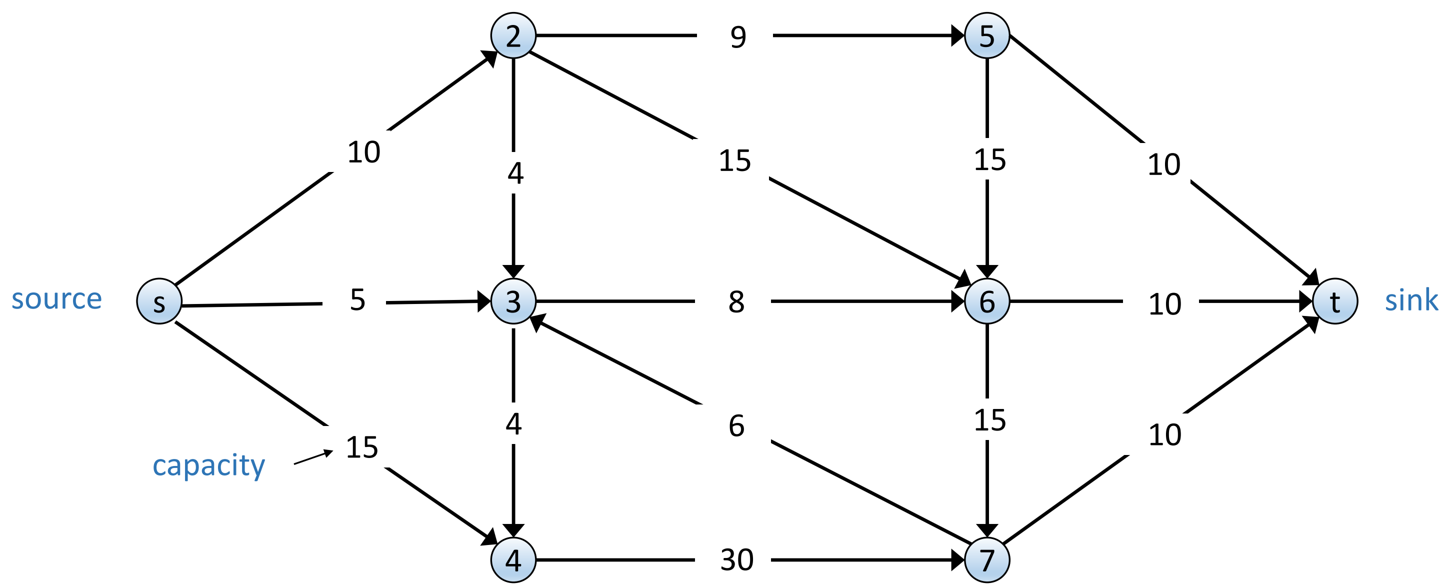
CS3000: Algorithms & Data
Jonathan Ullman

Lecture 18:
- Network Flow: choosing good paths

Nov 9, 2018

Flow Networks

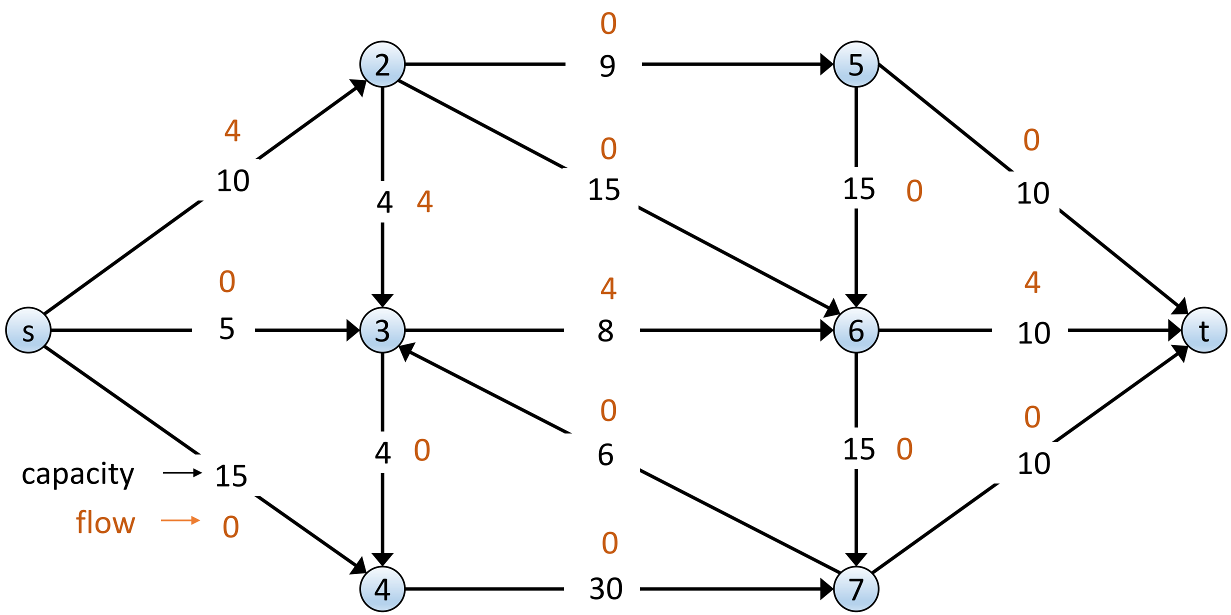
- Directed graph $G = (V, E)$
- Two special nodes: source s and sink t
- Edge capacities $c(e)$



