CS630 Graduate Algorithms

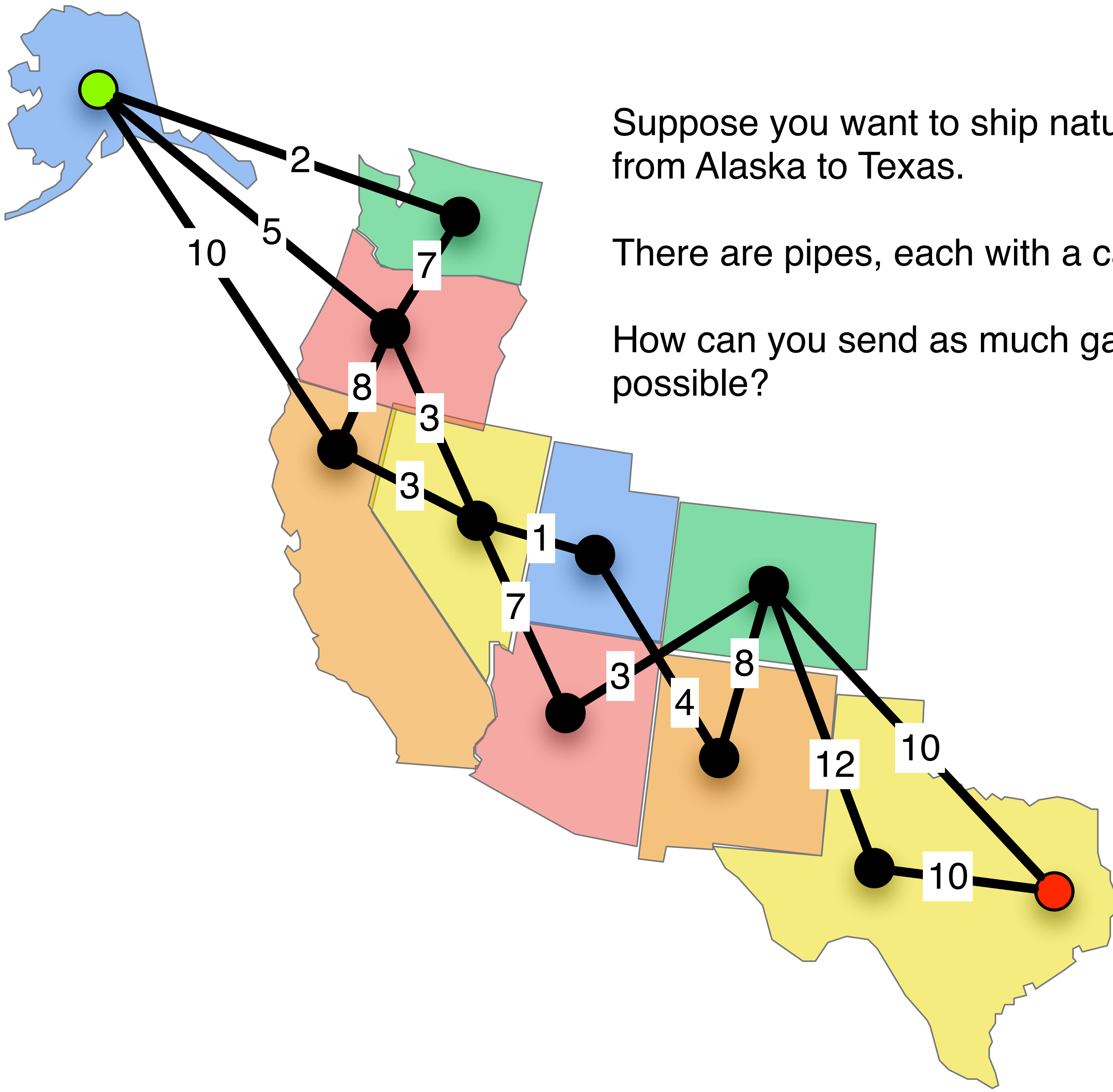
September 10, 2024

by Dora Erdos and Jeffrey Considine

Today:

- Maximum flow proofs
- More reductions to maximum flow

Shipping through a pipeline



Suppose you want to ship natural gas from Alaska to Texas.

There are pipes, each with a capacity.

How can you send as much gas as possible?

image by Carl Kingsford , Univ. of Maryland, College Park

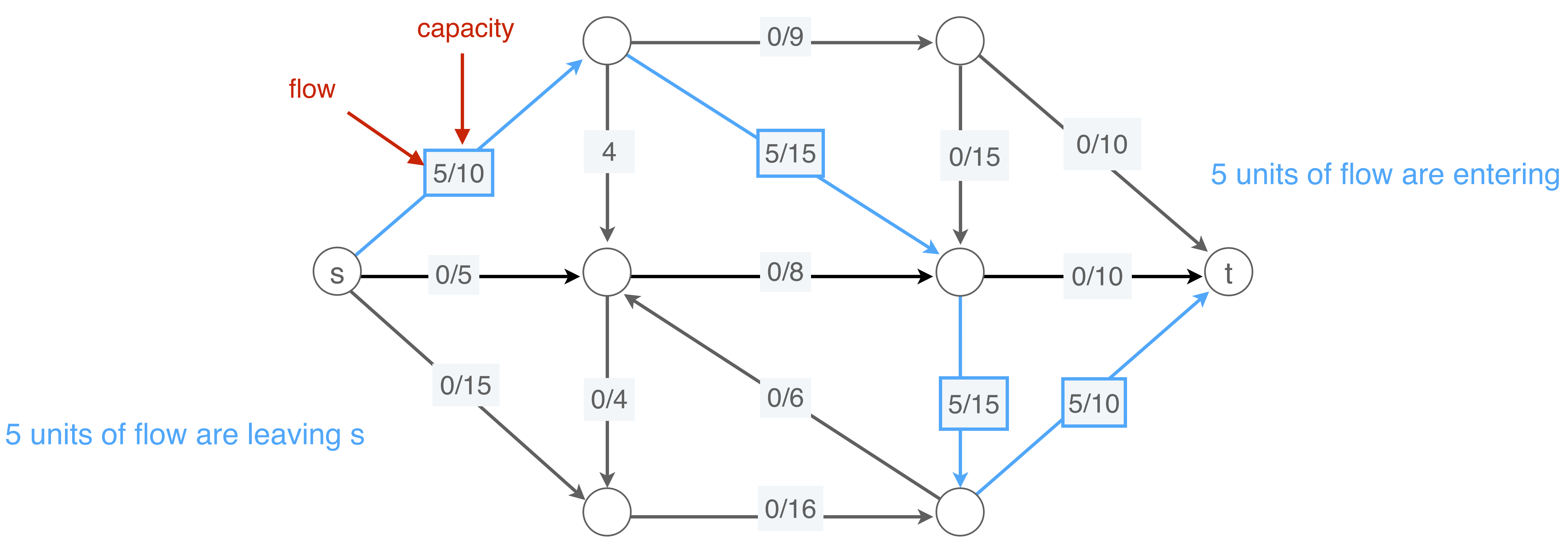
Recap - Network flow

A flow network is a tuple $G = (V, E, s, t, c)$.

- Directed graph (digraph) (V, E) with source $s \in V$ and sink $t \in V$.
- Non-negative capacity $c(e)$ for each $e \in E$.
  - intuition: the capacity is the *throughput* of each edge

Def. An st-flow (flow) f is a function that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ [capacity]
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [flow conservation]



Recap - Maximum-flow problem

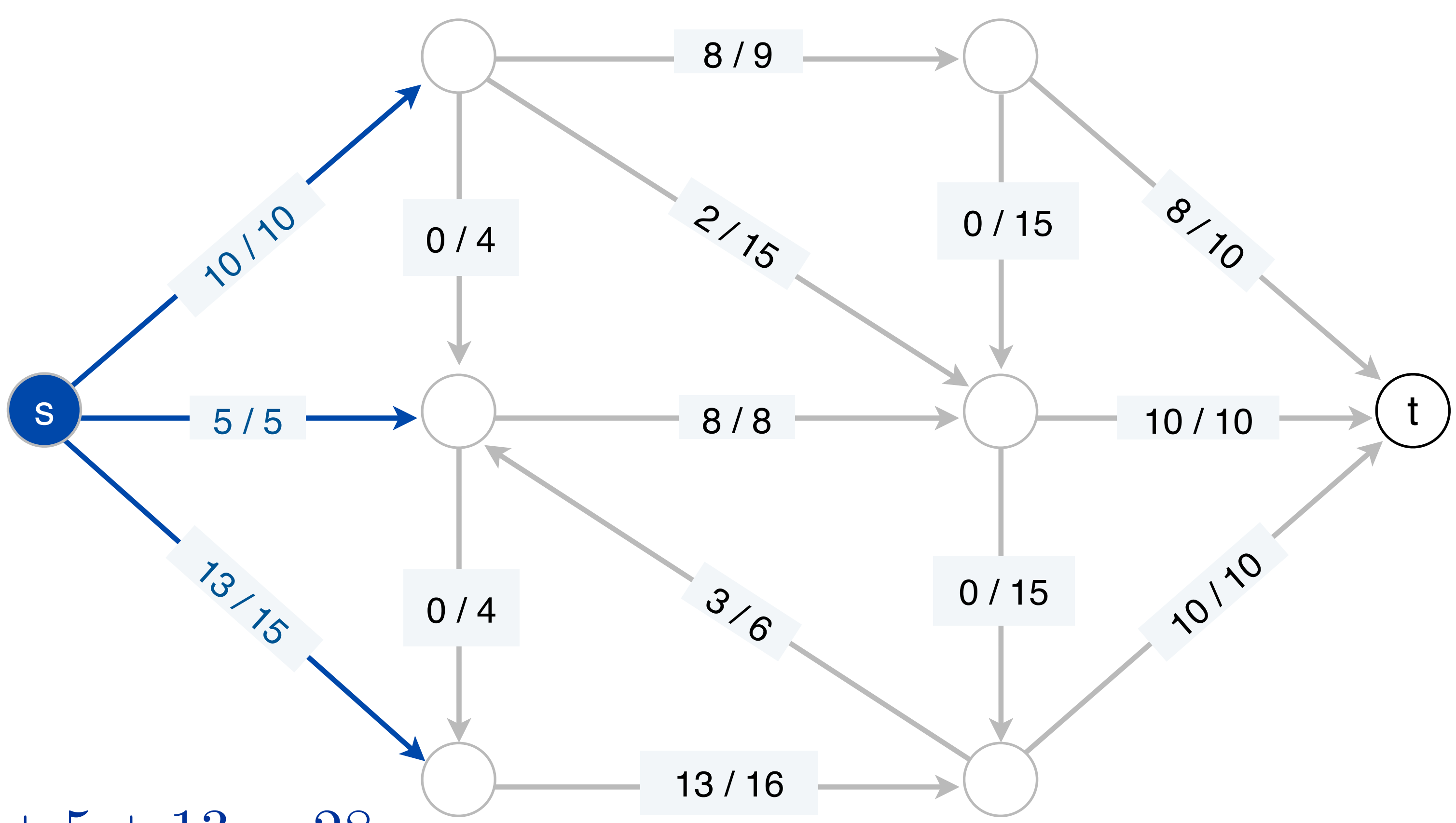
Def. An st-flow (flow) f is a function that satisfies:

- For each $e \in E$: $\quad 0 \leq f(e) \leq c(e)$ [capacity]
- For each $v \in V - \{s, t\}$: $\quad \sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [flow conservation]

Def. The value of a flow f is: $\quad val(f) = \sum_{edges\ (s,u)} f(s,u) = \sum_{edges\ (v,t)} f(v,t)$

Max-Flow problem:

Given a directed graph with edge capacities, find the maximum value flow.



$val(f) = 10 + 5 + 13 = 28$

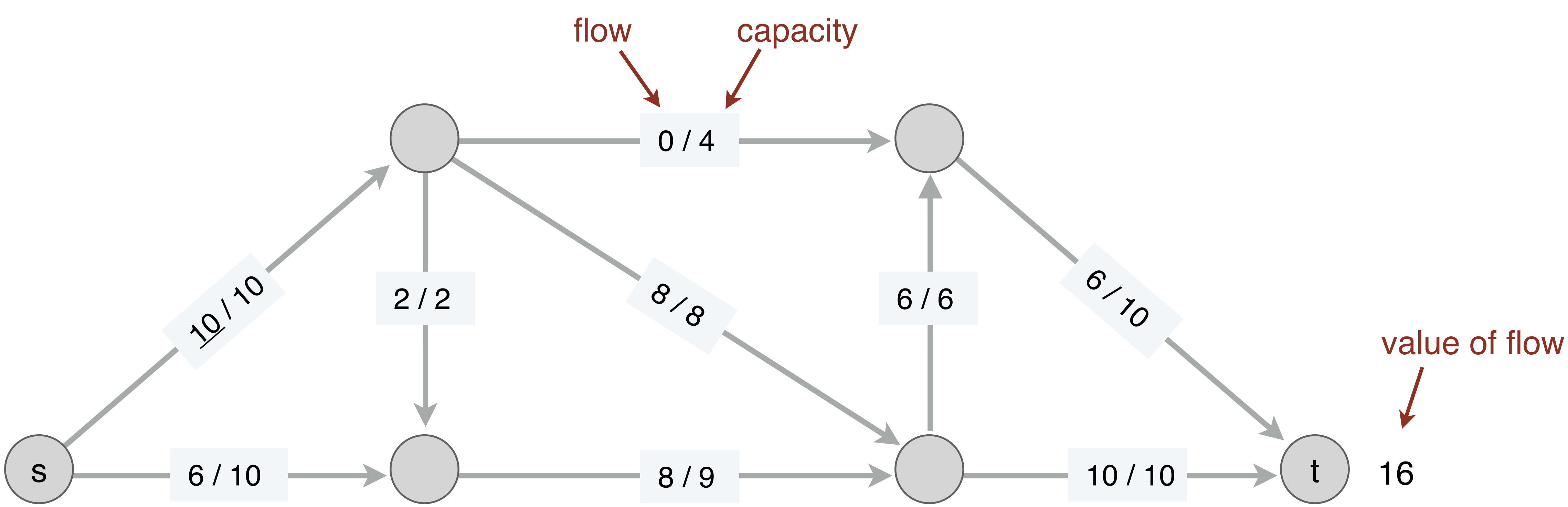
Recap - Augmenting paths

Observation: we can send additional flow along an st-path if there is free capacity along every edge.

Augmenting path: an st-path with free capacity, along which we augment the flow.

The value of this flow is 16.

But this is not the max flow in this network.



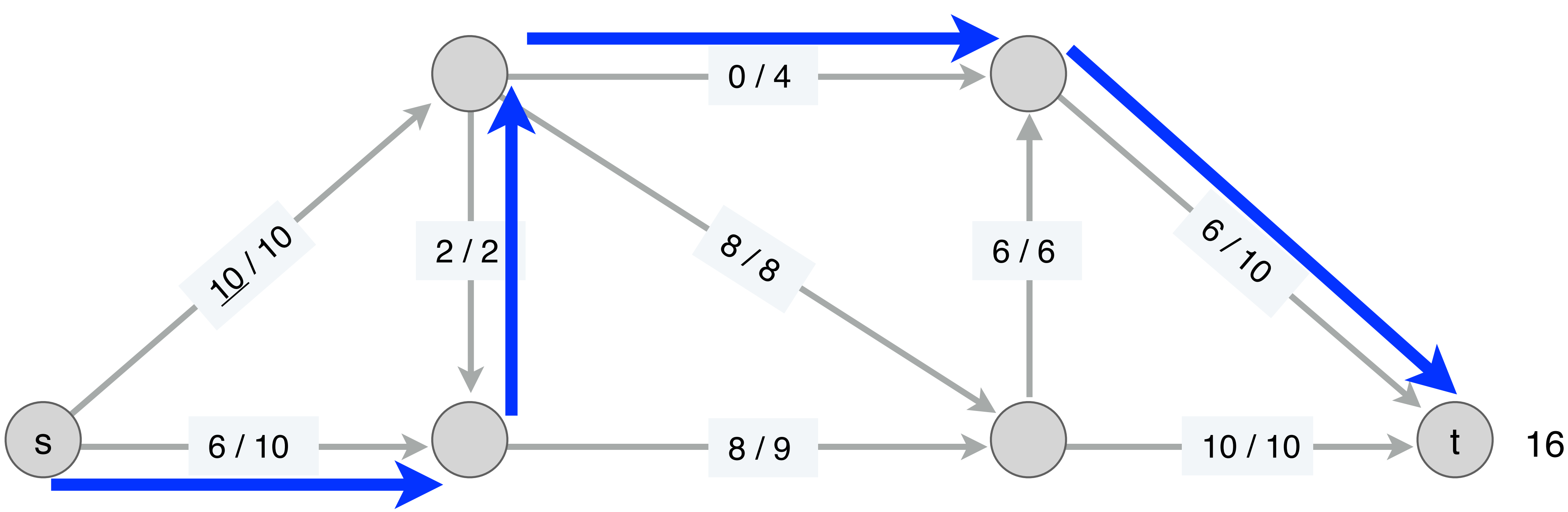
Recap - Augmenting paths

Observation: we can send additional flow along an st-path if there is free capacity along every edge.

Augmenting path: an st-path with free capacity, along which we augment the flow.

The value of this flow is 16.

But this is not the max flow in this network.



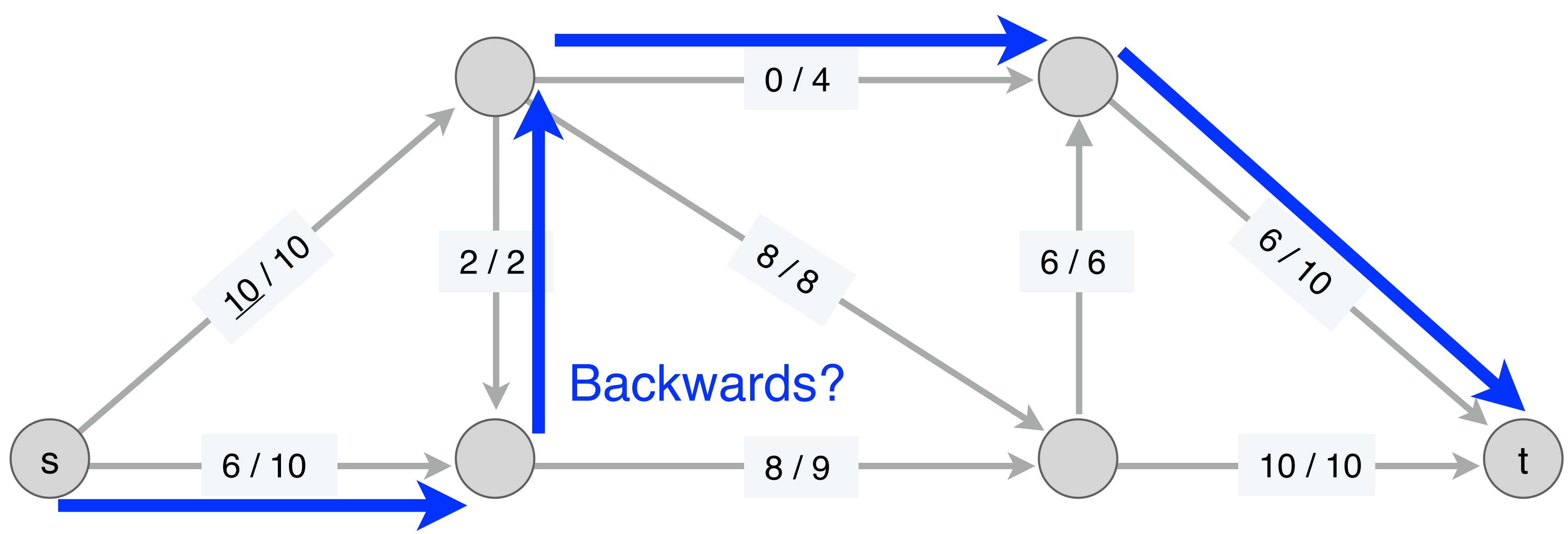
Recap - Augmenting paths

Observation: we can send additional flow along an st-path if there is free capacity along every edge.

Augmenting path: an st-path with free capacity, along which we augment the flow.

The value of this flow is 16.

But this is not the max flow in this network.



