Travel: I will be attending a conference next week.
Tuesday: Recorded Lecture + return midterms (hopefully)
Thursday: No class
Midterm Regrade? Must be completed within 2 weeks (syllabus). Please e-mail us before then.
Midterm Solutions: Will post on blackboard before Tuesday.
Max Flow Recap

Max-Flow Problem, Min Cut Problem
- Definition of a s-t flow \( f(e) \) and a s-t cut \( (A,B) \)
- Value of a flow \( f \)
- Capacity of a s-t cut \( (A,B) \)

Weak Duality Lemma: For any flow \( f \) and s-t cut \( A,B \) we have \( v(f) \leq cap(A,B) \) (i.e., capacity of minimum cut is upper bound on max-flow)

Finding a Max-Flow:
- Greedy algorithm fails!
- Residual Graph
- Ford-Fulkerson Algorithm
  - Repeatedly find augmenting path in residual graph
  - Proof of Correctness
  - Max-Flow Min-Cut Equivalence
Ford-Fulkerson Algorithm

$G$: 

\[
\begin{align*}
&\text{s} & 10 & \text{2} & 10 & \text{10} & \text{3} & 8 & 9 & \text{5} & 6 & \text{10} & \text{5} & 10 & \text{t} \\
&10 & 2 & \text{4} & 4 & \text{4} & 6 & 10 & \text{5} & 10 & \text{10} & \text{10} & \text{10} & \text{10} & \text{t}
\end{align*}
\]
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 $v(f) = cap(A, B)$.
(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) $\implies$ (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 $f$, $t \notin A$.

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

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

$$= \cap(A, B)$$

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. ▪
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 $\Delta$. 

![Diagram of network flow](image)

$G_f$  

$G_f(100)$
Capacity Scaling

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.
Proof. 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 $v(f) + m \Delta$.

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 $f$, $t \notin A$.

\[
v(f) = \sum_{e \in A} f(e) - \sum_{e \notin A} f(e) \\
\geq \sum_{e \in A} (c(e) - \Delta) - \sum_{e \in A} \Delta \\
= \sum_{e \in A} c(e) - \sum_{e \in A} \Delta - \sum_{e \in A} \Delta \\
\geq \text{cap}(A, B) - m\Delta
\]
Dinic’s Max Flow Min-Cut Algorithm

Use Breadth First Search to Compute Level Graph

$G$: Level 0

$G_L$: Discard cross-layer edges
Dinic’s Max Flow Min-Cut Algorithm

Use Breadth First Search to Compute Level Graph

$G$: Level 0

$G_L$: Discard cross-layer edges
Dinic’s Max Flow Min-Cut Algorithm

Use Breadth First Search to Compute Level Graph

$G$: Level 0

$G_L$: Level 1

Discard cross-layer edges
Find Blocking Flow
Dinic’s Max Flow Min-Cut Algorithm

Create Residual Graph $G_f$

$G_f$:

$G_L$:

Total Flow: 15
Dinic’s Max Flow Min-Cut Algorithm

Run BFS on $G_f$ to create level graph $G_{f,L}$

$G_f$:

Remark: Number of levels increased. This is not a coincidence!
Dinic’s Max Flow Min-Cut Algorithm

Run BFS on $G_f$ to create level graph $G_{f,L}$

$G_f$: 

- **Level 0**: Source node $s$ connected to nodes $3$ and $4$
- **Level 1**: Node $2$ connected to nodes $3$ and $4$
- **Level 2**: Node $5$ connected to node $t$
- **Level 3**: Node $4$ connected to node $t$
- **Level 4**: Node $t$
Dinic’s Max Flow Min-Cut Algorithm

Run BFS on $G_f$ to create level graph $G_{f,L}$

$G_f$: Level 0

$G_{f,L}$ Level 0

Level 1

Level 2

Level 3

Level 4
Dinic’s Max Flow Min-Cut Algorithm

Run BFS on $G_f$ to create level graph $G_{f,L}$

$G_f$:

$G_{f,L}$:

Blocking Flow for level graph $G_{f,L}$

Total Extra Flow: 5
Dinic’s Max Flow Min-Cut Algorithm

New Residual Graph $G_f$

$G_f$:  

Blocking Flow for level graph $G_{f,L}$  

Total Extra Flow: 5
Dinic’s Max Flow Min-Cut Algorithm

New Residual Graph \( G_f \)

\[ G_f: \]

\[ s \] 10 2 2 4 \[ 2 \] 4 \[ 4 \] 10
\[ 3 \] 9 2 6 \[ 5 \] 10
\[ 5 \] 9
\[ 9 \] 6
\[ 10 \] 10

