CS330 Introduction to Analysis of Algorithms

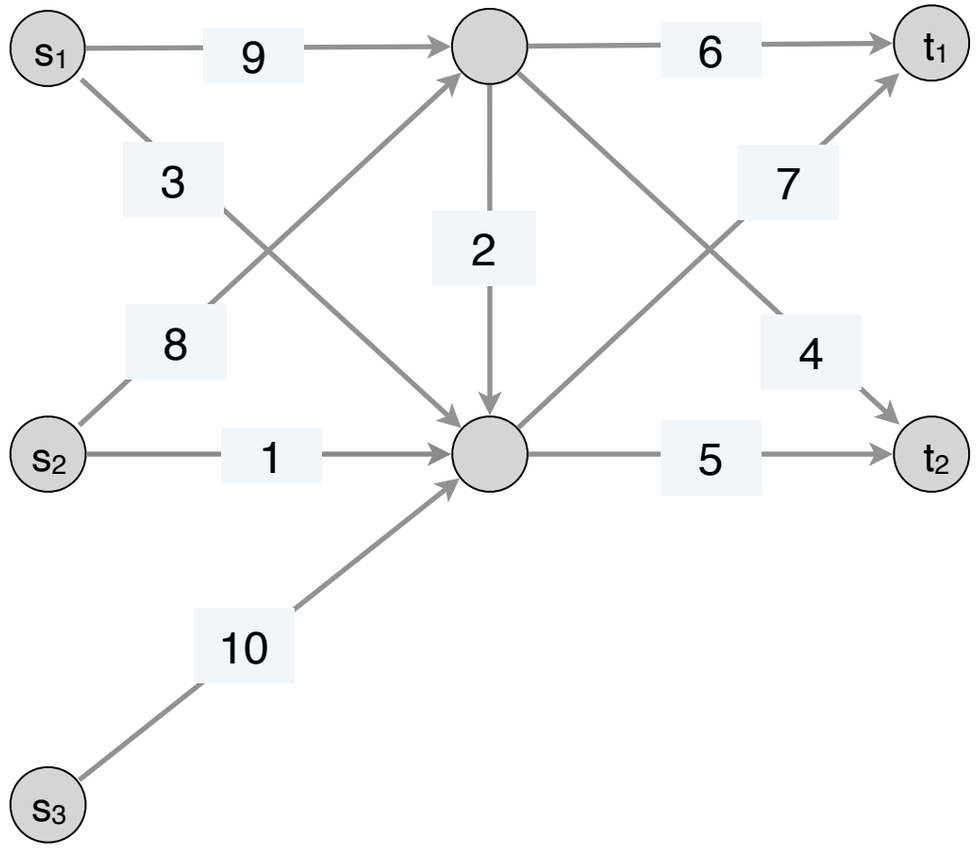
September, 2024

by Dora Erdos and Jeffrey Considine

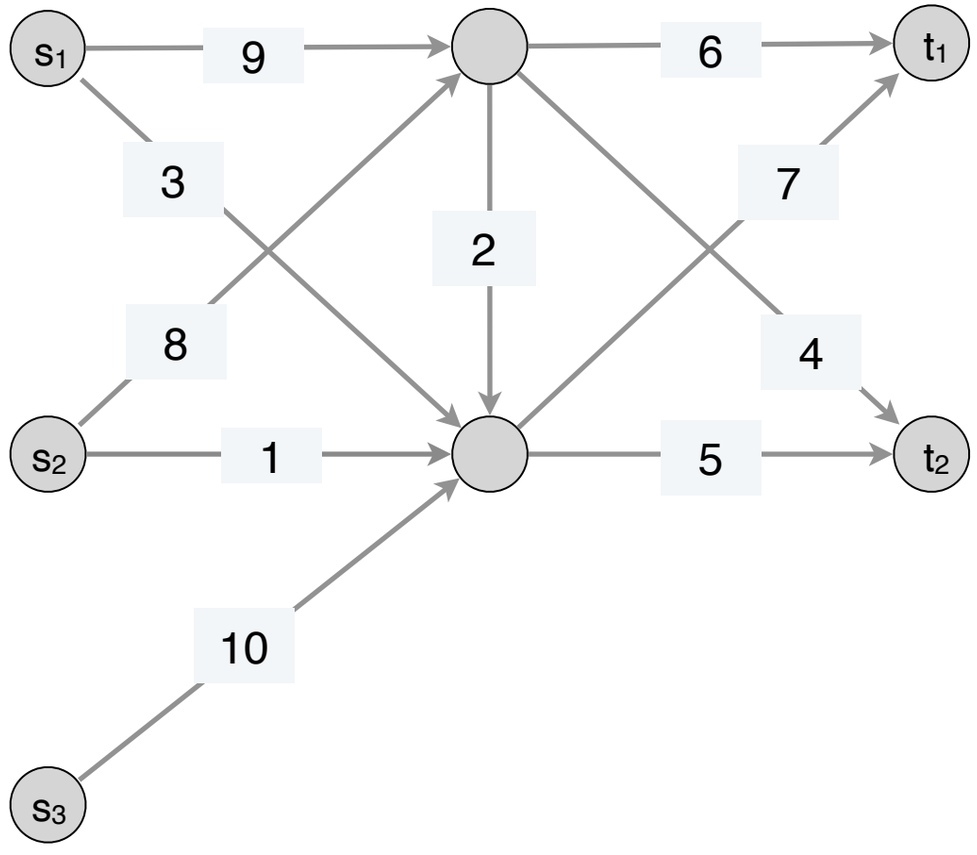
- max flow variations
- Max Flow Min Cut
- Flow applications
- Certificates

Multiple sources and sinks

The network below represents the supply network of a company. The source nodes are factories that ship product to the stores represented by the sinks. The edge weights correspond to shipping capacity between the locations. Find the maximum amount of products that can be shipped from the factories to the stores.



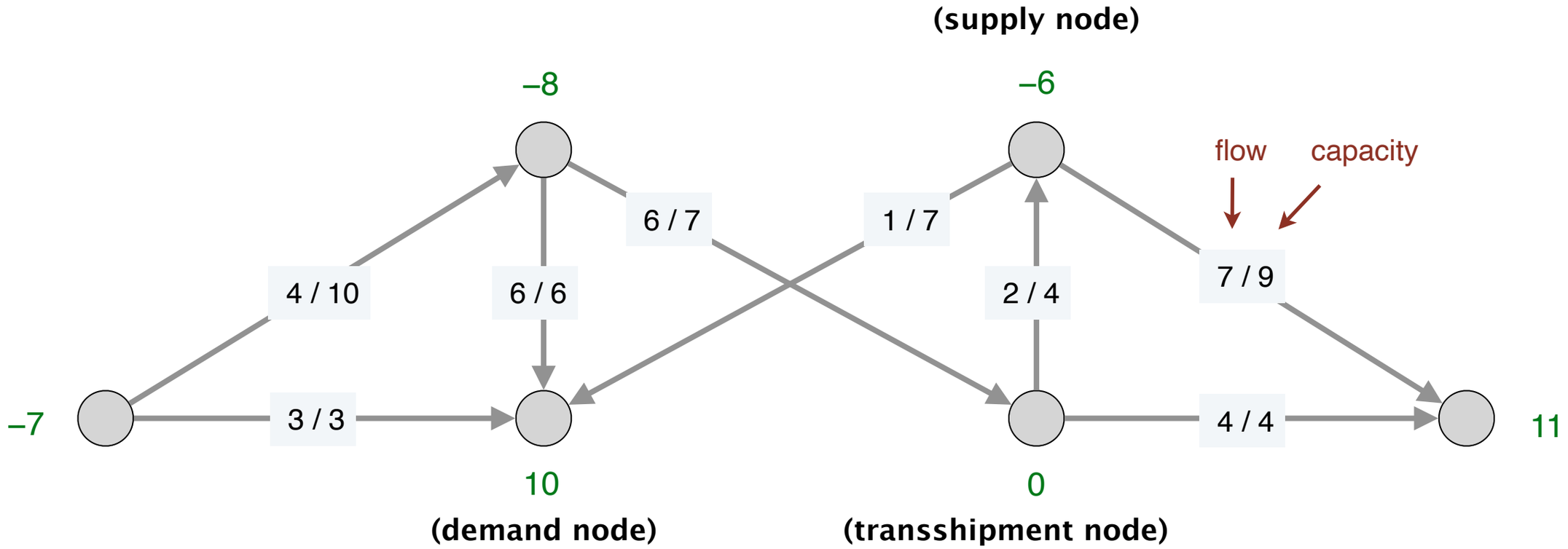
Multiple sources and sinks



Circulation with demand

G(V,E) is a directed graph with edge capacities c(u,v). Each node has either a demand (= needs material) or supply (= has excess material). Find out whether it's possible to meet all demands by sending flow from the nodes with supply.

