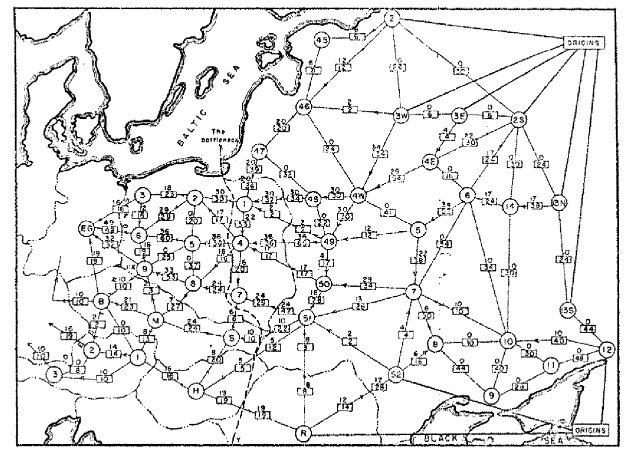
Max Flow, Min Cut
COS 521

Kevin Wayne
Fall 2005

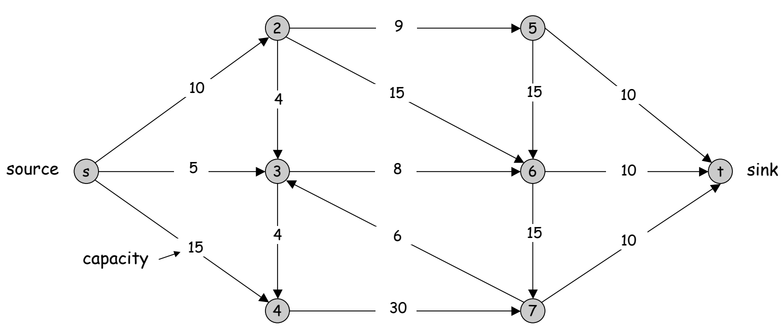
Soviet Rail Network, 1955



Reference: *On the history of the transportation and maximum flow problems.* Alexander Schrijver in Math Programming, 91: 3, 2002.

Minimum Cut Problem

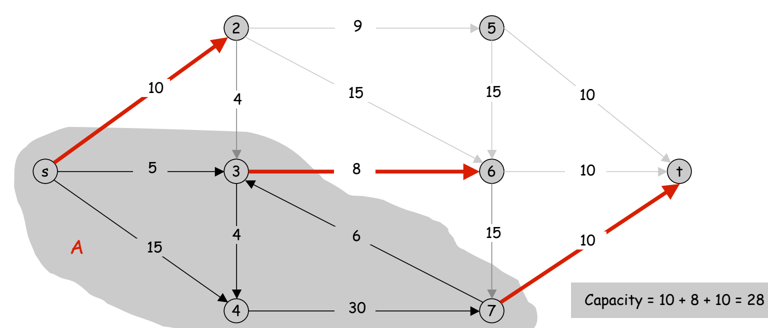
Flow network.
- Digraph G = (V, E), nonnegative edge capacities c(e).
- Two distinguished nodes: s = source, t = sink.
- Assumptions: no parallel edges, no edges entering s or leaving t.



Cuts

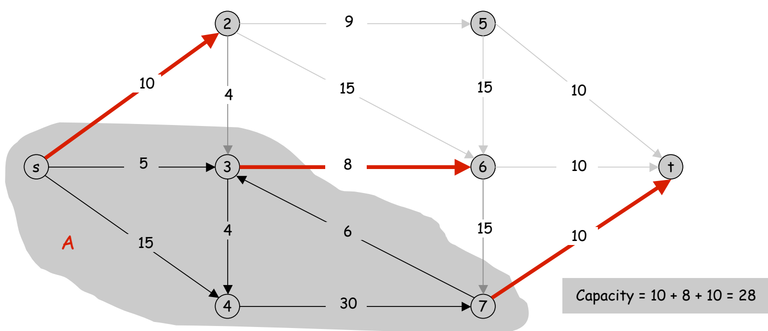
Def. An s-t cut is a partition (A, B) of V with s ∈ A and t ∈ B.

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



Minimum Cut Problem

Min s-t cut problem. Find an s-t cut of minimum capacity.

