EE360C: Algorithms

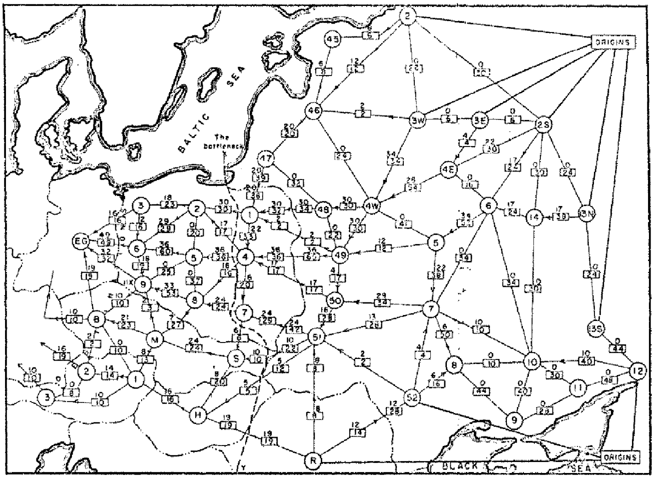
Network Flow

Spring 2018

Department of Electrical and Computer Engineering
University of Texas at Austin

Introduction

The Soviet Rail Network (1955)



On the history of the transportation and maximum flow problems. Alexander Schrijver, Math Programming, 2002.

Maximum Flow and Minimum Cut

Max flow and min cut

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

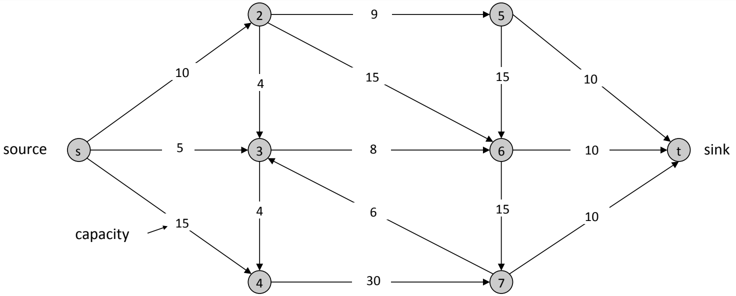
Nontrivial applications/reductions

- Data mining
- Open pit mining
- 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

A Flow Network

- An abstraction for material flowing through the edges
- $G = (V, E)$ is a directed graph with no parallel edges
- There are two distinguished nodes: a source (s) and a sink (t)
- $c(e)$ is the capacity of edge e

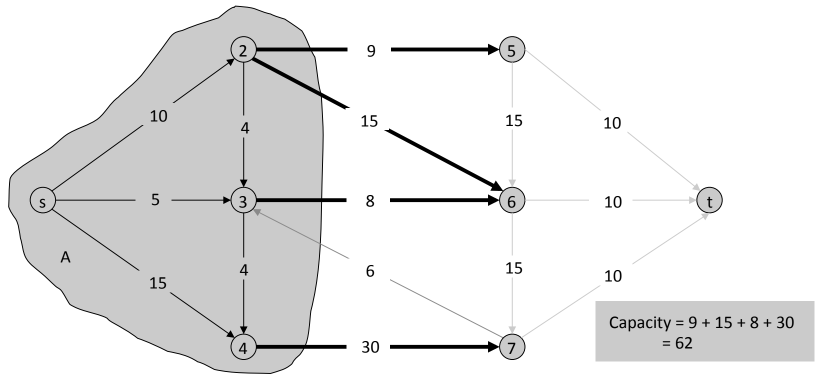


Definition

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

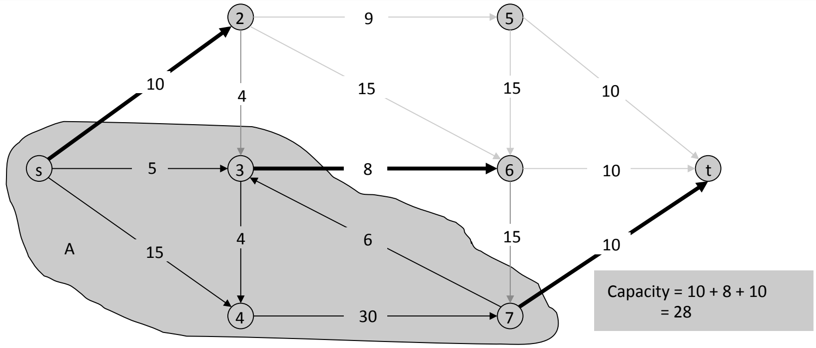
Definition

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



The Min s-t Cut Problem

Find an s-t cut of minimum capacity.



Flows

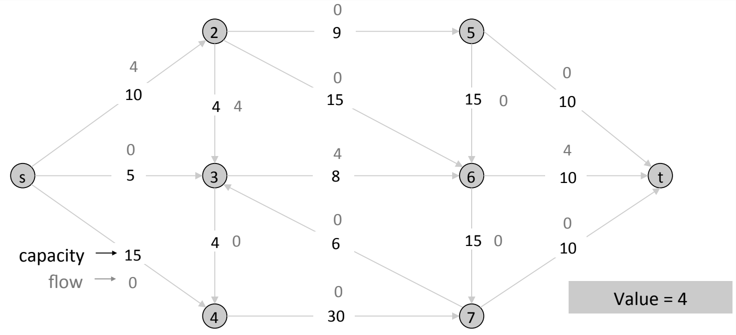
Definition

An *s*-*t* flow is a function that satisfies:

- Capacity: for each $e \in E$: $0 \leq f(e) \leq c(e)$
- Conservation: for each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$

Definition

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



Flows

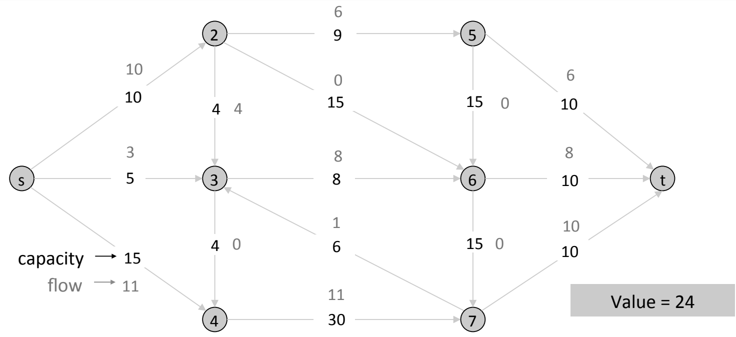
Definition

An *s*-*t* flow is a function that satisfies:

