Midterm Exam Tomorrow Night: Wed, Feb 20 (8PM-10PM) @ EE 170
Office Hours this Week: Wed @ 10AM (note time change)
Course Recap: (Or, What Could be On the First Midterm?)

Gale-Shapley, Stable Matching Problem

Asymptotic Analysis (e.g., Big O notation)

Recurrence Relationships

Greedy Algorithms

Graph Algorithms

Divide-And-Conquer + Recurrence Relationships

Dynamic Programming

Basic Questions about Network Flow (today)
Midterm 1

Practice Midterm and Solutions Posted on Blackboard

- Solutions posted yesterday (Monday)

- No electronics (laptop, calculator, smart phone etc...)

- May prepare one 3x5 inch index card with any notes you want
  - No additional notes

- Exam is 2 hours (8PM to 10PM)
  - Practice exam is longer than the real midterm
  - Topics are reasonably representative of real midterm
Typo Correction (Master Theorem)

\[ T(n) \leq \begin{cases} 1 & \text{if } n = 1 \\ a \times T\left(\frac{n}{b}\right) + n^c & \text{otherwise} \end{cases} \]

**Case 1:** \( \left(\frac{a}{b^c}\right) < 1 \) \( T(n) = \Theta(n^c) \)

**Case 2:** \( \left(\frac{a}{b^c}\right) = 1 \) \( T(n) = \Theta(n^c \log n) \)

**Case 3:** \( \left(\frac{a}{b^c}\right) > 1 \) \( T(n) = \Theta(n^{\log_b a}) \)
Chapter 7

Network Flow
Soviet Rail Network, 1955

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 ...
Flow network.

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

![Minimum Cut Problem Diagram]
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:

\[
\text{cap}(A, B) = \sum_{e \text{ out of } A} c(e)
\]

Capacity = 10 + 5 + 15 = 30
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: 
\[
\text{cap}(A, B) = \sum_{e \text{ out of } A} c(e)
\]

![Graph with nodes and edges labeled with weights]
**Minimum Cut Problem**

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

![Graph with labeled edges and capacities](image)

- **Edge Capacities:**
  - s to 3: 5
  - 3 to 4: 4
  - 4 to 7: 30
  - 3 to 2: 10
  - 2 to 9: 4
  - 5 to 6: 10
  - 6 to 10: 15
  - 6 to 7: 10

**Capacity Calculation:**

\[
\text{Capacity} = 10 + 8 + 10 = 28
\]
Def. An s-t flow is a function that satisfies:

- For each \( e \in E \):
  \[ 0 \leq f(e) \leq c(e) \]  
  [capacity]

- For each \( v \in V - \{s, t\} \):
  \[ \sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e) \]  
  [conservation]

Def. The value of a flow \( f \) is:

\[ v(f) = \sum_{e \text{ out of } s} f(e) \]
Def. An \textit{s-t flow} is a function that satisfies:

- For each \( e \in E \):
  \[ 0 \leq f(e) \leq c(e) \]  
  \text{[capacity]} 

- For each \( v \in V - \{s, t\} \):
  \[ \sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e) \]  
  \text{[conservation]} 

Def. The value of a flow \( f \) is:
\[ v(f) = \sum_{e \text{ out of } s} f(e) \]
Maximum Flow Problem

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

Value = 28
**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 $s$.

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

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

Pf. \[ v(f) = \sum_{e \text{ out of } s} f(e) + 0 \]

\[ = \sum_{e \text{ out of } s} f(e) + 0 \sum_{v \in A \setminus \{s\}} \left( \sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) \right) \]

by flow conservation, all terms except $v = s$ are 0
Flows and Cuts

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

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

Pf. 

$$v(f) = \sum_{e \text{ out of } s} f(e) + 0$$

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

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

Flow into s is 0
Flows and Cuts

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

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

**Pf.** \( v(f) = \sum_{e \text{ out of } s} f(e) + 0 \)

\[
= \sum_{e \text{ out of } s} f(e) + \sum_{v \in A\{s\}} \left( \sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) \right)
\]

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

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

If \( e=(u,v) \) with \( u \) and \( v \) in \( A \) then \( f(e) \) was added & subtracted in prior sum
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.

\[
\text{Cut capacity} = 30 \implies \text{Flow value} \leq 30
\]
Weak duality. Let $f$ be any flow. Then, for any $s$-$t$ cut $(A, B)$ we have $v(f) \leq \text{cap}(A, B)$.

