CS7800: Advanced Algorithms

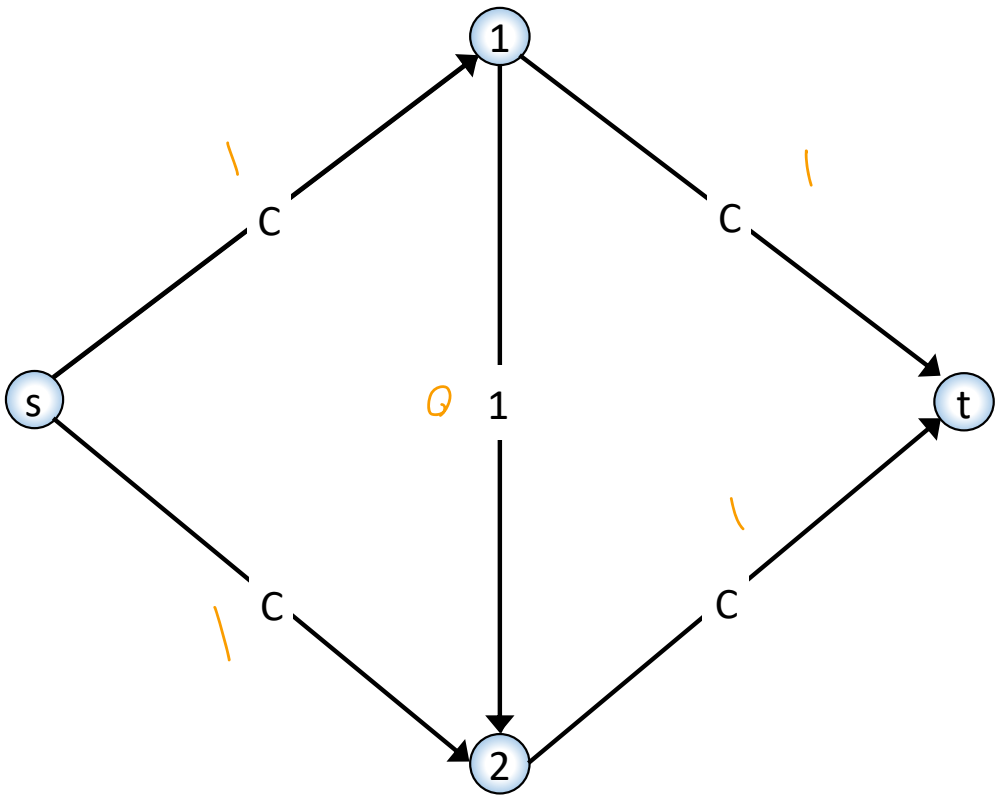
Soheil Behnezhad

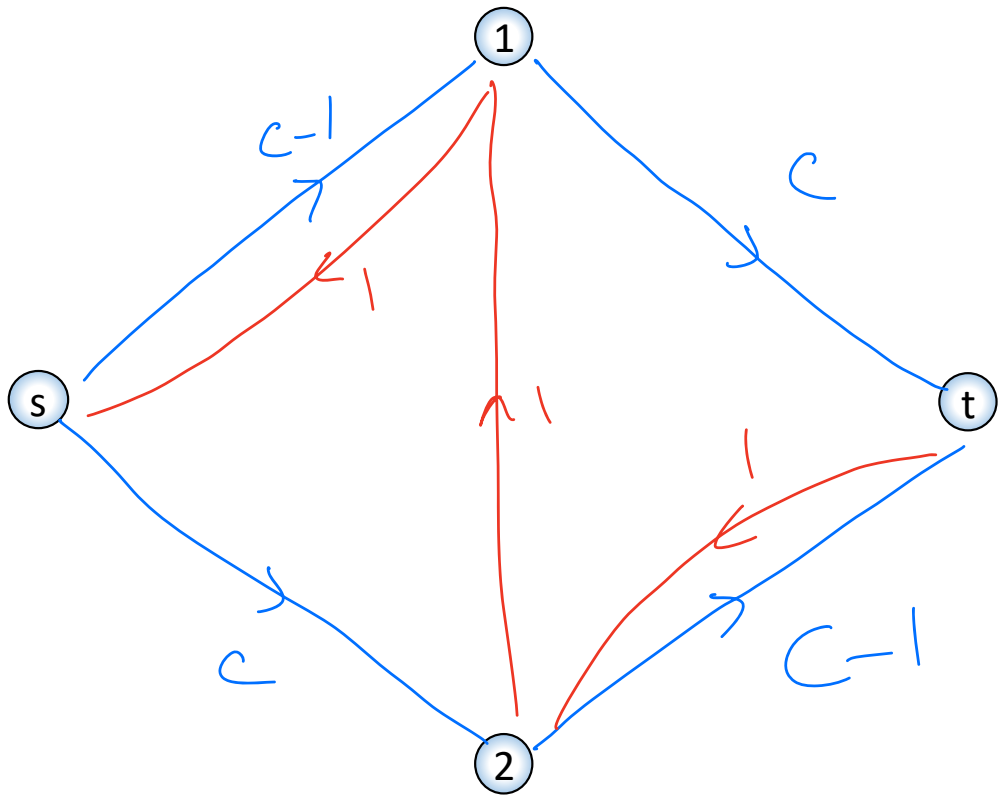
Flow Augmentation

Speeding Up Ford-Fulkerson

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

Naive implementation of FF takes $O(mf^*)$.





Choosing Good Augmenting Paths

- **Last time:** arbitrary augmenting paths
  - If Ford-Fulkerson terminates, then we have found a max flow
  - Can construct capacities where the algorithm never terminates
  - Can require many augmenting paths to terminate

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

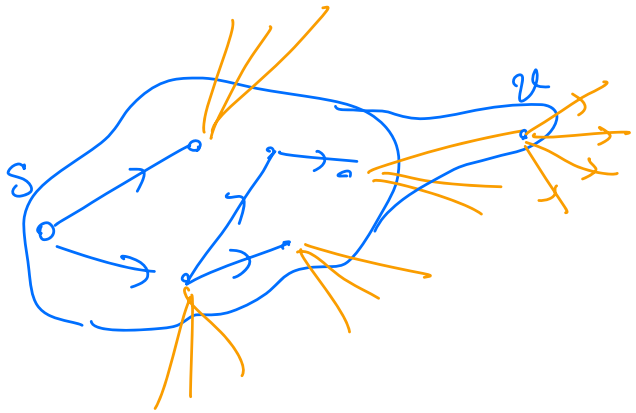
Both modifications are due to Edmonds & Karp from early 70's.

