Announcements

• Reading for this week
  – 6.8, 7.1, 7.2 [7.3-7.4 will not be covered]
  – Next week: 7.5-7.12

• Final exam, March 18, 6:30 pm. At UW.
  – 2 hours
  – In class (CSE 303 / CSE 305)
  – Comprehensive
    • 67% post midterm, 33% pre midterm
Bellman-Ford Shortest Paths Algorithm

- Computes shortest paths from a starting vertex \( V \)
- Allows negative cost edges
  - Negative cost cycles identified
- Runtime \( O(nm) \)
- Easy to code
Bellman Ford Algorithm, Version 2

foreach \( w \)

\[ M[0, w] = \text{infinity}; \]

\[ M[0, v] = 0; \]

for \( i = 1 \) to \( n-1 \)

foreach \( w \)

\[ M[i, w] = \min(M[i-1, w], \min_x(M[i-1, x] + \text{cost}[x, w])) \]
Bellman Ford Algorithm, Version 3

foreach w

    M[w] = infinity;

M[v] = 0;

for i = 1 to n-1

    foreach w

        M[w] = \min(M[w], \min_x (M[x] + cost[x,w]))
Bellman Ford Example
Finding the longest path in a graph

- Negate weights
- Short paths
- Negate answer
Foreign Exchange Arbitrage

<table>
<thead>
<tr>
<th></th>
<th>USD</th>
<th>EUR</th>
<th>CAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>USD</td>
<td>----</td>
<td>0.8</td>
<td>1.2</td>
</tr>
<tr>
<td>EUR</td>
<td>1.2</td>
<td>----</td>
<td>1.6</td>
</tr>
<tr>
<td>CAD</td>
<td>0.8</td>
<td>0.6</td>
<td>----</td>
</tr>
</tbody>
</table>

Cycle product > 1
Network Flow
Outline

• Network flow definitions
• Flow examples
• Augmenting Paths
  • Residual Graph
  • Ford Fulkerson Algorithm
• Cuts
  • Maxflow-MinCut Theorem
Network Flow Definitions

- Capacity
- Source, Sink $s, t$
- Capacity Condition
- Conservation Condition
  \[ \text{Flow in} = \sum \text{Flow out} \]
- Value of a flow

\[ c(e) \geq 0 \]
\[ f(e) \geq 0 \]
Flow Example
Flow assignment and the residual graph

\[ \text{Flow assignment: } \frac{15}{20} \quad \frac{0}{10} \quad 15/30 \quad 5/10 \quad 20/20 \]

\[ \text{Residual graph: } 5 \quad 15 \quad 15 \quad 15 \quad 20 \]

\[ c: \xi \]

\[ c - \xi \]
Network Flow Definitions

- Flowgraph: Directed graph with distinguished vertices $s$ (source) and $t$ (sink)
- Capacities on the edges, $c(e) \geq 0$
- Problem, assign flows $f(e)$ to the edges such that:
  - $0 \leq f(e) \leq c(e)$
  - Flow is conserved at vertices other than $s$ and $t$
    - Flow conservation: flow going into a vertex equals the flow going out
  - The flow leaving the source is as large as possible
Flow Example
Find a maximum flow

Construct a maximum flow and indicate the flow value

Value of flow:
Find a maximum flow

Diagram of a network flow problem with capacities on the edges.
Augmenting Path Algorithm

- **Augmenting path**
  - Vertices $v_1, v_2, \ldots, v_k$
    - $v_1 = s$, $v_k = t$
    - Possible to add $b$ units of flow between $v_j$ and $v_{j+1}$ for $j = 1 \ldots k-1$

![Graph with labeled vertices and edges showing augmenting path]

$s, u, v, t$ (5)
$s, v, u, t$ (5)
Find two augmenting paths
Residual Graph

- Flow graph showing the remaining capacity
- Flow graph $G$, Residual Graph $G_R$
  - $G$: edge $e$ from $u$ to $v$ with capacity $c$ and flow $f$
  - $G_R$: edge $e'$ from $u$ to $v$ with capacity $c - f$
  - $G_R$: edge $e''$ from $v$ to $u$ with capacity $f$
Residual Graph

Graph 1:
- s to u: 15/20
- u to t: 0/10
- s to v: 5/10
- v to t: 20/20
- s to v: 15/30

Graph 2:
- s to u: 5
- u to v: 15
- v to u: 15
- v to t: 15
- s to t: 10
- s to v: 5
- v to s: 5
- v to t: 20
Augmenting Path Lemma