- Capacity: for each $e \in E$: $0 \leq f(e) \leq c(e)$
- Conservation: for each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$

Definition

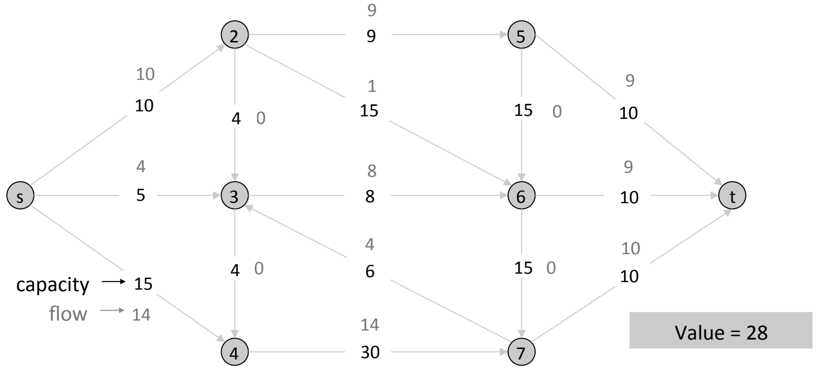
The value of a flow f is: $v(f) = \sum_{e \text{ out of } s} f(e)$.



Maximum Flow

The Max Flow Problem

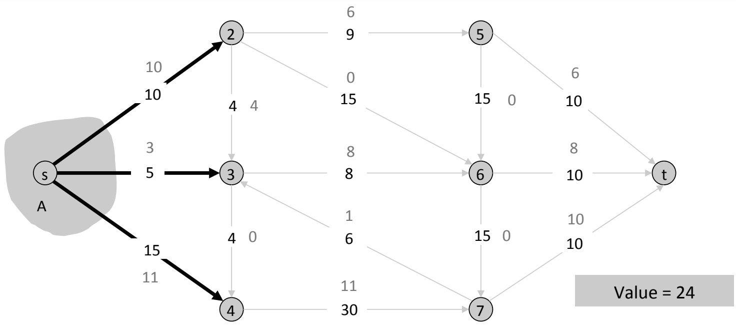
Find the s-t flow of maximum value.



Flow Value Lemma

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

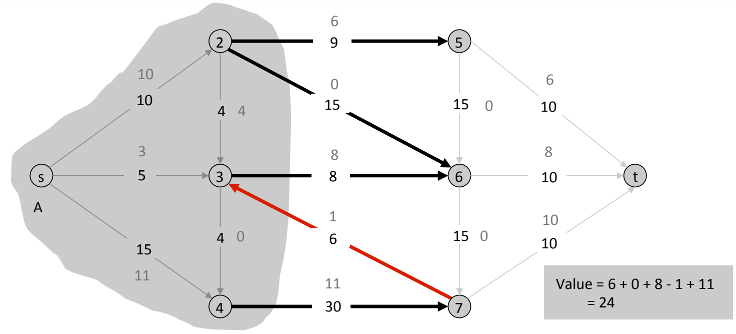
$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$



Flow Value Lemma

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

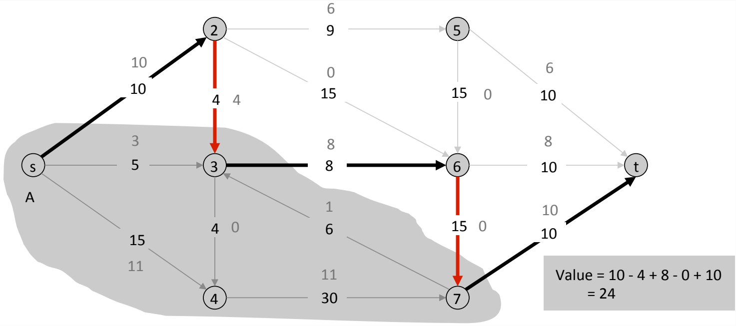
$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$



Flow Value Lemma

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

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



Flows and Cuts

Flow Value Lemma

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

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

Proof

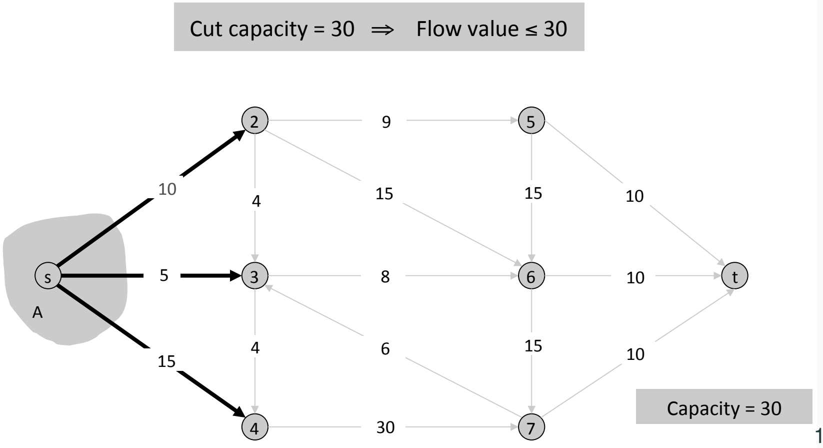
$$\begin{aligned} v(f) &= \sum_{e \text{ out of } s} f(e) \\ &= \sum_{v \in A} \Big(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) \Big) \\ &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \end{aligned}$$

In the second step, by flow conservation, all terms except $v = s$ are 0. In the third step, f_{uw} cancels for $u, w \in A$.

Flows, Cuts, and Capacity

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.

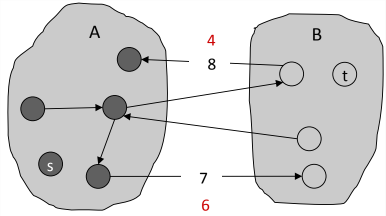


Flows, Cuts, and Capacity

Weak Duality

Let f be any flow. Then for any s-t cut (A, B), we have $v(f) \leq cap(A, B)$.