Fattest Augmenting Path

Choose the augmenting path with largest bottleneck value.

- **Claim:** Can find the fattest augmenting path in time $O(m \log n)$

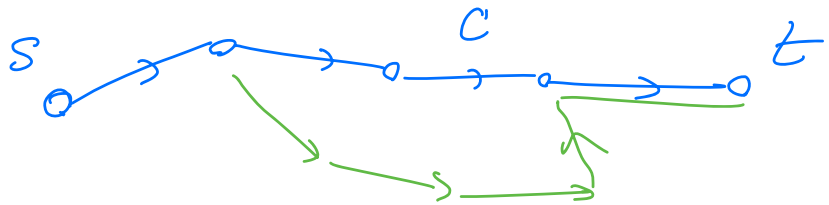
Proof: Can use variants of Prim's or Kruskal's algorithms.



Extracting max: $O(\lg n)$ time

Insert d^+_v many edges in $O(d^+_v \lg n)$ time

Total time $\leq \sum_v d^+_v \cdot \lg n \leq m \lg n$.



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^* (in the residual graph)
- Value of aug path: $\geq v^*/m$
- Flow remaining in G_f: $v^* - \frac{v^*}{m} = (1 - \frac{1}{m})\, v^*$
- # of aug paths:

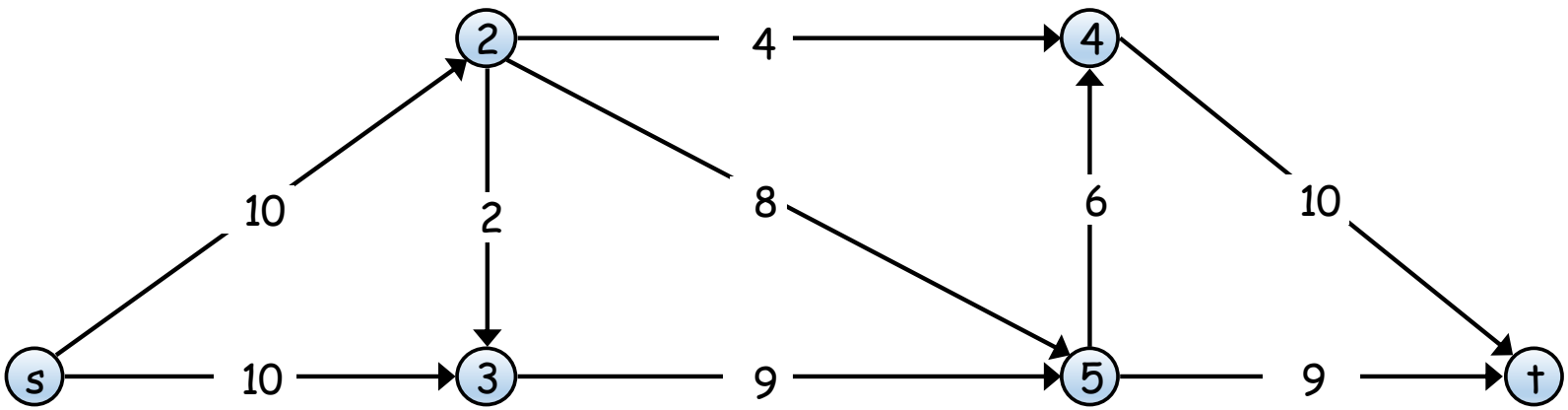
after $O(m \lg f^*)$ augmentations, the flow remaining is