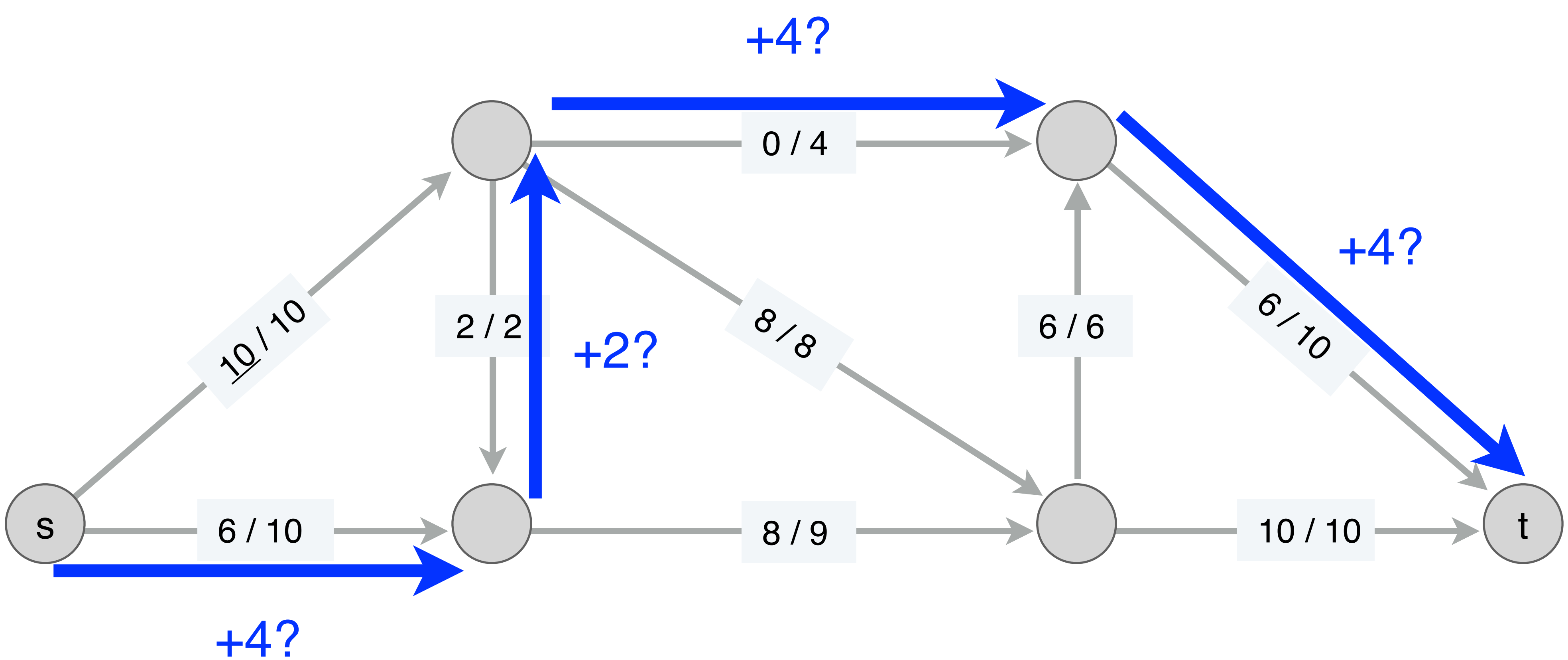
Recap - Augmenting paths

Observation: we can send additional flow along an st-path if there is free capacity along every edge.

Augmenting path: an st-path with free capacity, along which we augment the flow.

The value of this flow is 16.

But this is not the max flow in this network.



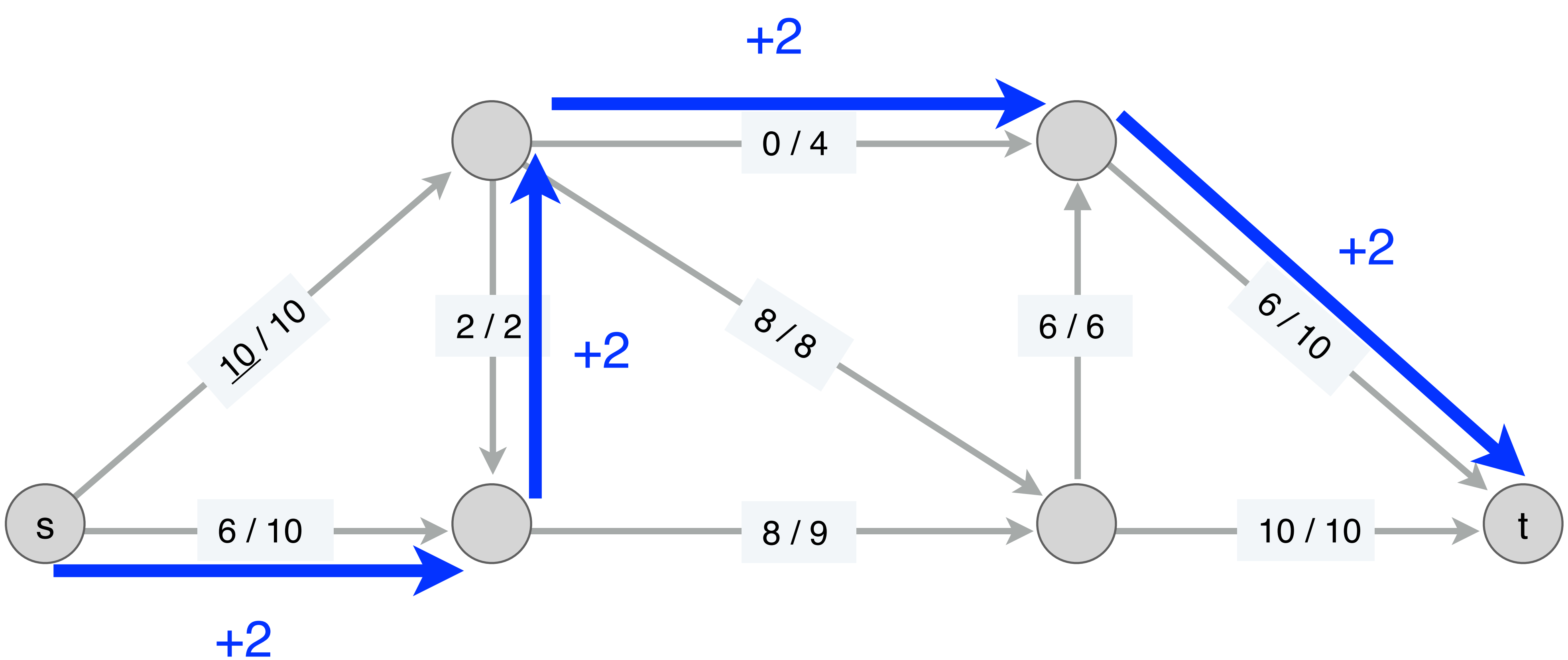
Recap - Augmenting paths

Observation: we can send additional flow along an st-path if there is free capacity along every edge.

Augmenting path: an st-path with free capacity, along which we augment the flow.

The value of this flow is 16.

But this is not the max flow in this network.



Intuition Towards Maximum Flow Solutions

Suppose we have a (possibly suboptimal) flow f that we want to compare to an (unknown) maximum flow $f_{\max}$.

What can we say about their difference, $f_\Delta(u,v) = f_{\max}(u,v) - f(u,v)$?

- For each $e \in E$: $-f(u,v) \le f_\Delta(u,v) \le c(u,v) - f(u,v)$ [residual capacity]
- For each $v \in V - \{s,t\}$: $\displaystyle\sum_{e \text{ in to } v} f_\Delta(e) = \sum_{e \text{ out of } v} f_\Delta(e)$ [flow conservation]
- Finally, $\displaystyle\sum_{u \in V} f_\Delta(s,u) = \sum_{v \in V} f_\Delta(v,t) \ge 0$ [max flow ≥ others]

This last equation is an equality if f is a maximum flow, and a strict inequality otherwise. (Both from defining $f_{\max}$ to be a maximum flow.)

- Looks like another flow problem, except $f_\Delta(u,v)$ can have negative values.
- Want $val(f_\Delta(u,v)) > 0$ to improve on f_1

Recap - Residual graph

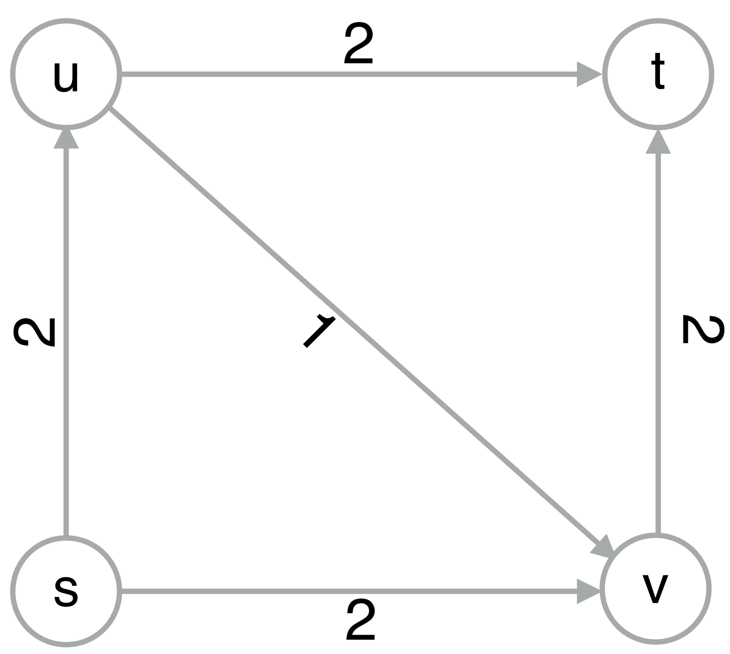
G: nodes V, edges E, capacities c(e)

residual graph G_f:

- nodes V
- edges $E \cup E^{reverse}$
- residual capacity

$$c_f(e) = \begin{cases} c(e) - f(e) & e \in E \\ f(e) & e \in E^{reverse} \end{cases}$$

- This is equivalent to the flow problem from f_Δ , but replacing cases where capacity with negative with positive capacity in the opposite direction.
- So now we have a flow problem where all the capacities are non-negative again.



Recap - Residual graph

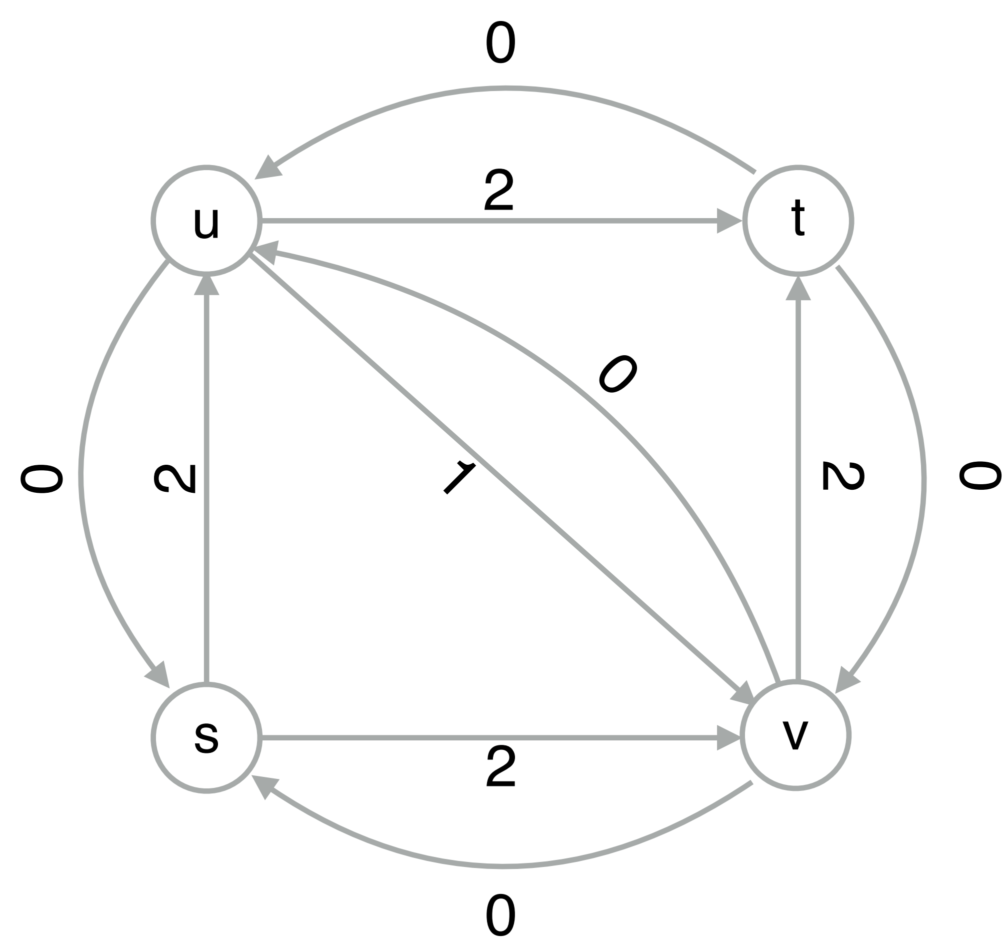
G: nodes V, edges E, capacities c(e)

residual graph G_f:

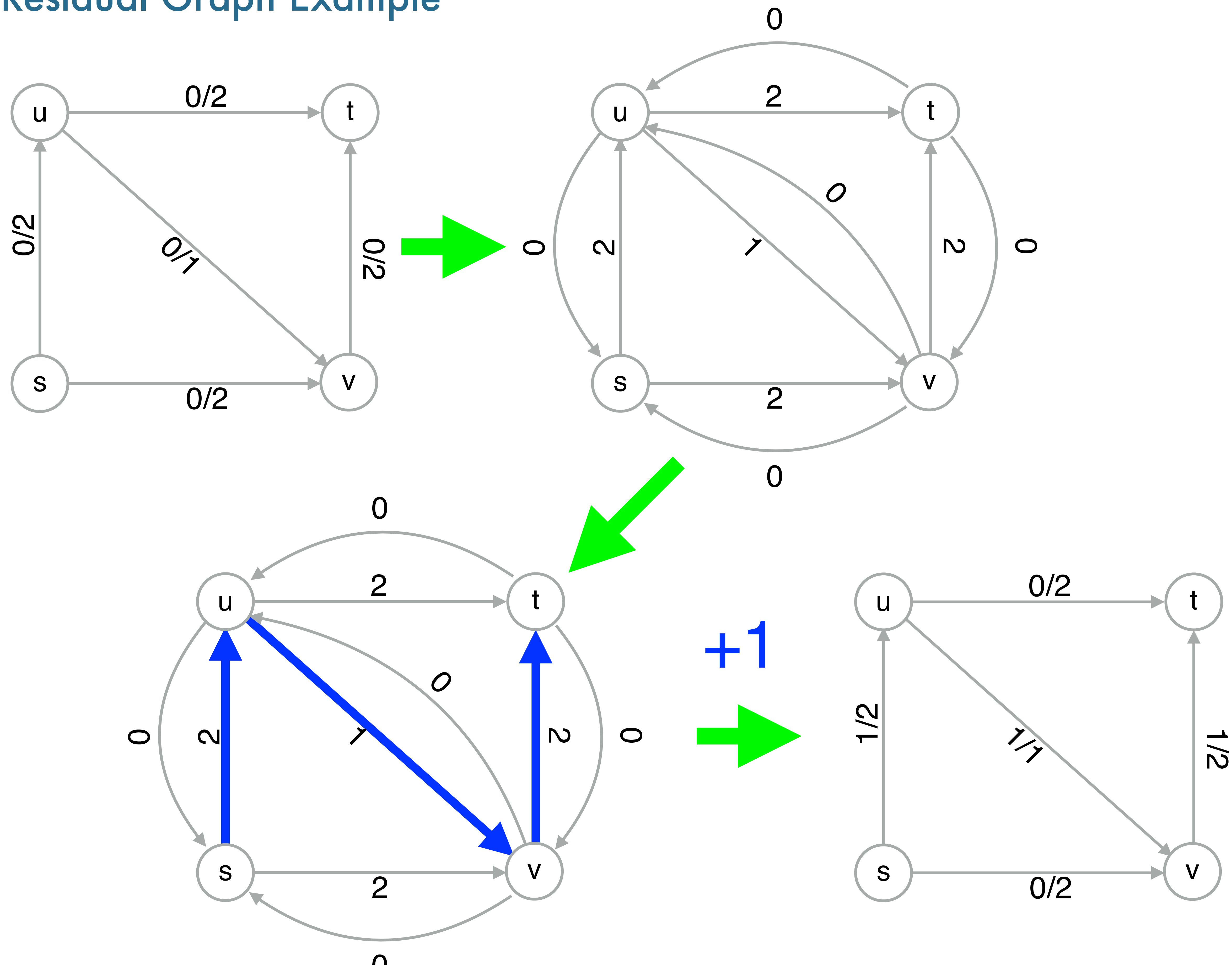
- nodes V
- edges $E \cup E^{reverse}$
- residual capacity

$$c_f(e) = \begin{cases} c(e) - f(e) & e \in E \\ f(e) & e \in E^{reverse} \end{cases}$$

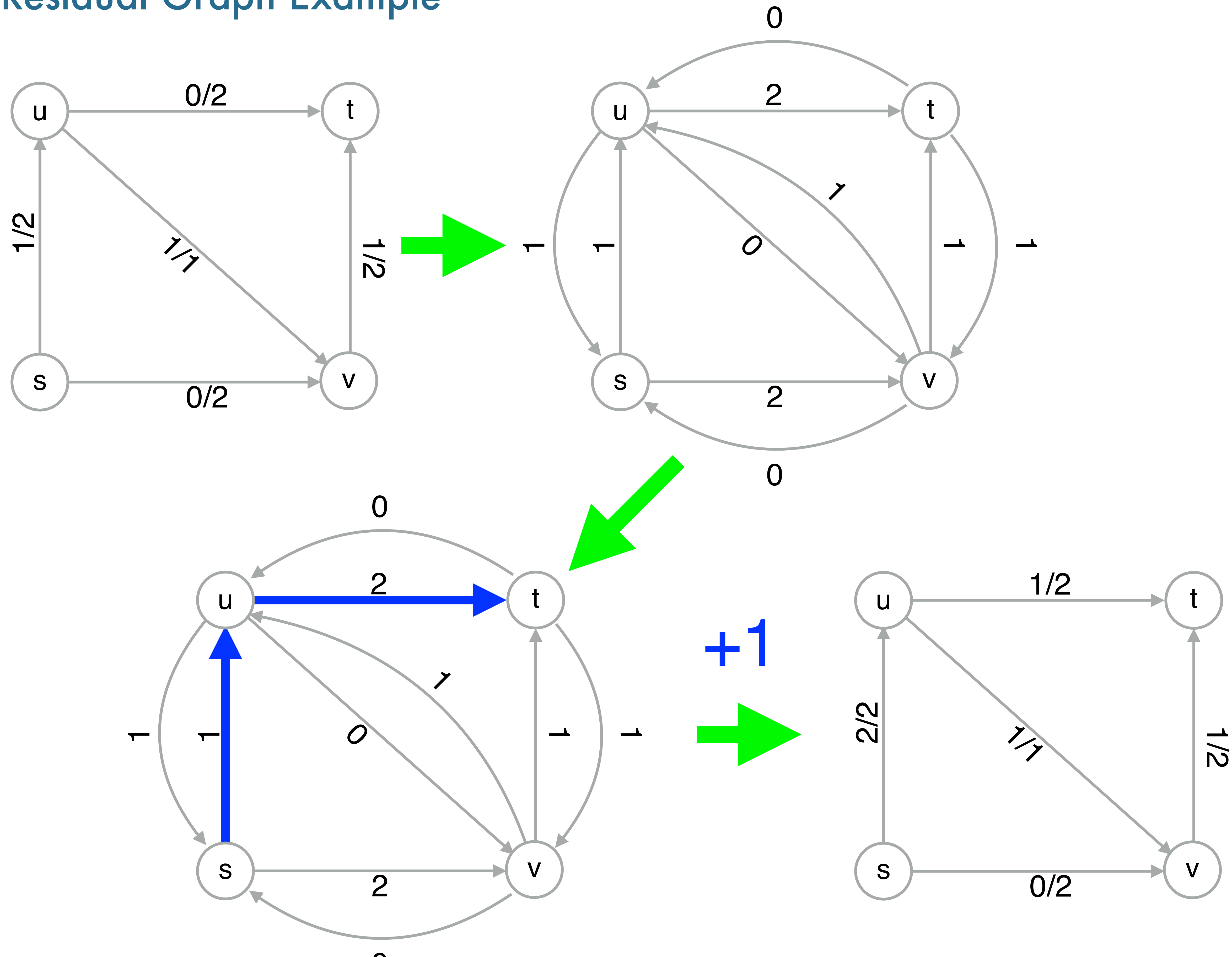
- This is equivalent to the flow problem from f_Δ, but replacing cases where capacity with negative with positive capacity in the opposite direction.
- So now we have a flow problem where all the capacities are non-negative again.