Flows

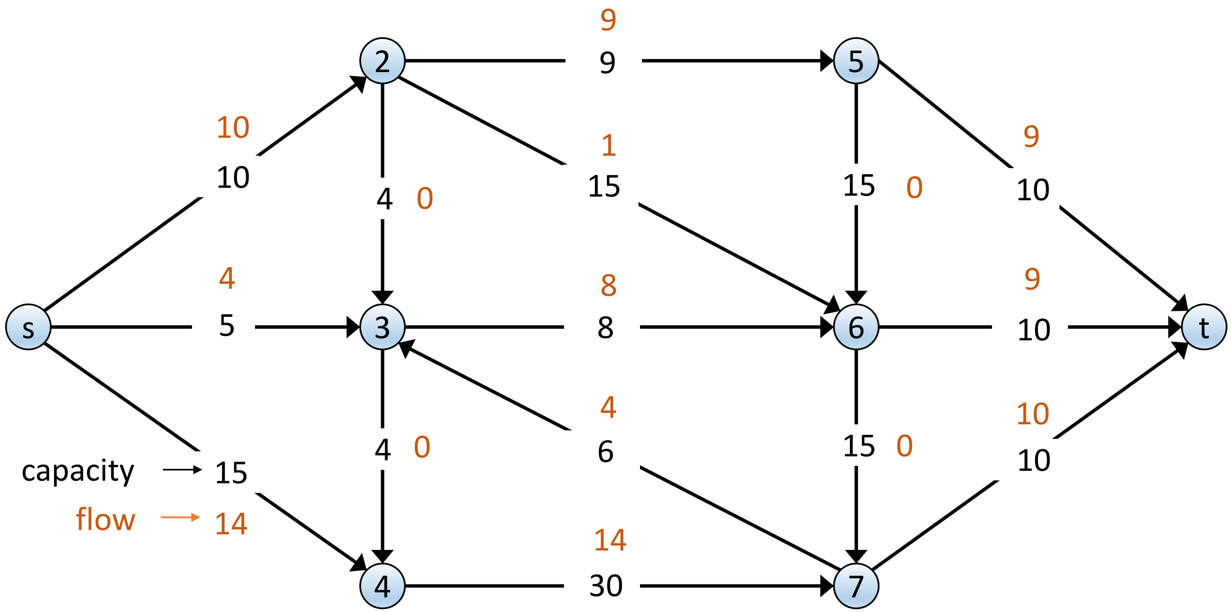
- An s-t flow is a function $f(e)$ such that
  - For every $e \in E$, $0 \leq f(e) \leq c(e)$ (capacity)
  - For every $v \in E$, $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

- The value of a flow is $val(f) = \sum_{e \text{ out of } s} f(e)$



Maximum Flow Problem

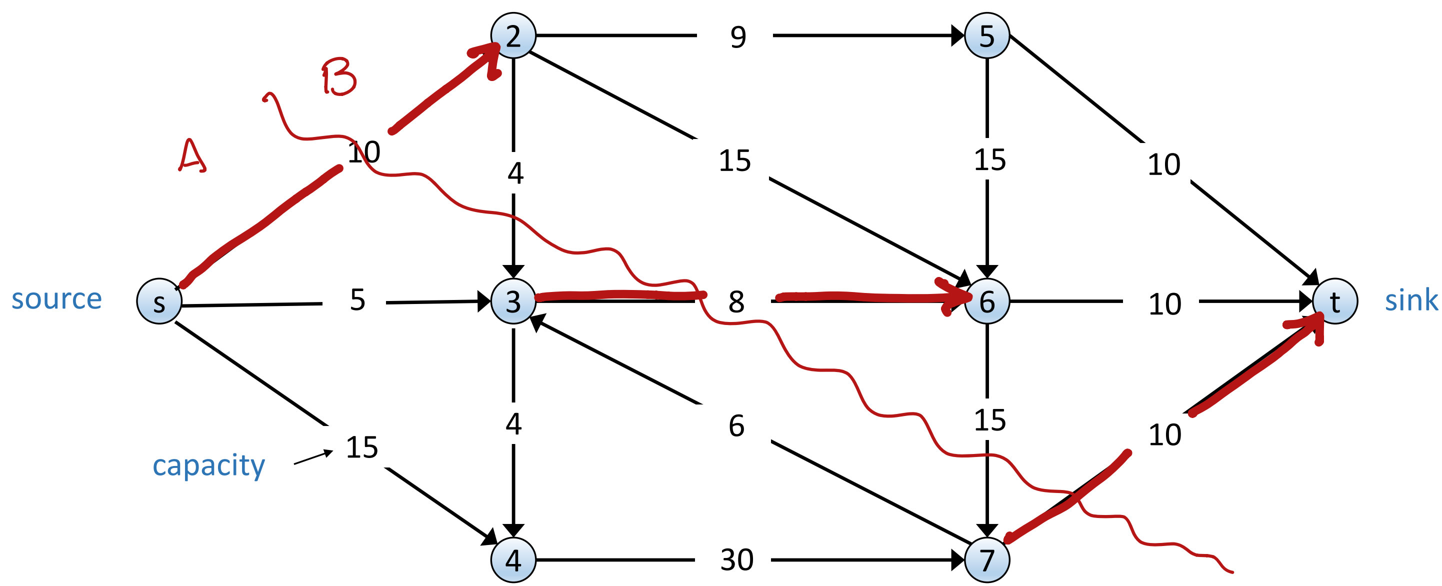
- Given G = (V,E,s,t,{c(e)}), find an s-t flow of maximum value