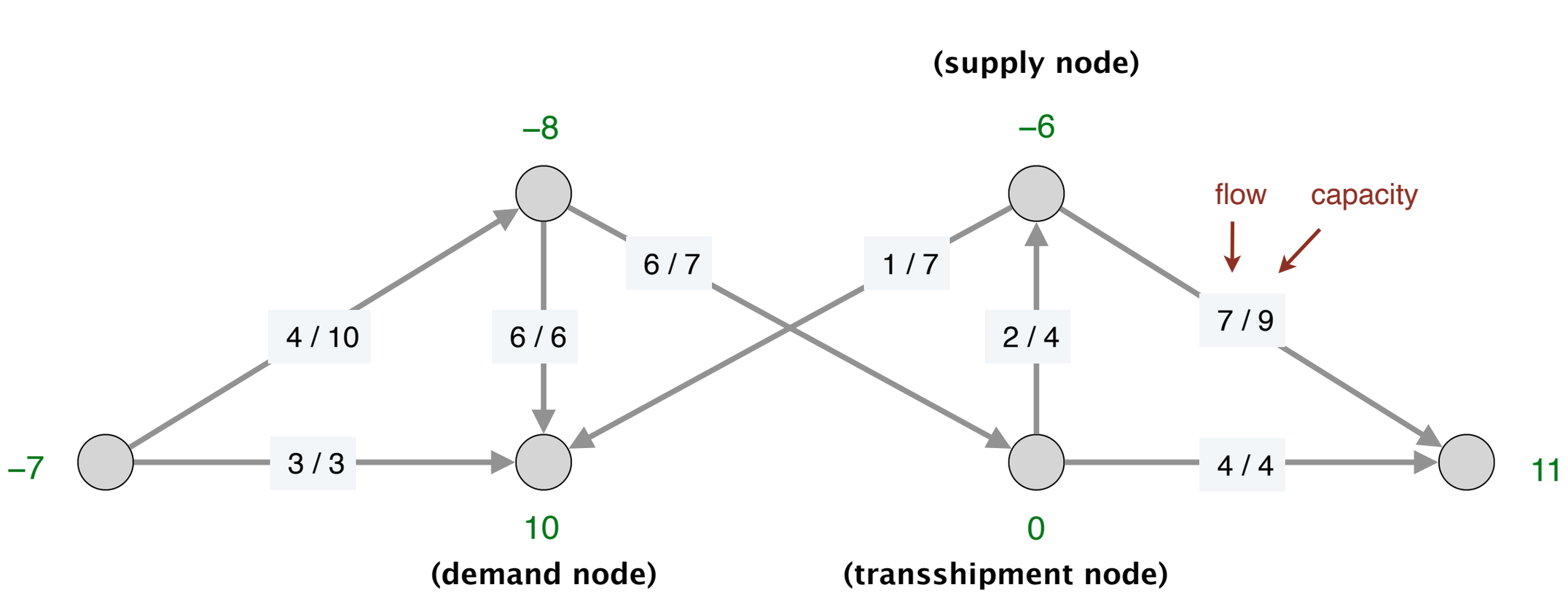
- d(v) for each node represents demand or supply
  - $d(v) < 0 \Rightarrow$ v has supply
  - $d(v) > 0 \Rightarrow$ v has demand



Circulation with demand

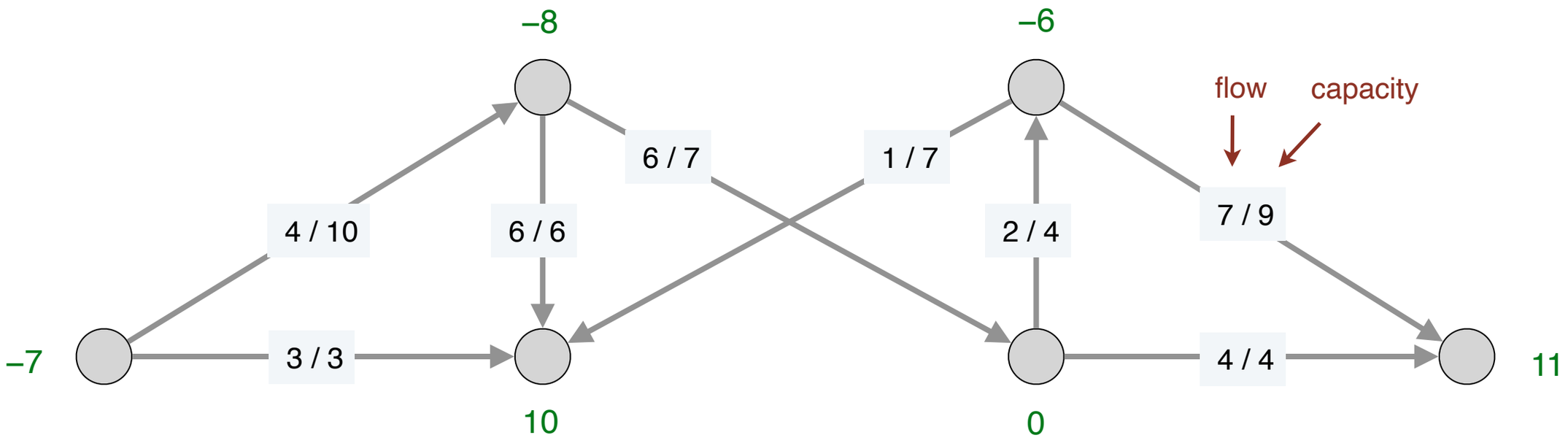
Circulation: (a variation on flow) a function f(e) on the edges that satisfies:

- for each edge e: $0 \leq f(e) \leq c(e)$ [capacity]
- for each node v: $\sum_{e \text{ into } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ [conservation]



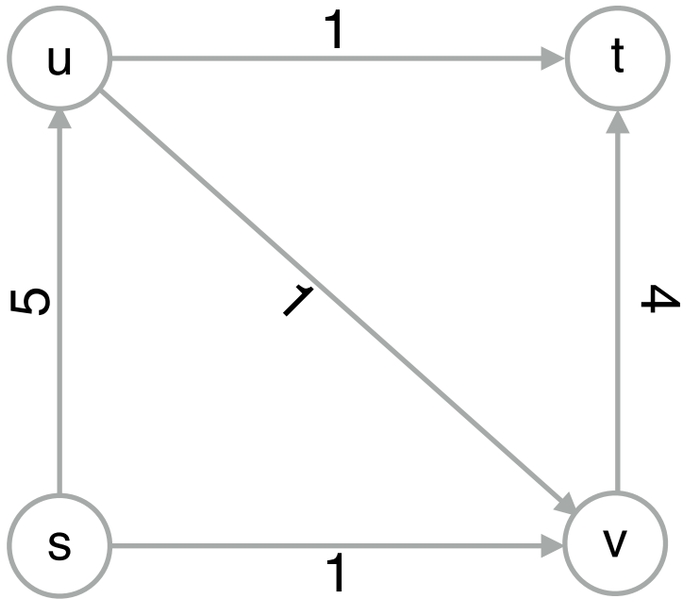
Circulation with demand

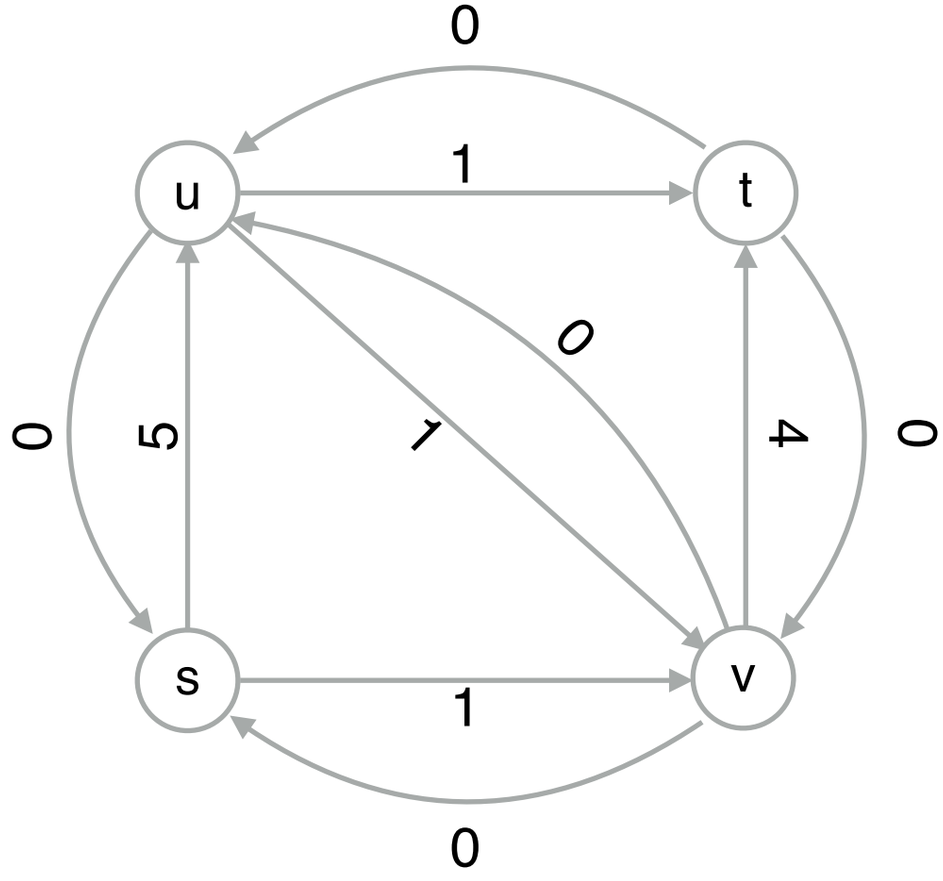
Circulation: G has a valid circulation iff value(f) =



