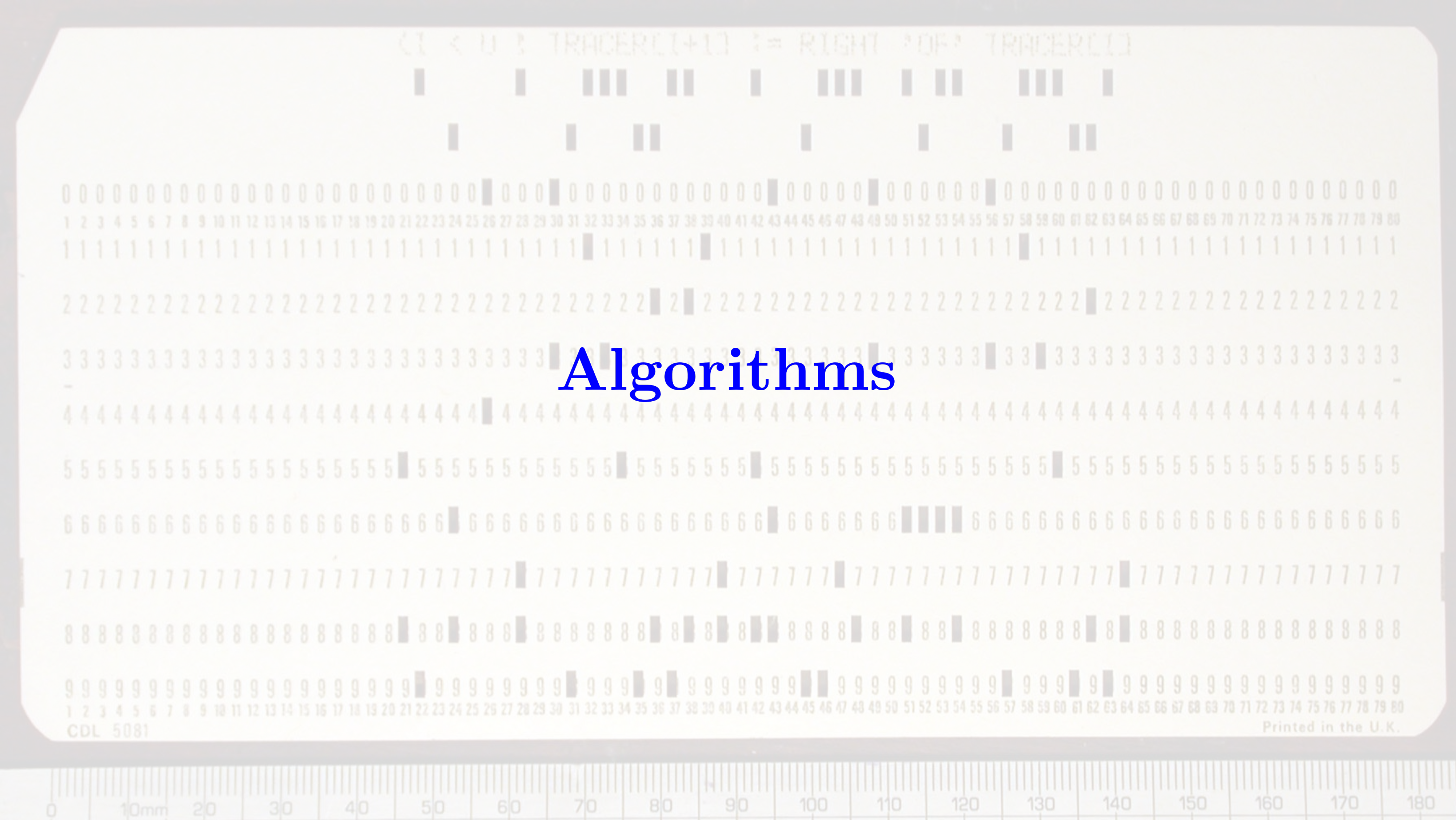


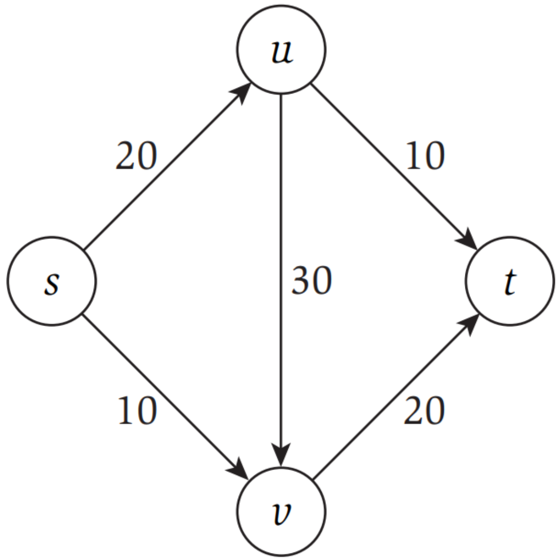
Algorithms

Reading

Sections 7.1-7.3, 7.5 of the textbook.

Suggested. As many exercises as you can do from Chapter 7 of the textbook.

Network Flows

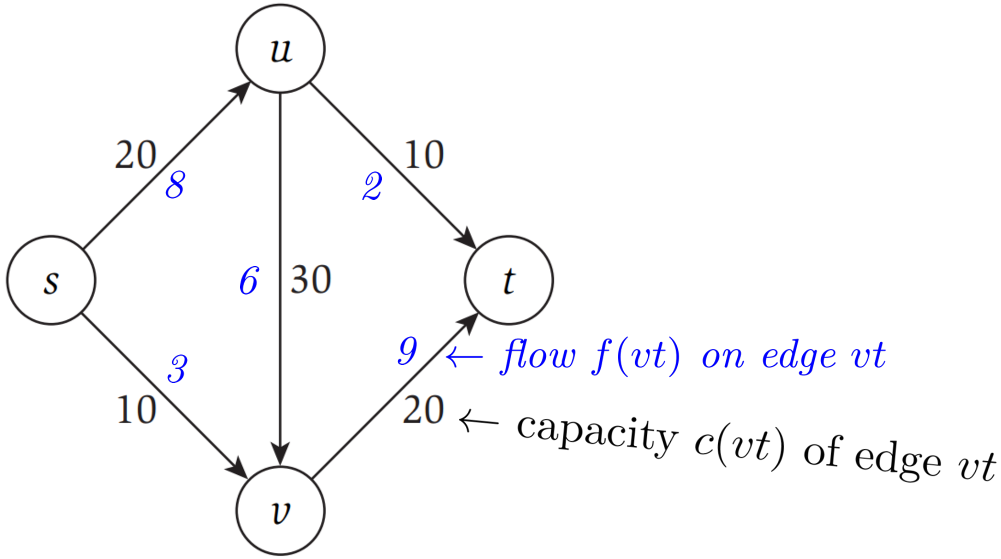


We are given a directed graph $G = (V, E)$ with a *source* node s, a *sink* node t and *capacities* on the edges.

Goal. Find the maximum "flow" from s to t without violating the capacity constraints.

Intuition: Material flowing through a transportation network; material originates at source and is sent to sink.

Network Flows

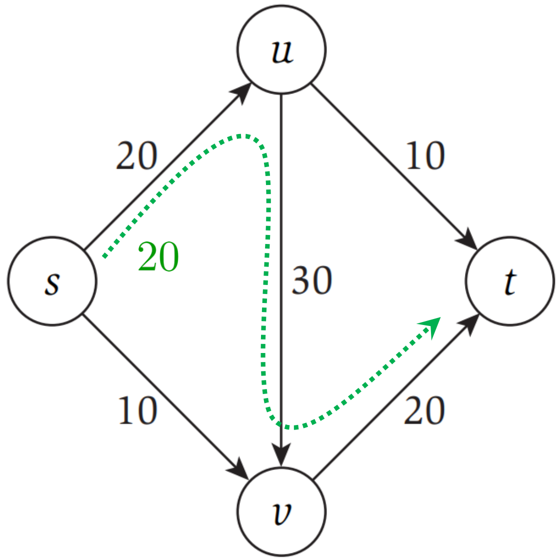


A **flow** is described by a function $f : E \mapsto \mathbb{R}^+$ s.t.

