All-Pairs Shortest Paths

6.046 Design and Analysis of Algorithms
MIT

Michael Kapralov

20 March 2014
Single-source shortest paths (6.006)

- Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(s, v) =$ shortest-path weight $s \rightarrow v$ for all $v \in V$
Single-source shortest paths (6.006)

- Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \to \mathbb{R}$
- Find $\delta(s, v) =$ shortest-path weight $s \to v$ for all $v \in V$
  (well-defined if no negative cycles)
Single-source shortest paths (6.006)

- Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(s, v)$ = shortest-path weight $s \rightarrow v$ for all $v \in V$ (well-defined if no negative cycles)
Single-source shortest paths (6.006)

- Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \to \mathbb{R}$
- Find $\delta(s, v) =$ shortest-path weight $s \to v$ for all $v \in V$ (well-defined if no negative cycles)
Single-source shortest paths (6.006)

- Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(s, v) =$ shortest-path weight $s \rightarrow v$ for all $v \in V$ (well-defined if no negative cycles)
Single-source shortest paths (6.006)

- Given directed graph \( G = (V, E) \), vertex \( s \in V \), edge weights \( w : E \rightarrow \mathbb{R} \)
- Find \( \delta(s, v) \) = shortest-path weight \( s \rightarrow v \) for all \( v \in V \) (well-defined if no negative cycles)

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted ((w = 1))</td>
<td>BFS</td>
<td>( O(V + E) )</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>Dijkstra</td>
<td>( O(E + V \log V) )</td>
</tr>
<tr>
<td>general</td>
<td>Bellman-Ford</td>
<td>( O(VE) )</td>
</tr>
<tr>
<td>acyclic graph (DAG)</td>
<td>topological sort</td>
<td>( O(V + E) ) + 1 pass Bellman-Ford</td>
</tr>
</tbody>
</table>
Single-source shortest paths (6.006)

- Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \to \mathbb{R}$
- Find $\delta(s, v) =$ shortest-path weight $s \to v$ for all $v \in V$ (well-defined if no negative cycles)

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted ($w = 1$)</td>
<td>BFS</td>
<td>$O(V + E)$</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td></td>
<td></td>
</tr>
<tr>
<td>general</td>
<td></td>
<td></td>
</tr>
<tr>
<td>acyclic graph (DAG)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \rightarrow \mathbb{R}$

Find $\delta(s, v)$ = shortest-path weight $s \rightarrow v$ for all $v \in V$ (well-defined if no negative cycles)

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted ($w = 1$)</td>
<td>BFS</td>
<td>$O(V + E)$</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>Dijkstra</td>
<td>$O(E + V \log V)$</td>
</tr>
<tr>
<td>general</td>
<td></td>
<td></td>
</tr>
<tr>
<td>acyclic graph (DAG)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Single-source shortest paths (6.006)

- Given directed graph \( G = (V, E) \), vertex \( s \in V \), edge weights \( w : E \to \mathbb{R} \)
- Find \( \delta(s, v) = \) shortest-path weight \( s \to v \) for all \( v \in V \) (well-defined if no negative cycles)

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted ((w = 1))</td>
<td>BFS</td>
<td>( O(V + E) )</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>Dijkstra</td>
<td>( O(E + V \log V) )</td>
</tr>
<tr>
<td>general</td>
<td>Bellman-Ford</td>
<td>( O(VE) )</td>
</tr>
<tr>
<td>acyclic graph ((DAG))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Single-source shortest paths (6.006)

- Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(s, v) =$ shortest-path weight $s \rightarrow v$ for all $v \in V$ (well-defined if no negative cycles)

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted ($w = 1$)</td>
<td>BFS</td>
<td>$O(V + E)$</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>Dijkstra</td>
<td>$O(E + V \log V)$</td>
</tr>
<tr>
<td>general</td>
<td>Bellman-Ford</td>
<td>$O(VE)$</td>
</tr>
<tr>
<td>acyclic graph (DAG)</td>
<td>topological sort</td>
<td>$O(V + E)$</td>
</tr>
<tr>
<td></td>
<td>+1 pass Bellman-Ford</td>
<td></td>
</tr>
</tbody>
</table>
Single-source shortest paths (6.006)

- Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(s, v)$ = shortest-path weight $s \rightarrow v$ for all $v \in V$ (well-defined if no negative cycles)

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted ($w = 1$)</td>
<td>BFS</td>
<td>$O(V + E)$</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>Dijkstra</td>
<td>$O(E + V \log V)$</td>
</tr>
<tr>
<td>general</td>
<td>Bellman-Ford</td>
<td>$O(VE)$</td>
</tr>
<tr>
<td>acyclic graph (DAG)</td>
<td>topological sort</td>
<td>$O(V + E)$</td>
</tr>
<tr>
<td></td>
<td>+1 pass Bellman-Ford</td>
<td></td>
</tr>
</tbody>
</table>
All-pairs shortest paths

- Given directed graph $G = (V, E)$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(u, v)$ for all $u, v \in V$

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted ($w = 1$)</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>general</td>
<td>$</td>
<td>V</td>
</tr>
</tbody>
</table>

Results 1, 2, 4 are (almost) best known. Do not know how to get polynomial speedup over $|V| \times$ Dijkstra, but [Williams'13] $n^{3/2} \Omega(p \log n / \log \log n)$. 

9 / 61
All-pairs shortest paths

- Given directed graph $G = (V, E)$, edge weights $w : E \to \mathbb{R}$
- Find $\delta(u, v)$ for all $u, v \in V$

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted ($w = 1$)</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>general</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>general</td>
<td>TODAY</td>
<td>$O(VE + V^2 \log V)$</td>
</tr>
</tbody>
</table>
All-pairs shortest paths

- Given directed graph $G = (V, E)$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(u, v)$ for all $u, v \in V$

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted (w = 1)</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(V^3)$</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>general</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>general</td>
<td>TODAY</td>
<td>$O(VE + V^2 \log V)$</td>
</tr>
</tbody>
</table>

Results 1, 2, 4 are (almost) best known. Do not know how to get polynomial speedup over $|V| \times$ Dijkstra, but [Williams’13]

$$n^3 / 2^{\Omega(\sqrt{\log n / \log \log n})}$$
All-pairs shortest paths

- Dynamic programming
- Matrix multiplication
- Floyd-Warshall algorithm
- Johnson’s algorithm
  - Difference constraints
Dynamic program #1

- Given directed graph $G = (V, E)$, edge weights $w : E \to \mathbb{R}$
- Find $\delta(u, v)$ for all $u, v \in V$

Subproblem: $d^{(m)}_{uv} =$ weight of shortest $u \to v$ path using $\leq m$ edges

Guess:
Dynamic program #1

- Given directed graph \( G = (V, E) \), edge weights \( w : E \rightarrow \mathbb{R} \)
- Find \( \delta(u, v) \) for all \( u, v \in V \)

Subproblem: \( d_{uv}^{(m)} \) = weight of shortest \( u \rightarrow v \) path using \( \leq m \) edges

Guess: what is the last edge \((x, v)\)?

\[
d_{uv}^{(m)} = \min_{x \in V} \left\{ d_{ux}^{(m-1)} + w(x, v) \right\}
\]
Dynamic program #1

- Given directed graph $G = (V, E)$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(u, v)$ for all $u, v \in V$

Subproblem: $d^{(m)}_{uv} =$ weight of shortest $u \rightarrow v$ path using $\leq m$ edges

Guess: what is the last edge $(x, v)$?

$$d^{(m)}_{uv} = \min_{x \in V} \left\{ d^{(m-1)}_{ux} + w(x, v) \right\}$$

Initialization:

$$d^{(0)}_{uv} =$$
Dynamic program #1

- Given directed graph $G = (V, E)$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(u, v)$ for all $u, v \in V$

Subproblem: $d_{uv}^{(m)}$ = weight of shortest $u \rightarrow v$ path using $\leq m$ edges

Guess: what is the last edge $(x, v)$?

$$d_{uv}^{(m)} = \min_{x \in V} \left\{ d_{ux}^{(m-1)} + w(x, v) \right\}$$

Initialization:

$$d_{uv}^{(0)} = \begin{cases} 0 & \text{if } u = v \\ \infty & \text{o.w.} \end{cases}$$

How large should $m$ be?

