Comparison

<table>
<thead>
<tr>
<th></th>
<th>Greedy</th>
<th>Divide and Conquer</th>
<th>Dynamic Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formulate problem</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Design algorithm</td>
<td>easy</td>
<td>hard</td>
<td>hard</td>
</tr>
<tr>
<td>Prove correctness</td>
<td>hard</td>
<td>easy</td>
<td>easy</td>
</tr>
<tr>
<td>Analyze running time</td>
<td>easy</td>
<td>hard</td>
<td>easy</td>
</tr>
</tbody>
</table>

Where does network flow fit in?

Network Flow

- Previous topics: design techniques
- Network flow: specific class of problems with many applications
- Direct applications: commodities in networks
  - wheat/rail networks
  - packets/internet
  - water/pipes
- Indirect applications:
  - Matching in bipartite graphs
  - Airline scheduling
  - Baseball elimination
Network Flow

- Previous topics: design techniques
- Network flow: specific class of problems with many applications
- Direct applications: commodities in networks
  - wheat/rail networks
  - packets/internet
  - water/pipes
- Indirect applications:
  - Matching in bipartite graphs
  - Airline scheduling
  - Baseball elimination

Plan: design and analyze algorithms for max-flow problem, then apply to solve other problems

First, a Story About Flow and Cuts

Key theme: flows in a network are intimately related to cuts

Soviet rail network in 1955

Defining Flows

Example/definitions on board

- Motivating example
- Flow network
  - Directed graph
  - Source $s$, target $t$
  - Edge capacities $c(e)$
- Flow
  - Flow $f(e)$ on each edge
  - Respects capacity and flow conservation
  - Value $v(f)$
- Max flow problem: find a flow of maximum value
Designing a Max-Flow Algorithm

Idea (false start): repeatedly choose paths and “augment” flow on those paths until we can no longer do so

Board work
- Explore this idea
- See where it gets stuck
- Introduce and define residual graph $G_f$
  - Forward edges
  - Backward edges
  - Augmenting path
  - Bottleneck edge

Augmenting Path

Revised Idea: use paths in the residual graph to augment flow

Augment flow $f$ on path $P$ in $G_f$

Let $b = \text{bottleneck}(P, f)$  \( \triangleright \) least residual capacity in $P$

for edge $e = (u, v)$ in $P$ do
  if $e$ is a forward edge then
    $f(e) = f(e) + b$  \( \triangleright \) increase flow on forward edges
  else
    $f(e) = f(e) - b$  \( \triangleright \) decrease flow on backward edges
  end if
end for

Example on board

Ford-Fulkerson Algorithm

Repeatedly find augmenting paths in the residual graph and use them to augment flow!

Ford-Fulkerson($G$, $s$, $t$)

\( \triangleright \) Initially, no flow
Initialize $f(e) = 0$ for all edges $e$
Initialize $G_f = G$

Ford-Fulkerson Algorithm

Repeatedly find augmenting paths in the residual graph and use them to augment flow!

Ford-Fulkerson($G$, $s$, $t$)

\( \triangleright \) Initially, no flow
Initialize $f(e) = 0$ for all edges $e$
Initialize $G_f = G$

\( \triangleright \) Augment flow as long as it is possible
while there exists an $s$-$t$ path $P$ in $G_f$ do
  $f = \text{Augment}(f, P)$
  update $G_f$
end while
return $f$
Ford-Fulkerson Examples

Toy example on board

Example 2

Flow value = 0

Flow value = 8

Flow value = 10

Flow value = 16

Flow value = 18
Ford-Fulkerson Analysis

- Step 1: argue that F-F returns a flow
- Step 2: analyze termination and running time
- Step 3: argue that F-F returns a maximum flow

Step 1: F-F returns a flow

Claim: Let $f$ be a flow and let $f' = \text{Augment}(f, P)$. Then $f'$ is a flow.

Proof idea. Verify two conditions for $f'$ to be a flow

- $f'$ satisfies capacity constraints
- $f'$ satisfies flow conservation

Example and proof sketch on board
### Step 2: Termination and Running Time

**Assumption:** all capacities are integers. Then, by nature of F-F, all flow values and residual capacities remain integers during the algorithm.

**Claim:** F-F terminates in at most \( v(f) \) iterations, where \( f \) is the returned flow. **Proof?**

**Running time:**

- Let \( C \) be the total capacity of edges leaving source \( s \)
- Then \( v(f) \leq C \)
- So F-F terminates in at most \( C \) iterations

**Running time per iteration?**

### Step 3: F-F returns a maximum flow

We will prove this by establishing a deep connection between flows and cuts in graphs: the **max-flow min-cut theorem**.

**Board work:** cut definitions

- \( s-t \) cut \( (A, B) \)
- Capacity \( c(A, B) \)
- Examples
Flow Value Lemma

First relationship between cuts and flows

Lemma: Let $f$ be any flow and $(A, B)$ be any $s$-$t$ cut. Then $\nu(f) = \sum_{e \in \text{out of } A} f(e) - \sum_{e \in \text{into } A} f(e)$.

Proof:

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

Corollary: Cuts and Flows

Really important corollary of flow-value lemma

Corollary: Let $f$ be any $s$-$t$ flow and let $(A, B)$ be any $s$-$t$ cut. Then $\nu(f) \leq c(A, B)$.

Proof:

$\nu(f) = \sum_{e \in \text{out of } A} f(e) - \sum_{e \in \text{into } A} f(e) \leq c(A, B)$.
Corollary: Cuts and Flows

Really important corollary of flow-value lemma

Corollary: Let \( f \) be any \( s-t \) flow and let \((A, B)\) be any \( s-t \) cut. Then \( v(f) \leq c(A, B) \).

Proof:

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

\[
\leq \sum_{e \text{ out of } A} c(e)
\]

\[
\leq c(A, B)
\]

Duality

Illustration on board (leave up). "Duality"

Claim: If there is a flow \( f^* \) and cut \((A^*, B^*)\) such that \( v(f^*) = c(A^*, B^*) \), then

\( f^* \) is a maximum flow

\( (A^*, B^*) \) is a minimum cut

F-F returns a maximum flow

Claim: Suppose \( f^* \) is an \( s-t \) flow such that there are no residual paths in \( G_{f^*} \) (e.g., the flow returned by F-F)

... and let \((A^*, B^*)\) be the \( s-t \) cut where \( A^* \) consists of all nodes reachable from \( s \) in the residual graph.

Then the following are true:
F-F returns a maximum flow

**Claim**: Suppose $f^*$ is an $s$-$t$ flow such that there are no residual paths in $G_{f^*}$ (e.g., the flow returned by F-F).

... and let $(A^*, B^*)$ be the $s$-$t$ cut where $A^*$ consists of all nodes reachable from $s$ in the residual graph.

Then the following are true:
- $v(f^*) = C(A^*, B^*)$
- $f^*$ is a maximum flow
- $(A^*, B^*)$ is a minimum cut

**Corollary**: F-F returns a maximum flow.

---

Max-Flow Min-Cut Theorem

**Theorem**: In every flow network, the value of the maximum $s$-$t$ flow is equal to the value of the minimum $s$-$t$ cut.

**Proof?**

Prove claim on board
Max-Flow Min-Cut Theorem

**Theorem:** in every flow network, the value of the maximum $s$-$t$ flow is equal to the value of the minimum $s$-$t$ cut.

**Proof?**

For any flow network, run F-F to get a maximum flow $f^*$ and minimum-cut $(A^*, B^*)$ such that $v(f^*) = c(A^*, B^*)$. 