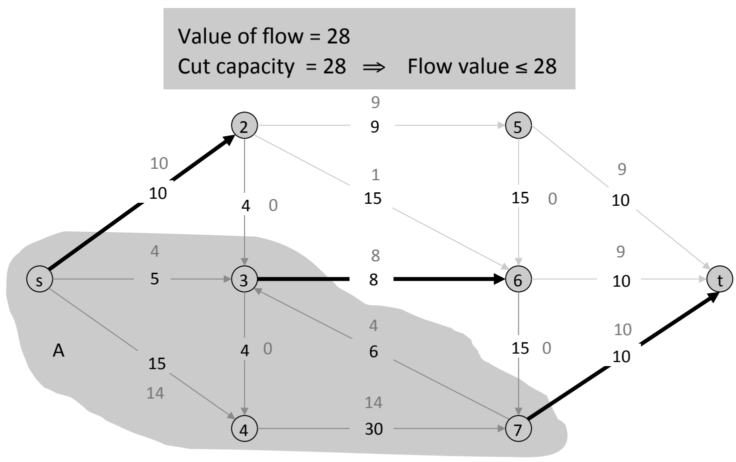
$$
\begin{aligned}
v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\
&\leq \sum_{e \text{ out of } A} f(e) \\
&\leq \sum_{e \text{ out of } A} c(e) \\
&= cap(A, B)
\end{aligned}
$$



Certificate of Optimality

Corollary

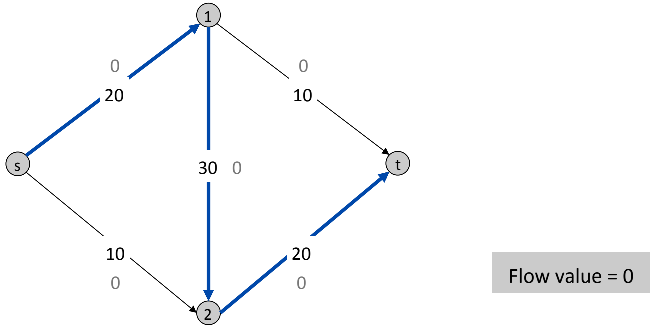
Let f be any flow and let (A, B) be any cut. If $v(f) = cap(A, B)$, then f is a max flow and (A, B) is a min cut.



Towards a Max Flow Algorithm

Greedy Algorithm

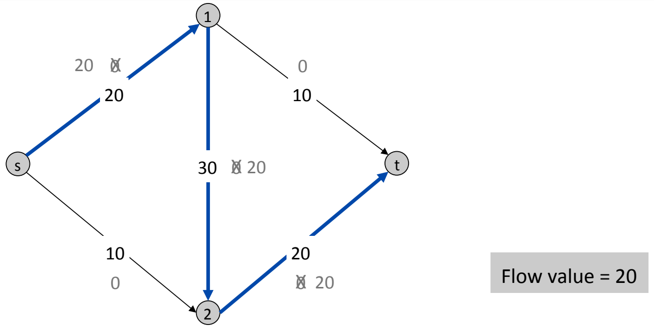
- Start with $f(e) = 0$ for all edges $e \in E$
- Find an s-t path P where each edge has $f(e) < c(e)$.
- Augment flow along path P
- Repeat until you get stuck



Towards a Max Flow Algorithm

Greedy Algorithm

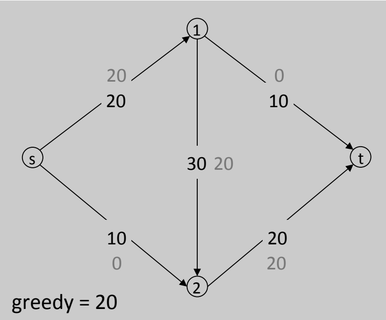
- Start with $f(e) = 0$ for all edges $e \in E$
- Find an s-$t$path P where each edge has $f(e) < c(e)$.
- Augment flow along path P
- Repeat until you get stuck

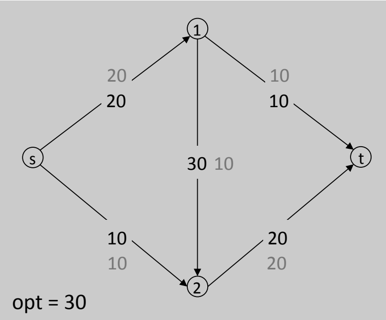


Towards a Max Flow Algorithm

Greedy Algorithm

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

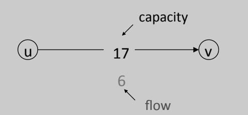




Residual Graph

Original Edge

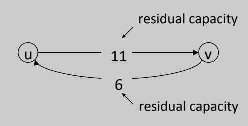
- $e = (u, v) \in E$; Flow $f(e)$; capacity $c(e)$



Residual Edge

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

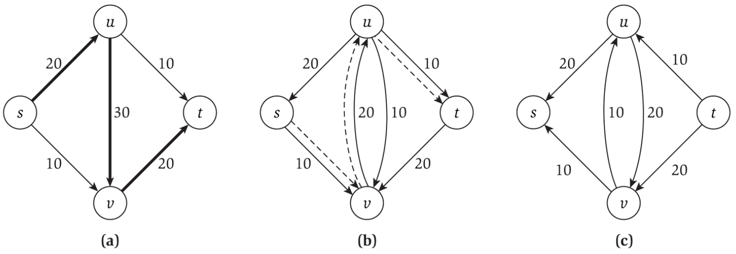
$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$



Residual Graph: $G_f = (V, E_f)$

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

Residual Graph Example

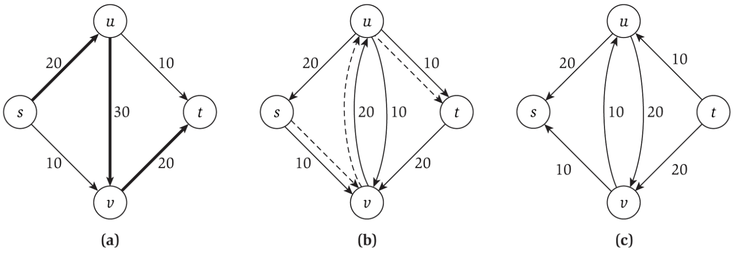


(a) G with 20 units of flow on the path s-u-v-t

(b) The resulting residual graph and the new augmenting path

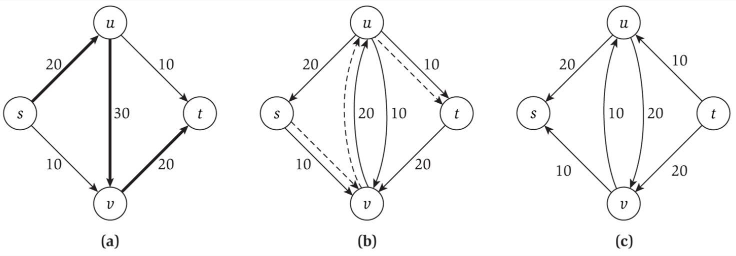
(c) The residual graph after an additional 10 units of flow on the path s-v-u-t