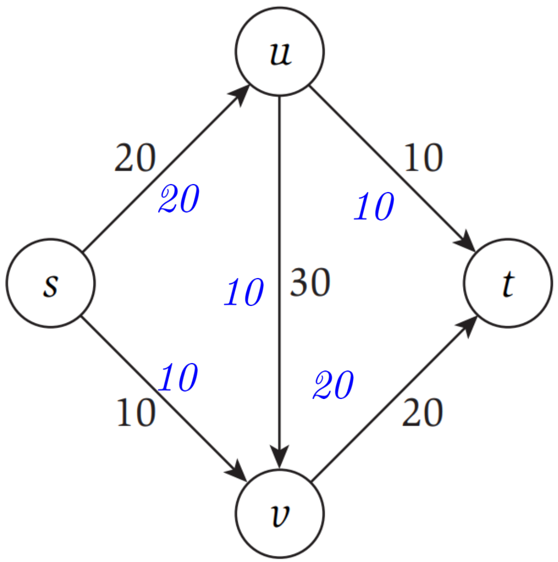
- (*Capacity constraints*) $0 \leq f(e) \leq c(e)$ ← *capacity of edge e*
- (*Flow conservation*) $\forall v \in V \setminus \{s, t\}, \quad \sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e).$

The **value** of a flow f is: $\nu(f) = \sum_{e \text{ out of } s} f(e) = \sum_{e \text{ into } t} f(e).$

Network Flows



v(f)=20 *send 20 units of flow along $s \to u \to v \to t$*



optimal flow of 30 units

Is this an optimal solution?

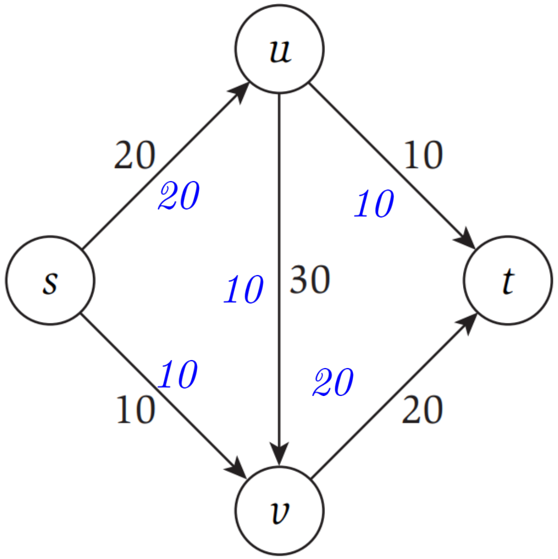
Goal. Find the maximum "flow" from s to t without violating the capacity constraints.

Would simple greedy approaches work here?

Push the maximum you can on the highest capacity edge

May not produce an optimal solution.

Network Flows



optimal flow of 30 *units*

v(f)=30

Goal. Find the maximum "flow" from s to t without violating the capacity constraints.

How can we formalize what we just did?

FORD Fulkerson algorithm

Network Flows

Residual Graph.

Given a flow network G, and a flow f on G, we define the residual Graph G_f of G with respect to f as follows:

- The node set of G_f is the same as that of G
- For each edge $e = (u, v)$ of G on which $f(e) < c_e$, there are $c_e - f(e)$ "leftover" units of capacity on which we could try pushing flow forward.

  So we include the edge $e = (u, v)$ in G_f, with a capacity of $c_e - f(e)$. We will call edges included this way forward edges.
- For each edge $e = (u, v)$ of G on which $f(e) > 0$, there are $f(e)$ units of flow that we can "undo" if we want to, by pushing flow backward. So we include the edge $e' = (v, u)$ in G_f, with a capacity of $f(e)$. Note that e' has the same ends as e, but its direction is reversed. We will call edges included this way backward edges.

