Shortest Paths with Dynamic Programming

1. **Announcements**
   - Dynamic Programming Reading:
     - 6.8 Shortest Paths (Bellman-Ford)
   - Network Flow Reading
     - 7.1-7.3, Network Flow Problem and Algorithms
     - 7.5-7.12, Network Flow Applications

2. **Shortest Path Problem**
   - Dijkstra’s Single Source Shortest Paths Algorithm
     - $O(m \log n)$ time, positive cost edges
   - Directed Acyclic Graphs
     - $O(n + m)$, Topological Sort + DP
   - Bellman-Ford Algorithm
     - $O(mn)$ time for graphs which can have negative cost edges

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

4. **Shortest paths with a fixed number of edges**
   - Find the shortest path from $s$ to $w$ with exactly $k$ edges

5. **Express as a recurrence**
   - Compute distance from starting vertex $s$
     - $\text{Opt}_k(w) = \min_x [\text{Opt}_{k-1}(x) + c_{xw}]$
     - $\text{Opt}_0(w) = 0$ 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_x(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]));

Example:

Correctness Proof for Algorithm 3

- Key lemma – at the end of iteration i, for all w, M[w] ≤ M[i, 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];
Theorem

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

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

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

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>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>EUR</td>
<td>1.2</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>CAD</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
</tr>
</tbody>
</table>