- Let $P = v_1, v_2, \ldots, v_k$ be a path from $s$ to $t$ with minimum capacity $b$ in the residual graph.
- $b$ units of flow can be added along the path $P$ in the flow graph.
Proof

- Add $b$ units of flow along the path $P$
- What do we need to verify to show we have a valid flow after we do this?
  - Capacity constraints
    \[ c(e) \geq f(e) \geq 0 \]
  - Conservation of flow

\[ 0 + b + b = 0 \]
\[ -b = 0 \]

Case 1: forward edge
\[ c - f \geq b \]

Case 2: backward edge
\[ f \geq b \]
Ford-Fulkerson Algorithm (1956)

while not done

Construct residual graph $G_R$

Find an s-t path $P$ in $G_R$ with capacity $b > 0$

Add $b$ units along in $G$

If the sum of the capacities of edges leaving $S$ is at most $C$, then the algorithm takes at most $C$ iterations
Cuts in a graph

- Cut: Partition of $V$ into disjoint sets $S$, $T$ with $s$ in $S$ and $t$ in $T$.
- $\text{Cap}(S,T)$: sum of the capacities of edges from $S$ to $T$
- $\text{Flow}(S,T)$: net flow out of $S$
  - Sum of flows out of $S$ minus sum of flows into $S$
- $\text{Flow}(S,T) \leq \text{Cap}(S,T)$

3 + 2 + 2 - 4
What is $\text{Cap}(S,T)$ and $\text{Flow}(S,T)$

$S = \{s, a, b, e, h\}$, \hspace{0.5cm} $T = \{c, f, i, d, g, t\}$
Minimum value cut

\[ s \leq 3 \leq u, v, t \leq 3 \leq \{s, u, v, t \} \leq 3 \leq \{s, u, v, t \} \leq 3 \leq \{s, u, v, t \} \leq 3 \leq \{s, u, v, t \} \leq 3 \]

Diagram:
- Node s connected to u with weight 40
- Node u connected to v with weight 10
- Node v connected to t with weight 40
- Node s connected to t with weight 10
Find a minimum value cut

\[ \text{cap}(s, t) \geq \text{flow}(s, t) \]
\[ \text{cap}(s, t) = \text{flow}(s, t) \]
MaxFlow – MinCut Theorem

- Let $S, T$ be a cut, and $F$ a flow
  - $\text{Cap}(S,T) \geq \text{Flow}(S,T)$
- If $\text{Cap}(S,T) = \text{Flow}(S,T)$
  - $S, T$ must be a minimum cut
  - $F$ must be a maximum flow
- The amazing Ford-Fulkerson theorem shows that there is always a cut that matches a flow, and also shows how their algorithm finds the flow
MaxFlow – MinCut Theorem

- There exists a flow which has the same value of the minimum cut
- Proof: Consider a flow where the residual graph has no s-t path with positive capacity
- Let S be the set of vertices in $G_R$ reachable from s with paths of positive capacity
Let $S$ be the set of vertices in $G_R$ reachable from $s$ with paths of positive capacity.

What can we say about the flows and capacity between $u$ and $v$?
Max Flow - Min Cut Theorem

- Ford-Fulkerson algorithm finds a flow where the residual graph is disconnected, hence FF finds a maximum flow.

- If we want to find a minimum cut, we begin by looking for a maximum flow.
Performance

- The worst case performance of the Ford-Fulkerson algorithm is horrible
Better methods of finding augmenting paths

• Find the maximum capacity augmenting path
  – $O(m^2 \log(C))$ time algorithm for network flow

• Find the shortest augmenting path
  – $O(m^2 n)$ time algorithm for network flow

• Find a blocking flow in the residual graph
  – $O(mn \log n)$ time algorithm for network flow
Problem Reduction

- Reduce Problem A to Problem B
  - Convert an instance of Problem A to an instance of Problem B
  - Use a solution of Problem B to get a solution to Problem A
- Practical
  - Use a program for Problem B to solve Problem A
- Theoretical
  - Show that Problem B is at least as hard as Problem A
Problem Reduction Examples

- Reduce the problem of finding the Maximum of a set of integers to finding the Minimum of a set of integers

Find the maximum of: 8, -3, 2, 12, 1, -6

Construct an equivalent minimization problem
Undirected Network Flow

- Undirected graph with edge capacities
- Flow may go either direction along the edges (subject to the capacity constraints)

Construct an equivalent flow problem
Bipartite Matching

• A graph $G=(V,E)$ is bipartite if the vertices can be partitioned into disjoint sets $X, Y$.

• A matching $M$ is a subset of the edges that does not share any vertices.

• Find a matching as large as possible.