COMPSCI 311: Introduction to Algorithms

Lecture 17: Network Flow

Marius Minea

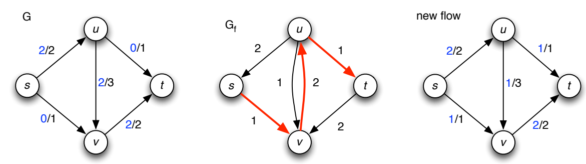
University of Massachusetts Amherst

slides credit: Dan Sheldon (adapted)

Review: Augmenting Flows

residual graph; edges: forward (difference), reverse (existing flow)

augmenting path: $s \rightsquigarrow r$ in residual graph, bottleneck capacity



Review: Ford-Fulkerson Algorithm

▷ Augment flow as long as it is possible
while there exists an s-t path P in G_f **do**
  f = Augment(f, P)
  update G_f
end while
return f

Relate maximum flow to minimum cut

Clicker Question 1

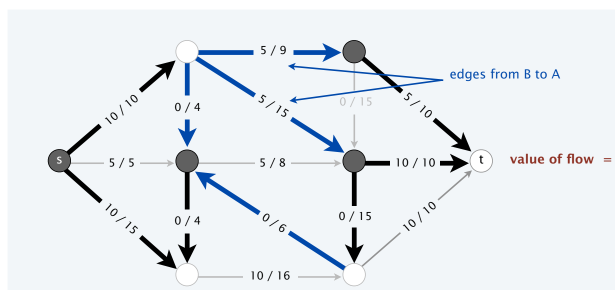
What is the value of the flow across the black-white cut?

A: 10

B: 15

C: 20

D: 25



Flow Value Lemma

First relationship between cuts and flows

Lemma: let f be any flow and (A, B) be any s-t cut. Then

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

Proof (see book) use conservation of flow:
all the flow out of s must leave A eventually.

Rewrite flow as $v(f) = \sum_{v \in A} f^{\text{out}}(v) - f^{\text{in}}(v)$

only nonzero difference is $f(s)$

Consider cases: edge in A, leading out of A, leading into A

Network flow: quiz 3

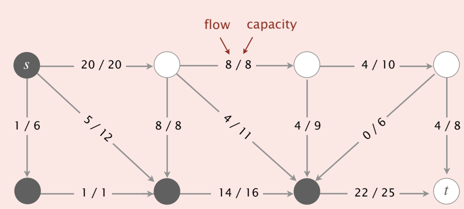
Which is the net flow across the given cut?

A. 11 (20 + 25 – 8 – 11 – 9 – 6)

B. 26 (20 + 22 – 8 – 4 – 4)

C. 42 (20 + 22)

D. 45 (20 + 25)



slide credit: Kevin Wayne / Pearson

Corollary: Cuts and Flows

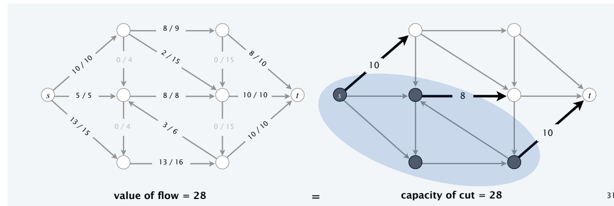
Really important corollary of flow-value lemma

Corollary: Let f be **any** s-t flow and let (A, B) be **any** s-t cut. Then $v(f) \leq c(A, B)$.

Proof:

$$
\begin{aligned}
v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \\
&\leq \sum_{e \text{ out of } A} f(e) \\
&\leq \sum_{e \text{ out of } A} c(e) \\
&= c(A, B)
\end{aligned}
$$

Duality: Max Flow – Min Cut



Claim If there is a flow f^* and cut (A^*, B^*) such that $v(f^*) = c(A^*, B^*)$, then

- f^* is a maximum flow
- (A^*, B^*) is a minimum cut

F-F returns a maximum flow

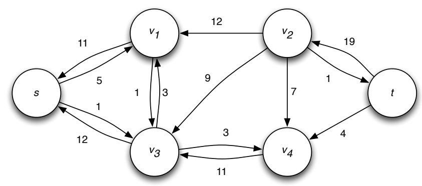
Theorem: The s-t flow f returned by F-F is a maximum flow.