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

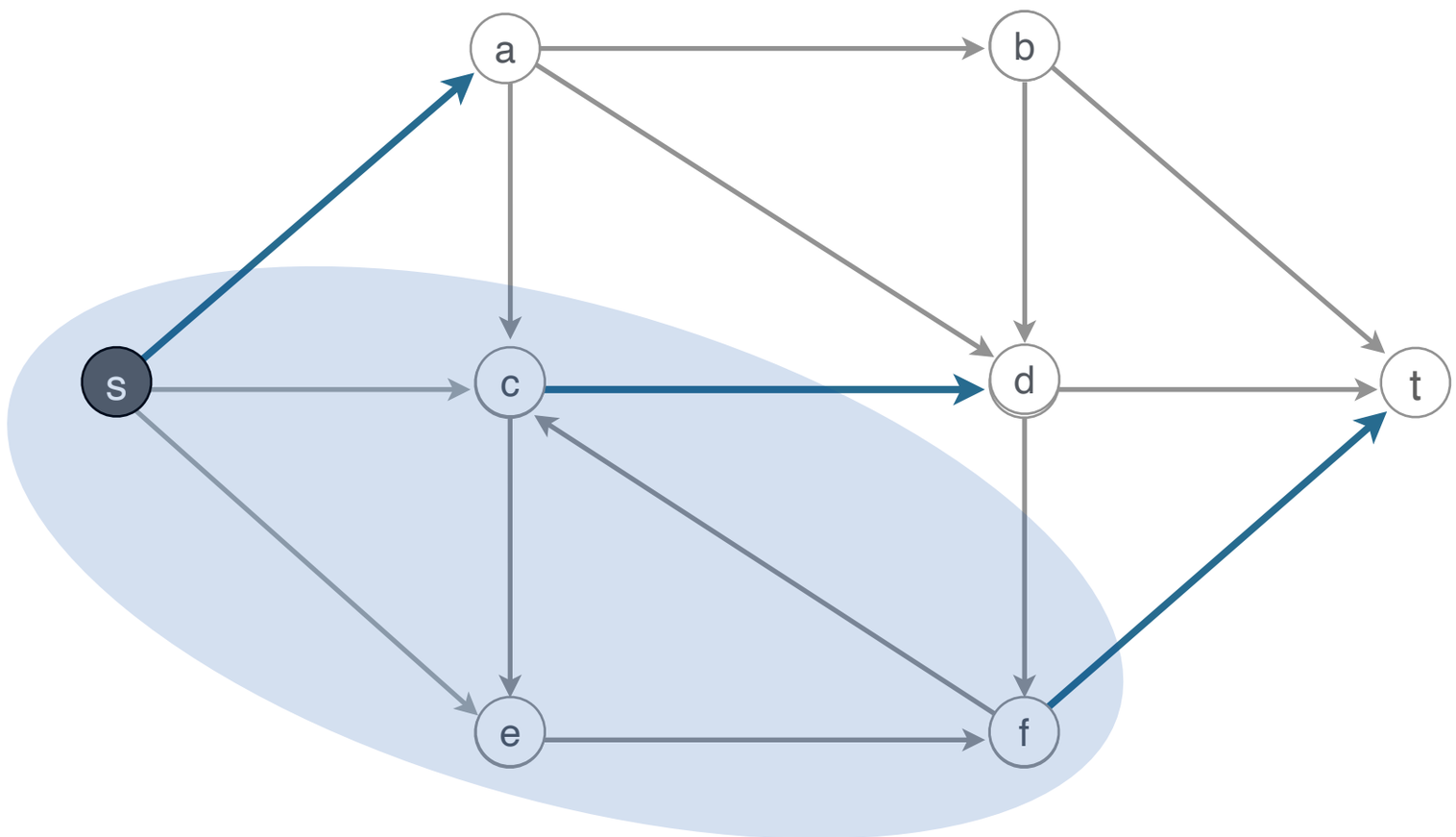
Def. An st-cut (cut) is a partition $(S, V\text{-}S)$ of the vertices with $s \in S$ and $t \in V - S$.