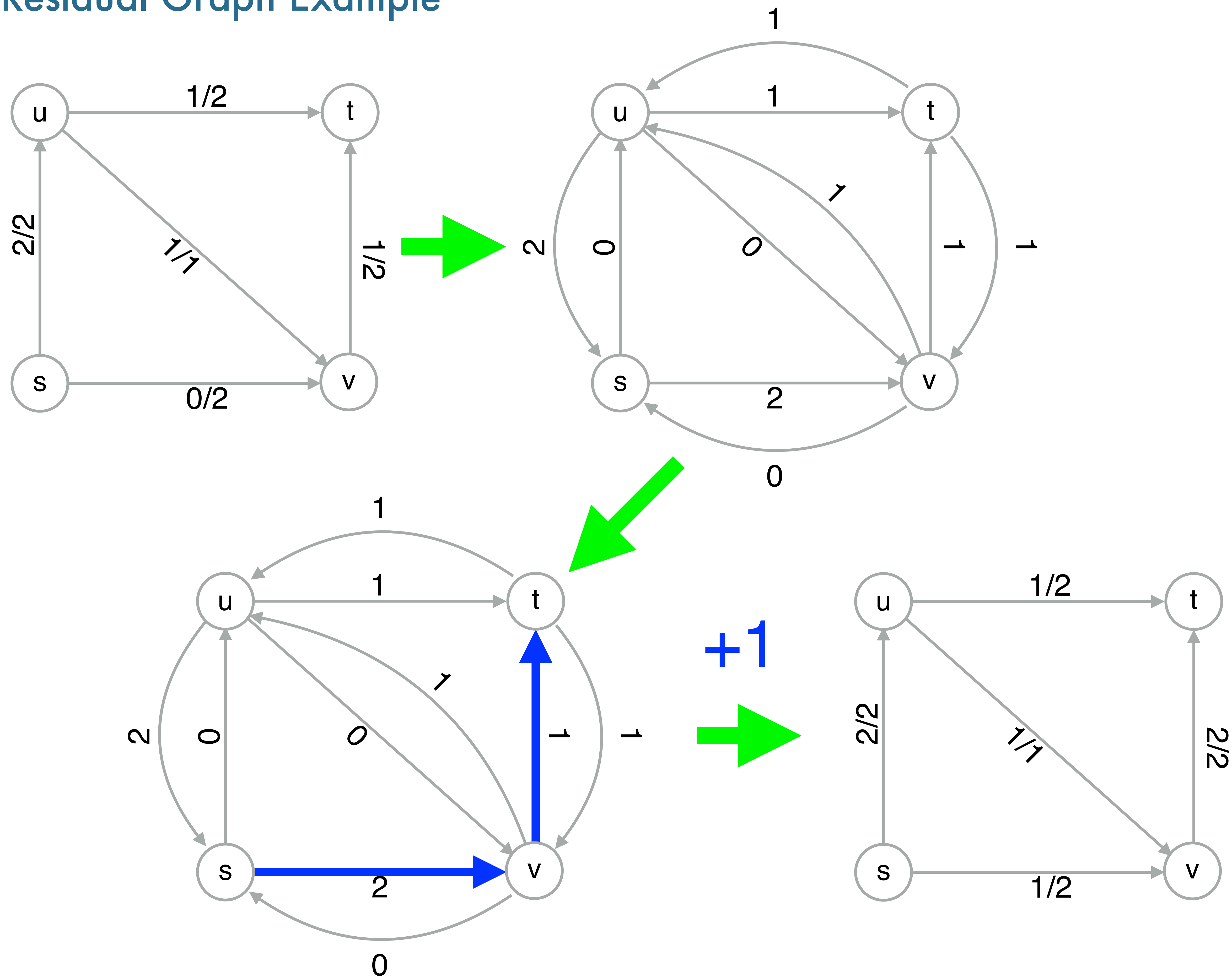
Residual Graph Example



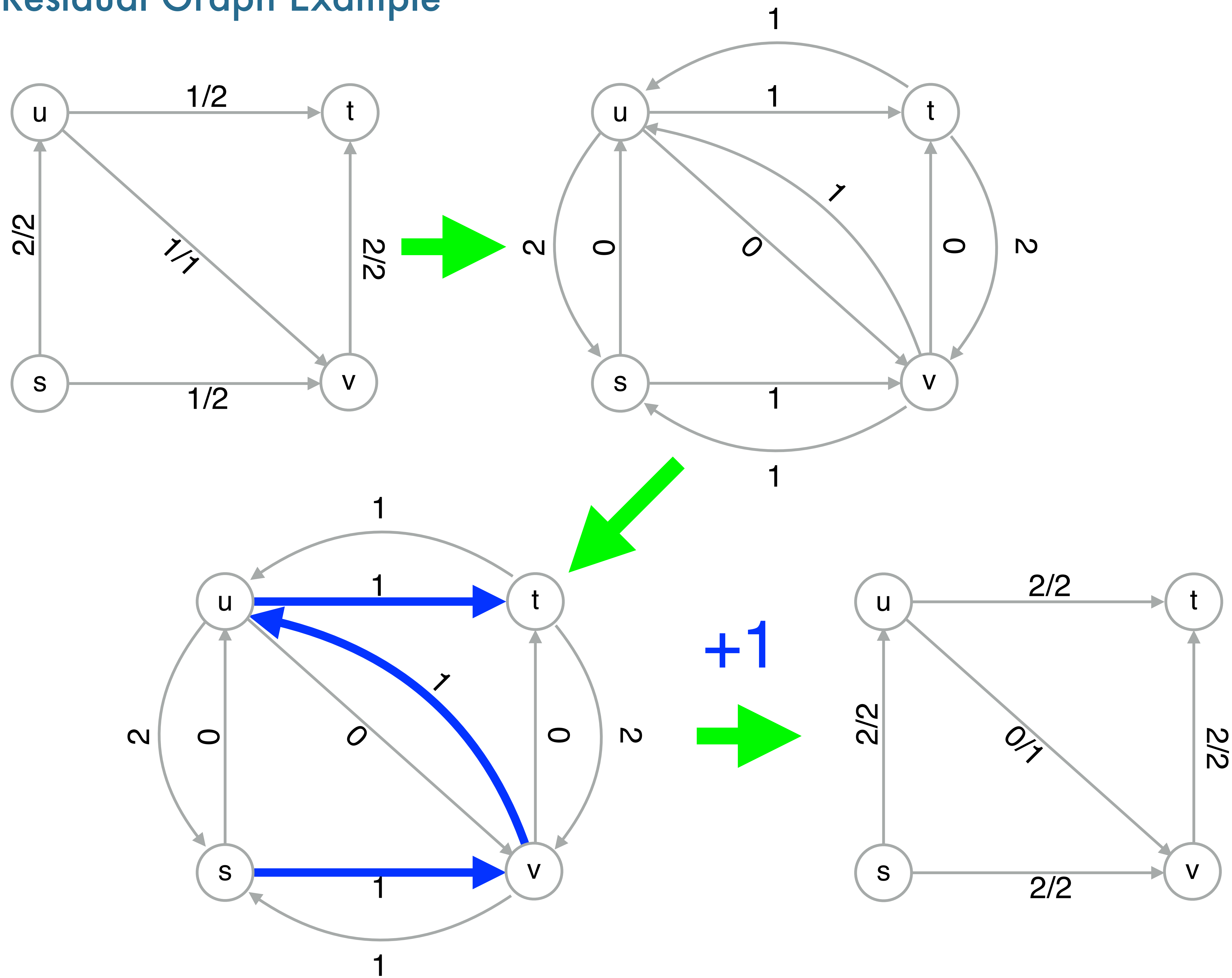
Residual Graph Example



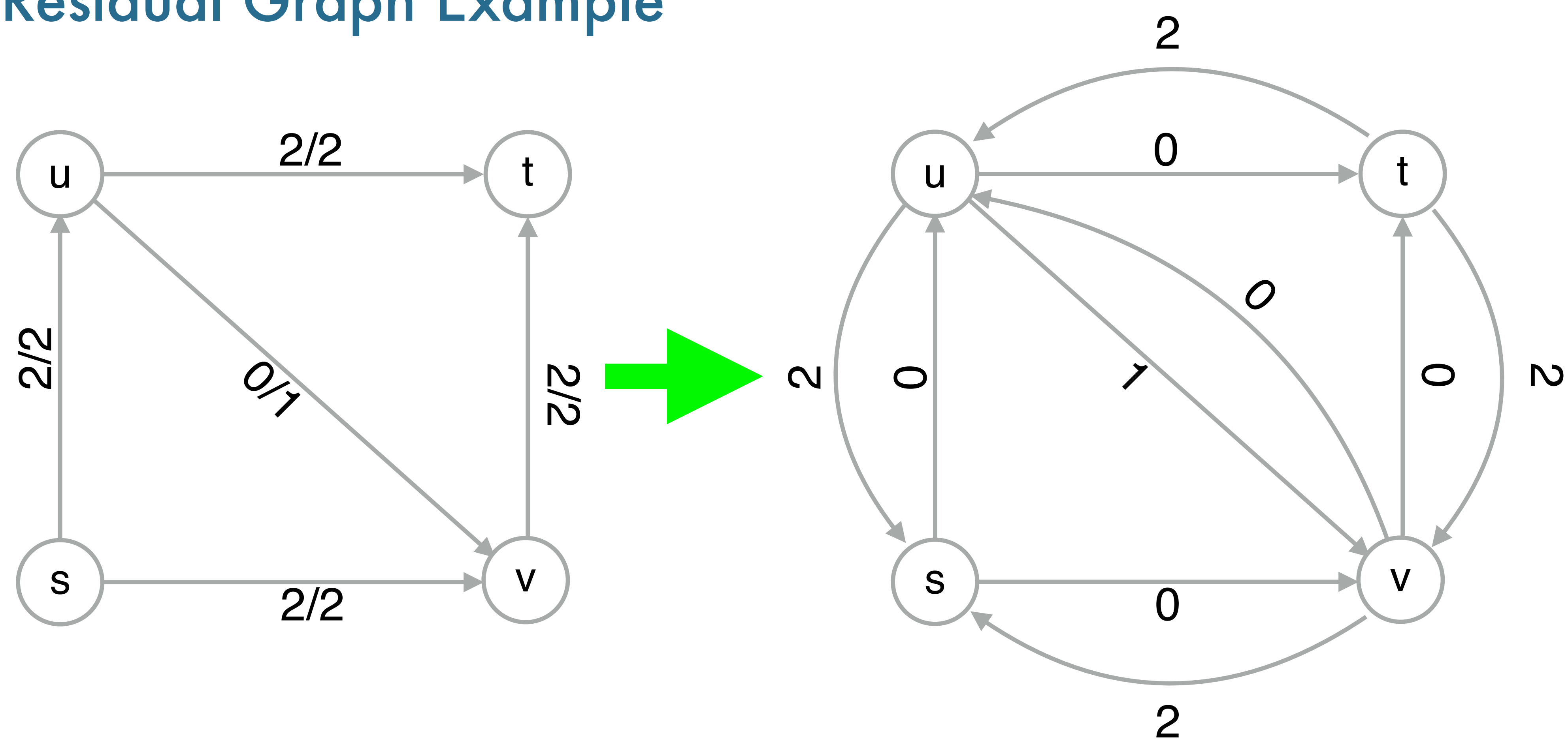
Residual Graph Example



Residual Graph Example



Residual Graph Example



No more augmenting paths → done!

Pause for TopHat Quiz

Augmenting Path Theorem

A flow f is a maximum flow if and only if its residual graph G_f does not have any augmenting paths.

Claims:

1. f is a maximum flow $\iff \max(val(f_\Delta)) = 0$.
   (by definition of f_Δ)
2. $\max(val(f_\Delta)) = 0 \iff$ the maximum flow of G_f is zero.
   (by construction of G_f)
3. The maximum flow of G_f is zero $\iff G_f$ has no augmenting paths.
   (next slide)

These three claims together prove the theorem.

The maximum flow of is zero $\iff G_f$ has no augmenting paths.

Actually proving

The maximum flow of G_f is positive $\iff G_f$ has augmenting paths.

- If there is an augmenting path,

1. Construct a positive flow using that augmenting path.
   (by definition)

- If the maximum flow is greater than zero, can construct an augmenting path.

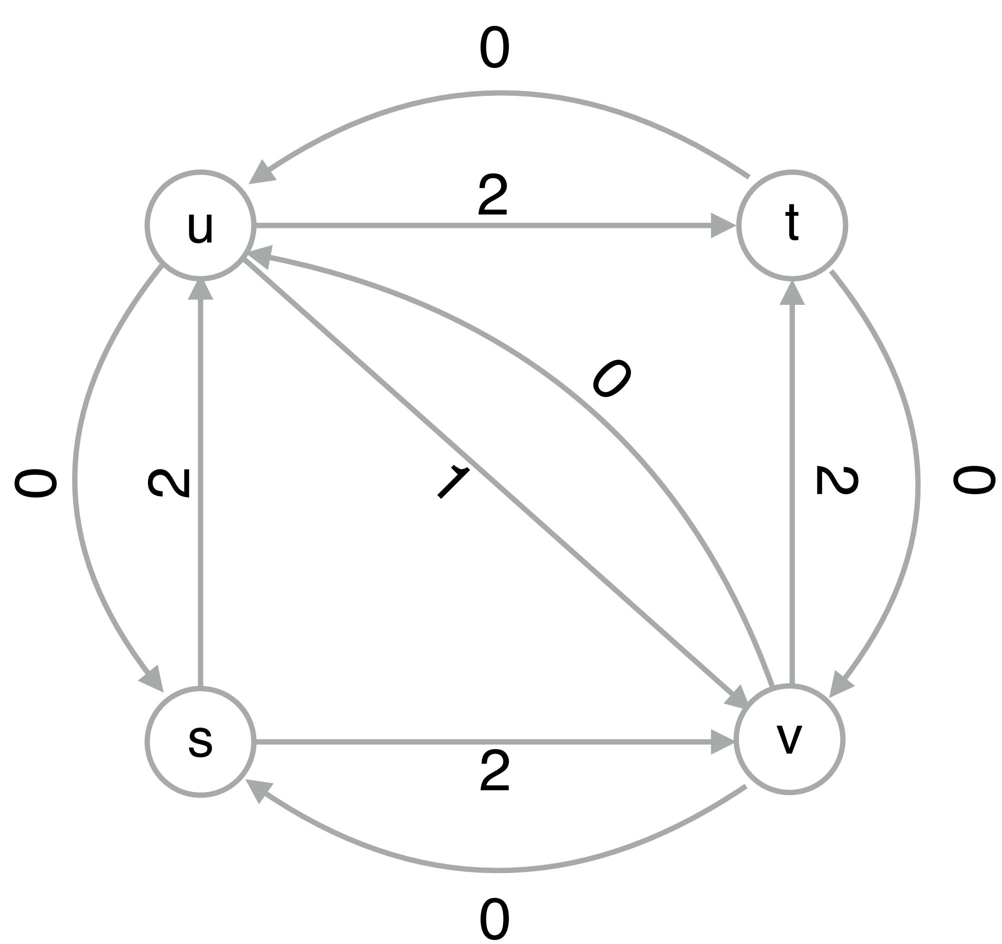
1. Start from the source.
2. While not at the sink,
   1. Pick any outgoing edge with positive flow.
      (Exists by flow conservation. Imagine flow counters at each end.)
   2. Subtract one* from that flow.
   3. Traverse that edge.
3. Prune any loops in this traversal to get the desired augmenting path.

This proof is not specific to residual graphs, but is not about maximizing flows.

Ford-Fulkerson algorithm

Algorithm:

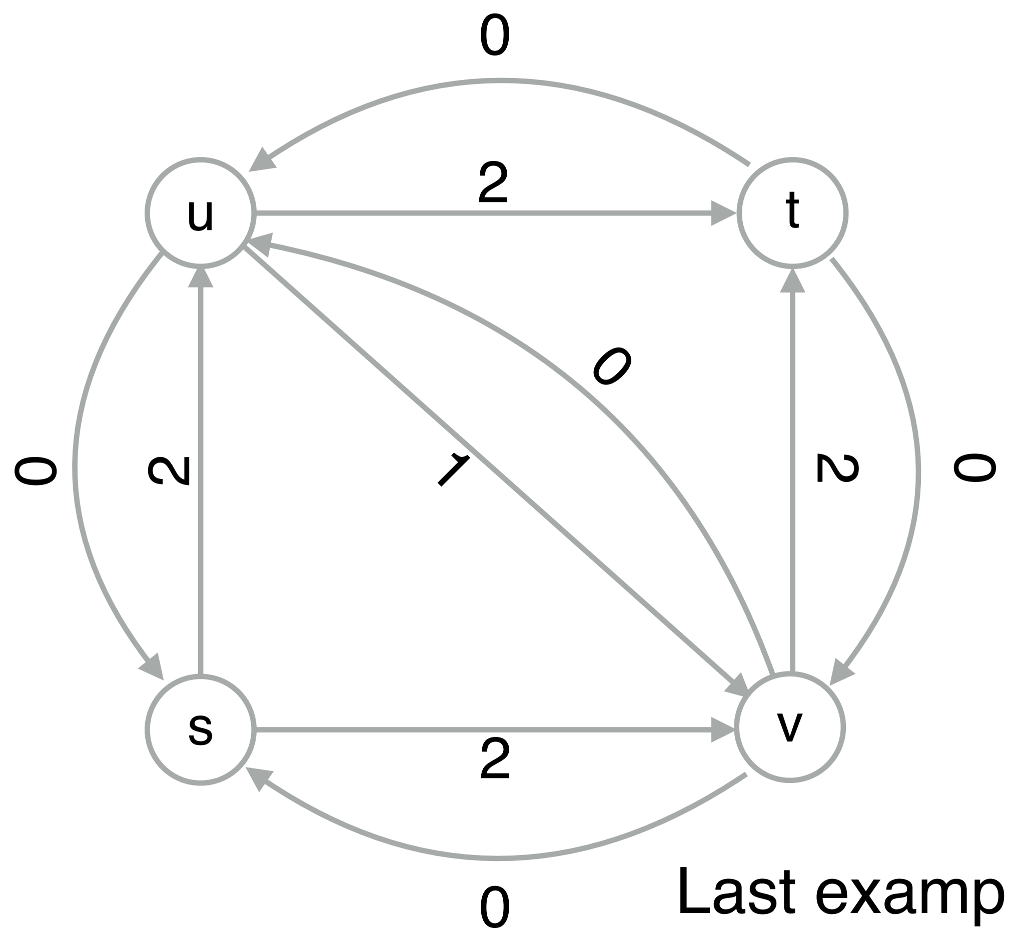
- Initialize flow f to zero.
- Initialize G_f.
- While any augmenting path exists in G_f,
  - Update f along that path taking into account reversed edges, using the minimum capacity along the augmenting path.
  - Update G_f to match f.



Ford-Fulkerson algorithm

Algorithm:

- Initialize flow f to zero.
- Initialize G_f .
- While any augmenting path exists in G_f ,
  - Update f along that path taking into account reversed edges, using the minimum capacity along the augmenting path.
  - Update G_f to match f.



Last example was actually Ford-Fulkerson on this graph.

Integral Path Theorem

If all capacities are integers, then there exists a maximum flow only using integer flow values.

Proof: Ford-Fulkerson always creates a maximum flow only using integer flow values if all the capacities are integers.

- Update f along that path taking into account reversed edges, using the minimum capacity along the augmenting path.
- All operations involved start with integers and end with integers.

This is handy when reducing logic and combinatorial problems to maximum flow, since a capacity of one can be assumed (forced) to always be zero or one.

Ford-Fulkerson Variations

n = number of vertices, m = number of edges

Basic version

- At most C iterations $\rightarrow O(Cm)$ time

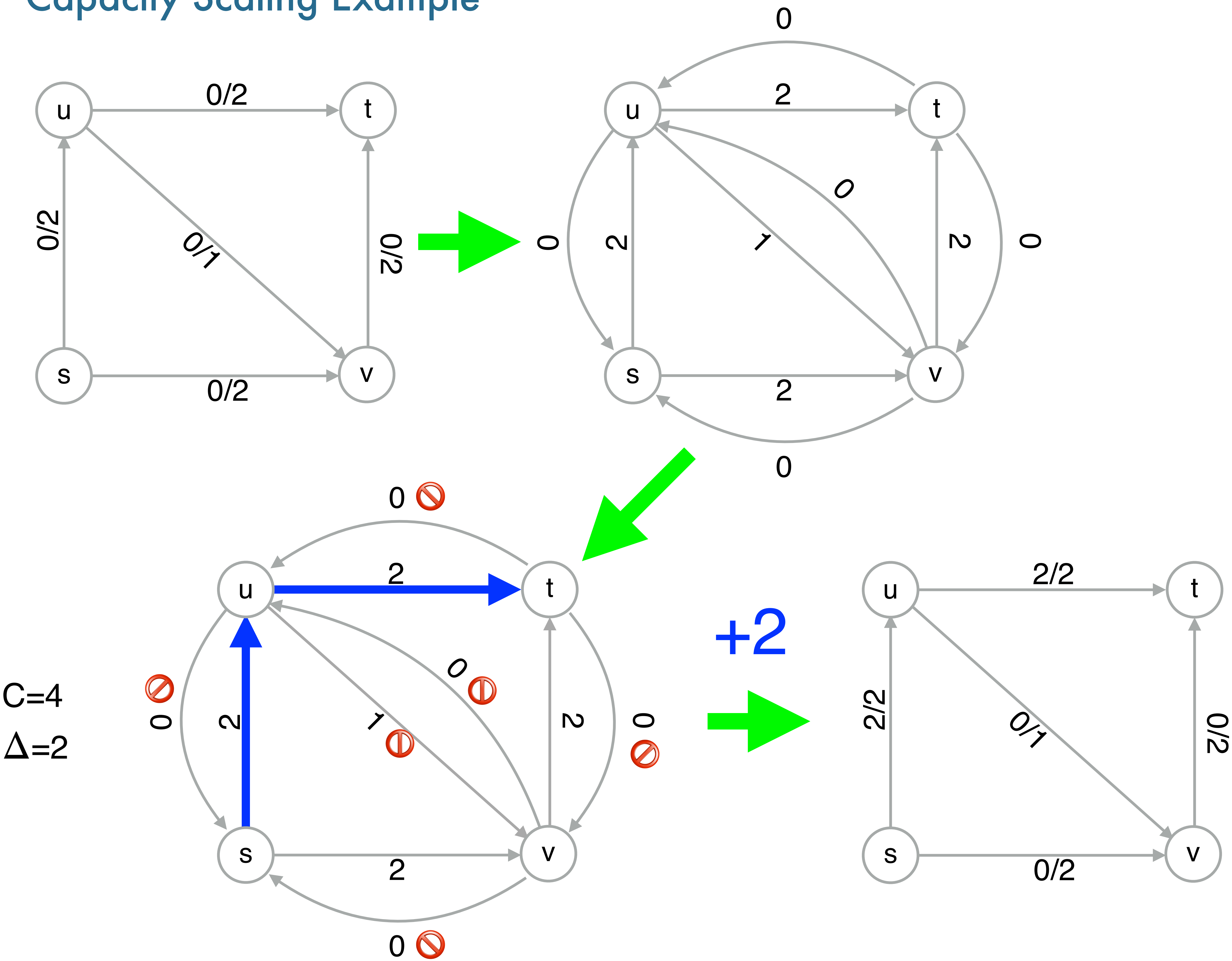
Edmunds-Karp

- Pick augmenting paths in order of increasing length.
- Use breadth first search to find shortest augmenting paths.
- At most nm iterations $\rightarrow O(nm^2)$ time