Reachable

Breadth First Search: Yields minimum \( s-t \) cut! \( \Rightarrow \) We are done!

\[ G: \]

\[ s \] 10 2 8 1 \[ 2 \] 10
\[ 3 \] 9
\[ 10 \]

capacity

A
Finding a Blocking Flow in $G_{f,L}$

**Definition:** We let $C_{f,L}(e)$ denote the capacity of an edge $e$ in $G_{f,L}$

**Definition:** Given an augmenting flow $f'$ for $G_{f,L}$ and a s-t path $P$ we define $B(P) = \min_{e \in P} C_{f,L}(e)$

**FindBlockingFlow($G_{f,L}$)**

- Initialize $\text{RemCap}(e) = C_{f,L}(e)$
- While there exists a path $P$ with $B(P) > 0$
  - Set $f'(e) = f'(e) + B(P)$ for each edge $e \in P$
  - Set $\text{RemCap}(e) = \text{RemCap}(e) - B(P)$ for each edge $e \in P$

**Analysis:** Each iteration of while loop “eliminates” at least one edge.

**Implication:** Terminates after at most $m$ rounds.

**Naïve Running Time:** $O((m+n)m)$

**Amortization:** Can enumerate paths in amortized time $O(n)$ per path

**Even Better:** Find blocking flow in time $O(m \log n)$ with dynamic trees.
Dinic's Algorithm

1. Start with empty flow $f$
2. Construct $G_f$
3. Repeat until $s$ and $t$ are disconnected (no augmenting path)
   1. (Level Graph) Run BFS on $G_f$ to build $G_{f,L}$
   2. (Blocking Flow) Find blocking flow $f'$ in $G_{f,L}$
   3. (Augment) Let $f=f+f'$ and Construct $G_f$
4. Output $f$

Analysis:
Claim: Each time we iterate the loop we increase the depth of $G_f$

Implication: Must terminate in at most $n$ iterations!

Time Per Iteration: $O(nm)$ to find blocking flow $f'$

Total Time: $O(n^2m)$
Dinic’s Algorithm: Correctness and Running Time

Correctness follows directly from Augmenting Path Theorem.

Augmenting path theorem. Flow $f$ is a max flow iff there are no augmenting paths.

Running Time Analysis: Let $f_i$ denote residual graph after iteration $i$ ($G_{f_0} = G$)

Definition: $\text{depth}(G_{f_i}) = \text{length of the shortest directed path from s to t}$.

Key Claim: $\text{depth}(G_{f_{i+1}}) > \text{depth}(G_{f_i})$ (depth always increases)
Dinic’s Algorithm: Correctness and Running Time

**Running Time Analysis:** Let $f_i$ denote residual graph after iteration $i$ ($G_{f_0} = G$)

**Definition:** $\text{depth}(G_{f_i}) = \text{length of the shortest directed path from } s \text{ to } t$.

**Key Claim:** $\text{depth}(G_{f_{i+1}}) > \text{depth}(G_{f_i})$ (depth always increases)

**Proof:** Suppose (for contradiction) that $\text{depth}(G_{f_{i+1}}) \leq \text{depth}(G_{f_i})$.

- Then $G_{f_{i+1}}$ contains an $s$-$t$ path of length $\leq \text{depth}(G_{f_i})$.
- This path corresponds to an augmenting path for the flow $f' = f_{i+1} - f_i$ in $G_{f_i}$.
- But since the augmenting path has length $\text{depth}(G_{f_i})$ it is also an augmenting path in the level graph $G_{f_i,L}$.
- This contradicts the claim that $f'$ is a blocking flow in $G_{f_i,L}$!
Dinic’s Algorithm: Correctness and Running Time

**Running Time Analysis:** Let $f_i$ denote residual graph after iteration $i$ ($G_{f_0} = G$)

**Definition:** $\text{depth}(G_{f_i}) = \text{length of the shortest directed path from } s \text{ to } t$.

**Key Claim:** $\text{depth}(G_{f_{i+1}}) > \text{depth}(G_{f_i})$ (depth always increases)

**Implication:** \#iterations is at most $n$

Time to Compute Blocking Flow in Level Graph: $O(mn)$
- Using special data-structure called dynamic trees $O(m \log n)$

Total Time: $O(mn \log n)$ with dynamic trees or $O(mn^2)$ without.
7.7 Extensions to Max Flow
**Circulation with Demands**

- Directed graph $G = (V, E)$.
- Edge capacities $c(e), e \in E$.
- Node supply and demands $d(v), v \in V$.