- This is the same notion as we discussed for spanning trees

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

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

- Note, that this only contains edges *directed from S to V-S*

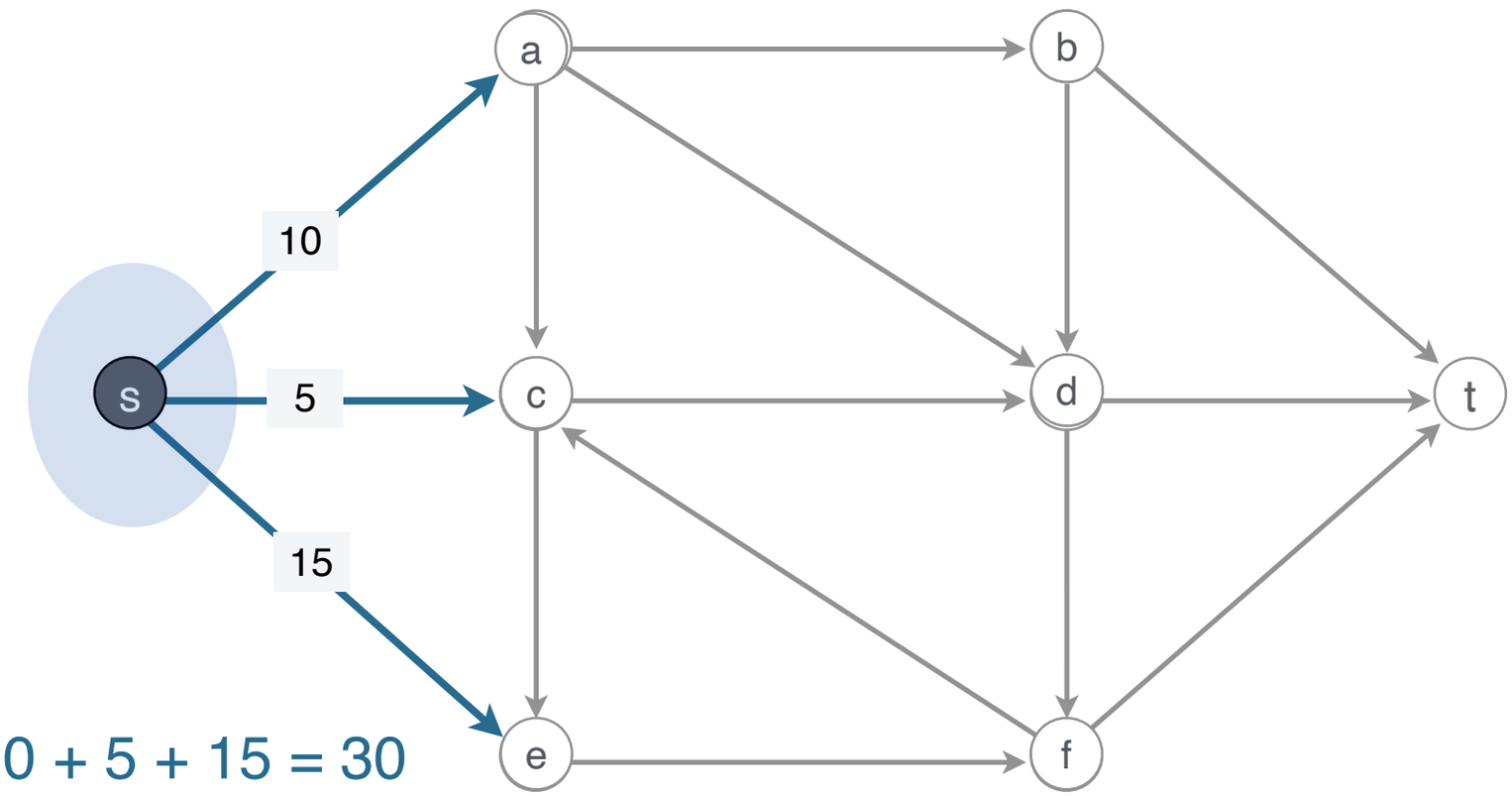


Minimum-cut problem

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

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

$$cap(S) = \sum_{e \text{ out of } S} c(e)$$



S = {s}

capacity = 10 + 5 + 15 = 30

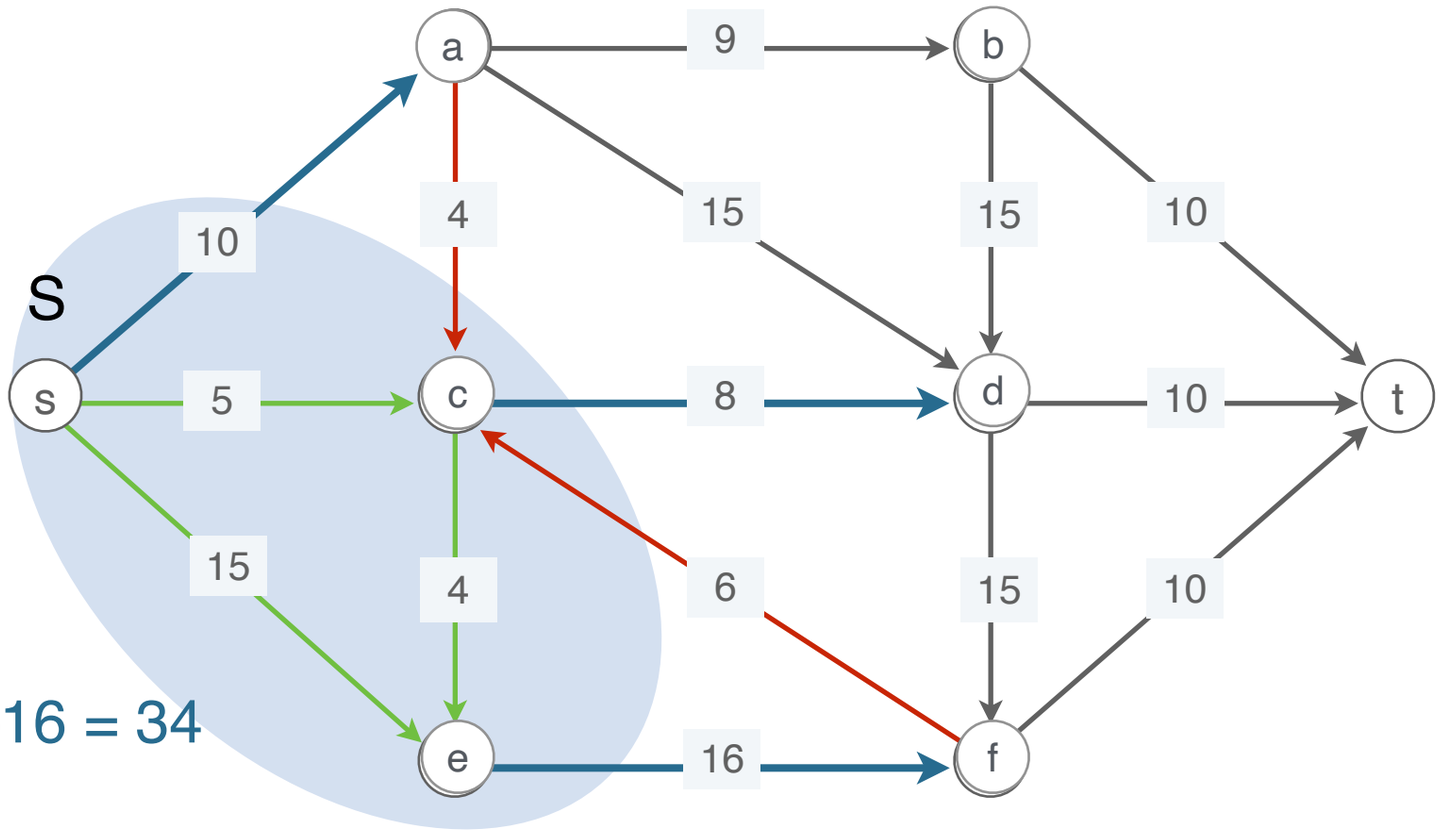
st-cuts

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

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

$$cap(S) = \sum_{e \ out \ of \ S} c(e)$$

Note: red and green edges are not counted towards the capacity of the cut S



S= {s, c, e}

cap(A) = 10+8+16 = 34

Minimum-cut problem

Def. The capacity of a cut is the sum of the capacities of the edges *from* S *to* V-S.

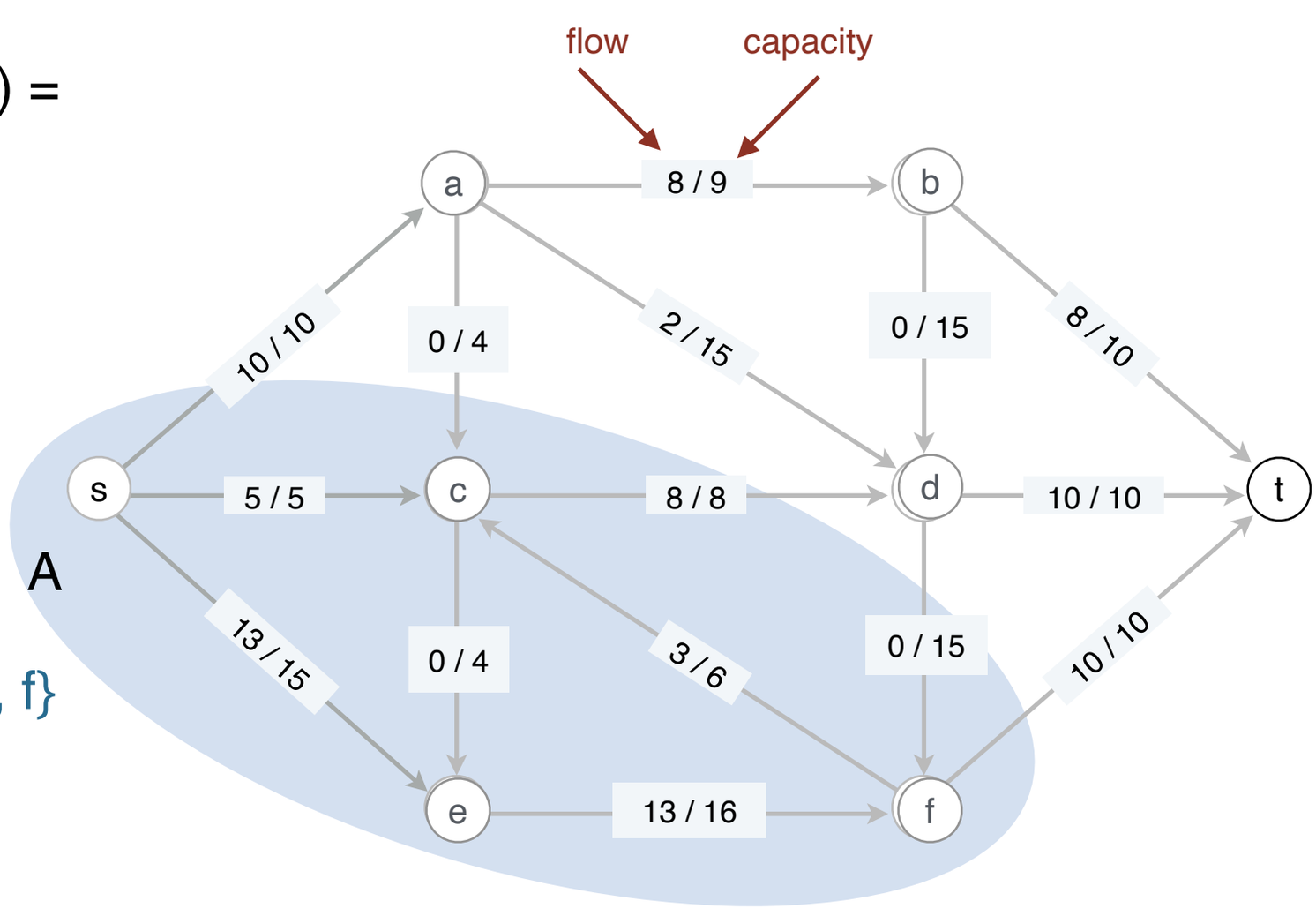
$$cap(S) = \sum_{e \text{ out of } S} c(e)$$

Min st-cut problem: Find an st-cut with minimum capacity.

cap(S) =

value(f) =

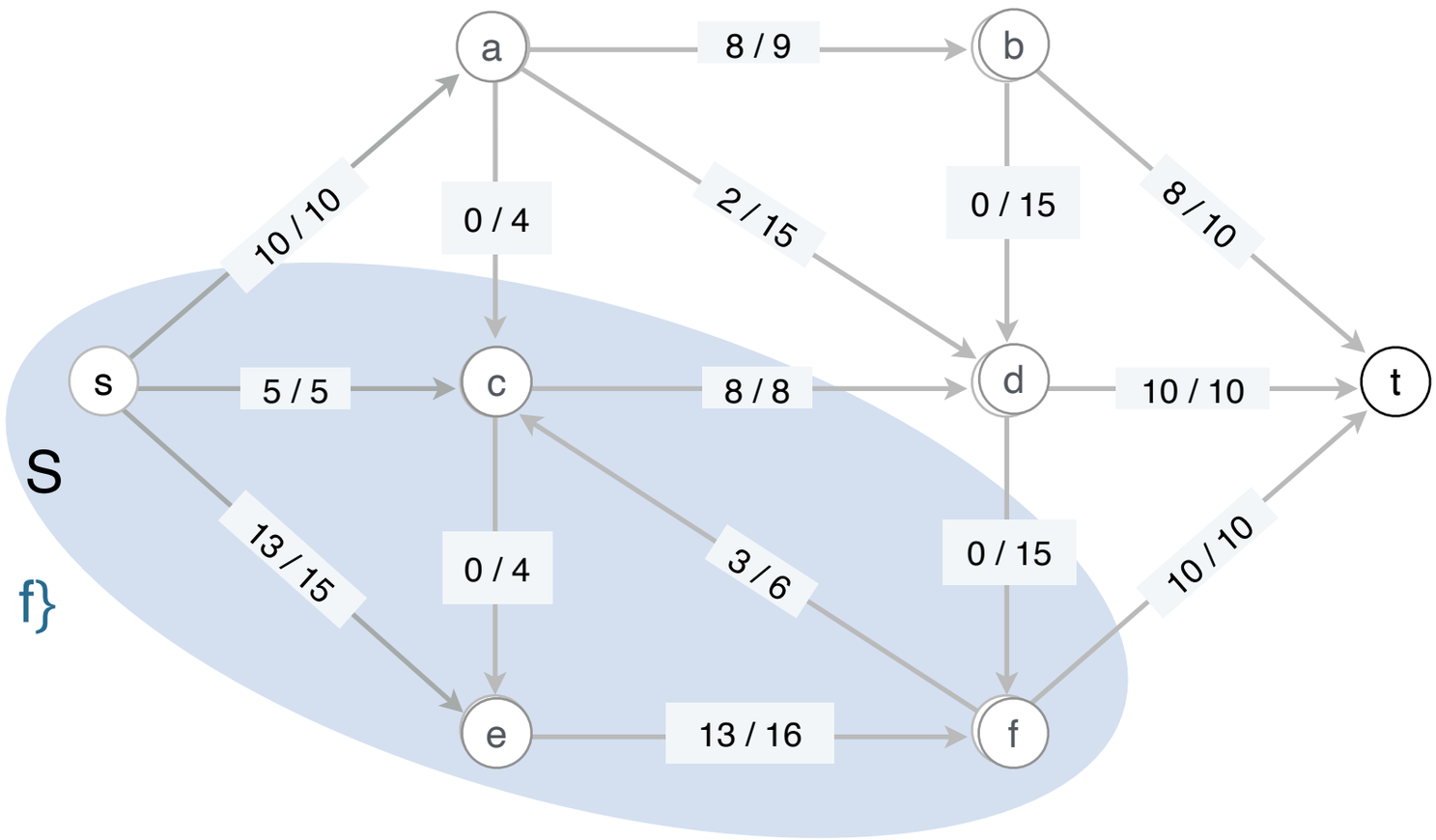
S= {s, c, e, f}



Relationship between the max flow and min cut

intuition:

- any amount of flow from s to t will cross one of the edges in the min-cut
- the capacity of a cut is the “throughput” of that cut
- min-cut is the bottleneck throughput



S= {s, c, e, f}

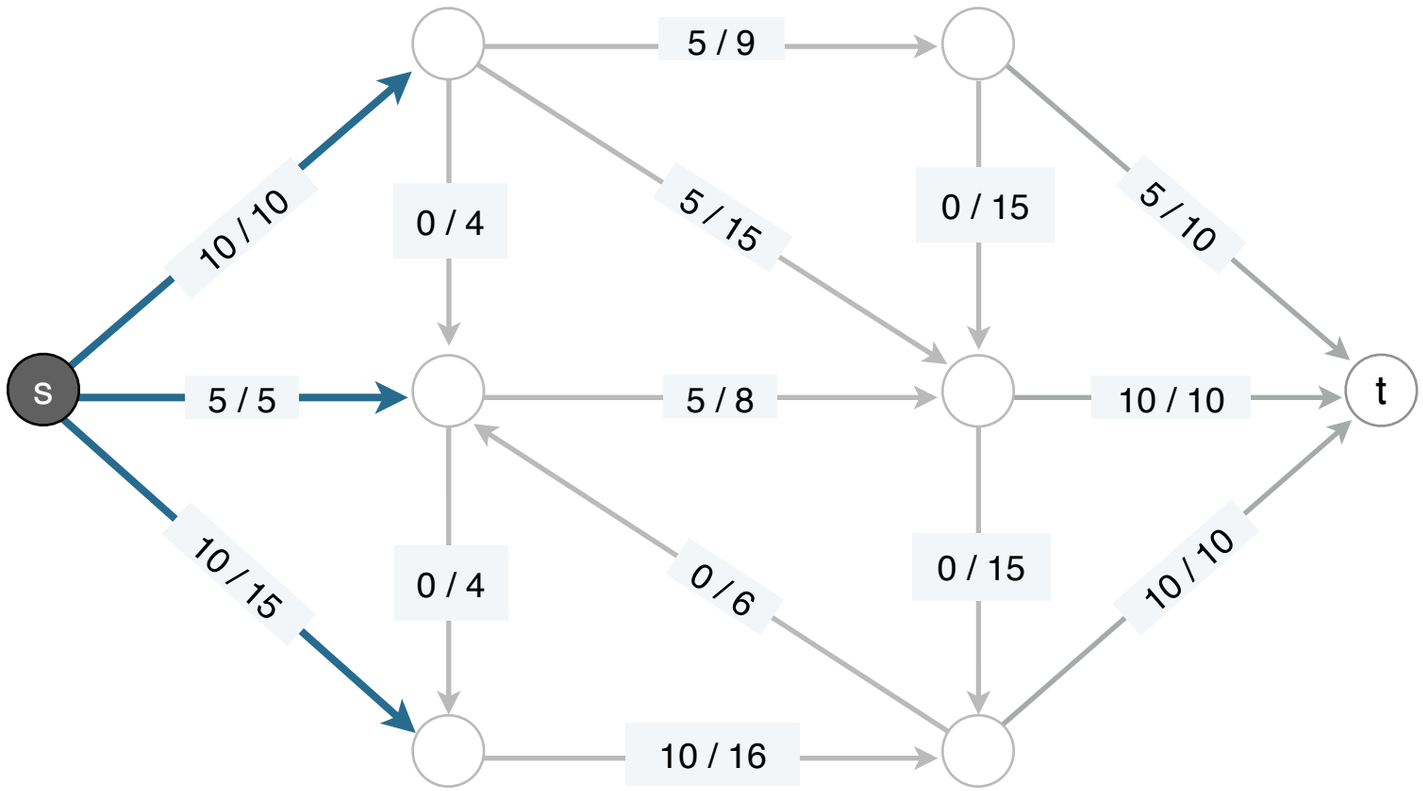
Relationship between flows and cuts

Flow value lemma. Let f be any flow and let S be any cut. Then, the value of the flow f equals the net flow across the cut S to V-S.

$$value(f) = \sum_{(u,v):u \in S, v \in V-S} f(u,v) - \sum_{(w,z):w \in V-S, z \in S} f(w,z)$$

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

S = {s}



value of flow = 25

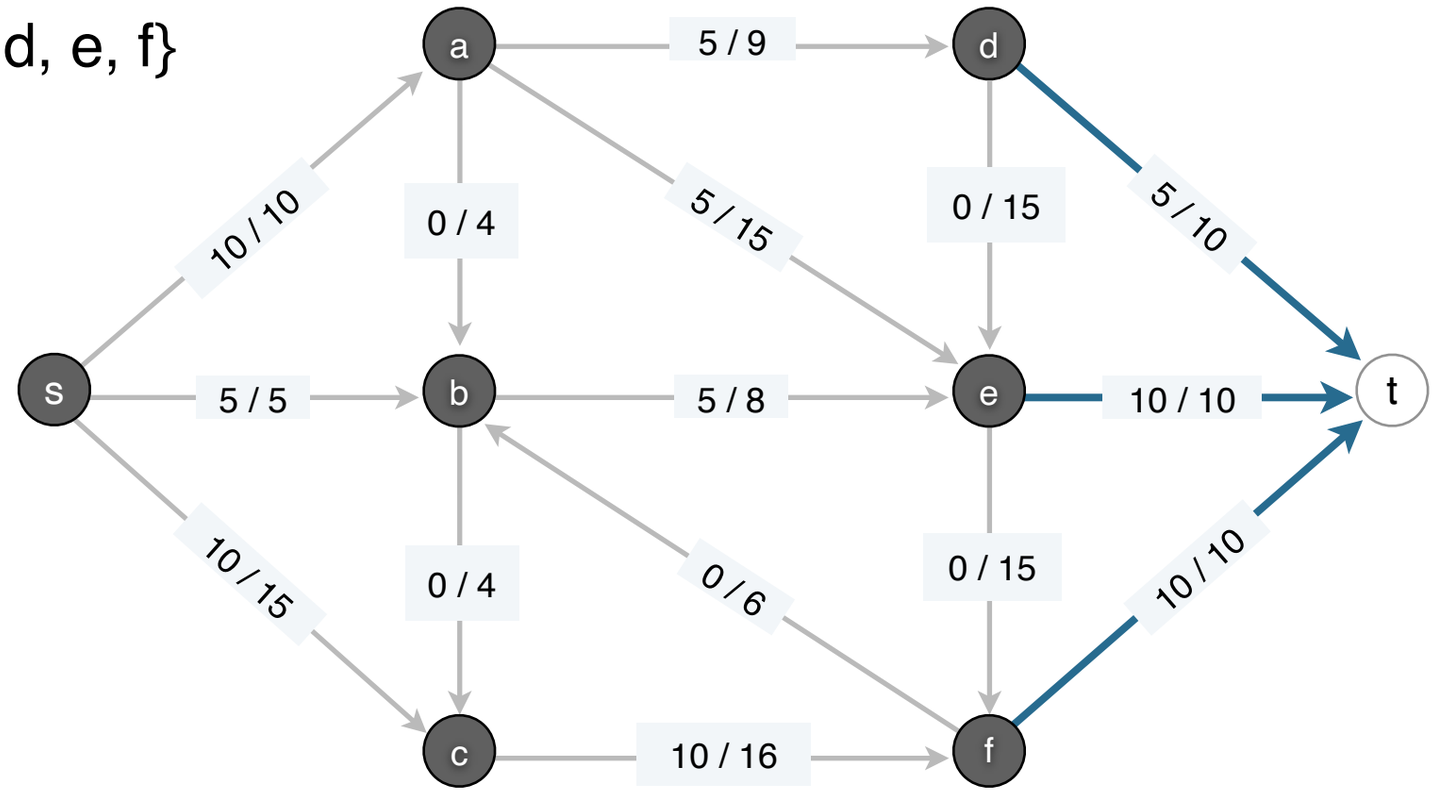
Relationship between flows and cuts

Flow value lemma. Let f be any flow and let S be any cut. Then, the value of the flow f equals the net flow across the cut S to V-S.