1. if no neg.-weight cycles, then

   $\delta(u, v) = d_{uv}^{(n-1)} = d_{uv}^{(n)} = \ldots$

2. neg. weight cycle iff $d_{vv}^{(n)} < 0$ for some $v \in V$
Dynamic program #1

- Given directed graph $G = (V, E)$, edge weights $w : E \to \mathbb{R}$
- Find $\delta(u, v)$ for all $u, v \in V$

Subproblem: $d_{uv}^{(m)}$ = weight of shortest $u \to v$ path using $\leq m$ edges

Guess: what is the last edge $(x, v)$?

$$d_{uv}^{(m)} = \min_{x \in V} \left\{ d_{ux}^{(m-1)} + w(x, v) \right\}$$

Initialization:

$$d_{uv}^{(0)} = \begin{cases} 0 & \text{if } u = v \\ \infty & \text{o.w.} \end{cases}$$

How large should $m$ be?

1. if no neg.-weight cycles, then $\delta(u, v) = d_{uv}^{(n-1)} = d_{uv}^{(n)} = \ldots$
2. neg. weight cycle iff $d_{vv}^{(n)} < 0$ for some $v \in V$
Dynamic program #1

Initialize:
\[ d_{uv}^{(0)} = \begin{cases} 
0 & \text{if } u = v \\
\infty & \text{o.w.} 
\end{cases} \]

Main loop:

For \( m \) from 1 to \( V \):
  For \( u \in V \):
    For \( v \in V \):
      For \( x \in V \):
        if \( d_{uv}^{(m)} > d_{ux}^{(m-1)} + w_{xv} \):
          \[ d_{uv}^{(m)} = d_{ux}^{(m-1)} + w_{xv} \]
Dynamic program #1

Runtime?
Dynamic program #1

Runtime?

$V^3$ subproblems, $O(V)$ guess + combination work per subproblem = $O(V^4)$
Dynamic program #1

Initialize:
\[ d_{uv}^{(0)} = \begin{cases} 
0 & \text{if } u = v \\
\infty & \text{o.w.} 
\end{cases} \]

Main loop:

For \( m \) from 1 to \( V \):
   For \( u \in V \):
      For \( v \in V \):
         For \( x \in V \):
            if \( d_{uv}^{(m)} > d_{ux}^{(m-1)} + w_{xv} \):
               \[ d_{uv}^{(m)} = d_{ux}^{(m-1)} + w_{xv} \]
Dynamic program #1

Initialize:

\[ d_{uv}^{(0)} = \begin{cases} 
0 & \text{if } u = v \\
\infty & \text{o.w.} 
\end{cases} \]

Main loop:

For \( m \) from 1 to \( V \):
  
  For \( u \in V \):
    
    For \( v \in V \):
      
      For \( x \in V \):
        
        if \( d_{uv}^{(m)} > d_{ux}^{(m-1)} + w_{xv} \):
          
          \[ d_{uv}^{(m)} = d_{ux}^{(m-1)} + w_{xv} \]

\[ d_{uv}^{(m)} \leftarrow \min_{x \in V} (d_{ux}^{(m-1)} + w_{xv}) \]
Dynamic program #1

Initialize:
\[ d_{uv}^{(0)} = \begin{cases} 
0 & \text{if } u = v \\
\infty & \text{o.w.} 
\end{cases} \]

Main loop:

For \( m \) from 1 to \( V \):
   For \( u \in V \):
      For \( v \in V \):
         \[ d_{uv}^{(m)} \leftarrow \min_{x \in V} (d_{ux}^{(m-1)} + w_{xv}) \]
All-pairs shortest paths

- Dynamic programming
- **Matrix multiplication**
- Floyd-Warshall algorithm
- Johnson’s algorithm
  - Difference constraints
Matrix multiplication

