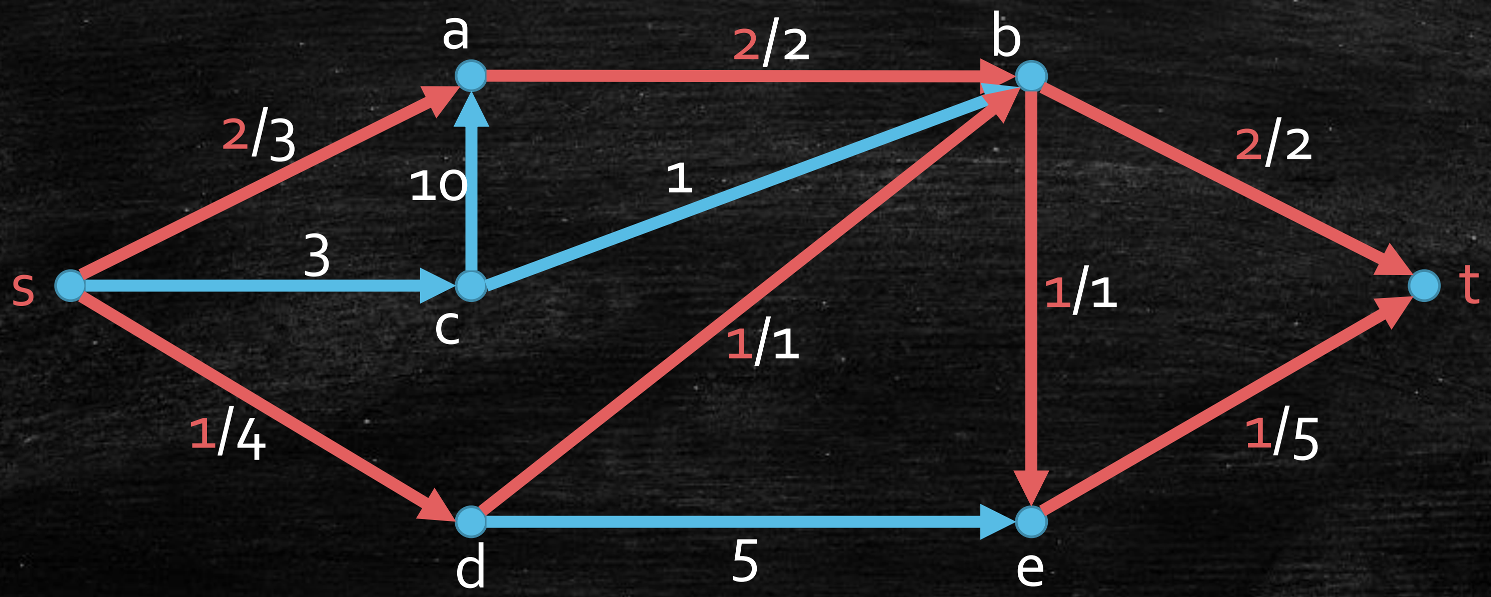
Network Flow

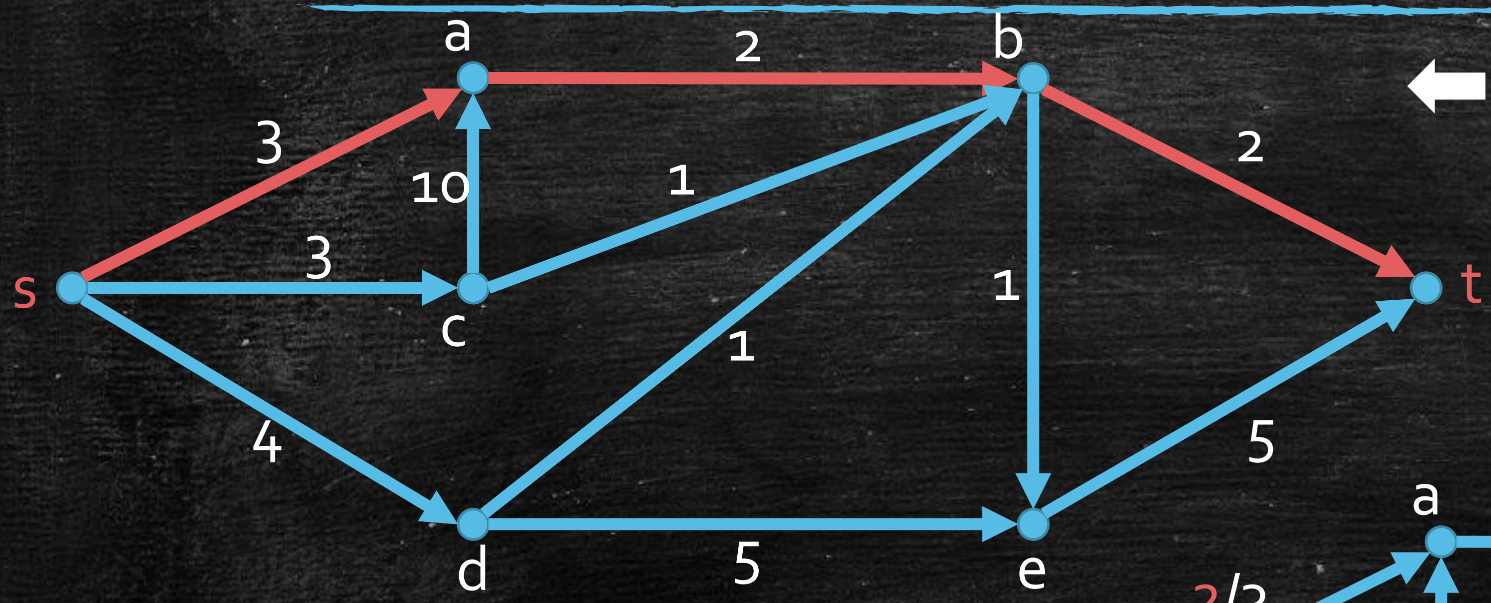
Max-Flow-Min-Cut Theorem, Max-Matching on Bipartite Graphs

Flow-Definition

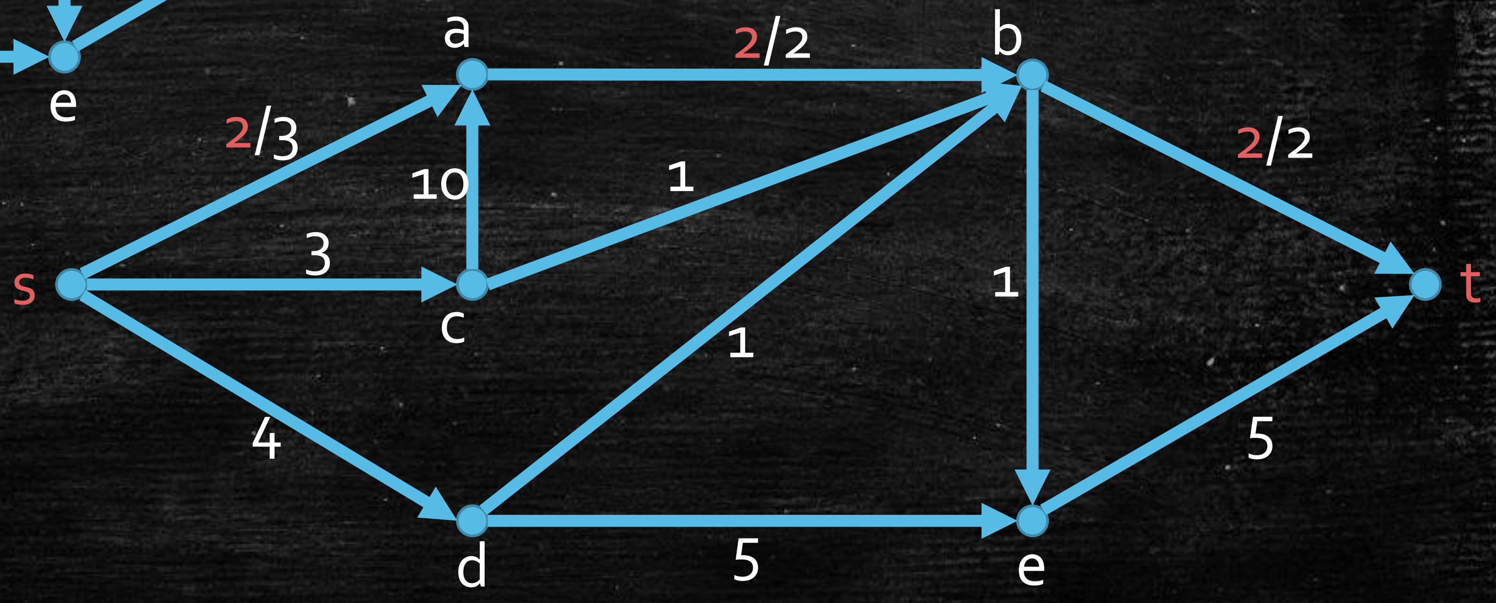
- Capacity Constraint:
  - for each $e \in E$, $f(e) \leq c(e)$.
- Flow Conservation:
  - for each $u \in V \setminus \{s, t\}$, $\sum_{v:(u,v)\in E} f(v, u) = \sum_{w:(u,w)\in E} f(u, w)$.
- Total flow:
  - $v(f) = \sum_{v:(s,v)\in E} f(s, v)$.