$$value(f) = \sum_{(u,v):u \in S, v \in V-S} f(u,v) - \sum_{(w,z):w \in V-S, z \in S} f(w,z)$$

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

S = {s, a, b, c, d, e, f}



value of flow = 25

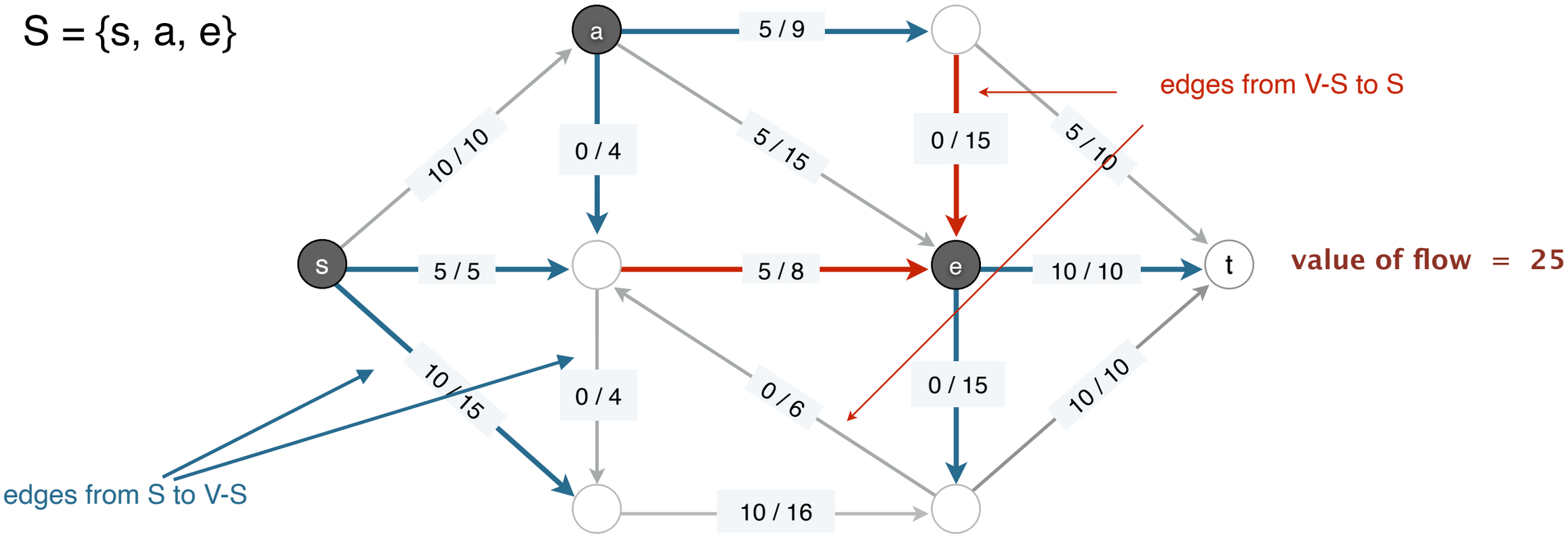
Relationship between flows and cuts

Flow value lemma. Let f be any flow and let S be any cut. Then, the value of the flow f equals the net flow across the cut S to V-S.

$$value(f) = \sum_{(u,v):u \in S, v \in V-S} f(u,v) - \sum_{(w,z):w \in V-S, z \in S} f(w,z)$$

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

S = {s, a, e}



Relationship between flows and cuts

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.

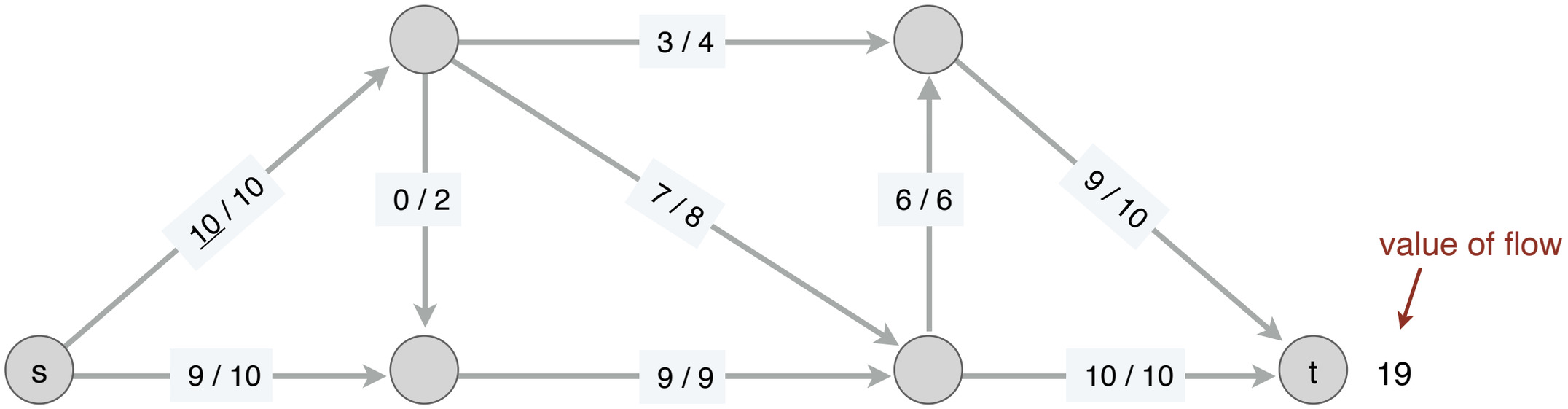
$$\max_{f\ flow} value(f) = \min_{st-cut\ S} cap(S)$$

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

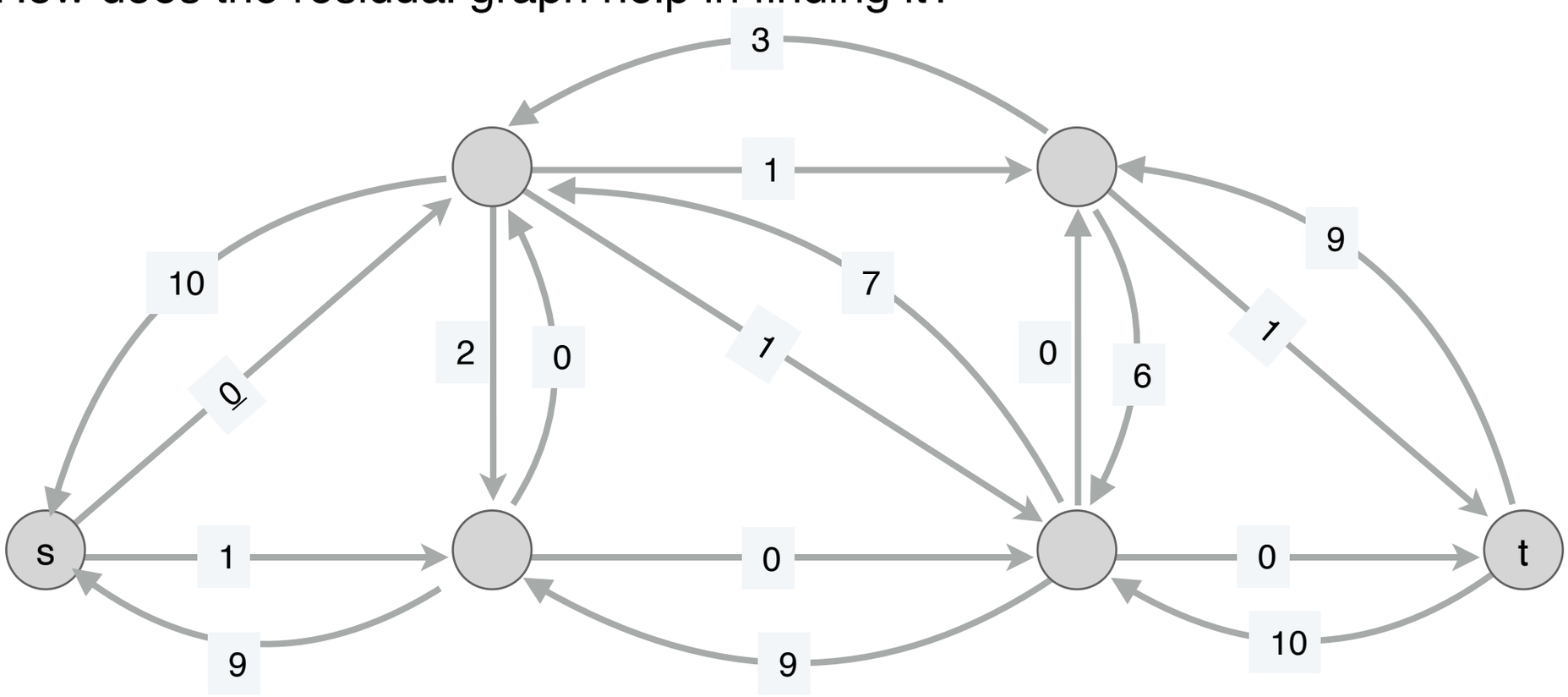
val(f) is maximum if there is a cut with its capacity equal to *val(f)*.

Certificate for the max flow

- Find evidence (a certificate) that the value of the max flow cannot be >19



- How does the residual graph help in finding it?



