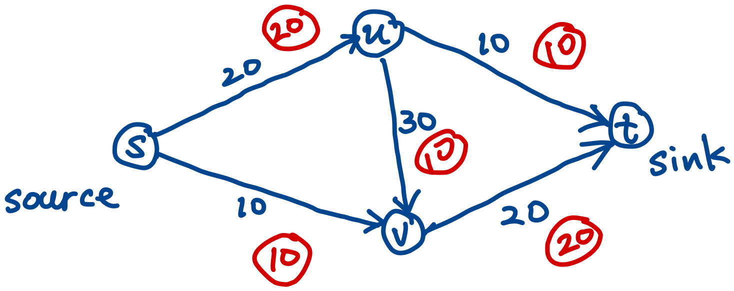
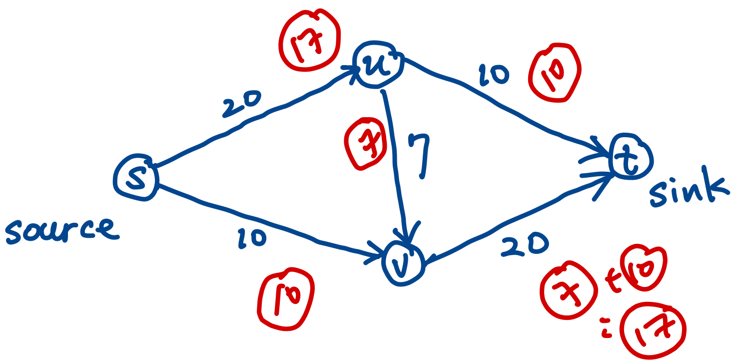
Network Flow [Chapter 7]

Example: weighted directed graph, weight = capacity on edge



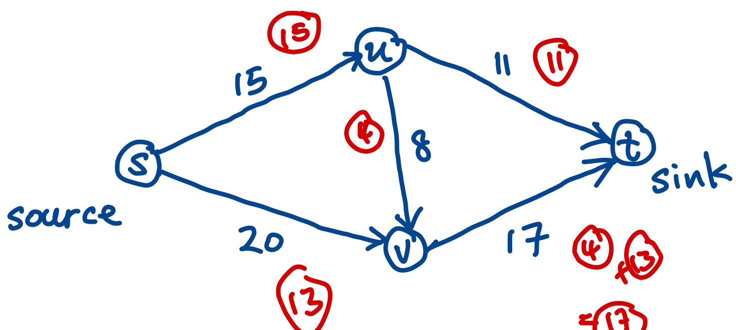
max flow value ?

30

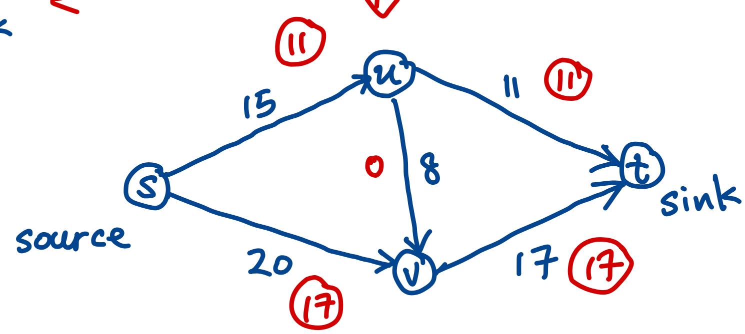


max flow value ?

27



flow value = 28



Defining <u>Flow network</u>: Directed graph $G = (V, E)$ and

- each edge e has a capacity $c(e) > 0$.
- a source node $s \in V$
- a sink node $t \in V$.

(nodes other than s and t are called internal nodes.)

Defining <u>Flow</u>: a function $f: E \to \mathbb{R}^{\geq 0}$ ($f(e)$ is the "flow" on edge e) that satifies the following:

a) Capacity conditions: for every edge, $0 \leq f(e) \leq c(e)$.

b) Conservation Conditions: for every internal node $v \neq s, t$.

$$\sum_{\substack{\text{edge } e \\ \text{come into } v}} f(e) = \sum_{\substack{\text{edge } e \\ \text{go out of } v}} f(e)$$

i.e. Flow in = Flow out.