Cuts

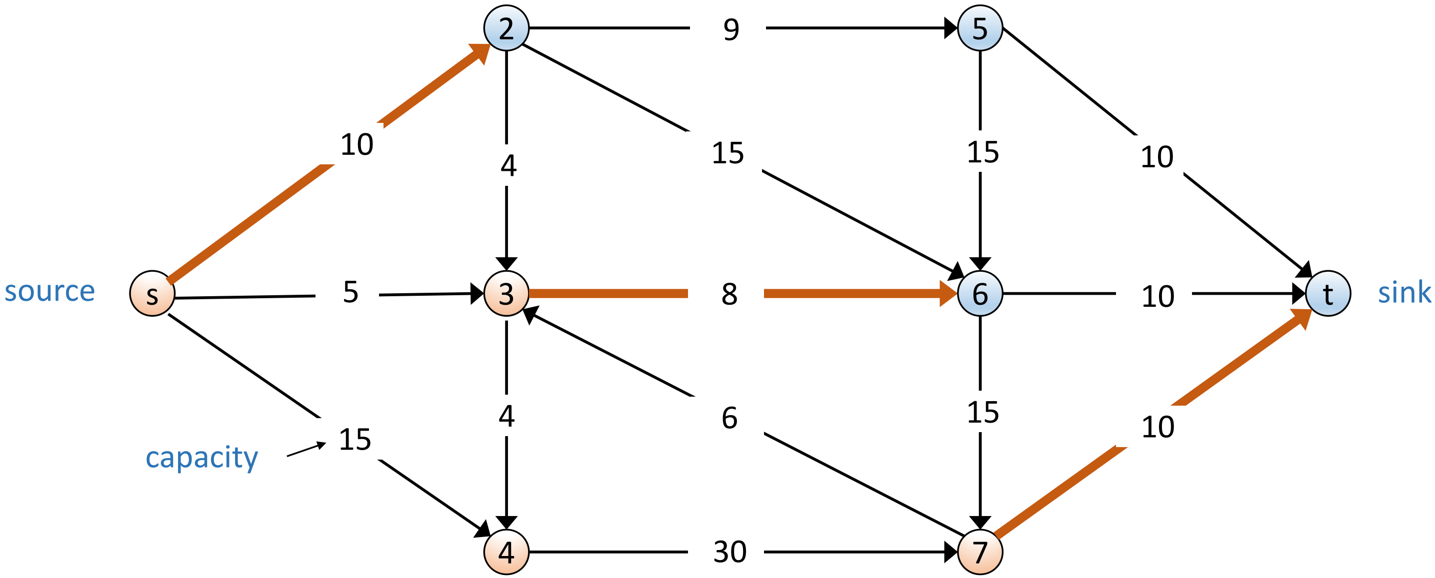
- An s-t cut is a partition (A, B) of V with $s \in A$ and $t \in B$