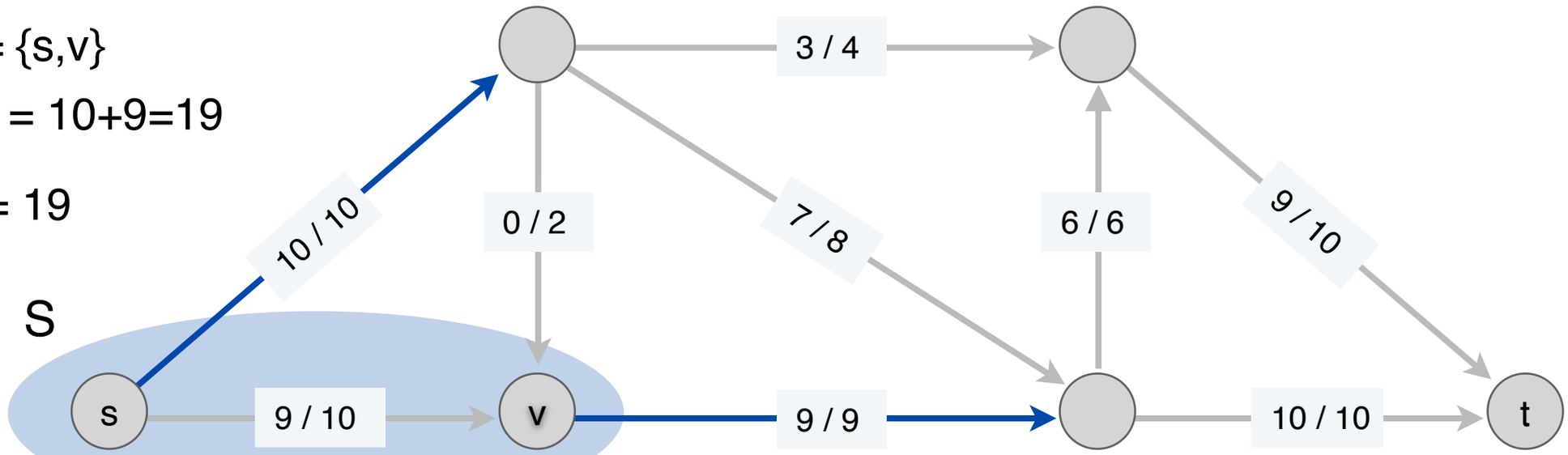
Certificate for the max flow

network G and flow f

cut S = {s,v}

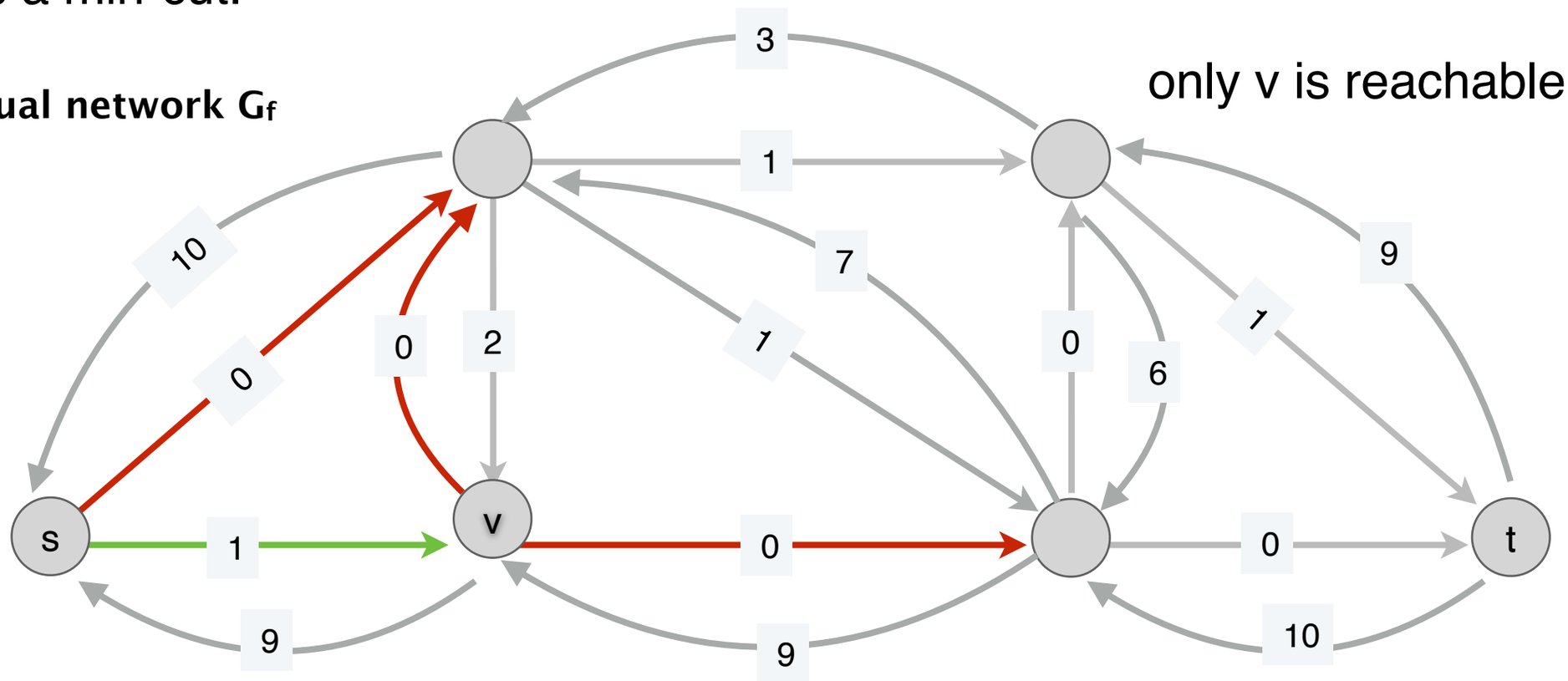
cap(S) = 10+9=19

val(f) = 19



claim: S is a min-cut.

residual network G_f



only v is reachable from s in G_f

Finding *a* min-cut

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

Nodes in S form one minimum capacity cut.

Relationship between flows and cuts

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.

$$\max_{f\ flow} value(f) = \min_{st-cut\ S} cap(S)$$

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)*.

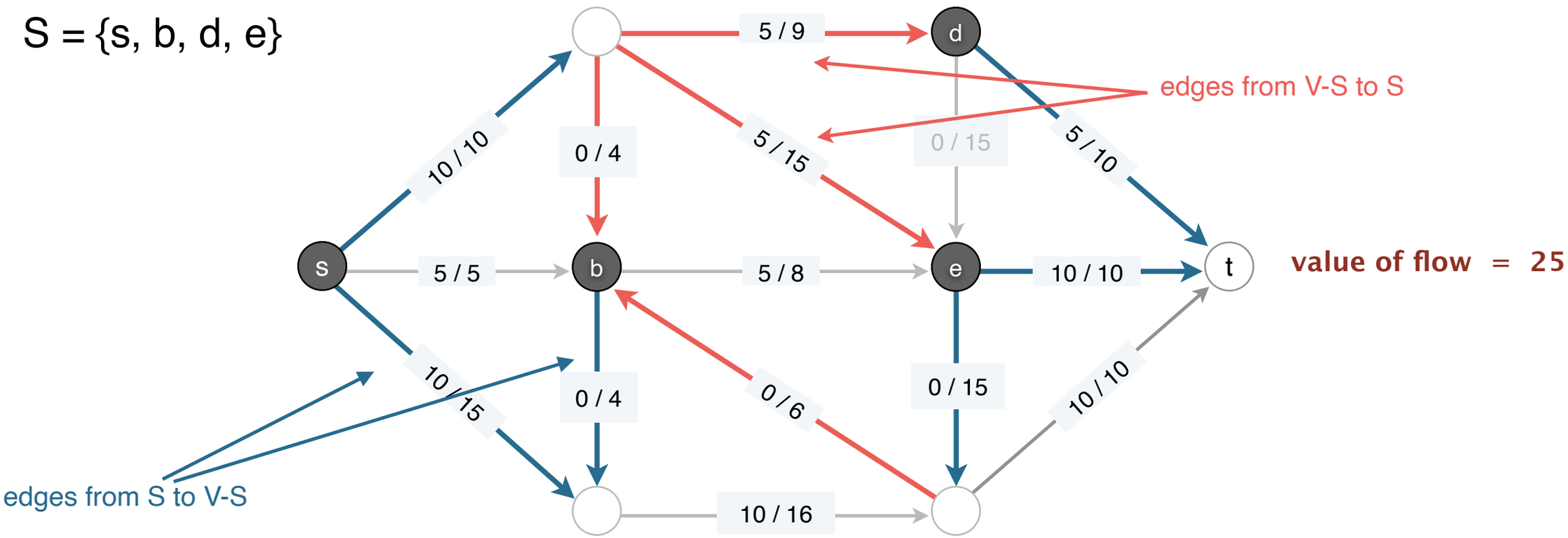
Relationship between flows and cuts — MFMC proof

Flow value lemma. Let f be any flow and let S be any cut. Then, the value of the flow f equals the net flow across the cut S to V-S.

$$value(f) = \sum_{(u,v):u \in S, v \in V-S} f(u,v) - \sum_{(w,z):w \in V-S, z \in S} f(w,z)$$

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

S = {s, b, d, e}



Relationship between flows and cuts – MFMC proof

Flow value lemma. Let f be any flow and let S be any cut. Then, the value of the flow f equals the net flow across the cut S to V-S.