Augmenting Paths in a Residual Graph



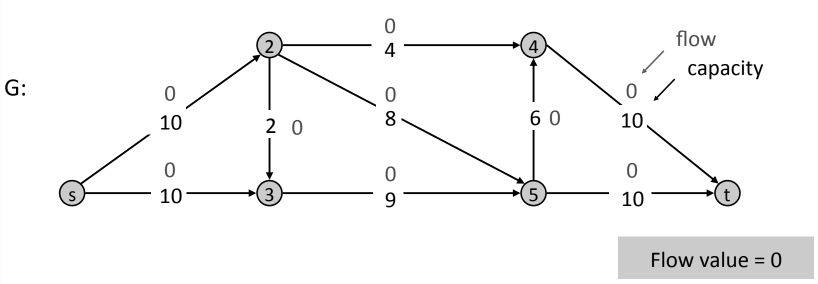

```
augment(f, P)
  Let b = bottleneck(P, f)
  For each edge (u, v) ∈ P
    If e = (u, v) is a forward edge then
      increase f(e) in G by b
    Else ((u, v) is a backward edge, and let e = (v, u))
      decrease f(e) in G by b
    Endif
  Endfor
  Return(f)
```

The Ford-Fulkerson Algorithm

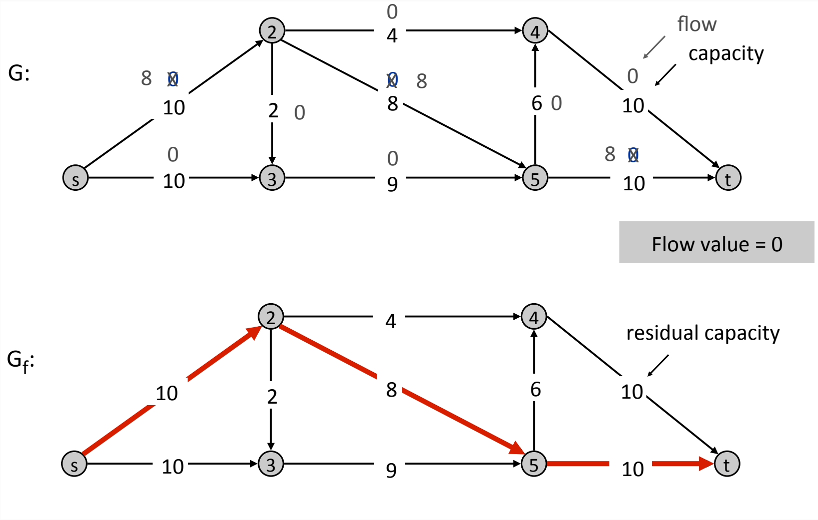



```
Max-Flow
  Initially f(e) = 0 for all e in G
  While there is an s-t path in the residual graph G_f
    Let P be a simple s-t path in G_f
    f' = augment(f, P)
    Update f to be f'
    Update the residual graph G_f to be G_{f'}
  Endwhile
  Return f
```

