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
- Allows negative cost edges
  - Negative cost cycles identified
- Runtime $O(nm)$
- Easy to code

Bellman Ford Algorithm, Version 2

```plaintext
foreach w
  M[0, w] = 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] + cost[x,w]))
```

Bellman Ford Algorithm, Version 3

```plaintext
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

![Graph](image)
Finding the longest path in a graph

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>

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
• Capacity Condition
• Conservation Condition
• Value of a flow

Flow Example
Flow assignment and the residual graph

Flow Example

Find a maximum flow

Find a maximum flow

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

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 $j = 1 \ldots k-1$

Value of flow:

Construct a maximum flow and indicate the flow value
Find two augmenting paths

Residual Graph

- Flow graph showing the remaining capacity
- Flow graph $G_1$, 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

Build the residual graph

Augmenting Path Lemma

- Let $P = v_1, v_2, ..., v_k$ 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?

-
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 $G$

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 $T$.
- Cap($S$, $T$): sum of the capacities of edges from $S$ to $T$.
- Flow($S$, $T$): net flow out of $S$.
  - Sum of flows out of $S$ minus sum of flows into $S$.
- Flow($S$, $T$) $\leq$ Cap($S$, $T$)

What is Cap($S$, $T$) and Flow($S$, $T$)?

$S = \{s, a, b, e, h\}, \quad T = \{c, f, i, d, g, t\}$

Minimum value cut

MaxFlow – MinCut Theorem

- Let $S$, $T$ be a cut, and $F$ a flow.
  - Cap($S$, $T$) $\geq$ Flow($S$, $T$).
- If Cap($S$, $T$) = 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.
Max Flow – Min Cut 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

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
  - \( O(m^3 \log(C)) \) 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

History

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

Undirected Network Flow

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

Bipartite Matching

- A graph $G=(V,E)$ is bipartite if the vertices can be partitioned into disjoints 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

Application

- A collection of teachers
- A collection of courses
- And a graph showing which teachers can teach which courses

Converting Matching to Network Flow
Finding edge disjoint paths

Theorem

• The maximum number of edge disjoint paths equals the minimum number of edges whose removal separates s from t