6. Dynamic Programming II

- sequence alignment
- Hirschberg’s algorithm
- Bellman-Ford
- distance vector protocols
- negative cycles in a digraph

Shortest paths

**Shortest path problem.** Given a digraph $G = (V, E)$, with arbitrary edge weights or costs $c_{vw}$, find cheapest path from node $s$ to node $t$.

```
source s
      5                  3
       4                  1
         8                12
              7--7--2
         11               9
      13                13
        5--10--6
        10
```

cost of path = $9 - 3 + 1 + 11 = 18$

destination $t$

Shortest paths: failed attempts

**Dijkstra.** Can fail if negative edge weights.

```
S         2    W
|         1    |
|           3  |

V         8    W
```

**Reweighting.** Adding a constant to every edge weight can fail.

```
S         5    1
|         2    |
|           2  |

V         5    6
|         3    |
|           3  |
```

Negative cycles

**Def.** A negative cycle is a directed cycle such that the sum of its edge weights is negative.

```
S         5    W
|         2    |
|           2  |

V         5    1
|         3    |
|           3  |
```

A negative cycle $W$: $c(W) = \sum_{e \in W} c_e < 0$
Shortest paths and negative cycles

**Lemma 1.** If some path from \( v \) to \( t \) contains a negative cycle, then there does not exist a cheapest path from \( v \) to \( t \).

**Pf.** If there exists such a cycle \( W \), then can build a \( v \rightarrow t \) path of arbitrarily negative weight by detouring around cycle as many times as desired. □

\[
W \quad c(W) < 0
\]

\[
\text{v to t}
\]

**Lemma 2.** If \( G \) has no negative cycles, then there exists a cheapest path from \( v \) to \( t \) that is simple (and has \( \leq n - 1 \) edges).

**Pf.**
- Consider a cheapest \( v \rightarrow t \) path \( P \) that uses the fewest number of edges.
- If \( P \) contains a cycle \( W \), can remove portion of \( P \) corresponding to \( W \) without increasing the cost. □

\[
W \quad c(W) \geq 0
\]

Shortest paths and negative cycle problems

**Shortest path problem.** Given a digraph \( G = (V, E) \) with edge weights \( c_{vw} \) and no negative cycles, find cheapest \( v \rightarrow t \) path for each node \( v \).

**Negative cycle problem.** Given a digraph \( G = (V, E) \) with edge weights \( c_{vw} \), find a negative cycle (if one exists).

\[
\begin{array}{c}
\text{shortest-paths tree} \\
\text{negative cycle}
\end{array}
\]

Shortest paths: dynamic programming

**Def.** \( OPT(i, v) \) = cost of shortest \( v \rightarrow t \) path that uses \( \leq i \) edges.

- **Case 1:** Cheapest \( v \rightarrow t \) path uses \( \leq i - 1 \) edges.
  - \( OPT(i, v) = OPT(i - 1, v) \)

- **Case 2:** Cheapest \( v \rightarrow t \) path uses exactly \( i \) edges.
  - if \((v, w)\) is first edge, then \( OPT \) uses \((v, w)\), and then selects best \( w \rightarrow t \) path using \( \leq i - 1 \) edges

\[
OPT(i, v) = \begin{cases}
\infty & \text{if } i = 0 \\
\min \left\{ OPT(i - 1, v), \min_{(v, w) \in E} \left\{ OPT(i - 1, w) + c_{w} \right\} \right\} & \text{otherwise}
\end{cases}
\]

**Observation.** If no negative cycles, \( OPT(n - 1, v) \) = cost of cheapest \( v \rightarrow t \) path.

**Pf.** By Lemma 2, cheapest \( v \rightarrow t \) path is simple. □
**Shortest paths: implementation**

**Shortest-Paths**\((V, E, c, t)\)

**FOREACH** node \(v \in V\)

\[
M[0, v] \leftarrow \infty.
\]

\[
M[0, t] \leftarrow 0.
\]

**FOR** \(i = 1\) **TO** \(n - 1\)

**FOREACH** node \(v \in V\)

\[
M[i, v] \leftarrow M[i-1, v].
\]

**FOREACH** edge \((v, w) \in E\)

\[
M[i, v] \leftarrow \min \{ M[i, v], M[i-1, w] + c_{vw} \}.
\]

---

**Finding the shortest paths.**

- **Approach 1:** Maintain a \(\text{successor}(i, v)\) that points to next node on cheapest \(v \rightarrow t\) path using at most \(i\) edges.
- **Approach 2:** Compute optimal costs \(M[i, v]\) and consider only edges with \(M[i, v] = M[i-1, w] + c_{vw}\).

