Announcements

• Midterm is graded! Please pick up after class
• Midterm average: 45.6/70
• HW3 is up! Due in class Wed May 18
• HW3 is a longer homework (with 4 questions)
Last Class: Algorithms for Max-Flow

\( n = \#\text{vertices}, m = \#\text{edges in } G, C_{\text{max}} = \text{max capacity}, F_{\text{max}} = \text{max flow} \)

Integer edge capacities

- **Ford-Fulkerson**: Running Time = \(O(m F_{\text{max}})\)

- **Other efficient Ford-Fulkerson Style Algorithms**:
  - **Edmonds-Karp**: Running Time = \(O(nm^2)\)
  - **Capacity Scaling**: Running Time = \(O(m^2 \log C_{\text{max}})\)

- **Preflow-Push**: Running Time = \(O(mn^2)\)
Last Class: Bipartite Matching

Given a bipartite graph \( G = (L, R) \), find a matching in \( G \) of maximum cardinality.

Recall: A matching \( M \) is a set of edges \((u, v)\) such that no two edges share a common vertex.

Reduction to Max-Flow:

**Graph G:**

- Arthur
- Bill
- Charles
- David

**Graph H:**

- Source \( s \): connected to all nodes in \( L \)
- Sink \( t \): connected to all nodes in \( R \)

Perfect matching: a matching of size \( n \), where \( n = \#\text{vertices in } L = \#\text{vertices in } R \)

**Property:** Size of max flow in \( H \) = cardinality of maximum matching in \( G \)
Bipartite Matching

• Reduction to Max Flow

• A Faster Algorithm for Bipartite Matching
Bipartite Matching

Graph G:

Arthur
Bill
Charles
David
Angela
Beth
Connie
Doris

L
R

Graph H:

Properties of flow graph H:

- Each edge has capacity 1
- Each node (except s and t) has either indegree 1 or outdegree 1

Such graphs are called unit graphs
Now: a faster algorithm for finding max flows in unit graphs
Blocking Flows

A Blocking Flow of G saturates at least one edge in all s-t paths in G

Example:

A Max Flow is always a Blocking Flow (Why?)
A Blocking Flow may not be a max flow
**Blocking Flows Algorithm (Dinic 70)**

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
   - Let \( H_f \) = subgraph of \( G_f \) containing only admissible edges
   - Find a blocking flow in \( H_f \), and add it to \( f \)

**Iteration 1**

\( H_f \) (before)  \[ \begin{array}{ccc}
 s & \rightarrow & a \\
 b & \rightarrow & t
\end{array} \]

\( f \)  \[ \begin{array}{ccc}
 s & \rightarrow & a & 10^6 \\
 a & \rightarrow & t & 10^6 \\
 s & \rightarrow & b & 10^6 \\
 b & \rightarrow & t & 10^6
\end{array} \]

\( G_f \) (after)  \[ \begin{array}{ccc}
 s & \rightarrow & a & 10^6 \\
 a & \rightarrow & t & 10^6 \\
 s & \rightarrow & b & 10^6 \\
 b & \rightarrow & t & 10^6
\end{array} \]

A Blocking Flow in \( G \) **saturates at least one edge** in all s-t paths in \( G \)
An edge e in a residual graph \( G_f \) is **admissible** if it lies on a s-t shortest path

Why is this algorithm correct?
**Blocking Flows Algorithm (Dinic 70)**

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
   - Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
   - Find a blocking flow in $H_f$, and add it to $f$

A blocking flow **saturates** $\geq 1$ edge in all s-t paths in $G$

An edge in a residual graph $G_f$ is **admissible** if it lies on some s-t shortest path.

Running Time of Blocking Flow Algorithm in unit graphs $= O(m\sqrt{n})$

**Proof Outline:**
1. There can be at most $O(\sqrt{n})$ iterations of the blocking flow algorithm in an unit graph
2. Each iteration can be implemented in $O(m)$ time in an unit graph

**Recall, in an unit graph:**
- Each edge has capacity 1
- Each node (except s and t) has either indegree 1 or outdegree 1
Blocking Flows: Unit Graphs

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
  Let $H_f$ = subgraph of $G_f$ containing only admissible edges
  Find a blocking flow in $H_f$, and add it to $f$

A blocking flow saturates $\geq 1$ edge in all s-t paths in $G$

An edge in a residual graph $G_f$ is admissible if it lies on some s-t shortest path.

**Property:** Augmenting along a blocking flow strictly increases the s-t distance in $G_f$
Before we prove this property, let us look at some properties of shortest paths and $H_f$
Blocking Flows: Unit Graphs

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
   Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
   Find a blocking flow in $H_f$, and add it to $f$

- A blocking flow **saturates** $\geq 1$ edge in all $s$-$t$ paths in $G$
- An edge in a residual graph $G_f$ is **admissible** if it lies on some $s$-$t$ shortest path.