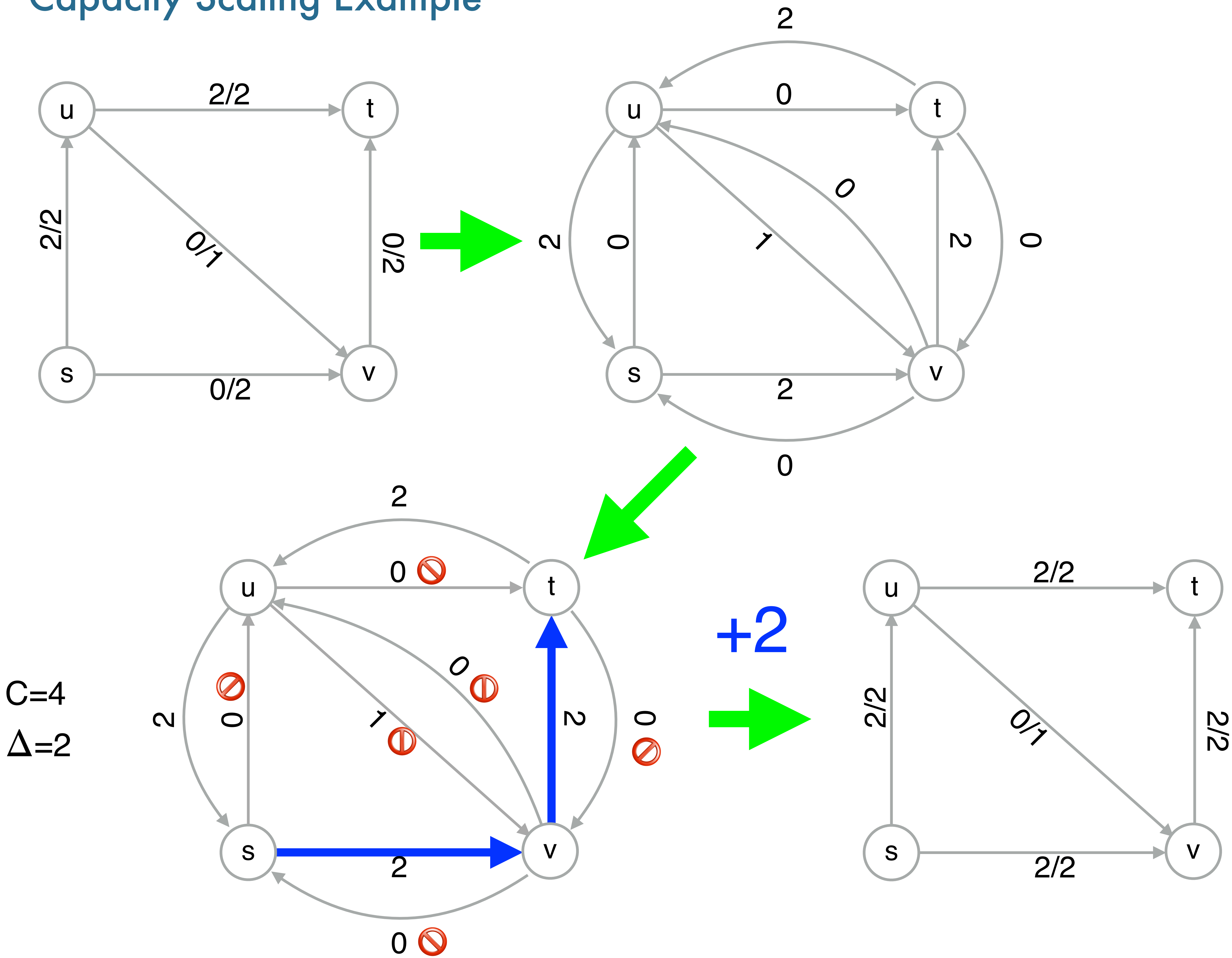
Capacity Scaling

- Force bottlenecks of augmenting paths to be big by (temporarily) filtering low capacity edges of G_f
- Filter with threshold $\Delta = \frac{C}{2}, \frac{C}{4}, \frac{C}{8}, \ldots, 1$ (last threshold running unrestricted)
- $O(\log C)$ thresholds, and most 2m augmentations per threshold $\rightarrow O(m^2 \log C)$

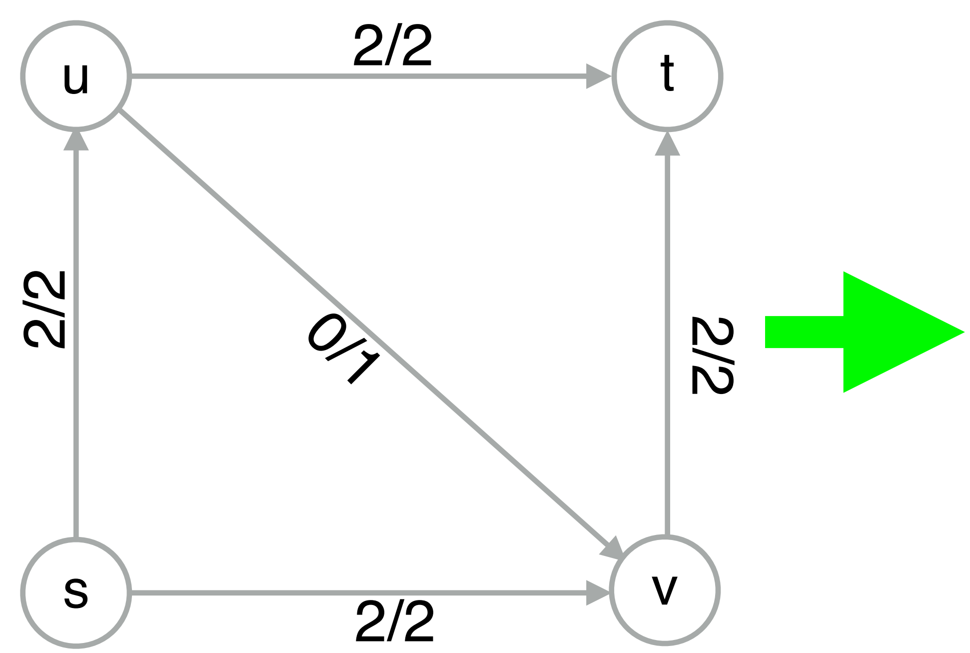
Capacity Scaling Example

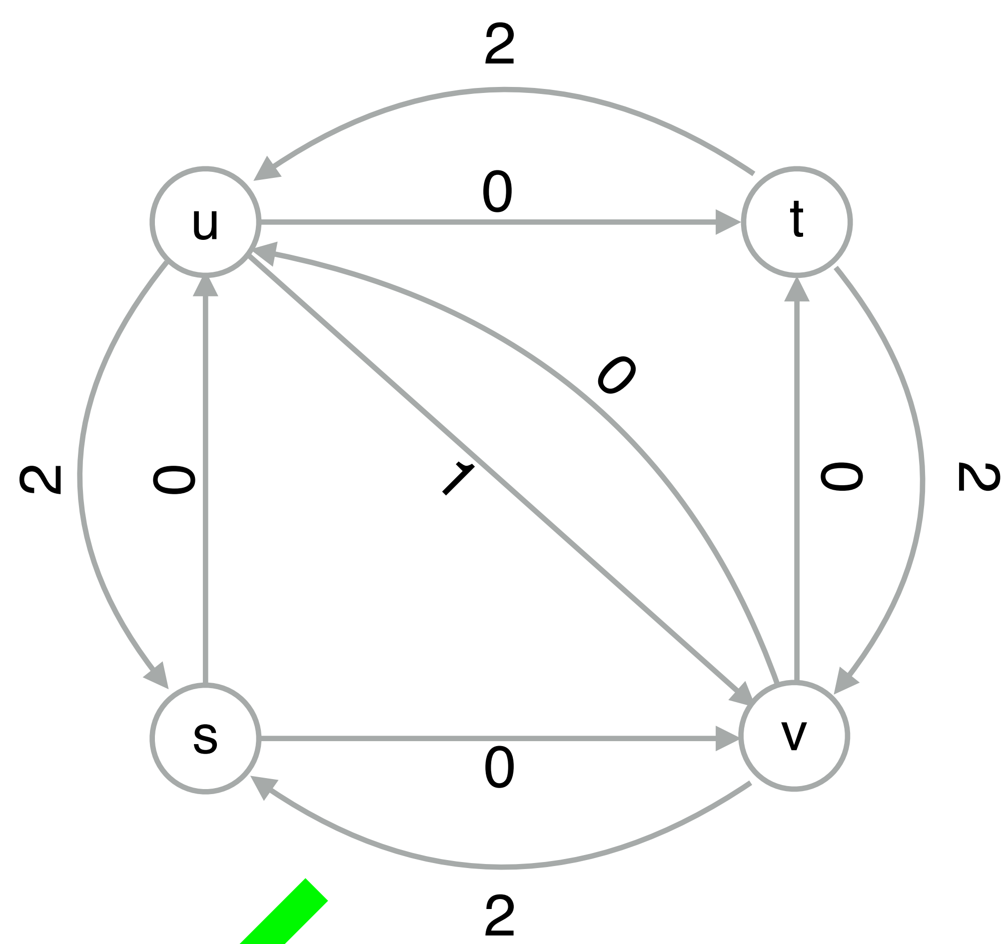


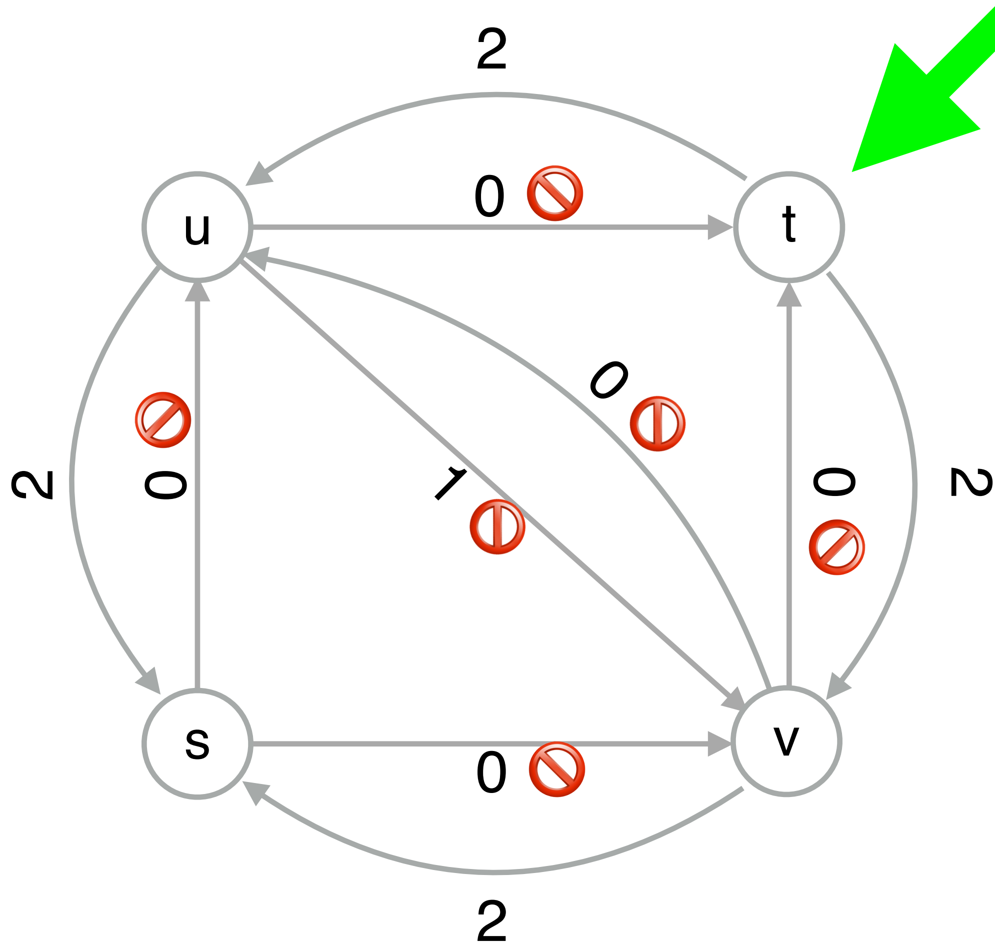
Capacity Scaling Example



Capacity Scaling Example





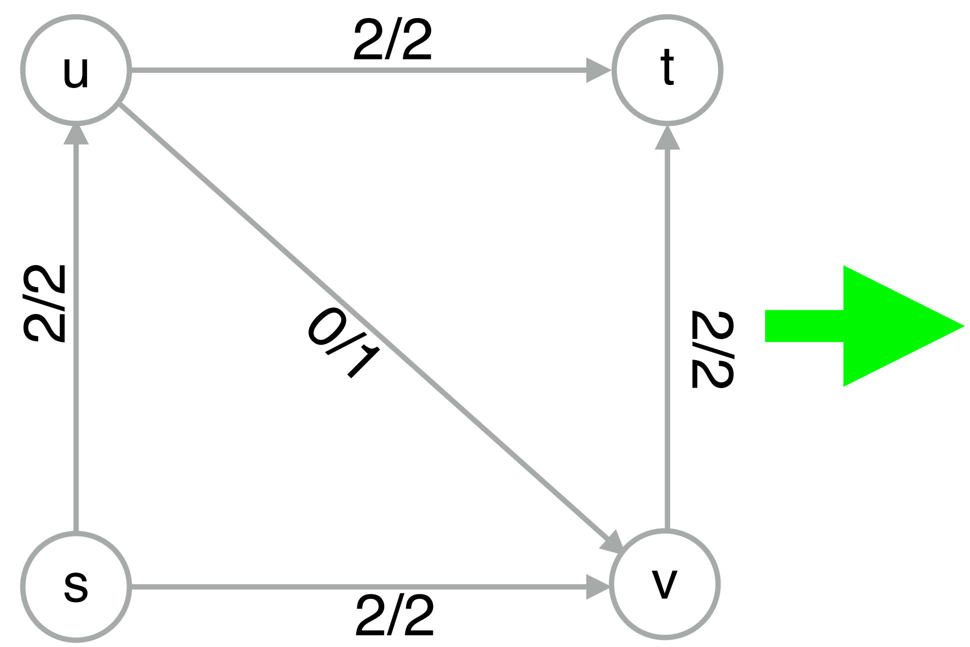


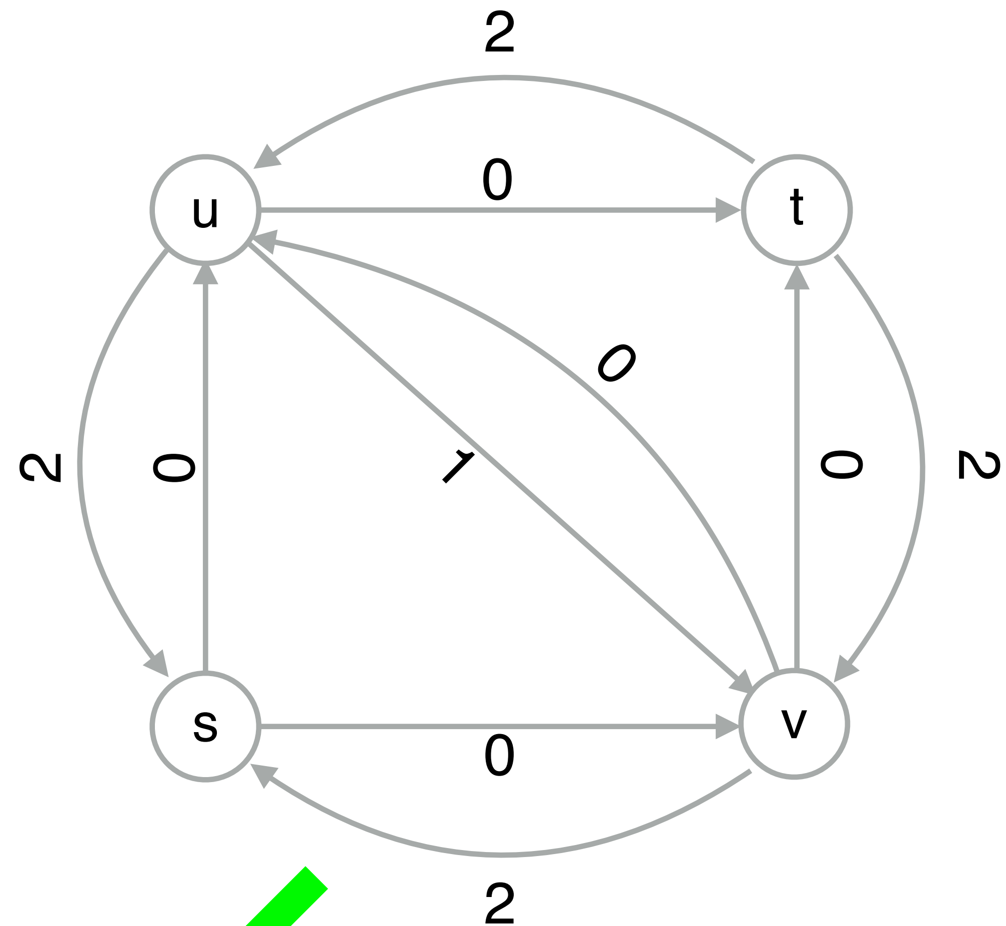
C=4

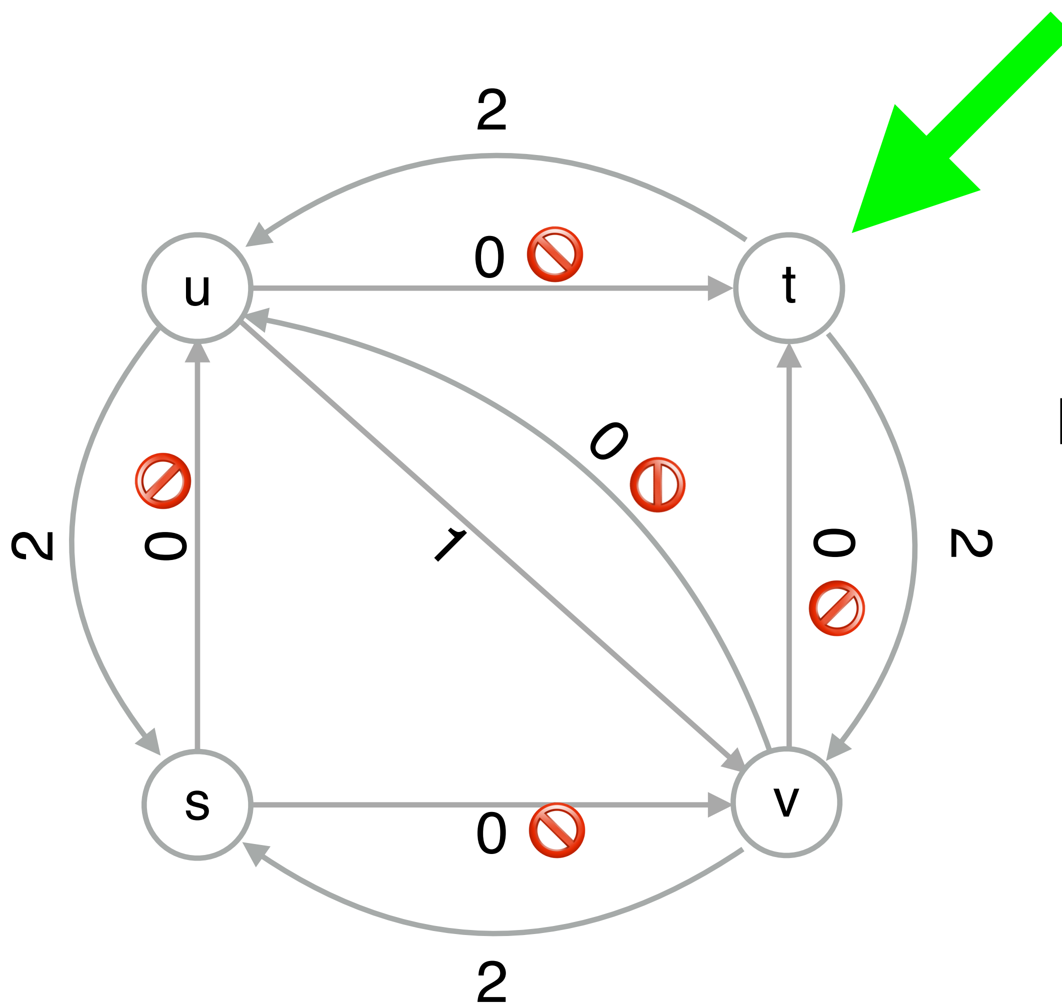
Δ=2

No more augmenting paths → Δ=1

Capacity Scaling Example







C=4

Δ=1

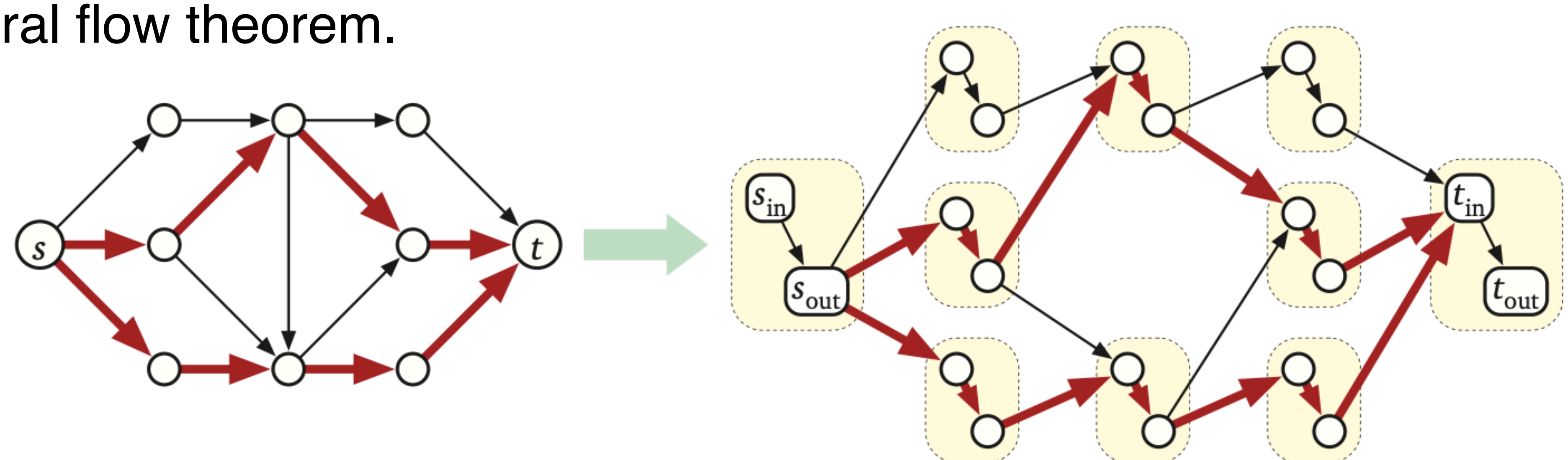
No more augmenting paths → done

Applications: Disjoint Paths

Usual application is finding/estimating redundancy.