**Def.** A circulation is a function that satisfies:

- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V$: $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

**Circulation problem:** given $(V, E, c, d)$, does there exist a circulation?
Circulation with Demands

Necessary condition: sum of supplies = sum of demands.

\[ \sum_{v : d(v) > 0} d(v) = \sum_{v : d(v) < 0} -d(v) =: D \]

Pf. Sum conservation constraints for every demand node \( v \).
Circulation with Demands

Max flow formulation.

\( G: \)

\begin{tikzpicture}[main_node/.style={circle,draw,fill=gray,minimum size=1cm}, edge_label/.style={fill=white,font={\small}}]
\node[main_node] (source) at (0,0) {3}
\node[main_node] (s1) at (1,1) {10}
\node[main_node] (s2) at (2,2) {-8}
\node[main_node] (s3) at (3,3) {7}
\node[main_node] (sink) at (4,4) {11}
\node[main_node] (d1) at (5,3) {9}
\node[main_node] (d2) at (6,2) {4}
\node[main_node] (d3) at (7,1) {6}
\node[main_node] (d4) at (8,0) {10}
\node[main_node] (d5) at (9,1) {-7}

\draw[->, thick] (source) edge node[edge_label] {3} (s1);
\draw[->, thick] (s1) edge node[edge_label] {10} (s2);
\draw[->, thick] (s2) edge node[edge_label] {7} (s3);
\draw[->, thick] (s3) edge node[edge_label] {7} (sink);
\draw[->, thick] (sink) edge node[edge_label] {9} (d1);
\draw[->, thick] (d1) edge node[edge_label] {4} (d2);
\draw[->, thick] (d2) edge node[edge_label] {4} (d3);
\draw[->, thick] (d3) edge node[edge_label] {6} (d4);
\draw[->, thick] (d4) edge node[edge_label] {10} (d5);
\draw[->, thick] (d5) edge node[edge_label] {-7} (source);
\end{tikzpicture}
Circulation with Demands

Max flow formulation.
- Add new source $s$ and sink $t$.
- For each $v$ with $d(v) < 0$, add edge $(s, v)$ with capacity $-d(v)$.
- For each $v$ with $d(v) > 0$, add edge $(v, t)$ with capacity $d(v)$.
- Claim: $G$ has circulation iff $G'$ has max flow of value $D$.
**Circulation with Demands**

**Integrality theorem.** If all capacities and demands are integers, and there exists a circulation, then there exists one that is integer-valued.

**Pf.** Follows from max flow formulation and integrality theorem for max flow.

**Characterization.** Given \((V, E, c, d)\), there does not exists a circulation if and only if there exists a node partition \((A, B)\) such that

\[
\sum_{v \in B} d_v > \text{cap}(A, B)
\]

**Pf idea.** Look at min cut in \(G'\).
Circulation with Demands and Lower Bounds

Feasible circulation.
- Directed graph $G = (V, E)$.
- Edge capacities $c(e)$ and lower bounds $\ell(e)$, $e \in E$.
- Node supply and demands $d(v)$, $v \in V$.

Def. A circulation is a function that satisfies:
- For each $e \in E$: $\ell(e) \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V$: $\sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v)$ (conservation)

Circulation problem with lower bounds. Given $(V, E, \ell, c, d)$, does there exist a circulation?
Circulation with Demands and Lower Bounds

Idea. Model lower bounds with demands.
- Send $\ell(e)$ units of flow along edge $e$.
- Update demands of both endpoints.

Theorem. There exists a circulation in $G$ iff there exists a circulation in $G'$. If all demands, capacities, and lower bounds in $G$ are integers, then there is a circulation in $G$ that is integer-valued.

Pf sketch. $f(e)$ is a circulation in $G$ iff $f'(e) = f(e) - \ell(e)$ is a circulation in $G'$.  

![Diagram](image-url)
7.8 Survey Design
Survey Design

Survey design.

- Design survey asking $n_1$ consumers about $n_2$ products.
- Can only survey consumer $i$ about product $j$ if they own it.
- Ask consumer $i$ between $c_i$ and $c_i'$ questions.
- Ask between $p_j$ and $p_j'$ consumers about product $j$.

Goal. Design a survey that meets these specs, if possible.

Bipartite perfect matching. Special case when $c_i = c_i' = p_i = p_i' = 1$. 

one survey question per product
Algorithm. Formulate as a circulation problem with lower bounds.

- Include an edge \((i, j)\) if consumer \(j\) owns product \(i\).
- Integer circulation \(\Leftrightarrow\) feasible survey design.