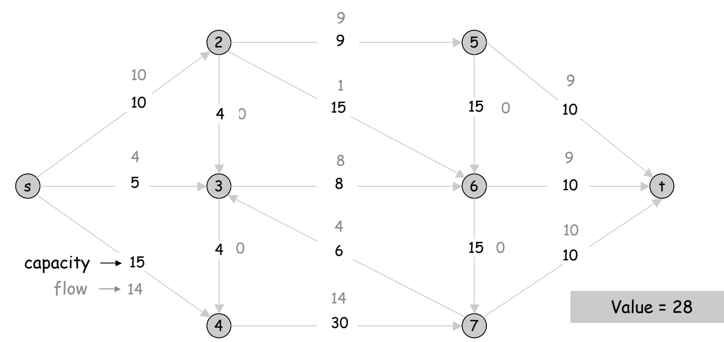


Flows

Def. An s-t flow is a function that satisfies:

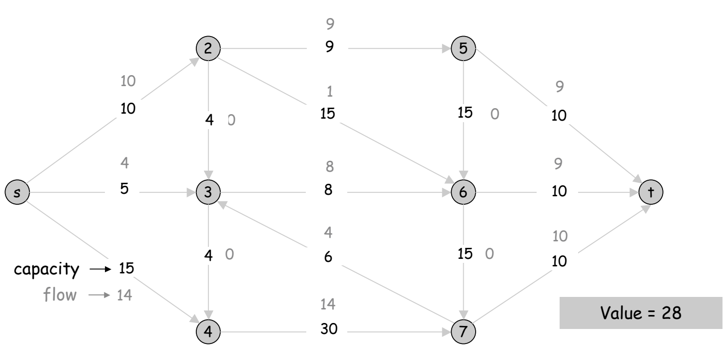
- For each $e \in E$: $0 \le f(e) \le c(e)$ (capacity)
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

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



Maximum Flow Problem

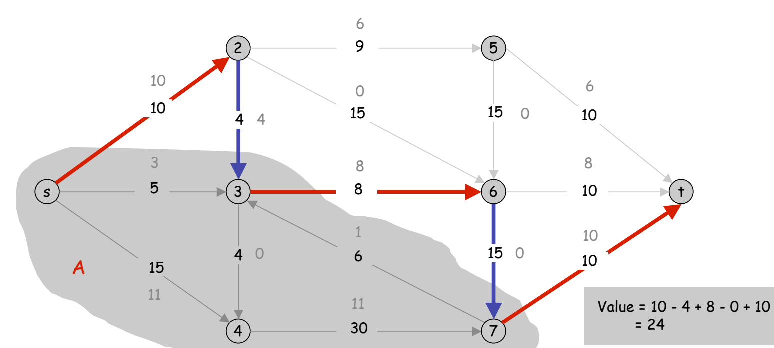
Max flow problem. Find s-t flow of maximum value.



Flows and Cuts

Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then, the net flow sent across the cut is equal to the amount leaving s.

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



Flow value lemma. Let f be any flow, and let (A, B) be any s-t cut. Then

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

Pf.

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

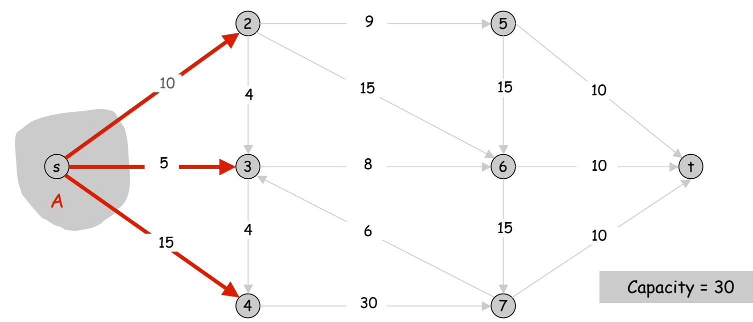
by flow conservation, all terms except v = s are 0 $\longrightarrow$

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

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

Weak duality. Let f be any flow, and let (A, B) be any s-t cut. Then the value of the flow is at most the capacity of the cut.

Cut capacity = 30 $\Rightarrow$ Flow value ≤ 30



Weak duality. Let f be any flow. Then, for any s-t cut (A, B) we have val(f) ≤ cap(A, B).