**Properties of Shortest Paths:**

- Let $d(u) = \text{shortest path distance of } u \text{ from } s$
- Layer($i$) = all nodes $u$ such that $d(u)=i$
- For any edge $(u,v)$, $d(v) \leq d(u) + 1$
- Edges($u,v$): Cannot jump downwards across $\geq 2$ layers

**Red Edge: Invalid:** Cannot be in the graph
**Blocking Flows: Unit Graphs**

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
  Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
  Find a blocking flow in $H_f$, and add it to $f$

**Property:** Augmenting along a blocking flow strictly increases the s-t distance in $G_f$

$d_i(u) = \text{SP-distance}(s,u) \text{ at round } i$
A downwards edge can’t jump across >1 layers

A blocking flow **saturates** $\geq 1$ edge in **all** s-t paths in $G$

An edge in a residual graph $G_f$ is **admissible** if it lies on some s-t shortest path.
**Blocking Flows: Unit Graphs**

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
  Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
  Find a blocking flow in $H_f$, and add it to $f$

**Property:** Augmenting along a blocking flow strictly increases the s-t distance in $G_f$

**Proof:** Consider any s-t shortest path in $G_f$ at round $i+1$

A blocking flow saturates $\geq 1$ edge in all s-t paths in $G$

An edge in a residual graph $G_f$ is admissible if it lies on some s-t shortest path.

A downwards edge can't jump across $>1$ layers

$d_i(u) = \text{SP-distance}(s,u)$ at round $i$

Layer 0
Layer 1
Layer 2
Layer 3
Layer 4

Proof:

```
\text{i+1} 0 1 2 3 4 5 6
s a b u v w t
```

- **d_i(u)** = SP-distance(s,u) at round i
- A downwards edge can't jump across >1 layers
**Blocking Flow Algorithm:**

Start with zero flow
Repeat:
  Let $H_f$ = subgraph of $G_f$ containing only admissible edges
  Find a blocking flow in $H_f$, and add it to $f$

**Property:** Augmenting along a blocking flow strictly increases the s-t distance in $G_f$

**Proof:** Consider any s-t shortest path in $G_f$ at round $i+1$

A blocking flow **saturates** $\geq 1$ edge in all s-t paths in $G$

An edge in a residual graph $G_f$ is **admissible** if it lies on some s-t shortest path.

$d_i(u) = $ SP-distance$(s,u)$ at round $i$

A downwards edge can’t jump across $>1$ layers
**Property:** Augmenting along a blocking flow strictly increases the s-t distance in $G_f$

**Proof:** Consider any s-t shortest path in $G_f$ at round $i+1$

Three kinds of edges $e = (u, v)$ in the path:

1. $e$ in $G_f$ and $H_f$ in round $i$: $d_{i+1}(v) = d_i(u) + 1$

$d_i(u) = \text{SP-distance}(s, u)$ at round $i$
A downwards edge can’t jump across $>1$ layers
**Blocking Flows: Unit Graphs**

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
  - Let $H_f = \text{subgraph of } G_f$ containing only admissible edges
  - Find a blocking flow in $H_f$, and add it to $f$

**Property:** Augmenting along a blocking flow strictly increases the s-t distance in $G_f$

**Proof:** Consider any s-t shortest path in $G_f$ at round $i+1$

- **Three kinds of edges** $e = (u, v)$ in the path:
  1. $e$ in $G_f$ and $H_f$ in round $i$: $d_{i+1}(v) = d_i(u) + 1$
  2. $e$ in $G_f$ but not $H_f$ in round $i$: $d_{i+1}(v) < d_i(u) + 1$

$\text{Layer 0}$
$\text{Layer 1}$
$\text{Layer 2}$
$\text{Layer 3}$
$\text{Layer 4}$

$\text{A blocking flow saturates} \geq 1 \text{ edge in all s-t paths in } G$

$\text{An edge in a residual graph } G_f \text{ is admissible} \text{ if it lies on some s-t shortest path.}$

$d_i(u) = \text{SP-distance(s,u) at round } i$

A downwards edge can’t jump across $> 1$ layers
Blocking Flows: Unit Graphs

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
- Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
- Find a blocking flow in $H_f$, and add it to $f$

**Property:** Augmenting along a blocking flow strictly increases the s-t distance in $G_f$

**Proof:** Consider any s-t shortest path in $G_f$ at round $i+1$

```
       s
      /   \
 a----b----u----v----w----t
|       |       |       |
0 1  2  3  4  5  6
```