Given $n \times n$ matrices $A, B$, compute $C = A \cdot B$:

$$c_{uv} = \sum_{k=1}^{n} a_{uk} b_{kv}$$

- $O(n^3)$ via standard approach
- $O(n^{2.807})$ via Strassen's algorithm
- $O(n^{2.376})$ via Coppersmith-Winograd algorithm
Matrix multiplication

Given $n \times n$ matrices $A, B$, compute $C = A \cdot B$:

$$c_{uv} = \sum_{k=1}^{n} a_{uk} b_{kv}$$

- $O(n^3)$ via standard approach
- $O(n^{2.807})$ via Strassen’s algorithm
- $O(n^{2.376})$ via Coppersmith-Winograd algorithm

In shortest paths:

$$d_{uv} \leftarrow \min_{x \in V} (d_{ux} + w_{xv})$$
Matrix multiplication

Given $n \times n$ matrices $A, B$, compute $C = A \cdot B$:

$$c_{uv} = \sum_{k=1}^{n} a_{uk} b_{kv}$$

- $O(n^3)$ via standard approach
- $O(n^{2.807})$ via Strassen’s algorithm
- $O(n^{2.376})$ via Coppersmith-Winograd algorithm

In shortest paths:

$$d_{uv} \leftarrow \min_{x \in V} (d_{ux} + w_{xv})$$

What if $\oplus = \min$ and $\odot = +$?
Matrix multiplication

Given $n \times n$ matrices $A, B$, compute $C = A \odot B$. If $\oplus = \min$ and $\odot = +$, then

$$c_{uv} = \min_{k=1..n} (a_{uk} + b_{kv})$$

turns into

$$c_{uv} = (a_{u1} \odot b_{1v}) \oplus (a_{u2} \odot b_{2v}) \oplus \ldots \oplus (a_{un} \odot b_{nv})$$
Matrix multiplication

Given $n \times n$ matrices $A, B$, compute $C = A \odot B$. If $\oplus = \min$ and $\odot = +$, then

$$c_{uv} = \min_{k=1..n} (a_{uk} + b_{kv})$$

turns into

$$c_{uv} = (a_{u1} \odot b_{1v}) \oplus (a_{u2} \odot b_{2v}) \oplus \ldots \oplus (a_{un} \odot b_{nv})$$

Define $D^{(m)} = (d^{(m)}_{ij}), W = (w_{ij}), V = \{1, 2, \ldots, n\}$. Then

$$D^{(m)} = D^{(m-1)} \odot W = W^{\ominus}$$

Where

$$W^{\ominus} =$$
Matrix multiplication

Given $n \times n$ matrices $A, B$, compute $C = A \odot B$. If $\oplus = \min$ and $\odot = +$, then

$$c_{uv} = \min_{k=1..n} (a_{uk} + b_{kv})$$

turns into

$$c_{uv} = (a_{u1} \odot b_{1v}) \oplus (a_{u2} \odot b_{2v}) \oplus \ldots \oplus (a_{un} \odot b_{nv})$$

Define $D^{(m)} = (d^{(m)}_{ij}), W = (w_{ij}), V = \{1,2,\ldots, n\}$. Then

$$D^{(m)} = D^{(m-1)} \odot W = W^{\odot m}$$

Where

$$W^{\oplus} = \begin{pmatrix}
0 & \infty & \infty & \infty \\
\infty & 0 & \infty & \infty \\
\infty & \infty & 0 & \infty \\
\infty & \infty & \infty & 0
\end{pmatrix}$$
Matrix multiplication

Given \( n \times n \) matrices \( A, B \), compute \( C = A \odot B \). If \( \oplus = \text{min} \) and \( \odot = + \), then

\[
c_{uv} = \min_{k=1..n} (a_{uk} + b_{kv})
\]

turns into

\[
c_{uv} = (a_{u1} \odot b_{1v}) \oplus (a_{u2} \odot b_{2v}) \oplus \ldots \oplus (a_{un} \odot b_{nv})
\]

Define \( D^{(m)} = (d_{ij}^{(m)}) \), \( W = (w_{ij}) \), \( V = \{1, 2, \ldots, n\} \). Then

\[
D^{(m)} = D^{(m-1)} \odot W = W^{\odot m}
\]

