Bellman-Ford Algorithm
For Solving the Single Source
Shortest Path Problem

https://cs.pomona.edu/classes/cs140/
Outline

Topics and Learning Objectives
• Discuss and analyze the Bellman-Ford Algorithm

Assessments
• Bellman-Ford Walk-through
Dynamic Programming

An algorithm design technique/paradigm that typically takes one of the following forms:

1. Top-Down (memoization—cache results and use recursion)
2. Bottom-Up (tabulation—store results in a table)

Used to solve problems with the following properties:

• Overlapping subproblems and
• Optimal substructure
The Bellman-Ford Algorithm

A dynamic programming solution to the Single-Source Shortest Path problem (same problem solved by Dijkstra’s)

Input:
• a weighted graph $G = (V, E)$ where each edge has a length $c_e$ and
• a source vertex $s$

Output:
• The length of the shortest path from $s$ to all other vertices, or
• We output that we detected a negative cycle (invalid path lengths)

Key Idea: leverage overlapping subproblems and optimal substructure.
Example 1

What is the shortest path from S to T using 0 edges?

Subproblem: consider only a subset of the possible paths.
Example 1

What is the shortest path from S to T using 1 edge?
Example 1

What is the shortest path using 2 edges?
Example 1

What is the shortest path using 2 edges?

What is the shortest path using 3 edges?
Example 2

What is the shortest path with at most 1 edge?
Example 2

Shortest path with at most 2 edges
Example 2

We didn’t gain anything by adding the edge $d$.

Shortest path with \textbf{at most} 3 edges

Shortest path with \textbf{at most} 2 edges
Example 2

Shortest path with at most 4 edges
Example 2

If rainbow is the shortest path from S to T using at most 4 edges, then the red dashed line must be the shortest path from S to C using at most 3 edges.

This must be shortest path from S to C with at most 3 edges!
Example 2

The path from D to C is used as part of the shortest path from S to T. And as part of the shortest path from S to C.

Shortest path with at most 4 edges

Overlapping Subproblems

The path from D to C is used as part of the shortest path from S to T and from D to T (and ...
FUNCTION BellmanFord(G, start_vertex)

n = G.vertices.length

edges_lengths = [[INFINITY FOR v IN G.vertices] FOR _ IN [0 ..< n]]
edges_lengths[0, start_vertex] = 0

FOR num_edges IN [1 ..< n]     Why won’t we need more than n-1 edges?

    FOR v IN G.vertices
        min_len = INFINITY

        FOR (vFrom, v) IN G.edges     Cost to get to vFrom using i-1 edges
            len = edges_lengths[num_edges - 1, vFrom] + G.edges[vFrom, v].cost
            IF len < min_len
                min_len = len

        edges_lengths[num_edges, v] = min(edges_lengths[num_edges - 1, v], min_len)  Cost using at most i-1 edges

        Cost using at most i edges
FOR \text{num\_edges} \ \text{IN} \ [1 \ldots n] \\
\text{FOR} \ v \ \text{IN} \ \text{G.vertices} \\
\quad \text{min\_len} = \text{INFINITY} \\
\text{FOR} \ (v\text{From}, v) \ \text{IN} \ \text{G.edges} \\
\quad \text{len} = \text{lens}[\text{num\_edges} - 1, v\text{From}] + c \\
\quad \text{IF} \ \text{len} < \text{min\_len} \\
\quad\quad \text{min\_len} = \text{len} \\
\quad \text{lens}[\text{num\_edges}, v] = \min( \\
\text{lens}[\text{num\_edges} - 1, v], \text{min\_len})
What does a single cell denote?

```
FOR num_edges IN [1..<n]
    FOR v IN G.vertices
        min_len = INFINITY
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v], min_len)
```
Initialize first row
Lengths of paths from s to all other vertices using zero edges

```
FOR num_edges IN [1 ..< n]
    FOR v IN G.vertices
        min_len = INFINITY
    FOR (vFrom, v) IN G.edges
        len = lens[num_edges - 1, vFrom] + c
        IF len < min_len
            min_len = len
        lens[num_edges, v] = min(lens[num_edges - 1, v], min_len)
```

```
edges_lengths = [[INFINITY FOR v IN G.vertices] FOR _ IN [0 ..< n]]
edges_lengths[0, start_vertex] = 0
```