Ford-Fulkerson Algorithm

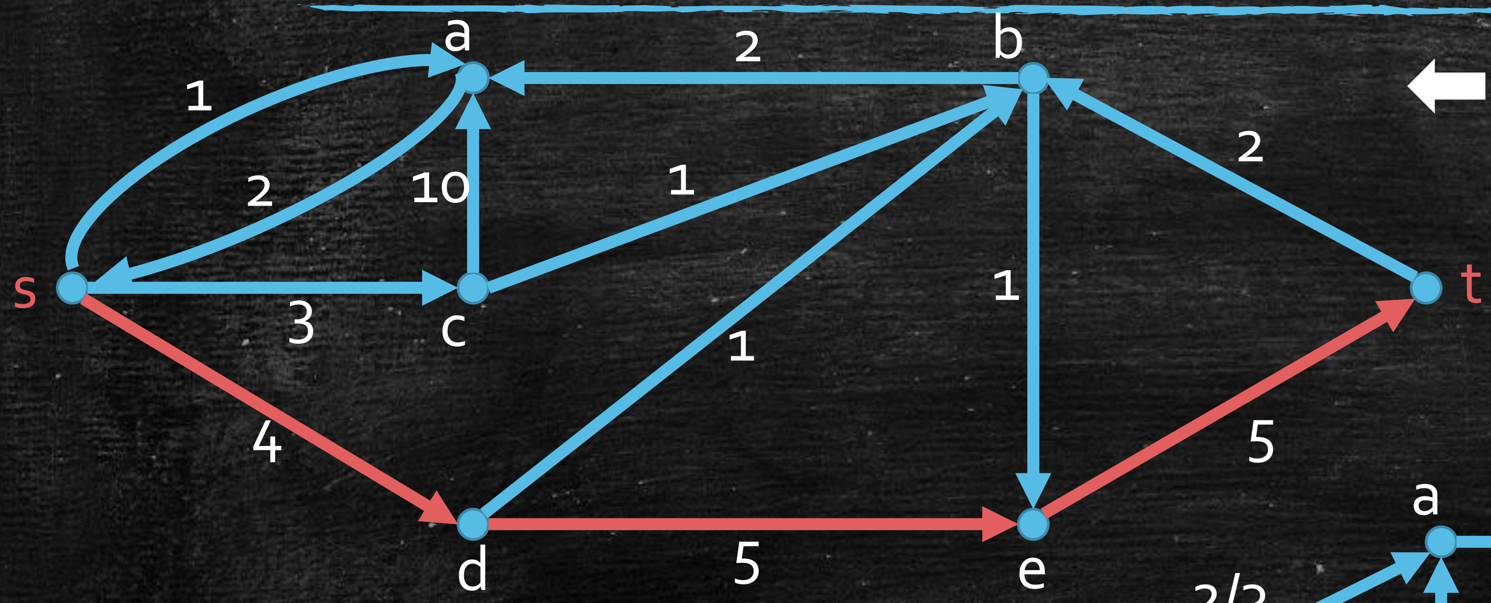


Residual Network



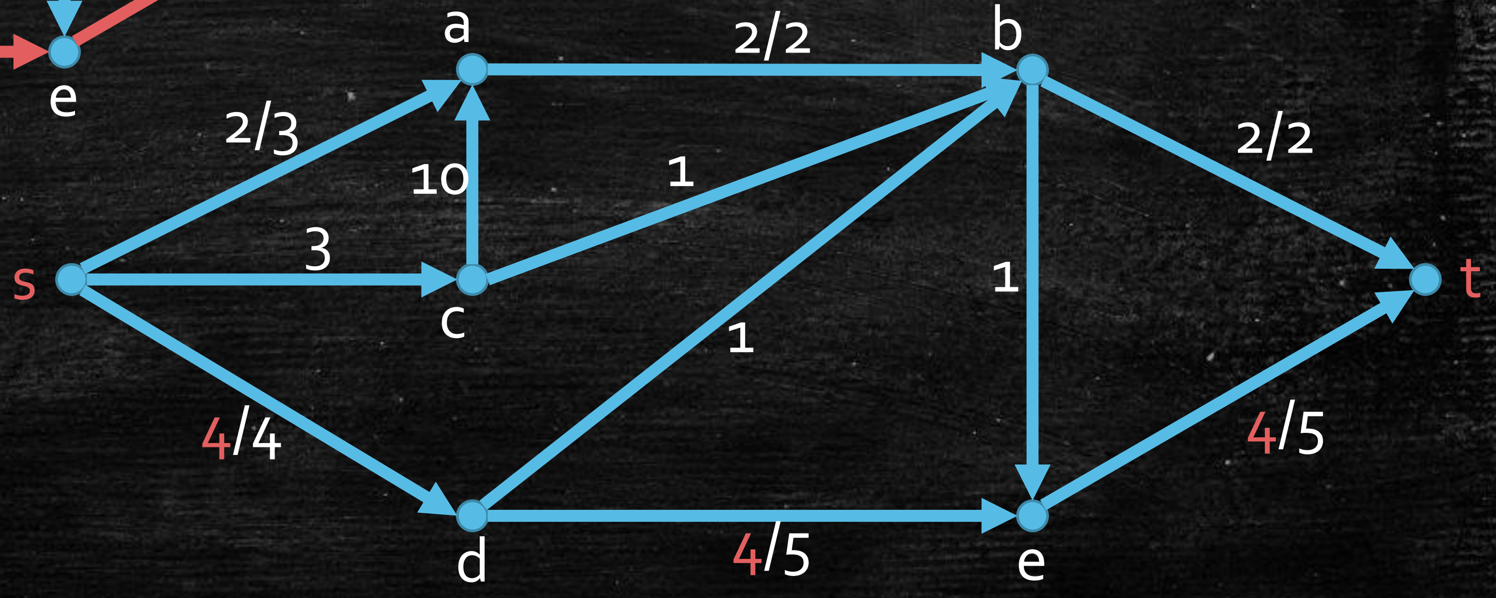
Current Flow

Ford-Fulkerson Algorithm

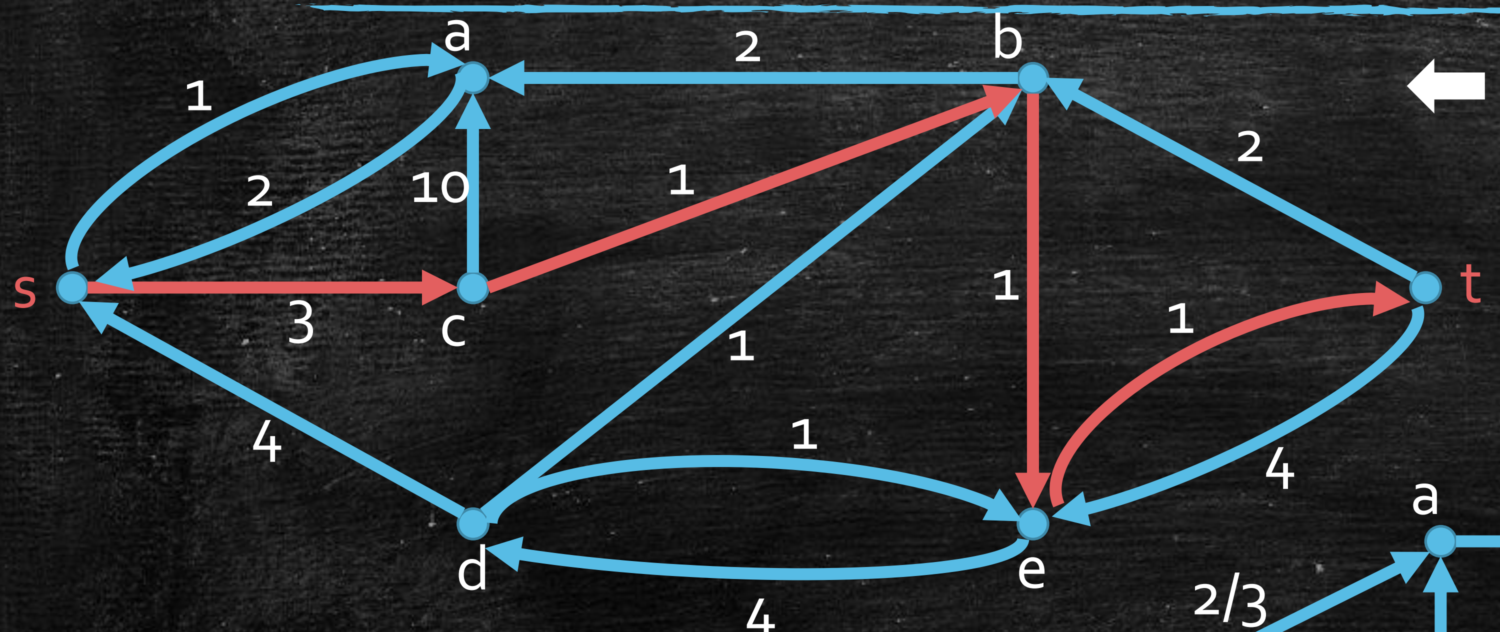


Residual Network

Current Flow

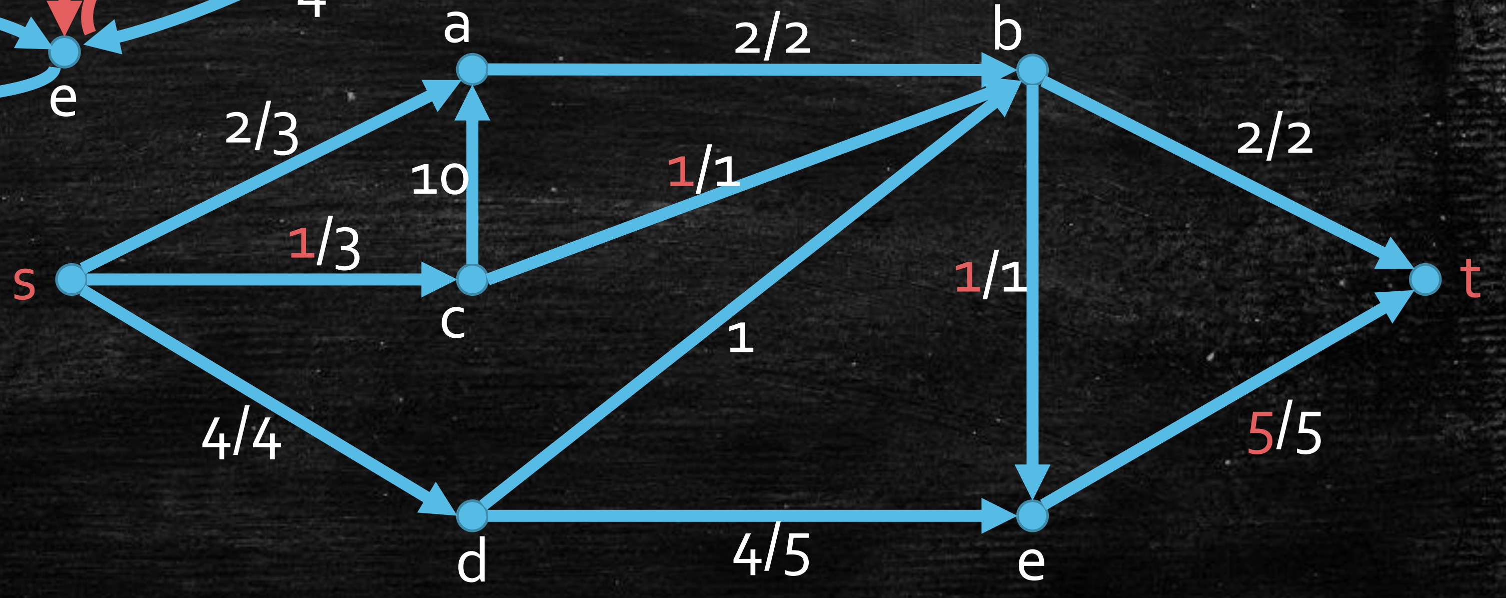


Ford-Fulkerson Algorithm

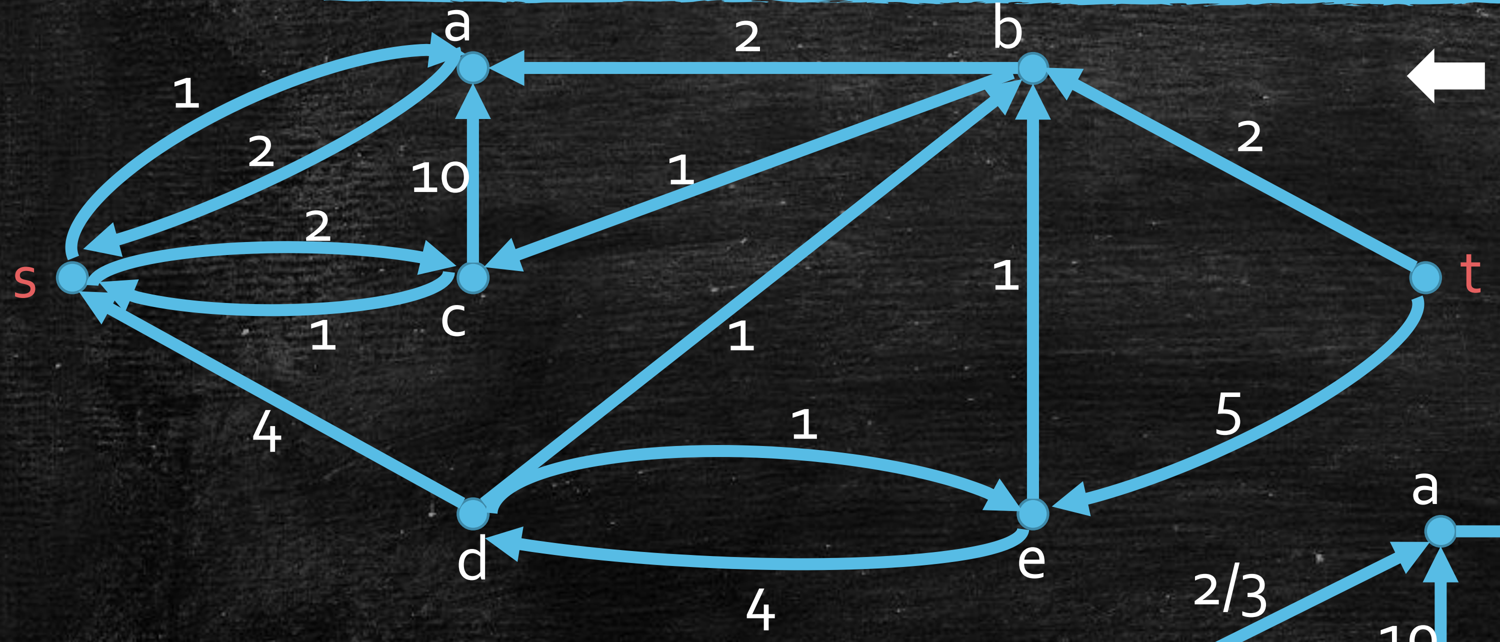


Residual Network

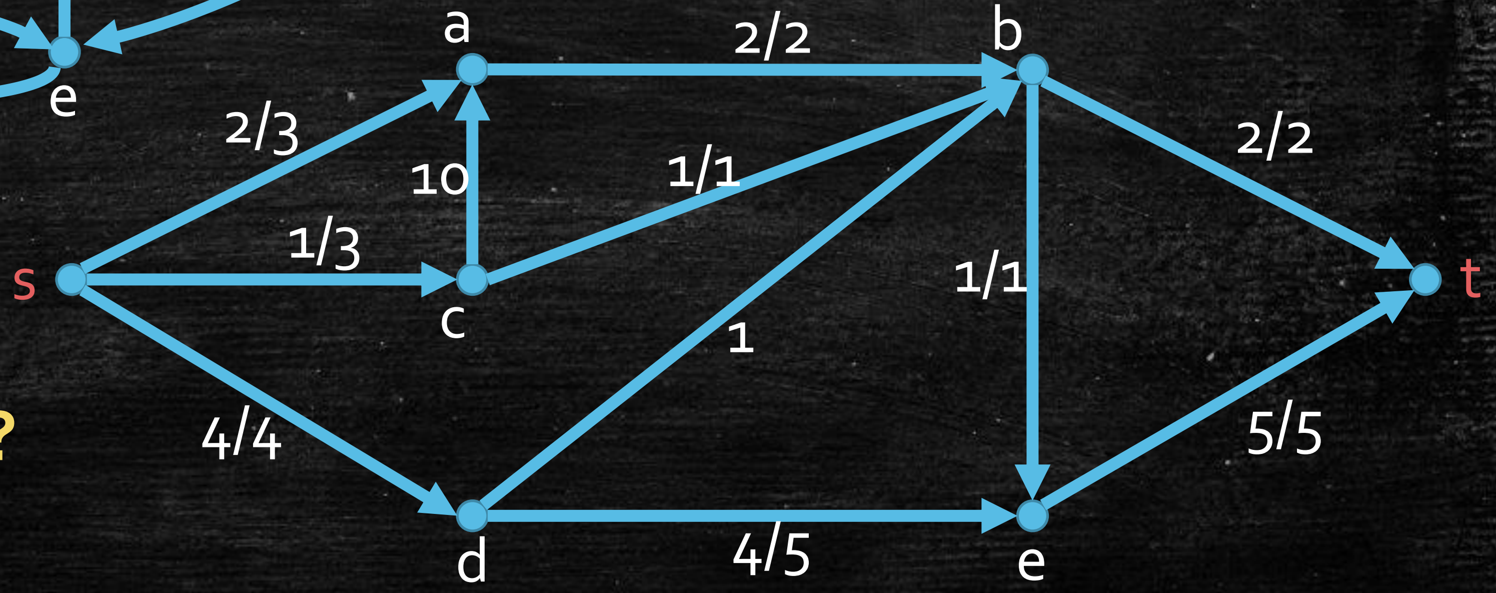
Current Flow



Correctness of Ford-Fulkerson



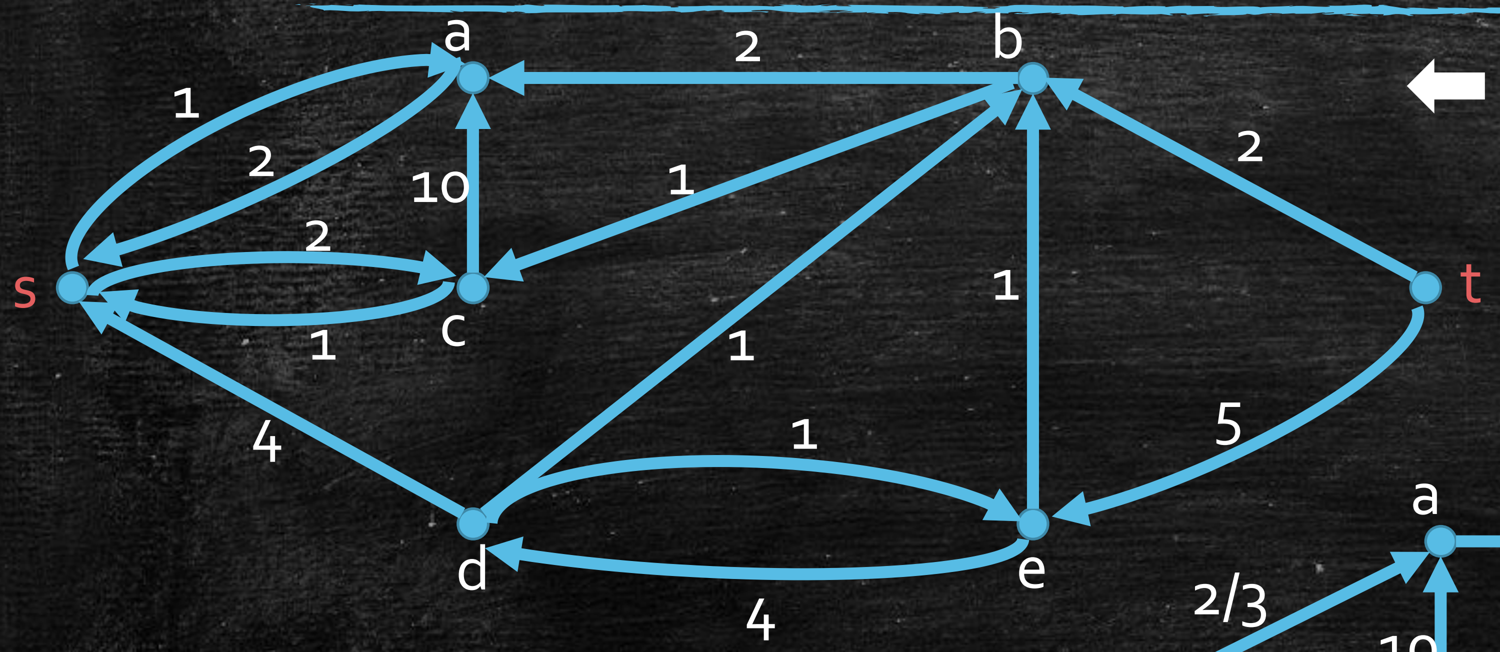
Residual Network



Current Flow

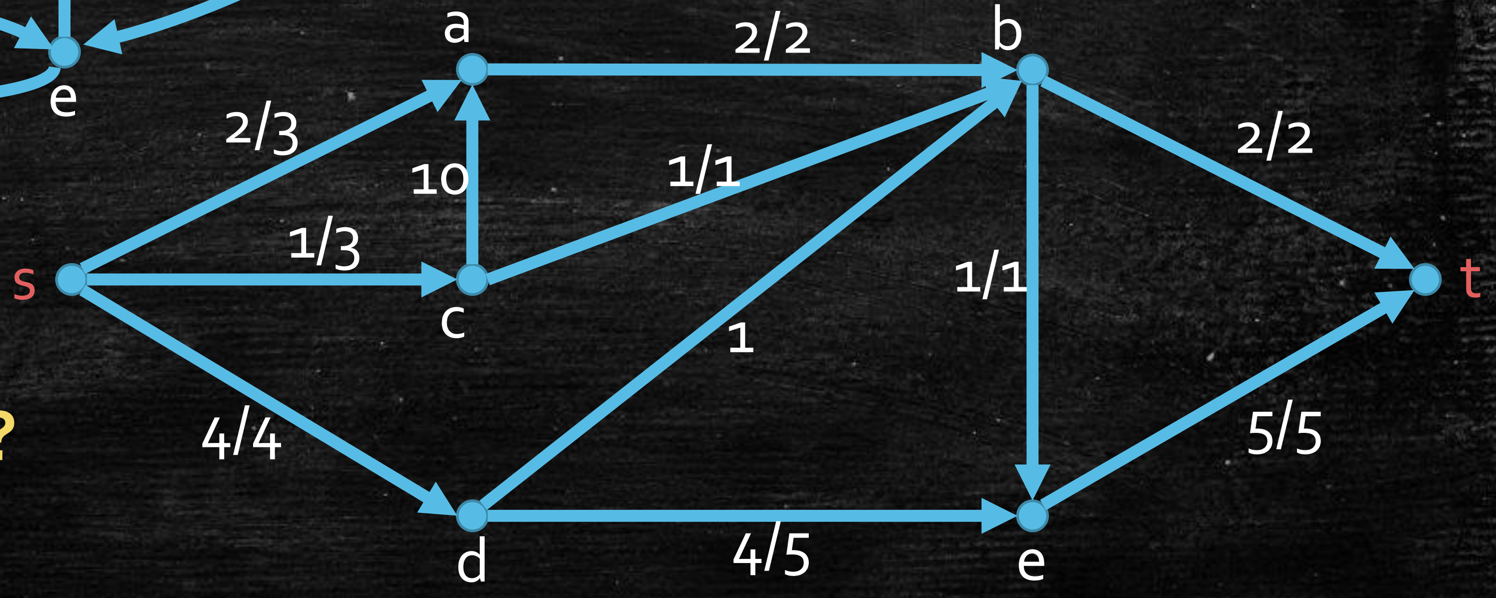
Is $v(f) = 7$ optimal?

What we have?