$$value(f) = \sum_{(u,v):u \in S, v \in V-S} f(u,v) - \sum_{(w,z):w \in V-S, z \in S} f(w,z)$$

proof:

Relationship between flows and cuts – MFMC proof

Flow value lemma. Let f be any flow and let S be any cut. Then, the value of the flow f equals the net flow across the cut S to V-S.

$$value(f) = \sum_{(u,v):u\in S, v\in V-S} f(u,v) - \sum_{(w,z):w\in V-S, z\in S} f(w,z)$$

proof:

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

definition of val(f)

edges pointing from v

$$= \sum_{v \text{ in } S} \left(\sum_{edge\ (u,v)} f(u,v) - \sum_{edge\ (w,v)} f(w,v) \right) =$$

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

edges pointing towards v

$$= \sum_{e \text{ leaving } S} f(e) - \sum_{e \text{ entering } S} f(e) \leq$$

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.

$$\leq \sum_{e \text{ leaving } S} c(e) - \sum_{e \text{ entering } S} c(e) = cap(S, V-S)$$

cap is an upper bound on flow

definition of cap(S, V-S)

Relationship between flows and cuts — MFMC proof

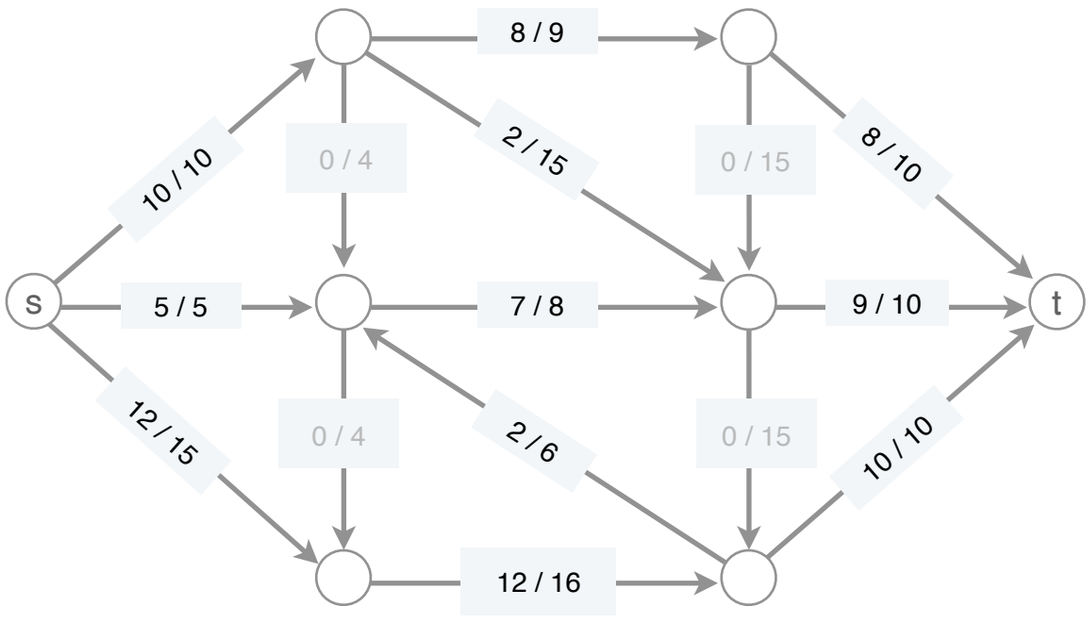
Weak duality. Let f be *any* flow and $(S, V\text{-}S)$ be *any* cut. Then, $v(f) \leq cap(S,V\text{-}S)$.

Pf.

$$val(f) = \sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e) \leq$$

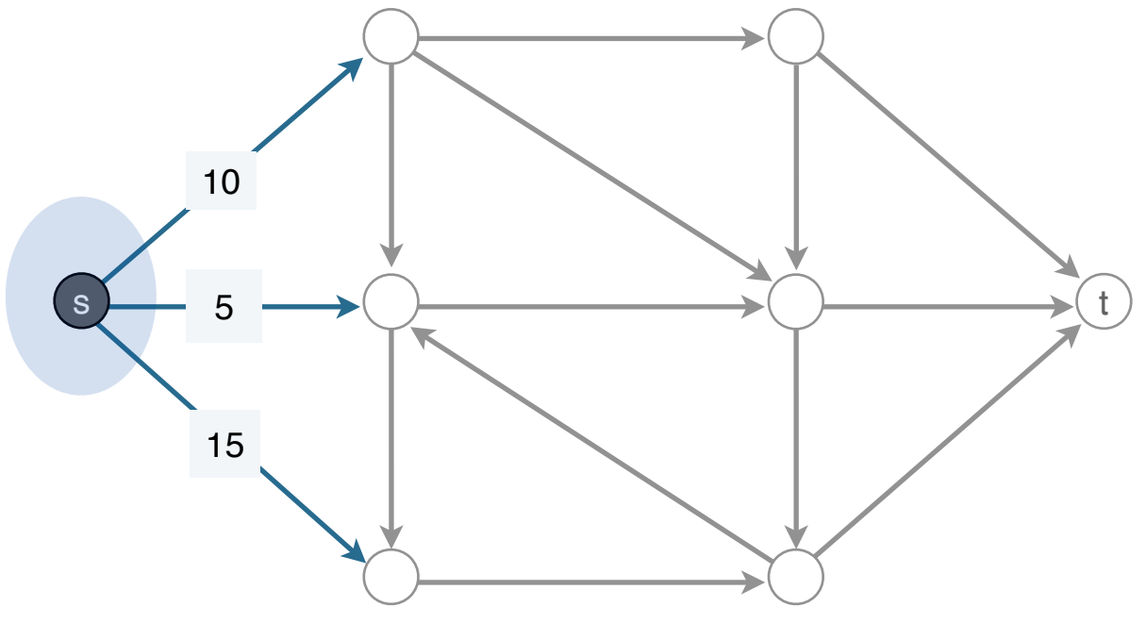
flow-value lemma

$$\leq \sum_{e \text{ out of } S} f(e) \leq \sum_{e \text{ out of } S} c(e) = cap(S, V - S)$$



value of flow = 27

≤



capacity of cut = 30

Certificate of optimality

Corollary. Let f ' be a flow and let $(S', V\text{-}S')$ be any cut. (we don't know whether they are max or min) Then $val(f') = cap(S', V - S')$ iff f ' is a max flow and $(S', V\text{-}S')$ is a min cut.

Pf. Let f be the max flow and $(S, V - S)$be the min cut.

$$val(f') \leq val(f) \leq cap(S, V - S) \leq cap(S', V - S')$$

f' is a flow, f is the max flow

weak duality

S' is a cut, S is the min cut

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

Polynomial Reduction to Problems (informal)

Think of the max-flow solving algorithm (in our case FF - there are others) as a call to an API.

In the applications we used this API to solve other problems.

- max bipartite matching
- disjoint paths
- baseball elimination
- blood donation, dance competition, spies, etc.

Can we do this API-style trick with other algorithms/solvers too?

Polynomial-time reductions

Suppose there is an algorithm to solve problem Y.

- the algorithm is unknown to us, we treat it as a *black-box* or *API*

Polynomial Reduction. Problem X is polynomial-time reducable to problem Y if arbitrary instances of problem X can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to an oracle/black box that solves problem Y.

Notation. $X \leq_P Y$.

Bipartite Matching $\leq_p$ Max Flow

- define flow graph corresponding to the bipartite graph
- find max flow (using FF)
- iterate over middle edges and assign pairs with non-zero flow to the matching

Longest Path and Hamiltonian Path problems

Longest Path: Given a weighted graph *G* and an integer *k*, is there a simple path of lengths at least *k*?

- if G does have a path of length k, it is easy to ”prove” by returning the path itself

Longest Path and Hamiltonian Path problems

Longest Path: Given a weighted graph *G* and an integer *k*, is there a simple path of lengths at least *k*?

- if G does have a path of length k, it is easy to ”prove” by returning the path itself

Hamiltonian-path problem: Given an *un*weighted graph G(V,E) is there a simple path that contains every node exactly once?

review: Depth First Search (DFS) in graphs: given a graph G and a source node s, it explores all nodes reachable from s. The graph traversal explores nodes in such an order that a node v’s child’s child is explored before other children of v are visited. Intuitively we explore the “furthest” node from s first.

Longest Path and Hamiltonian Path problems

Longest Path: Given a weighted graph *G* and an integer *k*, is there a simple path of lengths at least *k*?

- NP-Complete → we don't know of any polynomial algorithm for it
- if G does have a path of length k, it is easy to "prove" by returning the path itself

Hamiltonian-path problem: Given an unweighted graph G(V,E) is there a simple path that contains every node exactly once?

There is a polynomial-time reduction from Hamiltonian-path to Longest Path. Which means that any algorithm solving LP can also be used to solve Ham-Path with polynomial number of extra steps.

Polynomial-time reductions

Suppose there is an algorithm to solve problem Y.

- the algorithm is unknown to us, we treat it as a *black-box* or *API*

Polynomial Reduction. Problem X is polynomial-time reducable to problem Y if arbitrary instances of problem X can be solved using:

- Polynomial number of standard computational steps, plus
- Polynomial number of calls to an oracle/black box that solves problem Y.

Notation. $X \leq_P Y$.