Network Flow

Maximum Flow and Minimum Cut

- Two very rich algorithmic problems
- Cornerstone problems in combinatorial optimization

- 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
- …..

Network Flow Definitions

- Capacity
- Source, Sink
- Capacity Condition
- Conservation Condition
- Value of a flow

Network Flow Definitions

- Flowgraph: **Directed graph** with distinguished vertices s (source) and t (sink)
- Capacities on the edges, c(e) >= 0
- Problem, assign flows f(e) to the edges such that:

  0 <= f(e) <= c(e)

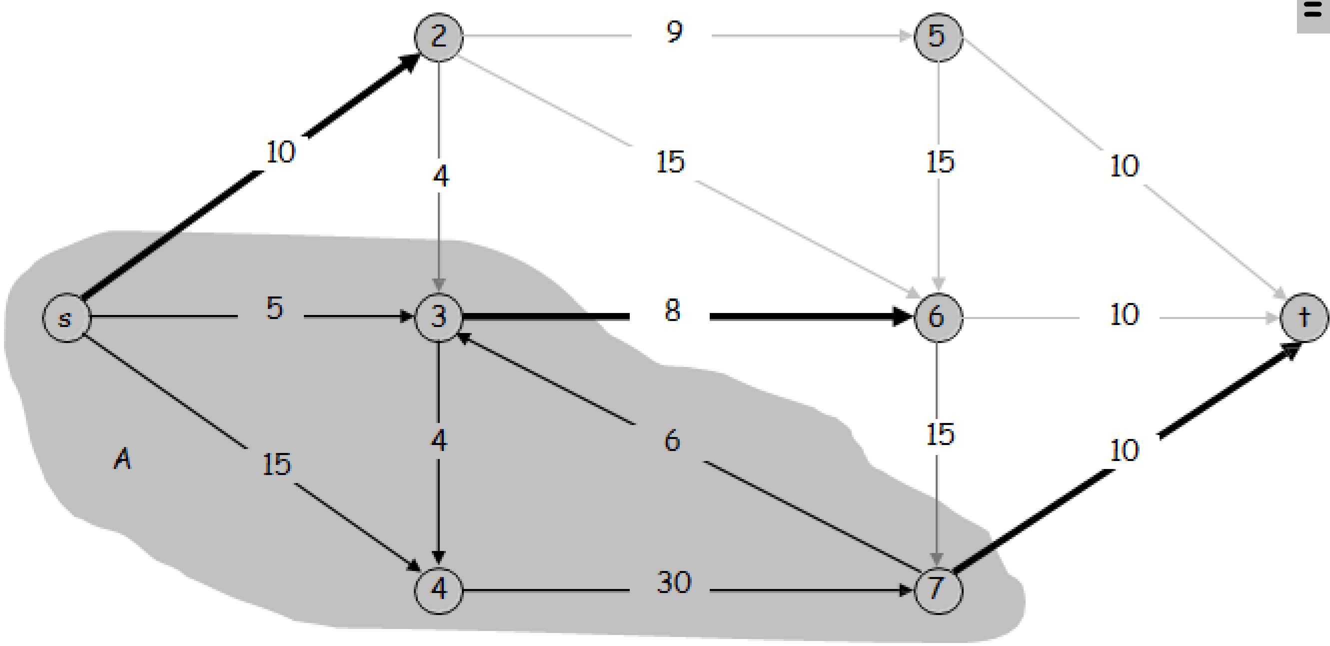
  - Flow is **conserved at vertices other than s and t**
    - Flow conservation: flow going into a vertex equals the flow going out **(flow across edges is lossless)**
  - The flow leaving the source is a large as possible