Residual Network

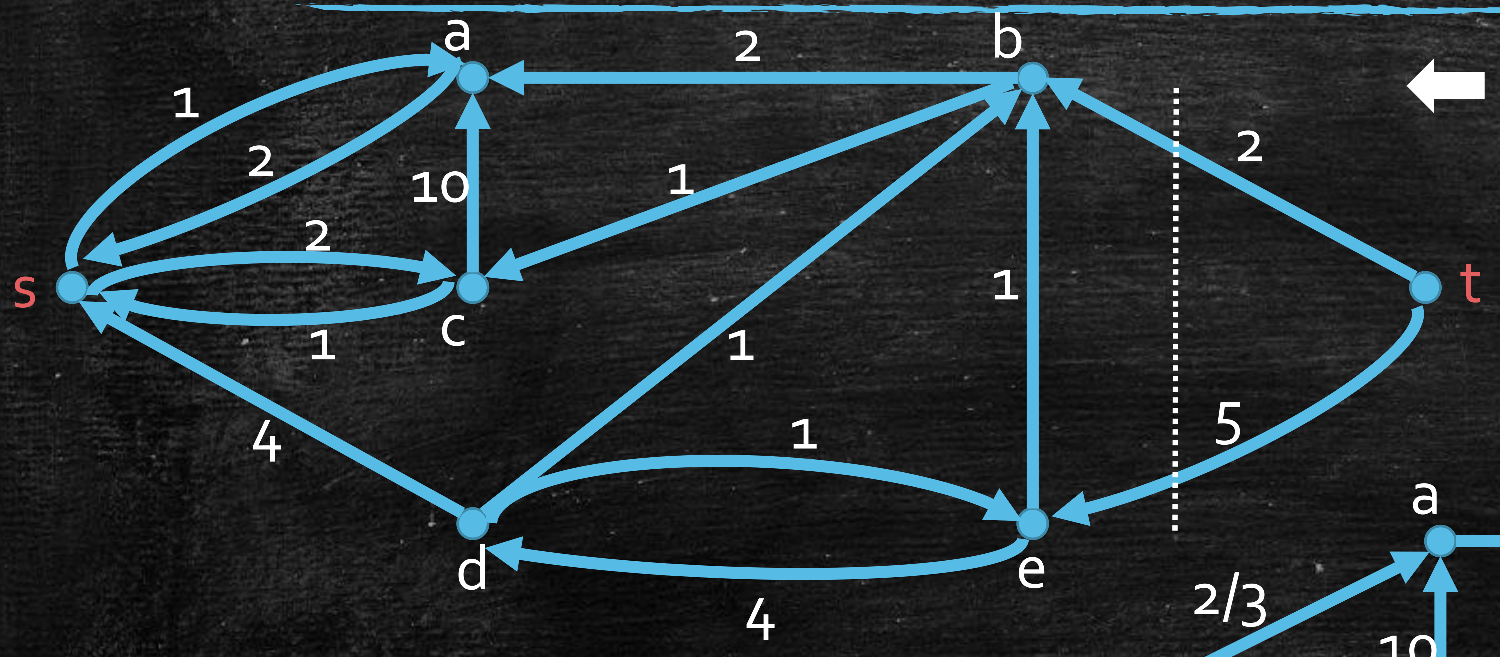
There is no augmenting path!



Current Flow

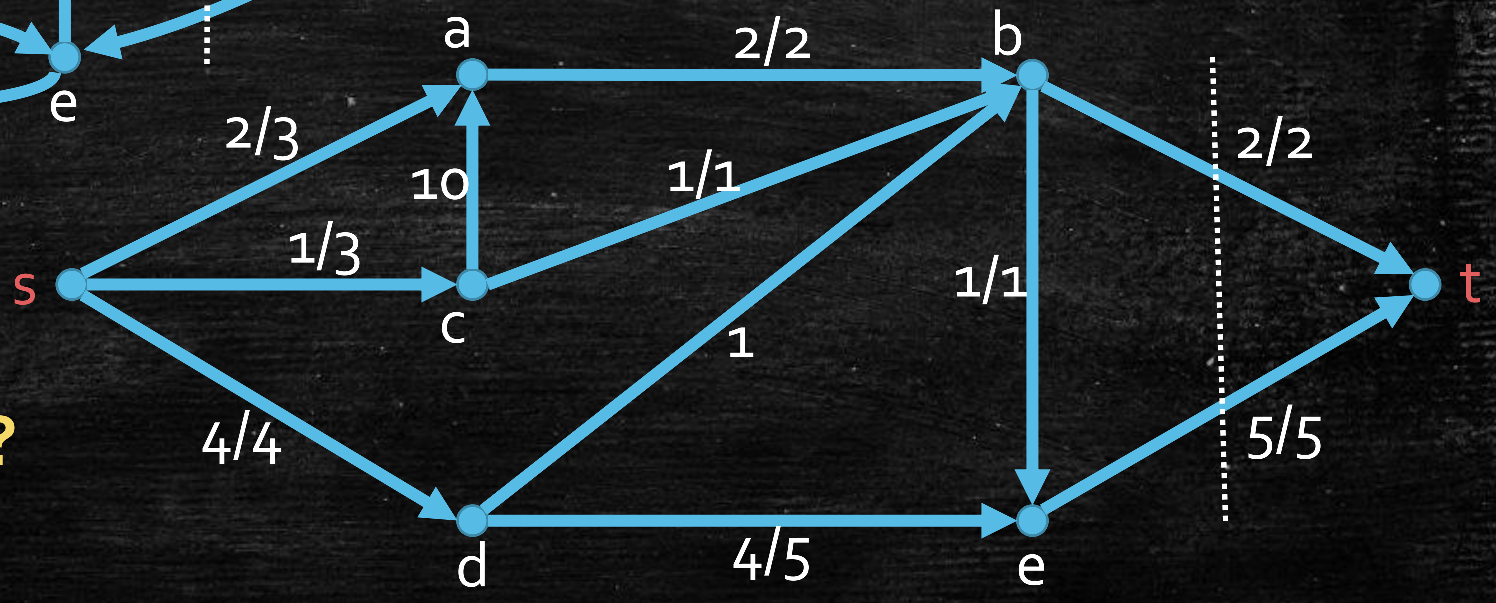
Is $v(f) = 7$ optimal?

What we have?



⬅ Residual Network

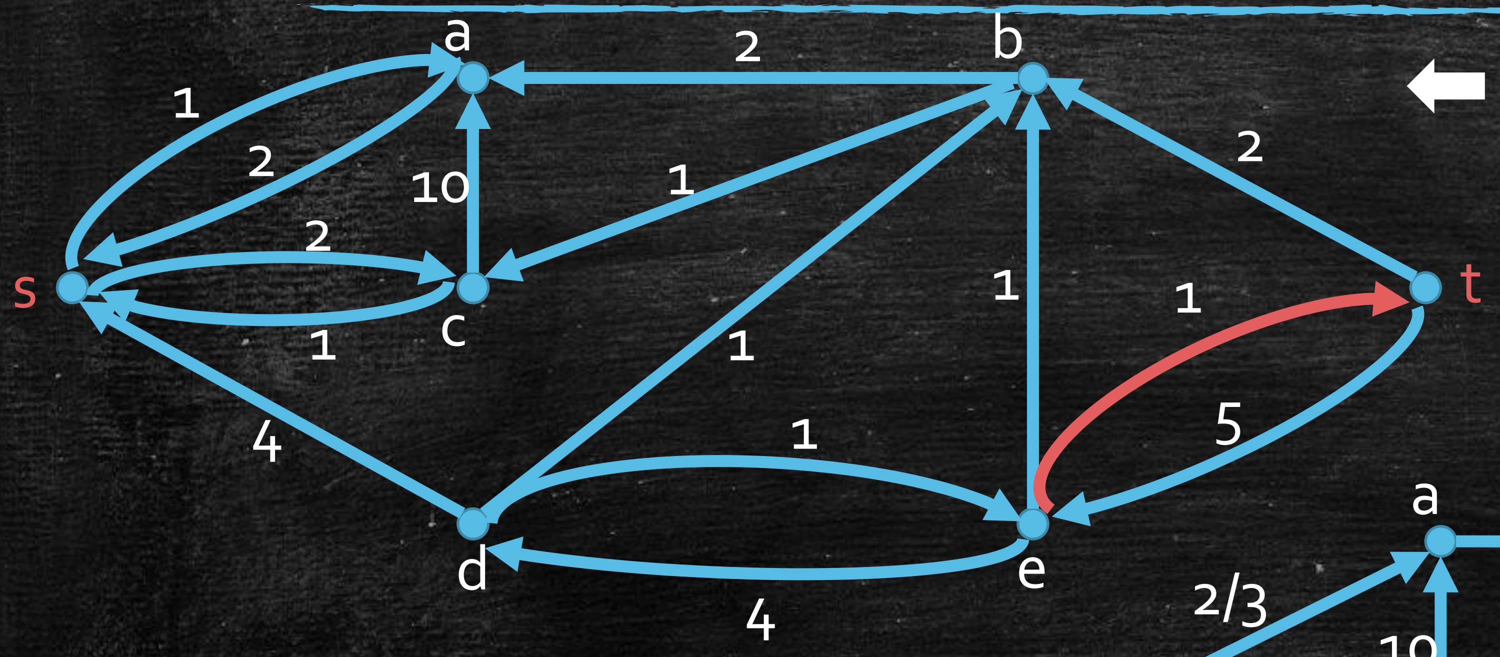
There is no augmenting path!



Current Flow ➡

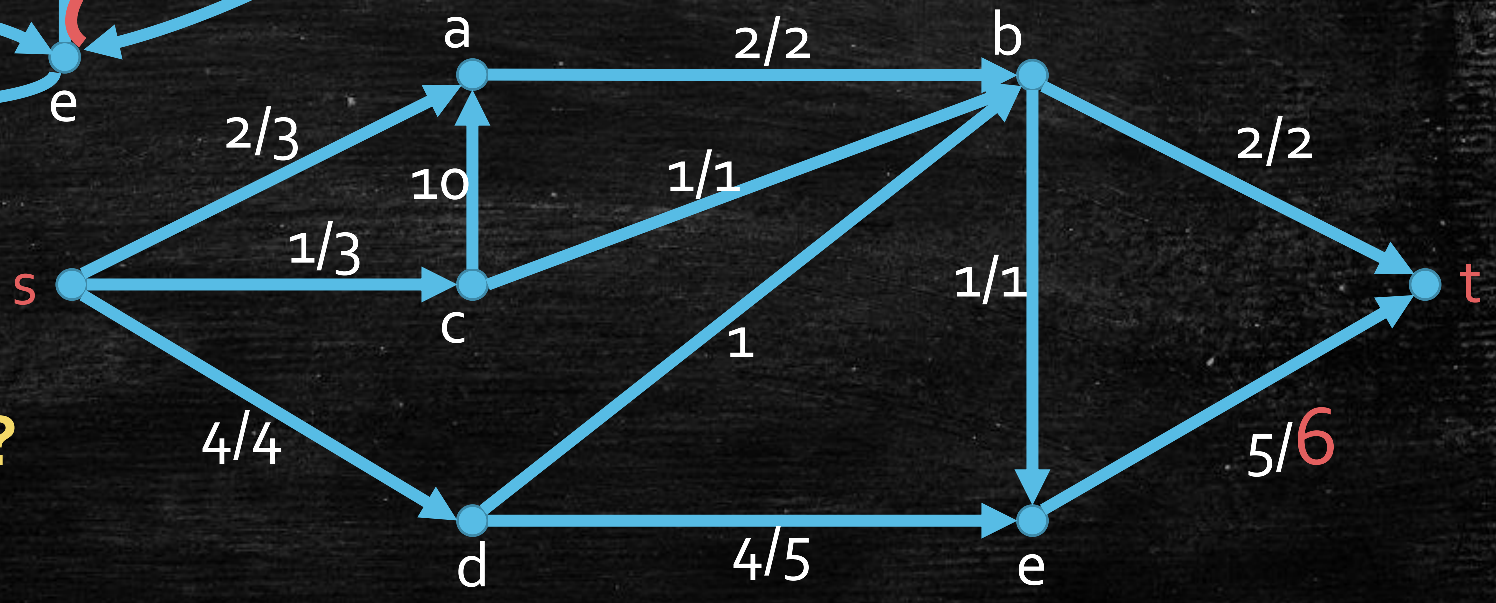
Is $v(f) = 7$ optimal?

What we have?



Residual Network

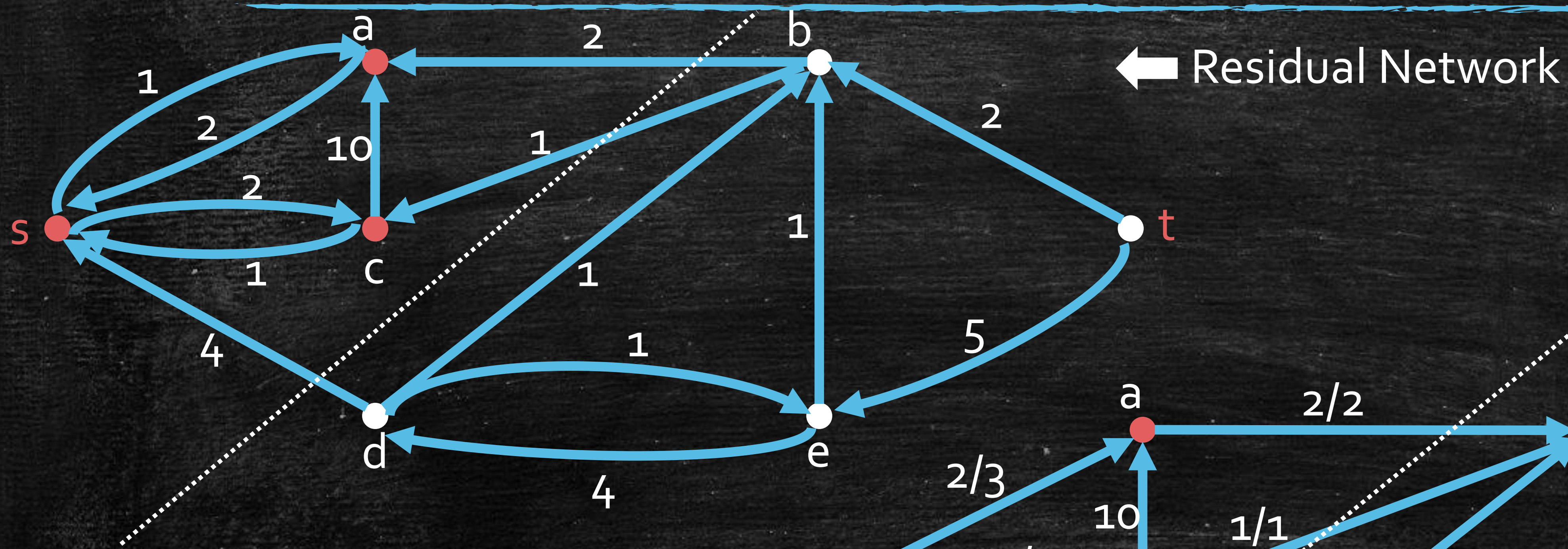
There is no augmenting path!

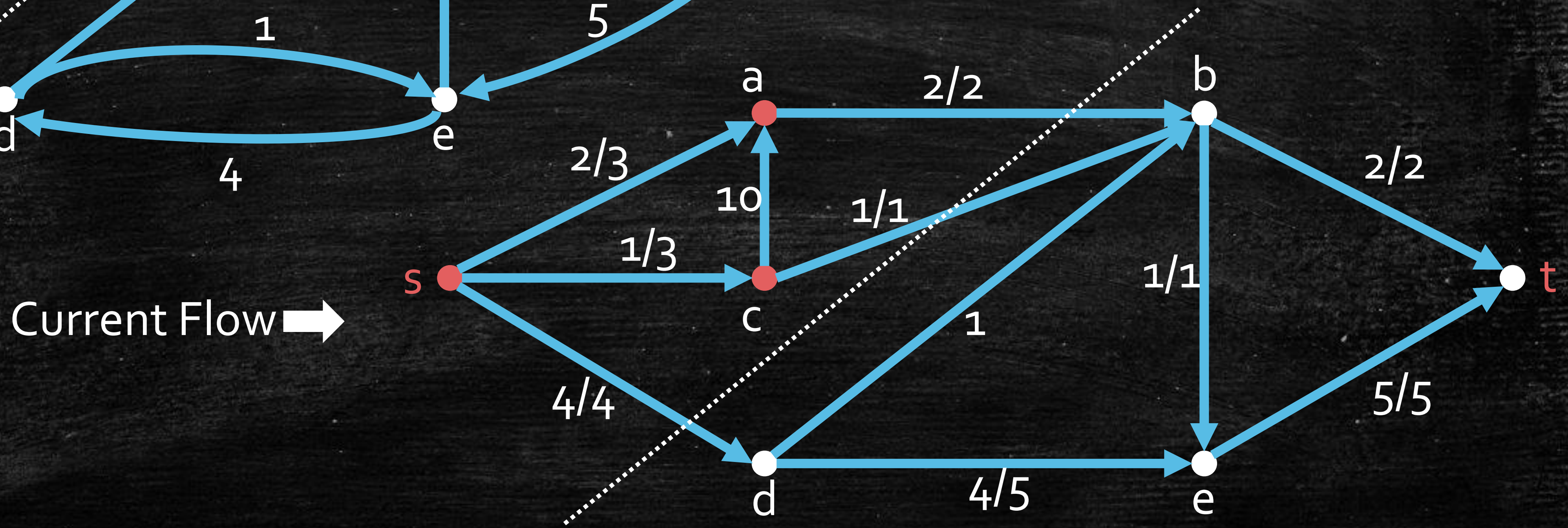


Current Flow

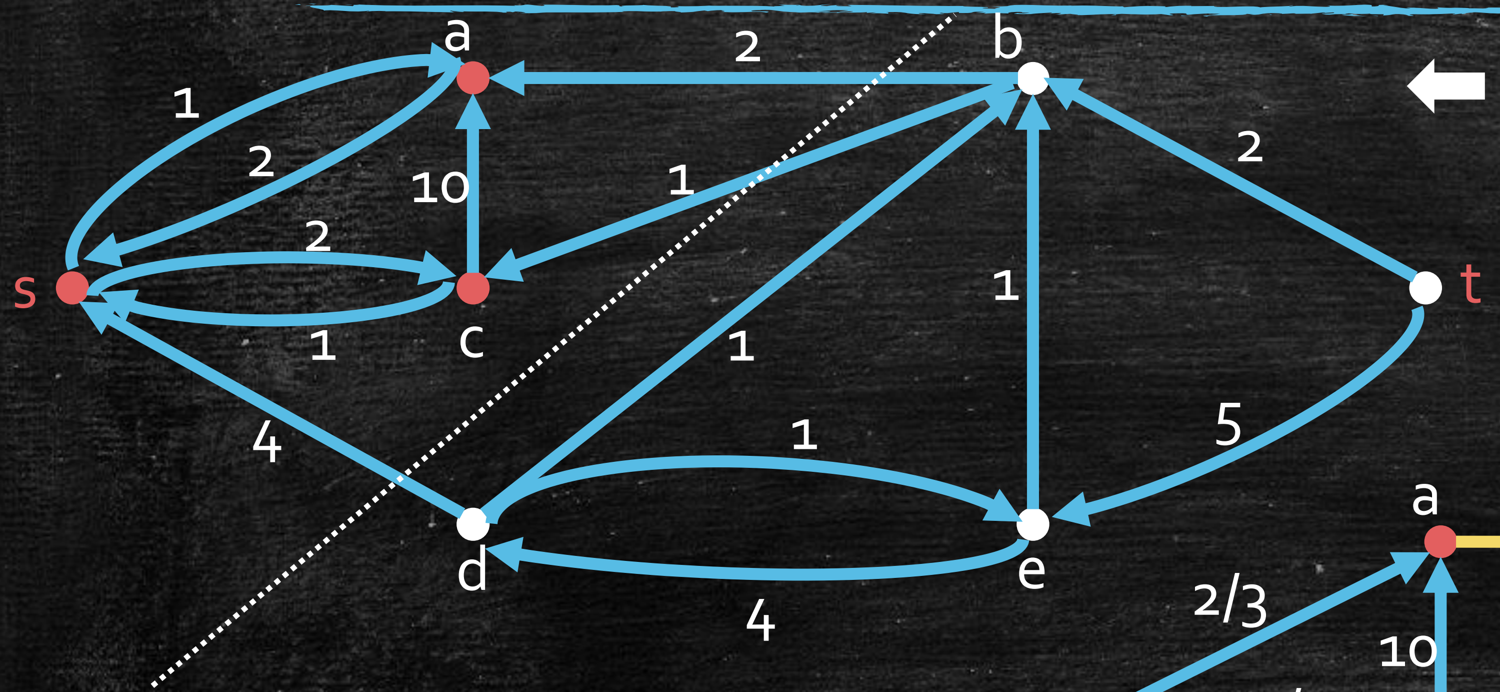
Is $v(f) = 7$ optimal?