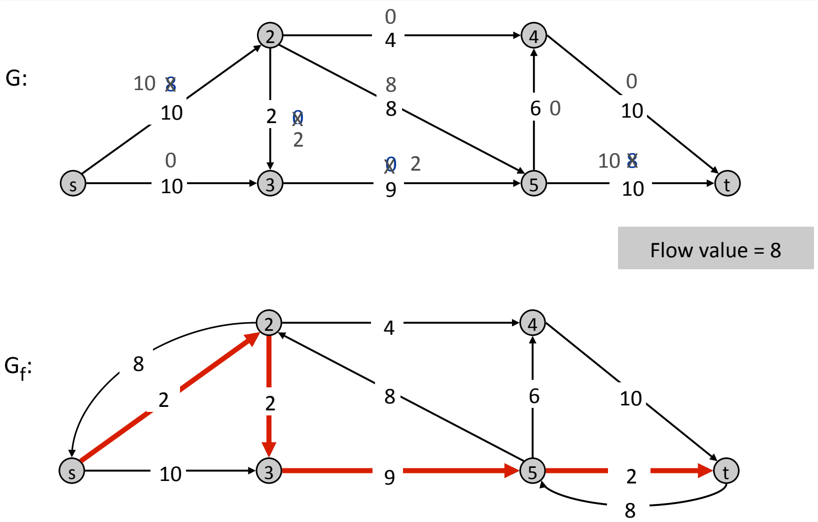
Ford-Fulkerson: An Example



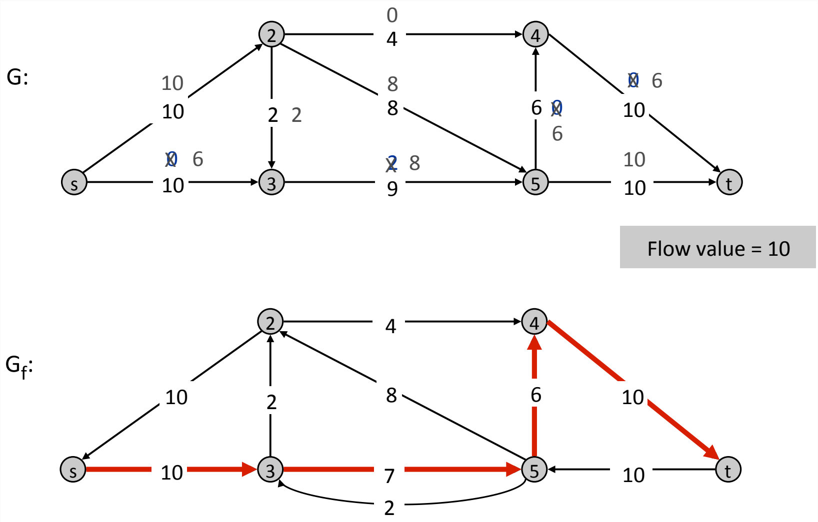
Ford-Fulkerson: An Example



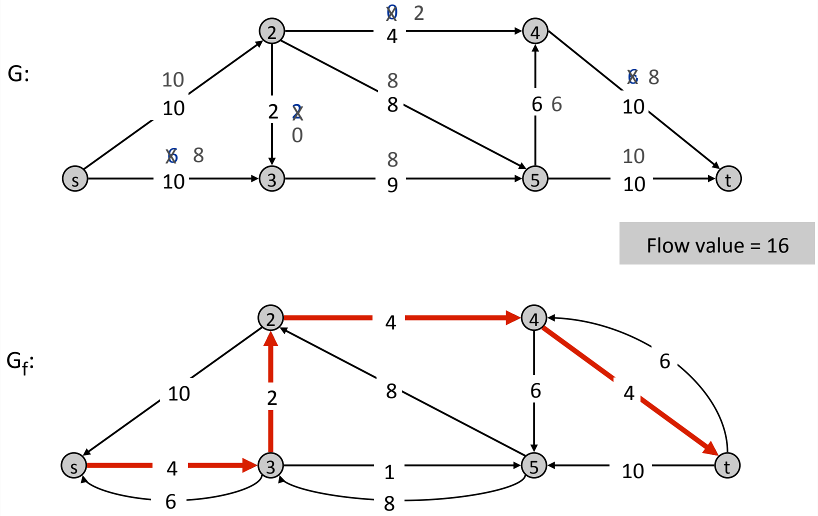
Ford-Fulkerson: An Example



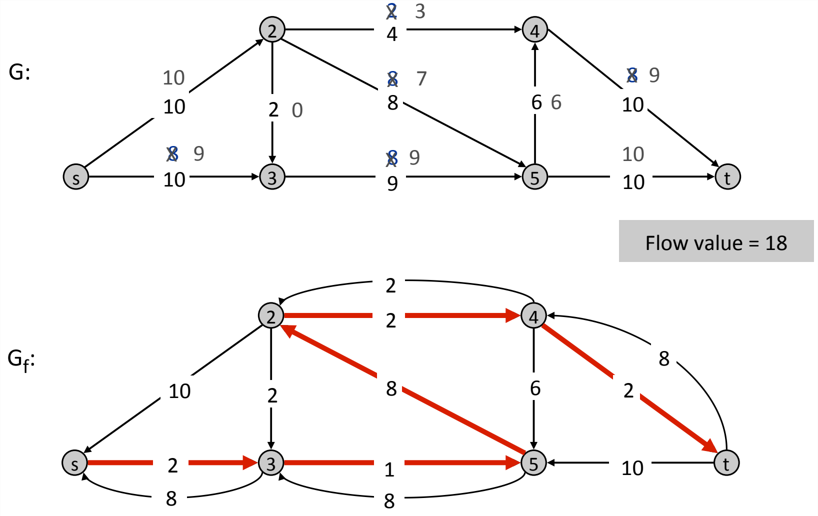
Ford-Fulkerson: An Example



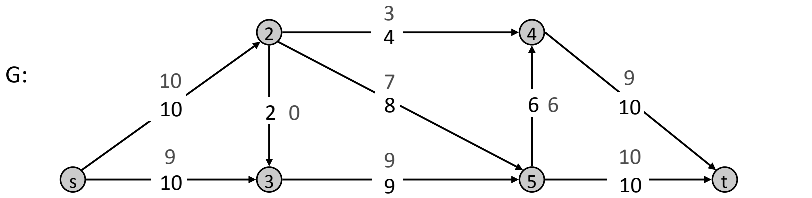
Ford-Fulkerson: An Example

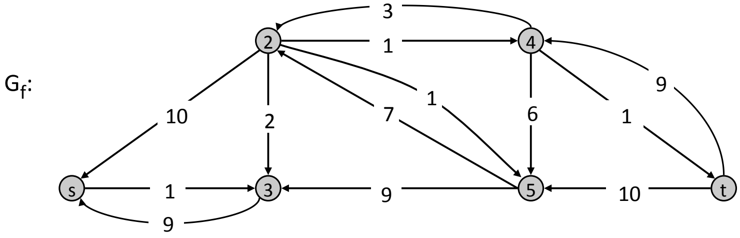


Ford-Fulkerson: An Example



Ford-Fulkerson: An Example





Max-Flow Min-Cut Theorem

Augmenting Path Theorem

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 is equal to the value of the min cut.

Proof Strategy

We can prove both theorems simultaneously by proving that the following are equivalent.

1. There exists a cut (A, B) such that $v(f) = cap(A, B)$.
2. Flow f is a max flow.
3. There is no augmenting path relative to f.

Max-Flow Min-Cut Theorem Proof

Proof of Equivalences

- "There exists a cut (A, B) such that $v(f) = cap(A, B)$" therefore "Flow f is a max flow" by the corollary to the weak duality lemma.
- "Flow f is a max flow" therefore "There is no augmenting path relative to f" by:
  - Let f be a flow. If there exists an augmenting path, then we can improve f by sending flow along this path, contradicting the fact that f is a max flow (and that (A, B) is a min cut).

Max-Flow Min-Cut Proof (cont.)

Proof of Equivalences (cont.)

- "There is no augmenting path relative to f" therefore "There exists a cut (A, B) such that $v(f) = cap(A, B)$" by:
  - Let f be a flow with no augmenting paths.
  - Let A be a set of vertices reachable from s in the residual graph.
  - By definition of A, $s \in A$; by definition of t, $t \notin A$.

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

Running Time

Invariant

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

Assumption

Let $C = \sum_{e \text{ out of } s} c_e$. Then, $v(f) \leq C$ for all s-t flows f.

Theorem

The algorithm terminates in at most C iterations of the while loop.

Proof: Each augmentation increases the value by at least 1.

Corollary

Ford-Fulkerson runs in $O(mC)$ time. Proof: At most C iterations, each iteration (finding an augmenting path) is $O(m)$ (how?).

Integrality Theorem

If all capacities are integers, then there exists a max flow f for which every flow value $f(e)$ is an integer.

Proof: Since the algorithm terminates, this follows from the invariant.

Exercise

You are given a directed graph $G = (V, E)$ with positive integer capacities on each edge, a designated source (s), and a designated sink (t). You are given an integer max flow in G defined by an f_e on each edge.

I choose one edge $e \in E$ and increase its capacity by 1. Show how to find a max flow in the resulting graph (G') in $O(m + n)$ time.

Hint: first prove that the max flow in G' is either the same as in G or one more than the max flow in G.

Augmenting Paths

Exponential Number of Augmentations

Question

Is the (generic) Ford-Fulkerson algorithm polynomial in the input size?

Answer

No. It's also polynimal in C, the max capacity on a link. (Think knapsack.) In such a case, the algorithm can take C iterations (on a pathological input, sure, but asymptotically, the time is still proportional to C).