Defining <u>Value of a flow f</u>: Given a flow f, it value

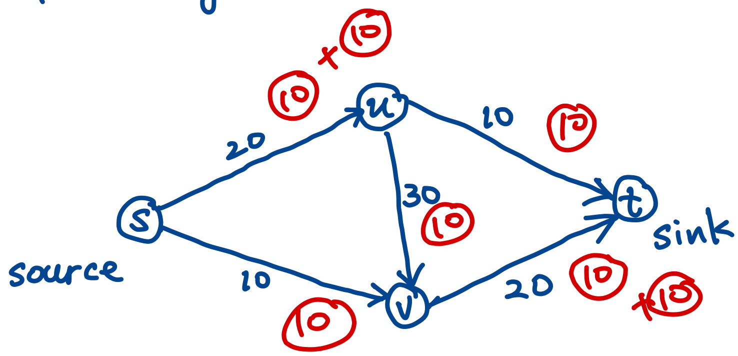
$$\text{val}(f) = \sum_{e \text{ out of } s} f(e) = \sum_{e \text{ into } t} f(e).$$

<u>Maximum-Flow Problem</u>

Given a flow network, find a flow of maximum possible value.

Algorithm for max-flow (Ford-Fulkerson algorithm)

Keep looking for s-t path and push as much flow as possible on that edge.



Path from s to t.

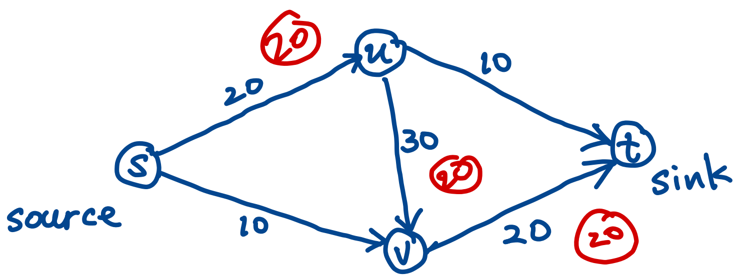
① s, u, t (10)

② s, v, t (10)

③ s, u, v, t (10)

(30)

Don't get stuck...



Path from s to t.

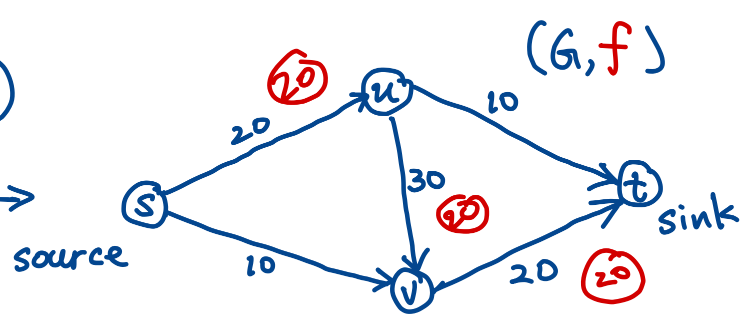
① s, u, v, t (20)

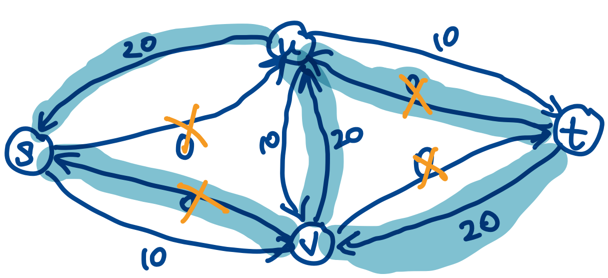
Get stuck after 1 path.

FF pushes flows along s-t paths, but allow us to "fix our mistakes".

Defining Residual Graph (G, f)

Residual graph for this (G, f) ⟶

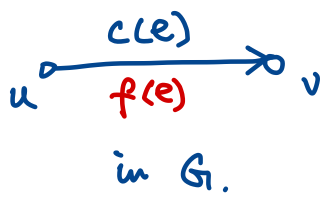




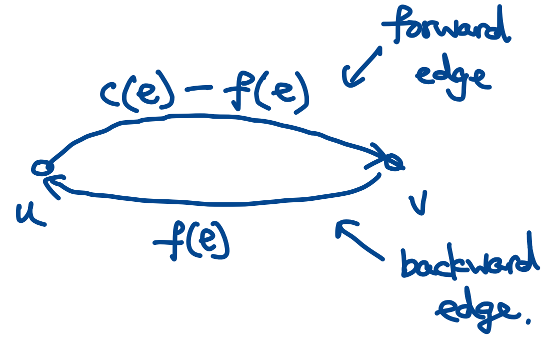
• for each edge, now we make a backward edge, which allow us to "fix our mistake" (later).

* Consider 0-weight edges in R(G, f) as nonexistant.

To build Residual Graph :



in residual graph



FF finds paths in residual graph & push flow on them.

Exercises: build residual graphs on the examples we did at beginning.

