CSE 417: Algorithms with Complexity

Lecture 23
Shortest Paths Problem and
Dynamic Programming

Announcements
• No class on Friday

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

Dynamic Programming
• Express problem as an optimization
• Order subproblems so that results are computed in proper order

Shortest Paths as DP
• $\text{Dist}_s[s] = 0$
• $\text{Dist}_s[v] = \min_w [\text{Dist}_w[w] + c_{wv}]$
• How do we order the computation
  • Directed Acyclic graph: Topological Sort
  • Dijkstra’s algorithm determines an order

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 given 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

\[ \text{Opt}_k(w) = \min_x \left[ \text{Opt}_{k-1}(x) + c_{xw} \right] \]
\[ \text{Opt}_0(w) = 0 \text{ if } w = s \text{ and infinity otherwise} \]

Algorithm, Version 1

```plaintext
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

```plaintext
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

```plaintext
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 lemmas, for all w:
  - There exists a path of length $M[w]$ from s to w
  - At the end of iteration $i$, $M[w] \leq M[i, w]$;

Algorithm, Version 4

for each w
  $M[w] = \infty$;
  $M[s] = 0$;
for $i = 1$ to $n-1$
  for each w
    for each x
      if ($M[w] > M[x] + c[x, w]$)
        $P[w] = x$;
        $M[w] = M[x] + c[x, w]$;

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

- If $P[w] = x$ then $M[w] \geq M[x] + c(x, w)$
  - 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
    - $M[v_j] \geq M[v_{j+1}] + c(v_j, v_{j+1})$ for $j < k$
    - $M[v_k] > M[v_1] + c(v_1, v_k)$
  - Adding everything up
    - $0 > c(v_1, v_2) + c(v_2, v_3) + \ldots + c(v_k, v_1)$

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?
Finding the longest Path in a DAG

What about finding Longest Paths in a directed graph
- 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>0.8</td>
<td>0.6</td>
</tr>
<tr>
<td>EUR</td>
<td>1.2</td>
<td>1.2</td>
<td>1.6</td>
</tr>
<tr>
<td>CAD</td>
<td>0.8</td>
<td>0.6</td>
<td>1.6</td>
</tr>
</tbody>
</table>