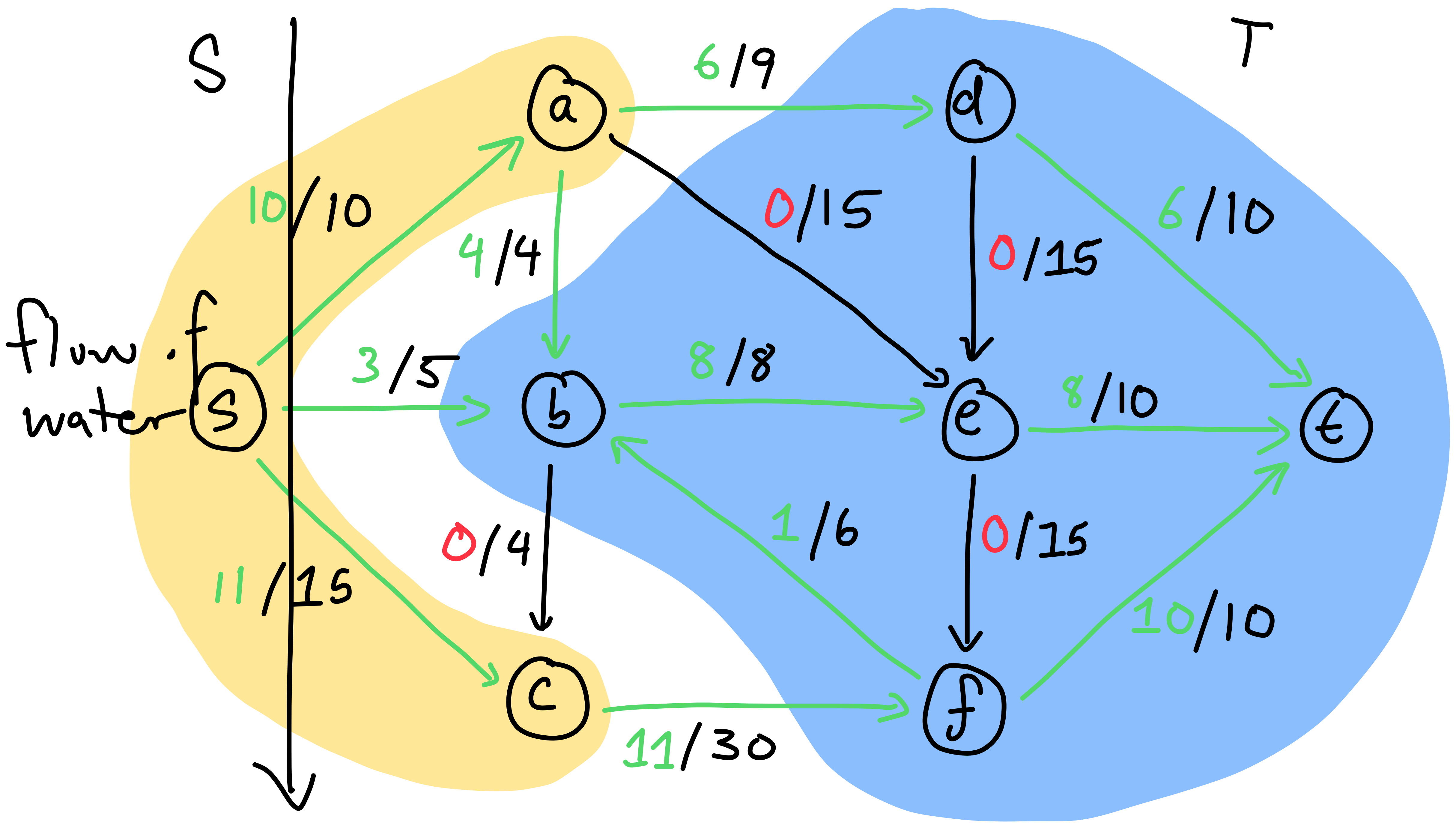
The maximum flow problem

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



The water intuition

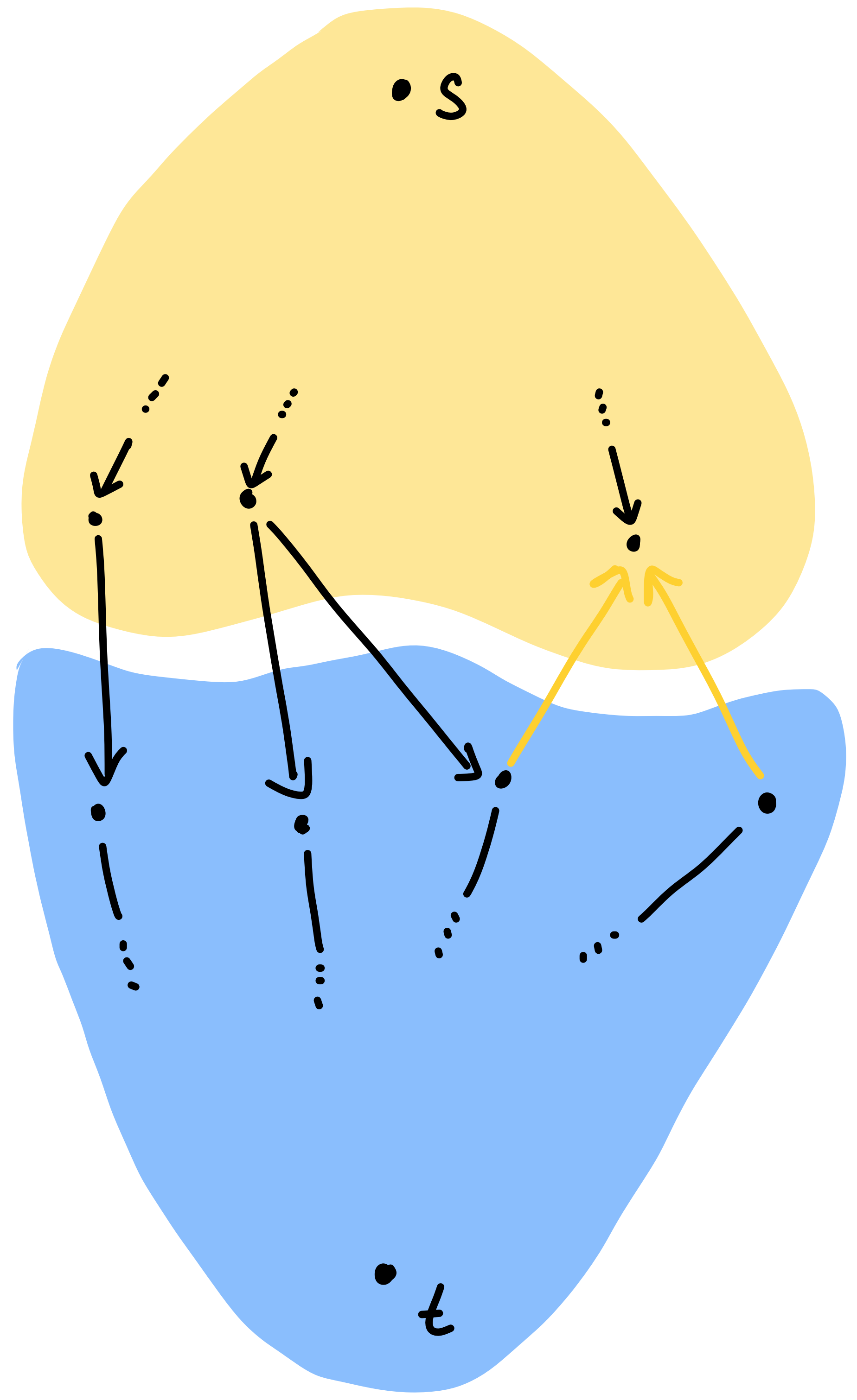
- Imagine the edges as pipes and water is flowing from s at a *steady* rate of $v(f)$.
- The flow of water leaving s must equal the flow of water leaving S.
- Water moving within S or T is inconsequential to the total flow



Today

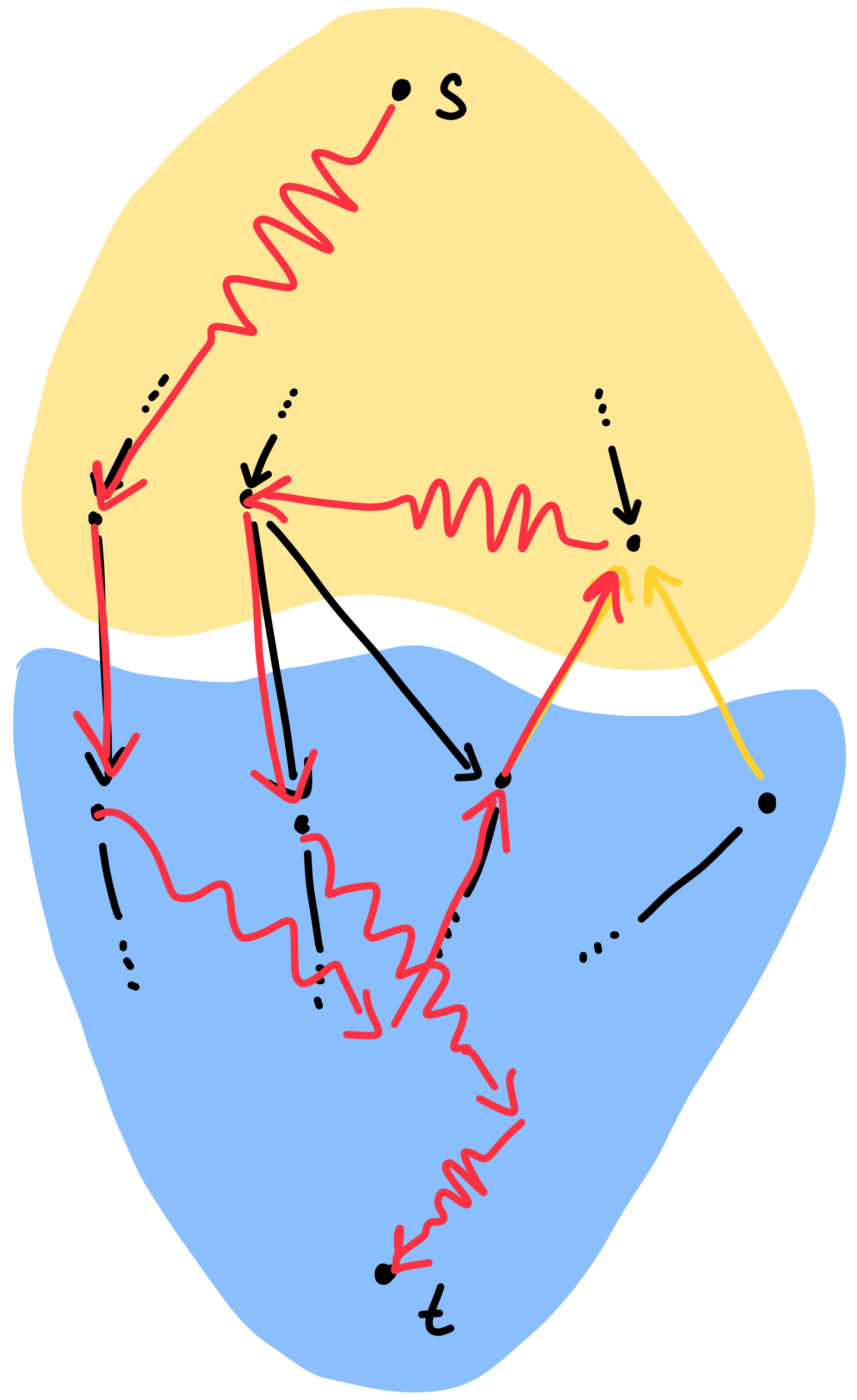
The relationship between flows and cuts

- **Weak duality:** For any s-t cut (S, T) and any valid flow f, $v(f) \leq C(S, T)$.
- **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) and any valid flow f, $v(f) \leq C(S, T)$.
- **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.

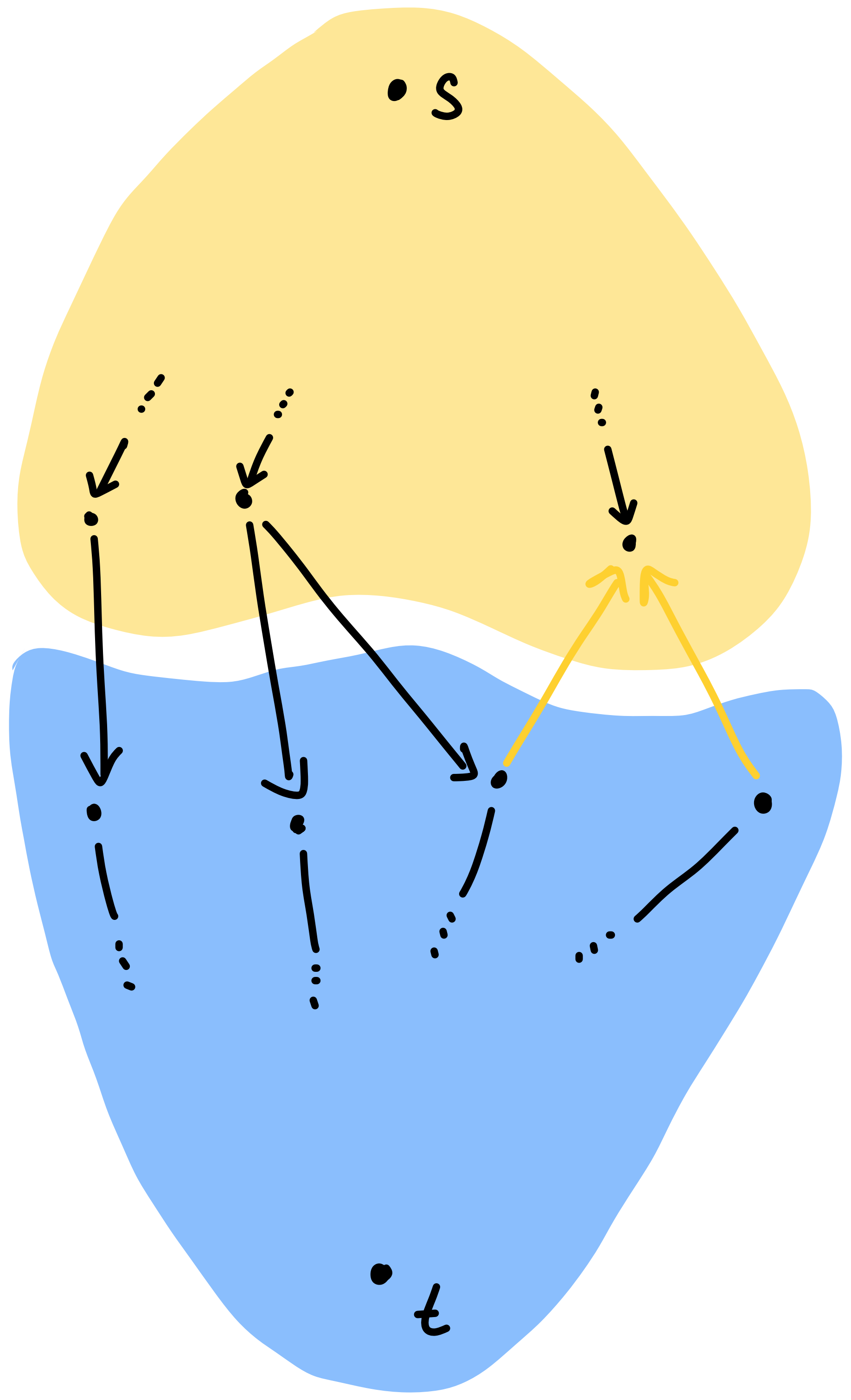


The relationship between flows and cuts

- **Weak duality:** For any s-t cut (S, T) and any valid flow f, $v(f) \leq C(S, T)$.
- **Proof:**
- $v(f) = \sum_{e \text{ from } S \text{ to } T} f(e) - \sum_{e \text{ from } T \text{ to } S} f(e)$
- $\leq \sum_{e \text{ from } S \text{ to } T} f(e)$
- $\leq \sum_{e \text{ from } S \text{ to } T} c(e)$
- $= C(S, T)$