Cut: An edge set to partition vertices.



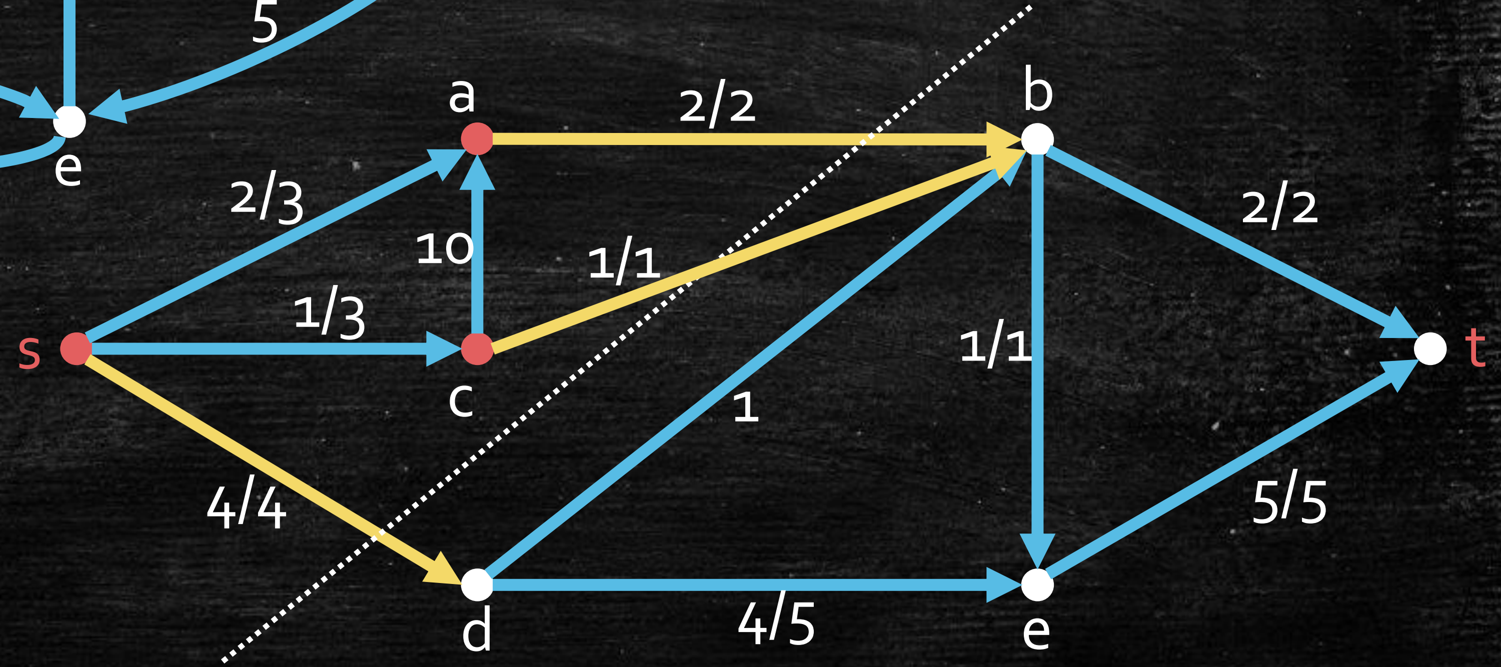


Some cut block us!

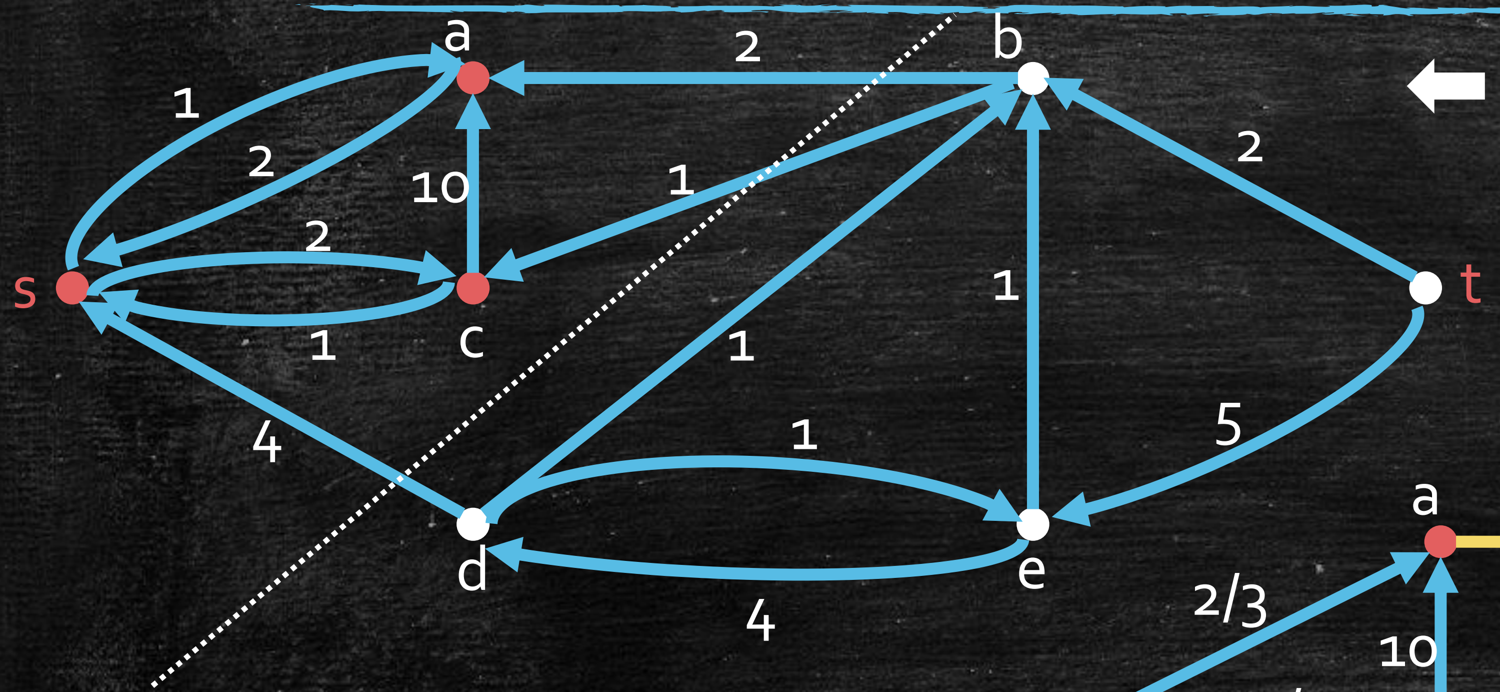


⬅ Residual Network

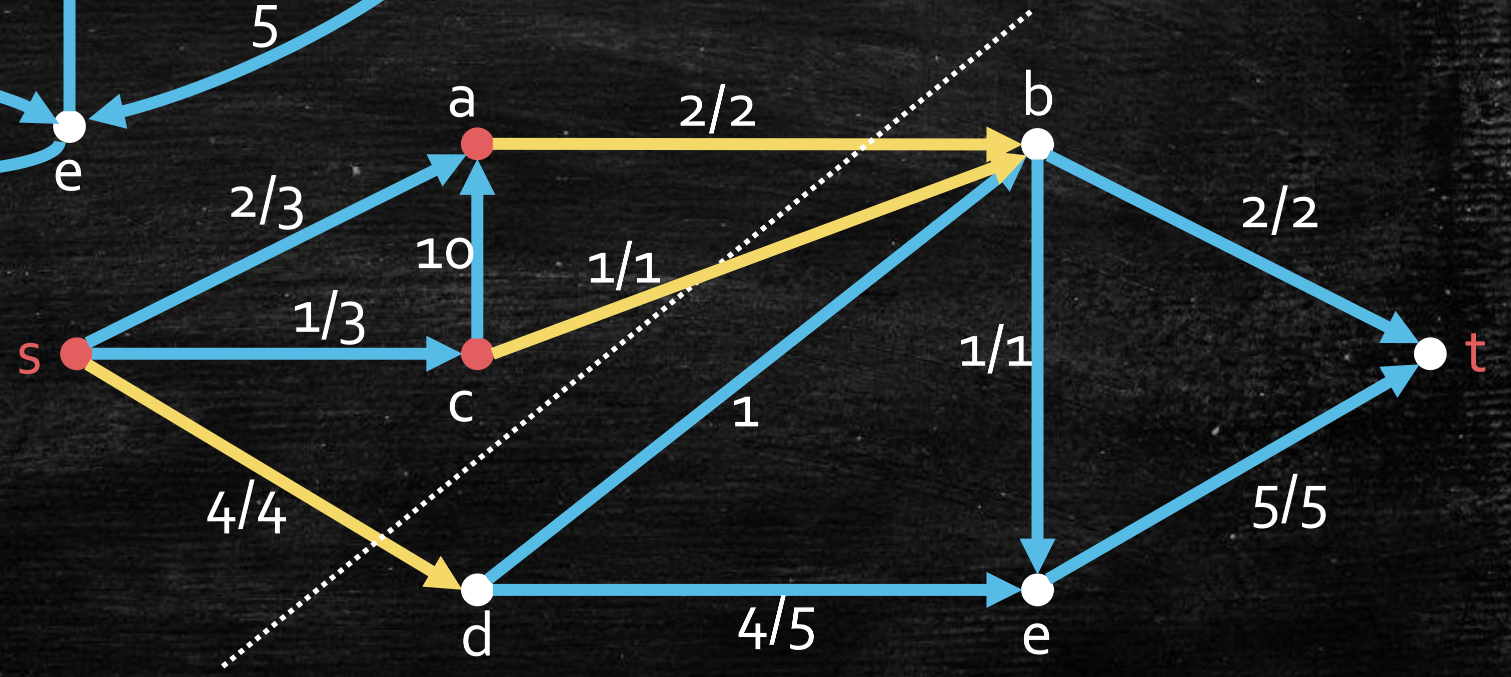
Current Flow ➡



The blocking cut: No one can do better!

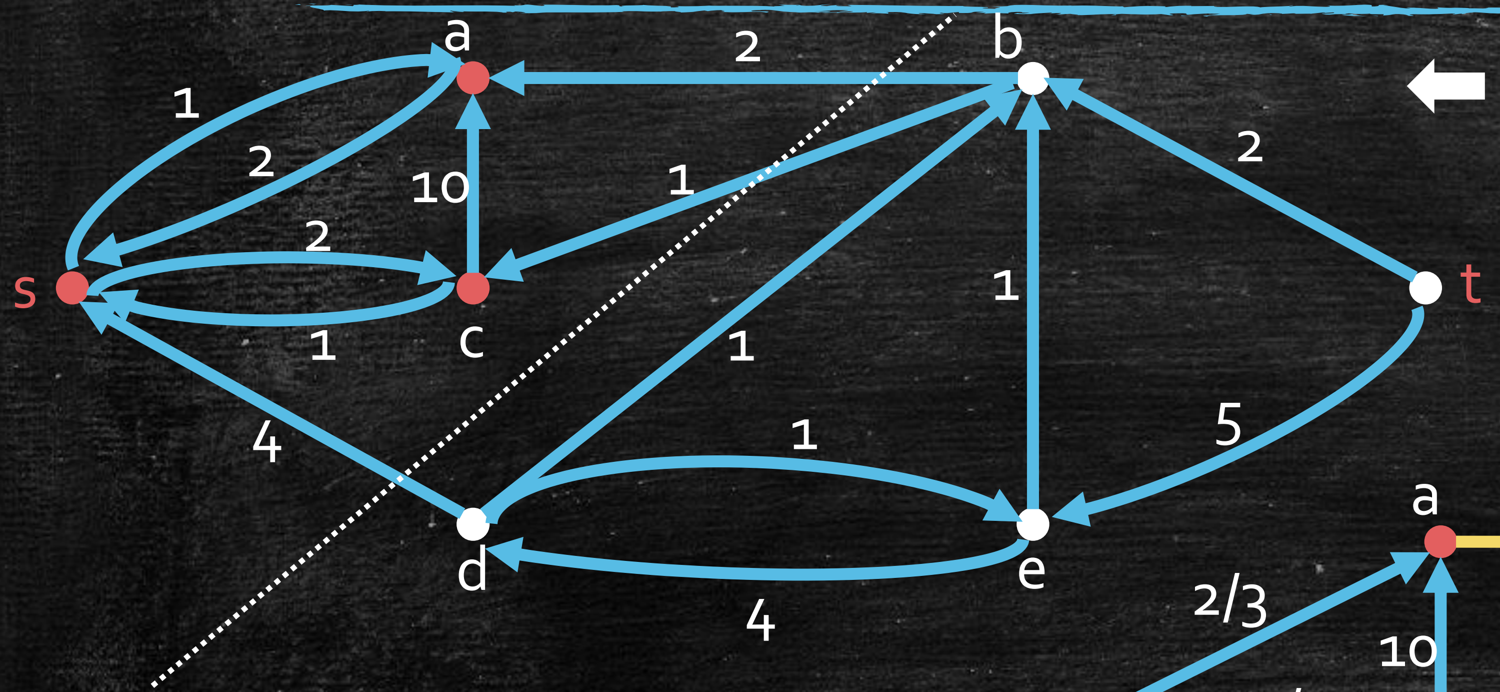


Residual Network

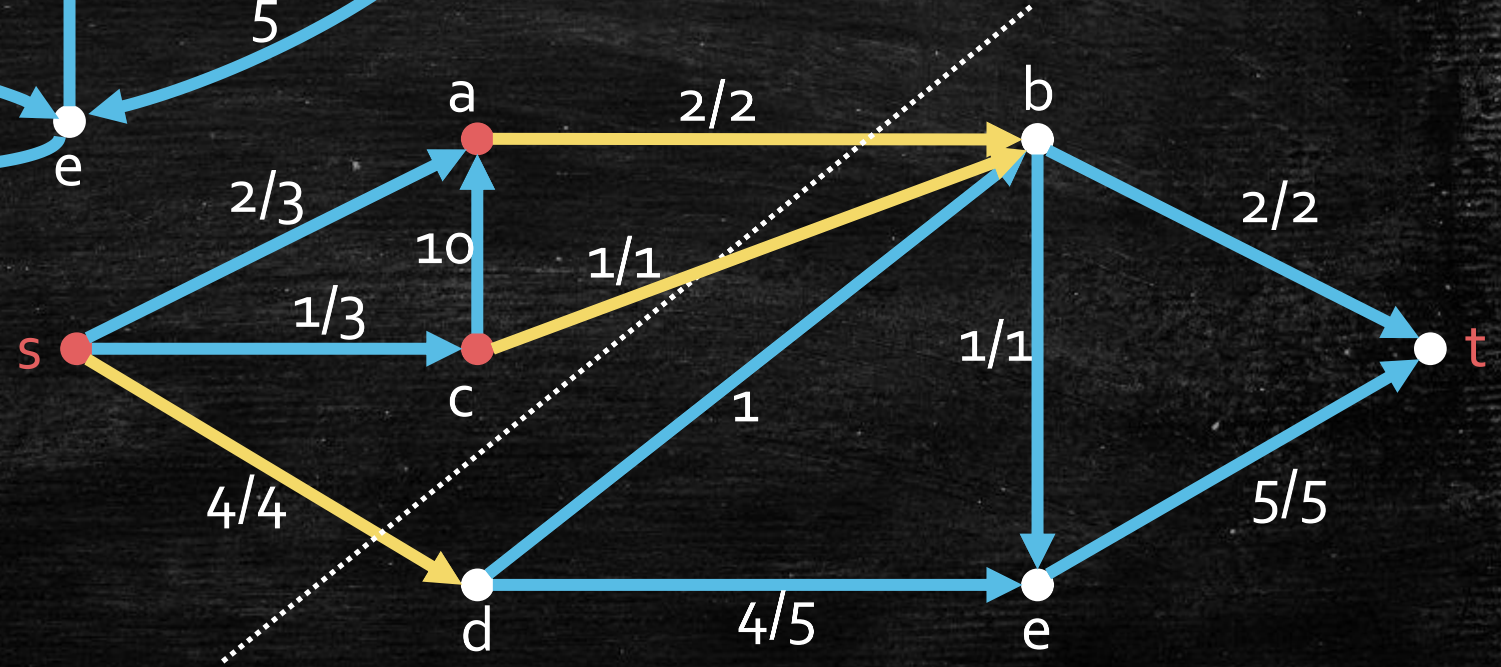


Current Flow

The blocking cut: FF has made it!