Choosing Good Augmenting Paths

Use care when selecting augmenting paths

- Some choices lead to exponential algorithms
- Clever choices lead to polynomial algorithms
- If capacities are irrational, the algorithm is not guaranteed to terminate!

Goal: choose augmenting paths so that...

- Can find augmenting paths efficiently
- Results in few iterations of the while loop

Choose augmenting paths with...

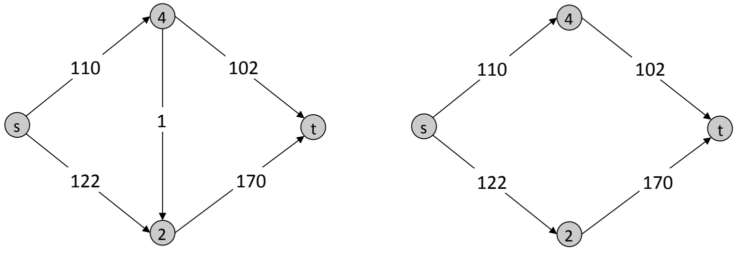
- Max bottleneck capacity
- Sufficiently large bottleneck capacity
- Fewest number of edges

Capacity Scaling

Intuition

Choosing the path with the highest bottleneck capacity increases the flow by the maximum possible amount

- Don't worry about finding the exact highest bottleneck path
- Instead, maintain a scaling parameter Δ
- Let $G_f(\Delta)$ be the subgraph of the residual graph consisting of only edges with capacity at least Δ



Capacity Scaling (cont.)

```
Scaling Max-Flow
  Initially f(e) = 0 for all e in G
  Initially set Δ to be the largest power of 2 that is no larger
        than the maximum capacity out of s: Δ ≤ max_{e out of s} c_e
    While Δ ≥ 1
       While there is an s-t path in the graph G_f(Δ)
          Let P be a simple s-t path in G_f(Δ)
          f' = augment(f, P)
          Update f to be f' and update G_f(Δ)
       Endwhile
       Δ = Δ/2
    Endwhile
Return f
```

Capacity Scaling: Correctness

Assumption

Let $C = \sum_{e \text{ out of } s} c_e$.

Integrality invariant

All flow and residual capacity values are integral.

Correctness

If the algorithm terminates, then f is a max flow.

Proof

- By the integrality invariant, when $\Delta = 1$, $G_f(\Delta) = G_f$
- Upon termination of the $\Delta = 1$ phase, there are no augmenting paths.

Capacity Scaling: Running Time

Lemma 1

The outer while loop repeats $1 + \lceil \log_2 C \rceil$ times

Proof: Initially Δ is at most C. Δ decreases by a factor of 2 each iteration of the outer while loop and never goes below 1.

Lemma 2

Let f be the flow at the end of a Δ-scaling phase. Then the value of the maximum flow is at most $v(f) + m\Delta$. (Proof on next slide.)

Lemma 3

There are at most $2m$ augmentations per scaling phase.

- Let f be the flow at the end of the previous scaling phase (2Δ).
- Lemma 2 tells us that $v(f^*) \le v(f) + m(2\Delta)$
- Each augmentation in a Δ-phase increases $v(f)$ by at least Δ

Theorem

The scaling max-flow algorithm finds a max flow in $O(m \log_2 C)$ augmentations. It can be implemented to run in $O(m^2 \log_2 C)$ time.

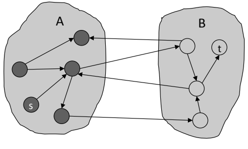
Capacity Scaling: Running Time

Lemma 2

Let f be the flow at the end of a Δ-scaling phase. Then the value of the max flow is at most $v(f) + m\Delta$. The following proof is almost identical to the max-flow min-cut theorem.

- We show that at the end of a Δ-phase, there exists a cut (A, B) such that $cap(A, B) \leq v(f) + m\Delta$.
- Choose A to be the set of nodes reachable from s in $G_f(\Delta)$. Then, $s \in A$ and $t \notin A$ (because it is the end of the Δ phase).
- For edge $e = (u, v) \in G$ with $u \in A$ and $v \in B$, $f(e) + \Delta > c(e)$.
- For edge $e = (u, v) \in G$ with $u \in B$ and $v \in A$, $f(e) < \Delta$, else there is a reverse edge $e' = (v, u) \in G_f(\Delta)$ with $f(e') \geq \Delta$.

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



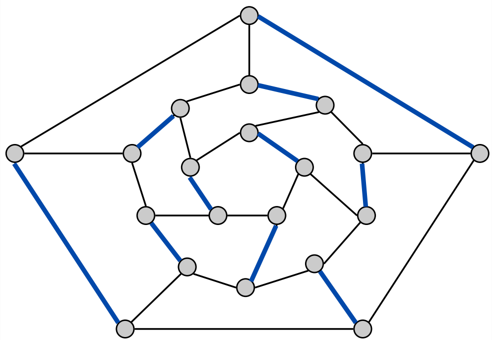
original network

Bipartite Matching

Matching

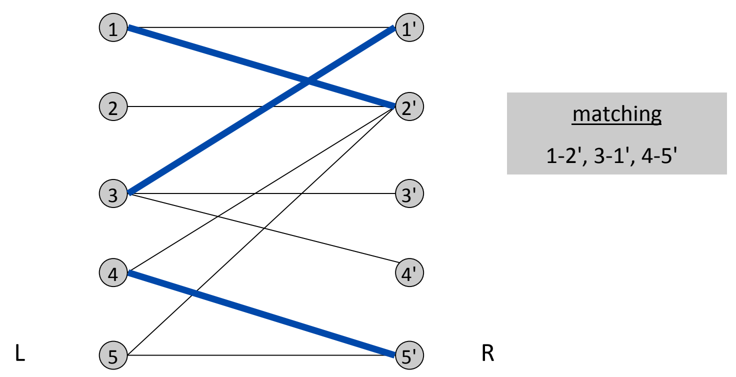
Matching

- Input: undirected graph $G = (V, E)$
- $M \subseteq E$ is a matching if each node appears in at most one edge in M
- Max matching: find a max cardinality matching