- Disjoint ~ independent operation / independent failures

Edge disjoint paths from capacity one edges and integral flow theorem.



Vertex disjoint path by “splitting” vertices in two, and adding a capacity one edge.

Can simulate vertex capacity by changing that edge’s capacity.

Image source: https://jeffe.cs.illinois.edu/teaching/algorithms/book/11-maxflowapps.pdf

Exam Scheduling Problem

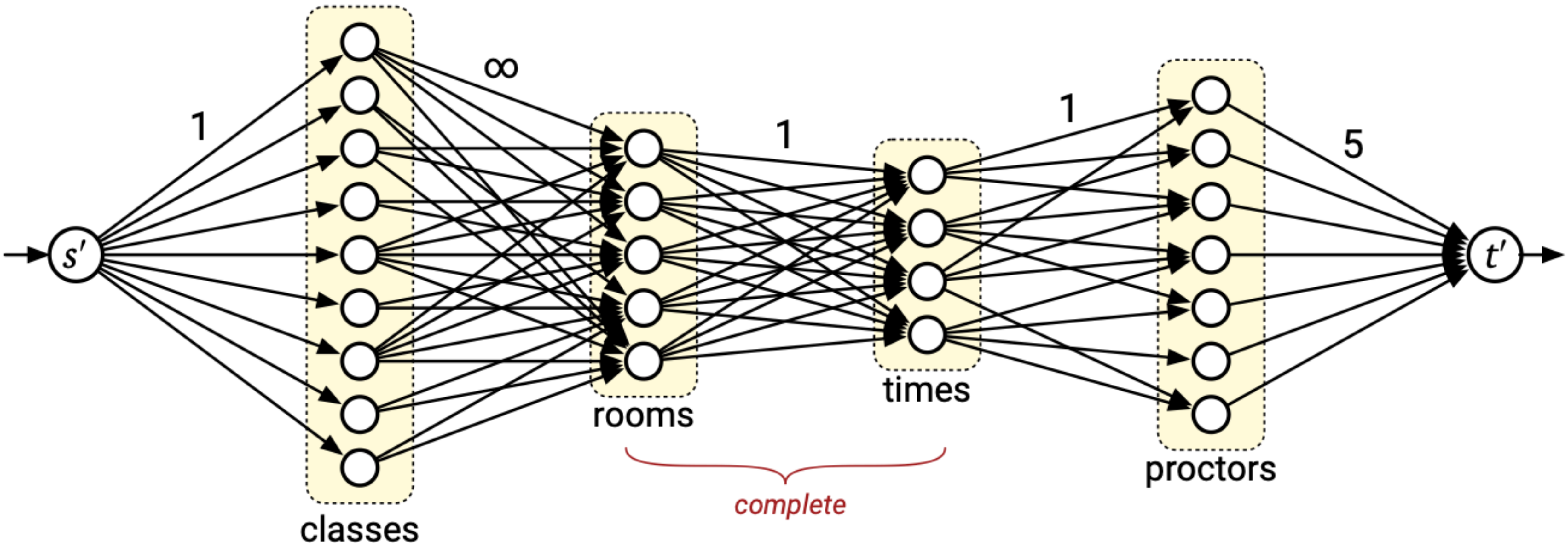


Figure 11.5. A flow network for the exam scheduling problem.

Image source: https://jeffe.cs.illinois.edu/teaching/algorithms/book/11-maxflowapps.pdf

Baseball Elimination

If my favorite team wins all their remaining games this season, could they win?

- Can all the other teams lose enough to not win, simultaneously?
- Challenge is splitting losses between teams that are close…

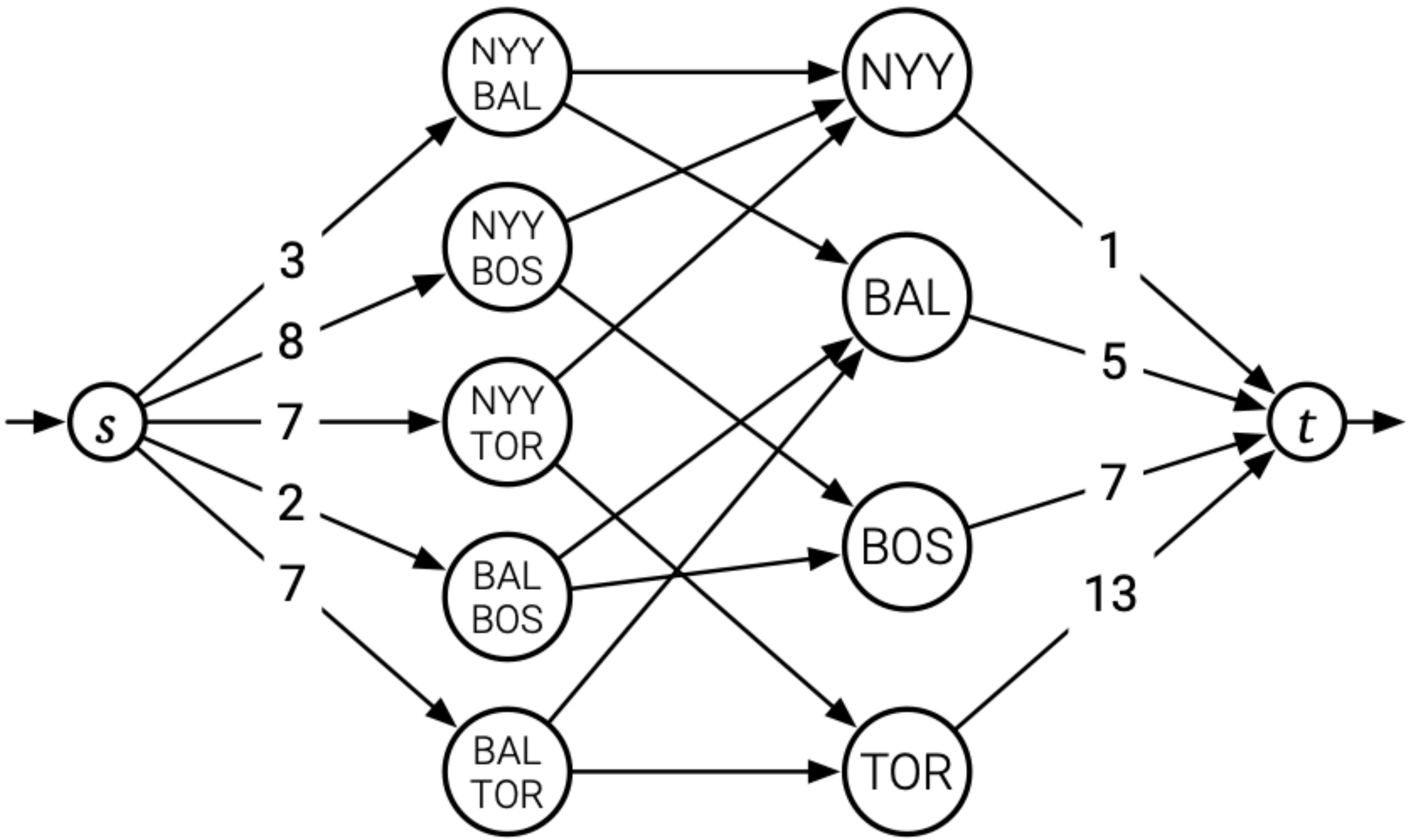


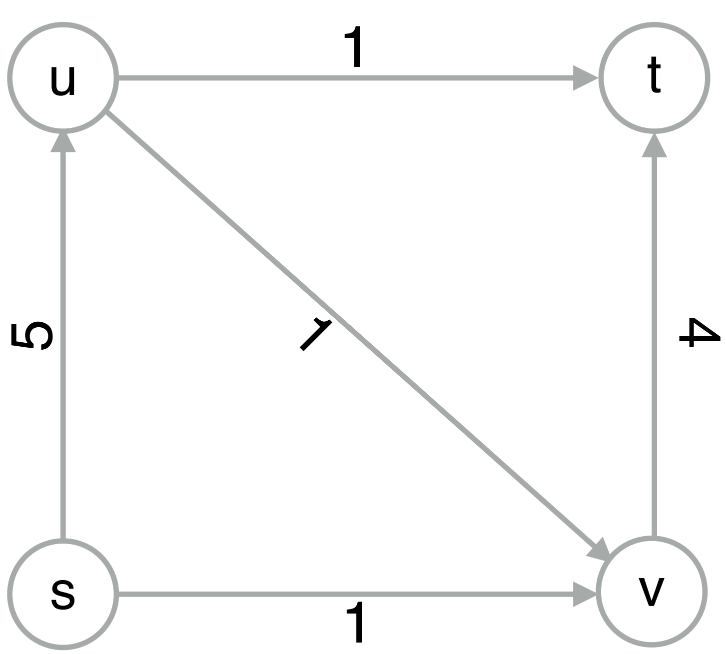
Image source: https://jeffe.cs.illinois.edu/teaching/algorithms/book/11-maxflowapps.pdf

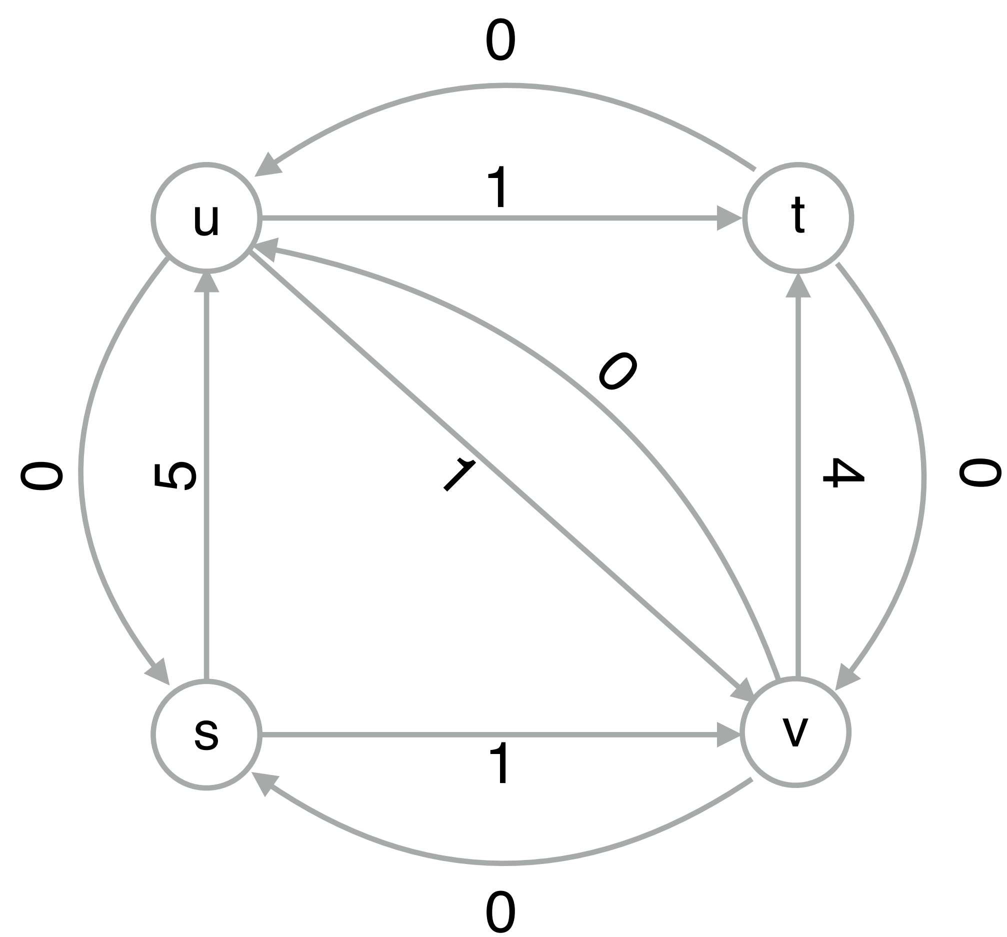
Checking a Max Flow from Ford-Fulkerson

Max Flow Min Cut theorem

Bottleneck cuts

- What is the maximum flow in the graph on the left?
- Find the max flow by running Ford-Fulkerson on the residual graph.
- Can you see some “proof” in either graph why the value of the max flow cannot be more?





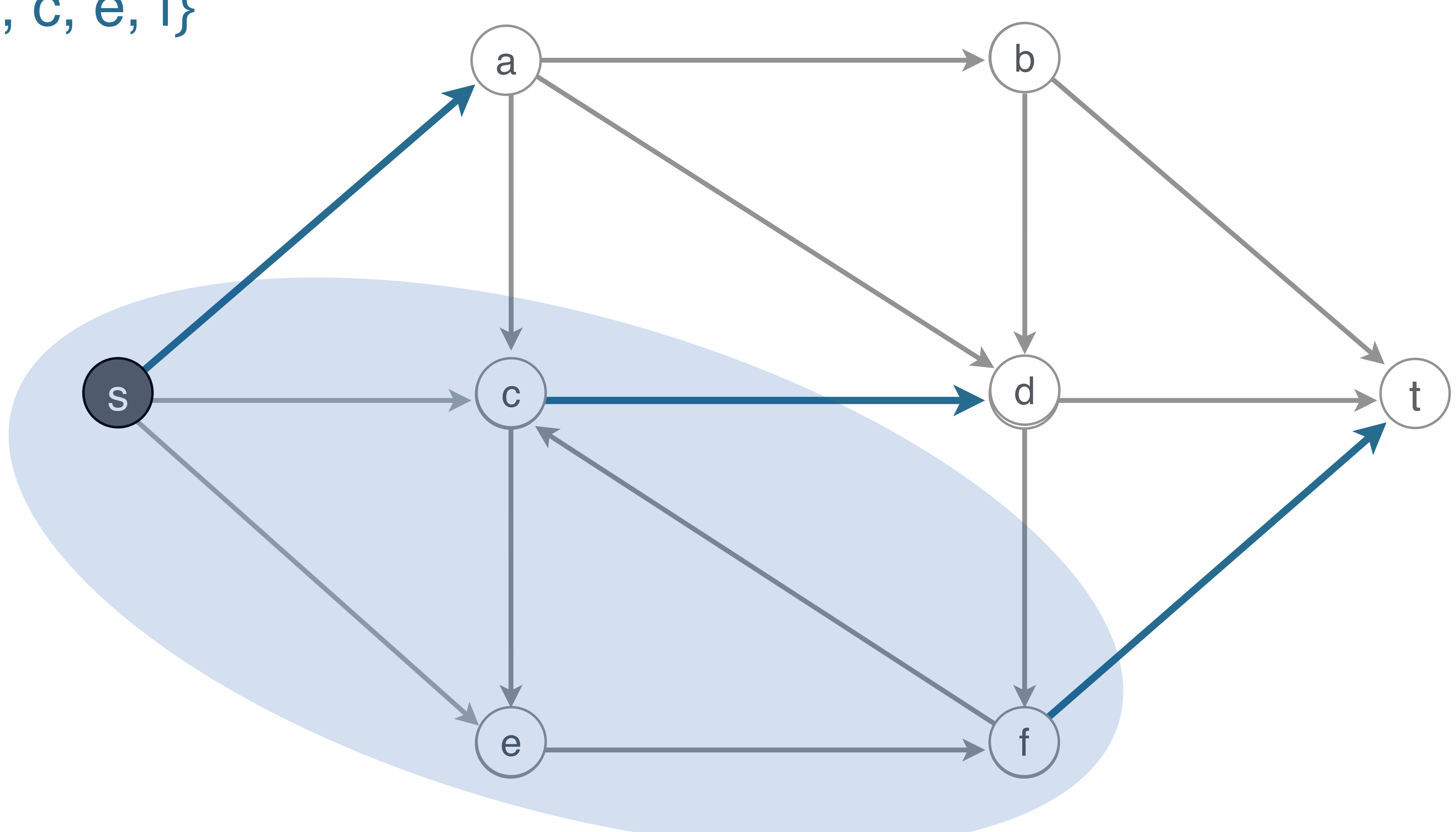
Minimum-cut problem

Def. An st-cut (cut) is a partition (A, B) of the vertices with $s \in A$ and $t \in B$.

cut-set: directed edges from nodes in A to B {(s,a), (c,d), (f,t)}

- Note, that this only contains edges *directed from A to B*

cut: A = {s, c, e, f}

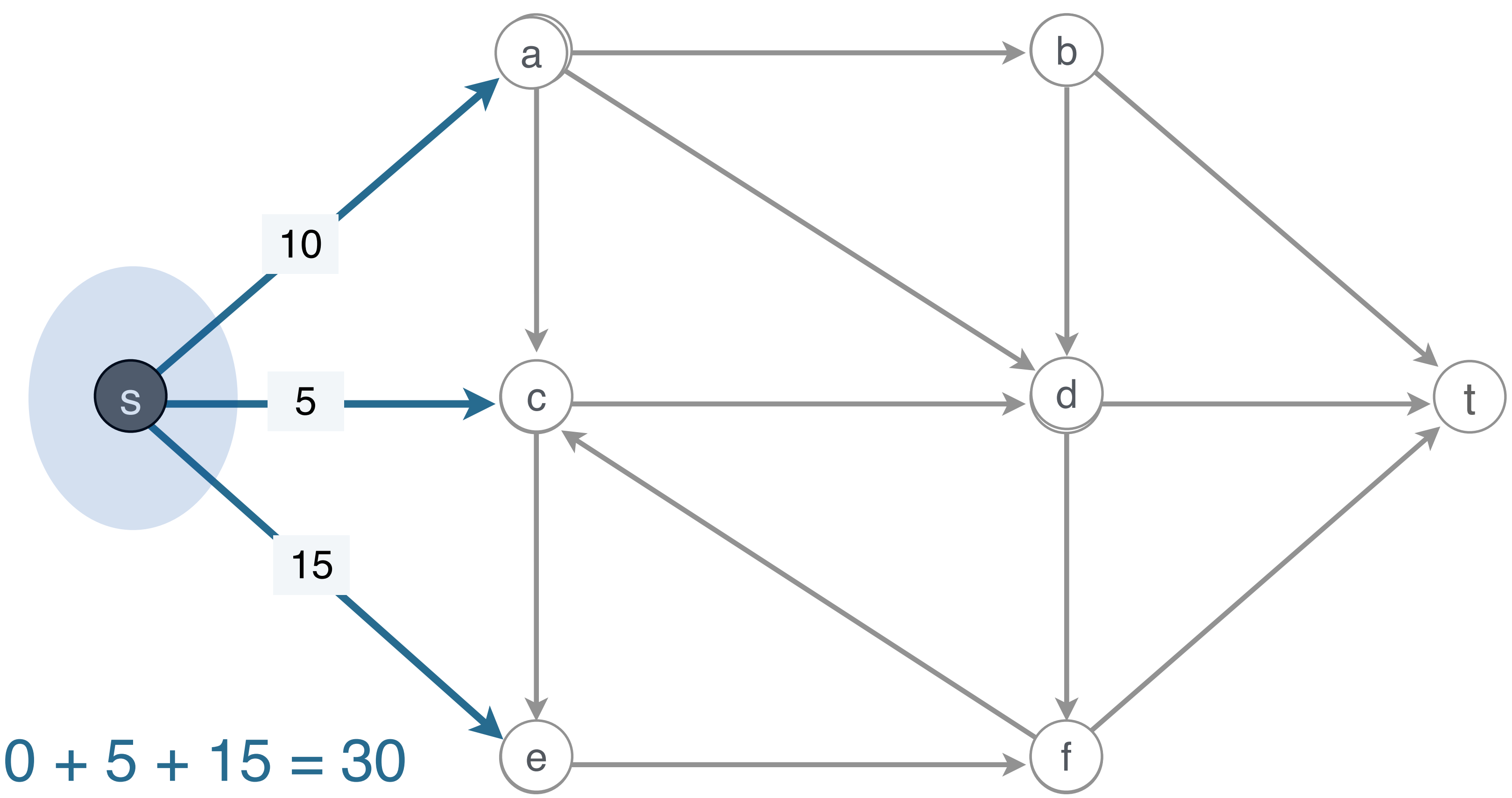


Minimum-cut problem

Def. An st-cut (cut) is a partition (A, B) of the vertices with $s \in A$ and $t \in B$.