Minimum Cut Problem

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

Capacity = 10 + 8 + 10

= 28

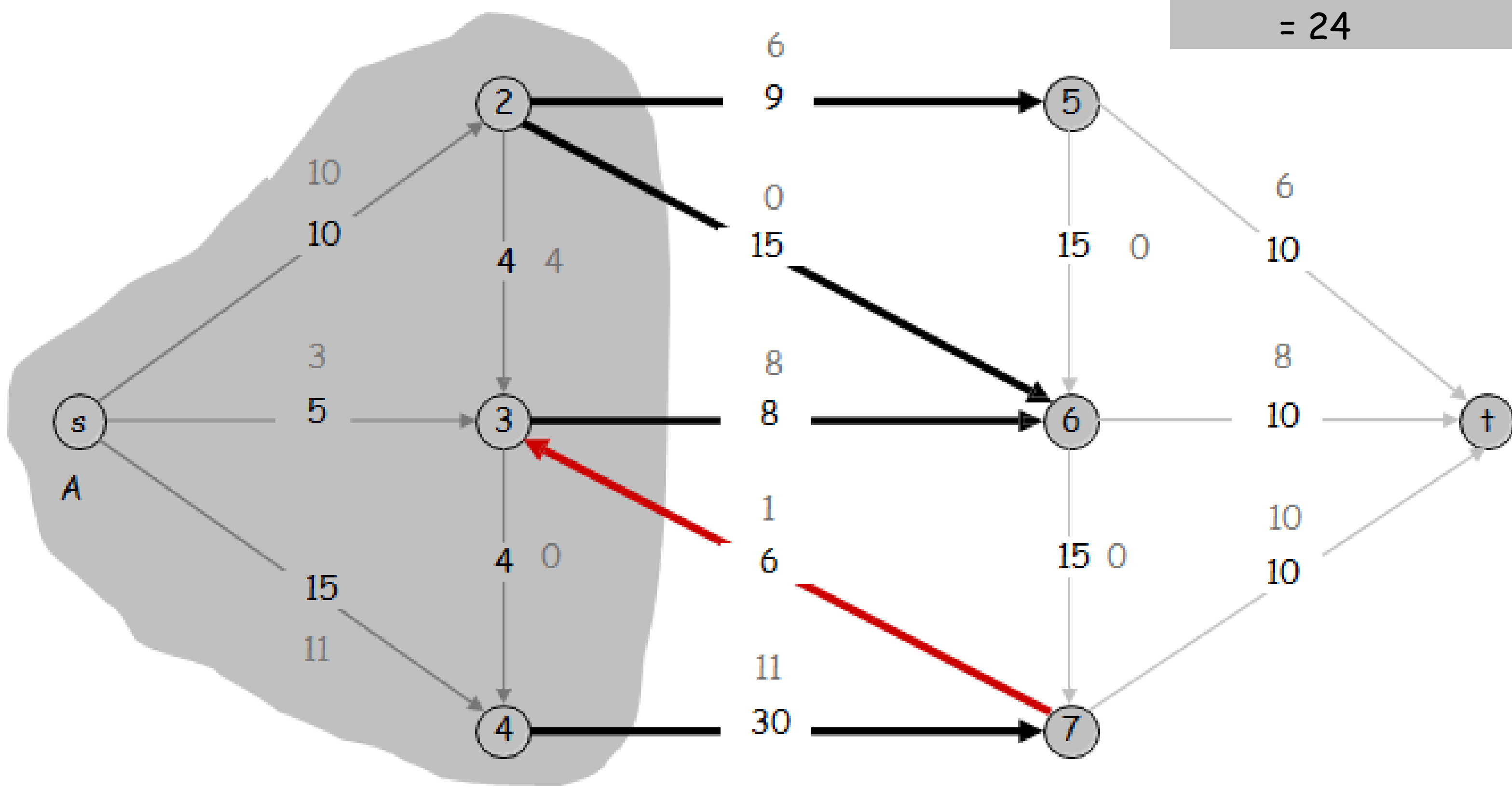


Flows and Cuts

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.