---

**Bellman-Ford: efficient implementation**

**Bellman-Ford**\((V, E, c, t)\)

**FOREACH** node \(v \in V\)

\[
d(v) \leftarrow \infty.
\]

\[
\text{successor}(v) \leftarrow \text{null}.
\]

\[
d(t) \leftarrow 0.
\]

**FOR** \(i = 1\) **TO** \(n - 1\)

**FOREACH** node \(w \in V\)

**IF** \((d(w)\) was updated in previous iteration)

**FOREACH** edge \((v, w) \in E\)

**IF** \((d(v) > d(w) + c_{vw})\)

\[
d(v) \leftarrow d(w) + c_{vw}.
\]

\[
\text{successor}(v) \leftarrow w.
\]

**IF** no \(d(w)\) value changed in iteration \(i\), **STOP**.
Bellman-Ford: analysis

Claim. After the $i^{th}$ pass of Bellman-Ford, $d(v)$ equals the cost of the cheapest $v \rightarrow t$ path using at most $i$ edges.

Counterexample. Claim is false!

Consider nodes in order: $t, 1, 2, 3$

<table>
<thead>
<tr>
<th>Node</th>
<th>$d(v)$</th>
<th>$s(v)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v$</td>
<td>3</td>
<td>$s(2) = 1$</td>
</tr>
<tr>
<td>$w$</td>
<td>2</td>
<td>$s(1) = t$</td>
</tr>
<tr>
<td>$t$</td>
<td>0</td>
<td>$s(3) = t$</td>
</tr>
</tbody>
</table>

Bellman-Ford: analysis

Lemma 3. Throughout Bellman-Ford algorithm, $d(v)$ is the cost of some $v \rightarrow t$ path; after the $i^{th}$ pass, $d(v)$ is no larger than the cost of the cheapest $v \rightarrow t$ path using $\leq i$ edges.

Pf. [by induction on $i$]

* Assume true after $i^{th}$ pass.
* Let $P$ be any $v \rightarrow t$ path with $i+1$ edges.
* Let $(v, w)$ be the first edge on $P$ and let $P'$ be subpath from $w$ to $t$.
* By inductive hypothesis, $d(w) \leq c(P')$ since $P'$ is a $w \rightarrow t$ path with $i$ edges.
* After considering $v$ in pass $i+1$: $d(v) \leq c_{vw} + d(w) \leq c_{vw} + c(P') = c(P)$. □

Theorem 2. Given a digraph with no negative cycles, Bellman-Ford computes the costs of the cheapest $v \rightarrow t$ paths in $O(mn)$ time and $\Theta(n)$ extra space.

Pf. Lemmas 2 + 3. □

can be substantially faster in practice
Bellman-Ford: finding the shortest path

**Counterexample.** Claim is false!

\( \text{Cost of successor } v \rightarrow t \text{ path may have strictly lower cost than } d(v). \)

\( \text{Successor graph may have cycles.} \)

Consider nodes in order: \( t, 1, 2, 3, 4 \)

\[
\begin{align*}
d(1) &= 3 \\
d(2) &= 8 \\
d(3) &= 10 \\
d(4) &= 11 \\
d(t) &= 0
\end{align*}
\]

Bellman-Ford: finding the shortest path

**Theorem 3.** Given a digraph with no negative cycles, Bellman-Ford finds the cheapest \( s \rightarrow t \) paths in \( O(mn) \) time and \( \Theta(n) \) extra space.

**Pf.**

- The successor graph cannot have a negative cycle. \([\text{Lemma 4}]\)
- Thus, following the successor pointers from \( s \) yields a directed path to \( t \).
- Let \( s = v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k = t \) be the nodes along this path \( P \).
- Upon termination, if \( \text{successor}(v) = w \), we must have \( d(v) = d(w) + c_{vw} \).
  \( \text{(LHS and RHS are equal when } \text{successor}(v) \text{ is set; } d(v) \text{ decreases only when } \text{successor}(v) \text{ is reset)} \)
- Let \( v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k \) be the nodes along the cycle \( W \).
- Assume that \((v_k, v_1)\) is the last edge added to the successor graph.
- Just prior to that:
  \[
  \begin{align*}
d(v_1) &\geq d(v_2) + c(v_1, v_2) \\
d(v_2) &\geq d(v_1) + c(v_2, v_3) \\
\vdots & \vdots \\
d(v_{k-1}) &\geq d(v_k) + c(v_{k-1}, v_k) \\
d(v_k) &> d(v_{k-1}) + c(v_k, v_1)
  \end{align*}
  \]
- Adding inequalities yields
  \[
  c(v_1, v_2) + c(v_2, v_3) + \ldots + c(v_{k-1}, v_k) + c(v_k, v_1) < 0. \]