<table>
<thead>
<tr>
<th>i</th>
<th>s</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Diagram:
- `s` to `a`: 2
- `s` to `b`: 4
- `s` to `c`: 4
- `s` to `d`: 4
- `a` to `c`: 2
- `a` to `d`: 2
- `b` to `c`: 1
- `b` to `d`: 4
- `c` to `d`: 2
Initialize first row

Lengths of paths from $s$ to all other vertices using zero edges

Initialize first row

Lengths of paths from $s$ to all other vertices using zero edges

 edges_lengths = \[
\begin{array}{lllllll}
 0 & 0 & \infty & \infty & \infty & \infty & \infty \\
\end{array}
\]

Initialize first row

Lengths of paths from $s$ to all other vertices using zero edges

 edges_lengths[0, start_vertex] = 0

FOR num_edges IN [1..<n]
    FOR v IN G.vertices
        min_len = INFINITY
    FOR (vFrom, v) IN G.edges
        len = lens[num_edges - 1, vFrom] + c
        IF len < min_len
            min_len = len
        lens[num_edges, v] = min(
            lens[num_edges - 1, v], min_len)

FOR num_edges IN [1..<n]
    FOR v IN G.vertices
        min_len = INFINITY
    FOR (vFrom, v) IN G.edges
        len = lens[num_edges - 1, vFrom] + c
        IF len < min_len
            min_len = len
        lens[num_edges, v] = min(
            lens[num_edges - 1, v], min_len)
FOR num_edges IN [1 ..< n]
  FOR v IN G.vertices
    min_len = INFINITY
    FOR (vFrom, v) IN G.edges
      len = lens[num_edges - 1, vFrom] + c
      IF len < min_len
        min_len = len
        lens[num_edges, v] = min(len, lens[num_edges - 1, v], min_len)
FOR num_edges IN [1..<n]
    FOR v IN G.vertices
        min_len = INFINITY
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
        lens[num_edges, v] = min(lens[num_edges - 1, v], min_len)

num_edges = 1
v = a
minW = inf
minW = 2
FOR num_edges IN [1 ..< n]
  FOR v IN G.vertices
    min_len = INFINITY
    FOR (vFrom, v) IN G.edges
      len = lens[num_edges - 1, vFrom] + c
      IF len < min_len
        min_len = len
        lens[num_edges, v] = min(
          lens[num_edges - 1, v], min_len)

num_edges = 1
v = a
minW = inf
minW = 2
num_edges = 1
v = b
minW = inf
minW = 4

FOR num_edges IN [1 ..< n]
    FOR v IN G.vertices
        min_len = INFINITY
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v], min_len)
num_edges = 1
v = b
minW = inf
minW = 4

FOR num_edges IN [1 ..< n]
FOR v IN G.vertices
min_len = INFINITY
FOR (vFrom, v) IN G.edges
len = lens[num_edges - 1, vFrom] + c
IF len < min_len
min_len = len
lens[num_edges, v] = min(lens[num_edges - 1, v], min_len)
num_edges = 1
v = b
minW = inf
minW = 4

FOR num_edges IN [1 .. < n]
    FOR v IN G.vertices
        min_len = INFINITY
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v], min_len)

FOR num_edges IN [1 .. < n]
    FOR v IN G.vertices
        min_len = INFINITY
There are not any paths of length 1 from s to c or d.
num_edges = 2
v = s
minW = inf

FOR num_edges IN [1 ..< n]
    FOR v IN G.vertices
        min_len = INFINITY
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v], min_len)

v

<table>
<thead>
<tr>
<th>i</th>
<th>4</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>s</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
num_edges = 2
v = s
minW = inf
num_edges = 2
v = a
minW = inf
minW = 2

FOR num_edges IN [1..<n]
    FOR v IN G.vertices
        min_len = INFINITY
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v],
                    min_len)

FOR num_edges IN [1..<n]
    FOR v IN G.vertices
        min_len = INFINITY
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v],
                    min_len)
num_edges = 2
v = a
minW = inf
minW = 2
num_edges = 2
v = b
minW = inf
minW = 4
FOR $\text{num\_edges}$ IN $[1..\langle n \rangle]$
    FOR $v$ IN $G$.vertices
        $\text{min\_len} = \text{INFINITY}$
        FOR $(v\text{From}, v)$ IN $G$.edges
            $\text{len} = \text{lens}[\text{num\_edges} - 1, v\text{From}] + c$
            IF $\text{len} < \text{min\_len}$
                $\text{min\_len} = \text{len}$
                $\text{lens}[\text{num\_edges}, v] = \text{min}(\text{lens}[\text{num\_edges} - 1, v], \text{min\_len})$