≥ 0 since $f(e) \geq 0$ for all edges

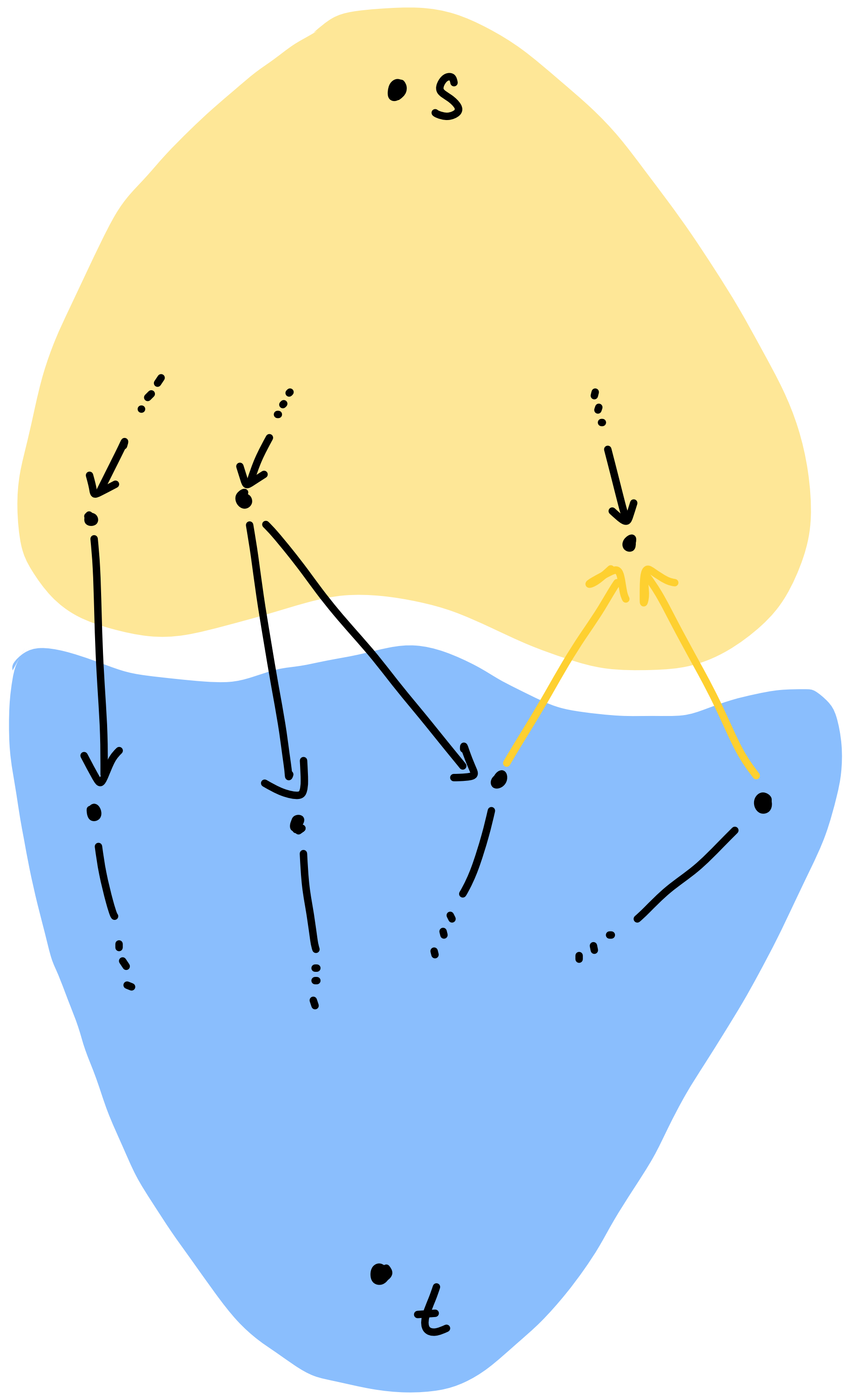
← since $f(e) \leq c(e)$ for all edges



The relationship between flows and cuts

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

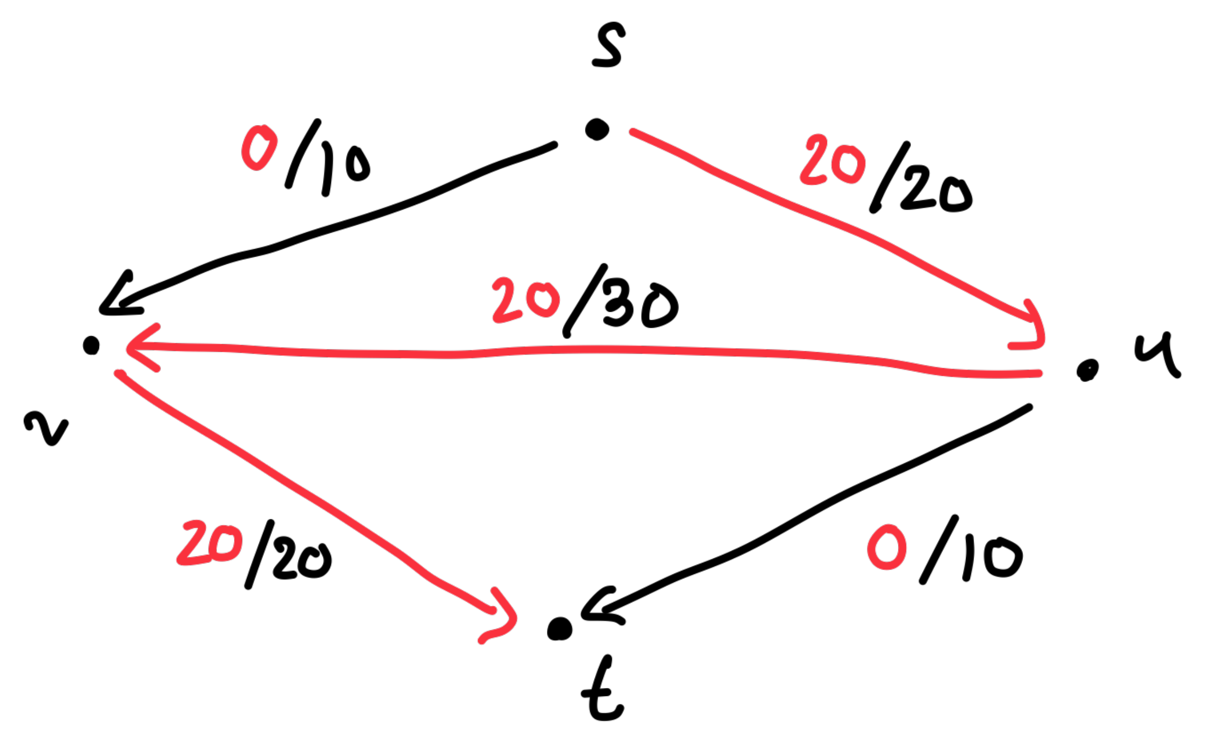
  The max flow is always $\leq$ the min cut.
- **Theorem:** If there exists a flow f and a cut (S, T) such that $v(f) = C(S, T)$ then f must be a maximal flow and (S, T) must be a minimizing cut.
- **Proof:** $v(f_{\max}) \geq v(f)$ and $C(S_{\min}, T_{\min}) \leq C(S, T)$. This with $v(f) = C(S, T)$ sandwiches everything to get an equal max flow and min cut.



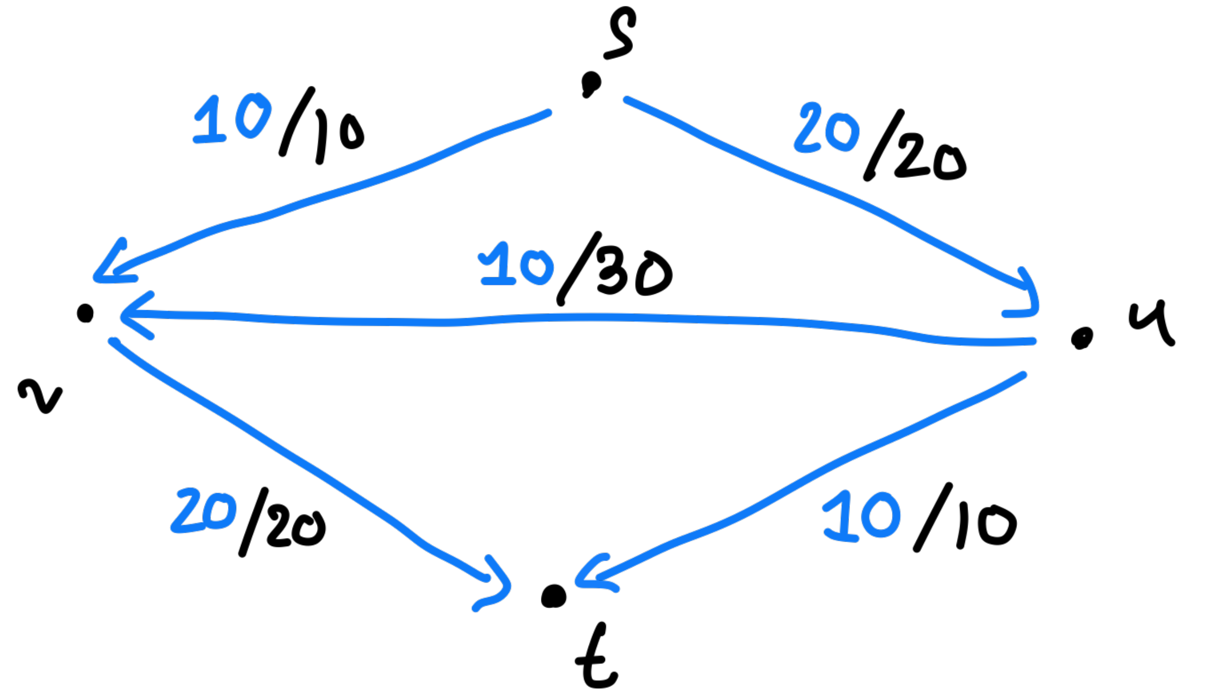
Algorithms for max flow

- **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 $f(e) \leq c(e)$,
    - "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:

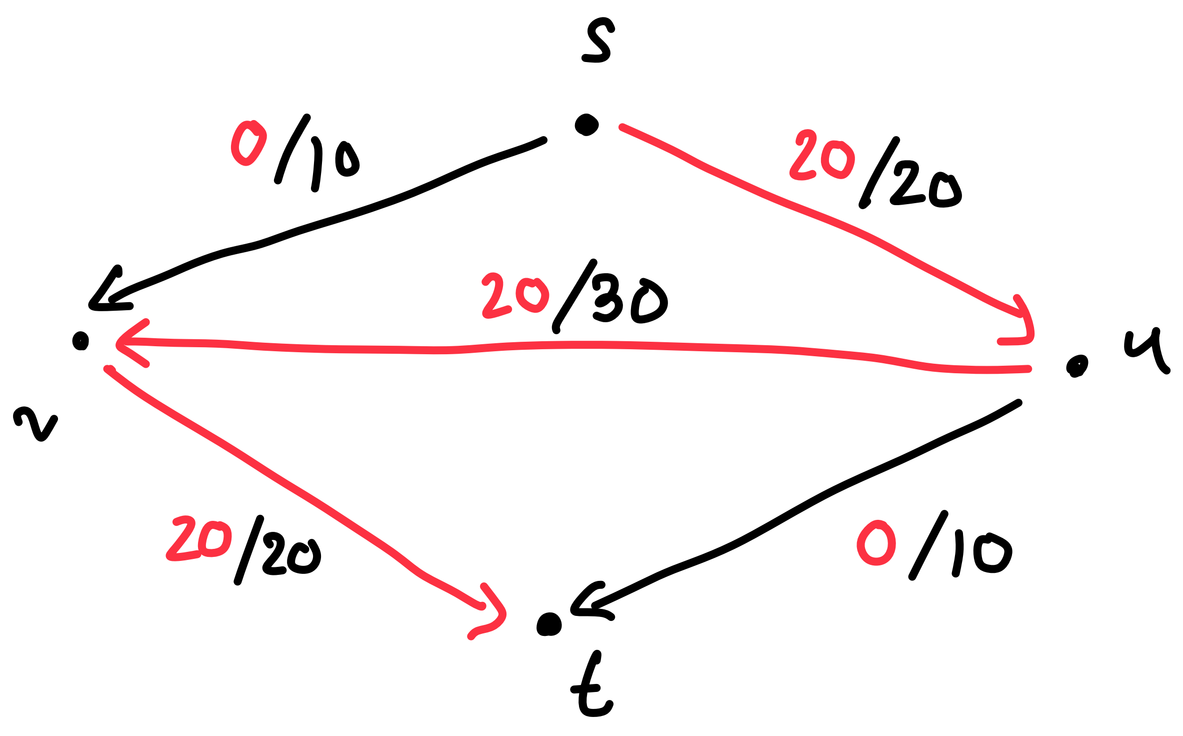


Greedy algorithms get stuck

- 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 $c(e) - f(e)$ additional flow
    - Or remove up to $f(e)$ flow from this edge.