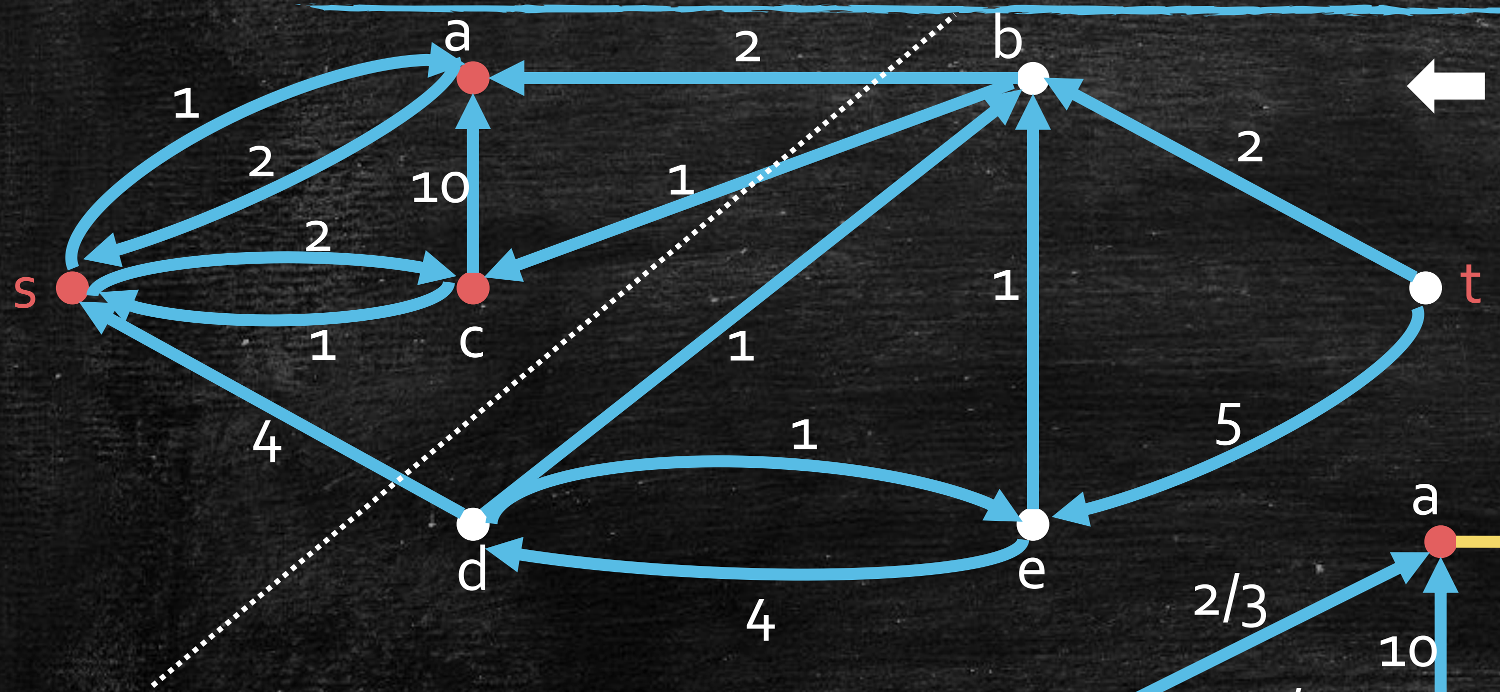


Residual Network

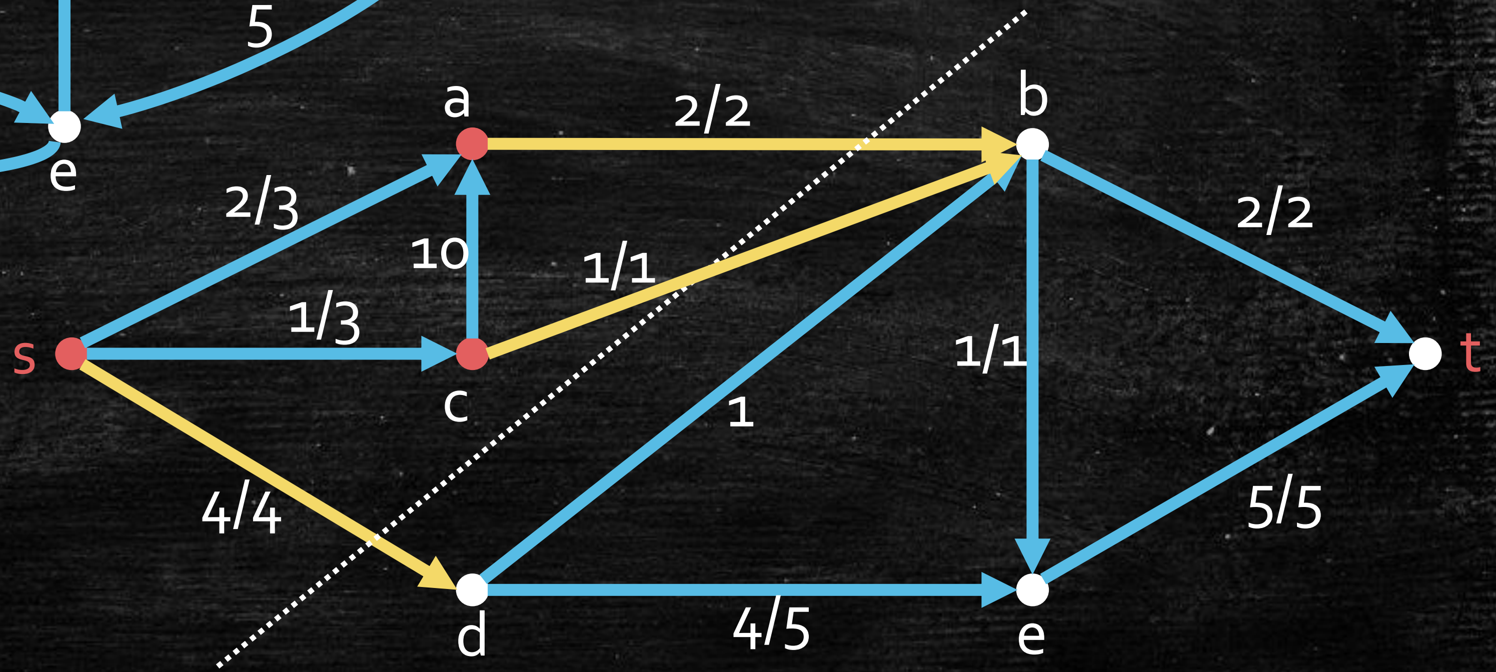


Current Flow

Thus, $v(f) = 7$ is optimal!

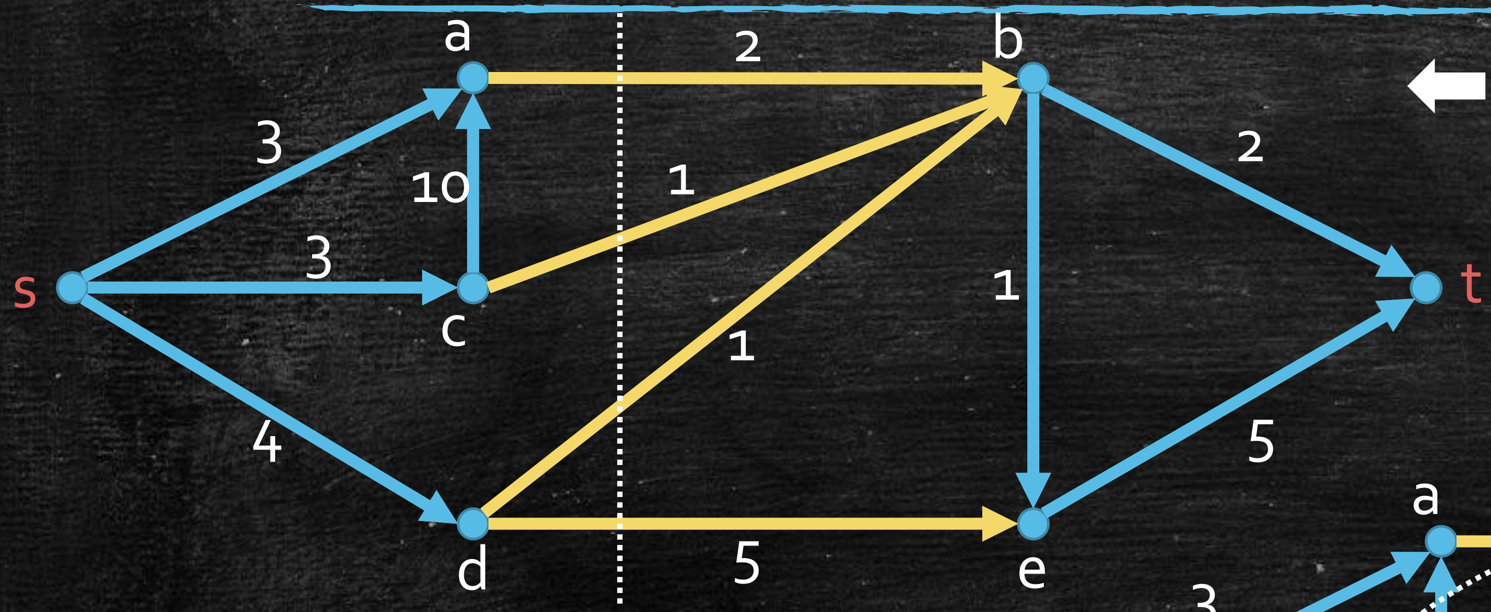


Residual Network

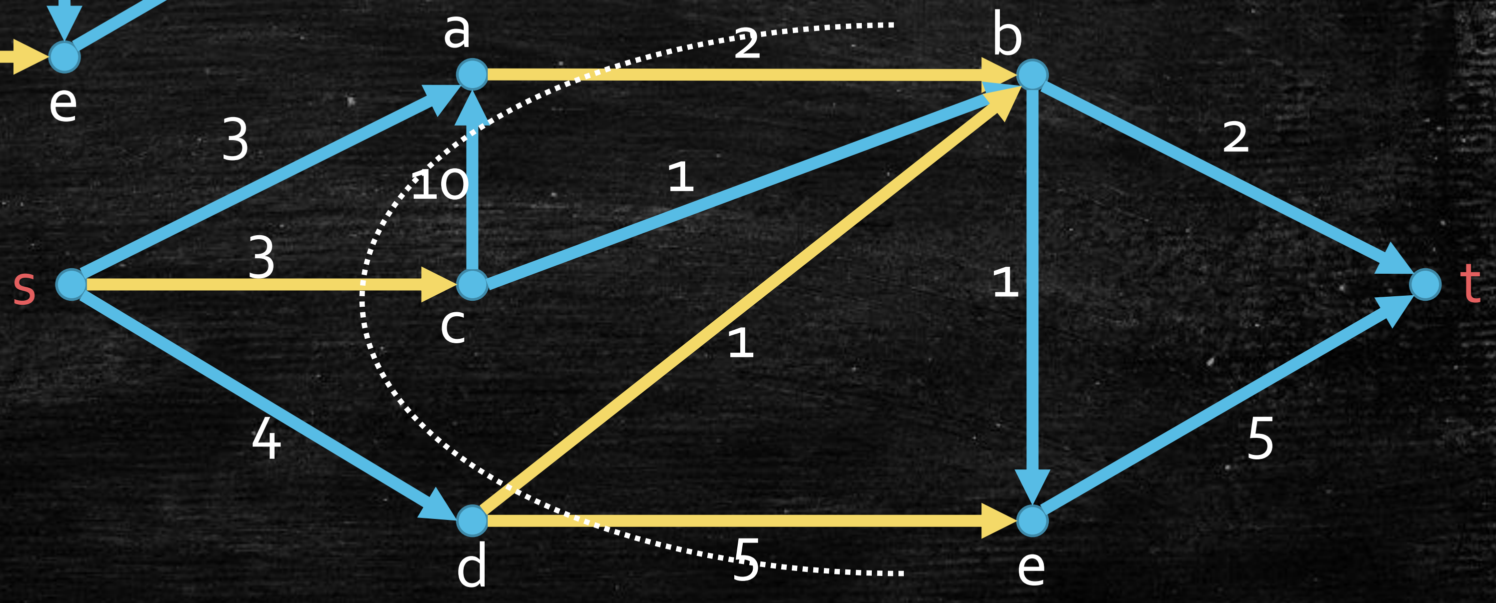


Current Flow

In fact, every "cut" gives an upper-bound to $v(f)$.



$v(f) \leq 2 + 1 + 1 + 5 = 9$



$v(f) \leq 3 + 2 + 5 + 1 = 11$

The Minimum Cut Problem

- We want to find a tightest upper-bound to $v(f)$ by a carefully chosen cut.
- Given weighted graph $G = (V, E, c)$ and $s, t \in V$, an s-t cut is a partition of V to L, R such that $s \in L$ and $t \in R$.
- The value of the cut is defined by

$$c(L, R) = \sum_{(u,v)\in E, u\in L, v\in R} c(u, v)$$

- **Min-Cut Problem**: Given $G = (V, E, c)$ and $s, t \in V$, find the s-t cut with the minimum value.

Max-Flow-Min-Cut Theorem

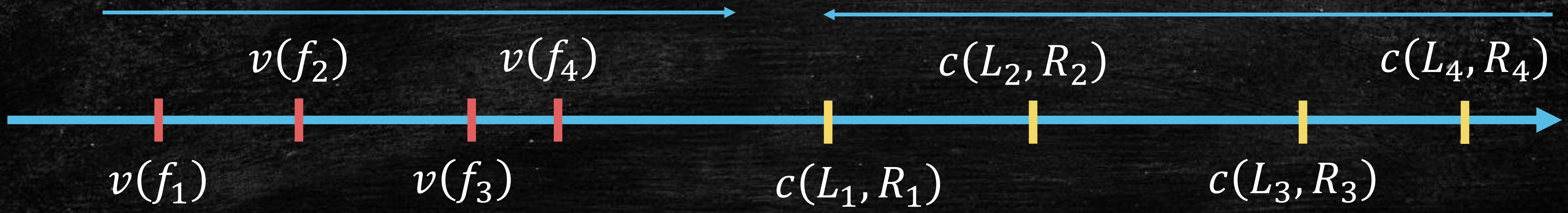
- The value of every s-t cut is an upper-bound to $v(f)$.