- The capacity of a cut (A,B) is $cap(A, B) = \sum_{e \text{ out of } A} c(e)$



Minimum Cut problem

- Given G = (V,E,s,t,{c(e)}), find an s-t cut of minimum capacity

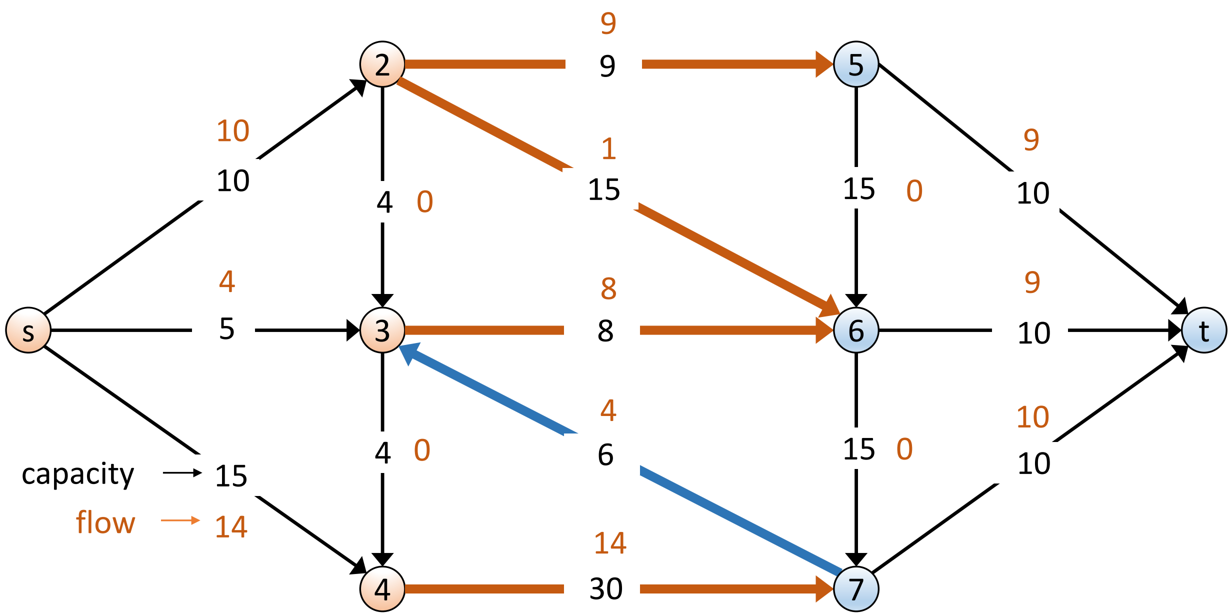


Flows vs. Cuts

$\nearrow\!\!\!\nearrow$ max flow ≤ min cut

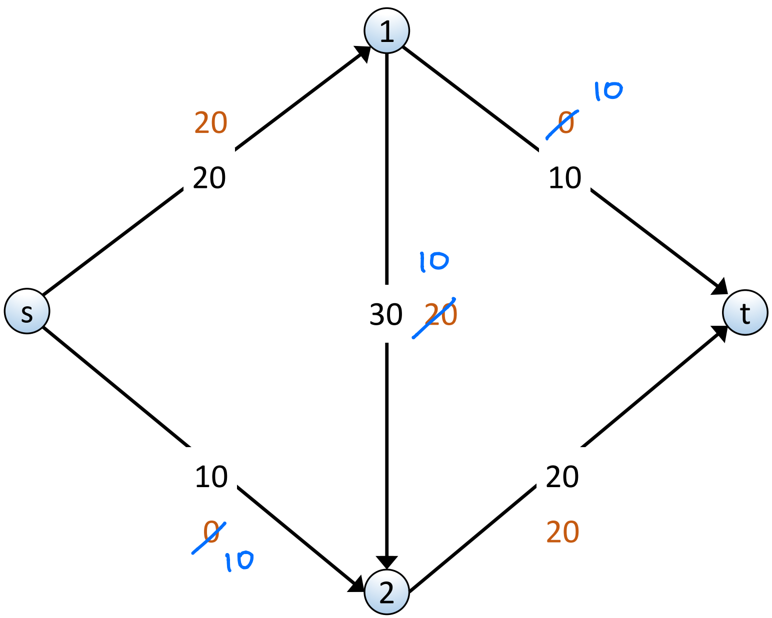
- **Fact:** If f is any s-t flow and (A, B) is any s-t cut, then the net flow across (A, B) is equal to the amount leaving s

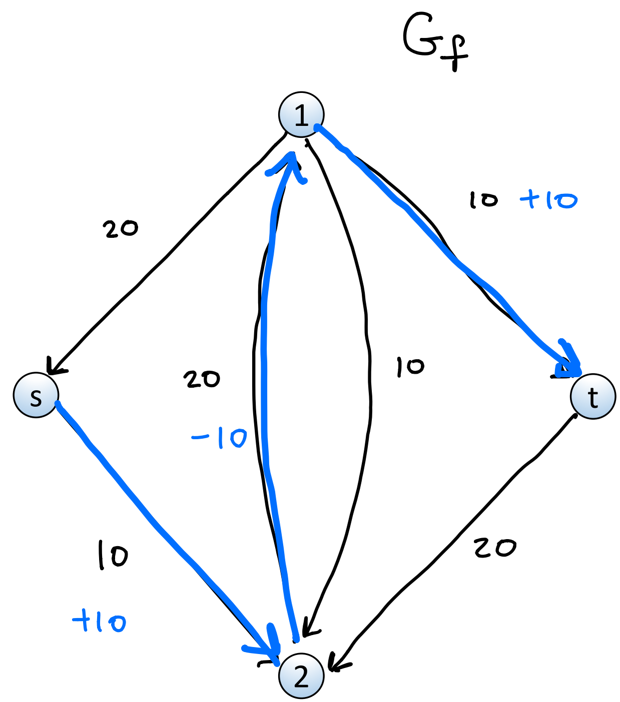
$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = val(f)$$



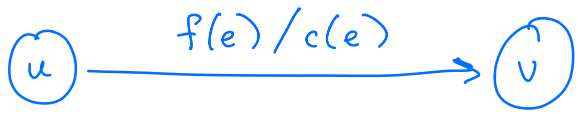
Ford-Fulkerson Algorithm

- Start with $f(e) = 0$ for all edges $e \in E$
- Find an **augmenting path** P in the **residual graph**
- Repeat until you get stuck

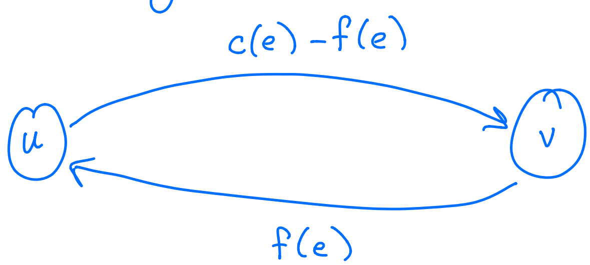




original graph



residual graph



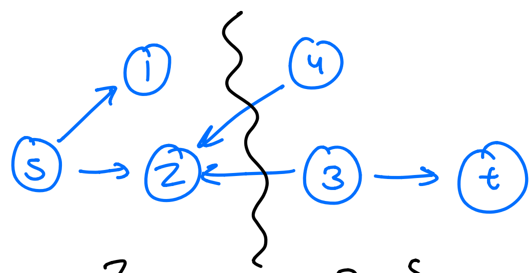
(remove edges of capacity 0)