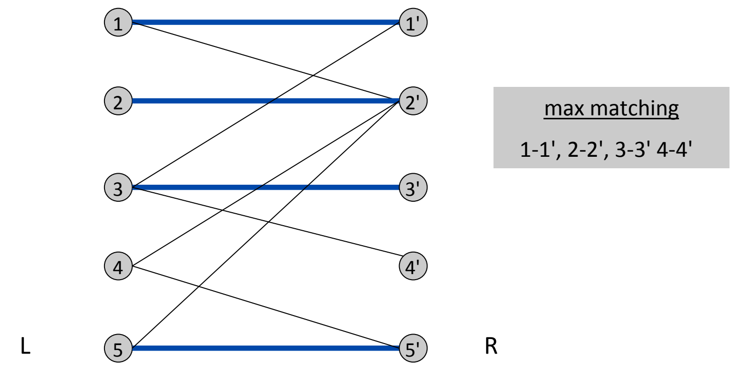
Bipartite Matching

- Input: undirected, bipartite graph $G = (L \cup R, E)$
- $M \subseteq E$ is a matching if each node appears in at most one edge
- Max matching: find a max cardinality matching



Bipartite Matching

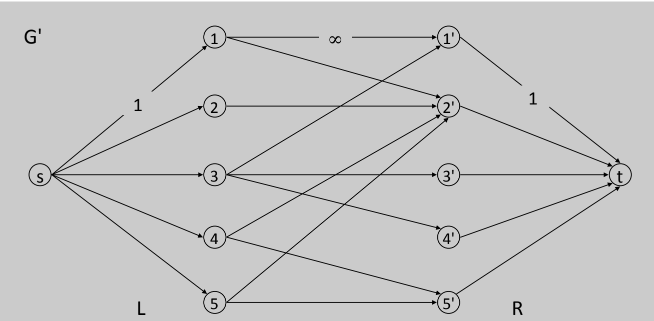
- Input: undirected, bipartite graph $G = (L \cup R, E)$
- $M \subseteq E$ is a matching if each node appears in at most one edge
- Max matching: find a max cardinality matching



Bipartite Matching

Max flow formulation

- Create directed graph $G' = (L \cup R \cup \{s, t\}), E')$.
- Direct all edges from L to R and assign infinite (or unit) capacities.
- Add source s and unit capacity edges from s to each node in L.
- Add sink t and unit capacity edges from each node in R to t.

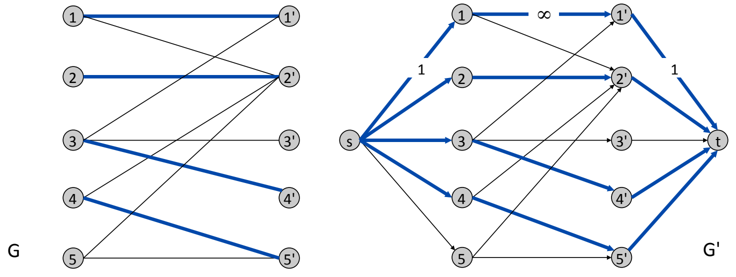


Bipartite Matching: Correctness

Theorem

The max cardinality of a matching in G equals the value of the max flow in G'

- Given a max matching M of cardinality k
- Consider a flow f that sends one unit along each of k paths.
- f is a flow and has cardinality k

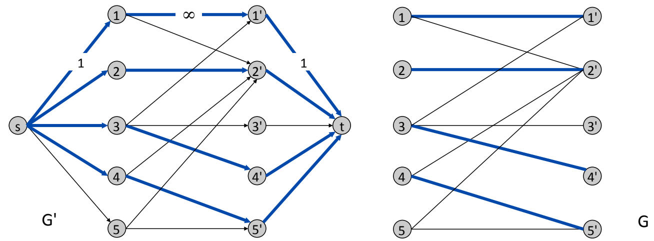


Bipartite Matching: Correctness (2)

Theorem

The max cardinality of a matching in G is the value of the max flow in G'

- Let f be a max flow in G' of value k
- The integrality theorem gives us that k is integral, and we can assume $f(e)$ is 0 or 1 for all e
- Consider M as the set of edges from L to R with $f(e) = 1$.
    - Each node in L and R participates in at most one edge in M (each has $c(e) = 1$ on a single input edge).
    - $|M| = k$. Consider the cut $(L \cup s, R \cup t)$.



Perfect Matching

Definition

A matching $M \subseteq E$ is perfect if each node appears in *exactly* one edge in M.

Question

When does a bipartite graph have a perfect matching?

Structure of bipartite graphs with perfect matchings

- Clearly, we must have $|L| = |R|$.
- What other conditions are necessary?
- What other conditions are sufficient?

Perfect Matching

Notation

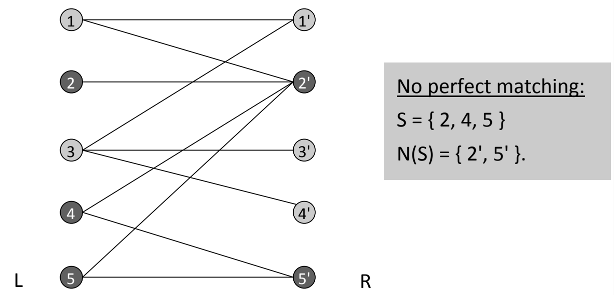
Let S be a subset of nodes and let $N(S)$ be the set of nodes adjacent to nodes in S.

Observation

If a bipartite graph $G = (L \cup R, E)$ has a perfect matching, then $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

Proof

Each node in S has to be matched to a different node in $N(S)$.



Marriage Theorem

Marriage Theorem (Frobenius 1917, Hall 1935)

Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$. Then G has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

The proof in the forward direction is the same as the previous observation. For the proof in the reverse direction...

Proof of the Marriage Theorem

Proof

Suppose G does not have a perfect matching. Then we need to show that there exists a set S such that $|N(S)| < |S|$.