$$f^* \cdot (1 - \frac{1}{m})^{O(m \lg f^*)} \leq f^{*} e^{-O(\lg f^*)} < f^* \cdot \frac{1}{f^*} = 1$$

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:** $B \geq \frac{v^*}{m}$

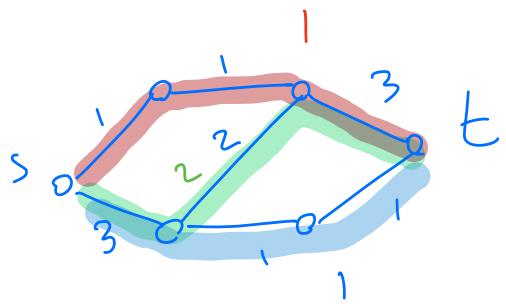
By the flow decomposition theorem, there are at most m $(s-t)$-paths whose flows sum up to v^*. Therefore the largest one has to have capacity $\frac{v^*}{m}$.

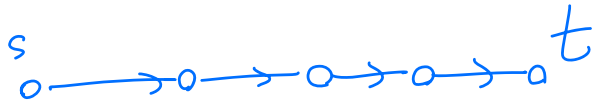


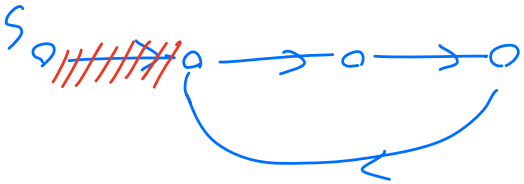
Flow Networks

Flow Decomposition Theorem. *Every non-negative (s,t)-flow f can be written as a positive linear combination of directed (s,t)-paths and directed cycles. Moreover, a directed edge $u \to v$ appears in at least one of these paths or cycles if and only if $f(u \to v) > 0$, and the total number of paths and cycles is at most the number of edges in the network.*

we write $h = a \cdot f + b \cdot f'$ to denote $h(u \to v) = a \cdot f(u \to v) + b \cdot f'(u \to v)$.







Proof: Take the flow f. Iteratively do the following:

Take the source s, and an outgoing edge $(s \to v)$ with positive flow. Continue a walk

$$s \to v \to v_2 \to \dots$$

on edges with positive flow until we revisit a vertex of the path or reach the sink.

Fattest Augmenting Path

Choose the augmenting path with largest bottleneck value.

- **Claim:** Can find the fattest augmenting path in time $O(m \log n)$

Proof: Can use variants of Prim's or Kruskal's algorithms.

Fattest Augmenting Path

Choose the augmenting path with largest bottleneck value.

- **Theorem:** The fattest augmenting path algorithm runs in $O(m^2 \log n \log f^*)$ time.

augmentations is at most $O(m \lg f^*)$

each augmentation takes $O(m \lg n)$ time.

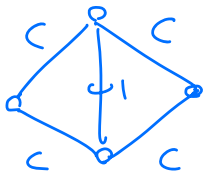
total time is $O(m^2 \lg n \lg f^*)$.

Shortest Augmenting Path

Choose the augmenting path with largest bottleneck value.

- **Theorem:** The shortest augmenting path algorithm runs in $O(m^2 n)$ time.

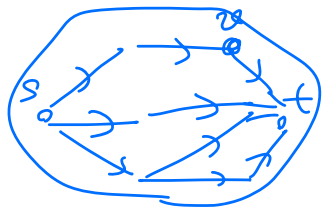
- **Key Claim:** During the execution of shortest augmenting path algorithm, every edge disappears from the residual graph at most $n/2$ times.



- Every edge disappears at most $n/2$ times.
- Every augmentation deletes at least one edge (the bottleneck edge) from the residual graph.
- Therefore, total # of augmentations is at most $\frac{mn}{2}$.
- Each augmentation can be implemented in $O(m)$ time using BFS on the residual graph.