**Three** kinds of edges $e = (u, v)$ in the path:
1. $e$ in $G_f$ and $H_f$ in round $i$: $d_i(v) = d_i(u) + 1$
2. $e$ in $G_f$ but not $H_f$ in round $i$: $d_i(v) < d_i(u) + 1$
3. $e$ not in $G_f$ in round $i$: $(v, u)$ in $H_f$ in round $i$: $d_i(u) = d_i(v) + 1$

A downwards edge can’t jump across $>1$ layers

A blocking flow **saturates** $>1$ edge in all s-t paths in $G$

An edge in a residual graph $G_f$ is **admissible** if it lies on some s-t shortest path.
Blocking Flows: Unit Graphs

**Property:** Augmenting along a blocking flow strictly increases the s-t distance in $G_f$

**Proof:** Consider any s-t shortest path in $G_f$ at round $i+1$

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>a</th>
<th>b</th>
<th>u</th>
<th>v</th>
<th>w</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_i$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>$d_{i+1}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

**Three** kinds of edges $e = (u, v)$ in the path:

1. $e$ in $G_f$ and $H_f$ in round $i$: $d_i(v) = d_i(u) + 1$
2. $e$ in $G_f$ but not $H_f$ in round $i$: $d_i(v) < d_i(u) + 1$
3. $e$ not in $G_f$ in round $i$: $(v, u)$ in $H_f$ in round $i$: $d_i(u) = d_i(v) + 1$

A downwards edge can't jump across $>1$ layers.

**Blocking Flow Algorithm:**

Start with zero flow

Repeat:

- Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
- Find a blocking flow in $H_f$, and add it to $f$

A blocking flow saturates $>=1$ edge in all s-t paths in $G$

An edge in a residual graph $G_f$ is admissible if it lies on some s-t shortest path.
**Blocking Flows: Unit Graphs**

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
  Let \( H_f = \) subgraph of \( G_f \) containing only admissible edges
  Find a blocking flow in \( H_f \), and add it to \( f \)

**Property:** Augmenting along a blocking flow strictly increases the s-t distance in \( G_f \)

**Proof:** Consider any s-t shortest path in \( G_f \) at round \( i+1 \)

Three kinds of edges \( e = (u, v) \) in the path:
1. \( e \) in \( G_f \) and \( H_f \) in round \( i \): \( d_i(v) = d_i(u) + 1 \)
2. \( e \) in \( G_f \) but not \( H_f \) in round \( i \): \( d_i(v) < d_i(u) + 1 \)
3. \( e \) not in \( G_f \) in round \( i \): \( (v, u) \) in \( H_f \) in round \( i \): \( d_i(u) = d_i(v) + 1 \)

Thus, \( d_i(t) \leq d_{i+1}(t) \). Equal only if all blue edges. But then, the path is in \( H_f \) in round \( i \), which can't be due to blocking flow!
Blocking Flows: Unit Graphs

**Property 1:** Augmenting along a blocking flow $f$ increases the $s$-$t$ distance in the residual graph $G_f$.

**Property 2:** After $d$ iterations of blocking flow, $\text{size}(\text{max flow}) \leq \text{size}(\text{current flow}) + \frac{n}{d}$.

**Blocking Flow Algorithm:**

Start with zero flow

Repeat:

1. Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
2. Find a blocking flow in $H_f$, and add it to $f$
**Blocking Flows: Unit Graphs**

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
  - Let $H_f =$ subgraph of $G_f$ containing only admissible edges
  - Find a blocking flow in $H_f$, and add it to $f$

**Property 1:** Augmenting along a blocking flow $f$ increases the $s$-$t$ distance in the residual graph

**Property 2:** After $d$ iterations of blocking flow, size(max flow) $\leq$ size(current flow) + $n/d$

**Proof:** After $d$ iterations of blocking flow, any $s$-$t$ path in residual graph is at least $d$ edges long
Since $G_f$ is a unit graph, each such $s$-$t$ path has to involve a disjoint set of vertices
Thus there can be at most $n/d$ more such augmenting paths
Blocking Flows: Unit Graphs

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
   Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
   Find a blocking flow in $H_f$, and add it to $f$

- A blocking flow **saturates** $\geq 1$ edge in all s-t paths in $G$
- An edge in a residual graph $G_f$ is **admissible** if it lies on some s-t shortest path.

**Property 1:** Augmenting along a blocking flow $f$ increases the s-t distance in the residual graph

**Property 2:** After $d$ iterations of blocking flow, $\text{size}(\text{max flow}) \leq \text{size}(\text{current flow}) + \frac{n}{d}$

**Property 3:** The total number of iterations of blocking flow is at most $O(\sqrt{n})$

**Proof:** In property 2, set $d = \sqrt{n}$ and use the fact that $\text{size}(\text{max flow}) \leq n$
Blocking Flows Algorithm (Dinic 70)

Blocking Flow Algorithm:
Start with zero flow
Repeat:
   Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
   Find a blocking flow in $H_f$, and add it to $f$

A blocking flow saturates $\geq 1$ edge in all s-t paths in $G$

An edge in a residual graph $G_f$ is admissible if it lies on some s-t shortest path.