Def. Its capacity is the sum of the capacities of the edges from A to B. (i.e. the capacity of edges in the cut-set)

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$



A = {s}

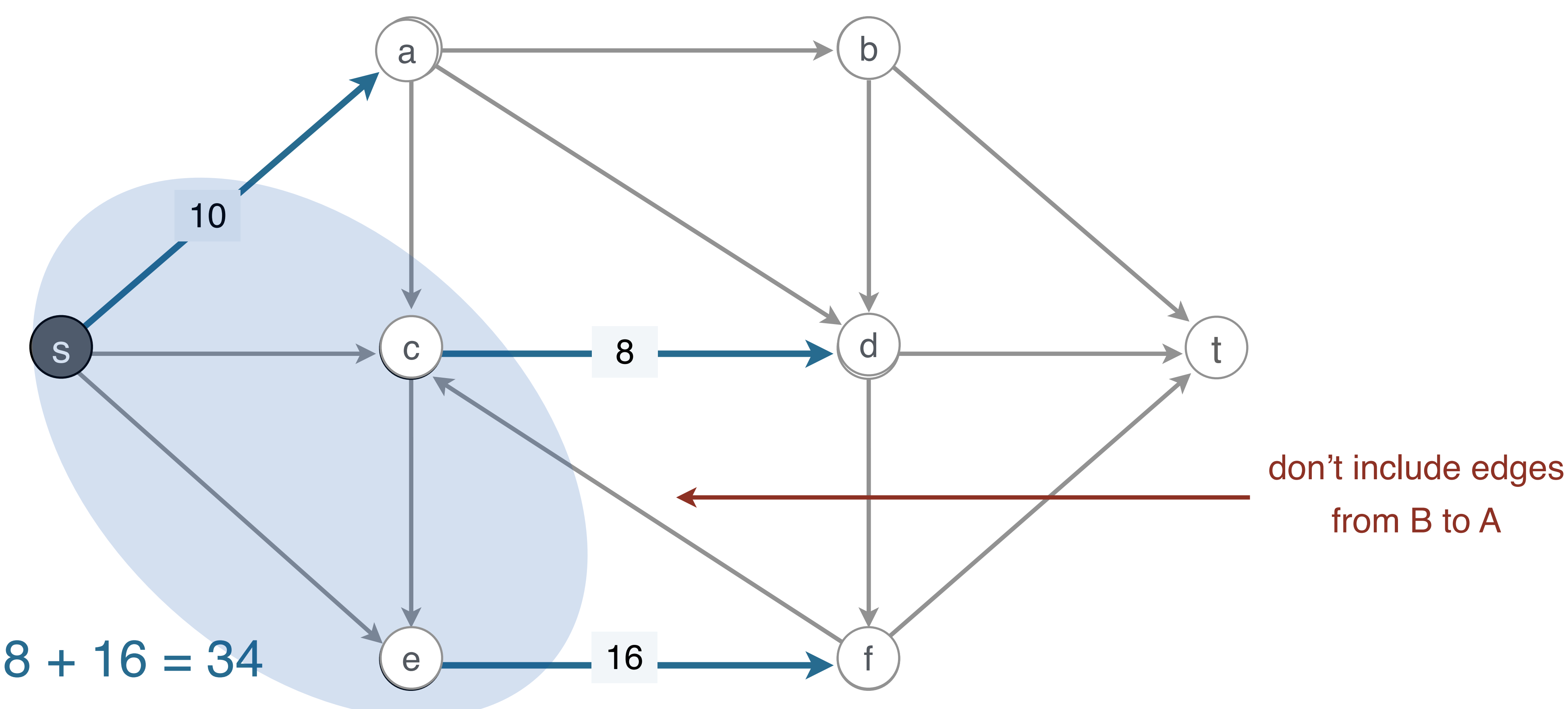
capacity = 10 + 5 + 15 = 30

Minimum-cut problem

Def. An st-cut (cut) is a partition (A, B) of the vertices with $s \in A$ and $t \in B$.

Def. Its capacity is the sum of the capacities of the edges from A to B.

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$



A = {s, c, e}

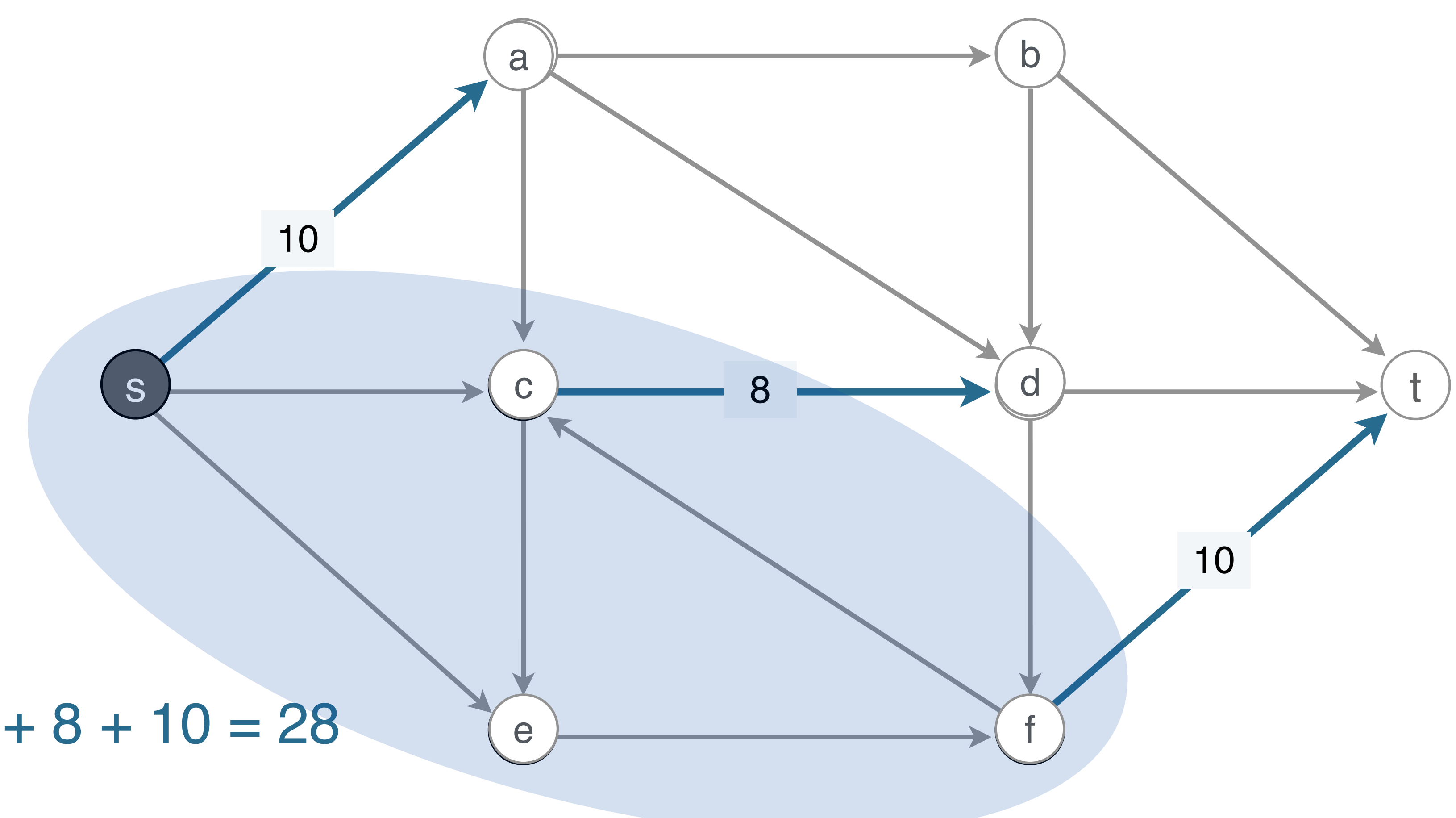
capacity = 10 + 8 + 16 = 34

Minimum-cut problem

Def. An st-cut (cut) is a partition (A, B) of the vertices with $s \in A$ and $t \in B$.

Def. Its capacity is the sum of the capacities of the edges from A to B.

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$



A = {s, c, e, f}

capacity = 10 + 8 + 10 = 28

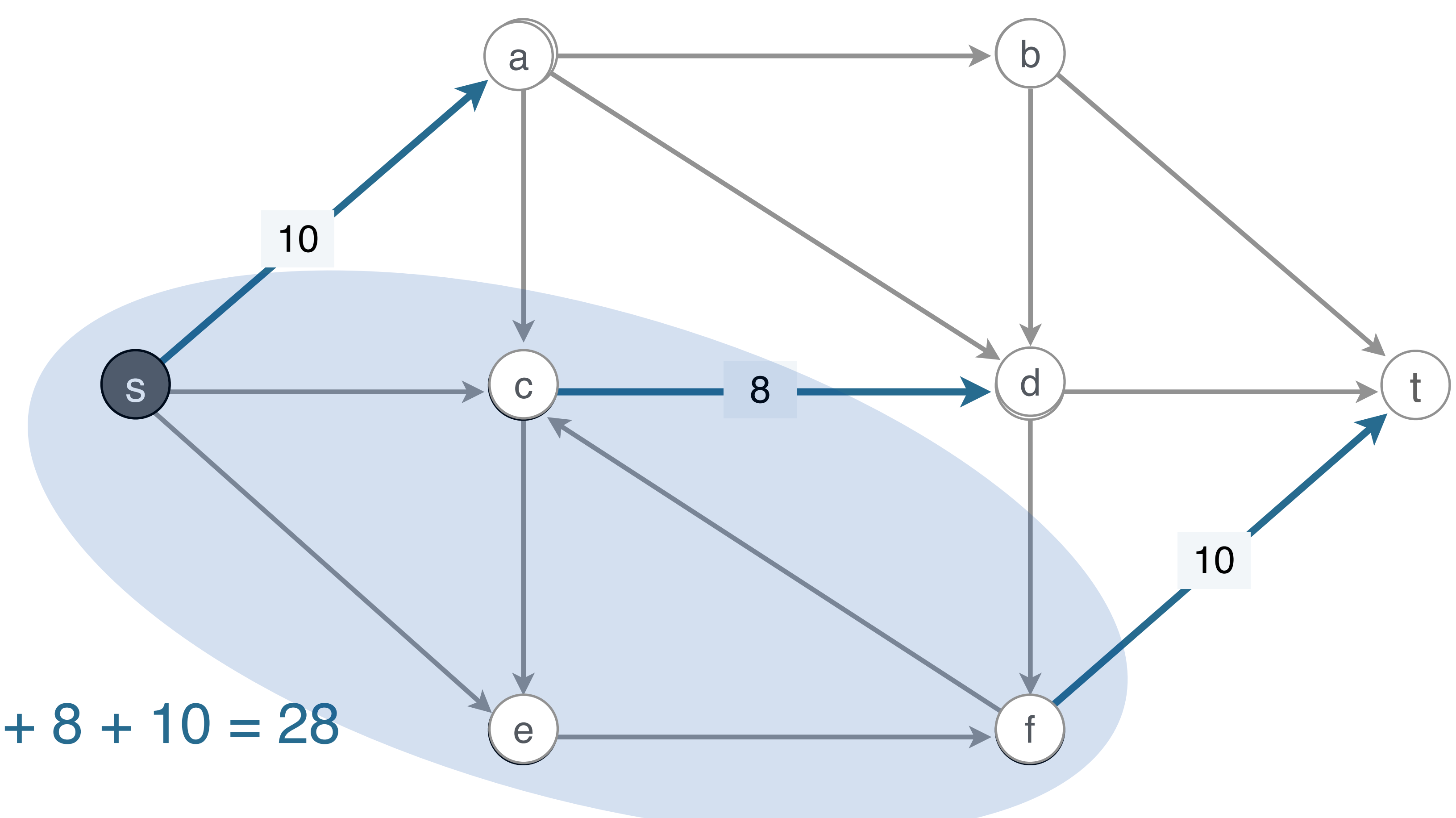
Minimum-cut problem

Def. Its capacity is the sum of the capacities of the edges from A to B.

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$

cap(A,B) = 28

Is this an upper bound on the maximum st-flow?



A = {s, c, e, f}

capacity = 10 + 8 + 10 = 28

Minimum-cut problem

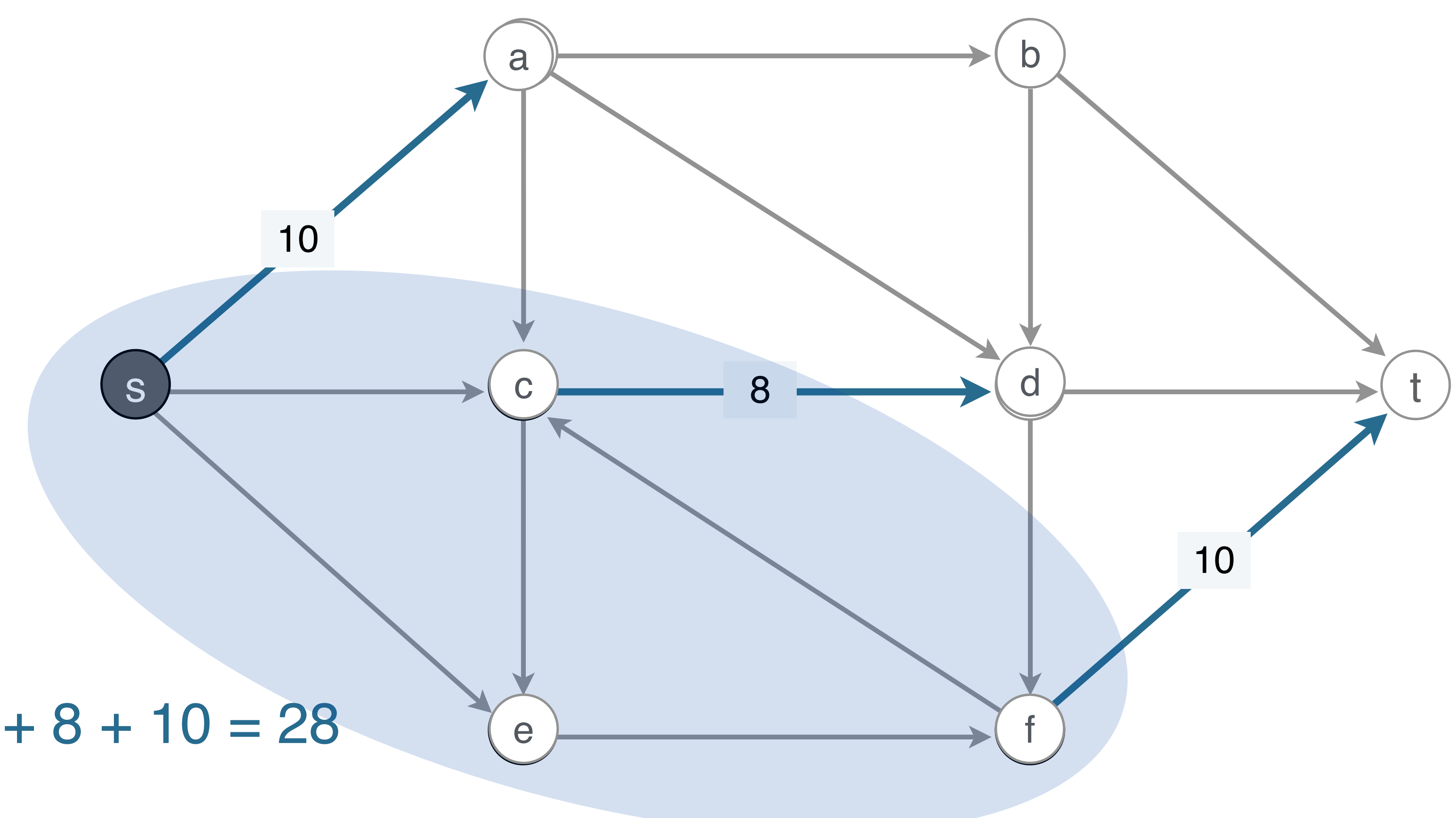
Def. Its capacity is the sum of the capacities of the edges from A to B.

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$

cap(A,B) = 28

Is this an upper bound on the maximum st-flow?

- think of the capacity of a cut as the “throughput” or “bottleneck” of the edges carrying flow from A to B.
- since s is in A and t in B, this is also an upper bound on the over flow value



A = {s, c, e, f}

capacity = 10 + 8 + 10 = 28

Minimum-cut problem

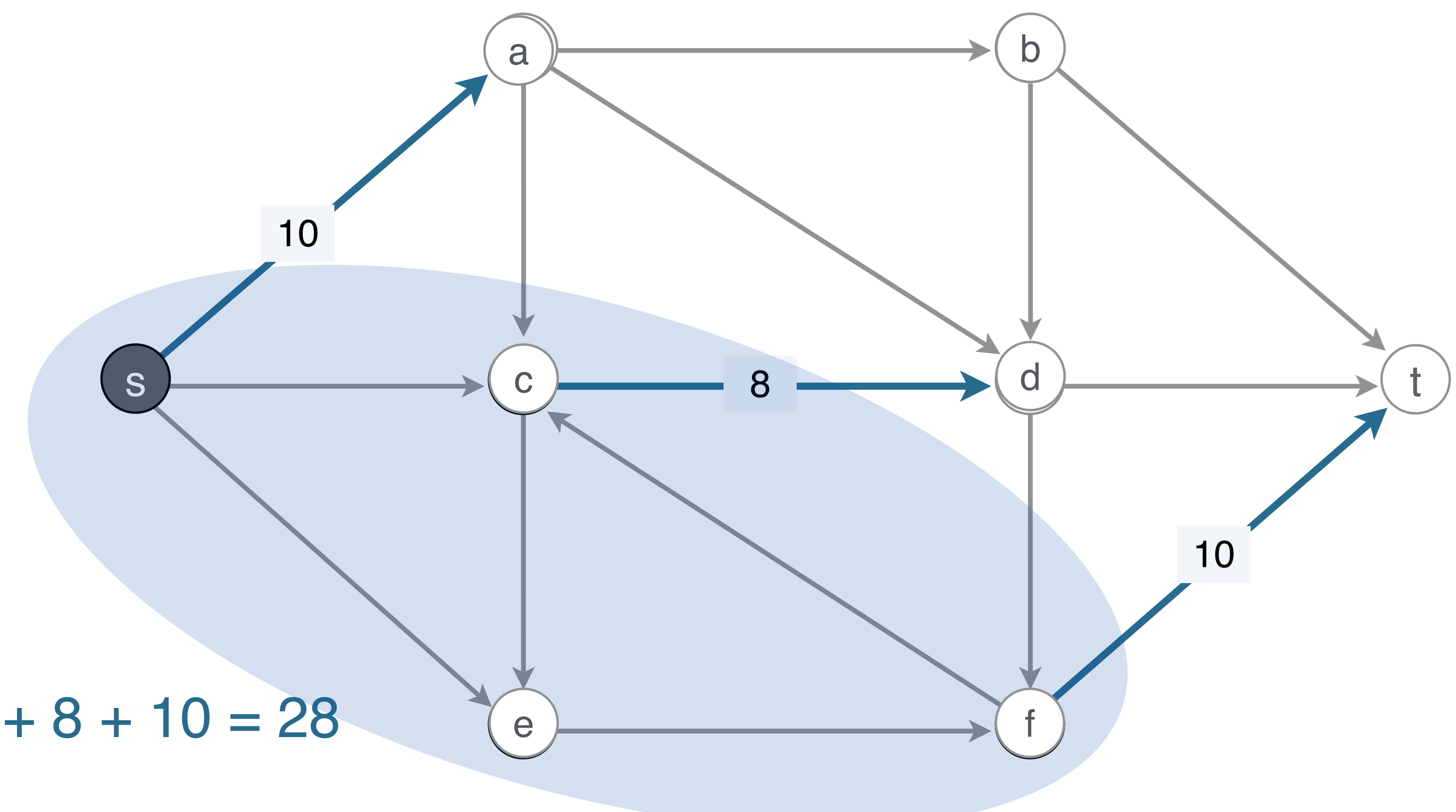
Def. Its capacity is the sum of the capacities of the edges from A to B.

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$

cap(A,B) = 28

Min-cut: the st-cut with the lowest capacity in a graph

Min-cut problem: find the minimum capacity st-cut.

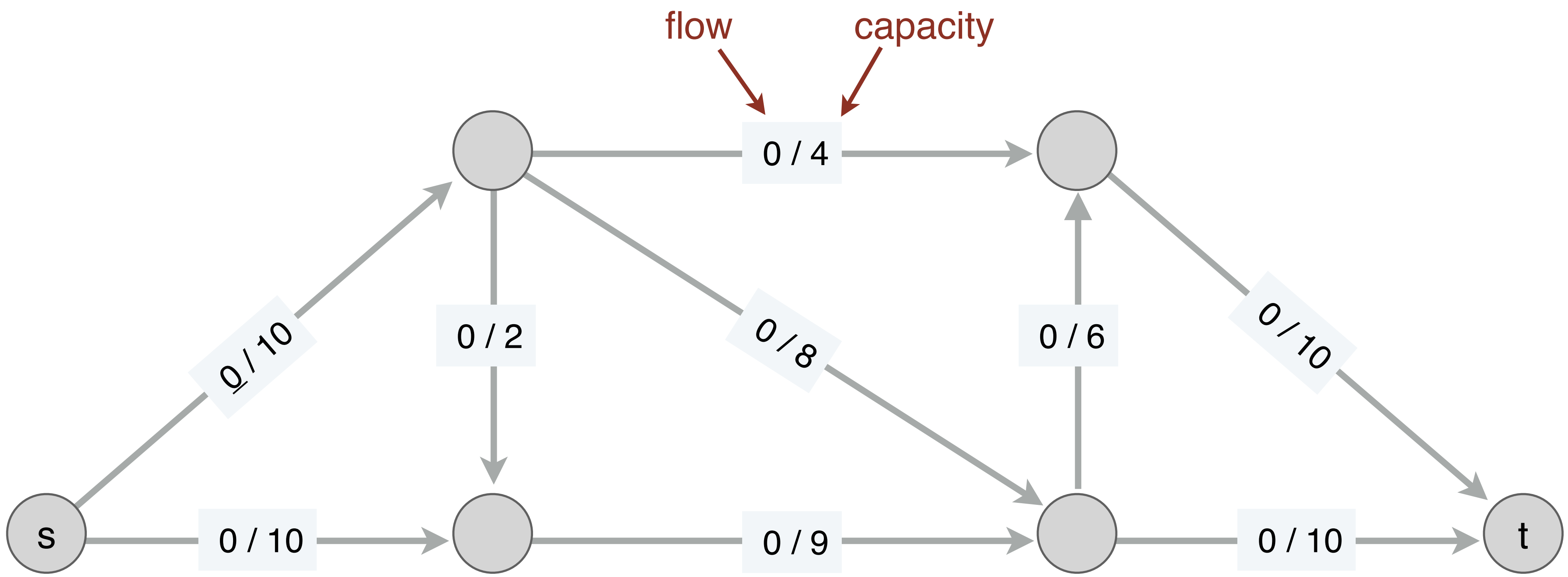


A = {s, c, e, f}

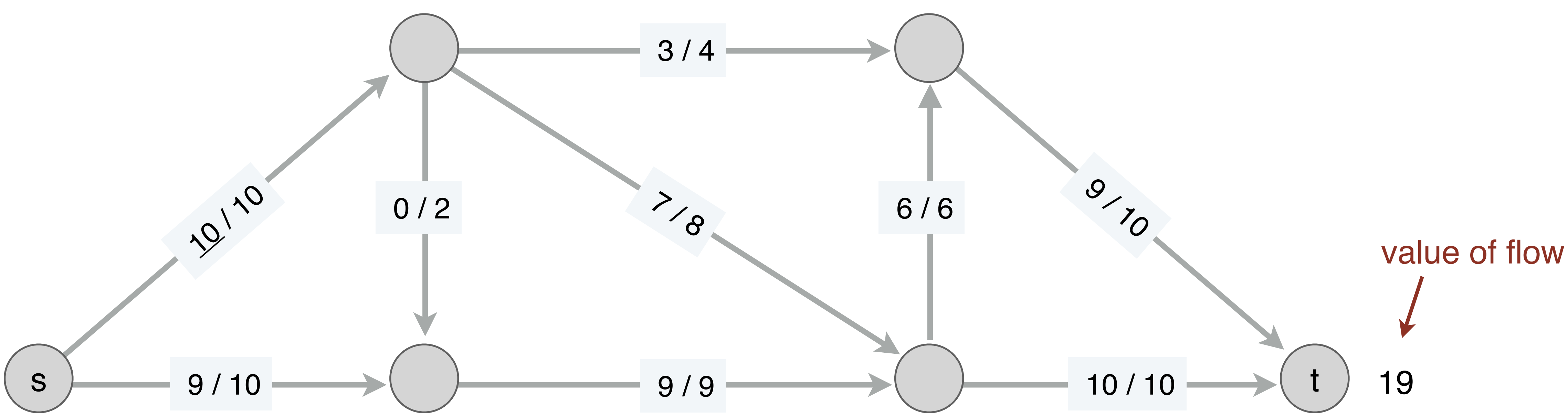
capacity = 10 + 8 + 10 = 28

Certificate for the max flow

- Can you find some proof/certificate for the value of the max flow in this graph?