Where

\[
W^{\ominus} = \begin{pmatrix}
0 & \infty & \infty & \infty & \infty \\
\infty & 0 & \infty & \infty & \infty \\
\infty & \infty & 0 & \infty & \infty \\
\infty & \infty & \infty & 0 & \infty \\
\infty & \infty & \infty & \infty & 0
\end{pmatrix}
\]

Makes sense because \((\mathbb{R}, \text{min}, +)\) is a semiring.
All-pairs shortest paths by matrix multiplication

\[ D^{(m)} = D^{(m-1)} \circ W = W^m \]

Runtime?
All-pairs shortest paths by matrix multiplication

\[ D^{(m)} = D^{(m-1)} \odot W = W^m \]

Runtime?

**Answer:** \( n \) multiplications, so \( \Theta(n^4) \) time.
All-pairs shortest paths by matrix multiplication

\[ D^{(m)} = D^{(m-1)} \odot W = W^m \]

Runtime?

**Answer:** \( n \) multiplications, so \( \Theta(n^4) \) time.

Can we do better?
All-pairs shortest paths by matrix multiplication

\[ D^{(m)} = D^{(m-1)} \odot W = W^{\circ m} \]

Runtime?

**Answer:** \( n \) multiplications, so \( \Theta(n^4) \) time.

Can we do better?

**Answer:** repeated squaring: \( O(n^3 \log n) \) time.
All-pairs shortest paths by matrix multiplication

\[ D^{(m)} = D^{(m-1)} \odot W = W^m \]

Runtime?

**Answer:** \( n \) multiplications, so \( \Theta(n^4) \) time.

Can we do better?

**Answer:** repeated squaring: \( O(n^3 \log n) \) time.
Transitive closure

Let

\[ t_{ij} = \begin{cases} 
1 & \text{if there is a path } i \to j \\
0 & \text{o.w.}
\end{cases} \]

(Want to check if \( \delta(u, v) < \infty \), special case of APSP)
Transitive closure

Let

\[ t_{ij} = \begin{cases} 
1 & \text{if there is a path } i \rightarrow j \\
0 & \text{o.w.}
\end{cases} \]

(Want to check if \( \delta(u, v) < \infty \), special case of APSP)
Transitive closure

Let

\[ t_{ij} = \begin{cases} 
1 & \text{if there is a path } i \rightarrow j \\
0 & \text{o.w.} 
\end{cases} \]

(Want to check if \( \delta(u,v) < \infty \), special case of APSP)

\(\{0, 1\}, \text{ OR, AND}\) is not a ring, but can still use fast matrix multiplication, so \(O(n^{2.376} \log n)\) time
All-pairs shortest paths

- Dynamic programming
- Matrix multiplication
- **Floyd-Warshall algorithm**
- Johnson’s algorithm
  - Difference constraints
Floyd-Warshall algorithm

A faster dynamic program.

Subproblem: $c_{uv}^{(k)} = \text{weight of shortest path } u \rightarrow v \text{ whose intermediate vertices are in } \{1,2,\ldots,k\}$

Guessing:
Floyd-Warshall algorithm

A faster dynamic program.

Subproblem: $c_{uv}^{(k)} = \text{weight of shortest path } u \rightarrow v \text{ whose intermediate vertices are in } \{1, 2, \ldots, k\}$

Guessing: does shortest path use vertex $k$?

$$c_{uv}^{(k)} = \min(c_{uv}^{(k-1)}, c_{uk}^{(k-1)} + c_{kv}^{(k-1)})$$

nodes in $\{1, 2, \ldots, k-1\}$
Floyd-Warshall algorithm

A faster dynamic program.

Subproblem: $c_{uv}^{(k)}$ = weight of shortest path $u \rightarrow v$ whose intermediate vertices are in $\{1, 2, \ldots, k\}$

Guessing: does shortest path use vertex $k$?

$$c_{uv}^{(k)} = \min(c_{uv}^{(k-1)}, c_{uk}^{(k-1)} + c_{kv}^{(k-1)})$$

Initialization: $c_{uv}^{(0)} = w_{uv}$

How do we read off the answer? Runtime?
Floyd-Warshall algorithm

A faster dynamic program.