— Total time is $O(mn) \times O(m) = O(m^2 n)$.

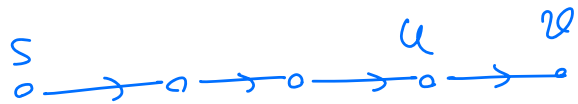
Shortest Augmenting Path



- Let f_i and G_i be the flow after i augmentations and the corresponding residual graph. $f_0 = 0$ $G_0 = G$
- Let $level_i(v)$ denote the unweighted shortest path distance from s to v in G_i.
- **Claim:** The levels can only increase. Namely, $level_i(v) \geq level_{i-1}(v)$ for all vertices v and all integers $i > 0$.

Proof: We prove this by an induction on $level_i(v)$ (and not i):

Let (u,v) be the last edge in the shortest path from s to v in G_i.



we have $level_i(v) = level_i(u) + 1$. ☆

By the induction hypothesis, $level_i(u) \geq level_{i-1}(u)$ □

If $u \to v$ is also an edge in G_{i-1} then $level_{i-1}(v) \leq level_{i-1}(u) + 1$. ○

Therefore $level_i(v) \overset{☆}{=} level_i(u) + 1 \overset{□}{\geq} level_{i-1}(u) + 1 \overset{○}{\geq} level_{i-1}(v)$

— Total time is $O(mn) \times O(m) = O(m^2 n)$.

Shortest Augmenting Path

- Let f_i and G_i be the flow after i augmentations and the corresponding residual graph.
- Let $level_i(v)$ denote the unweighted shortest path distance from s to v in G_i.
- **Claim:** The levels can only increase. Namely, $level_i(v) \geq level_{i-1}(v)$ for all vertices v and all integers $i > 0$.

If $u \to v$ is not an edge in G_{i-1}. This means that the a.p. we find in step i, includes the edge $(v \to u)$. This means $level_{i-1}(u) = level_{i-1}(v) + 1$.

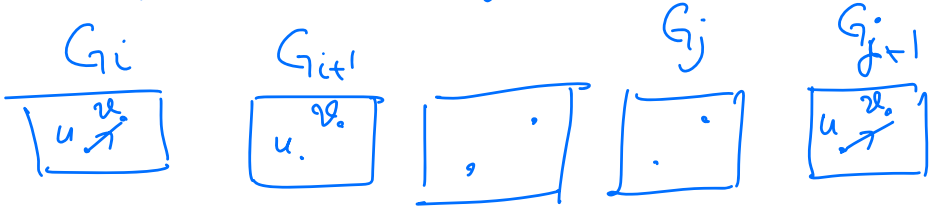
$level_{i-1}(v) = level_{i-1}(u) - 1 \leq level_{i-1}(u) + 1$.

$level_i(v) \overset{\star}{=} level_i(u) + 1 \geq level_{i-1}(u) + 1 \geq level_{i-1}(v)$

Shortest Augmenting Path

- **Key Claim:** During the execution of shortest augmenting path algorithm, every edge disappears from the residual graph at most $n/2$ times.
- **Claim:** Suppose (u, v) is in two residual graphs G_i, G_{j+1} but not in any of $G_{i+1}, \ldots, G_j$. Then $level_j(u) \geq level_i(u) + 2$.

G_i the residual graph after i augmentations.



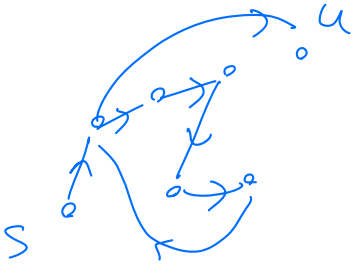
(1) $level_i(v) = level_i(u) + 1$

(2) $level_j(v) \geqslant level_i(v)$

(3) $level_j(u) = level_j(v) + 1 \overset{(2)}{\geqslant} level_i(v) + 1 \overset{(1)}{=} level_i(u) + 1 + 1$

Proof of key claim: Take a vertex u and fix any i. If $\text{level}_i(u) = \infty$ then by claim1 that levels keep increasing, u never becomes reachable, so never participates in any a.p. again. Otherwise $\text{level}_i(u) \leq n-1$.

By claim 2, every edge (u,v) appears/disappers from the residual graph at most $\frac{n-1}{2}$ times. □



Every augmentation deletes one edge from the res graph, and total deletions is upper bounded by $O(mn)$, so we can have at most $O(mn)$ augmentations. Each takes $O(m)$ time, so total runtime is $O(m^2n)$