Summary

when FF is "stuck", f is a max flow

- The **Ford-Fulkerson Algorithm** solves maximum s-t flow
  - Running time is $O(m)$ per augmentation step
  - $O(val(f^*))$ augmentinations in any graph with integer capacities

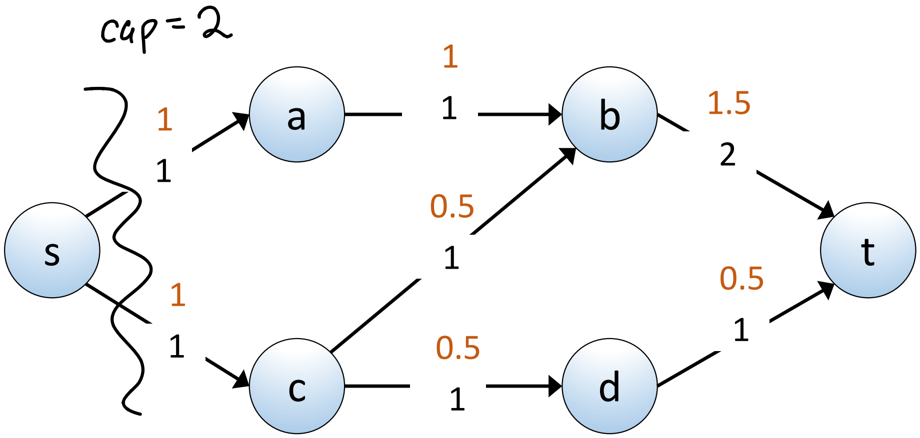
- **MaxFlow-MinCut Theorem:** The value of the max s-t flow equals the capacity of the min s-t cut
  - If f^* is a max flow, the nodes reachable from s in G_{f^*} are a min cut
  - Given a max flow, can find a min cut in time $O(n+m)$ via BFS



$A = \{s, 1, 2\}$ $B = \{3, 4, t\}$

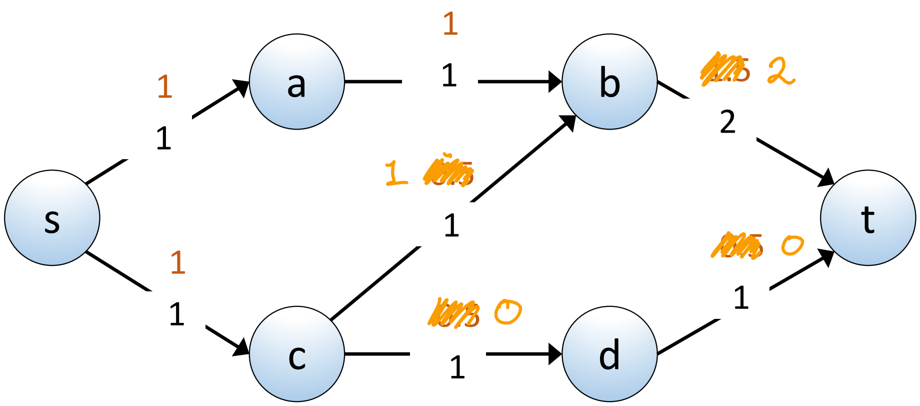
Ask the Audience

- Is this a maximum flow? Yes



Ask the Audience

- Is this a maximum flow?

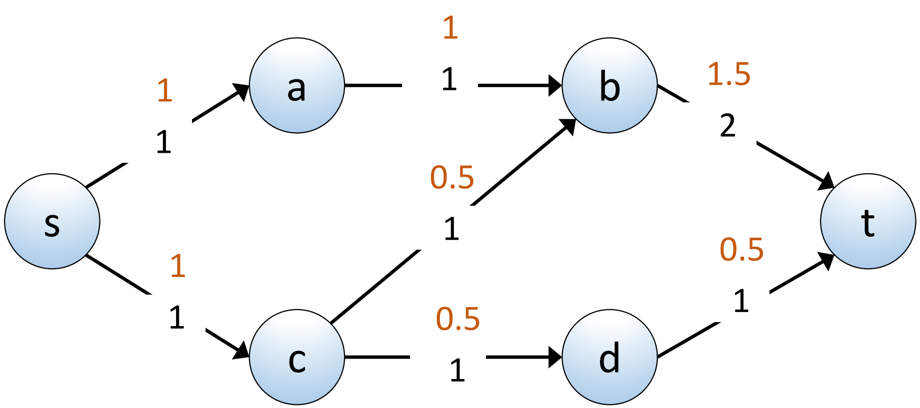


- Is there an integer maximum flow?

(A max flow where $f(e) \in \mathbb{Z}$ for every $e \in E$)

Ask the Audience

- Is this a maximum flow?



- Is there an integer maximum flow?
- Does every graph with integer capacities have an integer maximum flow?

Yes! And Ford-Fulkerson finds one.