**Pf.**

\[
\begin{align*}
v(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\
&\leq \sum_{e \text{ out of } A} f(e) \leq \sum_{e \text{ out of } A} c(e) \\
&= \text{cap}(A, B)
\end{align*}
\]
**Certificate of Optimality**

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

Value of flow = 28  
Cut capacity = 28 \implies Flow value \leq 28
Towards a Max Flow Algorithm

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

Flow value = 0
Towards a Max Flow Algorithm

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

Flow value = 20
Towards a Max Flow Algorithm

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

\[ \text{locally optimality} \not= \text{global optimality} \]
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($f$, $c$, $P$) {
  $b \leftarrow$ bottleneck($P$)
  foreach $e \in P$
    if ($e \in E$) $f(e) \leftarrow f(e) + b$
    else $f(e^R) \leftarrow f(e^R) - b$
  return $f$
}

Ford-Fulkerson($G$, $s$, $t$, $c$) {
  foreach $e \in E$ $f(e) \leftarrow 0$
  $G_f \leftarrow$ residual graph

  while (there exists augmenting path $P$) {
    $f \leftarrow$ 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 cut.

**Pf.** We prove both simultaneously by showing TFAE:

(i) There exists a cut $(A, B)$ such that $v(f) = \text{cap}(A, B)$.
(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 \( f \), \( t \notin A \).

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

Original network:

- Must be 0 since there is no edge from \( A \) to \( B \) in residual graph.
- Must be \( c(e) \) since there is no edge from \( A \) to \( B \) in residual graph.
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 $f$, $t \notin A$.

Must be 0 since there is no Edge from $A$ to $B$ in residual graph

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

$$= \sum_{e \text{ out of } A} c(e)$$

$$= \text{cap}(A, B)$$
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. □
7.3 Choosing Good Augmenting Paths
Ford-Fulkerson: Exponential Number of Augmentations

Q. Is generic Ford-Fulkerson algorithm polynomial in input size?

A. No. If max capacity is \( C \), then algorithm can take \( C \) iterations.

\[ m, n, \text{ and } \log C \]
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, algorithm not guaranteed to terminate!

Goal: choose augmenting paths so that:
- Can find augmenting paths efficiently.
- Few iterations.

Choose augmenting paths with: [Edmonds-Karp 1972, Dinitz 1970]
- Max bottleneck capacity.
- Sufficiently large bottleneck capacity.
- Fewest number of edges.
Intuition. Choosing path with highest bottleneck capacity increases flow by max possible amount.
- Don’t worry about finding exact highest bottleneck path.
- Maintain scaling parameter $\Delta$.
- Let $G_f(\Delta)$ be the subgraph of the residual graph consisting of only arcs with capacity at least $\Delta$. 

![Graph $G_f$](image1)

![Graph $G_f(100)$](image2)
Scaling-Max-Flow(G, s, t, c) {
    foreach e ∈ E  f(e) ← 0
    Δ ← smallest power of 2 greater than or equal to C
    G_f ← residual graph

    while (Δ ≥ 1) {
        G_f(Δ) ← Δ-residual graph
        while (there exists augmenting path P in G_f(Δ)) {
            f ← augment(f, c, P)
            update G_f(Δ)
        }
        Δ ← Δ / 2
    }
    return f
}
Capacity Scaling: Correctness

Assumption. All edge capacities are integers between 1 and $C$.

Integrality invariant. All flow and residual capacity values are integral.

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

Pf.

- By integrality invariant, when $\Delta = 1$ $\Rightarrow G_f(\Delta) = G_f$.
- Upon termination of $\Delta = 1$ phase, there are no augmenting paths. •
Capacity Scaling: Running Time

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

**Pf.** Initially $C \leq \Delta < 2C$. $\Delta$ decreases by a factor of 2 each iteration. □

**Lemma 2.** Let $f$ be the flow at the end of a $\Delta$-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.
- $L2 \Rightarrow v(f^*) \leq v(f) + m (2\Delta)$.
- Each augmentation in a $\Delta$-phase increases $v(f)$ by at least $\Delta$. □

**Theorem.** The scaling max-flow algorithm finds a max flow in $O(m \log C)$ augmentations. It can be implemented to run in $O(m^2 \log C)$ time. □
Lemma 2. Let f be the flow at the end of a $\Delta$-scaling phase. Then value of the maximum flow is at most $v(f) + m \Delta$.

Pf. (almost identical to proof of max-flow min-cut theorem)

- We show that at the end of a $\Delta$-phase, there exists a cut $(A, B)$ such that $\text{cap}(A, B) \leq v(f) + m \Delta$.
- 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 $f$, $t \notin A$.

\[
v(f) = \sum_{\text{e out of } A} f(e) - \sum_{\text{e in to } A} f(e) \geq \sum_{\text{e out of } A} (c(e) - \Delta) - \sum_{\text{e in to } A} \Delta = \sum_{\text{e out of } A} c(e) - \sum_{\text{e out of } A} \Delta - \sum_{\text{e in to } A} \Delta \geq \text{cap}(A, B) - m\Delta
\]