Max Flow and Min Cut

- Minimum cut
- Maximum flow
- Max-flow min-cut theorem

Max Flow and Min Cut

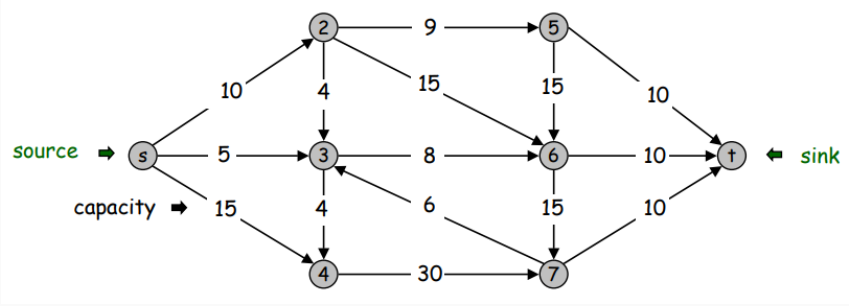
- Minimum cut
- Maximum flow
- Max-flow min-cut theorem
- Ford-Fulkerson augmenting path algorithm
- Edmonds-Karp heuristics
- Bipartite matching

Maximum Flow and Minimum Cut

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

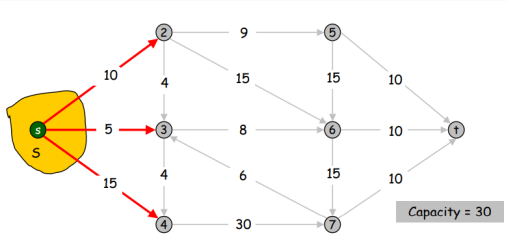
Minimum Cut Problem

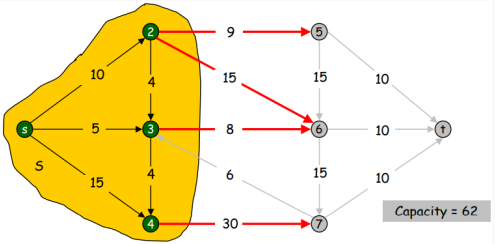
- Network: abstraction for material flowing through the edges
  - Directed graph
  - Capacities on edges
  - Source node s, sink node t
- Problem: delete “best” set of edges to disconnect t from s



Cuts

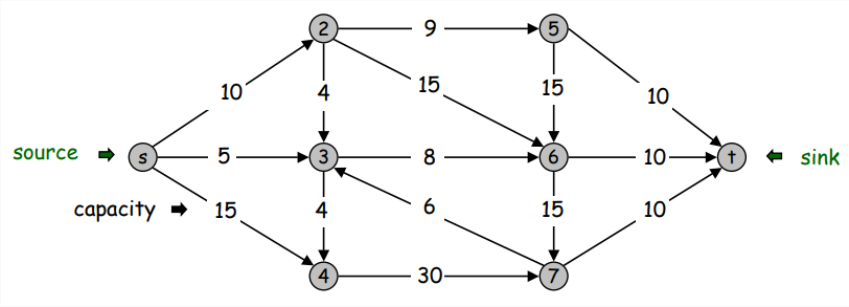
- A cut is a node partition (S, T) such that $s \in S$ and $t \in T$
- capacity of partition: sum of weights of edges leaving S
- Min cut problem: find an $s - t$ cut of minimum capacity





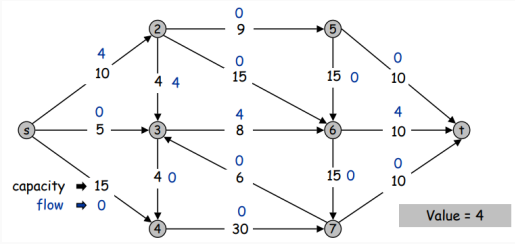
Maximum Flow Problem

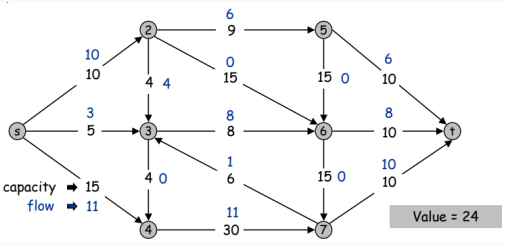
- Network: abstraction for material flowing through the edges
  - Directed graph
  - Capacities on edges
  - Source node s, sink node t
- Problem: assign flow to edges so as to:
  - Equalize inflow and outflow at every intermediate vertex
  - Maximize flow sent from s to t