Ford-Fulkerson Algorithm

Initially $f(e) = 0$ for edges.

While there is an s-t path in Residual(G, f)

Let p be such a simple (no cycle) path.

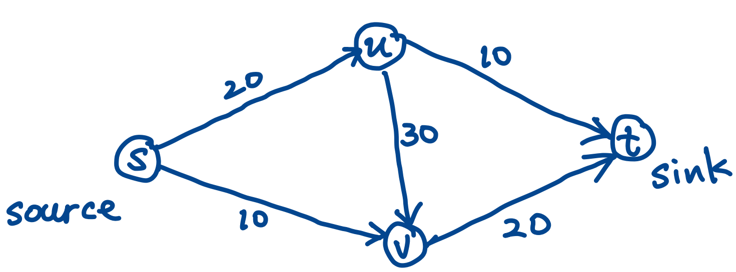
$f' = \text{augment}(f, p)$ ["push" flow along p]

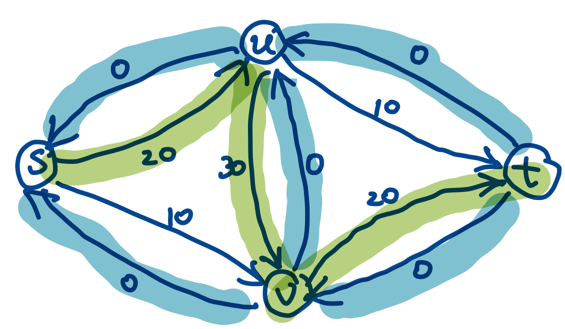
Update f to f'

Update $R(G, f)$ to $R(G, f')$

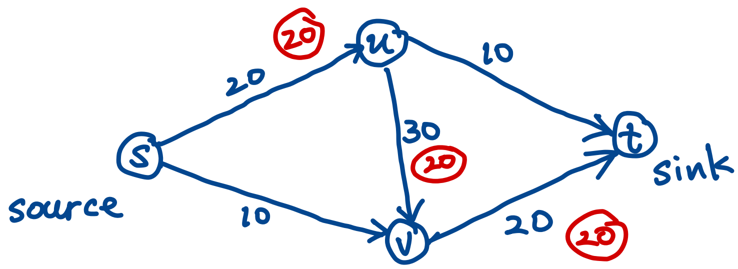
endwhile.

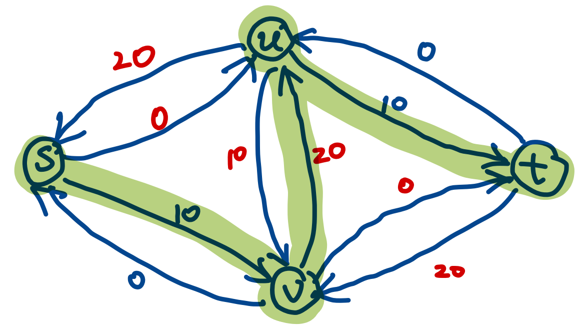
Example :



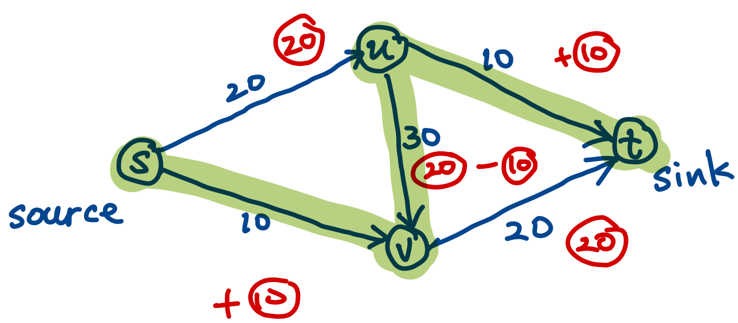


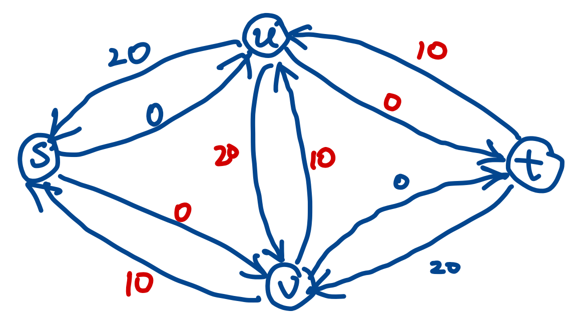
- find a s-t path : s,u,v,t.





- find a s-t path : s, v, u, t





- find a s-t path : None