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

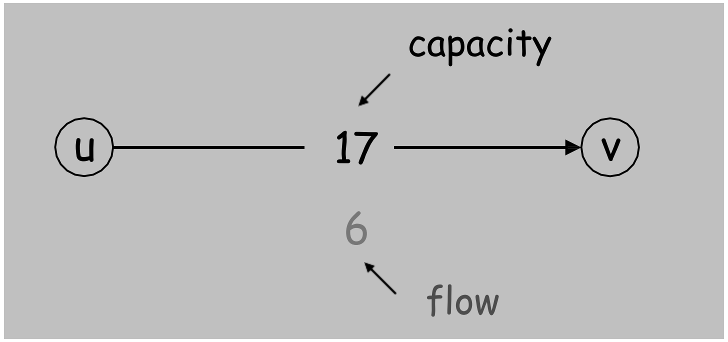
Value = 6 + 0 + 8 - 1 + 11
= 24



Residual Graph

Original edge: $e = (u, v) \in E$

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

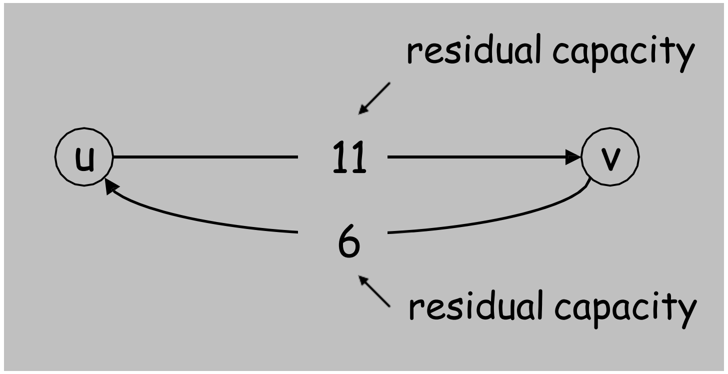


Residual edge. "Undo" flow sent

- $e = (u, v)$ 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

$E_f = \{e : f(e) < c(e)\} \cup \{e^R : f(e) > 0\}$

Augmenting Path Algorithm

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

forward edge

reverse edge

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

while (there exists augmenting path P) {
f ← Augment(f, c, P)
update Gf
}
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 cut.

Pf. We prove both simultaneously by showing TFAE (the following are equivalent):

(i) There exists a cut (A, B) such that v(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.

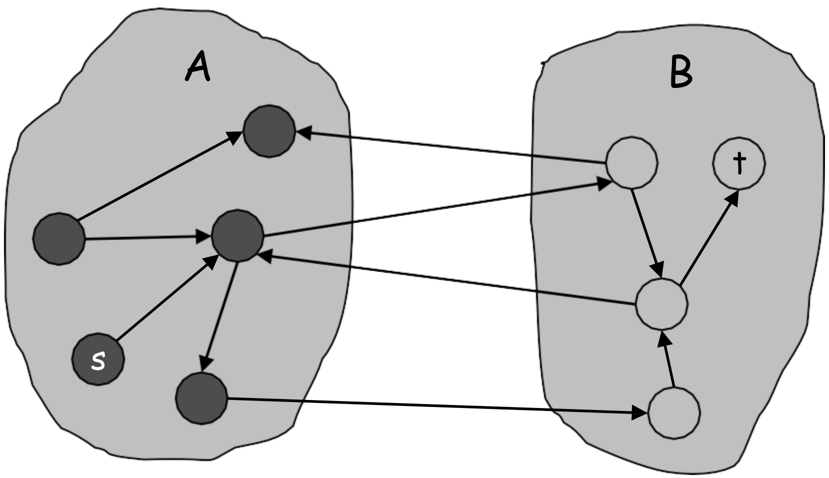
(ii) $\Rightarrow$ (iii) We show contrapositive.
Let f be a flow. If there exists an augmenting path, then we can improve f by sending flow along path.

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, $t \notin A$.

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



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 $v(f^*) \leq nC$ iterations.
Pf. Each augmentation increase value by at least 1. ▪

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.