Algorithm Design and Analysis

CSE 565

LECTURES 25, 26

Network Flow

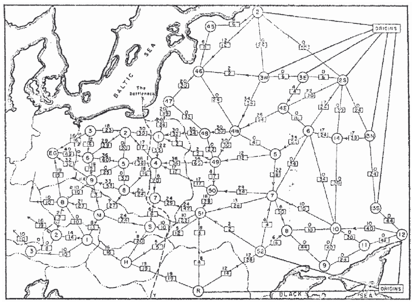
- Maximum Fow
- Minimum Cut
- Ford-Fulkerson

Sofya Raskhodnikova

S. Raskhodnikova; based on slides by E. Demaine, C. Leiserson, K. Wayne

---

Soviet Rail Network, 1955



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

Maximum Flow and Minimum Cut

Max flow and min cut.

- Two very rich algorithmic problems.
- Cornerstone problems in combinatorial optimization.
- Beautiful mathematical duality.

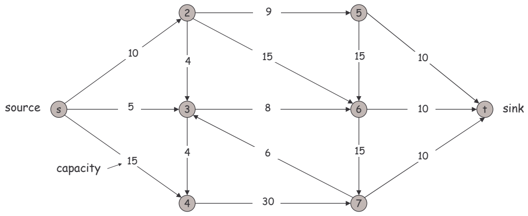
Nontrivial applications / reductions.

- Data mining.
- Open-pit mining.
- Project selection.
- Airline scheduling.
- Bipartite matching.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Network reliability.
- Distributed computing.
- Egalitarian stable matching.
- Security of statistical data.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- Many many more ...

Minimum Cut Problem

Flow network.

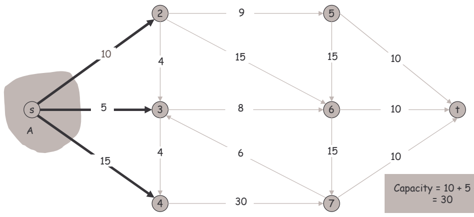
- Abstraction for material flowing through the edges.
- G = (V, E) = directed graph, no parallel edges.
- Two distinguished nodes: s = source, t = sink.
- c(e) = capacity of edge e.



