Lecture 12:
• Shortest Paths: Finish Dijkstra, Bellman-Ford

Feb 26, 2020
Shortest Paths:
Bellman-Ford
Dijkstra Recap

• **Input:** Directed, weighted graph $G = (V, E, \{\ell_e\})$, source node $s$
  - Non-negative edge lengths $\ell_e \geq 0$

• **Output:** Two arrays $d, p$
  - $d(u)$ is the length of the shortest $s \leadsto u$ path
  - $p(u)$ is the final hop on shortest $s \leadsto u$ path

• **Running time:** $O(m \log n)$
Invariant breaks down

Explored B, but don't know its distance yet.
Ask the Audience

• Show that Dijkstra’s Algorithm can fail in graphs with negative edge lengths
Why Care About Negative Lengths?

• Models various phenomena
  • Transactions (credits and debits)
  • Currency exchange (log exchange rate can be + or -)
  • Chemical reactions (can be exo- or endo-thermic)
  • ...

• Leads to interesting algorithms
  • Variants of Bellman-Ford are used in internet routing
Bellman-Ford

• **Input:** Directed, weighted graph $G = (V, E, \{\ell_e\})$, source node $s$
  • Possibly negative edge lengths $\ell_e \in \mathbb{R}$
  • No negative-length cycles!
    (Might not be a shortest path)

• **Output:** Two arrays $d, p$
  • $d(u)$ is the length of the shortest $s \leadsto u$ path
  • $p(u)$ is the final hop on shortest $s \leadsto u$ path
Ask the Audience

• Why won't the following work?
  • Take a graph $G = (V, E, \{\ell(e)\})$ with negative lengths
  • Add $\min \ell(e)$ to all lengths to make them non-negative
  • Run Dijkstra on the new graph
Structure of Shortest Paths

• If $(u, v) \in E$, then $d(s, v) \leq d(s, u) + \ell(u, v)$ for every node $s \in V$
  
  If "the" shortest path from $s \leadsto v$ ends with the edge $(u \rightarrow v)$ then $d(s, v) = d(s, u) + \ell(u \rightarrow v)$

  $d(s, v) = \min_{(u, v) \in E} d(s, u) + \ell(u, v)$

• If $(u, v) \in E$, and $d(s, v) = d(s, u) + \ell(u, v)$ then there is a shortest $s \leadsto v$-path ending with $(u, v)$
**Dynamic Programming**

$$OPT(v) = \text{length of the shortest path from } s \to v$$

$$OPT(v) = \min_{(u,v) \in E} \left( OPT(u) + l(u,v) \right)$$

$$OPT(s) = 0$$

*If the graph can't be topologically ordered then we cannot do bottom-up dynamic programming.*
Dynamic Programming
Dynamic Programming Take II

$$\text{OPT}(v, j) = \text{the length of the shortest } s \rightarrow v \text{ path using } \leq j \text{ hops.}$$

$$\text{OPT}(v, j) = \min_{(u,v) \in E} \text{OPT}(u, j-1) + l(u,v)$$

$$\text{OPT}(s, j) = 0 \quad \forall j$$

$$\text{OPT}(v, 0) = \infty \quad \forall v \neq s$$
Recurrence

• **Subproblems:** $\text{OPT}(v, j)$ is the length of the shortest $s \leadsto v$ path with at most $j$ hops

• **Case u:** $(u, v)$ is final edge on the shortest $s \leadsto v$ path with at most $j$ hops

**Recurrence:**

$$
\text{OPT}(v, j) = \min \left\{ \text{OPT}(v, j - 1), \min_{(u,v) \in E} \{ \text{OPT}(u, j - 1) + \ell_{u,v} \} \right\}
$$

$\text{OPT}(s, j) = 0$ for every $j$

$\text{OPT}(v, 0) = \infty$ for every $v$
Finding the paths

• $\text{OPT}(v, j)$ is the length of the shortest $s \sim v$ path with at most $j$ hops

• $P(v, j)$ is the last hop on some shortest $s \sim v$ path with at most $j$ hops

Recurrence:

$$\text{OPT}(v, j) = \min \left\{ \text{OPT}(v, i - 1), \min_{(u,v) \in E} \{ \text{OPT}(u, i - 1) + \ell_{u,v} \} \right\}$$
Example

Graph:

- s to b: -1
- b to c: 1
- s to c: 3
- b to e: 2
- c to d: 5
- d to e: -3

Table:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>∞</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>∞</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>∞</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>∞</td>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
Example

The diagram shows a graph with nodes labeled as `s`, `c`, `b`, `d`, and `e`. The edges between the nodes are labeled with weights, and there is a table showing the distances between each pair of nodes.