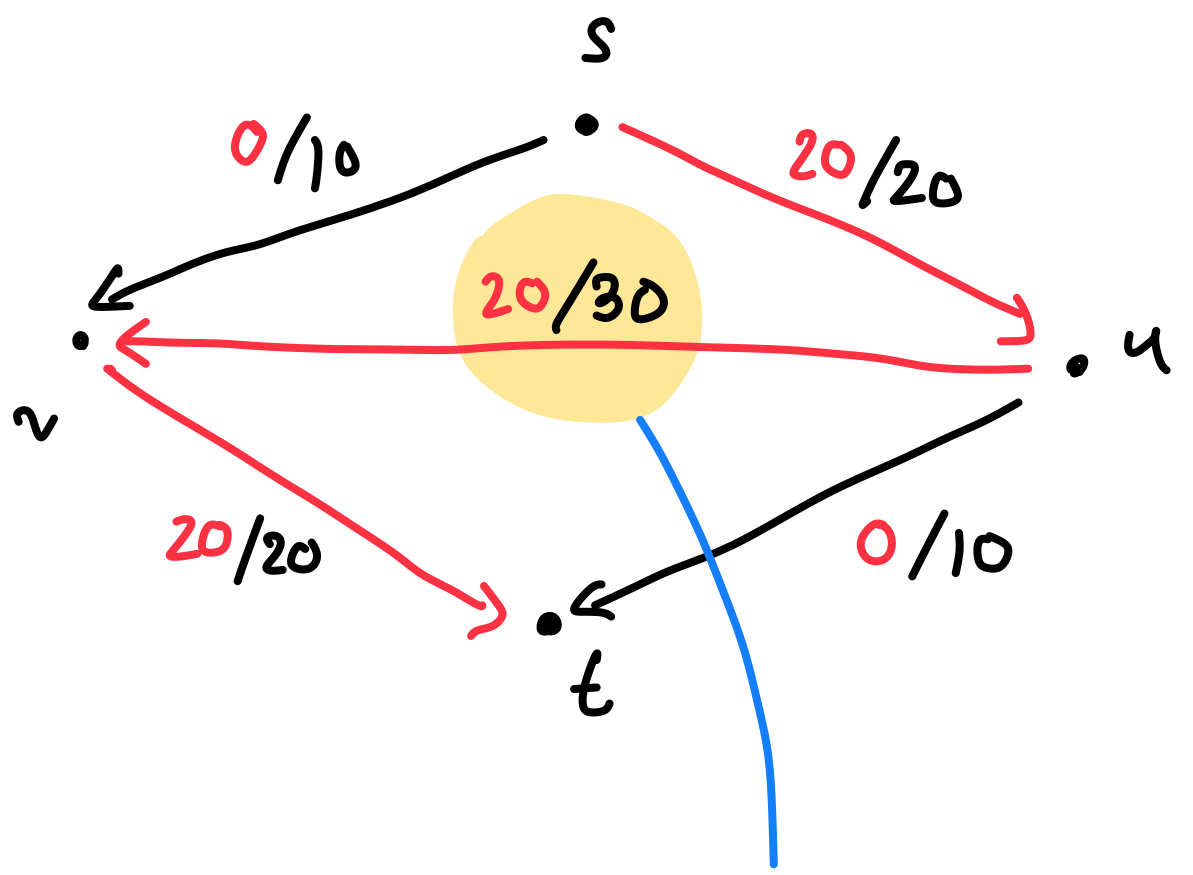
Augmenting paths through residual flow

Current choice of flow:



Augmenting paths through residual flow

Current choice of flow:

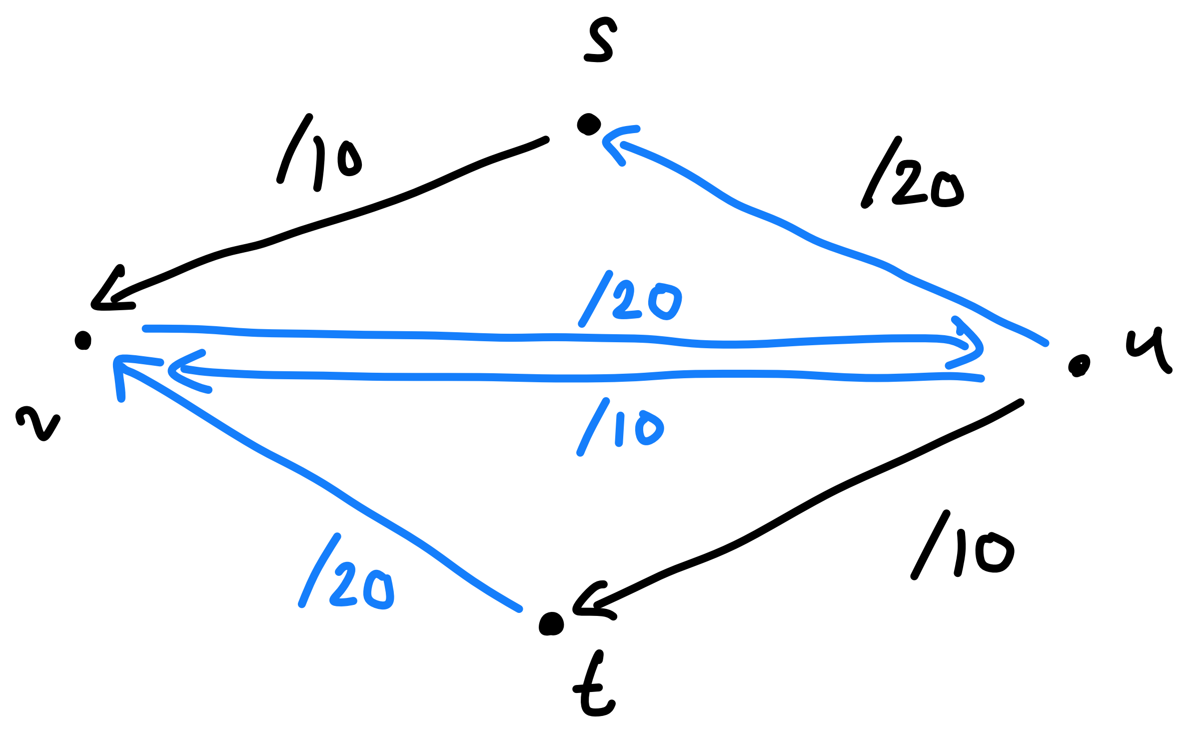


Can either add 10 flow to the left ←

or

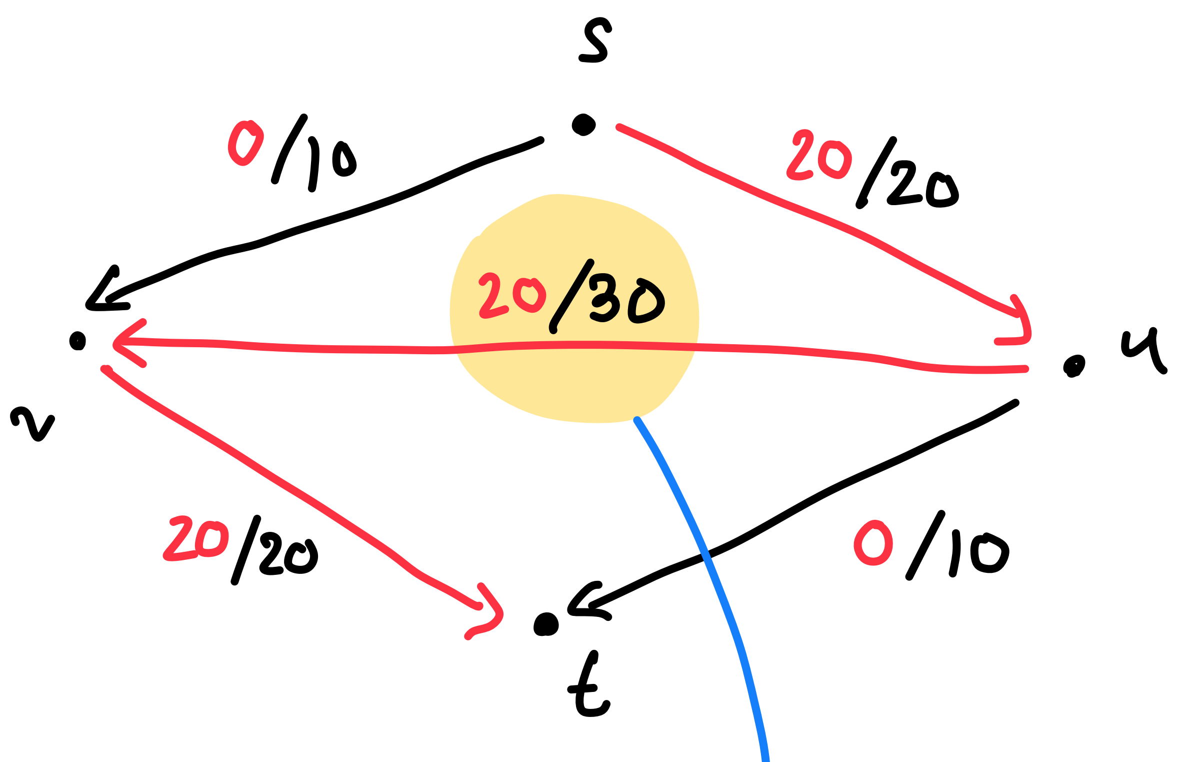
remove 20 flow, i.e. add 20 flow to the right ⟶

Residual flow network



Augmenting paths through residual flow

Current choice of flow:

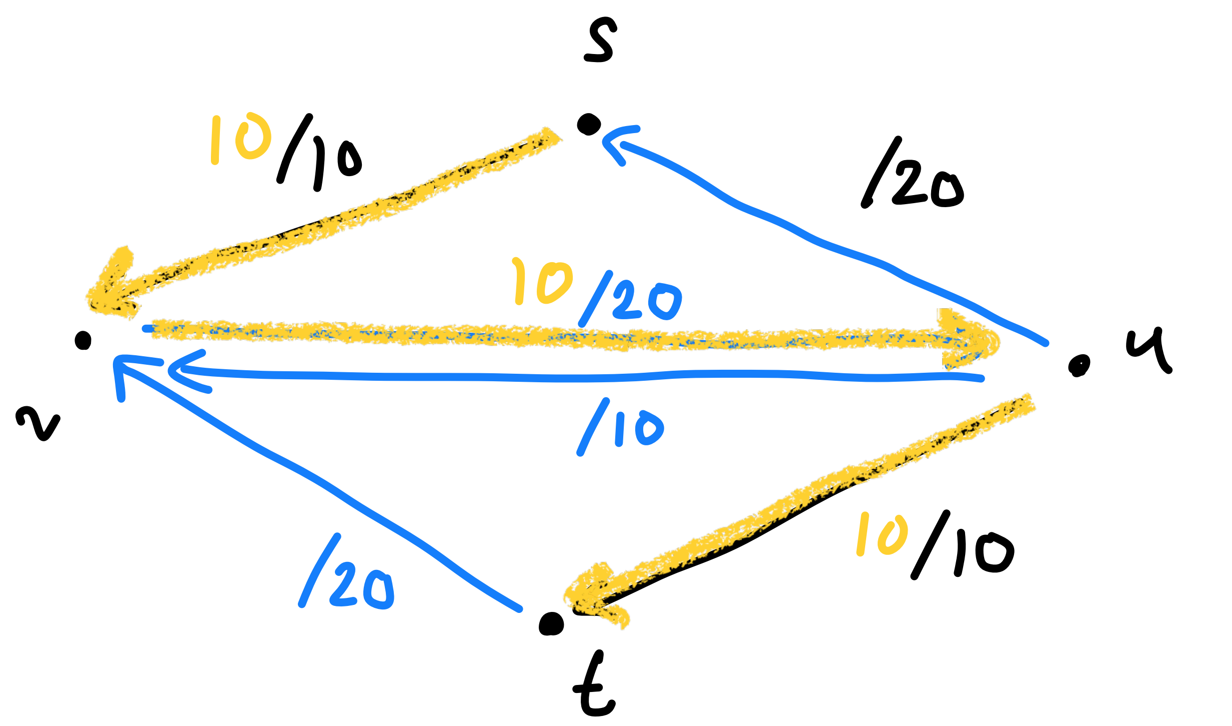


Can either add 10 flow to the left ←

or

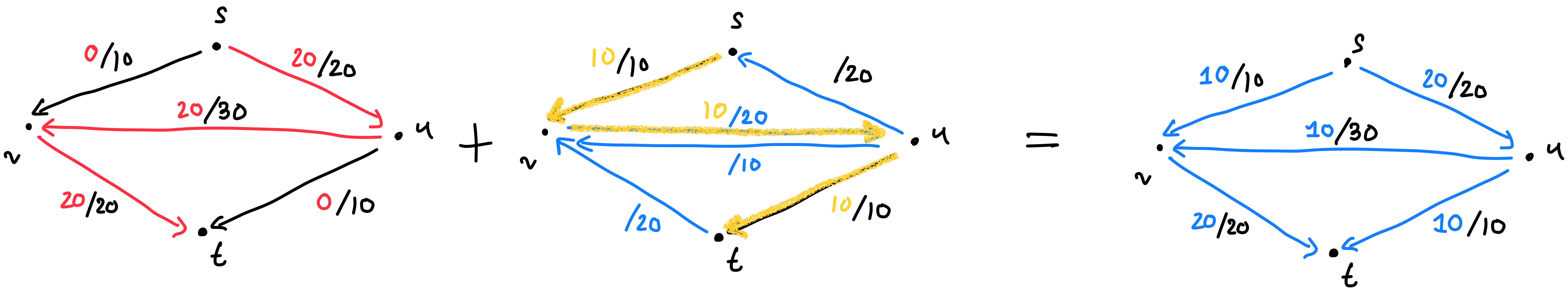
remove 20 flow, i.e. add 20 flow to the right →

Residual flow network



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

Augmenting paths through residual flow



original flow

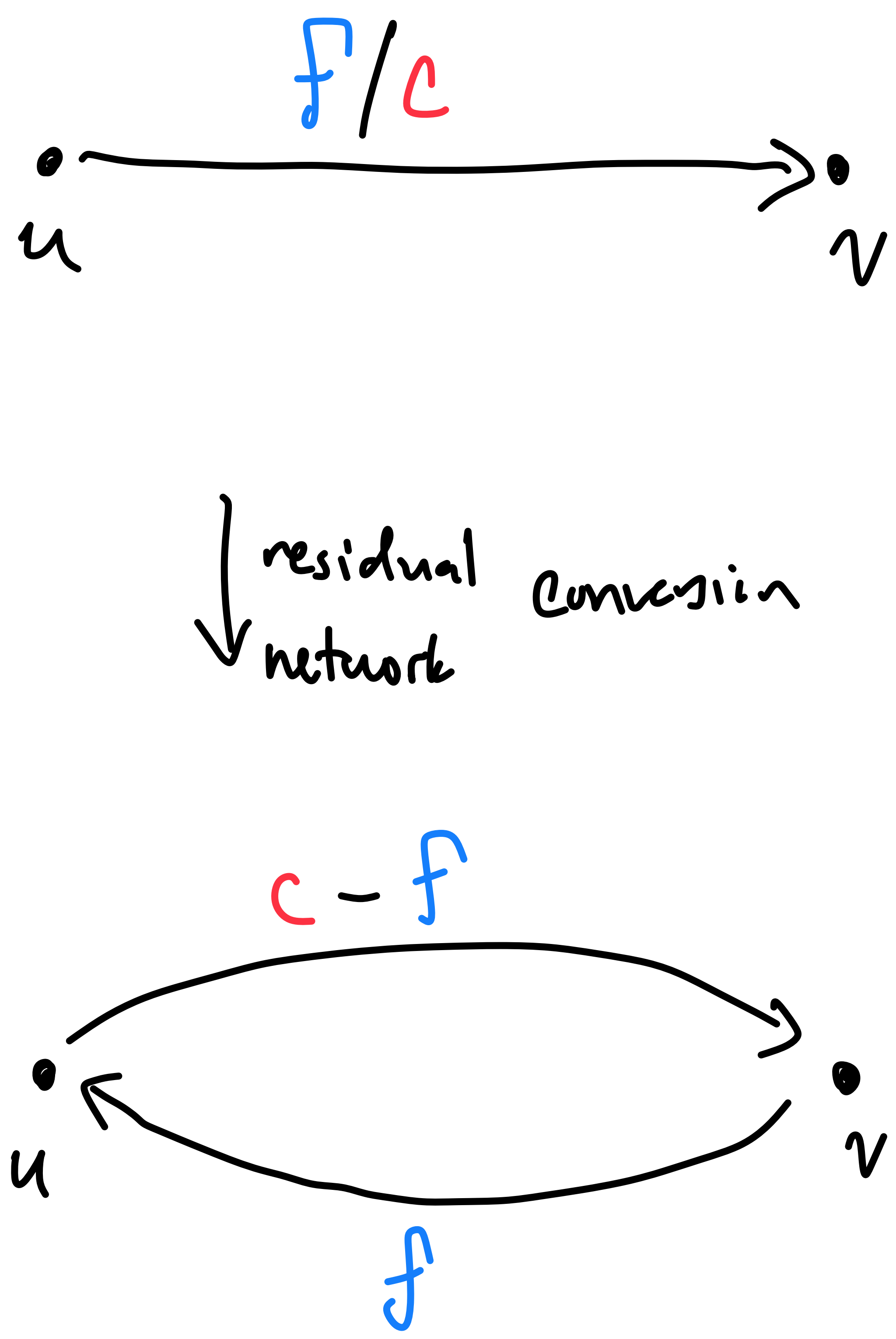
augmenting path in residual network

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 $e = (u \rightarrow v)$,
    - (Forward edge): Add an edge $u \rightarrow v$ of capacity $c(e) - f(e)$
    - (Backward edge): Add an edge $v \rightarrow u$ of capacity $f(e)$



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 $f \leftarrow f + f_{\text{aug}}$ ← O(n) time.
  - Update G_f along the edges of p

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?