Flows

- A flow f is an assignment of weights to edges so that
  - Capacity: $0 \leq f(e) \leq u(e)$
  - Flow conservation: flow leaving v = flow entering v, except s, t
- Max flow problem: find flow that maximizes net flow into sink

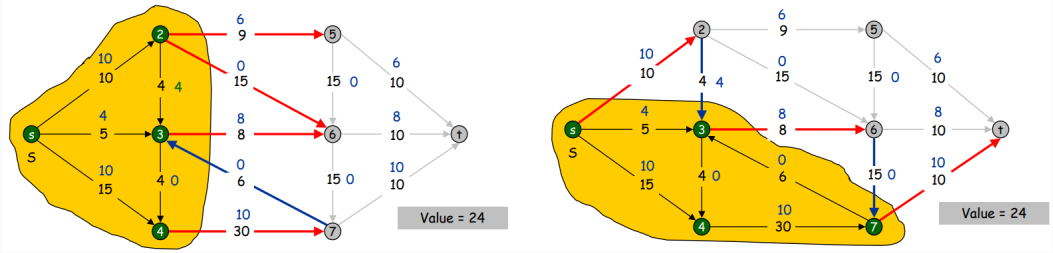




Flows and Cuts

Observation 1

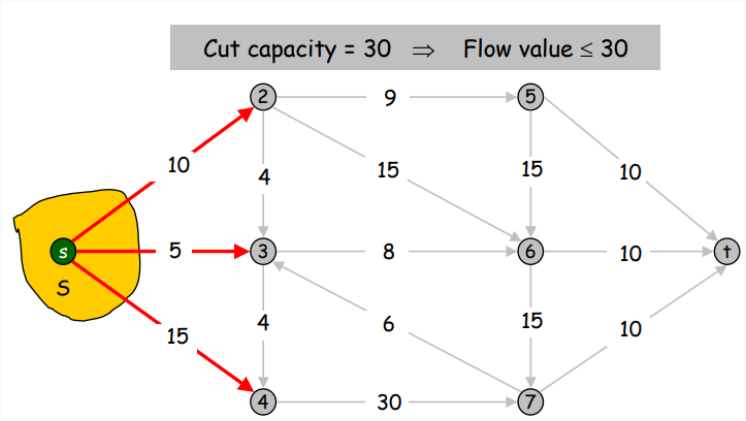
Let f be a flow, and (S, T) be any $s - t$ cut. Then, the net flow sent across the cut is equal to the amount reaching t



Flows and Cuts

Observation 2

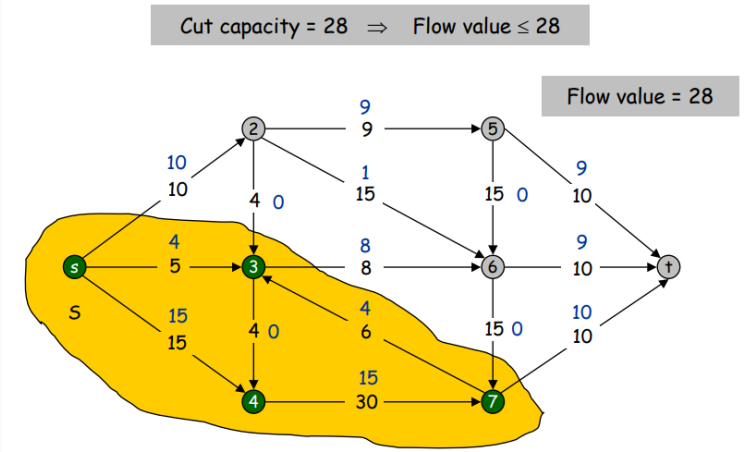
The value of the flow is at most the capacity of the cut



Flows and Cuts

Observation 3

Let f be a flow, (S, T) be an $s - t$ cut whose capacity equals the value of f. Then f is a max flow and (S, T) is a min cut