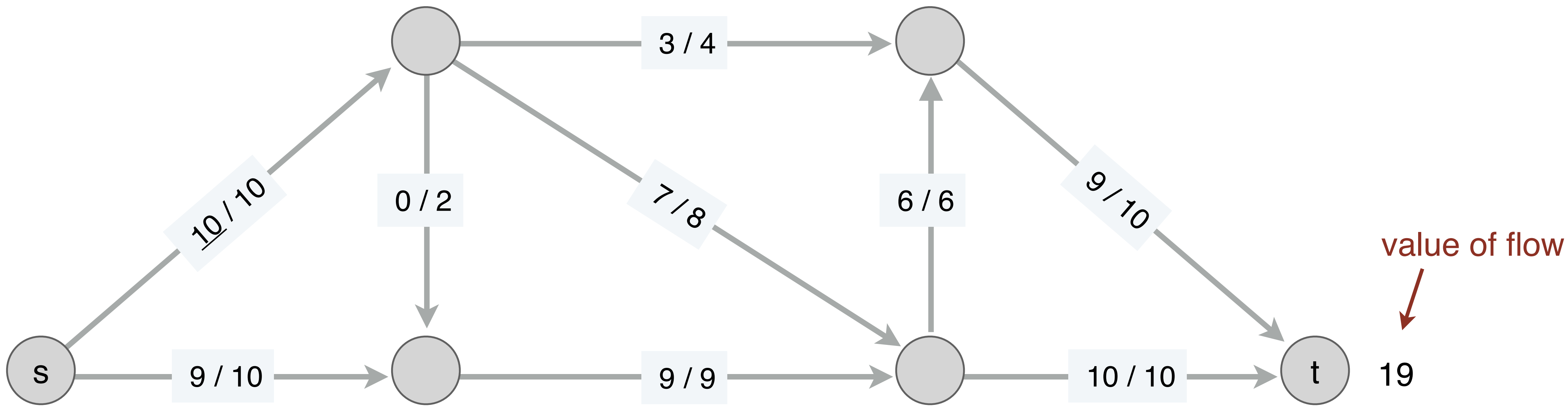


max flow:

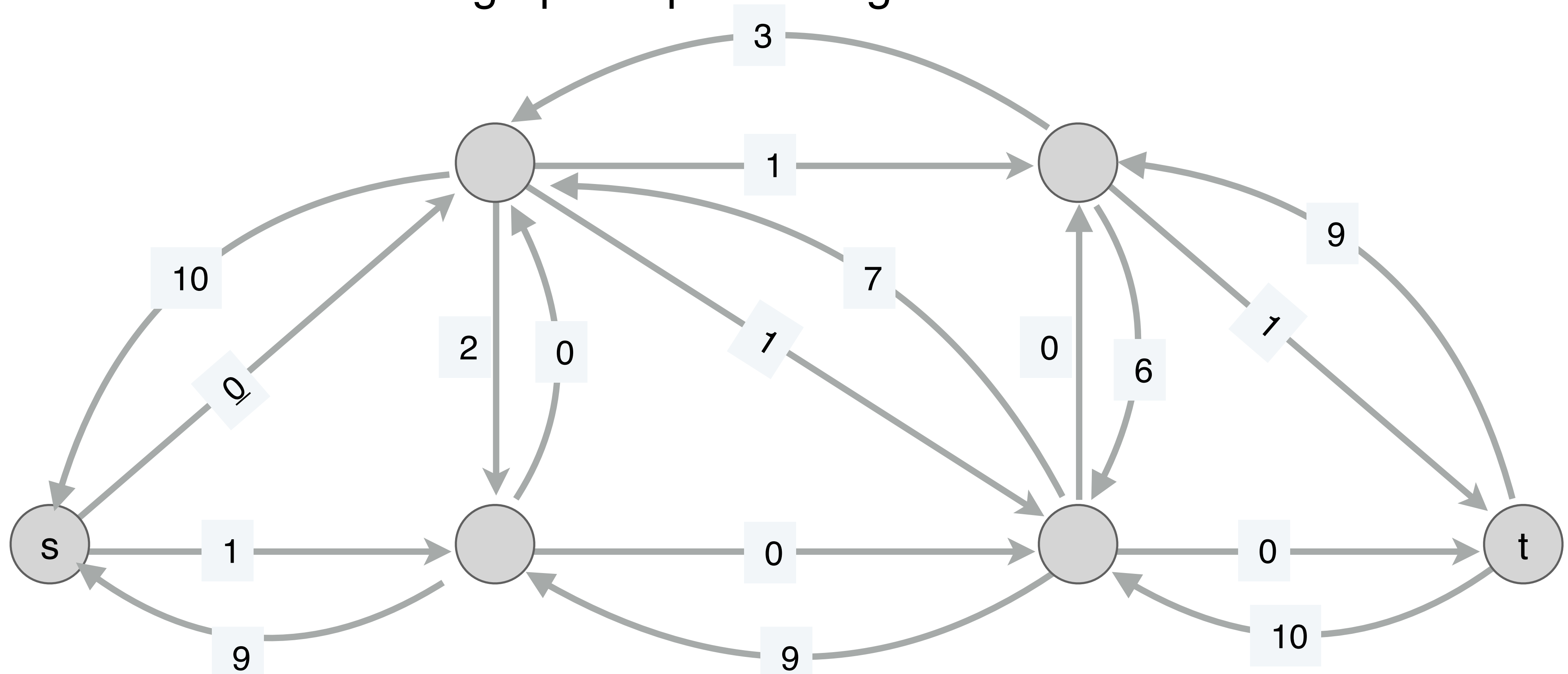


Certificate for the max flow

- Can you find some proof/certificate for the value of the max flow in this graph?



- How does the residual graph help in finding it?



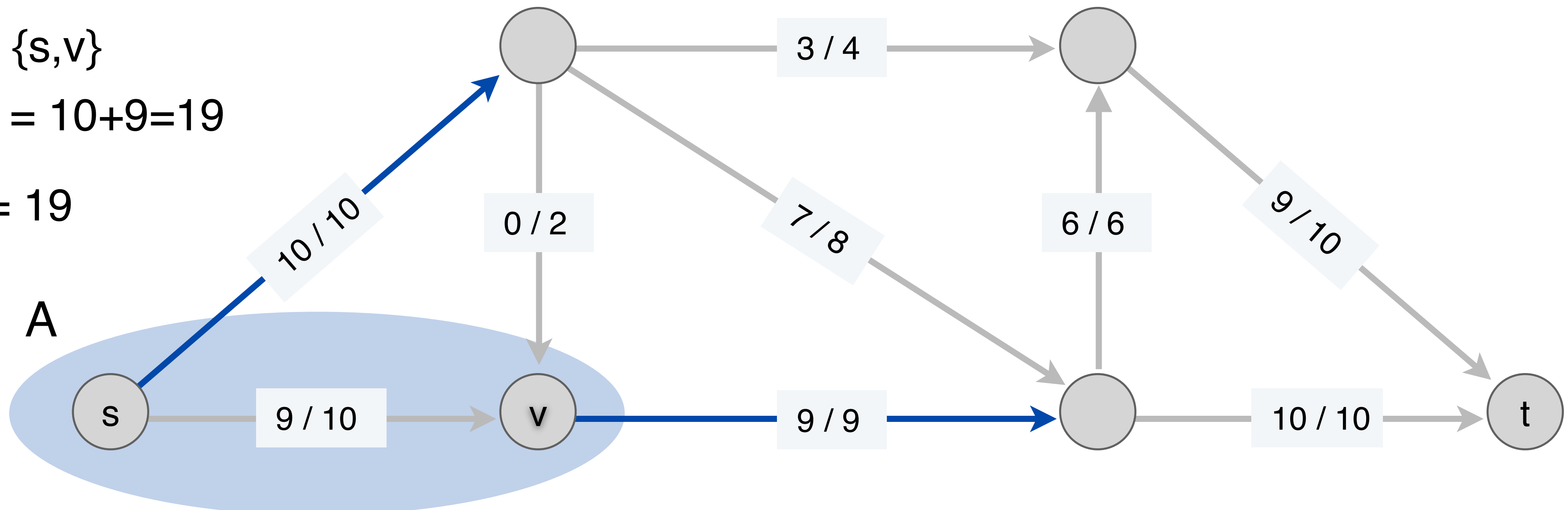
Certificate for the max flow

network G and flow f

cut A = {s,v}

cap(A) = 10+9=19

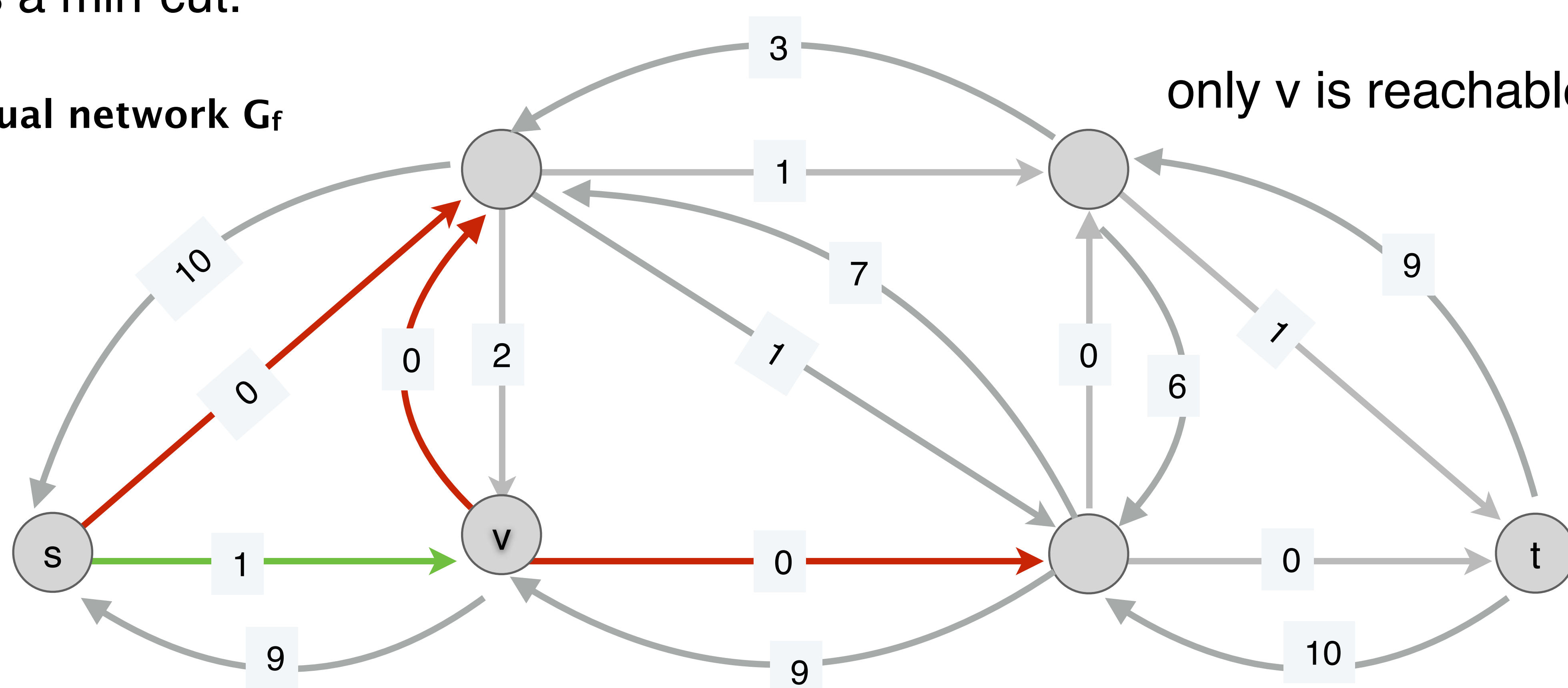
val(f) = 19



claim: A is a min-cut.

residual network G_f

only v is reachable from s in G_f



Max Flow Min Cut

Max Flow Min Cut (MFMC) theorem:

Given a directed graph *G(V,E)* with source *s*, sink *t* and non-negative capacities *c(e)*, the value of the maximum flow in *G* is equal to the capacity of the minimum st-cut.

$$\max_{f\ flow} val(f) = \min_{A \subseteq V,\ s \in A} cap(A,B)$$

Certificate of optimality: We can use the MFMC theorem to prove that a flow f is maximum;

val(f) is maximum if there is a cut with its capacity equal to *val(f)*.

Finding the min-cut

1. find the maximum flow in G , i.e. run Ford-Fulkerson
2. find the set A of all nodes that are still reachable from the source
   - run BFS from s in G_f to find A
   - A has at least one element, s

Nodes in A form the minimum capacity cut.

Properties of min-cuts

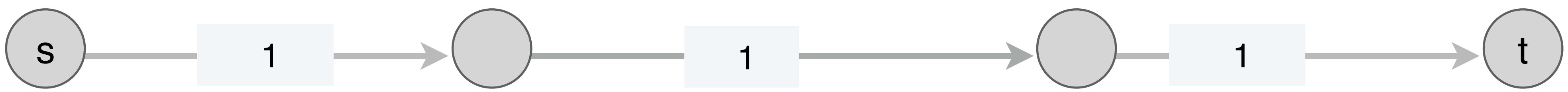
1. is the min-cut always unique?

2. what happens to the max flow if we decrease the capacity of an edge in the min-cut by 1?

Properties of min-cuts

1. is the min-cut always unique?

No - this graph has 3 min cuts.

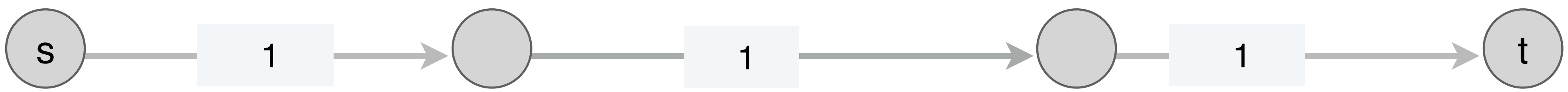


2. what happens to the max flow if we decrease the capacity of an edge in the min-cut by 1?

Properties of min-cuts

1. is the min-cut always unique?

No - this graph has 3 min cuts.



2. what happens to the max flow if we decrease the capacity of an edge in the min-cut by 1?

The max flow must decrease by one.

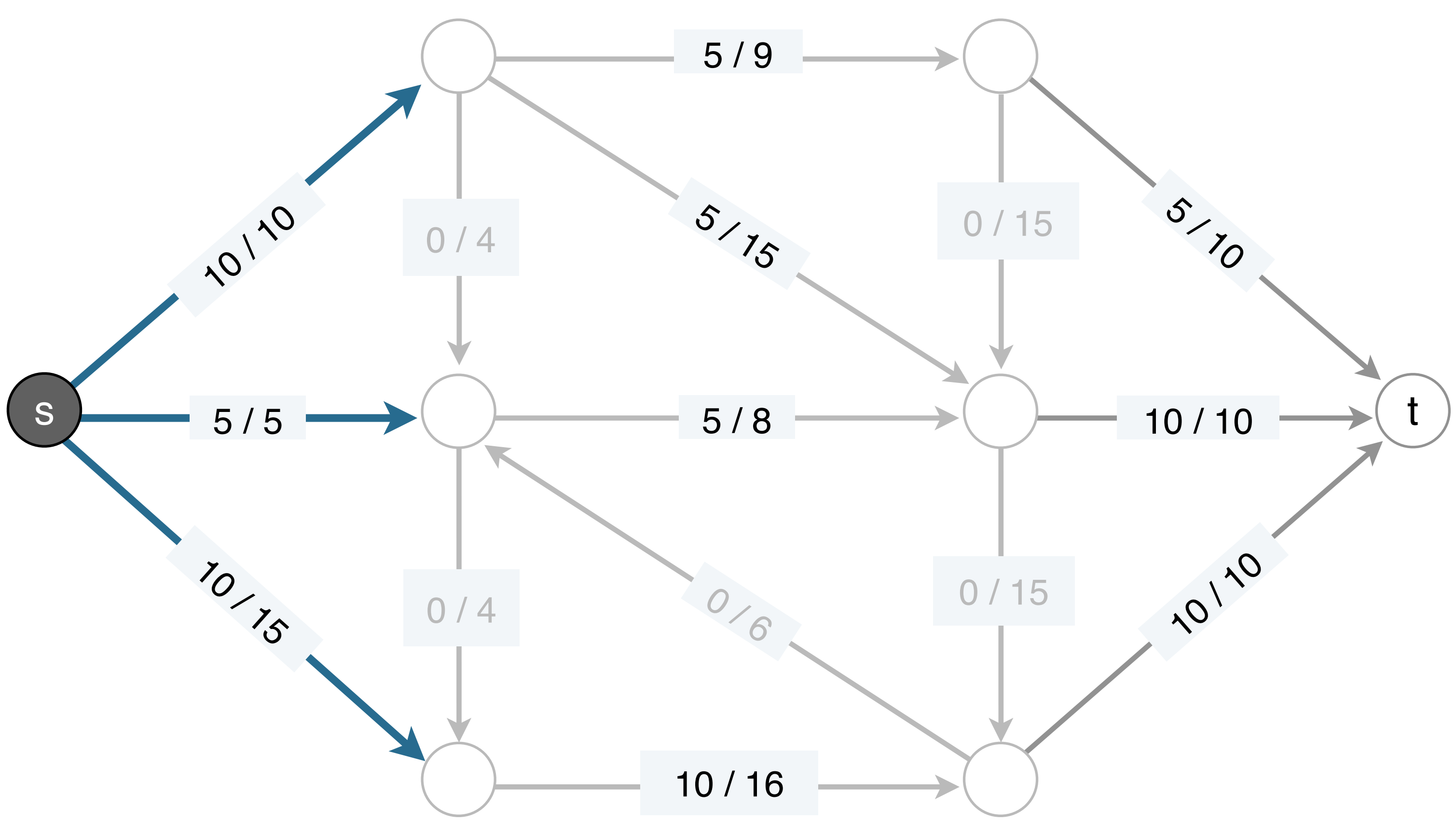
Relationship between flows and cuts – MFMC proof

Flow value lemma. Let f be *any* flow and let (A, B) be any cut. Then, the value of the flow f equals the net flow across the cut (A, B).