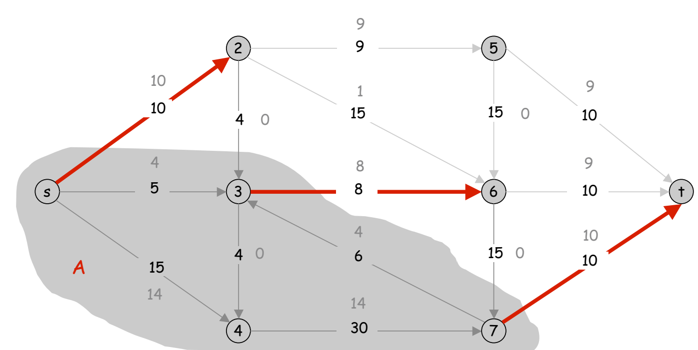
Pf.

$$\begin{aligned} val(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\ &\le \sum_{e \text{ out of } A} f(e) \\ &\le \sum_{e \text{ out of } A} c(e) \\ &= \text{cap}(A, B) \end{aligned}$$

▪

Corollary. Let f be any flow, and let (A, B) be any cut. If val(f) = cap(A, B), then f is a max flow and (A, B) is a min cut.

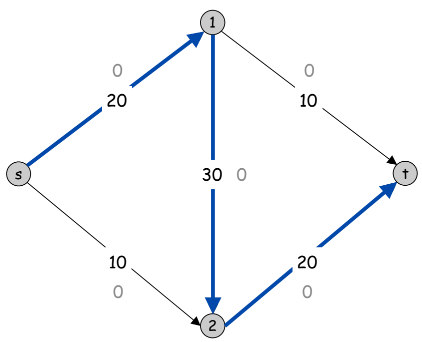
Value of flow = 28
Cut capacity = 28 $\Rightarrow$ Flow value ≤ 28



Towards a Max Flow Algorithm

Greedy algorithm.

- Start with f(e) = 0 for all edge e ∈ E.
- Find an s-t path P where each edge has f(e) < c(e).
- Augment flow along path P.
- Repeat until you get stuck.

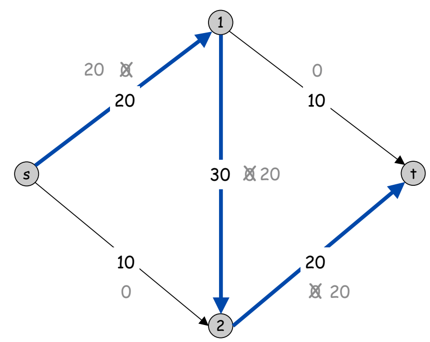


Flow value = 0

Towards a Max Flow Algorithm

Greedy algorithm.

- Start with f(e) = 0 for all edge e ∈ E.
- Find an s-t path P where each edge has f(e) < c(e).
- Augment flow along path P.
- Repeat until you get stuck.



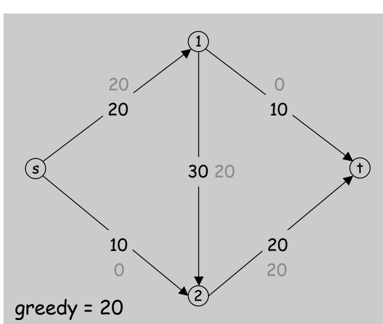
Flow value = 20

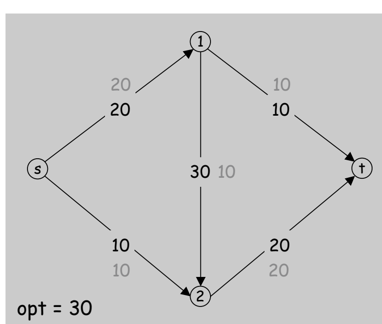
Towards a Max Flow Algorithm

Greedy algorithm.

- Start with f(e) = 0 for all edge e ∈ E.
- Find an s-t path P where each edge has f(e) < c(e).
- Augment flow along path P.
- Repeat until you get stuck.

locally optimality ⇏ global optimality

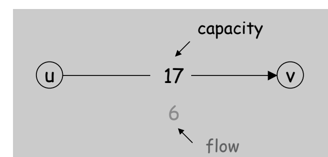




Residual Graph

Original edge: e = (u, v) ∈ E.

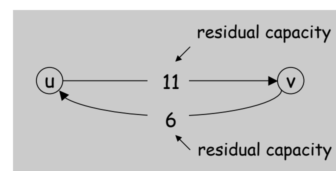
- Flow f(e), capacity c(e).



Residual edge.

- "Undo" flow sent.
- e = (u, v) and e^R = (v, u).
- Residual capacity:

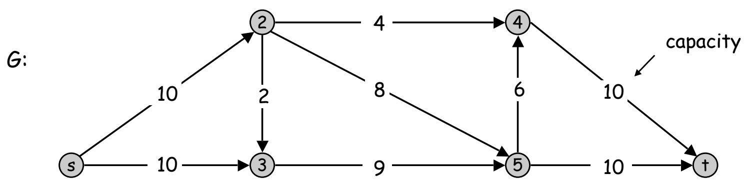
$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$



Residual graph: G_f = (V, E_f).