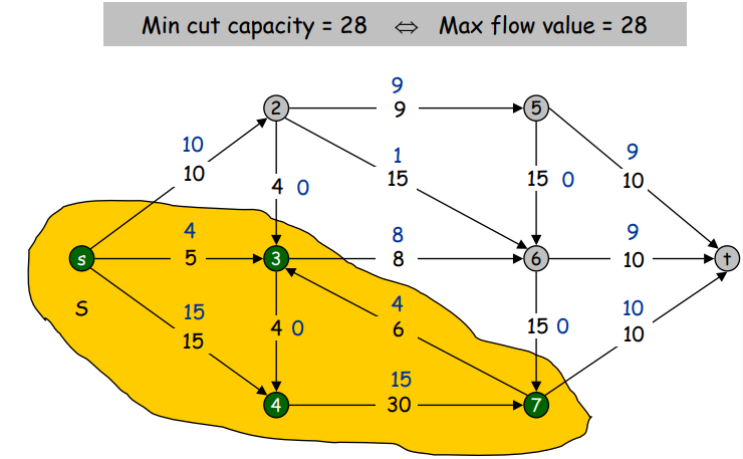


Max-Flow Min-Cut Theorem

- Max-flow min-cut theorem (Ford-Fulkerson, 1956): In any network, the value of max flow equals capacity of min cut

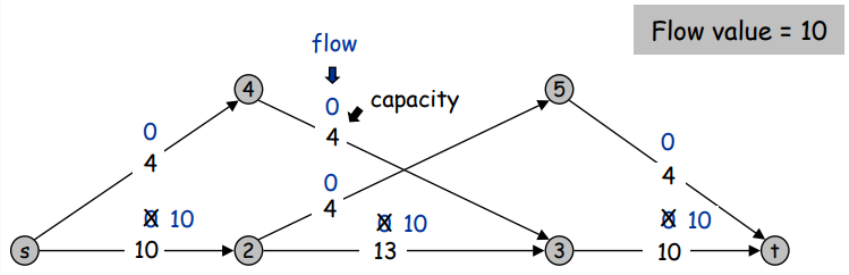
Max-Flow Min-Cut Theorem

- Max-flow min-cut theorem (Ford-Fulkerson, 1956): In any network, the value of max flow equals capacity of min cut
- Proof: we find flow and cut such that Observation 3 applies



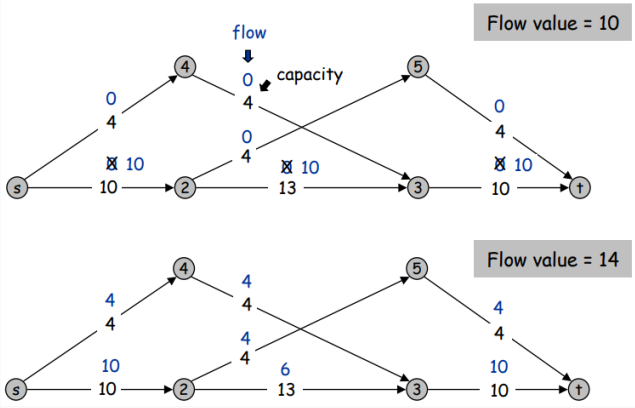
Towards an Algorithm

- Find $s - t$ path where each arc has $f(e) < u(e)$ and "augment" flow along it
- Repeat until you get stuck



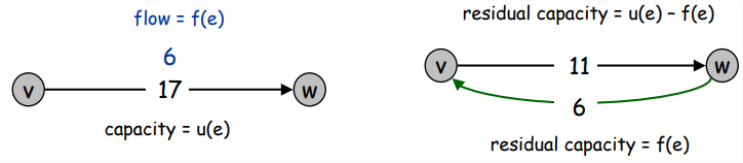
Towards an Algorithm

- Find $s - t$ path where each arc has $f(e) < u(e)$ and "augment" flow along it
- Repeat until you get stuck
- Need to be able to "backtrack"



