Multiplying Edges and Maximization

\[ d(A, D) = \frac{1}{5} + \frac{1}{15} + \frac{1}{7} = \log(5) + \log(15) + \log(7) = 2.7 \]

Max

\[ -s \quad -s \quad -10 \quad 10 \]

\[ -2 \quad -2 \quad -4 \]
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

Exercise

• 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

Shortest path with at most 2 edges

We didn’t gain anything by adding the edge

Shortest path with at most 3 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.

Optimal Substructure
This must be shortest path from S to C with at most 3 edges!

Shortest path with at most 4 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.

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 ...)

Shortest path with at most 4 edges
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]

  FOR v IN G.vertices

    min_len = INFINITY

    FOR (vFrom, v) IN G.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)

Why won't we need more than n-1 edges?

Cost to get to vFrom using at most i-1 edges

Cost using at most i-1 edges

Cost using at most i 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 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
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
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)

Nothing to loop over
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 = 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)

i   | 0  | 1  | 2  | 3  |
---  |--- |--- |--- |--- |
0    | 0  | 2  | inf| inf|
1    | 0  | inf| inf| inf|
v    | s  | a  | b  | c  |
     |    |    |    | d  |
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
There are not any paths of length 1 from $s$ to $c$ or $d$.

```
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 = s
minW = inf
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)
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)
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 = 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 = 2
v = b
minW = inf
minW = 4
minW = 3

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 = INF
    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 = c
minW = inf
minW = 4

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

s a b c d v
\[
\text{FOR } \text{num\_edges} \text{ IN } [1 ..< n]
\]
\[
\text{FOR } v \text{ IN } G.\text{vertices}
\]
\[
\text{min\_len} = \text{INFINITY}
\]
\[
\text{FOR } (v\text{From}, v) \text{ IN } G.\text{edges}
\]
\[
\text{len} = \text{lens}[\text{num\_edges} - 1, v\text{From}] + c
\]
\[
\text{IF } \text{len} < \text{min\_len}
\]
\[
\text{min\_len} = \text{len}
\]
\[
\text{lens}[\text{num\_edges}, v] = \min(\text{lens}[\text{num\_edges} - 1, v], \text{min\_len})
\]

\begin{align*}
\text{num\_edges} &= 2 \\
v &= d \\
\text{minW} &= \text{inf} \\
\text{minW} &= 8
\end{align*}
What is our output?

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>∞</td>
<td>∞</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>
<td>∞</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>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>∞</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>∞</td>
</tr>
</tbody>
</table>

\[
\text{FOR } \text{num\_edges } \text{IN [1 ..< n]}
\]
\[
\text{FOR } v \text{ IN G.vertices}
\]
\[
\text{min\_len } = \text{INFINITY}
\]
\[
\text{FOR (vFrom, v) IN G.edges}
\]
\[
\text{len } = \text{lens[num\_edges - 1, vFrom] + c}
\]
\[
\text{IF len < min\_len}
\]
\[
\text{min\_len } = \text{len}
\]
\[
\text{lens[num\_edges, v] } = \text{min}(
\text{lens[num\_edges - 1, v], min\_len})
\]
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)

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>
</tbody>
</table>

s a b c d v
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.

\[ O(n^2) \quad O(mn) \quad O(n^3) \quad O(m^2) \]
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(
          len, lens[num_edges - 1, v], min_len)
```
What is the maximum number of edges on any real (not negative infinity) \textit{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?

Initialization

<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></td>
</tr>
<tr>
<td>B</td>
<td>None</td>
<td>∞</td>
<td></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>

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>
<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>None</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>C</td>
<td>None</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>D</td>
<td>None</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>E</td>
<td>None</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

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)

<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>5</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>S</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>∞</td>
<td>6</td>
</tr>
<tr>
<td>E</td>
<td>B</td>
<td>∞</td>
<td>1</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)

Last iteration is only to detect negative cycles.
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