Announcements

- HW5 due now
- HW6 due next Thursday
- Mid-Semester feedback on Campuswire!
Introduction

Setup:
- Directed graph $G = (V, E)$
- Length $\ell(x, y)$ on each edge $(x, y) \in E$
- Length of path $P$ is $\ell(P) = \sum_{(x,y) \in P} \ell(x, y)$
- $d(x, y) = \min_{x \to y \text{ paths } P} \ell(P)$

Last time: All distances from source node $v \in V$.
Today: Distances between all pairs of nodes!
Introduction

Setup:
- Directed graph $G = (V, E)$
- Length $\ell(x, y)$ on each edge $(x, y) \in E$
- Length of path $P$ is $\ell(P) = \sum_{(x, y) \in P} \ell(x, y)$
- $d(x, y) = \min_{x \to y \text{ paths } P} \ell(P)$

Last time: All distances from source node $v \in V$.
Today: Distances between all pairs of nodes!

Obvious solution:
Introduction

Setup:

- Directed graph $G = (V, E)$
- Length $\ell(x, y)$ on each edge $(x, y) \in E$
- Length of path $P$ is $\ell(P) = \sum_{(x, y) \in P} \ell(x, y)$
- $d(x, y) = \min_{x \rightarrow y \text{ paths}} \ell(P)$

Last time: All distances from source node $v \in V$.
Today: Distances between all pairs of nodes!

Obvious solution: single-source from each $v \in V$
Introduction

Setup:

- Directed graph $G = (V, E)$
- Length $\ell(x, y)$ on each edge $(x, y) \in E$
- Length of path $P$ is $\ell(P) = \sum_{(x, y) \in P} \ell(x, y)$
- $d(x, y) = \min_{x \rightarrow y} \text{paths } P \ell(P)$

Last time: All distances from source node $v \in V$.
Today: Distances between all pairs of nodes!

Obvious solution: single-source from each $v \in V$

- No negative weights: $n$ runs of Dijkstra, time $O(n(m + n \log n))$
- Negative weights: $n$ runs of Bellman-Ford, time $O(nmn) = O(mn^2)$
Introduction

Setup:
- Directed graph $G = (V, E)$
- Length $\ell(x, y)$ on each edge $(x, y) \in E$
- Length of path $P$ is $\ell(P) = \sum_{(x,y) \in P} \ell(x, y)$
- $d(x, y) = \min_{x \rightarrow y \text{ paths } P} \ell(P)$

Last time: All distances from source node $v \in V$.
Today: Distances between all pairs of nodes!

Obvious solution: single-source from each $v \in V$
- No negative weights: $n$ runs of Dijkstra, time $O(n(m + n \log n))$
- Negative weights: $n$ runs of Bellman-Ford, time $O(nmn) = O(mn^2)$

Can we do better? Particularly for negative edge weights?
- Main goal today: Negative weights as fast as possible.
Floyd-Warshall Algorithm
Floyd-Warshall: A Different Dynamic Programming Approach

To simplify notation, let $V = \{1, 2, \ldots, n\}$ and $\ell(i, j) = \infty$ if $(i, j) \notin E$

Bellman-Ford subproblems: length of shortest path with at most some number of edges
Floyd-Warshall: A Different Dynamic Programming Approach

To simplify notation, let $V = \{1, 2, \ldots, n\}$ and $\ell(i, j) = \infty$ if $(i, j) \notin E$

Bellman-Ford subproblems: length of shortest path with at most some number of edges

New subproblems:

- Intuition: “shortest path from $u$ to $v$ either goes through node $n$, or it doesn’t”
  - If it doesn't: shortest uses only first nodes in $\{1, 2, \ldots, n-1\}$.
  - If it does: consists of a path $P_1$ from $u$ to $n$ and a path $P_2$ from $n$ to $v$, neither of which uses $n$ (internally).
Floyd-Warshall: A Different Dynamic Programming Approach

To simplify notation, let $V = \{1, 2, \ldots, n\}$ and $\ell(i, j) = \infty$ if $(i, j) \notin E$

Bellman-Ford subproblems: length of shortest path with at most some number of edges

New subproblems:
- Intuition: “shortest path from $u$ to $v$ either goes through node $n$, or it doesn’t”
  - If it doesn’t: shortest uses only first nodes in $\{1, 2, \ldots, n-1\}$.
  - If it does: consists of a path $P_1$ from $u$ to $n$ and a path $P_2$ from $n$ to $v$, neither of which uses $n$ (internally).
- Subproblems: shortest path from $u$ to $v$ that only uses nodes in $\{1, 2, \ldots, k\}$ for all $u, v, k$. 
Formalizing Subproblems

\[ u \to v \text{ path } P: \text{ “intermediate nodes” are all nodes in } P \text{ other than } u, v. \]

\[ d^k_{ij}: \text{ distance from } i \text{ to } j \text{ using only } i \to j \text{ paths with intermediate vertices in } \{1, 2, \ldots, k\}. \]

- Goal: compute \( d^k_{ij} \) for all \( i, j, k \in [n] \).
- Return \( d^n_{ij} \) for all \( i, j \in V \).
Formalizing Subproblems

\( u \rightarrow v \) path \( P \): “intermediate nodes” are all nodes in \( P \) other than \( u, v \).