Max-Flow-Min-Cut Theorem. The value of the maximum flow is exactly the value of the minimum cut:

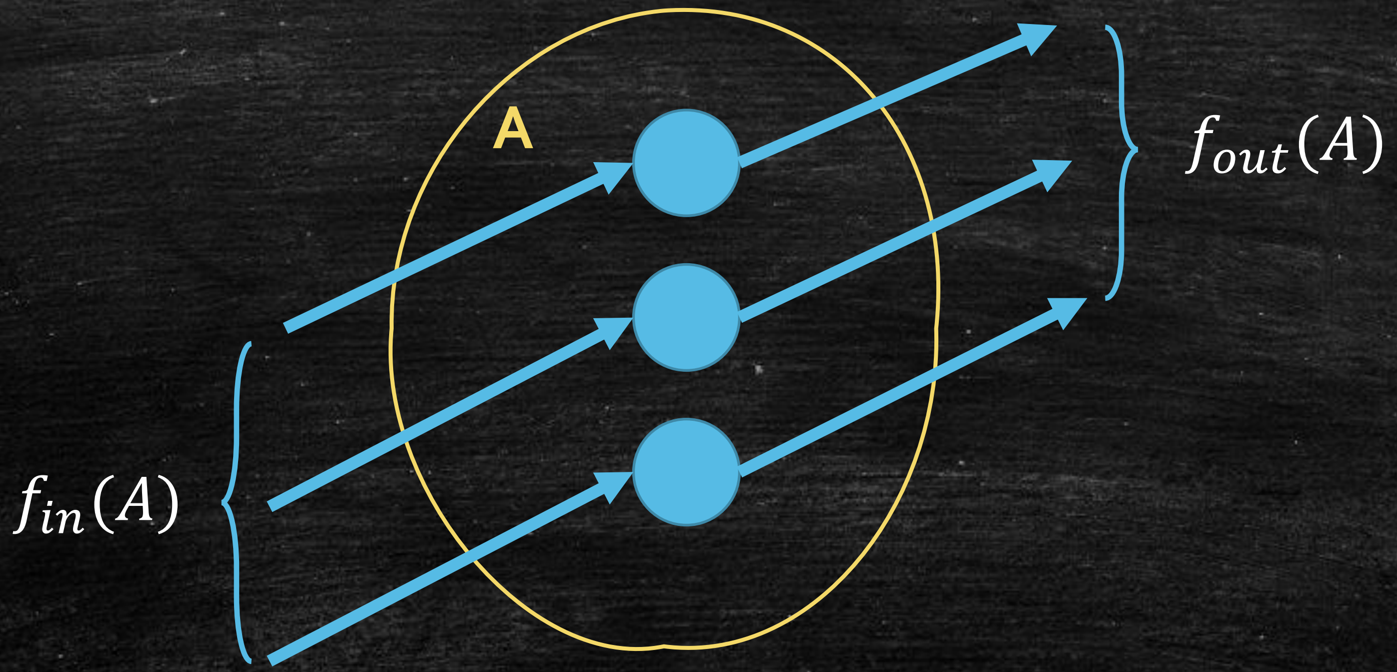
$$\max_{f} v(f) = \min_{L,R} c(L, R)$$

Proving Max-Flow-Min-Cut Theorem

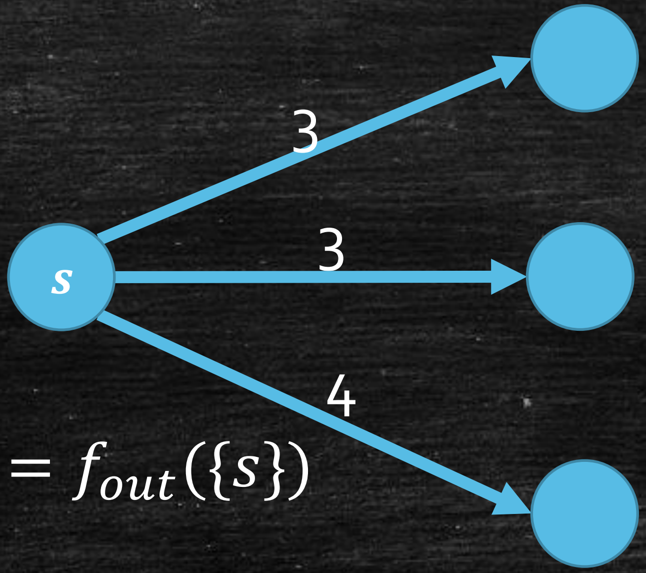
- Lemma 1. For any flow f and any cut $\{L, R\}$, we have $v(f) \leq c(L, R)$.
  - Formalize the idea that the value of any cut is an upper-bound to the value of any flow.
- Lemma 2. There exists a cut $\{L, R\}$ such that the flow f output by Ford-Fulkerson Algorithm satisfies $v(f) = c(L, R)$.
  - Concludes Max-Flow-Min-Cut Theorem.
  - Proves the correctness of Ford-Fulkerson Algorithm.



Notations: in flow and out flow

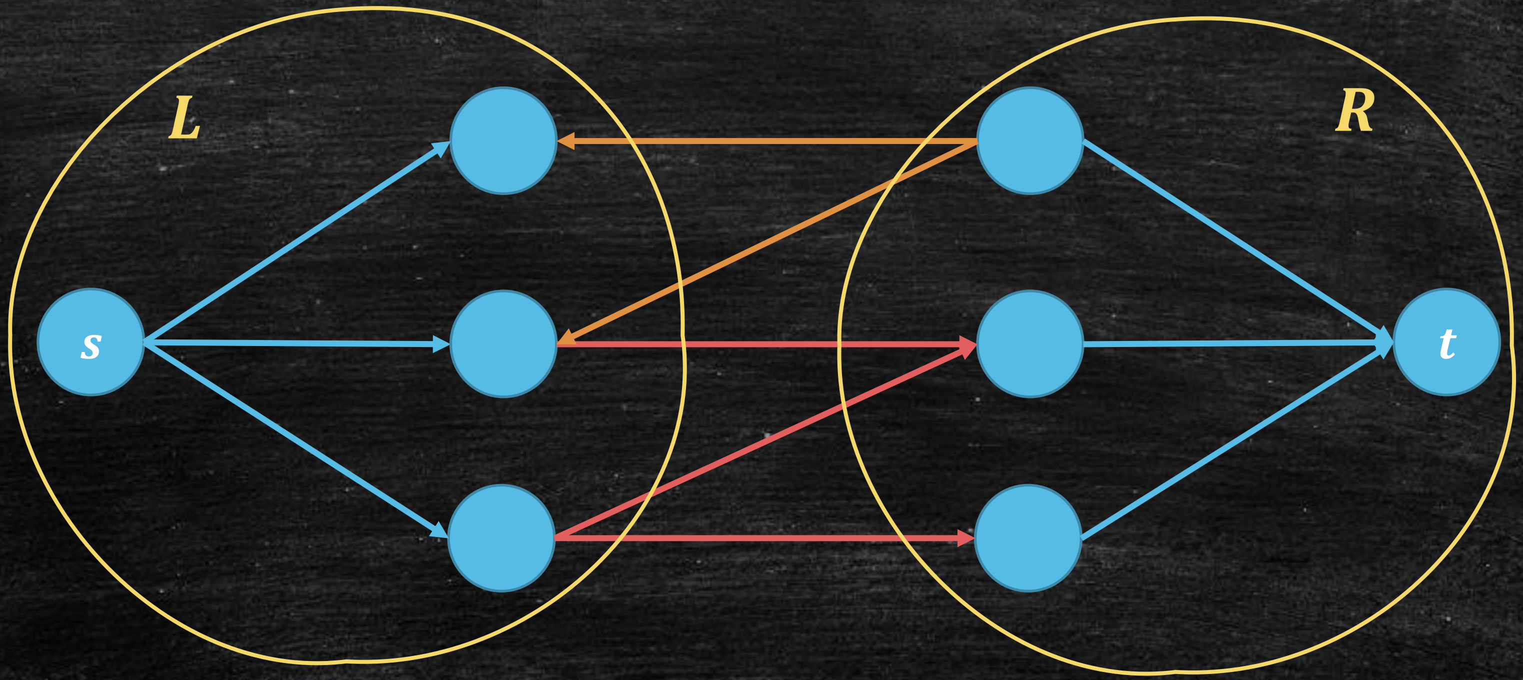


Notations: value of a flow



$v(f) = f_{out}(\{s\})$

Notations: flow across a s-t cut

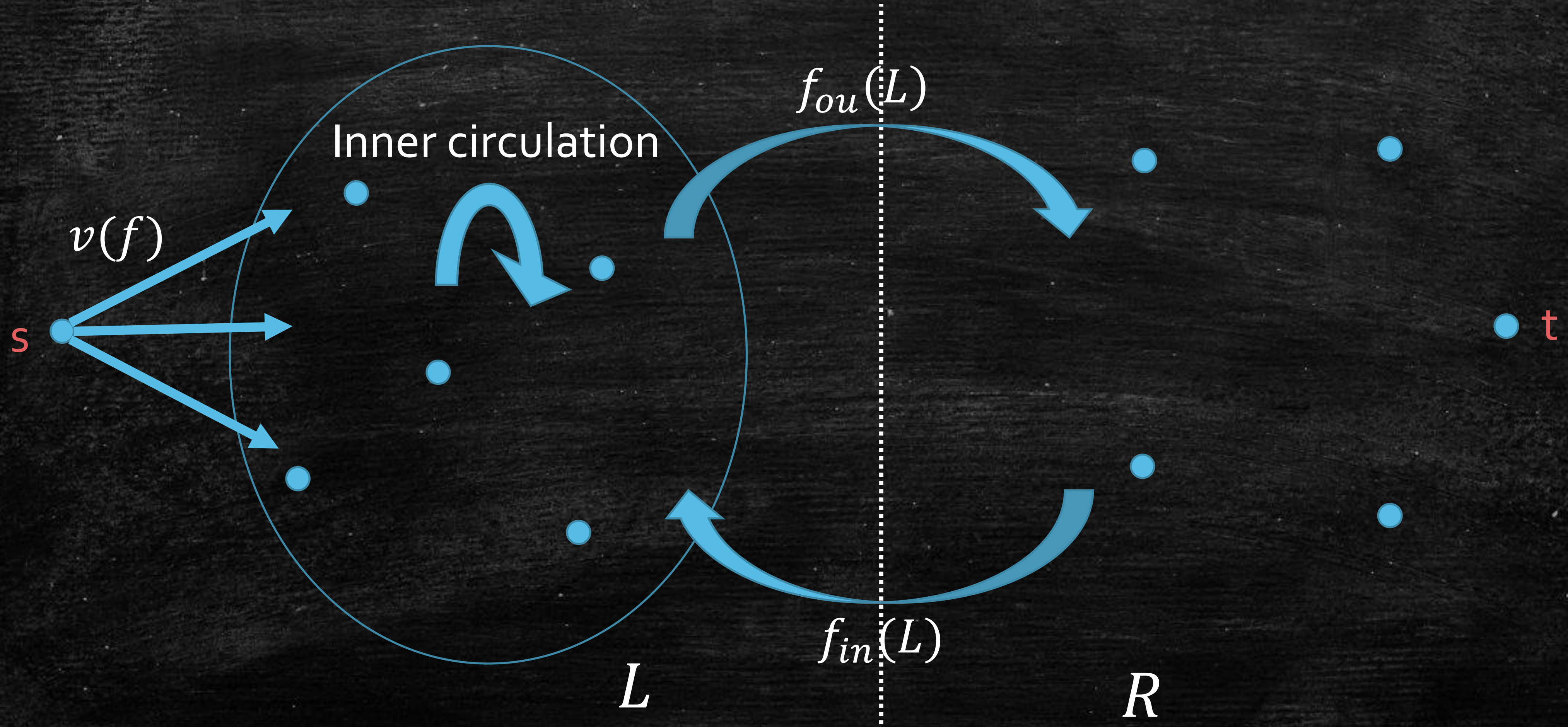


$$f_{in}(L) = \sum_{(u,v)\in E, u\in L, v\in R} f(v,u)$$

$$f_{ou}(L) = \sum_{(u,v)\in E, u\in L, v\in R} f(u,v)$$

Proving generalized flow conservation

Claim: $v(f) = f_{ou}(L) - f_{in}(L)$



Proving generalized flow conservation

Claim: $v(f) = f_{ou}(L) - f_{in}(L)$

- Flow conservation:
  - $f_{ou}(u) = f_{in}(u)$ for $u \in V \setminus \{s, t\}$
  - $f_{ou}(s) = v(f)$, $f^{\text{in}}(s) = 0$
- Summing up vertices in L:

$$\sum_{u \in L} \left(f_{ou}(u) - f_{in}(u) \right) = f_{ou}(s) + \sum_{u \in L \setminus \{s\}} 0 = v(f).$$

Proving generalized flow conservation

Claim: $v(f) = f_{ou}(L) - f_{in}(L)$

- Look at the summation again. Can you see the following?

$$\sum_{u \in L} \left(f_{ou}(u) - f_{in}(u) \right) = f_{ou}(L) - f_{in}(L).$$

- For each $f(u, v)$ with $u, v \in L$, it contributes $+f(u, v)$ to the summation by $f_{ou}(u)$ and contributes $-f(u, v)$ by $f_{in}(v)$. Cancelled!
- For each $f(u, v)$ with $u \in L, v \in R$, it contributes $+f(u, v)$ to the summation.
- For each $f(u, v)$ with $u \in R, v \in L$, it contributes $-f(u, v)$ to the summation.

Proving generalized flow conservation

Claim: $v(f) = f_{ou}(L) - f_{in}(L)$

- We have

$$\sum_{u \in L} \left(f_{ou}(u) - f_{in}(u) \right) = f_{ou}(s) + \sum_{u \in L \setminus \{s\}} 0 = v(f).$$

- and

$$\sum_{u \in L} \left(f_{ou}(u) - f_{in}(u) \right) = f_{ou}(L) - f_{in}(L).$$

- Putting together:

$$v(f) = f_{ou}(L) - f_{in}(L).$$

Proof of Lemma 1

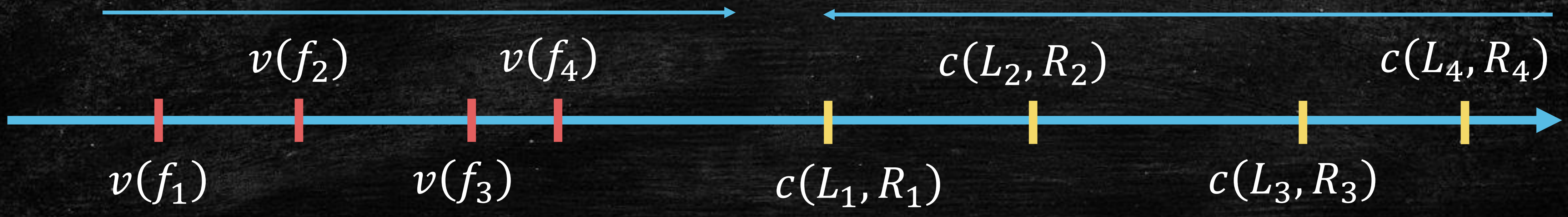
Lemma 1. For any flow f and any cut $\{L, R\}$, we have $v(f) \leq c(L, R)$.

- Fix L, R.
- Claim: $v(f) = f_{ou}(L) - f_{in}(L)$
- If the claim holds, Lemma 1 is proved:

$$v(f) \leq f_{ou}(L) = \sum_{(u,v)\in E, u\in L, v\in R} f(u, v) \leq \sum_{(u,v)\in E, u\in L, v\in R} c(u, v) = c(L, R)$$

Proving Max-Flow-Min-Cut Theorem

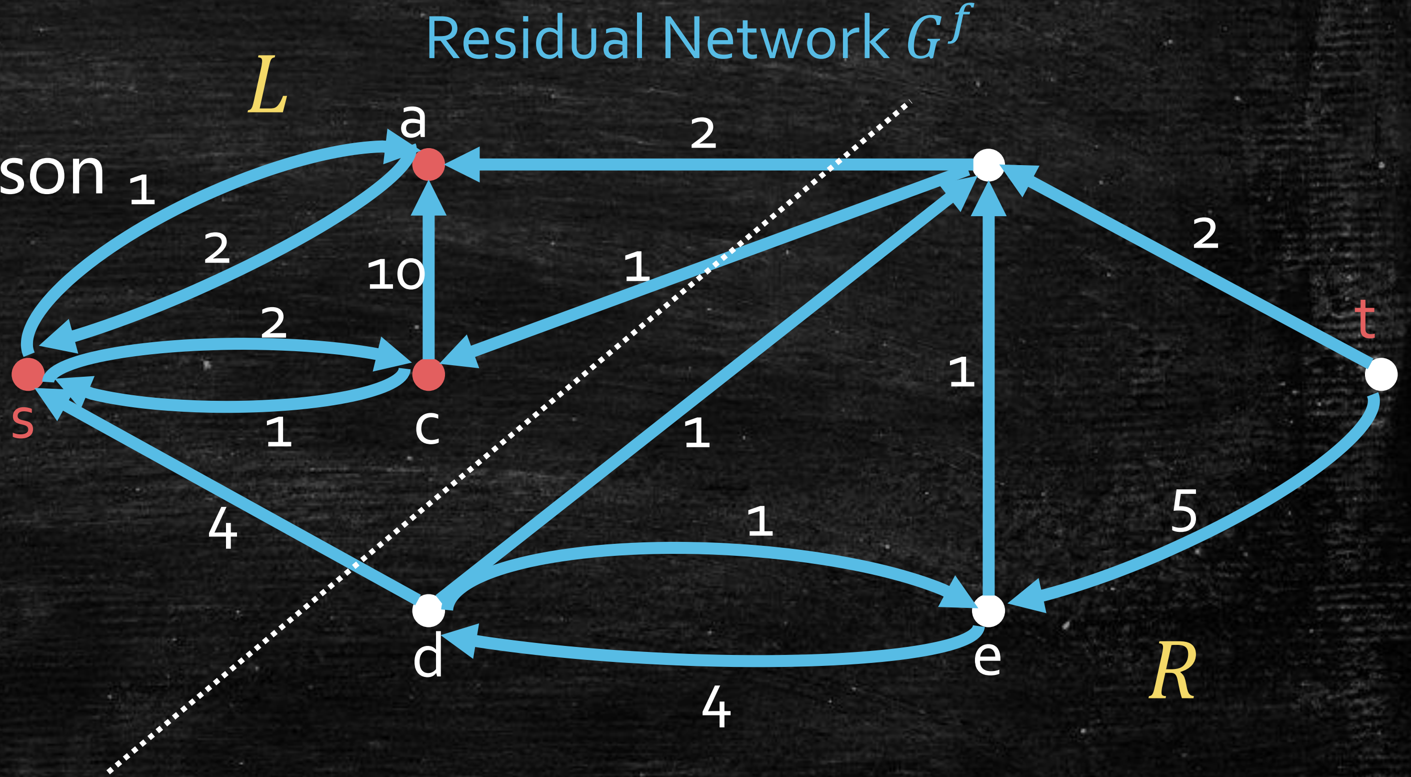
- ✓ Lemma 1. For any flow f and any cut $\{L, R\}$, we have $v(f) \leq c(L, R)$.
  - Formalize the idea that the value of any cut is an upper-bound to the value of any flow.
- Lemma 2. There exists a cut $\{L, R\}$ such that the flow f output by Ford-Fulkerson Algorithm satisfies $v(f) = c(L, R)$.
  - Concludes Max-Flow-Min-Cut Theorem.
  - Proves the correctness of Ford-Fulkerson Algorithm.