**Subproblem:** $c_{uv}^{(k)} = \text{weight of shortest path } u \rightarrow v \text{ whose intermediate vertices are in } \{1, 2, \ldots, k\}$

**Guessing:** does shortest path use vertex $k$?

$$c_{uv}^{(k)} = \min(c_{uv}^{(k-1)}, c_{uk}^{(k-1)} + c_{kv}^{(k-1)})$$

**Initialization:** $c_{uv}^{(0)} = w_{uv}$

How do we read off the answer? Runtime?

Time: $O(V^3)$ problems, 2 choices, so $O(V^3)$
Floyd-Warshall algorithm

Initialize:
\[ C = (w_{uv}) \]

Main loop:
For \( k \) from 1 to \( n \):
  For \( u \in V \):
    For \( v \in V \):
      if \( c_{uv} > c_{uk} + c_{kv} \):
        \[ c_{uv} = c_{uk} + c_{kv} \]

Runtime: \( O(V^3) \)

Can we do better if graph is sparse, i.e. \( V \ll E \)?
All-pairs shortest paths

- Dynamic programming
- Matrix multiplication
- Floyd-Warshall algorithm
- Johnson’s algorithm
  - Difference constraints
Johnson’s algorithm

Better than Floyd-Warshall for sparse graphs: $O(VE + V^2 \log V)$ runtime
Single-source shortest paths (6.006)

- Given directed graph $G = (V, E)$, vertex $s \in V$, edge weights $w : E \rightarrow \mathbb{R}$
- Find $\delta(s, v) =$ shortest-path weight $s \rightarrow v$ for all $v \in V$ (well-defined if no negative cycles)

<table>
<thead>
<tr>
<th>Situation</th>
<th>Algorithm</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>unweighted ($w = 1$)</td>
<td>BFS</td>
<td>$O(V + E)$</td>
</tr>
<tr>
<td>nonneg. edge weights</td>
<td>Dijkstra</td>
<td>$O(E + V \log V)$</td>
</tr>
<tr>
<td>general</td>
<td>Bellman-Ford</td>
<td>$O(VE)$</td>
</tr>
<tr>
<td>acyclic graph (DAG)</td>
<td>topological sort</td>
<td>$O(V + E)$</td>
</tr>
<tr>
<td></td>
<td>+1 pass Bellman-Ford</td>
<td></td>
</tr>
</tbody>
</table>
Johnson’s algorithm

Better than Floyd-Warshall for sparse graphs: $O(VE + V^2 \log V)$ runtime

1. find function $h : V \to \mathbb{R}$ such that

$$w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0$$

for all $u, v \in V$ or determine that a negative cycle exists.
Johnson’s algorithm

Better than Floyd-Warshall for sparse graphs: $O(VE + V^2 \log V)$ runtime

1. find function $h: V \rightarrow \mathbb{R}$ such that

$$w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0$$

for all $u, v \in V$ or determine that a negative cycle exists.

2. Run Dijkstra on $(V, E, w_h)$ from every source $s \in V$, getting $\delta_h(u, v)$ for all $u, v \in V$. 

Johnson’s algorithm

Better than Floyd-Warshall for sparse graphs: $O(VE + V^2 \log V)$ runtime

1. find function $h: V \to \mathbb{R}$ such that

   $$w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0$$

   for all $u, v \in V$ or determine that a negative cycle exists.

2. Run Dijkstra on $(V, E, w_h)$ from every source $s \in V$, getting $\delta_h(u, v)$ for all $u, v \in V$.

3. Claim $\delta(u, v) = \delta_h(u, v) - h(u) + h(v)$
\[ \mathbf{w}(\mathbf{u}, \mathbf{v}) = \mathbf{w}(\mathbf{u}, \mathbf{v}) + h(\mathbf{u}) - h(\mathbf{v}) \geq 0 \]
\[ w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0 \]
$w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0$
\[ w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0 \]
\[ w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0 \]
\[ w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0 \]
Johnson’s algorithm

Suppose that for all $u, v \in V$

$$w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0$$

Claim $\delta(u, v) = \delta_h(u, v) - h(u) + h(v)$ for all $u, v \in V$

Proof.