Cuts

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

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

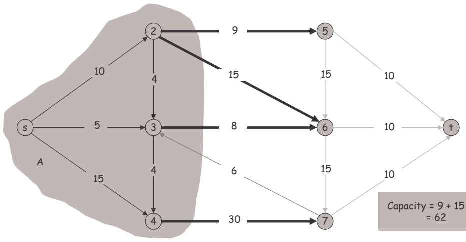


Capacity = 10 + 5 + 15 = 30

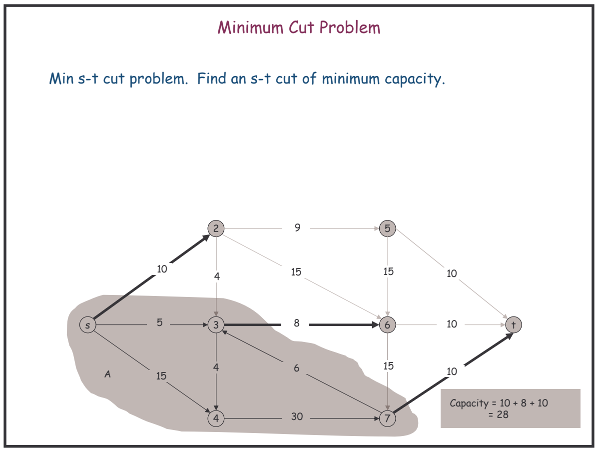
Cuts

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

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



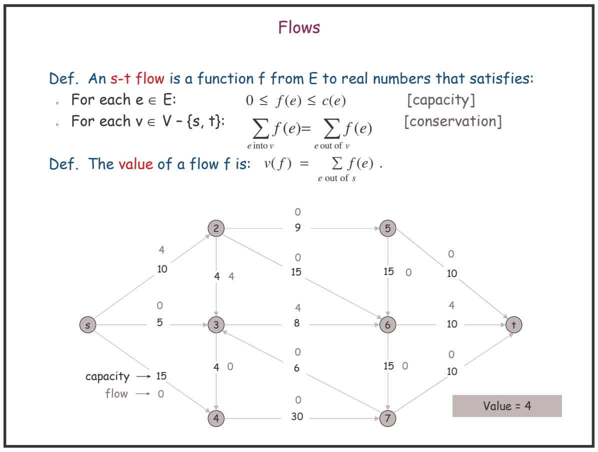
Capacity = 9 + 15 + 8 + 30 = 62



Minimum Cut Problem

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

Capacity = 10 + 8 + 10 = 28



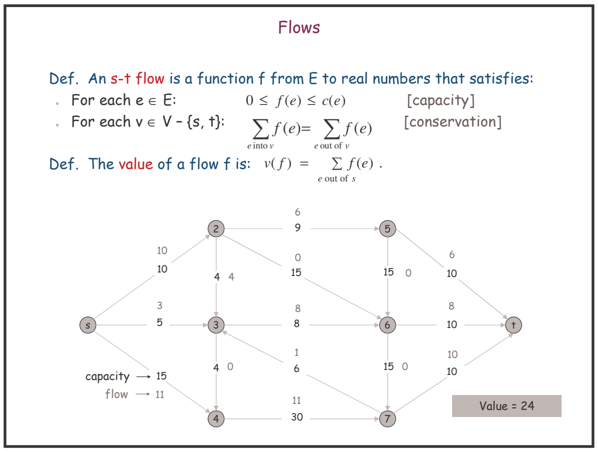
Flows

Def. An s-t flow is a function f from E to real numbers that satisfies:

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

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

Value = 4



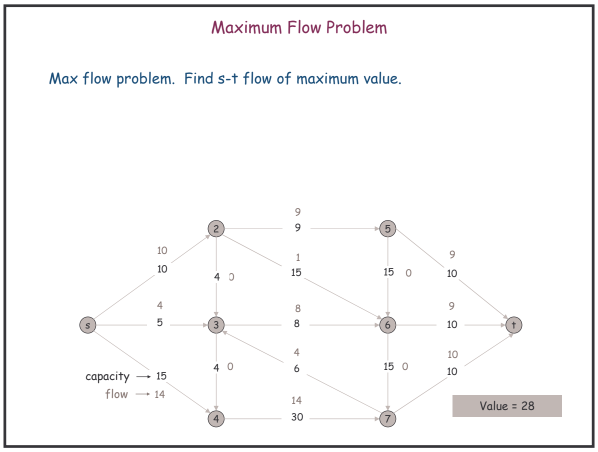
Flows

Def. An s-t flow is a function f from E to real numbers that satisfies:

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

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

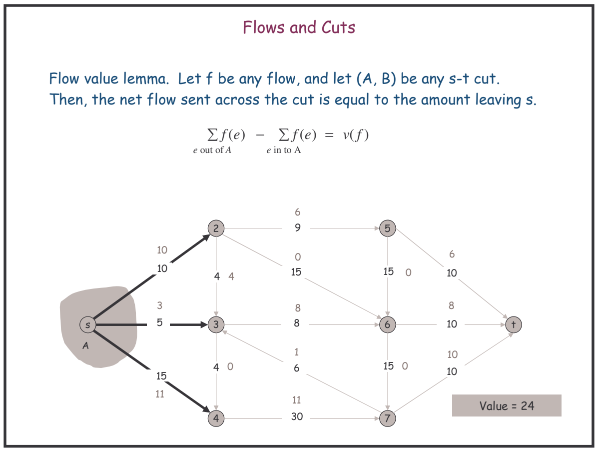
Value = 24



Maximum Flow Problem

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

Value = 28

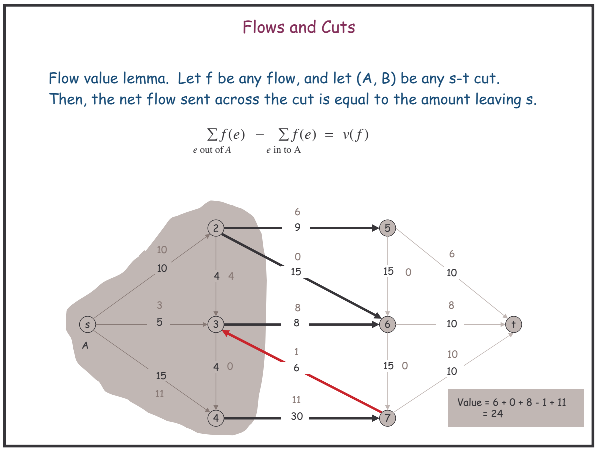


Flows and Cuts

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.

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

Value = 24



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.

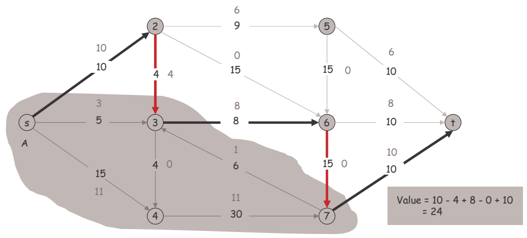
$$\sum_{e \text{ out of } A} f(e) \; - \sum_{e \text{ in to A}} f(e) \; = \; v(f)$$

Value = 6 + 0 + 8 - 1 + 11 = 24

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.

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



Value = 10 - 4 + 8 - 0 + 10
= 24

Flows and Cuts

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

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

Proof.

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

by flow conservation, all terms except v = s are 0 ⟶

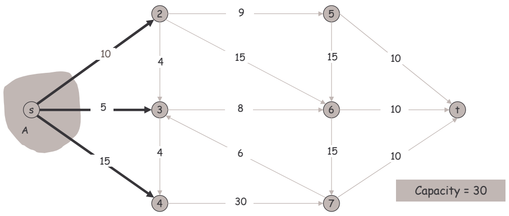
$$= \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to v}} f(e) \right)$$

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

Flows and Cuts

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 ⇒ Flow value ≤ 30



Capacity = 30

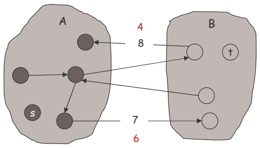
Flows and Cuts

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

Pf.

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

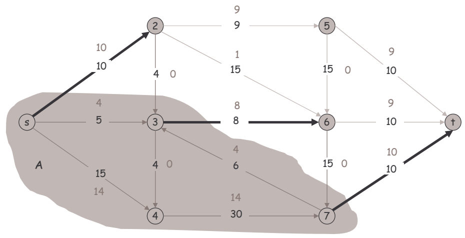
▪



Certificate of Optimality

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

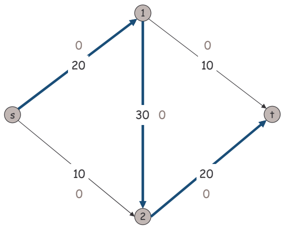
Value of flow = 28
Cut capacity = 28 ⇒ Flow value ≤ 28



Towards a Max Flow Algorithm

Greedy algorithm.

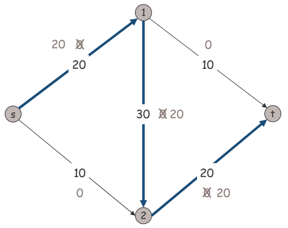
- Start with f(e) = 0 for all edge e ∈ E.
- Find an s-t path P where each edge has f(e) < c(e).
- Augment flow along path P.
- Repeat until you get stuck.



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.