Running Time of Blocking Flow Algorithm in unit graphs $= O(m\sqrt{n})$

Proof Outline:
1. There can be at most $O(\sqrt{n})$ iterations of the blocking flow algorithm in an unit graph
2. Each iteration can be implemented in $O(m)$ time in an unit graph

Recall, in an unit graph:
- Each edge has capacity 1
- Each node (except s and t) has either indegree 1 or outdegree 1
Blocking Flows in Unit Graphs: Implementing a Single Iteration

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
  Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
  Find a blocking flow in $H_f$, and add it to $f$

A blocking flow saturates $\geq 1$ edge in all s-t paths in $G$

An edge in a residual graph $G_f$ is admissible if it lies on some s-t shortest path.

To find a blocking flow, successively find s-t paths in $H_f$ and delete them.
To find an s-t path, start from s and use **two operations** until we hit t:
Blocking Flows in Unit Graphs: Implementing a Single Iteration

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
  Let \( H_f = \) subgraph of \( G_f \) containing only admissible edges
  Find a blocking flow in \( H_f \), and add it to \( f \)

To find a blocking flow, successively find s-t paths in \( H_f \) and delete them.
To find an s-t path, start from s and use **two operations** until we hit t:

<table>
<thead>
<tr>
<th>Current</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="s-v-u-s" /></td>
<td><img src="image" alt="s-v-u-s" /></td>
</tr>
</tbody>
</table>

**Advance:** Current vertex = v. If v is not t, and there is an outgoing edge (v, u), add (v, u) to current path, and set current vertex = u.
Blocking Flows in Unit Graphs: Implementing a Single Iteration

**Blocking Flow Algorithm:**
- Start with zero flow
- Repeat:
  - Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
  - Find a blocking flow in $H_f$, and add it to $f$

To find a blocking flow, successively find s-t paths in $H_f$ and delete them.

To find an s-t path, start from s and use **two operations** until we hit t:

- **Advance:** Current vertex = v. If v is not t, and there is an outgoing edge (v, u), add (v, u) to current path, and set current vertex = u.

- **Retreat:** Current vertex = v. If v is not t, and no outgoing edge (v, u), delete previous edge on current path, and retreat to previous vertex.
Blocking Flows in Unit Graphs: Implementing a Single Iteration

A blocking flow saturates \( \geq 1 \) edge in all s-t paths in \( G \).

An edge in a residual graph \( G_f \) is admissible if it lies on some s-t shortest path.

To find a blocking flow, successively find s-t paths in \( H_f \) and delete them.

To find an s-t path, start from s and use two operations until we hit t:

**Advance:** Current vertex = \( v \). If \( v \) is not t, and there is an outgoing edge \((v, u)\), add \((v, u)\) to current path, and set current vertex = \( u \).

**Retreat:** Current vertex = \( v \). If \( v \) is not t, and no outgoing edge \((v, u)\), delete previous edge on current path, and retreat to previous vertex.

**Blocking Flow Algorithm:**
- Start with zero flow
- Repeat:
  - Let \( H_f = \) subgraph of \( G_f \) containing only admissible edges
  - Find a blocking flow in \( H_f \), and add it to \( f \)

Total time to find a blocking flow = \( O(m) \) (each edge is examined at most \( O(1) \) times)
Blocking Flows Algorithm (Dinic 70)

**Blocking Flow Algorithm:**
Start with zero flow
Repeat:
  Let $H_f = \text{subgraph of } G_f \text{ containing only admissible edges}$
  Find a blocking flow in $H_f$, and add it to $f$

A blocking flow **saturates** $\geq 1$ edge in all $s$-$t$ paths in $G$

An edge in a residual graph $G_f$ is **admissible** if it lies on some $s$-$t$ shortest path.

Running Time of Blocking Flow Algorithm in unit graphs = $O(m\sqrt{n})$

**Proof Outline:**
1. There can be at most $O(\sqrt{n})$ iterations of the blocking flow algorithm in an unit graph
2. Each iteration can be implemented in $O(m)$ time in an unit graph

**Recall, in an unit graph:**
- Each edge has capacity 1
- Each node (except $s$ and $t$) has either indegree 1 or outdegree 1
Bipartite Matching

Properties of flow graph H:

- Each edge has capacity 1
- Each node (except s and t) has either indegree 1 or outdegree 1

Such graphs are called **unit graphs**

Running time of blocking flow algorithm on unit graphs = \( O(m\sqrt{n}) \)

Thus, time to compute bipartite matching = \( O(m\sqrt{n}) \)
Bipartite Matching

• Reduction to Max Flow

• A Faster Algorithm for Bipartite Matching