- Since f is the final flow there are no residual paths in G_f.
- Let (A, B) be the s-t cut where A consists of all nodes reachable from s in the residual graph.
  - Any edge out of A must have $f(e) = c(e)$ otherwise there would be more nodes than just A that reachable from s.
  - Any edge into A must have $f(e) = 0$ otherwise there would be more nodes than just A that reachable from s.
- Therefore
$$
\begin{aligned}
v(f) &= \sum_{e \text{ out of} A} f(e) - \sum_{e \text{ into} A} f(e) \\
&= \sum_{e \text{ out of} A} c(e) = c(A, B)
\end{aligned}
$$

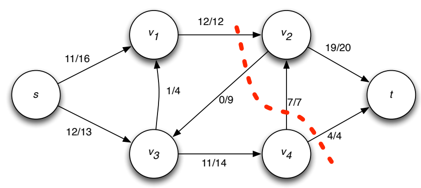
F-F finds a minimum cut

Theorem: The cut (A, B) where A is the set of all nodes reachable from s in the residual graph is a minimum-cut.

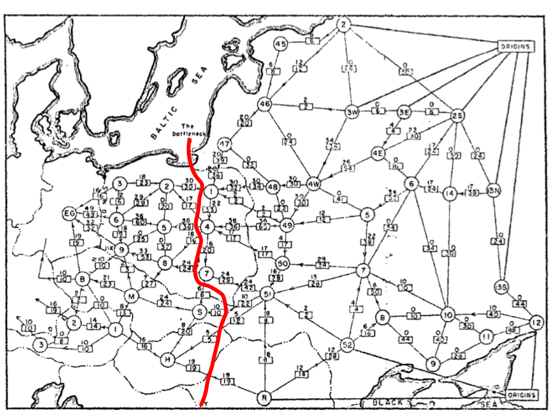
G_f



G



F-F finds a minimum cut



Capacity 163,000 tons per day [Harris and Ross 1955]

Ford-Fulkerson Running Time

- Flow increases at least one unit per iteration
- F-F terminates in at most C_s iterations, where C_s is sum of capacities leaving source.
- $C_s \leq n\,C_{\max}$ (in terms of maximum edge capacity)
- Running time: $O(m\,n\,C_{\max})$

Is this polynomial? pseudo-polynomial (exponential in $\log C_{\max}$)

Ford–Fulkerson: exponential example

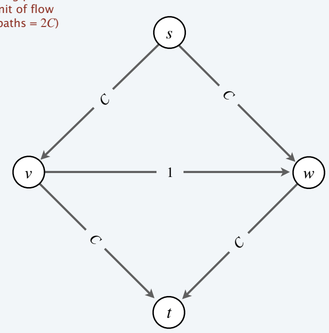
Q. Is generic Ford–Fulkerson algorithm poly-time in input size?

m, n, and $\log C$

A. No. If max capacity is C, then algorithm can take $\geq C$ iterations.

- $s \to v \to w \to t$
- $s \to w \to v \to t$
- $s \to v \to w \to t$
- $s \to w \to v \to t$
- …
- $s \to v \to w \to t$
- $s \to w \to v \to t$

each augmenting path sends only 1 unit of flow (# augmenting paths = $2C$)



Improving Running Time

Choose good augmenting paths, with

- Large enough bottleneck capacity

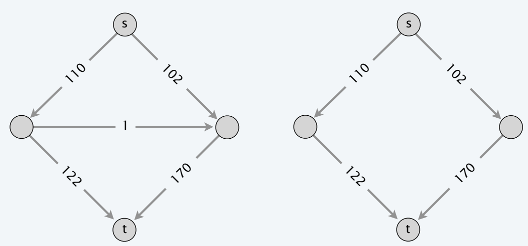
  Maximum hard to find, but can cleverly search for good values

- Fewest edges (Edmonds-Karp, Dinitz)

Capacity-scaling algorithm

Overview. Choosing augmenting paths with "large" bottleneck capacity.

- Maintain scaling parameter Δ. (though not necessarily largest)
- Let $G_f(\Delta)$ be the part of the residual network containing only those edges with capacity $\geq \Delta$.
- Any augmenting path in $G_f(\Delta)$ has bottleneck capacity $\geq \Delta$.



G_f $\quad\quad$ $G_f(\Delta)$, $\Delta = 100$

Capacity-scaling algorithm

CAPACITY-SCALING(G)

FOREACH edge $e \in E$: $f(e) \leftarrow 0$.

$\Delta \leftarrow$ largest power of 2 $\leq C$.

WHILE $(\Delta \geq 1)$

  $G_f(\Delta) \leftarrow \Delta$-residual network of G with respect to flow f.