Network Flows

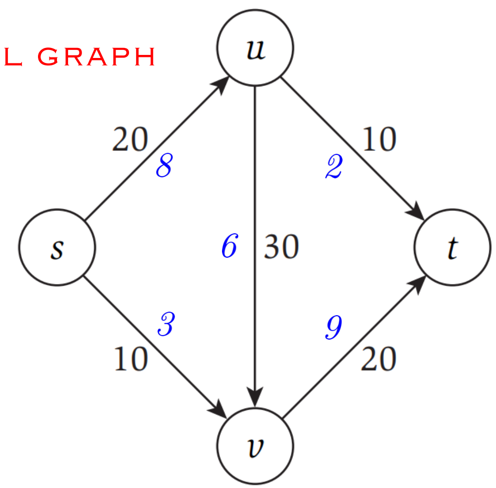
Residual Graph.

In residual graph corresponding to a flow f, for every edge $u \to v$ in the original network, we have:

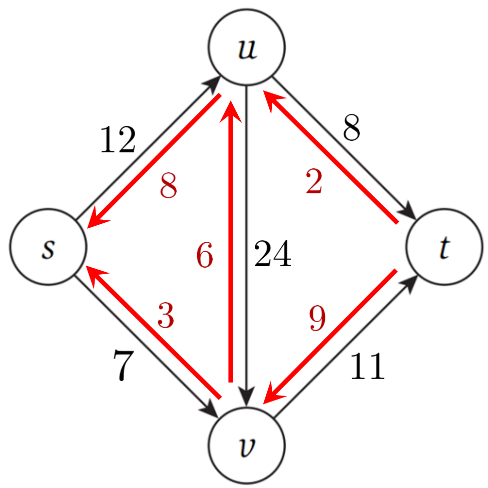
- an edge $u \to v$ with capacity $c(e) - f(e)$
- an edge $v \to u$ with capacity $f(e)$

Flow f

Construct residual graph



Residual Graph

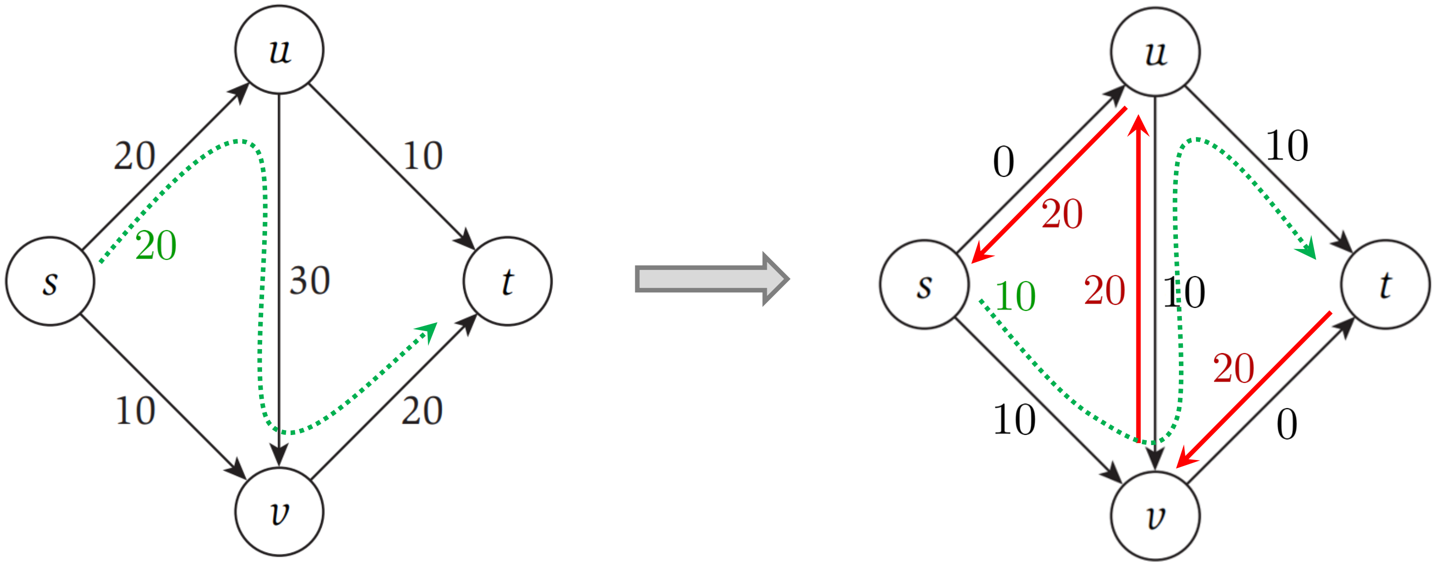


Network Flows

Ford Fulkerson Algorithm.