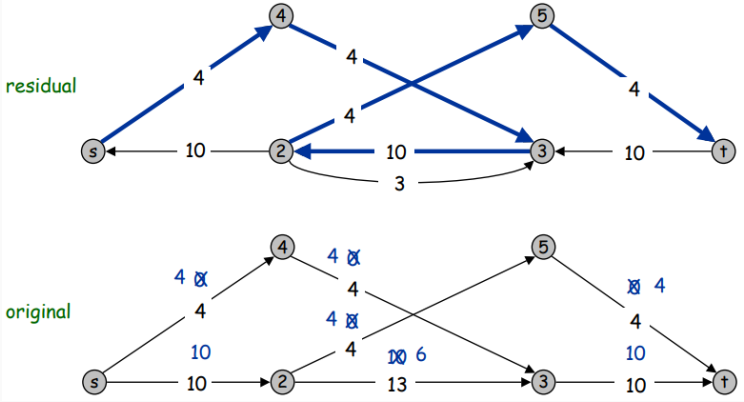
Residual Graph

- Original graph
  - Flow $f(e)$
  - Edge $e = v -> w$
- Residual edge
  - Edge $e = v -> w$ or $w -> v$
  - "Undo" flow sent
- Residual graph
  - All the edges that have strictly positive residual capacity



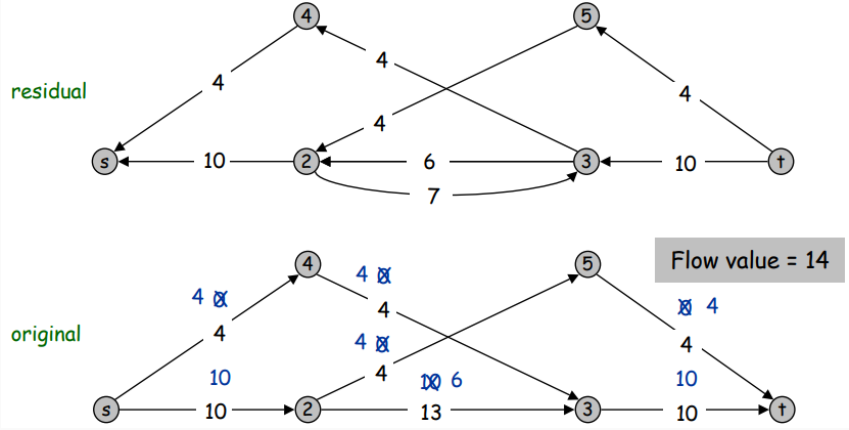
Augmenting Paths

- Augmenting path = path in residual graph
- Increase flow along forward edges
- Decrease flow along backward edges



Augmenting Paths

- Observation 4. If augmenting path, then not yet a max flow.
- If no augmenting path, is it a max flow?



Ford-Fulkerson Augmenting Path Algorithm

- while there exists an augmenting path
  - Find augmenting path P
  - Compute bottleneck capacity of P
  - Augment flow along P

Max-Flow Min-Cut Theorem

Augmenting path theorem

A flow f is a max flow if and only if there are no augmenting paths

Max-flow min-cut theorem

The value of the max flow equals the capacity of the min cut

Proof

We prove the following are equivalent

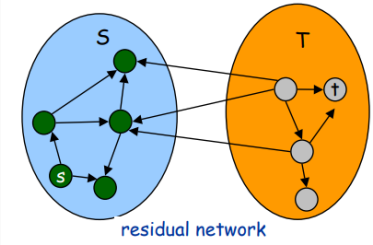
- f is a max flow
- There is no augmenting path relative to f
- There exists a cut whose capacity equals the value of f

Max-Flow Min-Cut Theorem Proof

- (1)->(2) equivalent to not (2)->not (1): Observation 4
- (3)->(1): Observation 3

Max-Flow Min-Cut Theorem Proof

- (1)->(2) equivalent to not (2)->not (1): Observation 4
- (3)->(1): Observation 3
- (2)->(3): Let f be a flow with no augmenting paths
  - Let S be set of vertices reachable from s in residual graph
    - S contains s; since no augmenting paths, S does not contain t
    - all edges e leaving S in original network have $f(e) = u(e)$
    - all edges e entering S in original network have $f(e) = 0$
    - $|f| = \sum_{e \in S_1} f(e) - \sum_{e \in S_2} f(e) = \sum_{e \in S_1} u(e) = capacity(S, T)$