Proof of Lemma 2

Lemma 2. There exists a cut $\{L, R\}$ such that the flow f output by Ford-Fulkerson Algorithm satisfies $v(f) = c(L, R)$.

- L: reachable from s in G^f.
- f: output of Ford-Fulkerson
- Claim A: $f_{ou}(L) = c(L, R)$
- Claim B: $f_{in}(L) = 0$
- $v(f) = f_{ou}(L) - f_{in}(L)$



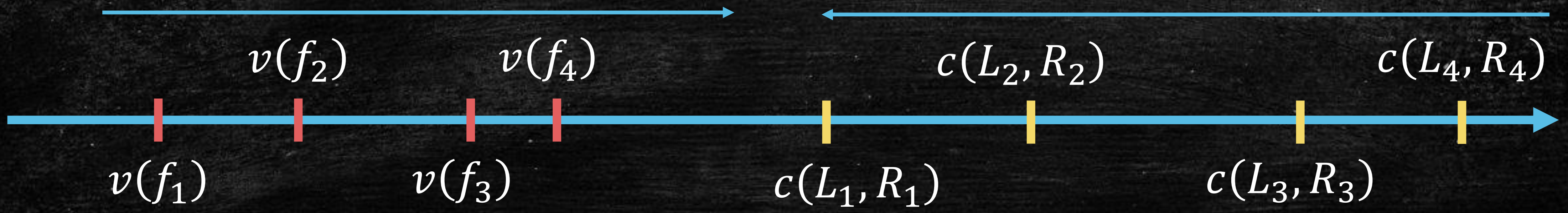
Proof of Lemma 2

Lemma 2. There exists a cut $\{L, R\}$ such that the flow f output by Ford-Fulkerson Algorithm satisfies $v(f) = c(L, R)$.

- Claim A: $f_{ou}(L) = c(L, R)$
  - Otherwise, exist (u, v) with $u \in L, v \in R$ such that $f(u, v) < c(u, v)$.
  - Thus, (u, v) is in G^f and v is reachable from s.
  - Contradict to $v \in R$ by our definition of L.
- Claim B: $f_{in}(L) = 0$
  - Otherwise, exist (v, u) with $u \in L, v \in R$ such that $f(v, u) > 0$.
  - Thus, (u, v) is in G^f and v is reachable from s
  - Contradict to $v \in R$ by our definition of L.

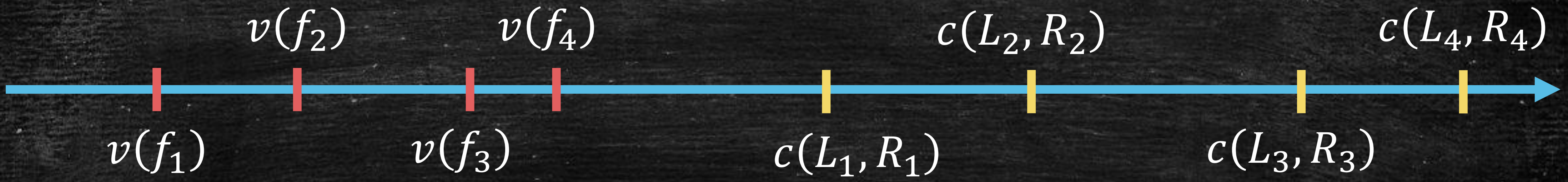
Proving Max-Flow-Min-Cut Theorem

- ✓ Lemma 1. For any flow f and any cut $\{L, R\}$, we have $v(f) \leq c(L, R)$.
  - Formalize the idea that the value of any cut is an upper-bound to the value of any flow.
- ✓ Lemma 2. There exists a cut $\{L, R\}$ such that the flow f output by Ford-Fulkerson Algorithm satisfies $v(f) = c(L, R)$.
  - Concludes Max-Flow-Min-Cut Theorem.
  - Proves the correctness of Ford-Fulkerson Algorithm.

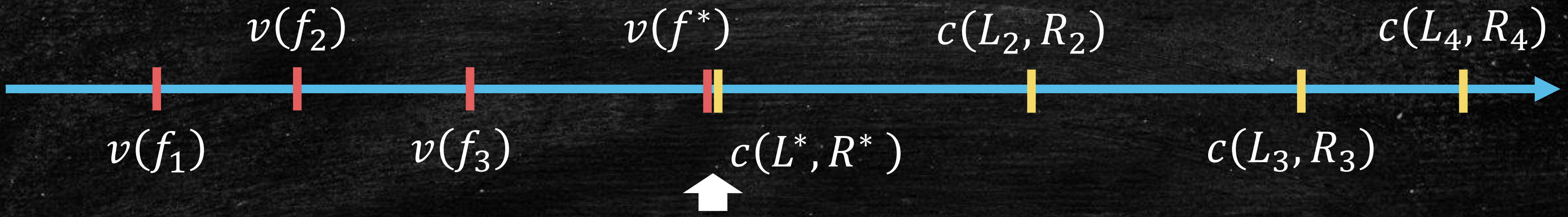


Proof of Max-Flow-Min-Cut Theorem

Lemma 1. For any flow f and any cut $\{L, R\}$, we have $v(f) \leq c(L, R)$.



Lemma 2. There exists a cut $\{L, R\}$ such that the flow f output by Ford-Fulkerson Algorithm satisfies $v(f) = c(L, R)$.



Max-Flow = Min-Cut

Do you know how to find a minimum $s - t$ cut?

Algorithm for finding a minimum cut

Min-Cut Problem: Given $G = (V, E, w)$ and $s, t \in V$, find the s-t cut with the minimum value.

- Solve the max-flow problem with $\forall(u, v) \in E$: $c(u, v) = w(u, v)$
- Let f be the maximum flow and construct G^f
- L: vertices reachable from s in G^f
- $R = V \setminus L$
- Return $\{L, R\}$

Time Complexity?

- Correctness: Max-Flow-Min-Cut Theorem ✓
- Time Complexity:
  - Question 1: Does the algorithm always halt?
  - Question 2: If so, what is the time complexity?

Does the algorithm always halt?

- Let’s start from simplest case: all the capacities are integers.
- Each while-loop iteration increase the value of f by at least 1.
- Thus, the algorithm will halt within f_{max} iterations.

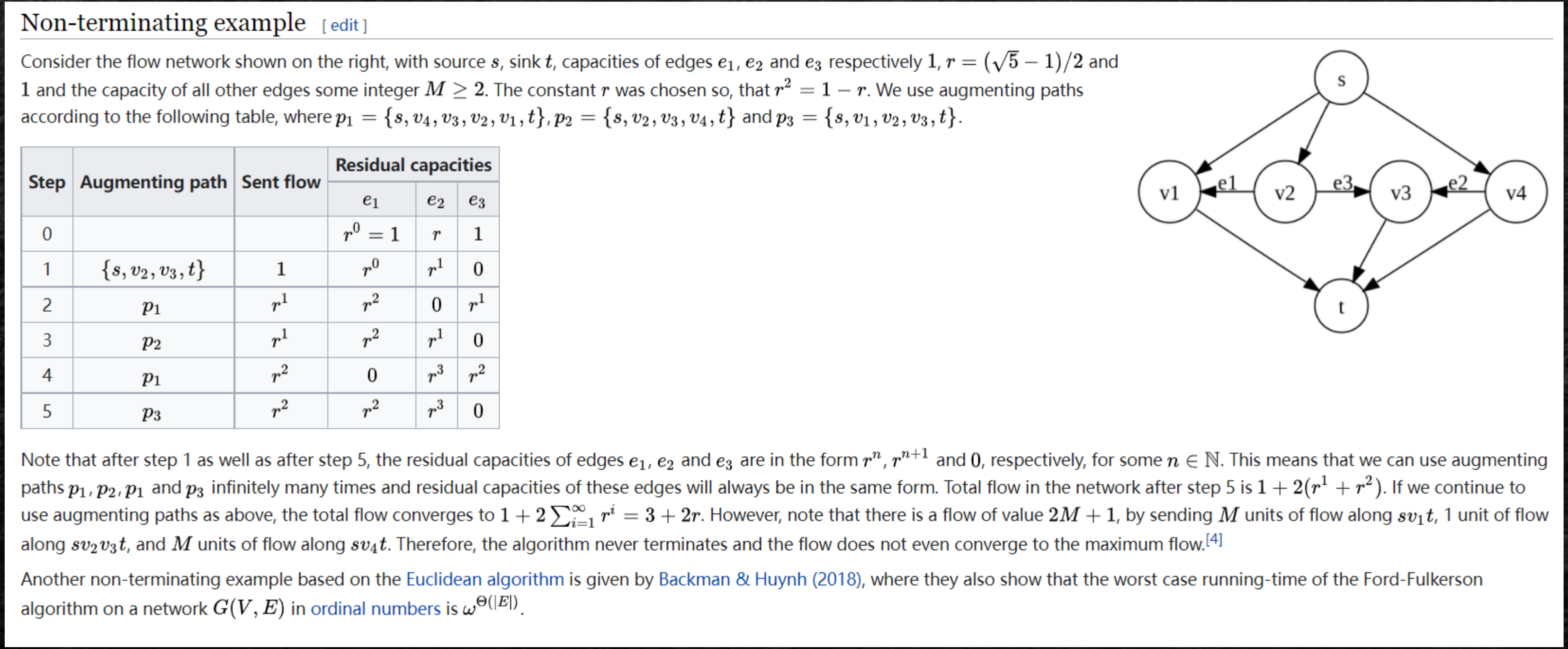
- **Theorem**. If each $c(e)$ is an integer, then the value of the maximum flow f is an integer.
- *Proof.* The value of f is always an integer throughout Ford-Fulkerson Algorithm.

Does the algorithm always halt?

- How about rational capacities?
- Rescale capacities to make them integers.
- Yes, the algorithm will halt!

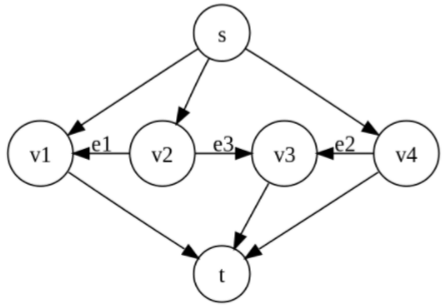
Does the algorithm always halt?

- How about possibly irrational capacities?
- No, the algorithm do not always halt!



Non-terminating example [edit]

Consider the flow network shown on the right, with source s, sink t, capacities of edges e_1, e_2 and e_3 respectively 1, $r = (\sqrt{5}-1)/2$ and 1 and the capacity of all other edges some integer $M \geq 2$. The constant r was chosen so, that $r^2 = 1 - r$. We use augmenting paths according to the following table, where $p_1 = \{s, v_4, v_3, v_2, v_1, t\}$, $p_2 = \{s, v_2, v_3, v_4, t\}$ and $p_3 = \{s, v_1, v_2, v_3, t\}$.



Step	Augmenting path	Sent flow	Residual capacities		
			e_1	e_2	e_3
0			$r^0 = 1$	r	1
1	$\{s, v_2, v_3, t\}$	1	r^0	r^1	0
2	p_1	r^1	r^2	0	r^1
3	p_2	r^1	r^2	r^1	0
4	p_1	r^2	0	r^3	r^2
5	p_3	r^2	r^2	r^3	0

Note that after step 1 as well as after step 5, the residual capacities of edges e_1, e_2 and e_3 are in the form r^n, r^{n+1} and 0, respectively, for some $n \in \mathbb{N}$. This means that we can use augmenting paths p_1, p_2, p_1 and p_3 infinitely many times and residual capacities of these edges will always be in the same form. Total flow in the network after step 5 is $1 + 2(r^1 + r^2)$. If we continue to use augmenting paths as above, the total flow converges to $1 + 2\sum_{i=1}^{\infty} r^i = 3 + 2r$. However, note that there is a flow of value $2M + 1$, by sending M units of flow along sv_1t, 1 unit of flow along sv_2v_3t, and M units of flow along sv_4t. Therefore, the algorithm never terminates and the flow does not even converge to the maximum flow.[4]

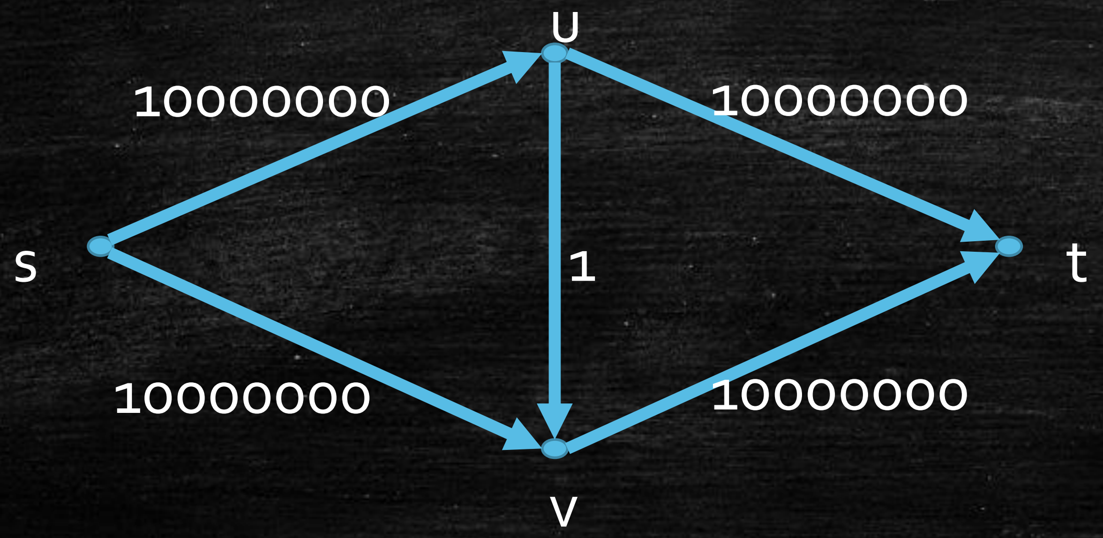
Another non-terminating example based on the Euclidean algorithm is given by Backman & Huynh (2018), where they also show that the worst case running-time of the Ford-Fulkerson algorithm on a network $G(V, E)$ in ordinal numbers is $\omega^{\Theta(|E|)}$.

Time Complexity?

- Assume all capacities are integers, what is the time complexity?
- Each iteration requires $O(|E|)$ time:
  - $O(|E|)$ is sufficient for finding p, updating f and G^f
- There are at most f_{max} iterations.
- Overall: $O(|E| \cdot f_{max})$
- Can we analyze it better?

Time Complexity?

- Can we analyze it better?
- It depends on how you choose p in each iteration!
- The complexity bound $O(|E| \cdot f_{max})$ is tight if choices of p are not carefully specified!



Method vs Algorithm

- Different choices of augmenting paths p give different implementation of Ford-Fulkerson.
- The description of Ford-Fulkerson Algorithm is incomplete.
- For this reason, it is sometimes called Ford-Fulkerson Method.
- Next Lecture Preview: Edmonds-Karp Algorithm, which implement Ford-Fulkerson Method with time complexity $O(|V| \cdot |E|^2)$.

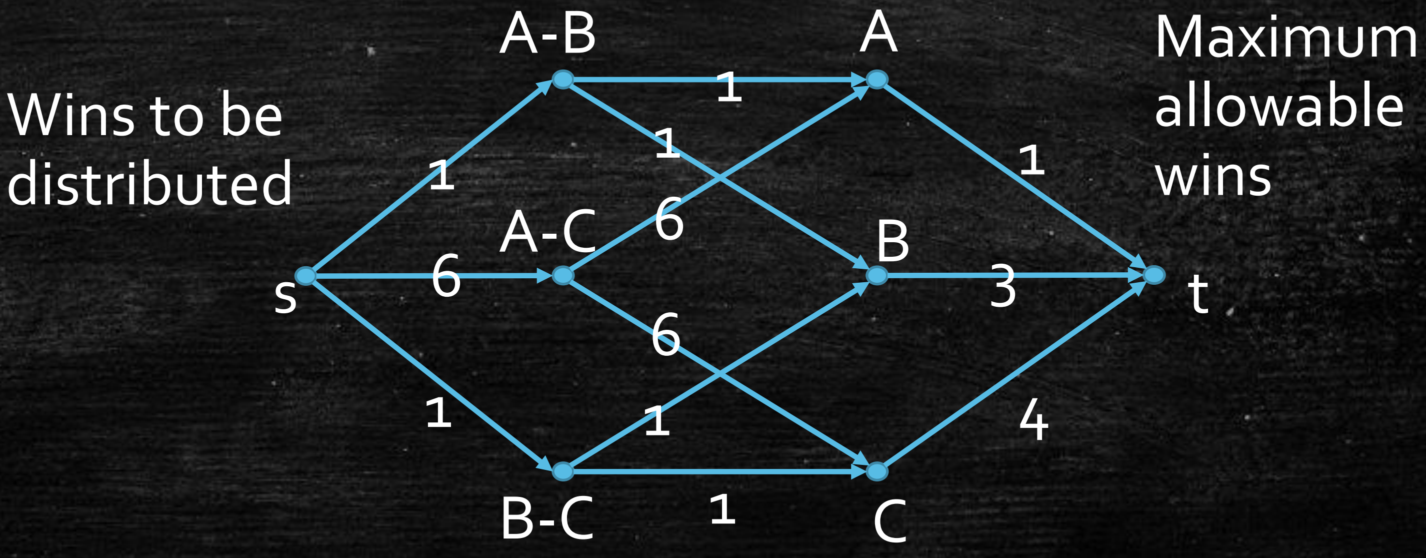
Integrality Theorem.