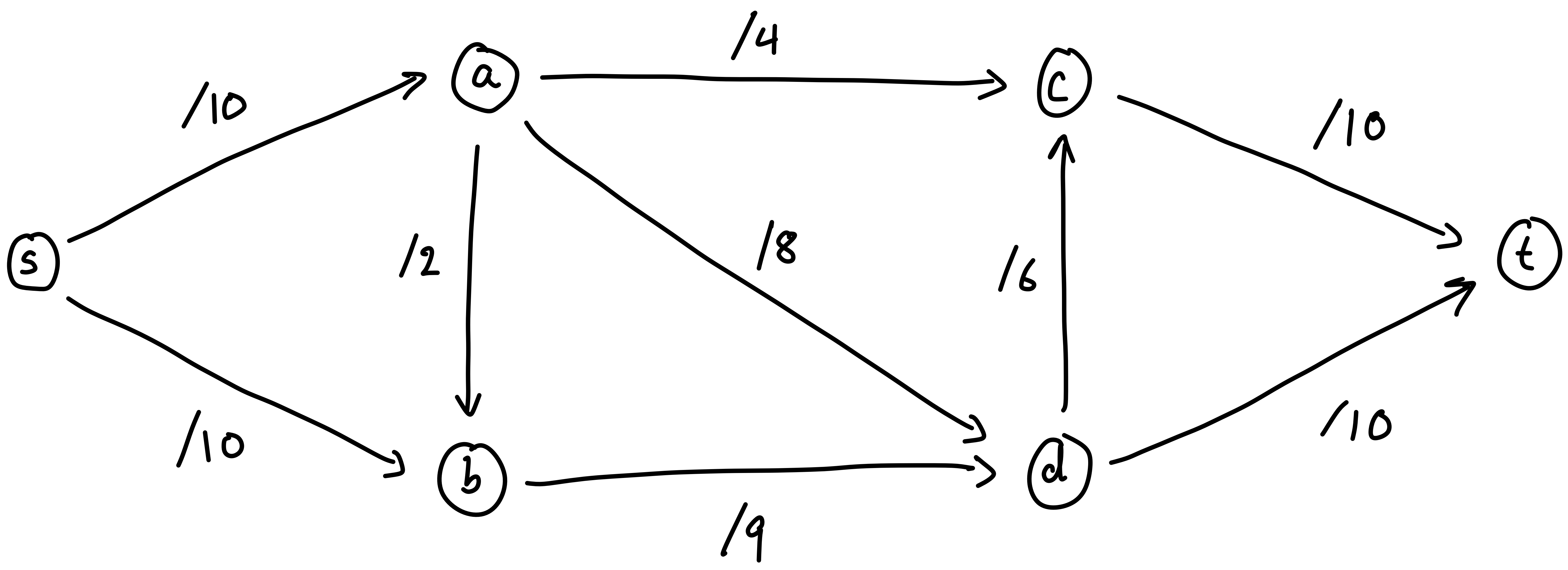
New greedy algorithm (Ford-Fulkerson)

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

call this the bottleneck capacity of the augmentation path p.

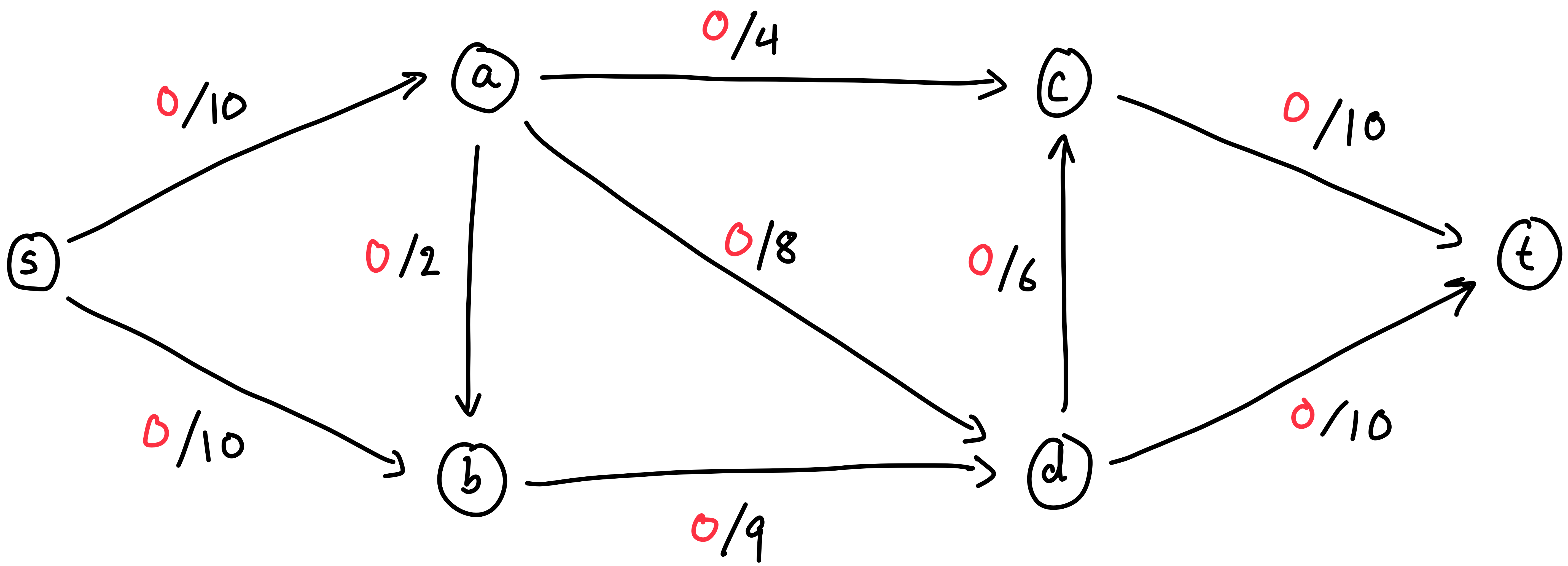
Ford-Fulkerson animation

Graph G and flow f :

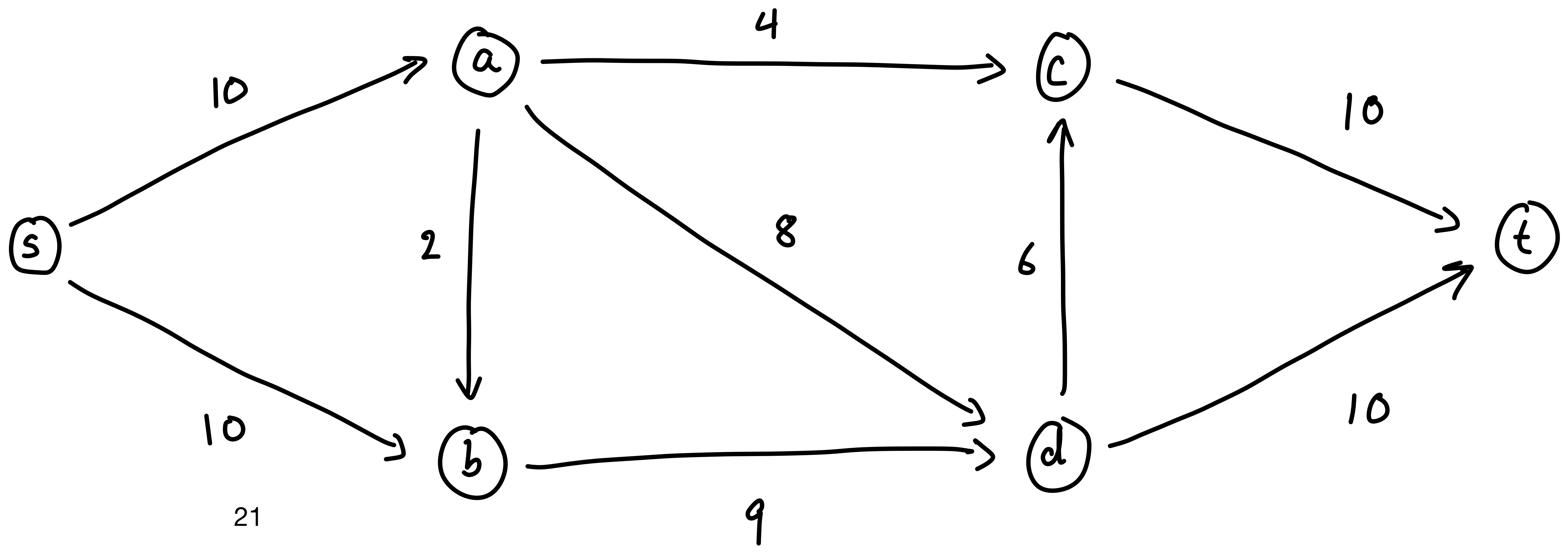


Ford-Fulkerson animation

Graph G and flow f :

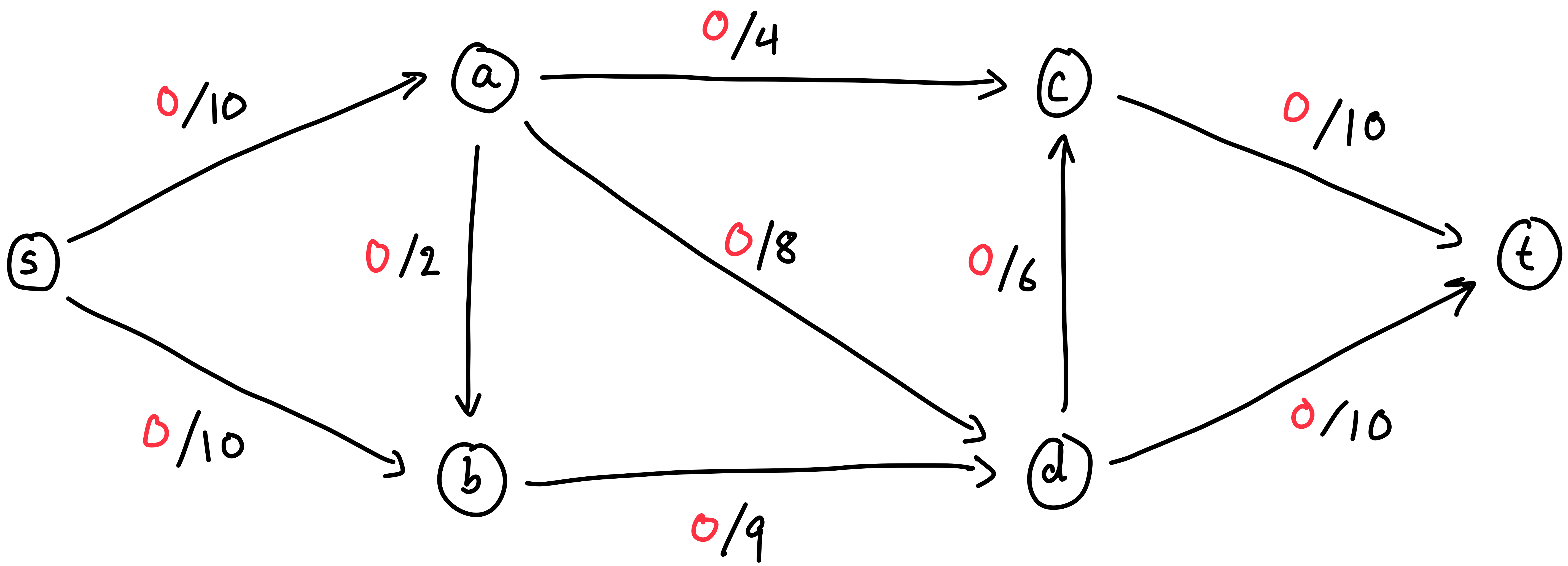


Residual graph G_f :



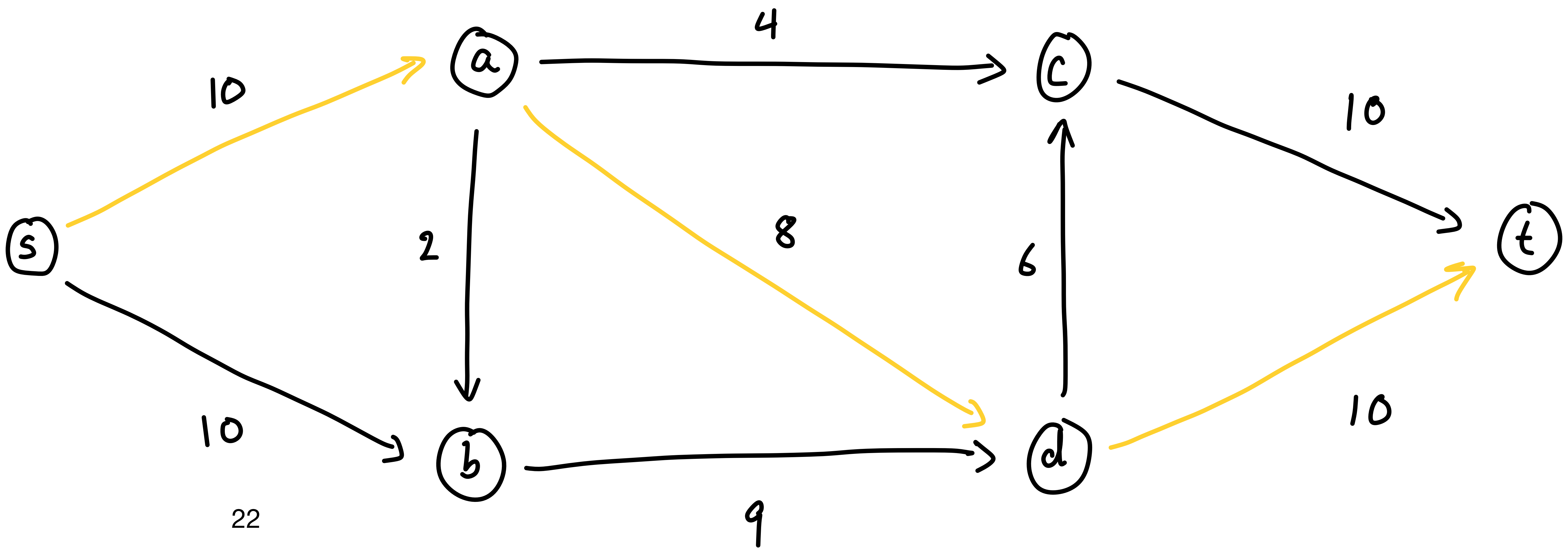
Ford-Fulkerson animation

Graph G and flow f:



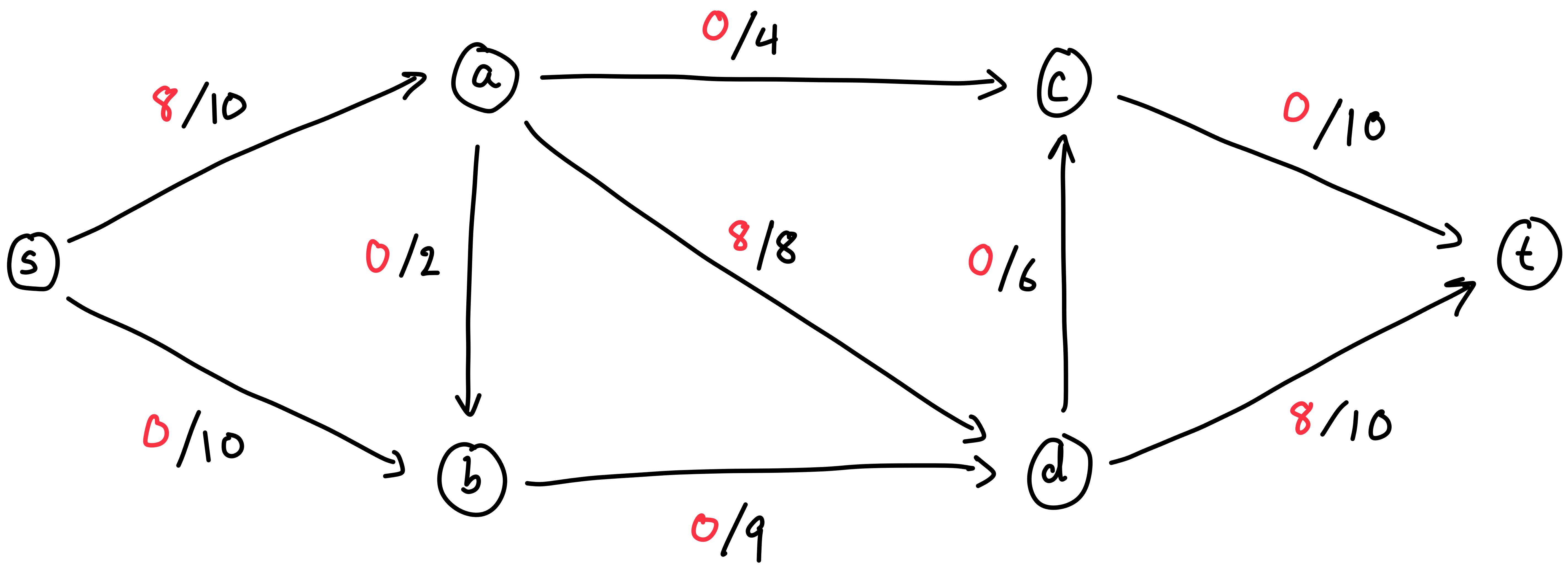
Found a path of bottleneck capacity 8.

Residual graph G_f:

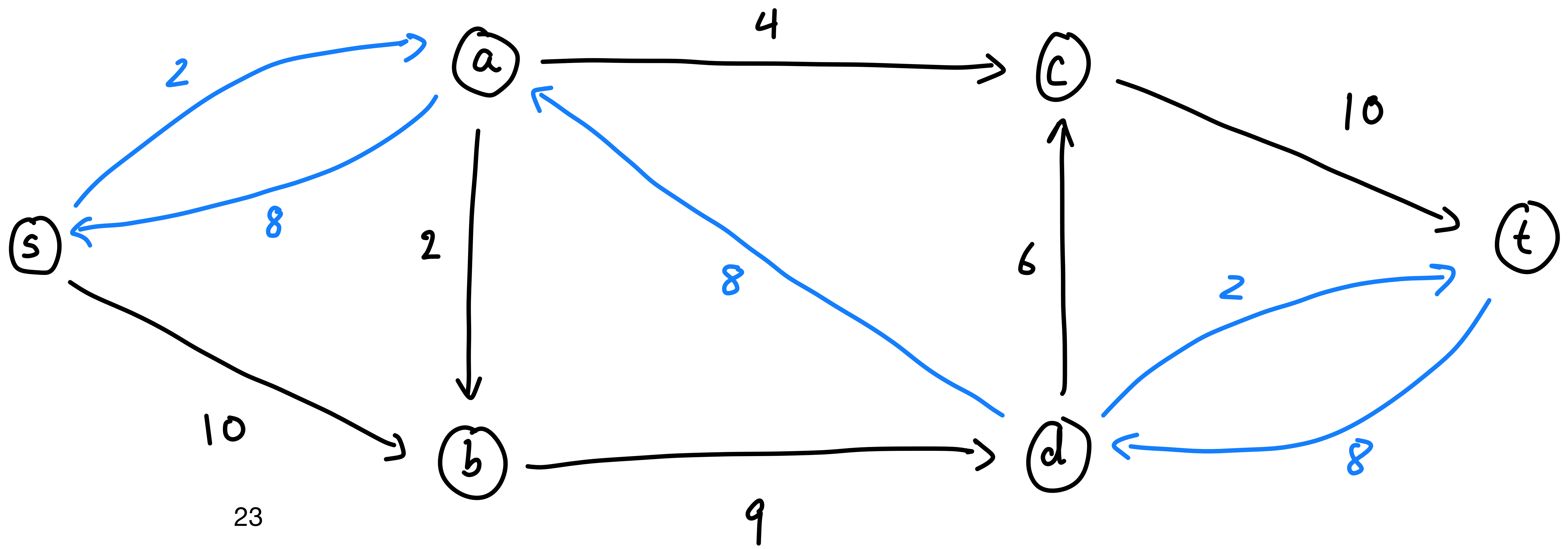


Ford-Fulkerson animation

Graph G and flow f :

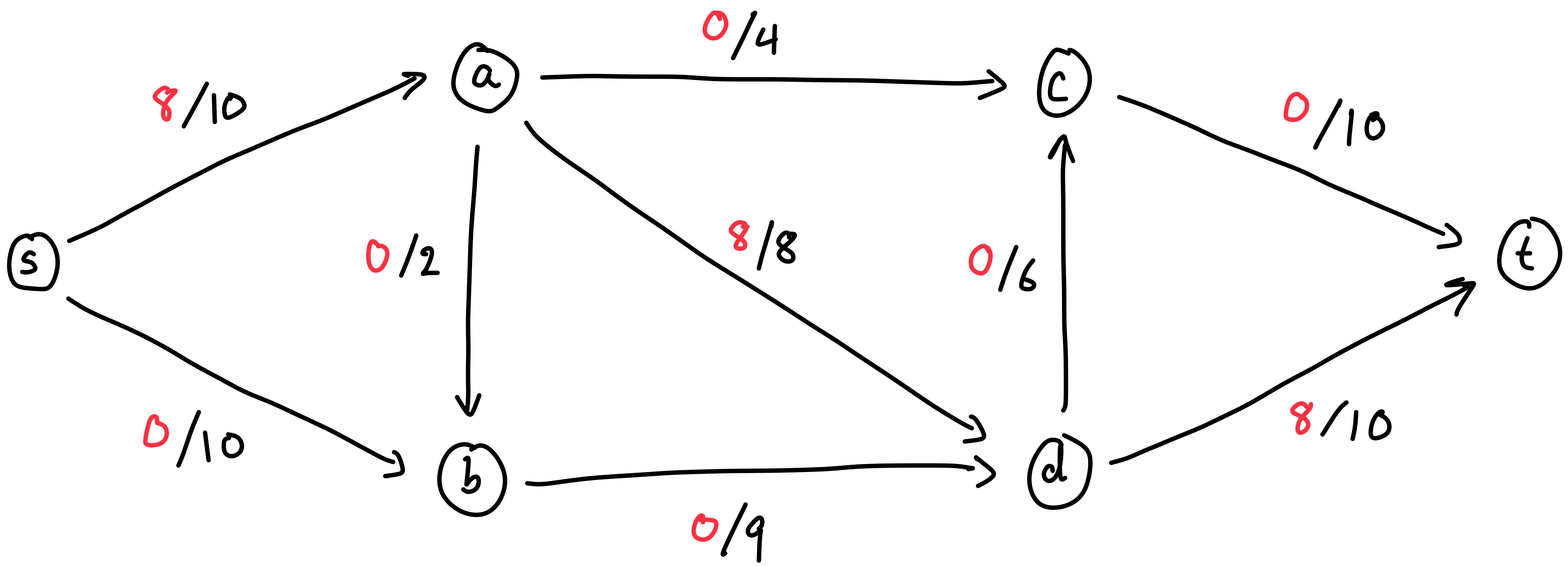


Residual graph G_f :



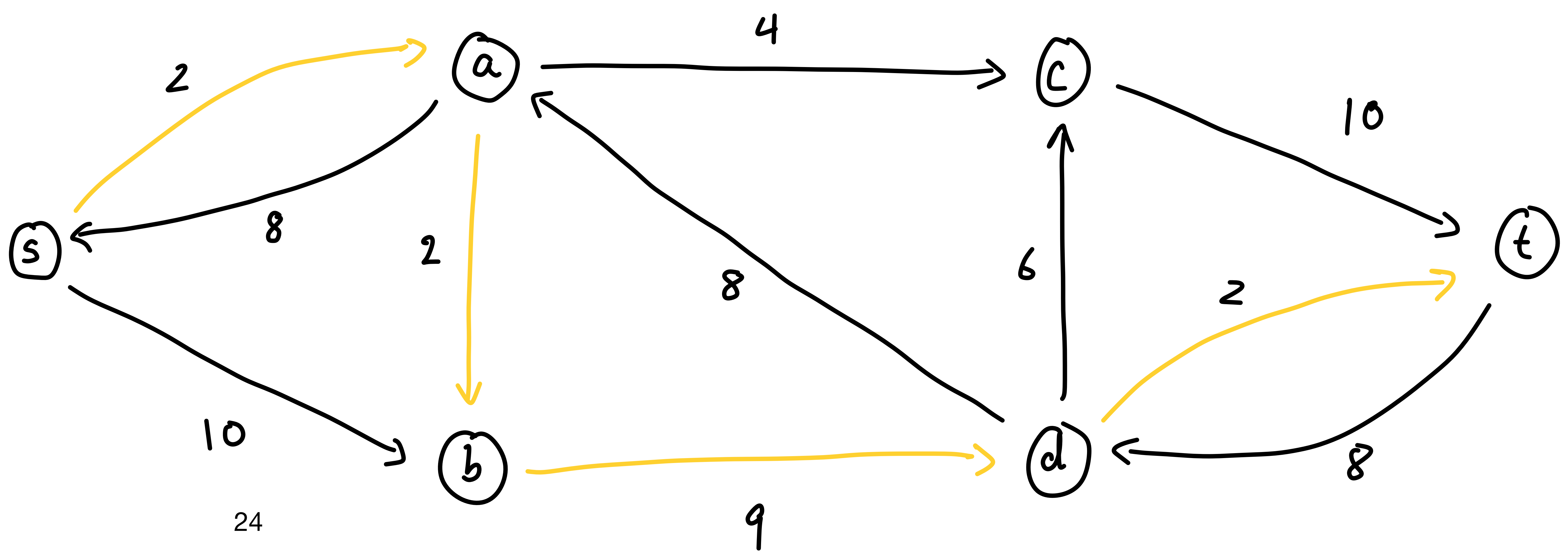
Ford-Fulkerson animation

Graph G and flow f :



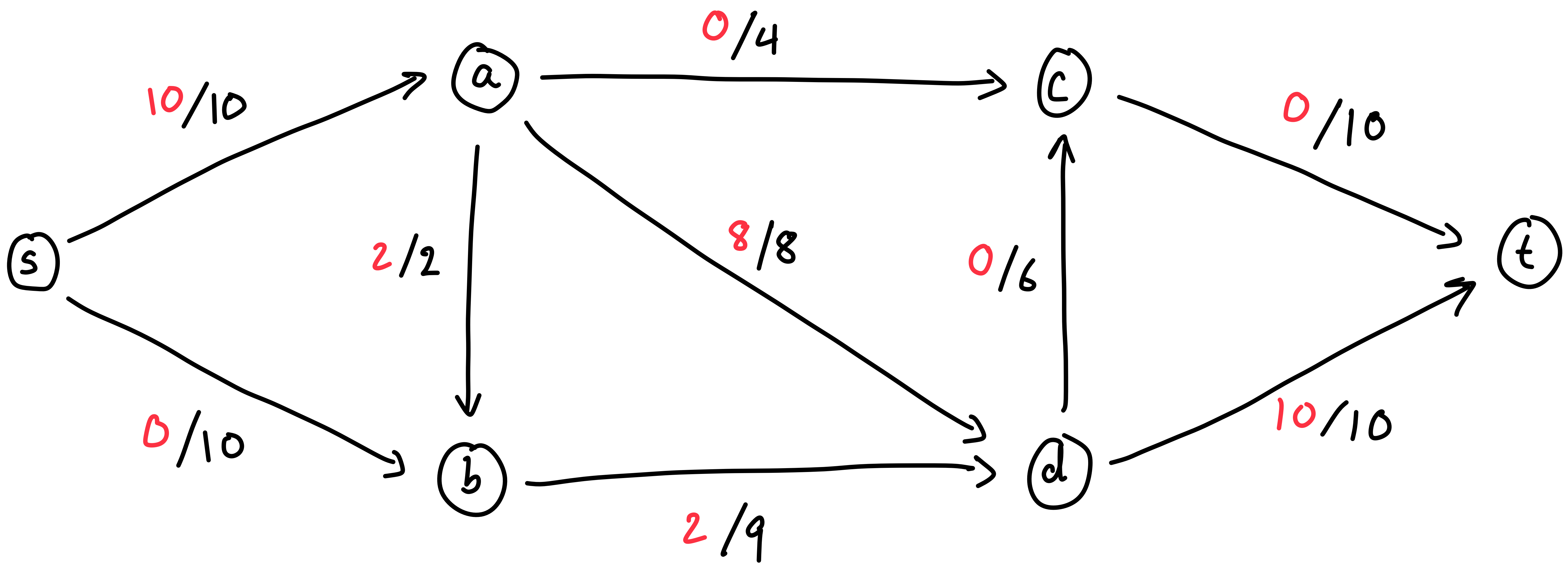
Found a path of bottleneck capacity 2.

