Shortest Paths and Network Flow

CSE 421
Algorithms
Richard Anderson
Lecture 21
Shortest Paths and Network Flow

Shortest Path Problem

• Dijkstra’s Single Source Shortest Paths Algorithm
  – $O(m \log n)$ time, positive cost edges
• Bellman-Ford Algorithm
  – $O(mn)$ time for graphs which can have negative cost edges

Lemma

• If a graph has no negative cost cycles, then the shortest paths are simple paths
• Shortest paths have at most $n-1$ edges

Shortest paths with a fixed number of edges

• Find the shortest path from $s$ to $w$ with exactly $k$ edges

Express as a recurrence

• Compute distance from starting vertex $s$
• $O_p_k(w) = \min_x [O_{p_{k-1}}(x) + c_{sw}]$
• $O_0(w) = 0$ if $w = s$ and infinity otherwise
Algorithm, Version 1

for each w
  M[0, w] = infinity;
  M[0, s] = 0;
for i = 1 to n - 1
  for each w
    M[i, w] = min(M[i-1, x] + cost(x, w));

Algorithm, Version 2

for each w
  M[0, w] = infinity;
  M[0, s] = 0;
for i = 1 to n - 1
  for each w
    M[i, w] = min(M[i-1, w], min_x(M[i-1, x] + cost(x, w)));

Algorithm, Version 3

for each w
  M[w] = infinity;
  M[s] = 0;
for i = 1 to n - 1
  for each w
    M[w] = min(M[w], min_x(M[x] + cost(x, w)));

Algorithm, Version 4

for each w
  M[w] = infinity;
  M[s] = 0;
for i = 1 to n - 1
  for each w
    for each x
      if (M[w] > M[x] + cost(x, w))
        P[w] = x;
        M[w] = M[x] + cost(x, w);

Correctness Proof for Algorithm 3

• Key lemma – at the end of iteration i, for all w, M[w] <= M[i, w];

If the pointer graph has a cycle, then the graph has a negative cost cycle

• If P[w] = x then M[w] >= M[x] + cost(x, w)
  – Equal when w is updated
  – M[x] could be reduced after update
• Let v_1, v_2, ..., v_k be a cycle in the pointer graph with (v_k, v_1) the last edge added
  – Just before the update
    • M[v_j] >= M[v_{j+1}] + cost(v_{j+1}, v_j) for j < k
    • M[v_k] > M[v_1] + cost(v_1, v_k)
  – Adding everything up
    • 0 > cost(v_1, v_2) + cost(v_2, v_3) + ... + cost(v_k, v_1)
Negative Cycles

- If the pointer graph has a cycle, then the graph has a negative cycle
- Therefore: if the graph has no negative cycles, then the pointer graph has no negative cycles

Finding negative cost cycles

- What if you want to find negative cost cycles?

What about finding Longest Paths

- Can we just change Min to Max?

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>1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EUR</td>
<td>0.6</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>CAD</td>
<td>0.8</td>
<td>0.6</td>
<td>1.2</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

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

Find a maximum flow

Value of flow:

Find a maximum flow

Construct a maximum flow and indicate the flow value
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 \(j = 1 \ldots k-1\)