- Consider a set $S \subset L$.
- Let (A, B) be a min cut in G'.
- Select (A, B) such that $S \subseteq A$ and that $N(S) \subseteq A$. How? Consider any $x \in S$ that is connected to a $y \notin A$ such that $c(x, y) = 1$. We can move y from B to A without changing the (total) capacity of the cut (because, by definition, y is also connected to t).
- By the max-flow min-cut theorem and the fact that the max flow is less than $|L|$, $cap(A, B) < |L|$.
- Because all of the edges that cross the cut must either leave s and go to some node in L but not in A or leave A and go to t, $cap(A, B) = |L \cap B| + |R \cap A|$.
- Consider $S = L \cap A$. $N(S) \subseteq A$ (by construction of A).
- $|L \cap B| = |L| - |S|$ and $|R \cap A| \geq |N(S)|$
- Putting it all together:
  $cap(A, B) = |L \cap B| + |R \cap A| \geq |L| - |S| + |N(S)|$ then
  $|L| - |S| + |N(S)| \leq cap(A, B) < |L|$ or
  $|L| - |S| + |N(S)| < |L|$, so $|N(S)| < |S|$

Disjoint Paths

Edge Disjoint Paths

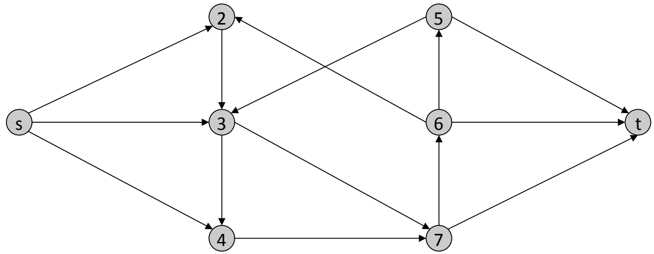
Disjoint path problem

Given a directed graph $G = (V, E)$ and two nodes s and t, find the max number of edge-disjoint s-t paths.

Definition

Two paths are edge-disjoint if they have no edge in common.

Example: communication networks



Edge Disjoint Paths

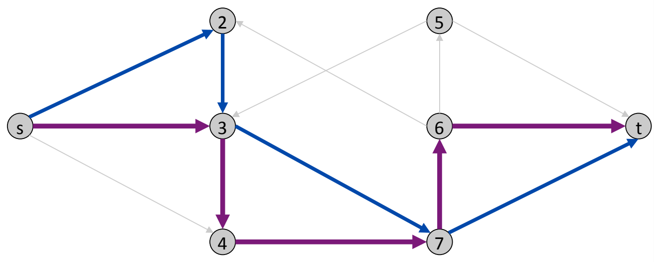
Disjoint path problem

Given a directed graph $G = (V, E)$ and two nodes s and t, find the max number of edge-disjoint s-t paths.

Definition

Two paths are edge-disjoint if they have no edge in common.

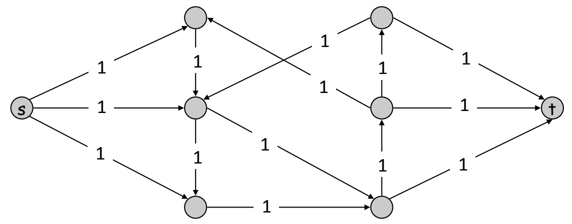
Example: communication networks



Edge Disjoint Paths

Max Flow Formulation

Assign unit capacity to every edge.



Theorem

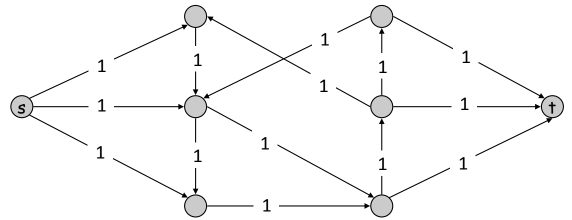
The max number of edge-disjoint s-t paths equals the max flow value.

Proof (Part 1)

- Suppose there are k edge-disjoint paths $P_1, \ldots, P_k$.
- Set $f(e) = 1$ if e participates in some path P_i; else set $f(e) = 0$.
- Since paths are edge-disjoint, f is a flow of value k.

Max Flow Formulation

Assign unit capacity to every edge.



Theorem

The max number of edge-disjoint s-t paths equals the max flow value.

Proof (Part 2)

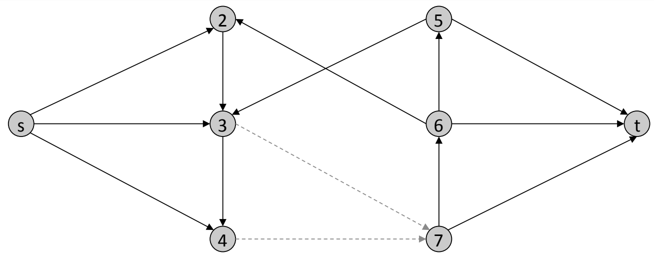
- Suppose the max flow value is k; the integrality theorem tells us that there exists a 0-1 flow f of value k
- Consider edge (s, u) with $f(s, u) = 1$; Bby conservation, $\exists$ an edge (u, v) with $f(u, v) = 1$
- Continue until we reach t, always choosing a new edge; this produces k edge disjoint paths.

Network Connectivity

Given a directed graph $G = (V, E)$ and two nodes s and t, find the minimum number of edges whose removal disconnects t from s.

Definition

A set of edges $F \subseteq E$ disconnects t from s if all s-t paths use at least one edge in F.

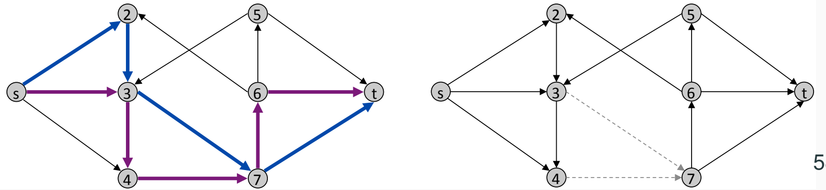


Theorem

The maximum number of edge-disjoint s-t paths is equal to the minimum number of edges whose removal disconnects t from s.

Proof (Part 1)

- Suppose the removal of $F \subseteq E$ disconnects t from s and $|F| = k$.
- All s-t paths use at least one edge in F. Hence, the number of edge-disjoint paths is at most k.



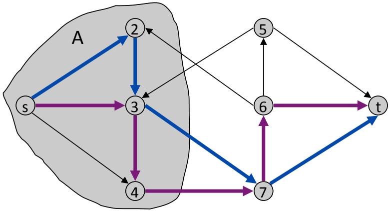
Edge Disjoint Paths and Network Connectivity

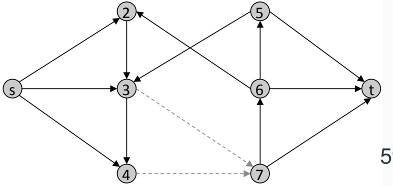
Theorem

The maximum number of edge-disjoint s-t paths is equal to the minimum number of edges whose removal disconnects t from s.

Proof (Part 2)

- Suppose that the max number of edge-disjoint paths is k
- Then the max flow value is k
- The max-flow min-cut theorem gives us that there is a cut (A, B) of capacity k
- Let F be the set of edges going from A to B.
- $|F| = k$ and disconnects t from s.





Questions