Residual graph G_f :

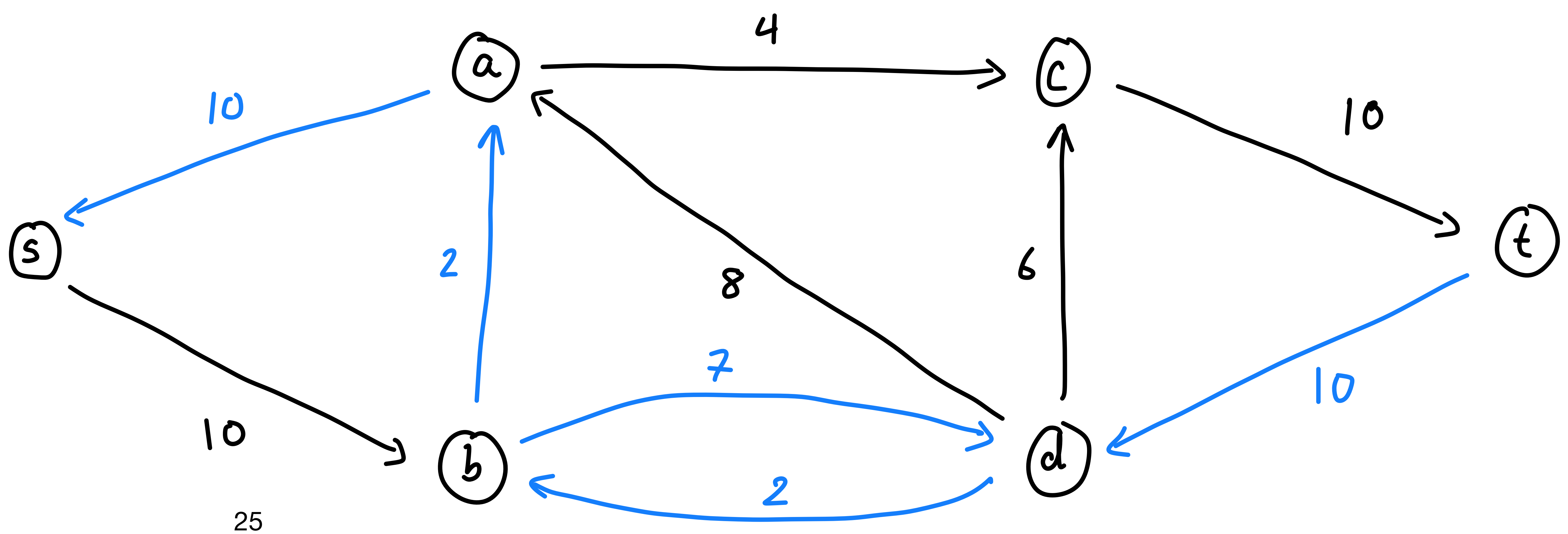


Ford-Fulkerson animation

Graph G and flow f:

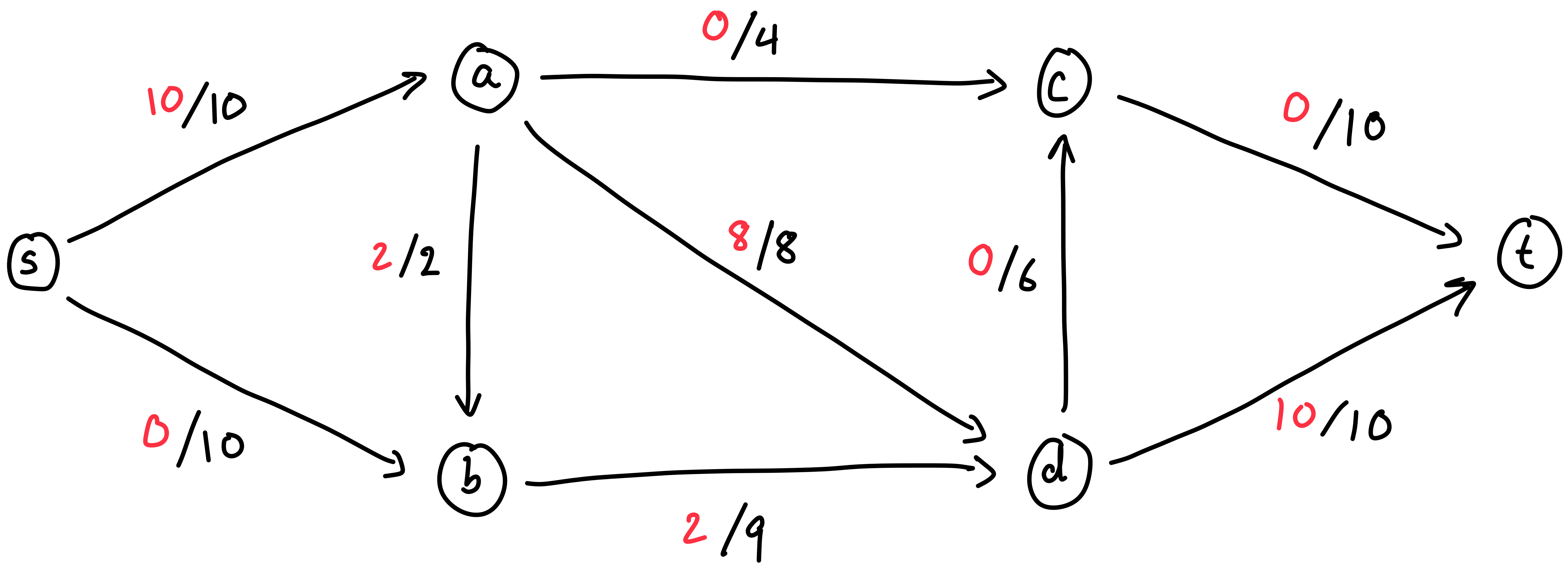


Residual graph G_f:



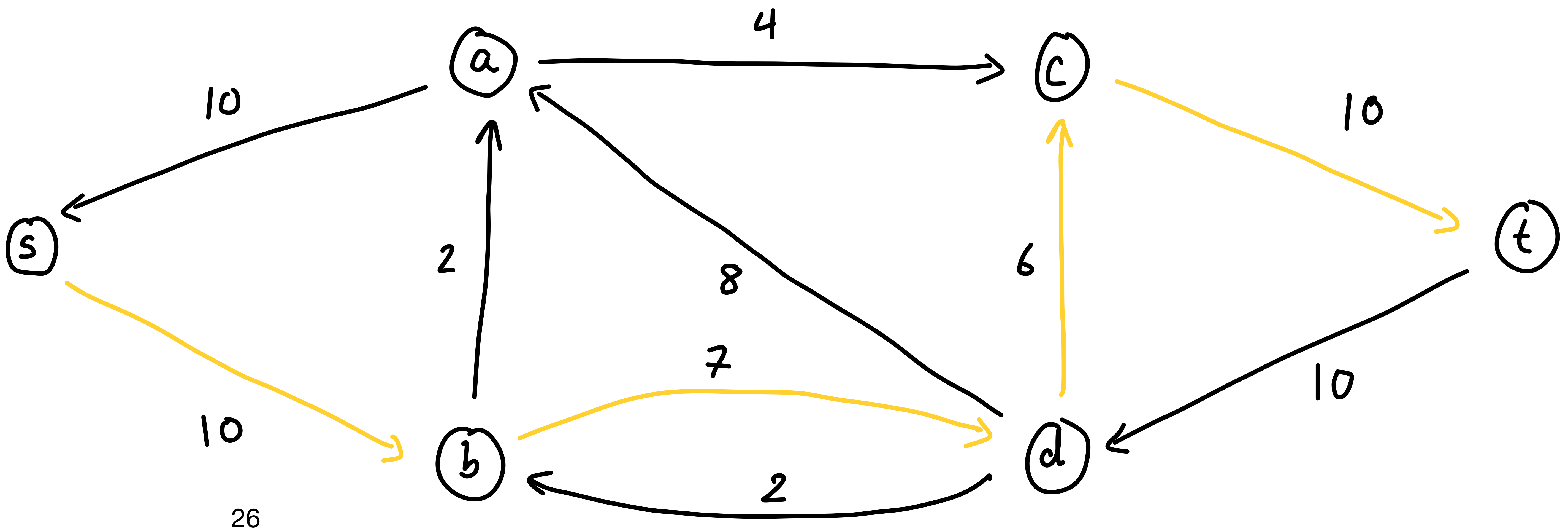
Ford-Fulkerson animation

Graph G and flow f :



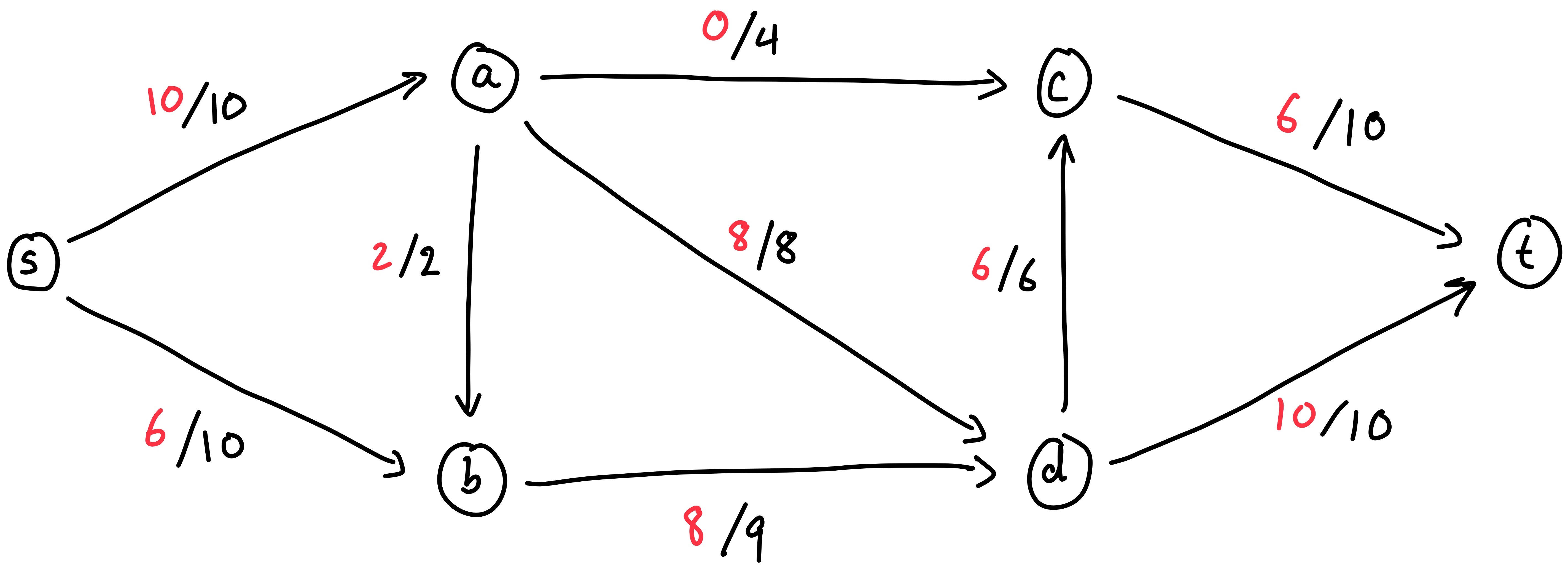
Found a path of bottleneck capacity 6.

Residual graph G_f :

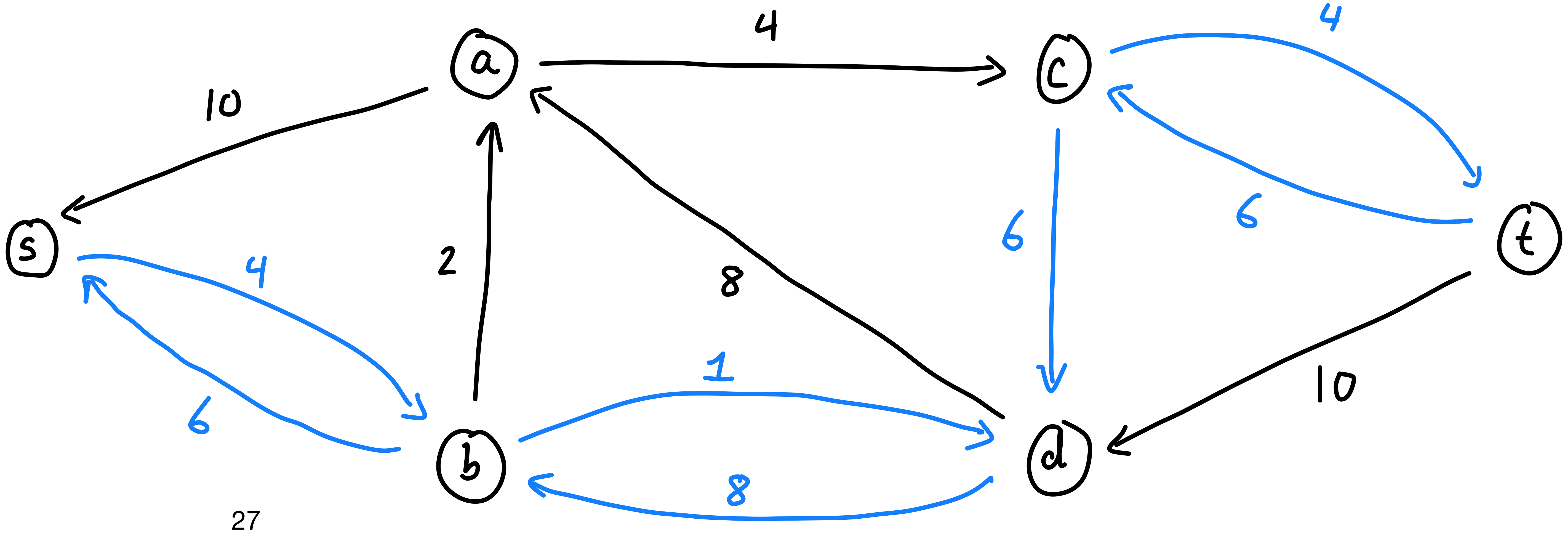


Ford-Fulkerson animation

Graph G and flow f :

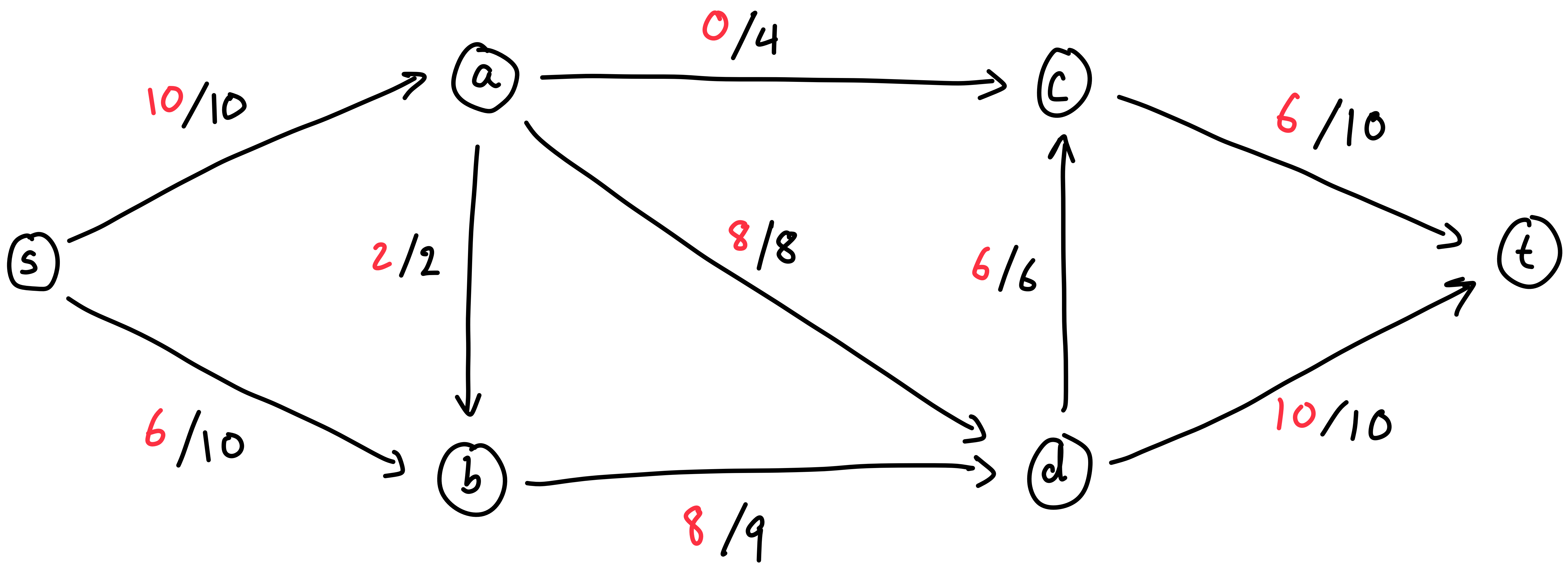


Residual graph G_f :