Summary

- The **Ford-Fulkerson Algorithm** solves maximum s-t flow
  - Running time is $O(m)$ per augmentation step
  - $O(val(f^*))$ augmentinations in any graph with integer capacities

- **MaxFlow-MinCut Theorem:** The value of the max s-t flow equals the capacity of the min s-t cut
  - If f^* is a max flow, the nodes reachable from s in G_{f^*} are a min cut
  - Given a max flow, can find a min cut in time $O(n + m)$ via BFS

- Every graph with integer capacities has an integer max flow
  - And Ford-Fulkerson finds an integer max flow

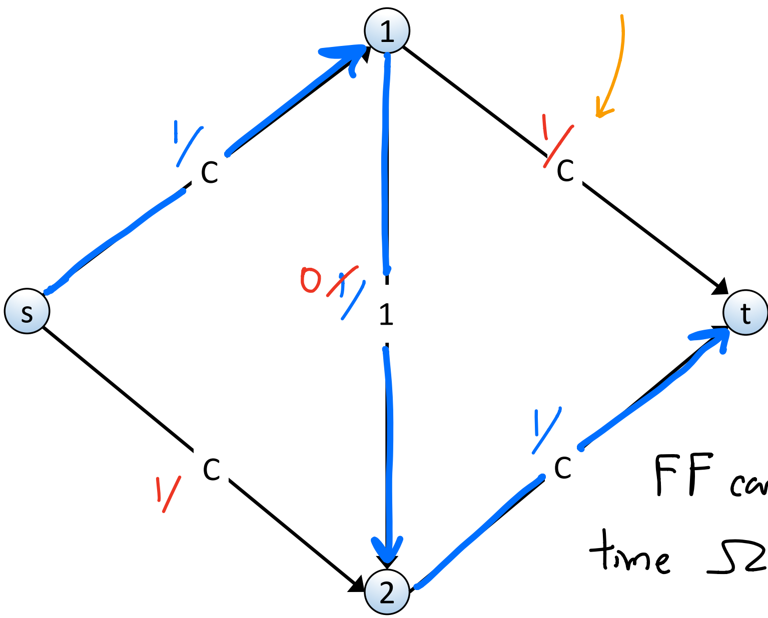
Ford-Fulkerson Algorithm

- Start with $f(e) = 0$ for all edges $e \in E$
- Find an **augmenting path** P in the **residual graph**
- Repeat until you get stuck

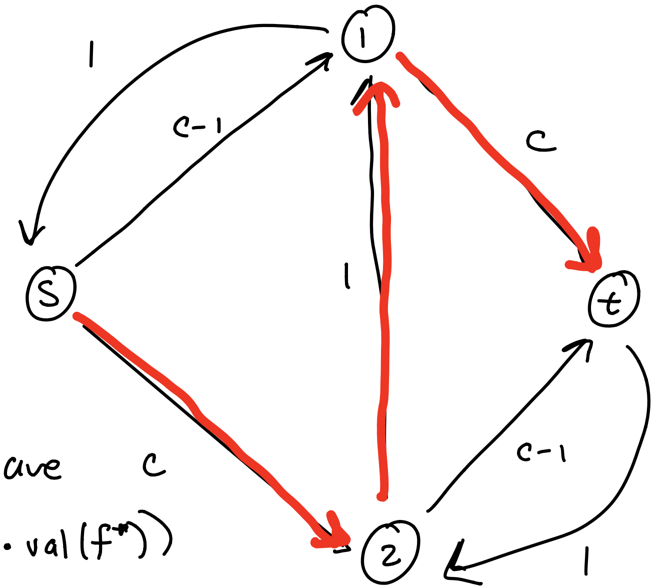
Might take 2C augmenting paths

val(f*) = 2C

C is a really big #



FF can have time $\Omega(m \cdot val(f^*))$



Choosing Good Augmenting Paths

- **Last time:** arbitrary augmenting paths
  - If FF terminates, it outputs a maximum flow
  - Might not terminate, or might require many augmentations

- **Today:** clever augmenting paths
  - Maximum-capacity augmenting path (“fattest augmenting path”)
  - Shortest augmenting paths (“shortest augmenting path”)

Fattest Augmenting Path

Fattest Augmenting Path

- Maximum-capacity augmenting path

$$P^* = \underset{\substack{s\text{-}t \text{ paths } P \\ \text{in } G_f}}{\operatorname{argmax}} \quad \text{bottleneck capacity}(P)$$

- Can find the fattest augmenting path in time $O(m \log n)$ in several different ways
  - Variants of Prim's or Kruskal's MST algorithms
  - BFS + binary search

- Not too much slower than choosing an arbitrary path

Fattest Augmenting Path

Arbitrary Paths

- **Assume integer capacities**
- Value of maxflow: v^*
- Value of aug path: ≥ 1
- Flow remaining in G_f: $\leq v^* - 1$
- $\boxed{\text{\# of aug paths: } \leq v^*}$