- Residual edges with positive residual capacity.
- E_f = {e : f(e) < c(e)} ∪ {e^R : c(e) > 0}.

Ford-Fulkerson Algorithm



Max-Flow Min-Cut Theorem

Augmenting path theorem. Flow f is a max flow iff there are no augmenting paths.

Max-flow min-cut theorem. [Elias-Feinstein-Shannon 1956, Ford-Fulkerson 1956] The value of the max flow is equal to the value of the min cut.

Pf. Let f be a flow. Then TFAE:

(i) There exists a cut (A, B) such that val(f) = cap(A, B).
(ii) Flow f is a max flow.
(iii) There is no augmenting path relative to f.

(i) $\Rightarrow$ (ii) This was the corollary to weak duality lemma.

(ii) $\Rightarrow$ (iii) We show contrapositive.

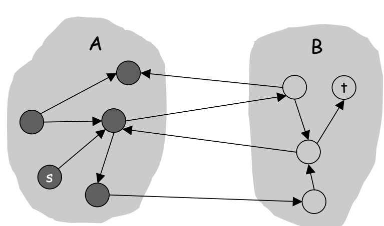
- Let f be a flow. If there exists an augmenting path, then we can improve f by sending flow along path.

Proof of Max-Flow Min-Cut Theorem

(iii) $\Rightarrow$ (i)

- Let f be a flow with no augmenting paths.
- Let A be set of vertices reachable from s in residual graph.
- By definition of A, s $\in$ A.
- By definition of f, t $\notin$ A.

$$\begin{aligned} val(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\ &= \sum_{e \text{ out of } A} c(e) \\ &= cap(A,\ B) \ \blacksquare \end{aligned}$$



original network

Analysis

Assumption. All capacities are integers between 1 and C.

Invariant. Every flow value f(e) and every residual capacities $c_f(e)$ remains an integer throughout the algorithm.

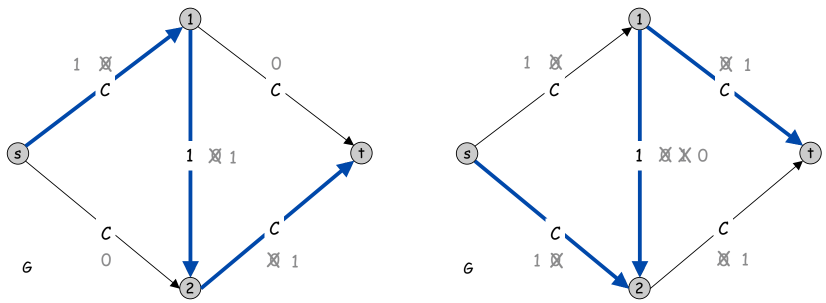
Theorem. The algorithm terminates in at most val(f*) $\leq$ nC iterations. It can be implemented in O(mnC) time.
Pf. Each augmentation increase value by at least 1. ▪

Integrality theorem. If all capacities are integers, then there exists a max flow f for which every flow value f(e) is an integer.
Pf. Since algorithm terminates, theorem follows from invariant. ▪

Ford-Fulkerson: An Exponential Input

Q. Is generic Ford-Fulkerson algorithm polynomial in input size?

m, n, and log C



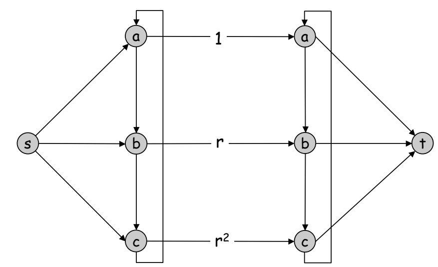
Ford-Fulkerson: A Pathological Input

Q. Is Ford-Fulkerson algorithm finite?

Let $r = \frac{-1 + \sqrt{5}}{2} \approx 0.618...$ $[\ r^{n+2} = r^n - r^{n+1}\ ]$

Max flow $= 1 + r + r^2$.

Augmentations: first augment 1 unit, then repeatedly choose path with lowest capacity.



Choosing Good Augmenting Paths

Goal: choose augmenting paths so that:
- Can find augmenting paths efficiently.
- Few iterations.