Repeatedly find a path in G_f from s to t in which all edges have some leftover capacity and send flow equal to the capacity of the *bottleneck edge*.

Residual Graph



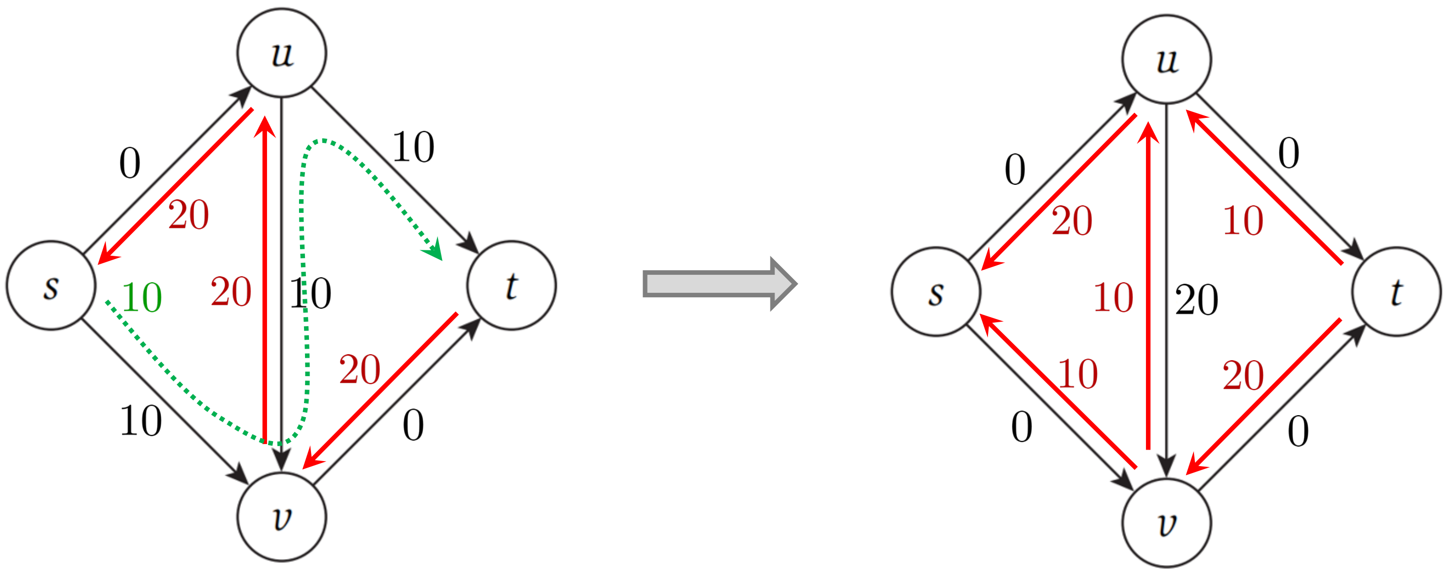
send 20 units of flow along $s \to u \to v \to t$

send 10 units of flow along $s \to v \to u \to t$

Network Flows

Ford Fulkerson Algorithm.

Repeatedly find a path in G_f from s to t in which all edges have some leftover capacity and send flow equal to the capacity of the *bottleneck edge*.