$$val(f) = \sum_{e \text{ leaving } A} f(e) - \sum_{e \text{ entering } A} f(e)$$

cut: A = {s}

net flow across cut = 10 + 5 + 10 = 25



value of flow = 25

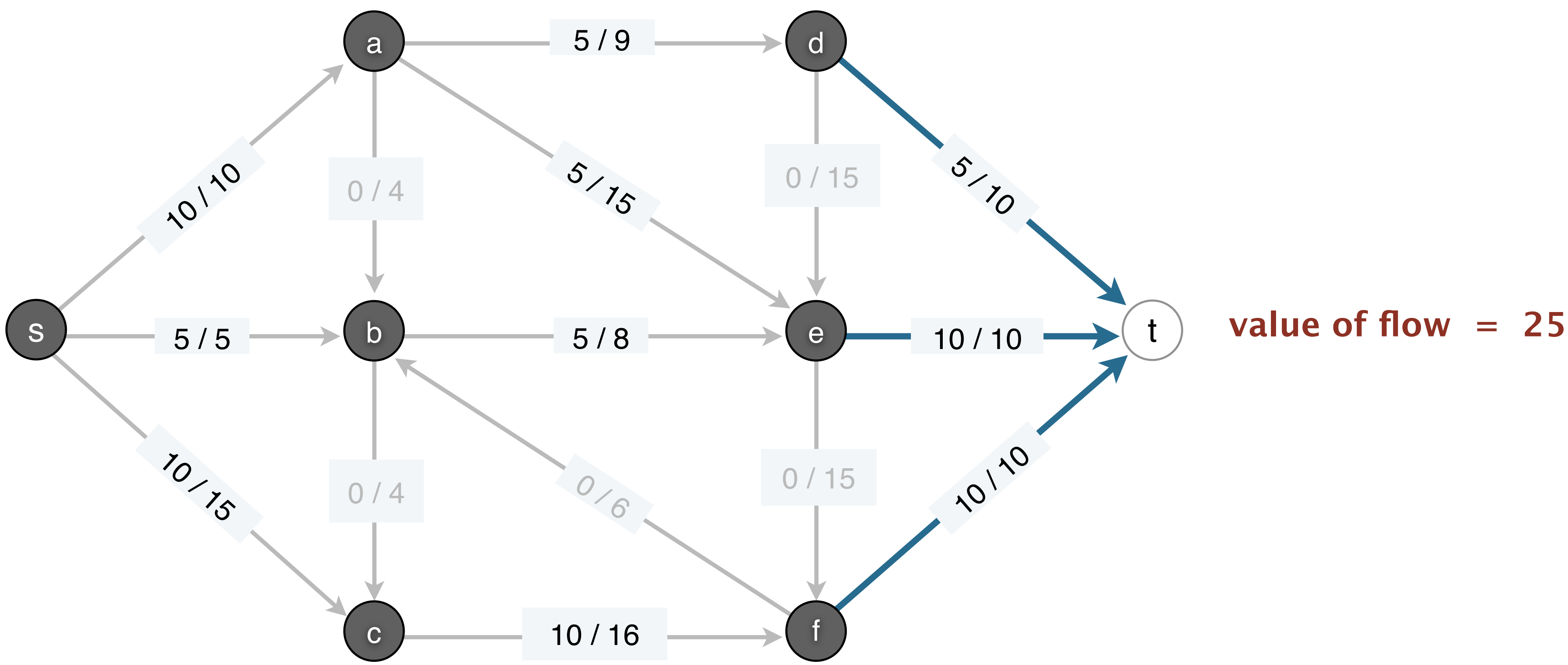
Relationship between flows and cuts – MFMC proof

Flow value lemma. Let f be *any* flow and let (A, B) be *any* cut. Then, the value of the flow f equals the net flow across the cut (A, B).

$$val(f) = \sum_{e \text{ leaving } A} f(e) - \sum_{e \text{ entering } A} f(e) = \sum_{u \in A, v \notin A} f(u,v) - \sum_{v \notin A, u \in A} f(v,u)$$

cut: A = {s, a, b, c, d, e, f}

net flow across cut = 5 + 10 + 10 = 25



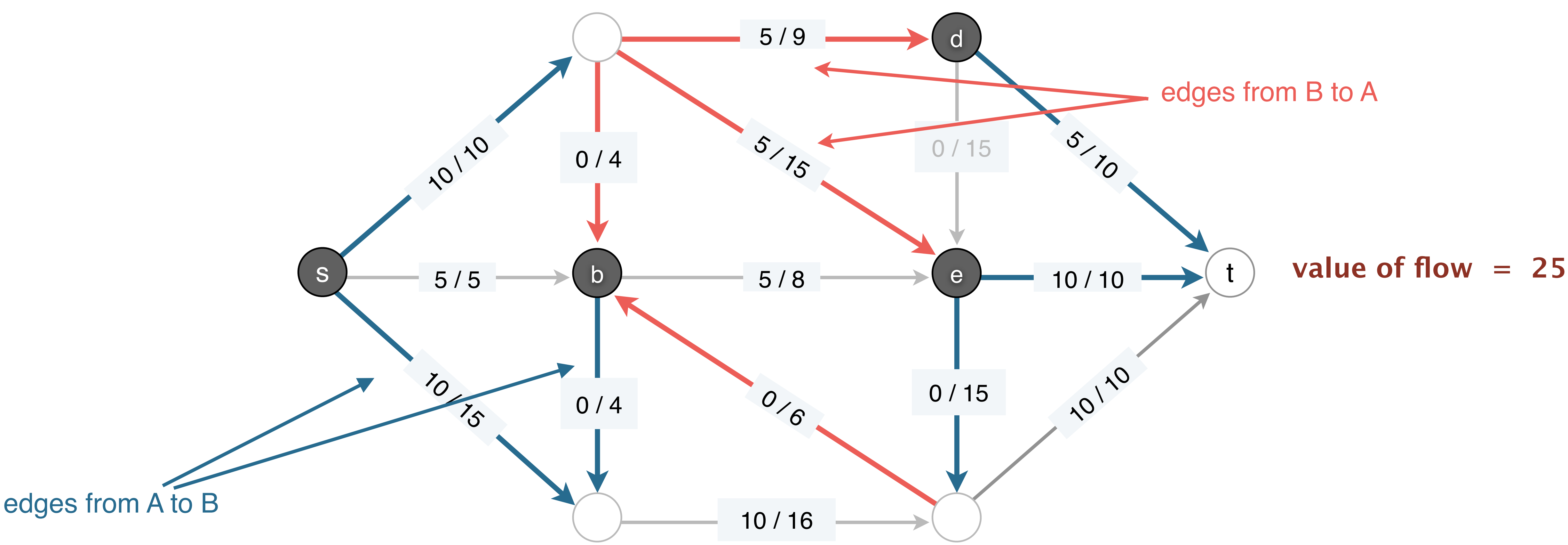
Relationship between flows and cuts – MFMC proof

Flow value lemma. Let f be any flow and let (A, B) be any cut. Then, the value of the flow f equals the net flow across the cut (A, B).

$$val(f) = \sum_{e \text{ leaving } A} f(e) - \sum_{e \text{ entering } A} f(e)$$

cut: A = {s, b, d, e}

net flow across cut = **(10 + 10 + 5 + 10 + 0 + 0) – (5 + 5 + 0 + 0) = 25**



Relationship between flows and cuts – MFMC proof

Flow value lemma. Let f be *any* flow and let (A, B) be any cut. Then, the value of the flow f equals the net flow across the cut (A, B).

$$val(f) = \sum_{e \text{ leaving } A} f(e) - \sum_{e \text{ entering } A} f(e)$$

Pf.

Relationship between flows and cuts – MFMC proof

Flow value lemma. Let f be *any* flow and let (A, B) be any cut. Then, the value of the flow f equals the net flow across the cut (A, B).

$$val(f) = \sum_{e \text{ leaving } A} f(e) - \sum_{e \text{ entering } A} f(e)$$

Pf.

$$val(f) = \sum_{e\ leaving\ s} f(e) - \sum_{e\ entering\ s} f(e) =$$

definition of val(f)

edges pointing from v

$$= \sum_{v \in A} \left(\sum_{e=(v,w)\in E} f(e) - \sum_{e=(w,v)\in E f(e)} f(e) \right) =$$

by flow conservation, all terms in the sum are 0, except for v = s

edges pointing towards v

$$= \sum_{e\ leaving\ A} f(e) - \sum_{e\ entering\ A} f(e)$$

edges with both ends in A appear once with '+' once with '-' in the sum and cancel out. Edges with only one end in A contribute to the sum.

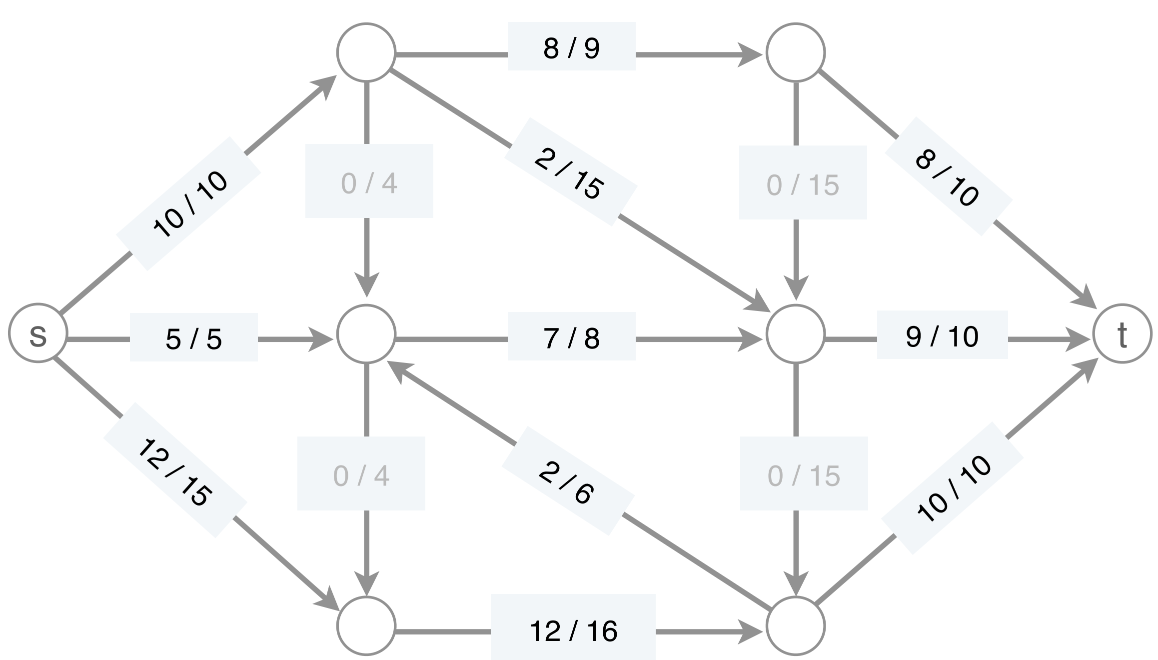
Relationship between flows and cuts – MFMC proof

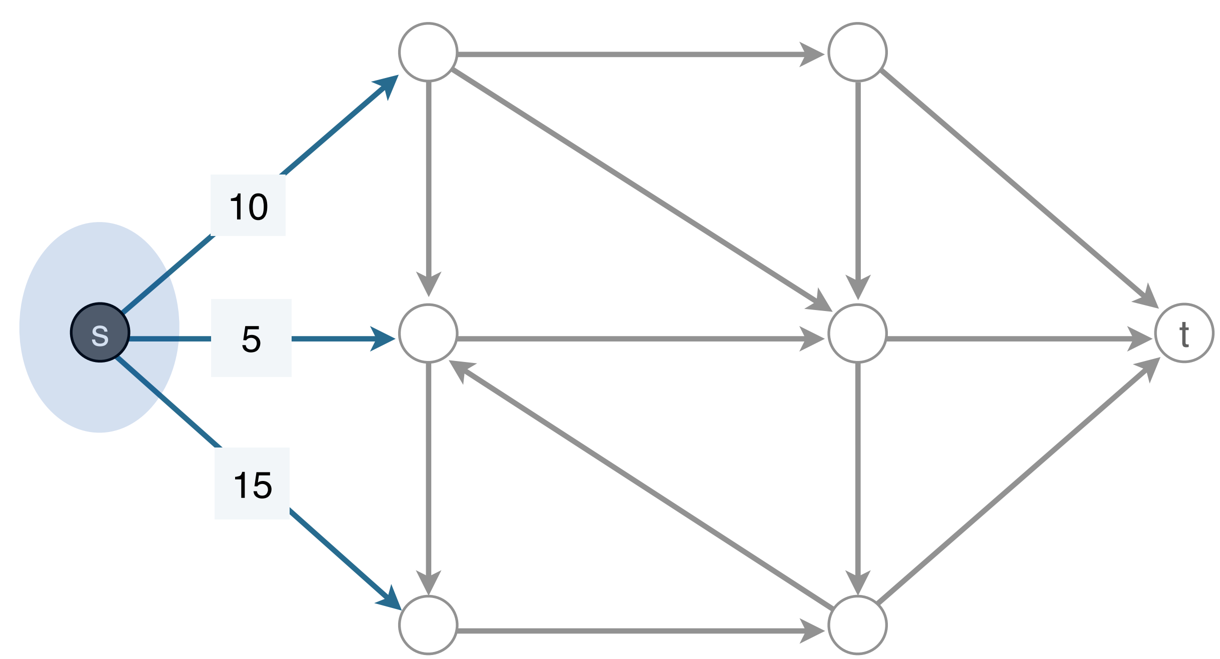
Weak duality. Let f be *any* flow and (A, B) be *any* cut. Then, $v(f) \leq cap(A, B)$.

Pf.

$$
\begin{aligned}
val(f) &= \sum_{e \text{ out of } A} f(e) \;-\; \sum_{e \text{ in to } A} f(e) \\
&\leq \sum_{e \text{ out of } A} f(e) \\
&\leq \sum_{e \text{ out of } A} c(e) \\
&= cap(A, B) \quad \blacksquare
\end{aligned}
$$

flow-value lemma





value of flow = 27 $\leq$ **capacity of cut = 30**

Certificate of optimality – MFMC

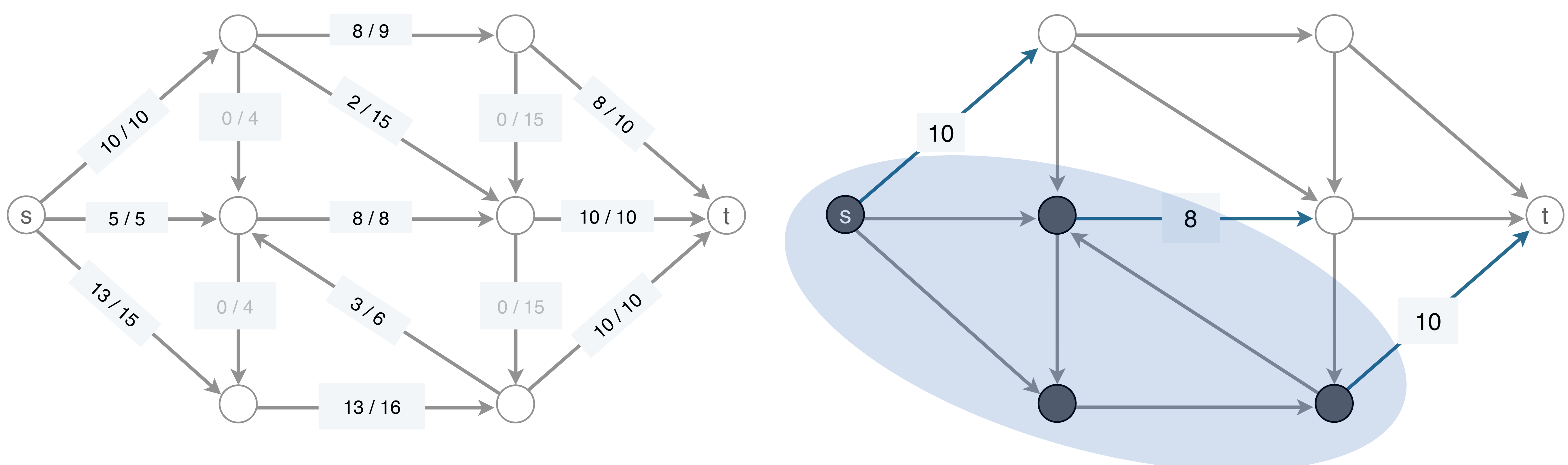
Corollary. Let f be a flow and let (A, B) be any cut.
If $val(f) = cap(A, B)$, then f is a max flow and (A, B) is a min cut.

Pf.

weak duality

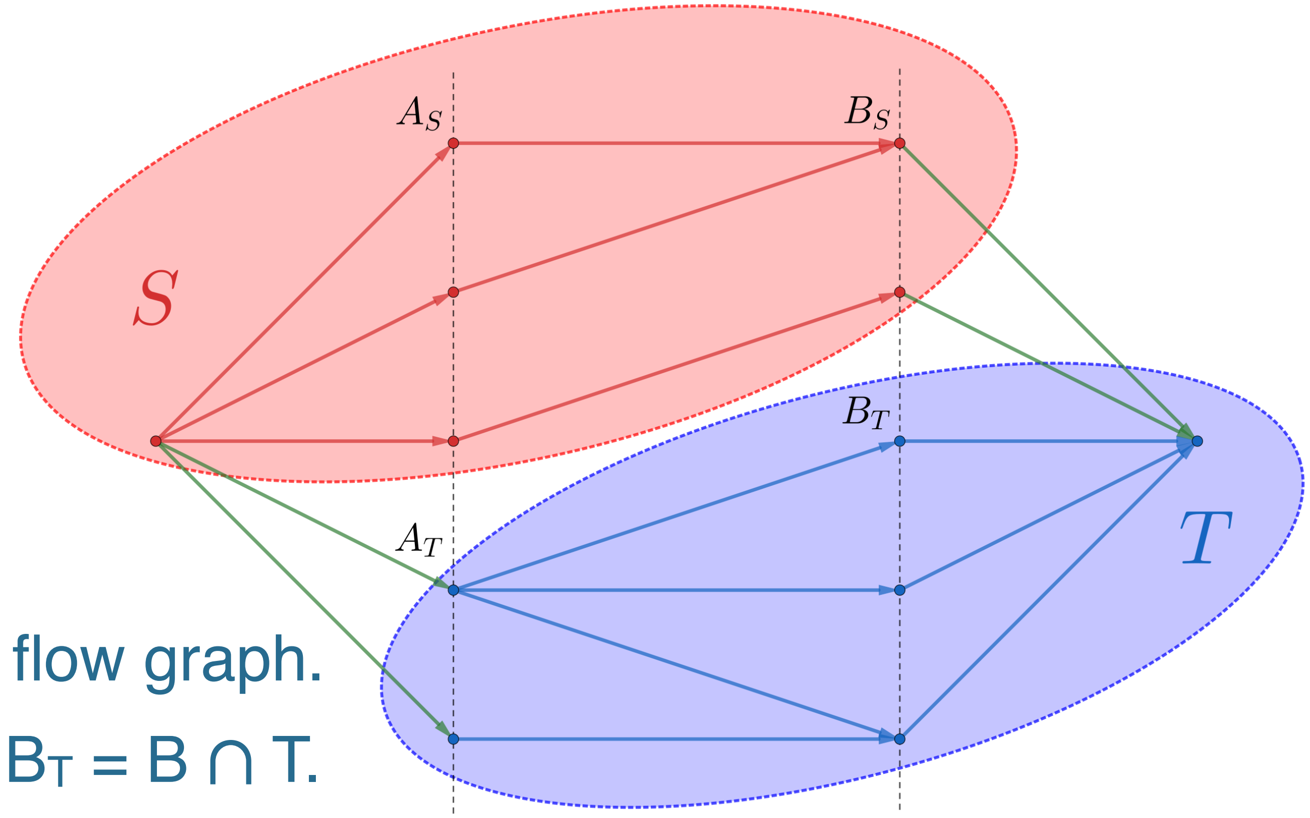
- For any flow f', $val(f') \leq cap(A, B) = val(f)$.
- For any cut (A', B'), $cap(A', B') \geq val(f) = cap(A, B)$.

Conclusion: if we can find a cut with the same capacity as the flow, then it's a maximum flow.



value of flow = 28 = **capacity of cut = 28**

Kőnig's theorem (1931)



The size of the maximum matching of a bipartite graph is the same size as its minimum vertex cover (not set cover!).

1. Let (S, T) be a min-cut of the maximum flow graph.
2. Let $A_S = A \cap S$, $A_T = A \cap T$, $B_S = B \cap S$, $B_T = B \cap T$.
3. The only cut edges are from s to A_T, and B_S to T.
    1. A_S and B_T cannot be connected, since the edge weights would be ∞ and the cut would not be a min-cut. Same for A_T and B_S.
    2. A_S to B_S are internal to S, and A_T to B_T are internal to T.
4. The size of the min-cut is $|A_T| + |B_S|$.
    1. Also the maximum flow and maximum matching.
    2. Lower bound on vertex cover since maximum matching edges are disjoint.
5. $A_T \cup B_S$ is a vertex cover of the bipartite graph.
    1. Any missing edge would have to be from As to B_T, but rejected those above.
    2. Matches lower bound, so this is minimum vertex cover.

Image source: https://en.wikipedia.org/wiki/Kőnig's_theorem_(graph_theory)

Faster Maximum Flow Algorithms

"A new approach to the maximum flow problem"

by Goldberg and Tarjan (1986). "A new approach to the maximum flow problem"

- Preflow-push (Push-relabel) algorithm
- $O(n^2m)$ with basic implementation
- $O(n^2m^{1/2})$ with highest label node selection rule (fastest in practice)
- $O(nm\ log(n^2/m))$ using dynamic trees (slow in practice)

Max Flows in $O(nm)$ Time, or Better

by Orlin (2013)

Maximum Flow and Minimum-Cost Flow in Almost-Linear Time

by Chen et al (2022)

- "There is an algorithm that on a graph G with m edges with integral capacities in [1, C] computes a maximum flow between two vertices in time $O(m^{1+o(1)}\ log\ C)$ with high probability."
  (variables tweaked to match these slides, added O() to statement)

Faster Maximum Flow Algorithms

"A new approach to the maximum flow problem"

by Goldberg and Tarjan (1986). "A new approach to the maximum flow problem"

- Preflow-push (Push-relabel) algorithm
- $O(n^2m)$ with basic implementation
- $O(n^2m^{1/2})$ with highest label node selection rule (fastest in practice)
- $O(nm \log(n^2/m))$ using dynamic trees (slow in practice)

Max Flows in $O(nm)$ Time, or Better

by Orlin (2013)

Maximum Flow and Minimum-Cost Flow in Almost-Linear Time

by Chen et al (2022)

- "There is an algorithm that on a graph G with m edges with integral capacities in [1, C] computes a maximum flow between two vertices in time $O(m^{1+o(1)}$ ***log C***$)$ **with high probability**."
  (variables tweaked to match these slides, added O() to statement)