Choose augmenting paths with: [Edmonds-Karp 1972, Dinitz 1970]
- Max bottleneck capacity.
- Sufficiently large bottleneck capacity.
- Fewest number of edges.

Shortest Augmenting Path: Overview of Analysis

L1. The length of the shortest augmenting path never decreases.

L2. After at most m augmentations, the length of the shortest augmenting path strictly increases.

Theorem. The shortest augmenting path algorithm performs at most $O(mn)$ augmentations. It can be implemented in $O(m^2n)$ time.
- $O(m)$ time to find shortest augmenting path via BFS.
- $O(m)$ augmentations for paths of exactly k edges. ▪

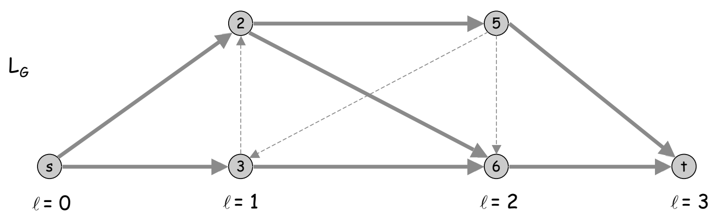
k < n

Shortest Augmenting Path: Analysis

Level graph.

number of edges

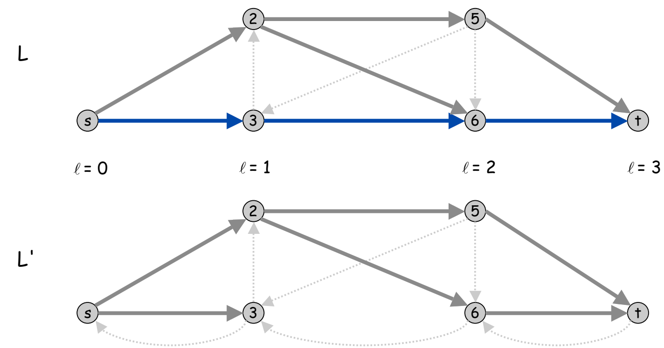
- Define $\ell(v)$ = length of shortest s-v path in G.
- L_G = (V, F) is subgraph of G that contains only those edges $(u, v) \in E$ with $\ell(v) = \ell(u) + 1$.
- Compute L_G in O(m+n) time using BFS, deleting back and side edges.
- P is a shortest s-u path in G iff it is an s-u path L_G.



Shortest Augmenting Path: Analysis

L1. The length of the shortest augmenting path never decreases.

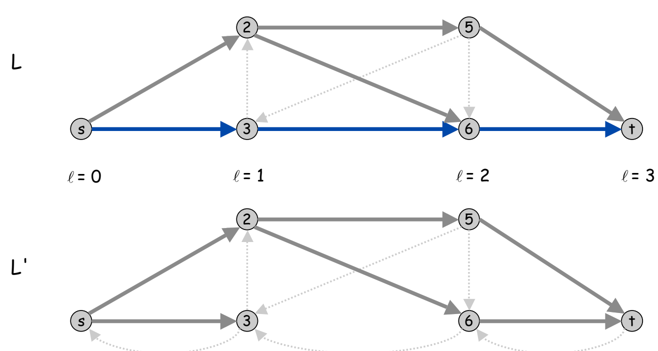
- Let f and f' be flow before and after a shortest path augmentation.
- Let L and L' be level graphs of G_f and $G_{f'}$.
- Only back edges added to $G_{f'}$.
- Path with back edge has length greater than previous length.



Shortest Augmenting Path: Analysis

L2. After at most m augmentations, the length of the shortest augmenting path strictly increases.

- At least one edge (the bottleneck edge) is deleted from L after each augmentation.
- No new edges added to L until length of shortest path strictly increases.



Shortest Augmenting Path: Review of Analysis

L1. The length of the shortest augmenting path never decreases.

L2. After at most m augmentations, the length of the shortest augmenting path strictly increases.

Theorem. The shortest augmenting path algorithm performs at most O(mn) augmentations. It can be implemented in $O(m^2n)$ time.

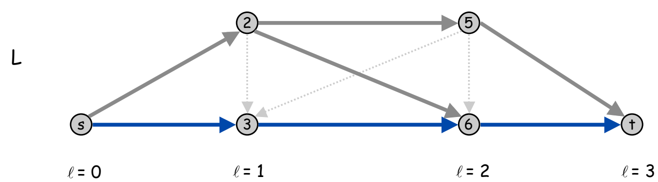
Note: $\Theta(mn)$ augmentations necessary on some networks.