send 10 units of flow along $s \to v \to u \to t$

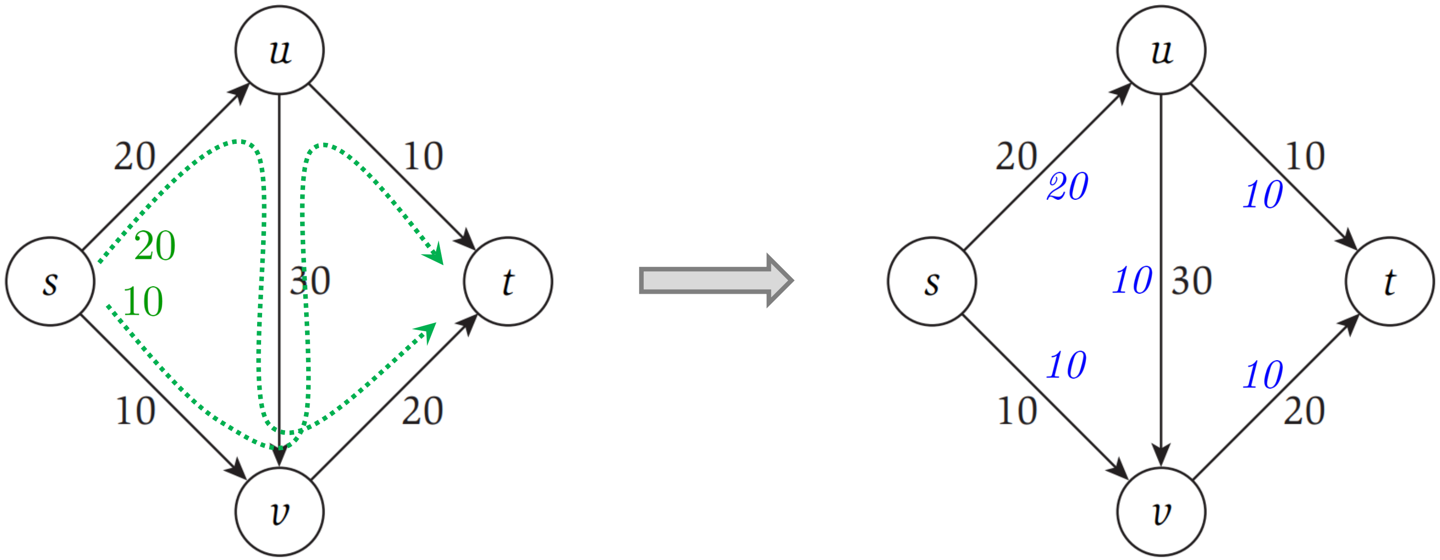
No more $s \rightsquigarrow t$ paths of non-zero capacity!

Total flow: 30 units.

Network Flows

Ford Fulkerson Algorithm.

Repeatedly find a path in G_f from s to t in which all edges have some leftover capacity and send flow equal to the capacity of the *bottleneck edge*.



The paths along which we sent flows are called *augmenting paths*.

EXERCISE

Network Flows

Loops over paths in the resudual graph

```
Max-Flow
    Initially f(e) = 0 for all e in G
    While there is an s-t path in the residual graph G_f
        Let P be a simple s-t path in G_f
        f' = augment(f, P)
        Update f to f'
        Update the residual grapgh G_f to be G_f'
    EndWhile
    Return f
```

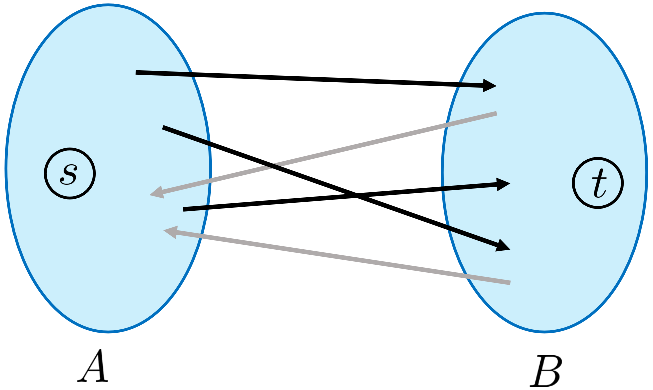
Uses the path to form the flow

```
augment(f, P):
    Let b = bottleneck(P, f)
    For each edge (u,v) P
        If e = (u, v) is a forward edge then
            increase f(e) in G by b
        Else ((u, v) is a backward edge, and let e = (u,v))
            decrease f(e) in G by b
        Endif
    Endfor
    Return(f)
```

Observation

Network Flows

An **s-t cut** in the graph is a partition (A, B) of the vertices into two sets A and B s.t. $s \in A$ and $t \in B$.



The capacity of the s-t cut, denoted $c(A, B)$ is the sum of the capacities of the edges going *from A to B.*

Observation. No flow can have value more than the capacity of an s-t cut.