  WHILE (there exists an $s \to t$ path P in $G_f(\Delta)$)

    $f \leftarrow$ AUGMENT(f, c, P).

    Update $G_f(\Delta)$.

  $\Delta \leftarrow \Delta / 2$.

RETURN f.

Δ-scaling phase

Capacity-Scaling: Running Time

How many scaling phases? $\log C_{\max}$ (precisely: $1 + \lfloor \log C_{\max} \rfloor$)

How much does the flow increase at every augmentation? $\geq \Delta$

How many augmentations per phase?

Intuition: at end of each Δ phase, residual capacity on an edge in minimum cut less than Δ, else we would have augmented more.

Capacity-scaling algorithm: analysis of running time

Lemma 2. Let f be the flow at the end of a Δ-scaling phase. Then, the max-flow value $\leq val(f) + m\,\Delta$.

Pf.

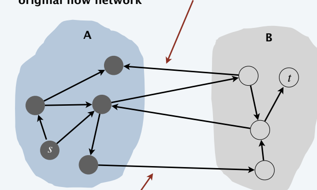
- We show there exists a cut (A, B) such that $cap(A, B) \leq val(f) + m\,\Delta$.
- Choose A to be the set of nodes reachable from s in $G_f(\Delta)$.
- By definition of A: $s \in A$.
- By definition of flow f: $t \notin A$.

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

flow value lemma

edge $e = (v, w)$ with $v \in B$, $w \in A$ must have $f(e) < \Delta$

original flow network



edge $e = (v, w)$ with $v \in A$, $w \in B$ must have $f(e) > c(e) - \Delta$

Capacity-scaling algorithm: analysis of running time

Lemma 1. There are $1 + \lfloor \log_2 C \rfloor$ scaling phases.
Pf. Initially $C/2 < \Delta \le C$; Δ decreases by a factor of 2 in each iteration. ▪

Lemma 2. Let f be the flow at the end of a Δ-scaling phase.
Then, the max-flow value $\le\ val(f) + m\,\Delta$.
Pf. Next slide.

Lemma 3. There are $\le 2m$ augmentations per scaling phase.
Pf.
- Let f be the flow at the beginning of a Δ-scaling phase. ← or equivalently, at the end of a 2Δ-scaling phase
- Lemma 2 $\Rightarrow$ max-flow value $\le\ val(f) + m\,(2\,\Delta)$.
- Each augmentation in a Δ-phase increases $val(f)$ by at least Δ. ▪

Theorem. The capacity-scaling algorithm takes $O(m^2 \log C)$ time.
Pf.
- Lemma 1 + Lemma 3 $\Rightarrow$ $O(m \log C)$ augmentations.
- Finding an augmenting path takes $O(m)$ time. ▪

Shortest augmenting path

Q. How to choose next augmenting path in Ford–Fulkerson?
A. Pick one that uses the fewest edges.
↖ can find via BFS

SHORTEST-AUGMENTING-PATH(G)

FOREACH $e \in E$: $f(e) \leftarrow 0$.
  $G_f \leftarrow$ residual network of G with respect to flow f.
  WHILE (there exists an $s \rightsquigarrow t$ path in G_f)
    $P \leftarrow$ BREADTH-FIRST-SEARCH(G_f).
    $f \leftarrow$ AUGMENT(f, c, P).
    Update G_f.
RETURN f.

Network flow: quiz 6

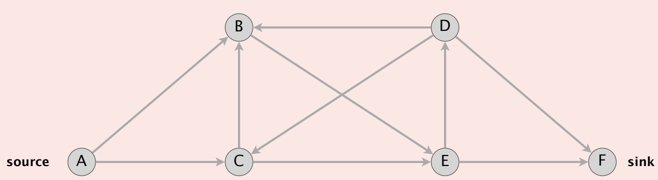
How to compute the level graph L_G efficiently?

A. Depth-first search.

B. Breadth-first search.

C. Both A and B.

D. Neither A nor B.



Shortest augmenting path: overview of analysis

Lemma 1. The length of a shortest augmenting path never decreases.
Pf. Ahead. ↖ number of edges

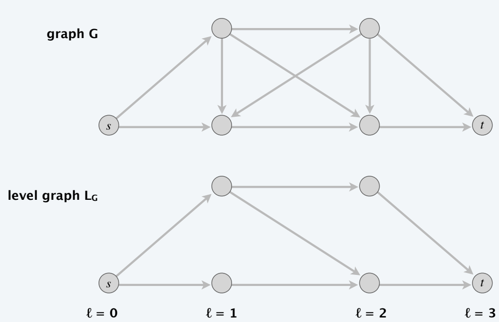
Lemma 2. After at most m shortest-path augmentations, the length of a shortest augmenting path strictly increases.
Pf. Ahead.