$\text{num\_edges} = 2$
$v = b$
$\text{minW} = \text{inf}$
$\text{minW} = 4$
$\text{minW} = 3$
num_edges = 2
v = c
minW = inf
minW = 4

FOR num_edges IN [1 ..< n]
    FOR v IN G.vertices
        min_len = INFINITY
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v], min_len)
            v

FOR num_edges IN [1 ..< n]
    FOR v IN G.vertices
        minW = inf
        minW = 4
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v], min_len)
            v

FOR num_edges IN [1 ..< n]
    FOR v IN G.vertices
        minW = inf
        minW = 4
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v], min_len)
            v
FOR `num_edges` IN [1 ..< n]
  FOR `v` IN G.vertices
    `min_len` = INFINITY
    FOR (vFrom, v) IN G.edges
      `len` = lens[num_edges - 1, vFrom] + c
      IF `len` < `min_len`
        `min_len` = `len`
        `minW` = min(  
          lens[num_edges, v], `min_len`  
        )
  
`num_edges` = 2  
`v` = `d`  
`minW` = `inf`  
`minW` = 8
What is our output?

```
FOR num_edges IN [1 ..< n]
    FOR v IN G.vertices
        min_len = INFINITY
        FOR (vFrom, v) IN G.edges
            len = lens[num_edges - 1, vFrom] + c
            IF len < min_len
                min_len = len
                lens[num_edges, v] = min(
                    lens[num_edges - 1, v], min_len)
```
What is our output?

<table>
<thead>
<tr>
<th>i</th>
<th>4</th>
<th>0</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>v</td>
<td></td>
</tr>
</tbody>
</table>
What is our output?

Do we need the other rows of the table?
What is our output?

Do we need the other rows of the table?
Running Time of Bellman-Ford Algorithm?

```
FOR num_edges IN [1 ..< n]
  FOR v IN G.vertices
    min_len = INFINITY
    FOR (vFrom, v) IN G.edges
      len = lens[num_edges - 1, vFrom] + c
      IF len < min_len
        min_len = len
    lens[num_edges, v] = min(lens[num_edges - 1, v], min_len)
```

The inner two loops go through every edge once, ordered by the vertices.

- \(O(n^2)\)
- \(O(mn)\)
- \(O(n^3)\)
- \(O(m^2)\)
What about negative edges?

```
FOR num_edges IN [1 ..< n]
    FOR v IN G.vertices
        min_len = INFINITY
    FOR (vFrom, v) IN G.edges
        len = lens[num_edges - 1, vFrom] + c
        IF len < min_len
            min_len = len
            lens[num_edges, v] = min(
                lens[num_edges - 1, v], min_len)
```
What is the maximum number of edges on any real (not negative infinity) shortest path?
What is the maximum number of edges on any real (not negative infinity) shortest path?

Any additional edges will increase the path length, or otherwise must be part of a negative cycle.
Exercise
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Predecessor</th>
<th>$i-1$</th>
<th>$i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>S</td>
<td>∞</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>None</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>None</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>None</td>
<td>∞</td>
<td></td>
</tr>
</tbody>
</table>
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Predecessor</th>
<th>i – 1</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>S</td>
<td>∞</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>S</td>
<td>∞</td>
<td>7</td>
</tr>
<tr>
<td>D</td>
<td>None</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>E</td>
<td>None</td>
<td>∞</td>
<td></td>
</tr>
</tbody>
</table>
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Table is rotated when compared to previous example (easier to fit on the slide)
What is the shortest path from S to B?

Vertex | Predecessor | i−1 | i
--- | --- | --- | ---
S | S | 0 | 0
B | D | 3 | 3
C | S | 7 | 7
D | E | 4 | 4
E | B | 1 | -1

Last iteration is only to detect negative cycles.

Table is rotated when compared to previous example (easier to fit on the slide)
Summary of Bellman-Ford

• Single-source shortest path problem (like Dijkstra’s)

• Running time is O(nm)

• Works with negative weights

• Can detect negative cycles
  • Run the loop $n$ times and if a path length goes down, then you’ve found a negative cycle