W is a negative cycle
6. Dynamic Programming II

- sequence alignment
- Hirschberg’s algorithm
- Bellman-Ford
- distance vector protocols
- negative cycles in a digraph

Distance vector protocols

**Distance vector protocols.** [*routing by rumor*]
- Each router maintains a vector of shortest path lengths to every other node (distances) and the first hop on each path (directions).
- Algorithm: each router performs \( n \) separate computations, one for each potential destination node.

**Ex.** RIP, Xerox XNS RIP, Novell’s IPX RIP, Cisco’s IGRP, DEC’s DNA Phase IV, AppleTalk’s RTMP.

**Caveat.** Edge costs may change during algorithm (or fail completely).

Path vector protocols

**Link state routing.**
- Each router also stores the entire path.
- Based on Dijkstra’s algorithm.
- Avoids “counting-to-infinity” problem and related difficulties.
- Requires significantly more storage.

**Ex.** Border Gateway Protocol (BGP), Open Shortest Path First (OSPF).
6. Dynamic Programming II

- sequence alignment
- Hirschberg’s algorithm
- Bellman-Ford
- distance vector protocol
- negative cycles in a digraph

Detecting negative cycles

Negative cycle detection problem. Given a digraph $G = (V, E)$, with edge weights $c_{vw}$, find a negative cycle (if one exists).

Currency conversion. Given $n$ currencies and exchange rates between pairs of currencies, is there an arbitrage opportunity?

Remark. Fastest algorithm very valuable!

Arbitrage opportunities

Currency conversion. Given $n$ currencies and exchange rates between pairs of currencies, is there an arbitrage opportunity?

Remark. Fastest algorithm very valuable!
Detecting negative cycles

**Lemma 5.** If $OPT(n, v) = OPT(n-1, v)$ for all $v$, then no negative cycle can reach $t$.

*Pf.* Bellman-Ford algorithm. •

**Lemma 6.** If $OPT(n, v) < OPT(n-1, v)$ for some node $v$, then (any) cheapest path from $v$ to $t$ contains a cycle $W$. Moreover $W$ is a negative cycle.

*Pf.* [by contradiction]

- Since $OPT(n, v) < OPT(n-1, v)$, we know that shortest $v \rightarrow t$ path $P$ has exactly $n$ edges.
- By pigeonhole principle, $P$ must contain a directed cycle $W$.
- Deleting $W$ yields a $v \rightarrow t$ path with $< n$ edges $\Rightarrow$ $W$ has negative cost. •

**Theorem 4.** Can find a negative cycle in $\Theta(mn)$ time and $\Theta(n^2)$ space.

*Pf.*

- Add new node $t$ and connect all nodes to $t$ with 0-cost edge.
- $G$ has a negative cycle iff $G'$ has a negative cycle than can reach $t$.
- If $OPT(n, v) = OPT(n-1, v)$ for all nodes $v$, then no negative cycles.
- If not, then extract directed cycle from path from $v$ to $t$.
  (cycle cannot contain $t$ since no edges leave $t$) •

**Theorem 5.** Can find a negative cycle in $O(mn)$ time and $O(n)$ extra space.

*Pf.*

- Run Bellman-Ford for $n$ passes (instead of $n-1$) on modified digraph.
- If no $d(v)$ values updated in pass $n$, then no negative cycles.
- Otherwise, suppose $d(s)$ updated in pass $n$.
- Define $pass(v) = \text{last pass in which } d(v) \text{ was updated}$.
- Observe $pass(s) = n$ and $pass(\text{successor}(v)) \geq pass(v) - 1$ for each $v$.
- Following successor pointers, we must eventually repeat a node.
- Lemma 4 $\Rightarrow$ this cycle is a negative cycle. •

*Remark.* See p. 304 for improved version and early termination rule.

(Tarjan’s subtree disassembly trick)