Look at any $u \rightarrow v$ path $p$ in $G$. Suppose

$$p = u = v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_{k-1} \rightarrow v_k = v$$
Johnson’s algorithm

Suppose that for all \( u, v \in V \)

\[
    w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0
\]

Claim \( \delta(u, v) = \delta_h(u, v) - h(u) + h(v) \) for all \( u, v \in V \)

Proof.

Look at any \( u \rightarrow v \) path \( p \) in \( G \). Suppose

\[
p = u = v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_{k-1} \rightarrow v_k = v
\]

\[
w_h(p) = \sum_{i=1}^{k} w_h(v_{i-1}, v_i)
\]

\[
= \sum_{i=1}^{k} (w(v_{i-1}, v_i) + h_{i-1} - h_i)
\]

\[
= \sum_{i=1}^{k} w(v_{i-1}, v_i) + h_0 - h_k
\]

\[
= w(p) + h_0 - h_k
\]
How to find $h$?

Need to satisfy

$$w_h(u, v) = w(u, v) + h(u) - h(v) \geq 0 \text{ for all } u, v \in V.$$ 

This is a system of difference constraints

$$h(v) - h(u) \leq w(u, v) \text{ for all } u, v \in V.$$
Theorem
If \((V, E, w)\) has a negative weight cycle, then no solution to difference constraints

Proof.
Suppose \(v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_k \rightarrow v_0\) is negative weight and

\[
\begin{align*}
    h(v_1) - h(v_0) &\leq w(v_0, v_1) \\
    h(v_2) - h(v_1) &\leq w(v_2, v_1) \\
    &\vdots \\
    h(v_k) - h(v_{k-1}) &\leq w(v_k, v_{k-1}) \\
    h(v_0) - h(v_k) &\leq w(v_0, v_k)
\end{align*}
\]
Theorem

If \((V, E, w)\) has a negative weight cycle, then no solution to difference constraints

Proof.

Suppose \(v_0 \rightarrow v_1 \rightarrow \ldots \rightarrow v_k \rightarrow v_0\) is negative weight and

\[
\begin{align*}
    h(v_1) - h(v_0) & \leq w(v_0, v_1) \\
    h(v_2) - h(v_1) & \leq w(v_2, v_1) \\
    & \vdots \\
    h(v_k) - h(v_{k-1}) & \leq w(v_k, v_{k-1}) \\
    h(v_0) - h(v_k) & \leq w(v_0, v_k)
\end{align*}
\]

Add up equations:

\[
0 \leq w(v_0, v_1) + w(v_2, v_1) + \ldots + w(v_k, v_{k-1}) + w(v_0, v_k)
\]
Theorem
If $G = (V, E, w)$ has no negative weight cycles, then can solve system of difference constraints.
Theorem

If $G = (V, E, w)$ has no negative weight cycles, then can solve system of difference constraints.
Theorem

If $G = (V, E, w)$ has no neg. weight cycles, then can solve system of difference constraints
Theorem
If $G = (V, E, w)$ has no negative weight cycles, then can solve system of difference constraints
Theorem
If \( G = (V, E, w) \) has no negative weight cycles, then can solve system of difference constraints

Proof.
Add vertex \( s \) to \( G \), connect by 0 weight edges to every other node. Let \( h(v) = \delta(s, v) \).

\[
h(v) - h(u) = \delta(s, v) - \delta(s, u) \leq w(u, v)
\]

by triangle inequality. \( \square \)
Johnson’s algorithm – runtime

1. Bellman-Ford from $s$ \(O(VE)\)
2. reweight all edges \(O(E)\)
3. $|V| \times$ Dijkstra \(O(VE + V^2 \log V)\)
4. reweight all pairs \(O(V^2)\)
   \[O(VE + V^2 \log V)\]