- Try to decrease time per augmentation instead.
- Dynamic trees $\Rightarrow$ O(mn log n) [Sleator-Tarjan, 1983]
- Simple idea $\Rightarrow$ $O(mn^2)$

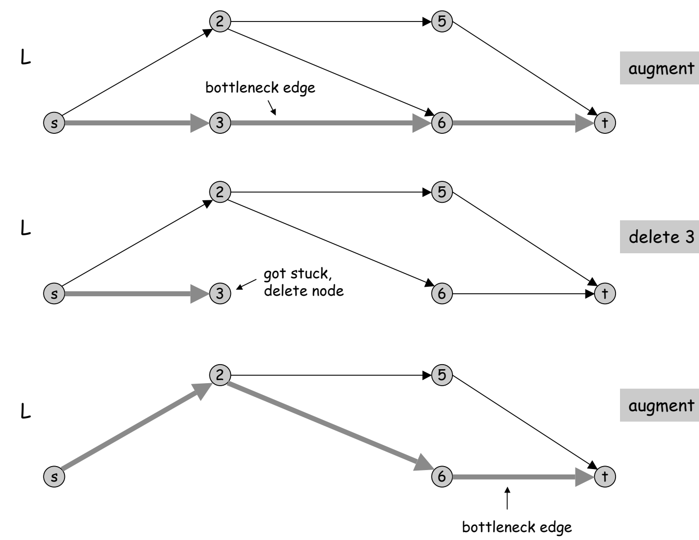
Shortest Augmenting Path: Improved Version

Two types of augmentations.
- Normal augmentation: length of shortest path doesn't change.
- Special augmentation: length of shortest path strictly increases.

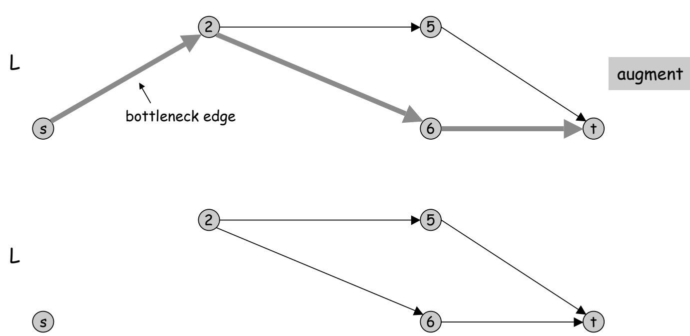
L3. Group of normal augmentations takes O(mn) time.
- Explicitly maintain level graph - it changes by at most 2n edges after each normal augmentation.
- Start at s, advance along an edge in L until reach t or get stuck.
  - if reach t, augment and delete at least one edge
  - if get stuck, delete node



Shortest Augmenting Path: Improved Version



Shortest Augmenting Path: Improved Version



Stop: length of shortest path must have strictly increased.

Shortest Augmenting Path: Improved Version

Two types of augmentations.
- Normal augmentation: length of shortest path doesn't change.
- Special augmentation: length of shortest path strictly increases.

L3. Group of normal augmentations takes O(mn) time.
- At most n advance steps before you either
  - get stuck: delete a node from level graph
  - reach t: augment and delete an edge from level graph

Theorem. Algorithm runs in $O(mn^2)$ time.
- O(mn) time between special augmentations.
- At most n special augmentations.

History of Worst-Case Running Times

Year	Discoverer	Method	Asymptotic Time
1951	Dantzig	Simplex	$m\, n^2 C$ †
1955	Ford, Fulkerson	Augmenting path	$m\, n\, C$ †
1970	Edmonds-Karp	Shortest path	$m^2 n$
1970	Edmonds-Karp	Fattest path	$m \log C\, (m \log n)$ †
1970	Dinitz	Improved shortest path	$m\, n^2$
1972	Edmonds-Karp, Dinitz	Capacity scaling	$m^2 \log C$ †
1973	Dinitz-Gabow	Improved capacity scaling	$m\, n \log C$ †
1974	Karzanov	Preflow-push	n^3
1983	Sleator-Tarjan	Dynamic trees	$m\, n \log n$
1986	Goldberg-Tarjan	FIFO preflow-push	$m\, n \log (n^2 / m)$
. . .	. . .	. . .	. . .
1997	Goldberg-Rao	Length function	$m^{3/2} \log (n^2 / m) \log C$ † $mn^{2/3} \log (n^2 / m) \log C$ †

† Edge capacities are between 1 and C.

next time