$v^* - k \geq 0$

$k \leq v^*$

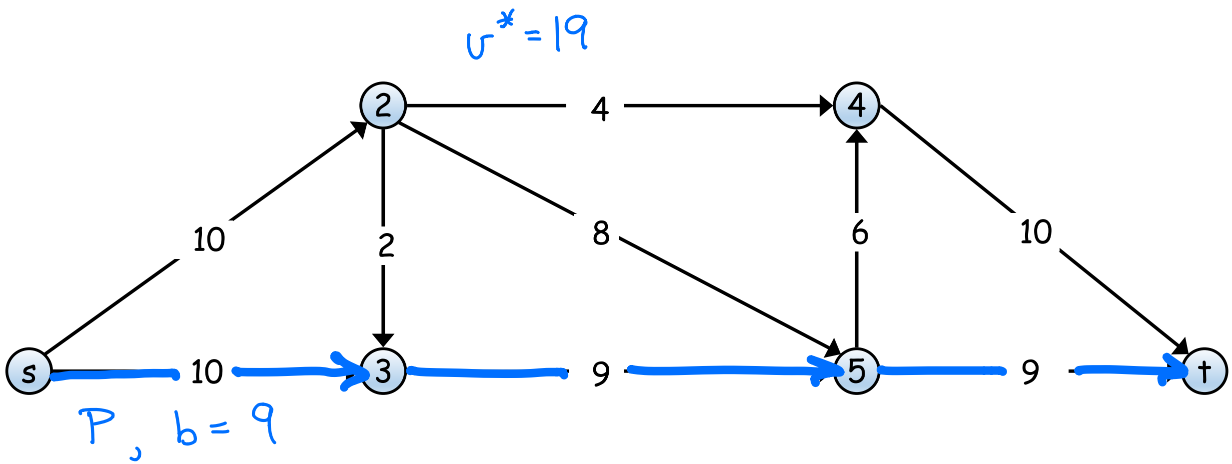
Maximum-Capacity Path

- **Assume integer capacities**
- Value of maxflow: v^*
- Value of aug path:
- Flow remaining in G_f:
- # of aug paths:

Fattest Augmenting Path

- f^* is a maximum flow with value $v^* = val(f^*)$
- P is a fattest augmenting s-t path with capacity b
- **Key Claim:** $b \geq \frac{v^*}{m}$

"capacity of the fattest path $\geq \frac{\text{max flow}}{\text{\# of edges}}$"

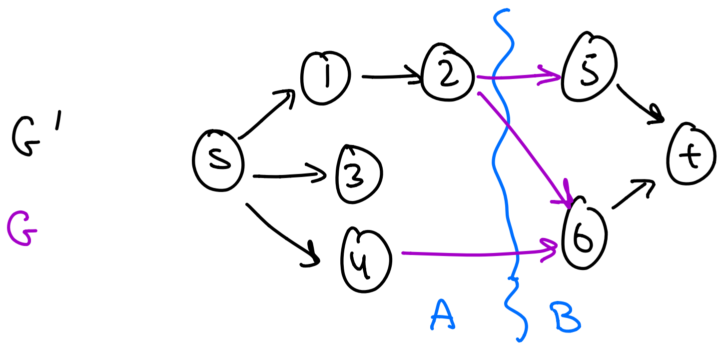


Fattest Augmenting Path

- f^* is a maximum flow with value $v^* = val(f^*)$
- P is a fattest augmenting s-t path with capacity b
- **Key Claim:** $b \geq \frac{v^*}{m}$
- **Proof:**
  - $\nexists$ a path of capacity $b+1$
  - Let G' be G but only with edges s.t. $c(e) \geq b+1$
  - G' doesn't contain any s-t path

$$v^* \leq cap(A,B) \leq b \cdot m$$

$$\frac{v^*}{m} \leq b$$



$A = \{$ nodes reachable from s in $G' \}$

$$cap(A,B) = \sum_{e \text{ out of } A} c(e) \leq b \cdot (\# \text{ of } e \text{ out of } A) \leq b \cdot m$$

Fattest Augmenting Path

- f^* is a maximum flow with value $v^* = val(f^*)$
- P is a fattest augmenting s-t path with capacity b
- **Key Claim:** $b \geq \frac{v^*}{m}$

Fattest Augmenting Path

Arbitrary Paths

- **Assume integer capacities**
- Value of maxflow: v^*
- Value of aug path: ≥ 1
- Flow remaining in G_f: $\leq v^* - 1$
- # of aug paths: $\leq v^*$

Maximum-Capacity Path

- **Assume integer capacities**
- Value of maxflow: v^*
- Value of aug path: $\geq \frac{v^*}{m}$
- Flow remaining in G_f: $\leq \left(1 - \frac{1}{m}\right) v^*$
- # of aug paths:

- If there are k aug. paths then after adding k paths the flow was at least 1.
- Flow remaining after k paths $\leq \left(1 - \frac{1}{m}\right)^k \cdot v^*$

- $$\left(1-\frac{1}{m}\right)^{k} \cdot v^{*} \geq 1$$

$$\left[\left(1-\frac{1}{m}\right)^{m}\right]^{\frac{k}{m}} \cdot v^{*} \geq 1$$

$$\left(e^{-1}\right)^{\frac{k}{m}} \cdot v^{*} \geq 1$$

$$e^{\frac{-k}{m}} \cdot v^{*} \geq 1$$

$$\frac{-k}{m} + \ln\left(v^{*}\right) \geq 0$$

$$\boxed{k \leq m \cdot \ln\left(v^{*}\right)} \implies \# \text{ of } \underline{\text{paths} \leq m \cdot \ln\left(v^{*}\right)+1}$$

Choosing Good Paths

- **Last time:** arbitrary augmenting paths
  - If FF terminates, it outputs a maximum flow
  - Bad paths ⇒ FF never terminates
- **Today:** clever augmenting paths
  - Maximum-capacity augmenting path ("fattest augmenting path")
    - $\leq m \ln v^*$ augmenting paths (assuming integer capacities)
    - $O(m^2 \ln n \ln v^*)$ total running time
    - See KT for a slightly faster variant ("fat-ish augmenting path"?)