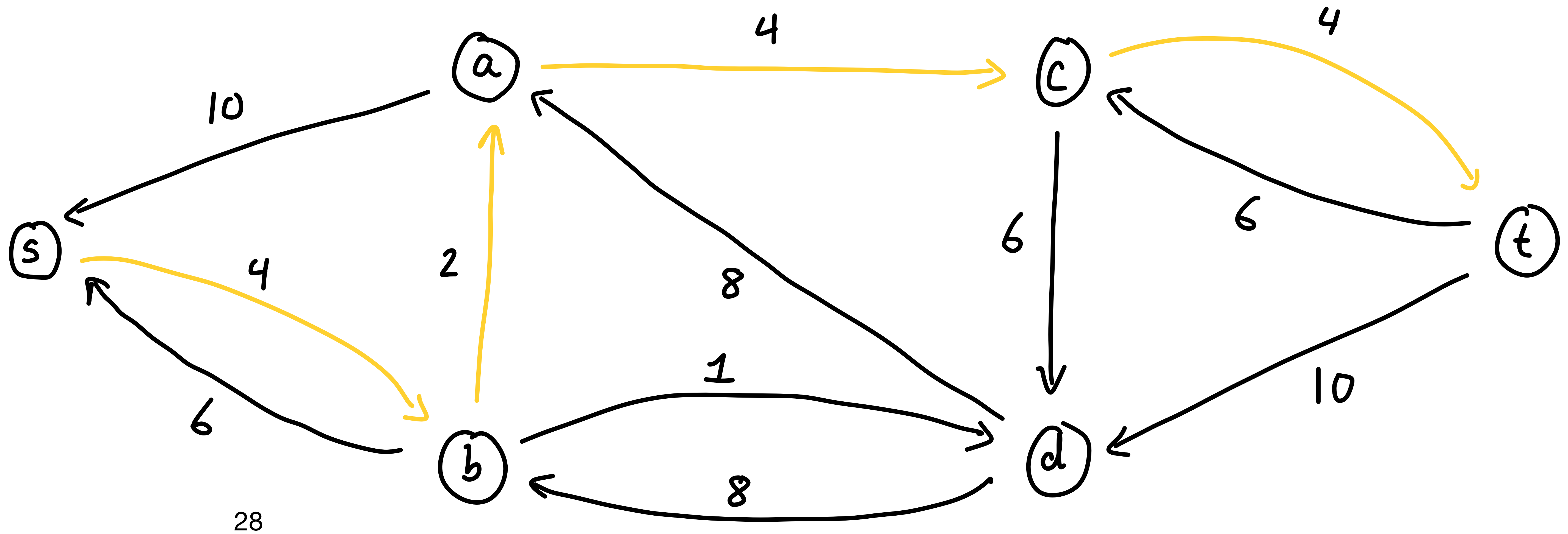
Ford-Fulkerson animation

Graph G and flow f:



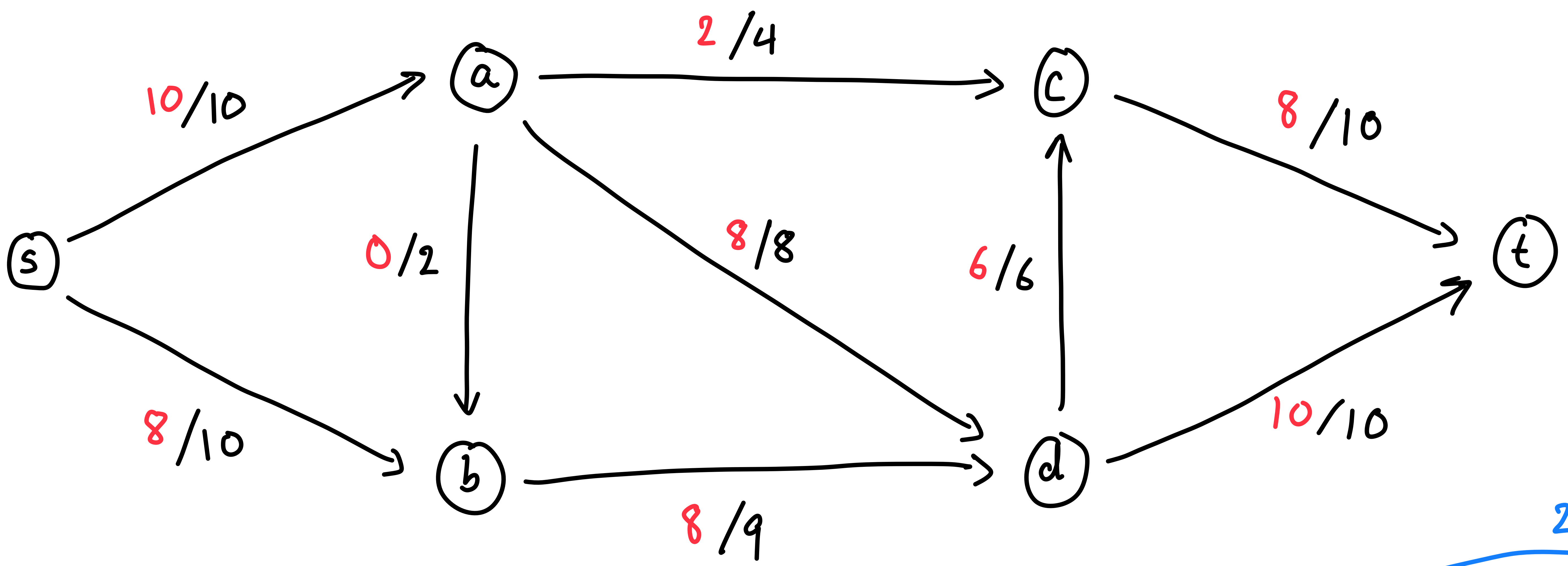
Found a path of bottleneck capacity 2.

Residual graph G_f:

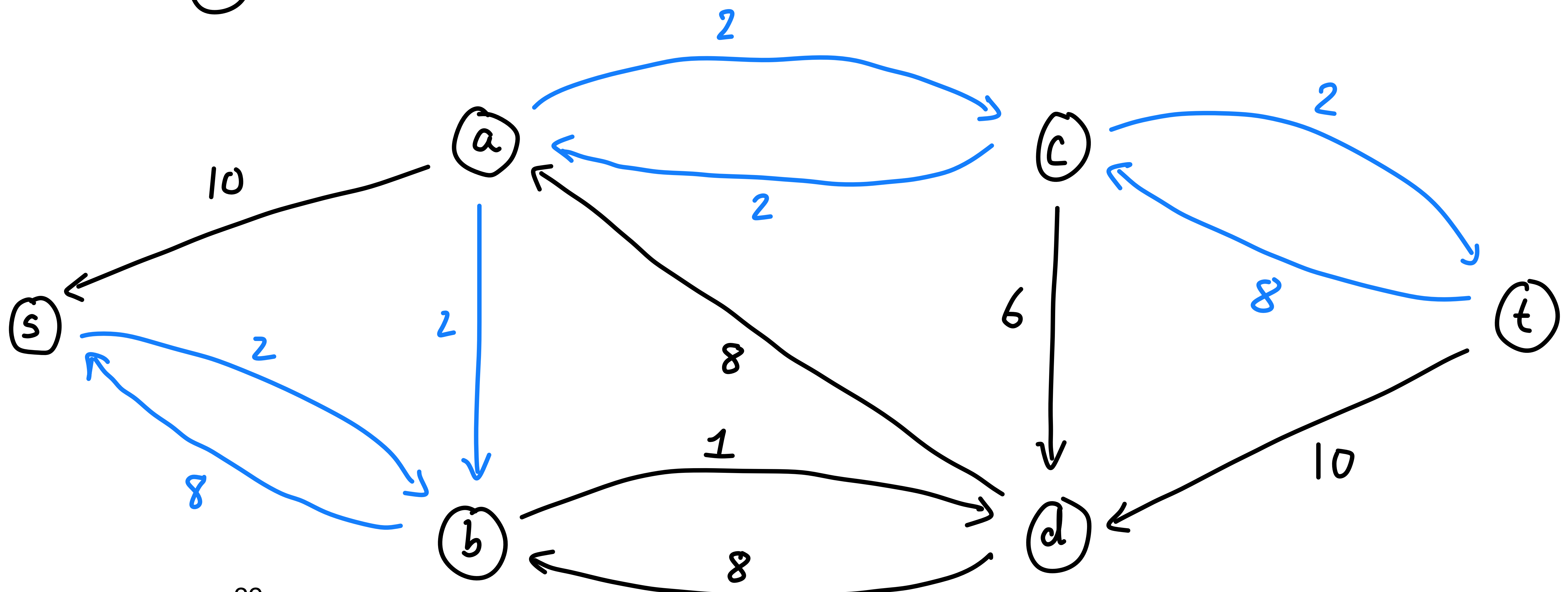


Ford-Fulkerson animation

Graph G and flow f:

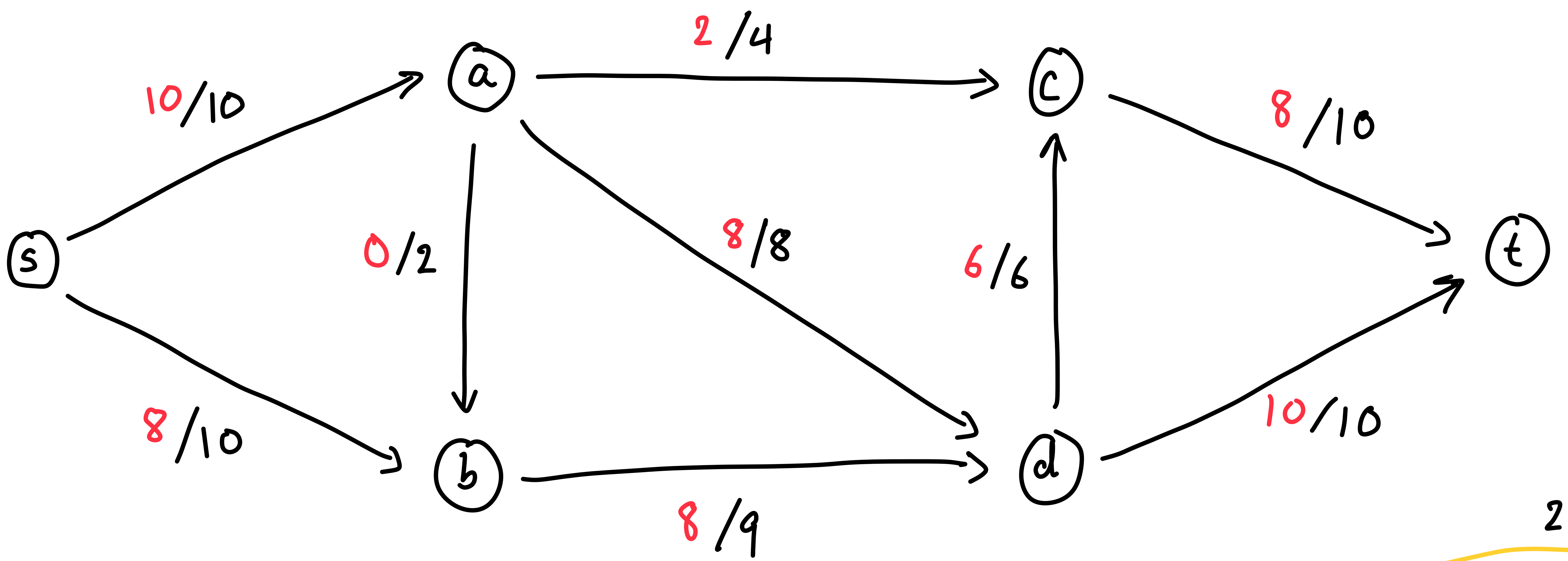


Residual graph G_f:



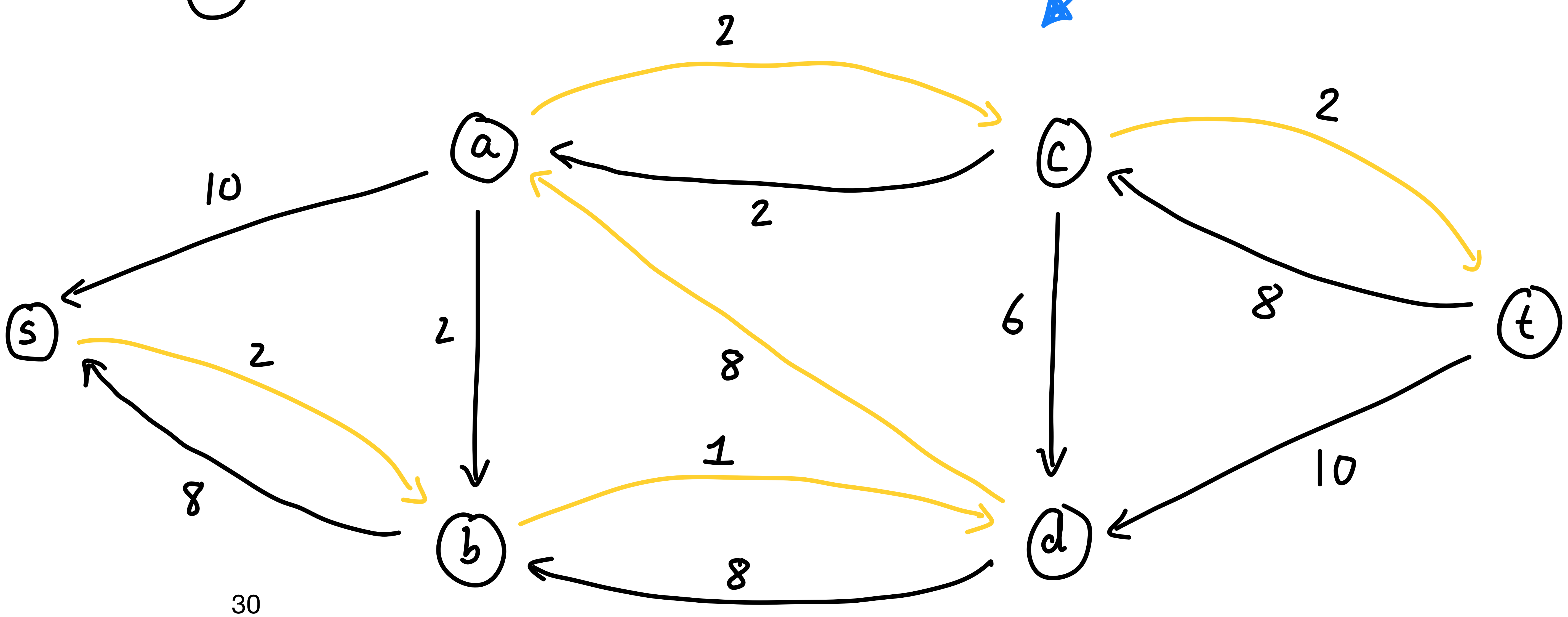
Ford-Fulkerson animation

Graph G and flow f :