Theorem. The shortest-augmenting-path algorithm takes $O(m^2 n)$ time.
Pf.
- $O(m)$ time to find a shortest augmenting path via BFS.
- There are $\le\ m\,n$ augmentations.
  - at most m augmenting paths of length k ← Lemma 1 + Lemma 2
  - at most $n-1$ different lengths ▪

  ↖ augmenting paths are simple paths

Shortest augmenting path: analysis

Def. Given a digraph $G = (V, E)$ with source s, its level graph is defined by:
- $\ell(v)$ = number of edges in shortest $s \rightsquigarrow v$ path.
- $L_G = (V, E_G)$ is the subgraph of G that contains only those edges $(v, w) \in E$ with $\ell(w) = \ell(v) + 1$.



Network flow: quiz 5

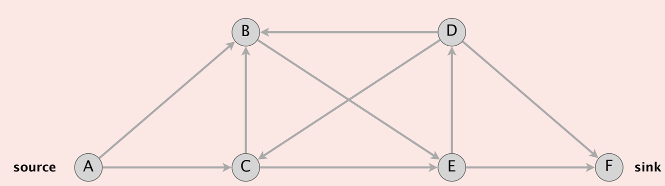
Which edges are in the level graph of the following digraph?

A. D→F.

B. E→F.

C. Both A and B.

D. Neither A nor B.

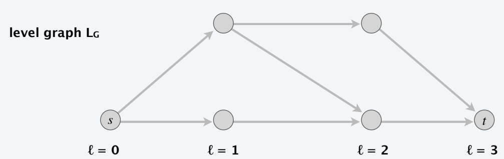


Shortest augmenting path: analysis

Def. Given a digraph $G = (V, E)$ with source s, its level graph is defined by:

- $\ell(v)$ = number of edges in shortest $s \rightsquigarrow v$ path.
- $L_G = (V, E_G)$ is the subgraph of G that contains only those edges $(v, w) \in E$ with $\ell(w) = \ell(v) + 1$.

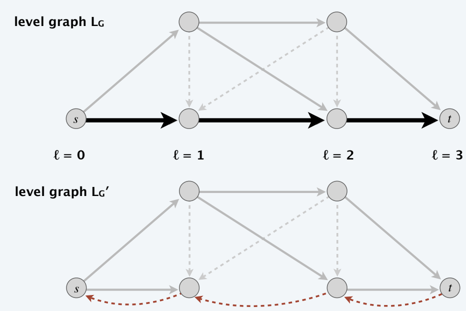
Key property. P is a shortest $s \rightsquigarrow v$ path in G iff P is an $s \rightsquigarrow v$ path in L_G.



Shortest augmenting path: analysis

Lemma 1. The length of a shortest augmenting path never decreases.

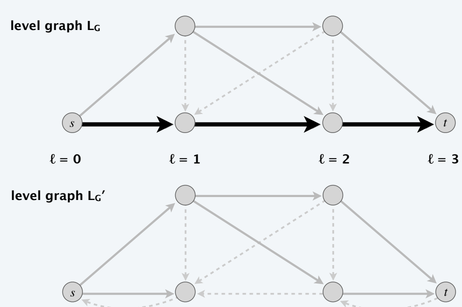
- Let f and f' be flow before and after a shortest-path augmentation.
- Let L_G and $L_{G'}$ be level graphs of G_f and $G_{f'}$.
- Only back edges added to $G_{f'}$ (any $s \rightsquigarrow t$ path that uses a back edge is longer than previous length) ▪



Shortest augmenting path: analysis

Lemma 2. After at most m shortest-path augmentations, the length of a shortest augmenting path strictly increases.

- At least one (bottleneck) edge is deleted from L_G per augmentation.
- No new edge added to L_G until shortest path length strictly increases. ▪



Shortest augmenting path: review of analysis

Lemma 1. Throughout the algorithm, the length of a shortest augmenting path never decreases.

Lemma 2. After at most m shortest-path augmentations, the length of a shortest augmenting path strictly increases.

Theorem. The shortest-augmenting-path algorithm takes $O(m^2 n)$ time.