$\min\{m^2 \ln(n) \ln(v^*), mv^*\}$

  - Shortest augmenting paths ("shortest augmenting path")
    - # of augmenting paths $\leq \frac{mn}{2}$ for <u>any</u> capacities
    - $O(m^2 n)$ total running

Shortest Augmenting Path

Shortest Augmenting Path

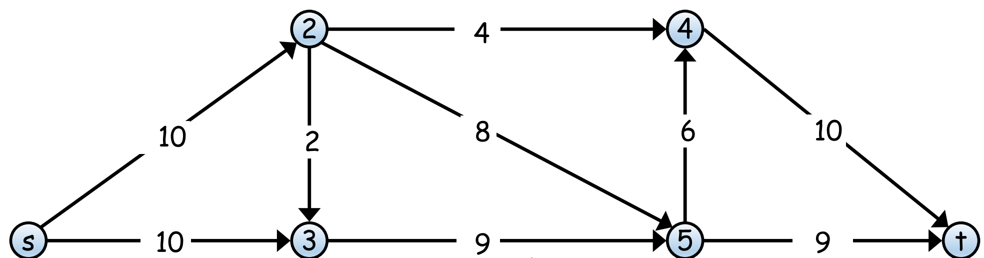
- Find the augmenting path with the fewest hops
  - Can find shortest augmenting path in $O(m)$ time using BFS

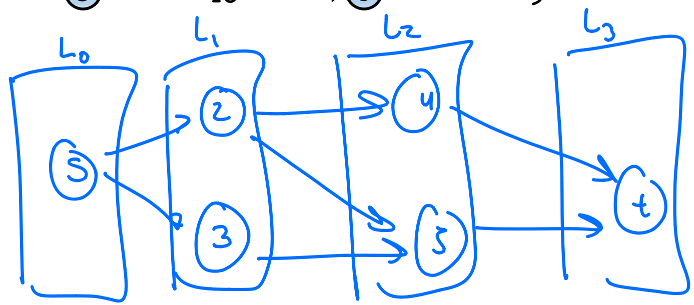
- **Theorem:** for any capacities $\frac{nm}{2}$ augmentations suffice
  - Overall running time $O(m^2 n)$
  - Works for any capacities!

- **Warning:** proof is challenging (you will not be tested on it)

Shortest Augmenting Path

- Let f_i be the flow after the i-th augmenting path
- Let $G_i = G_{f_i}$ be the i-th residual graph
- Let $L_i(v)$ be the distance from s to v in G_i
  - Recall that the shortest path in G_i moves layer-by-layer





Shortest Augmenting Path

- Every augmentation causes at least one edge to disappear from the residual graph, may also cause an edge to appear

  - Some edge on the augmenting path G_i is now at capacity, is not in G_{i+1}

- **Key Property:** each edge disappears at most $\frac{n}{2}$ times
  - Means that there are at most $\frac{mn}{2}$ augmentaitons

Shortest Augmenting Path

- **Claim 1:** for every $v \in V$ and every i, $L_{i+1}(v) \geq L_i(v)$
  - Obvious for $v = s$ because $L_i(s) = 0$
  - Suppose for the sake of contradiction that $L_{i+1}(v) < L_i(v)$
    - Let v be the smallest such node
  - Let $s \rightsquigarrow u \rightarrow v$ be a shortest path in G_{i+1}
    - By optimality of the path, $L_{i+1}(v) = L_{i+1}(u) + 1$
    - By assumption, $L_{i+1}(u) \geq L_i(u)$
  - Two Cases:
    - $(u, v) \in G_i$, so $L_i(v) \leq L_i(u) + 1$
    - $(u, v) \notin G_i$, so (v, u) was in the i-th path, so $L_i(v) = L_i(u) - 1$

Shortest Augmenting Path

- **Claim 2:** If an edge $u \to v$ disappears from G_i and reappears in G_{j+1} then $L_j(u) \geq L_i(u) + 2$
  - $u \to v$ is on the i-th augmenting path, $L_i(v) = L_i(u) + 1$
  - $v \to u$ is on the j-th augmenting path, $L_j(u) = L_j(v) + 1$
  - By Claim 1: $L_j(v) \geq L_i(v)$

- **Claim 3:** An edge (u, v) cannot reappear more than $\frac{n}{2}$ times
  - $0 \leq L_i(u) \leq n$
  - By Claim 2: length increases by 2 for each reappearance

Choosing Good Paths

- **Last time:** arbitrary augmenting paths
  - If FF terminates, it outputs a maximum flow

- **Today:** clever augmenting paths
  - Maximum-capacity augmenting path (“fattest augmenting path”)
    - $\leq m \ln v^*$ augmenting paths (assuming integer capacities)
    - $O(m^2 \ln n \ln v^*)$ total running time
    - See KT for a slightly faster variant (“fat-ish augmenting path”?)

  - Shortest augmenting paths (“shortest augmenting path”)
    - $\leq \frac{mn}{2}$ augmenting paths (for any capacities)
    - $O(m^2 n)$ total running time

- State-of-the-Art algorithms have $O(mn)$ time for any capacities