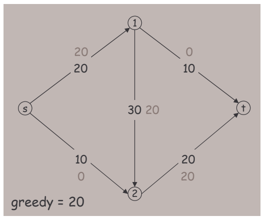


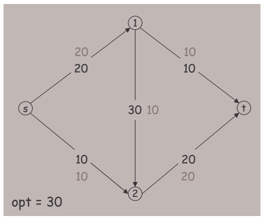
Flow value = 20

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.

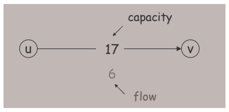
locally optimality ⇏ global optimality





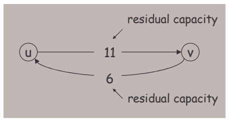
Residual Graph

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:

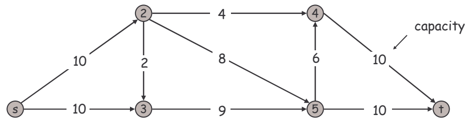
$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$



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}.

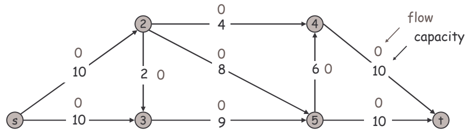
Ford-Fulkerson Algorithm

G:



Ford-Fulkerson Algorithm

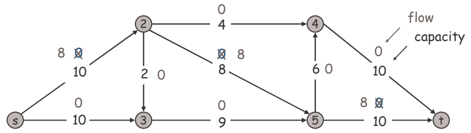
G:



Flow value = 0

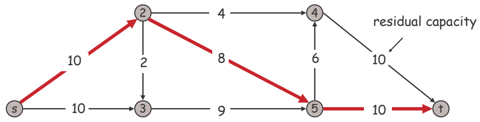
Ford-Fulkerson Algorithm

G:

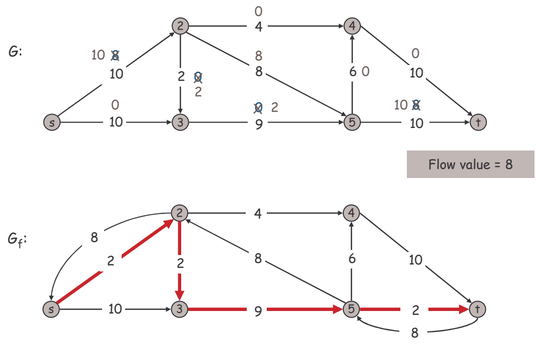


Flow value = 0

G_f:

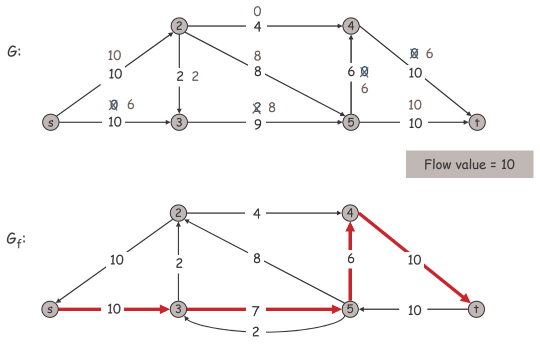


Ford-Fulkerson Algorithm



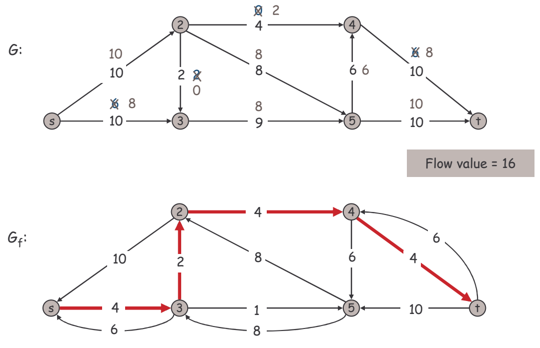
Flow value = 8

Ford-Fulkerson Algorithm



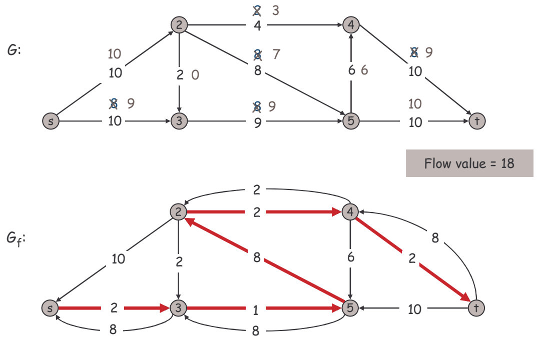
Flow value = 10

Ford-Fulkerson Algorithm



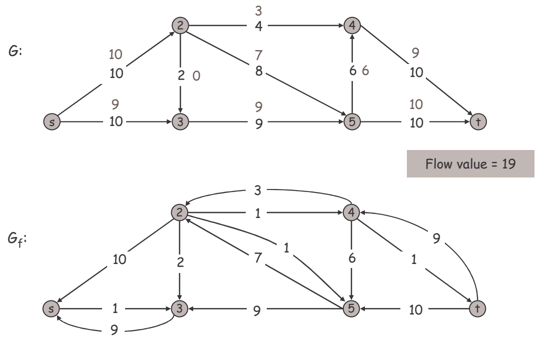
Flow value = 16

Ford-Fulkerson Algorithm



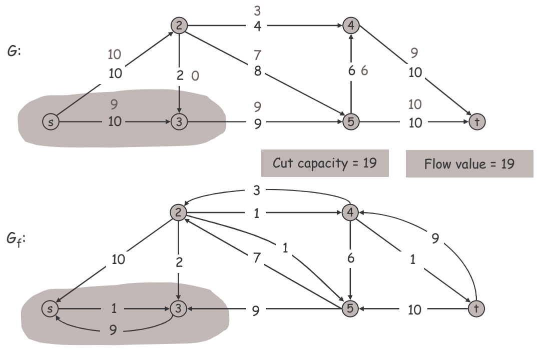
Flow value = 18

Ford-Fulkerson Algorithm



Flow value = 19

Ford-Fulkerson Algorithm



Cut capacity = 19

Flow value = 19

Augmenting Path Algorithm

```
Augment(f, c, P) {
   b ← bottleneck(P)
   foreach e ∈ P {
      if (e ∈ E) f(e) ← f(e) + b
      else       f(e) ← f(e) - b
   }
   return f
}
```

Min residual capacity of an edge in P

forward edge

reverse edge

```
Ford-Fulkerson(G, s, t, c) {
   foreach e ∈ E  f(e) ← 0
   G_f ← residual graph

   while (there is an s-t path P in G_f) {
      f ← Augment(f, c, P)
      update G_f
   }
   return f
}
```

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 s-t cut.

Pf. We prove both simultaneously by showing (i) -- (iii) are equivalent:

(i) There exists an s-t cut (A, B) such that v(f) = cap(A, B).
(ii) Flow f is a max flow, and (A,B) is min s-t cut.
(iii) There is no augmenting path relative to f.

(i) ⇒ (ii) This was the corollary to weak duality lemma.

(ii) ⇒ (iii) We show contrapositive.

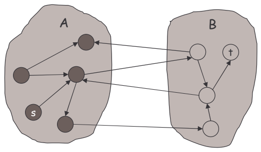
- Let f be a flow. If there exists an augmenting path, then we can improve f by sending flow along path.

Proof of Max-Flow Min-Cut Theorem

(iii) ⇒ (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 ∈ A.
- By definition of f, t ∉ A.

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



original network

Running Time

Assumption. All capacities are integers between 1 and C.

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

Theorem. The algorithm terminates in at most v(f*) ≤ nC iterations.
Pf. Each augmentation increase value by at least 1. ▪

Running time of Ford-Fulkerson: O(mnC). Space: O(m+n).
Corollary. If C = 1, Ford-Fulkerson runs in O(mn) time.

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. ▪