- **Theorem**. If each $c(e)$ is an integer, then the value of the maximum flow f is an integer.
- **Understanding**. If each $c(e)$ is an integer, there exists a flow f to maximize the total flow value, such that each edge's flow is an integer.
- Why?

Applications of Integrality Theorem

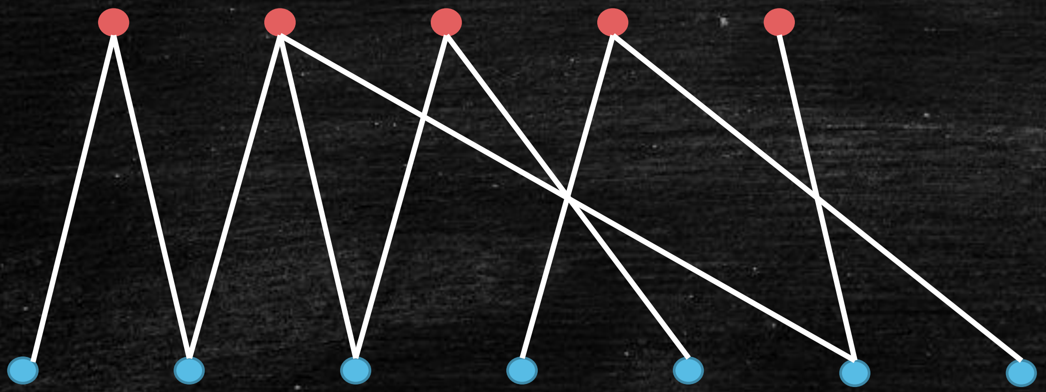
- **Theorem**. If each $c(e)$ is an integer, then the value of the maximum flow f is an integer.
- Application 1: Tournament example you have seen in the last lecture.
- The max-flow f must satisfy $\forall e$: $f(e) \in \mathbb{Z}$.

	Wins	Max Num of Additional Wins
A	40	1
B	38	3
C	37	4
D	41	



Application 2: Maximum Bipartite Matching

- Top vertices are girls, bottom vertices are boys.
- An edge represent a possible match for a boy and a girl.
- Problem: find a maximum matching for boys and girls.



Maximum Bipartite Matching - Formal

- Given a graph $G = (V, E)$, a matching M is a subset of edges that do not share vertices in common.
- The size of a matching is the number of edges in it.
- Problem: Given a bipartite graph $G = (A, B, E)$ find a matching with the maximum size.

Dessert

- A graph is regular if all the vertices have the same degree.
- A matching is perfect if all the vertices are matched.
- Prove that a regular bipartite graph always has a perfect matching.

Hall's Marriage Theorem

- Consider the matching problem on a bipartite graph $G = (A, B, E)$.
- For a subset $S \subseteq A$, let $N(S) \subseteq B$ be the set of vertices that are incident to vertices in S.
- **Hall's Marriage Theorem**. There exists a matching of size $|A|$ if and only if $|S| \leq |N(S)|$ for every $S \subseteq A$.

Proof of Hall’s Marriage Theorem

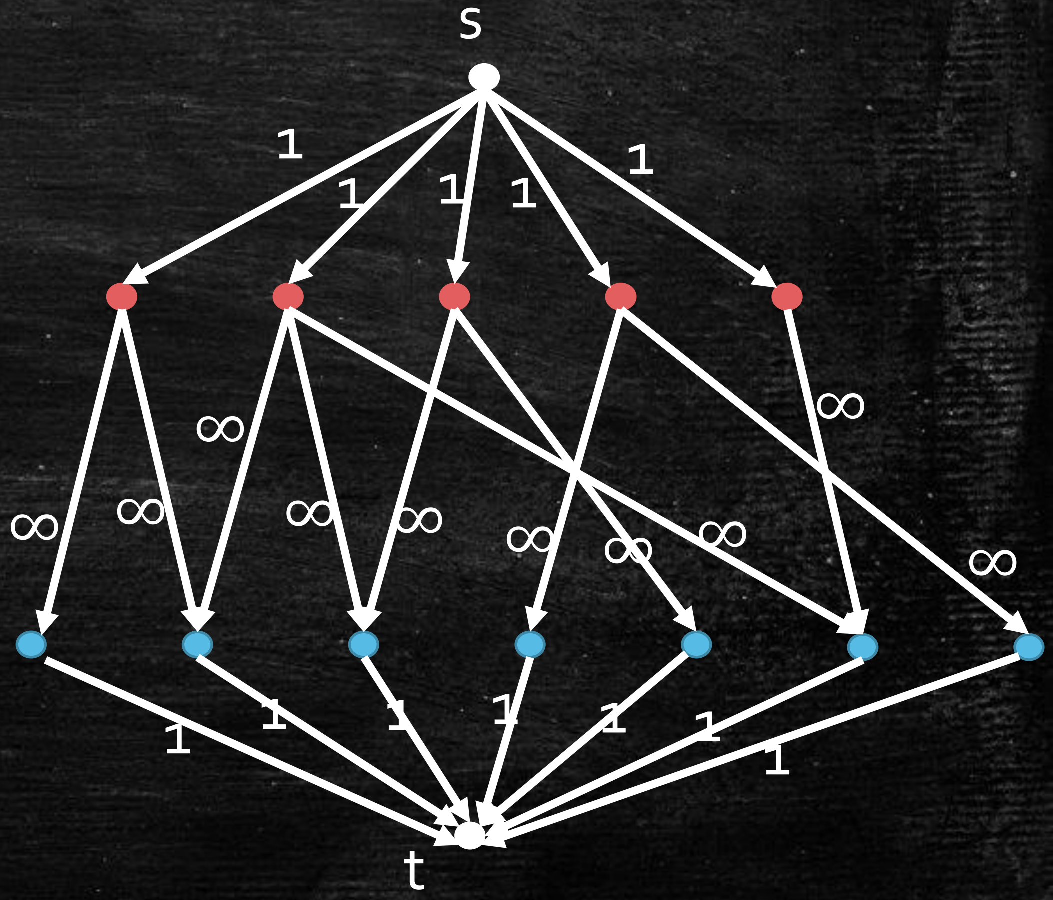
Exist a matching of size $|A| \Rightarrow \forall S: |S| \leq |N(S)|$.

- Suppose for the sake of contraction that $\exists S: |S| > |N(S)|$.
- There is no way to match all the vertices in S.
- Thus, there is no way to match all the vertices in A.

Proof of Hall's Marriage Theorem

Exist a matching of size $|A| \Longleftarrow \forall S: |S| \leq |N(S)|$.

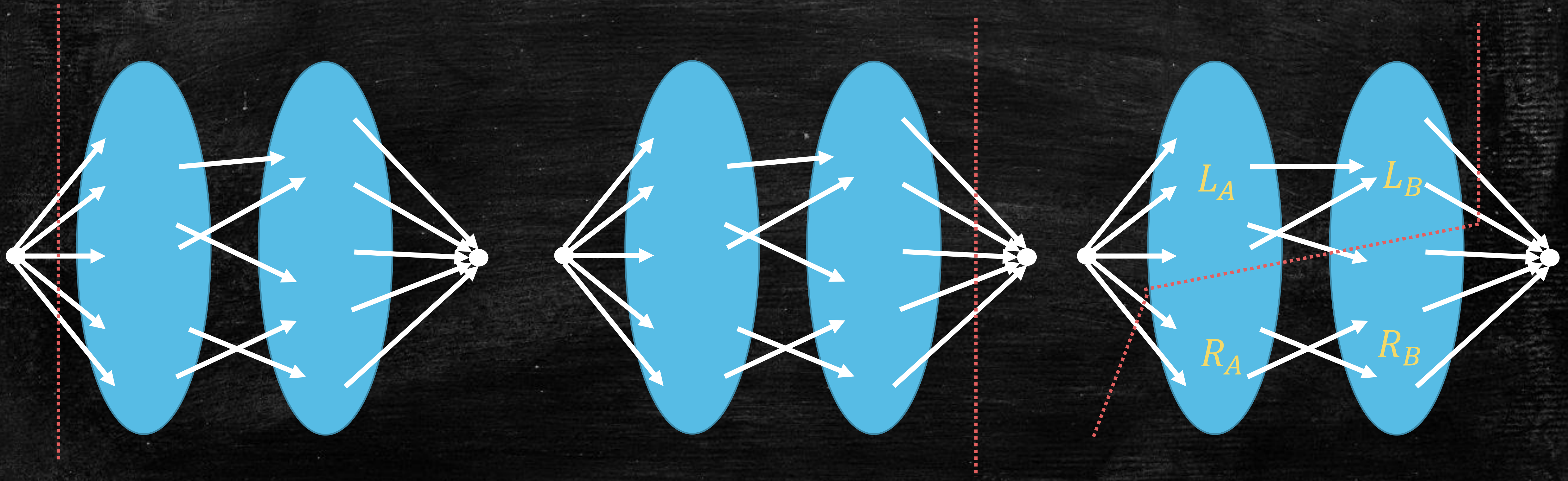
- Given $\forall S: |S| \leq |N(S)|$, suppose the maximum matching has size $M < |A|$.
- The maximum flow has value M.
  - Integrality Theorem
- The minimum cut has value M.
  - Max-Flow-Min-Cut Theorem



Proof of Hall's Marriage Theorem

Three cases for minimum cut $\{L, R\}$:

- 1) $L = \{s\}, R = A \cup B \cup \{t\}$, 2) $L = \{s\} \cup A \cup B$, 3) $L_A, L_B, R_A, R_B \neq \emptyset$.

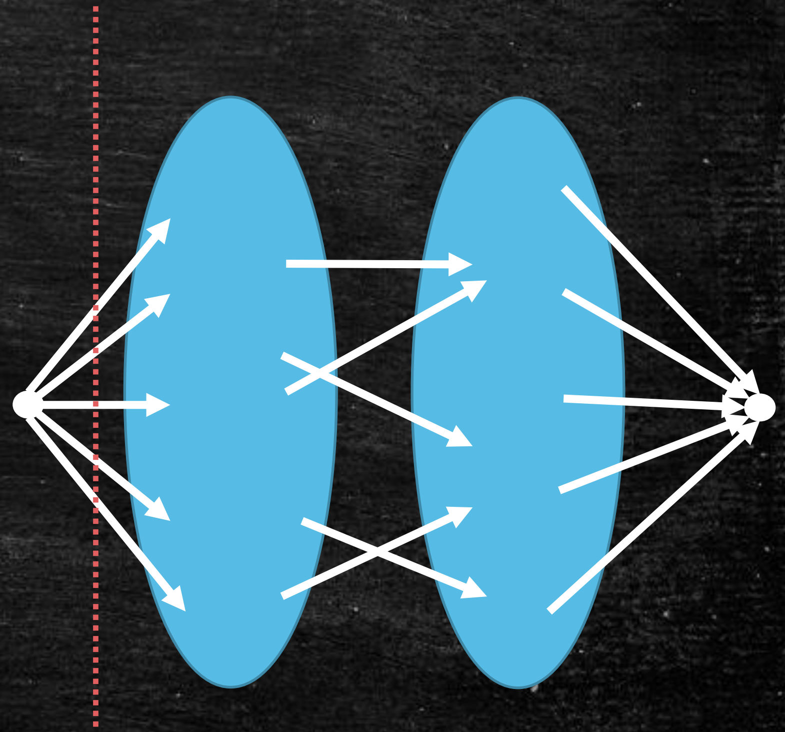


Proof of Hall's Marriage Theorem

Exist a matching of size $|A| \Longleftarrow \forall S: |S| \leq |N(S)|$.

Case 1) $L = \{s\}, R = A \cup B \cup \{t\}$:

- The minimum cut has size $|A|$
- But we have assumed the minimum cut has size $M < |A|$.
- Case 1) cannot happen!

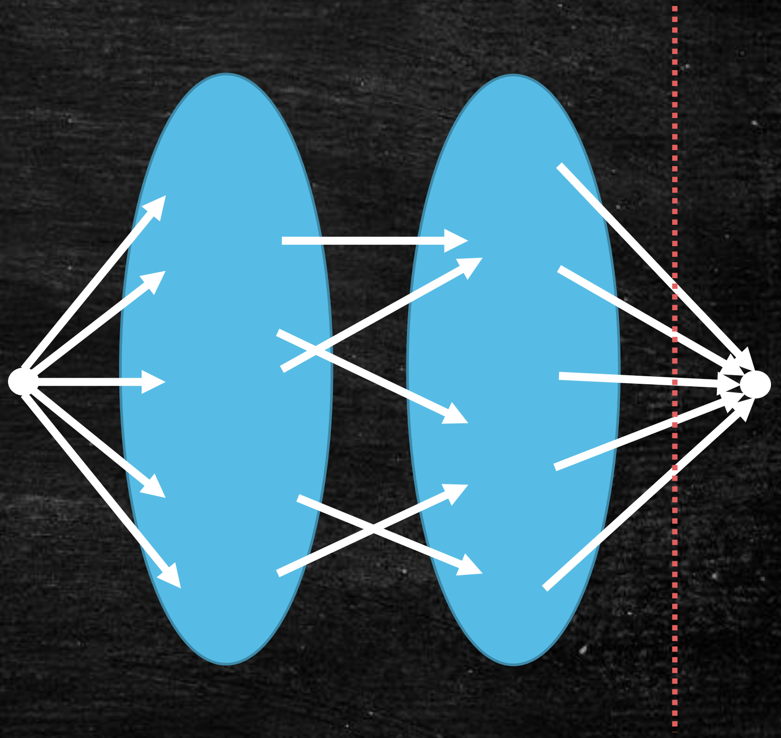


Proof of Hall’s Marriage Theorem

Exist a matching of size $|A| \Leftarrow \forall S: |S| \leq |N(S)|$.

Case 2) $L = \{s\} \cup A \cup B, R = \{t\}$:

- The minimum cut has size $|B|$
- We have assumed the minimum cut has size M, so $|B| = M < |A|$.
- Contradiction with $|A| \leq |N(A)| \leq |B|$

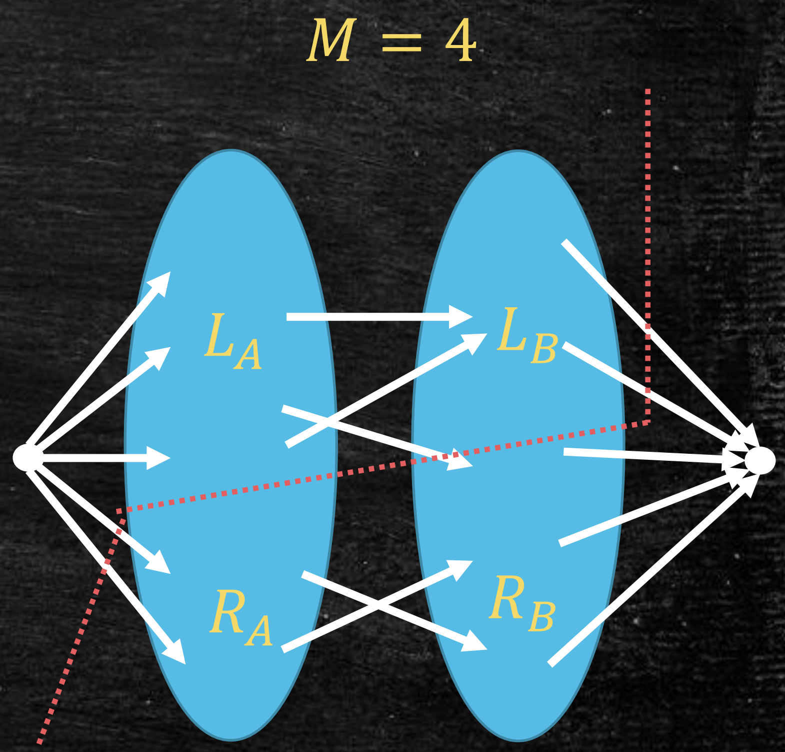


Proof of Hall's Marriage Theorem

Exist a matching of size $|A| \Longleftarrow \forall S: |S| \leq |N(S)|$.

Case 3) $L_A, L_B, R_A, R_B \neq \emptyset$:

- Minimum cut size: $M = |L_B| + |R_A|$
- We also have $|L_A| + |R_A| = |A|$
- $M < |A| \Longrightarrow |L_A| > |L_B|$
- No edge can go from L_A to R_B
  - Such an edge has weight ∞
- Thus, $N(L_A) \subseteq L_B$, which implies $|N(L_A)| \leq |L_B| < |L_A|$
- Contradicts to our assumption



Today's Lecture

- Max-Flow-Min-Cut Theorem
  - Equivalence of Max-Flow and Min-Cut problems
  - Correctness of Ford-Fulkerson Method
- Flow Integrality Theorem
  - Follows immediately from Ford-Fulkerson Method
- Maximum Bipartite Matching
  - Translate the problem to Max-Flow applying integrality theorem
  - Hall's Marriage Theorem: application of Max-Flow-Min-Cut Theorem
- Edmonds-Karp Algorithm