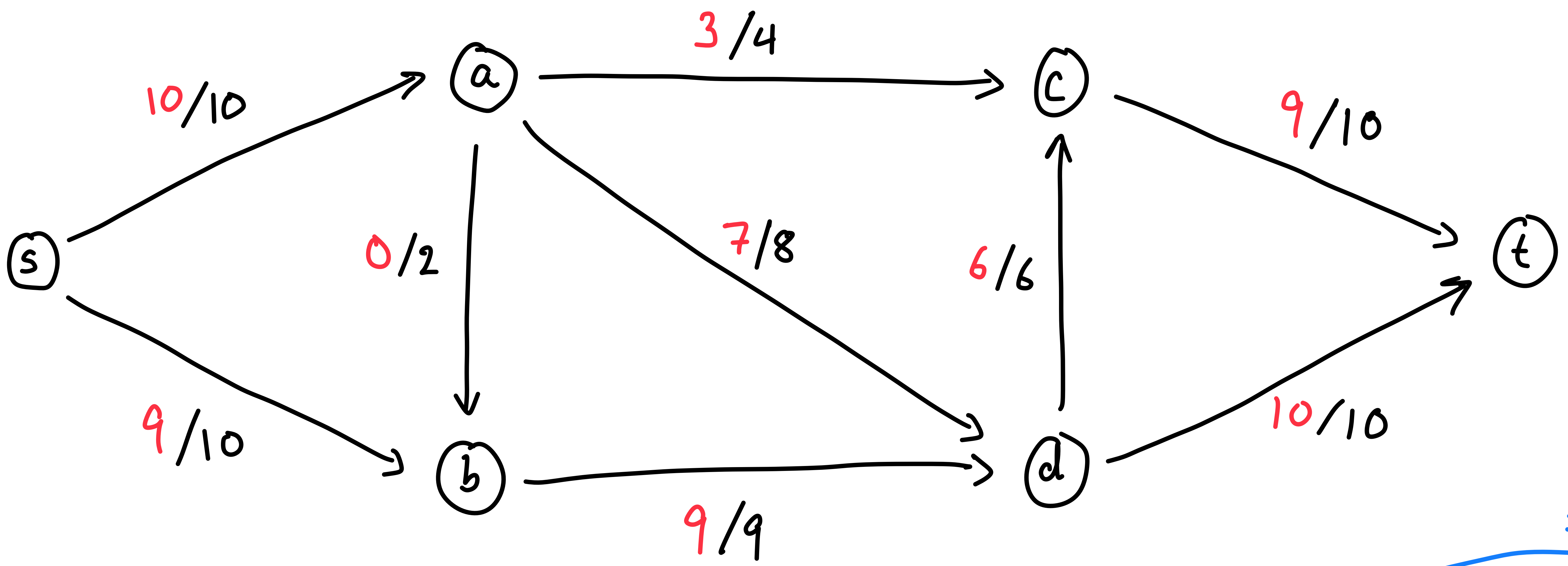
Found a path of bottleneck capacity 1.

Residual graph G_f :

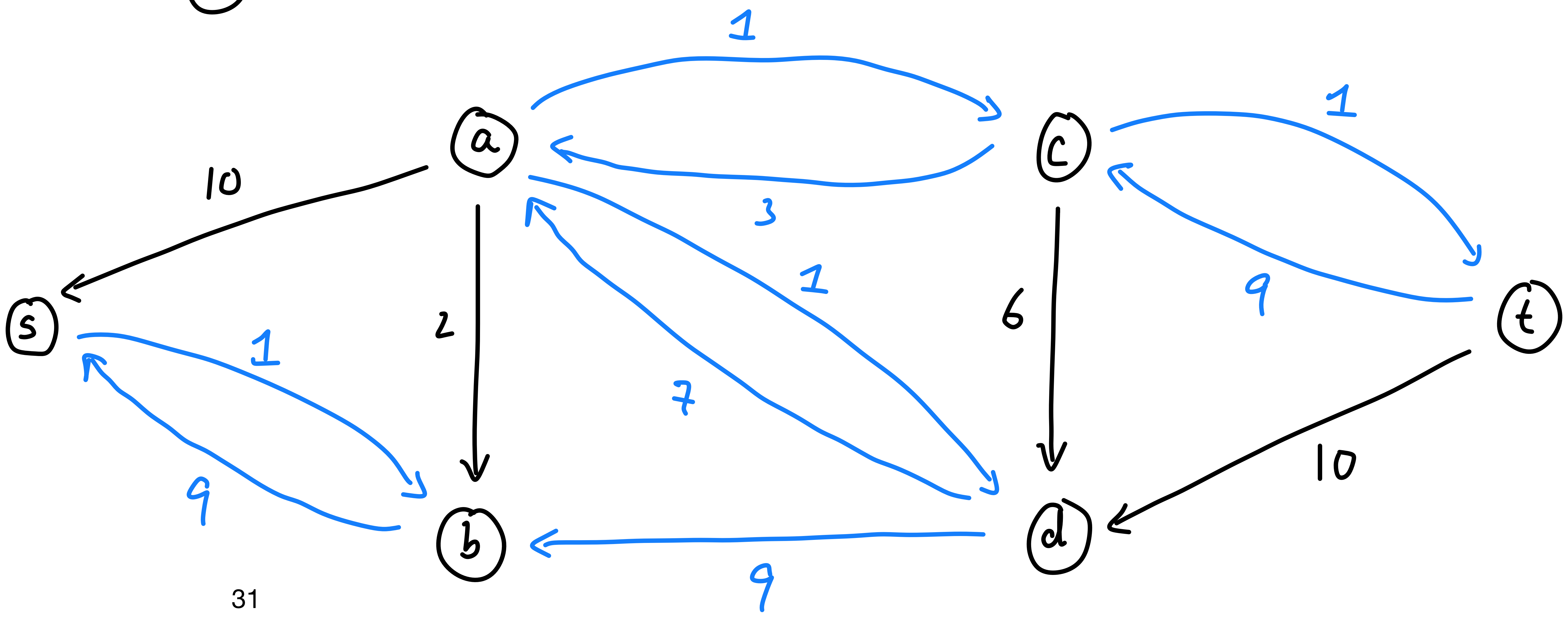


Ford-Fulkerson animation

Graph G and flow f:

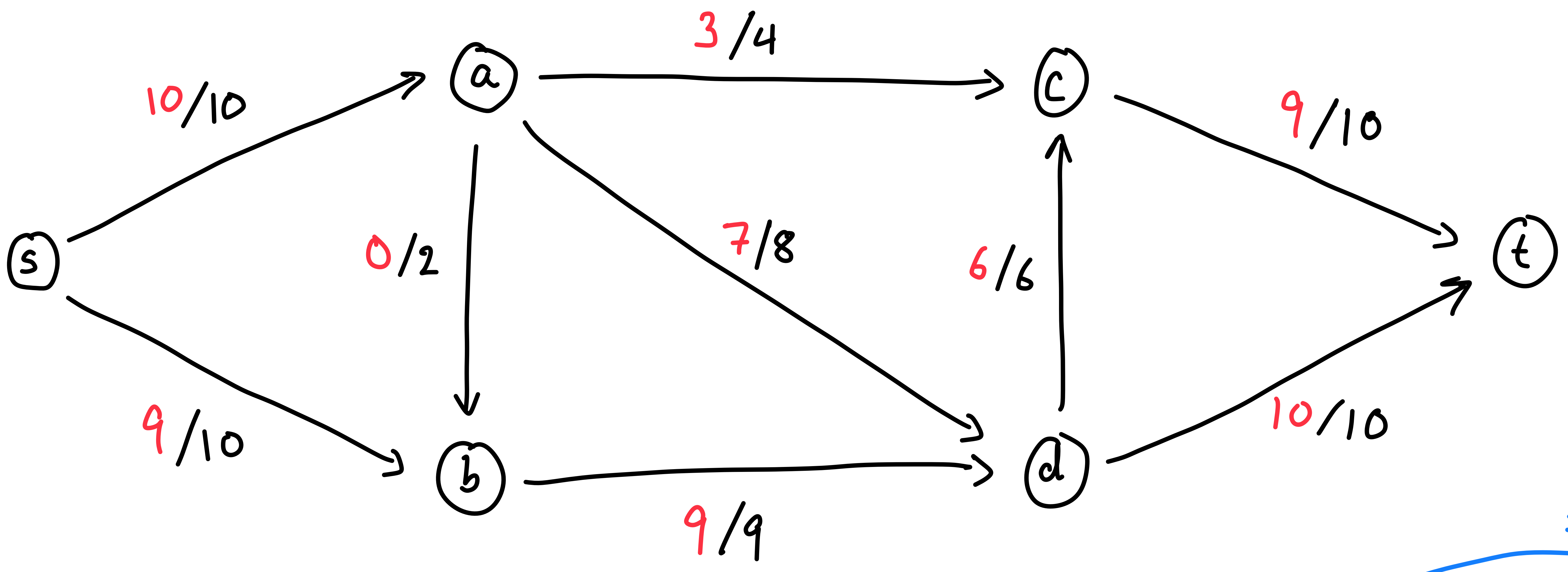


Residual graph G_f:



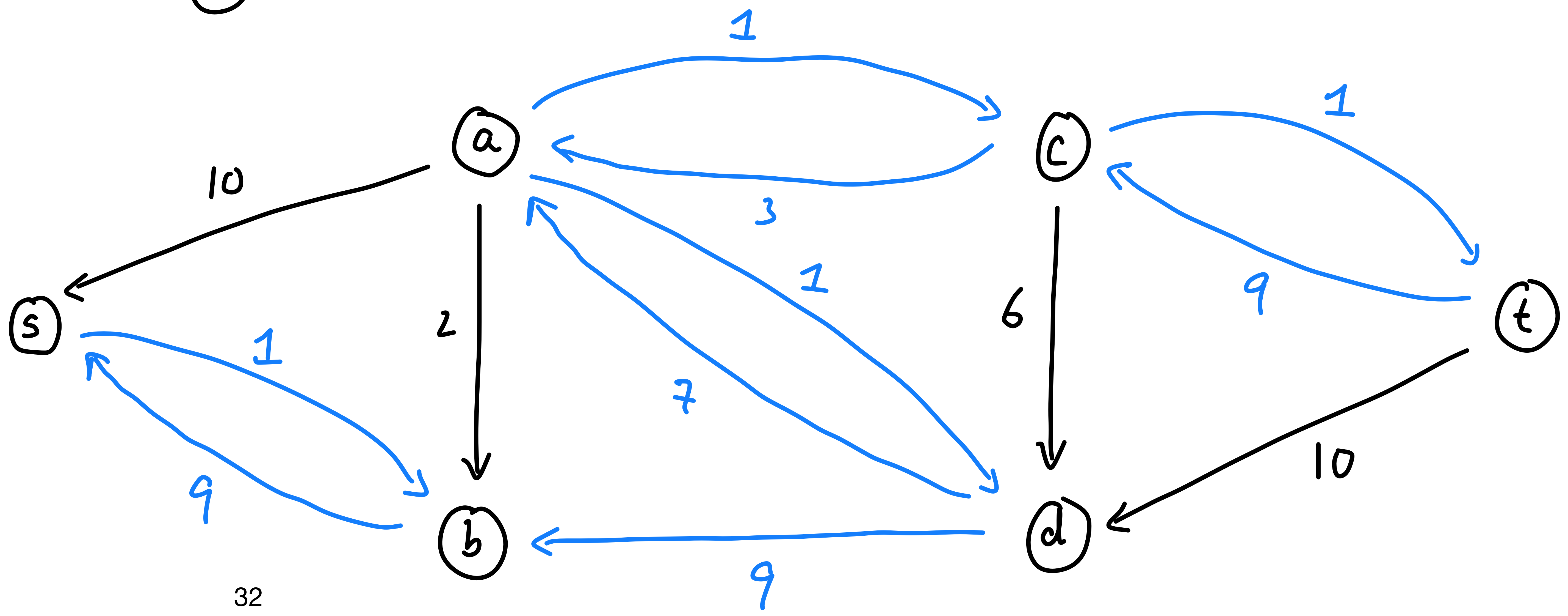
Ford-Fulkerson animation

Graph G and flow f:



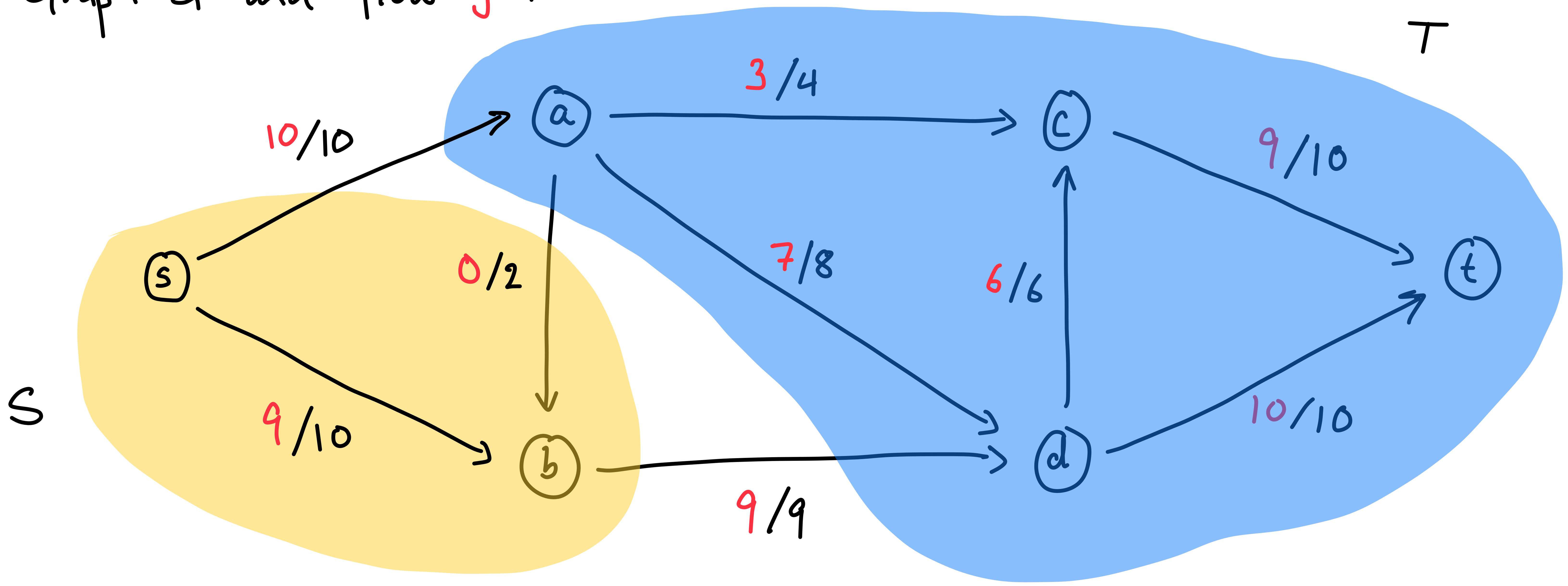
No augmenting path exists. Ford-Fulkerson terminates with an output flow of 19.

Residual graph G_f:



Ford-Fulkerson animation

Graph G and flow f :



Notice, $v(f) = c(S,T)$. By weak duality, this is an optimal flow.

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 due time.
  - However, we can taking a second to observe that it will output a valid flow!
  - Optimality, will require some work.

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:** If f is a flow in G and f_{aug} is an augmenting flow in G_f, then $f + f_{\text{aug}}$ 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.