### Graph

- **s** connects to **c** with a weight of 4
- **c** connects to **b** with a weight of 3
- **b** connects to **d** with a weight of 2
- **b** connects to **e** with a weight of 1
- **c** connects to **d** with a weight of 5
- **d** connects to **e** with a weight of -3

### Distance Table

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>s</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>b</strong></td>
<td>∞</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>c</strong></td>
<td>∞</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>d</strong></td>
<td>∞</td>
<td>∞</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>e</strong></td>
<td>∞</td>
<td>∞</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

![Graph diagram]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>∞</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>c</td>
<td>∞</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>∞</td>
<td>∞</td>
<td>1</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>e</td>
<td>∞</td>
<td>∞</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Every shortest path has at most \( n-1 \) edges.
Implementation (Bottom Up)

Shortest-Path(G, s)

\[
\text{foreach node } v \in V \\
M[0,v] \leftarrow \infty \\
P[0,v] \leftarrow \phi \\
M[0,s] \leftarrow 0 \\
\]

for i = 1 to n-1

\[
\text{foreach node } v \in V \\
M[i,v] \leftarrow M[i-1,v] \\
P[i,v] \leftarrow P[i-1,v] \\
\text{foreach edge } (v, w) \in E \\
\text{if } (M[i-1,w] + \ell_{wv} < M[i,v]) \\
\quad M[i,v] \leftarrow M[i-1,w] + \ell_{wv} \\
P[i,v] \leftarrow w
\]

Worst-case running time is \(O(nm)\)
Optimizations

• One array $d[v]$ containing shortest path found so far
• No need to check edges $(u, v)$ unless $d[u]$ has changed
• Stop if no $d[v]$ has changed for a full pass through $V$

• **Theorem:**
  • Throughout the algorithm $M[v]$ is the length of some $s - v$ path
  • After $i$ passes through the nodes, $M[v] \leq OPT(v, i)$
Efficient-Shortest-Path (G, s)

foreach node v ∈ V
    M[v] ← ∞
    P[v] ← φ
    M[s] ← 0

for i = 1 to n-1
    foreach node w ∈ V
        if (M[w] changed in the last iteration)
            foreach edge (w,v) ∈ E
                if (M[w] + ℓ_{wv} < M[v])
                    M[v] ← M[w] + ℓ_{wv}
                    P[v] ← w
    if (no M[w] changed): return M

Running time is $O(m \cdot \text{diameter})$
Negative Cycle Detection

• **Claim 1:** if $OPT(v, n) = OPT(v, n - 1)$ then there are no negative cycles reachable from $s$

• **Claim 2:** if $OPT(v, n) < OPT(v, n - 1)$ then any shortest $s \rightarrow v$ path contains a negative cycle
Negative Cycle Detection

**Algorithm:**
- Pick a node \( a \in V \)
- Run Bellman-Ford for \( n \) iterations
- Check if \( OPT(v, n) \neq OPT(v, n - 1) \) for some \( v \in V \)
  - If no, then there are no negative cycles
  - If yes, the shortest \( a - v \) path contains a negative cycle
Negative Cycle Detection

• **Algorithm:**
  • Add a new node $s \in V$, add edges $(s, v)$ for every $v \in V$
  • Run Bellman-Ford for $n$ iterations
  • Check if $OPT(v, n) \neq OPT(v, n - 1)$ for some $v \in V$
    • If no, then there are no negative cycles
    • If yes, the shortest $s - v$ path contains a negative cycle

![Graph Diagram]

- $a$ to $b$: $-1$
- $b$ to $c$: $2$
- $b$ to $e$: $2$
- $c$ to $a$: $3$
- $c$ to $d$: $1$
- $c$ to $b$: $5$
- $d$ to $c$: $-6$
- $d$ to $e$: $2$
- $e$ to $b$: $3$
• **Input:**

• **Informal Version:**
  - Maintain a set $S$ of explored nodes
  - Maintain an upper bound on distance
    - If $u$ is explored, then we know $d(u)$ **(Key Invariant)**
    - If $u$ is explored, and $(u, v)$ is an edge, then we know $d(v) \leq d(u) + \ell(u, v)$
  - Explore the “closest” unexplored node
  - Repeat until we’re done
Shortest Paths Summary

• **Input:** Directed, weighted graph $G = (V, E, \{\ell_e\})$, source node $s$

• **Output:** Two arrays $d, p$
  - $d(u)$ is the length of the shortest $s \leadsto u$ path
  - $p(u)$ is the final hop on shortest $s \leadsto u$ path

• **Non-negative lengths** ($\ell_e \geq 0$): Dijkstra’s Algorithm can solve in $O(m \log n)$ time

• **Negative lengths:** Bellman-Ford solves in $O(nm)$ time, or finds a negative-length cycle