Ford-Fulkerson Algorithm: Analysis

All capacities are integers between 1 and U

Theorem

The algorithm terminates in at most $|f^*| = VU$ iterations

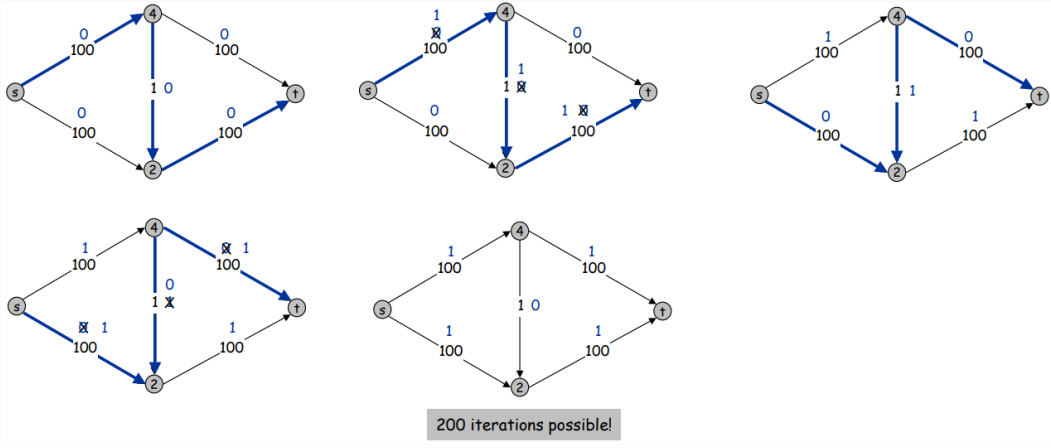
Corollary

If $U = 1$, then algorithm runs in at most V iterations

Integrality theorem

If all arc capacities are integers, then there exists a max flow for which every flow value is an integer

Choosing Good Augmenting Paths



Choosing Good Augmenting Paths

Use care when selecting augmenting paths

- Some choices lead to exponential algorithms
- Clever choices lead to polynomial algorithms
- Optimal choices for real world problems ?

Design goal is to choose augmenting paths so that

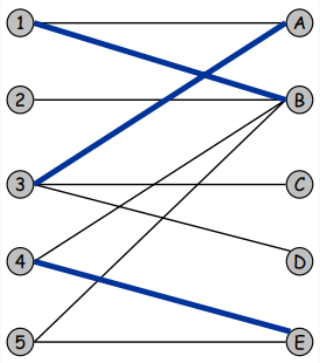
- Can find augmenting paths efficiently
- Few iterations

Edmonds-Karp heuristic: choose augmenting path with:

- Fewest number of arcs
- Max bottleneck capacity

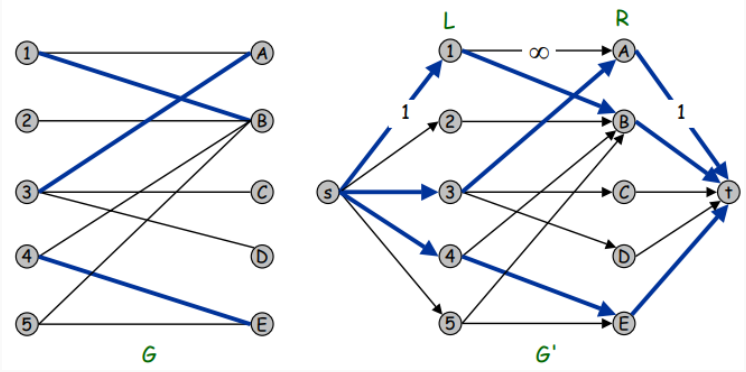
Application: Bipartite Matching

- Input: undirected and bipartite graph G
- Set of edges M is a matching if each vertex appears at most once
- Max matching: find a max cardinality matching



Reduce to max flow

- Create a directed graph G'
- Direct all arcs from Left to Right, give infinite (or unit) capacity
- Add source s, and unit capacity arcs from s to each node in Left
- Add sink t, and unit capacity arcs from each node in Right to t



Bipartite Matching: Proof of Correctness

- Matching in G of cardinality k induces flow in G' of value k
- Flow f of value k in G' induces matching of cardinality k in G
  - By integrality theorem, there exists 0-1 valued flow f of value k
  - Consider M = set of edges from L to R with $f(e) = 1$
    - each node in L and R incident to at most one edge in M; $|M| = k$