\( d_{ij}^k \): distance from \( i \) to \( j \) using only \( i \rightarrow j \) paths with intermediate vertices in \( \{1, 2, \ldots, k\} \).

\begin{itemize}
  \item Goal: compute \( d_{ij}^k \) for all \( i, j, k \in [n] \).
  \item Return \( d_{ij}^n \) for all \( i, j \in V \).
\end{itemize}

\[
\begin{align*}
  d_{ij}^k = & \\
  & \begin{cases} 
    \ell(i, j) & \text{if } k = 0 \\
    \min( & \\
    d_{ik}^{k-1}+d_{kj}^{k-1} & \text{if } k \geq 1
  \end{cases}
\end{align*}
\]
Formalizing Subproblems

**u → v path P**: “intermediate nodes” are all nodes in P other than u,v.

\(d_{ij}^k\): distance from i to j using only i → j paths with intermediate vertices in \(\{1, 2, \ldots, k\}\).

- **Goal**: compute \(d_{ij}^k\) for all \(i, j, k \in [n]\).
- **Return**: \(d_{ij}^n\) for all \(i, j \in V\).

\[
d_{ij}^k = \begin{cases} 
\ell(i, j) & \text{if } k = 0 \\
\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) & \text{if } k \geq 1 
\end{cases}
\]
Formalizing Subproblems

\( u \rightarrow v \) path \( P \): “intermediate nodes” are all nodes in \( P \) other than \( u, v \).

\( d_{ij}^{k} \): distance from \( i \) to \( j \) using only \( i \rightarrow j \) paths with intermediate vertices in \( \{1, 2, \ldots, k\} \).

- Goal: compute \( d_{ij}^{k} \) for all \( i, j, k \in [n] \).
- Return \( d_{ij}^{n} \) for all \( i, j \in V \).

\[
d_{ij}^{k} = \begin{cases} 
\ell(i, j) & \text{if } k = 0 \\
\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) & \text{if } k \geq 1 
\end{cases}
\]
Optimal Substructure

Theorem

For all $i, j, k \in [n]$:

$$d^k_{ij} = \begin{cases} 
\ell(i, j) & \text{if } k = 0 \\
\min(d^k_{ij}, d^{k-1}_{ik} + d^{k-1}_{kj}) & \text{if } k \geq 1
\end{cases}$$
Optimal Substructure

Theorem

For all $i, j, k \in [n]$:

\[
d_{ij}^k = \begin{cases} 
\ell(i, j) & \text{if } k = 0 \\
\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) & \text{if } k \geq 1 
\end{cases}
\]

If $k = 0$: ✓
Optimal Substructure

Theorem

For all \( i, j, k \in [n] \):

\[
 d_{ij}^k = \begin{cases} 
 \ell(i, j) & \text{if } k = 0 \\
 \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) & \text{if } k \geq 1 
\end{cases}
\]

If \( k = 0 \): √

If \( k \geq 1 \): prove ≤ and ≥
Optimal Substructure

Theorem

For all $i, j, k \in [n]$:

$$d_{ij}^k = \begin{cases} 
\ell(i, j) & \text{if } k = 0 \\
\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) & \text{if } k \geq 1 
\end{cases}$$

If $k = 0$: ✓

If $k \geq 1$: prove $\leq$ and $\geq$

$\leq$: 

Michael Dinitz
Lecture 15: APSP
October 19, 2021 7 / 13
Optimal Substructure

**Theorem**

For all $i, j, k \in [n]$:  

$$d^k_{ij} = \begin{cases} \ell(i, j) & \text{if } k = 0 \\ \min(d^k_{ij}, d^k_{ik} + d^k_{kj}) & \text{if } k \geq 1 \end{cases}$$

If $k = 0$: $\surd$

If $k \geq 1$: prove $\leq$ and $\geq$

$\leq$: Two feasible solutions
Optimal Substructure

Theorem

For all \( i, j, k \in [n] \):

\[
\begin{align*}
d_{ij}^k &= \begin{cases} 
\ell(i, j) & \text{if } k = 0 \\
\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) & \text{if } k \geq 1 
\end{cases}
\end{align*}
\]

If \( k = 0 \): \( \checkmark \)

If \( k \geq 1 \): prove \( \leq \) and \( \geq \)

\( \leq \): Two feasible solutions

\( \geq \): Let \( P \) be shortest \( i \rightarrow j \) path with all intermediate nodes in \([k]\)

- If \( k \) not an intermediate node of \( P \):
Optimal Substructure

**Theorem**

For all $i, j, k \in [n]$:

$$d_{ij}^k = \begin{cases} 
\ell(i, j) & \text{if } k = 0 \\
\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) & \text{if } k \geq 1 
\end{cases}$$

If $k = 0$: √

If $k \geq 1$: prove $\leq$ and $\geq$

$\leq$: Two feasible solutions

$\geq$: Let $P$ be shortest $i \rightarrow j$ path with all intermediate nodes in $[k]$

- If $k$ not an intermediate node of $P$: $P$ has all intermediate nodes in $[k - 1] \implies 
  \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) \leq d_{ij}^{k-1} \leq \ell(P) = d_{ij}^k$
Optimal Substructure

**Theorem**

For all \( i, j, k \in [n] \):

\[
d_{ij}^k = \begin{cases} 
\ell(i, j) & \text{if } k = 0 \\
\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) & \text{if } k \geq 1 
\end{cases}
\]

If \( k = 0 \): \( \checkmark \)

If \( k \geq 1 \): prove \( \leq \) and \( \geq \)

\( \leq \): Two feasible solutions

\( \geq \): Let \( P \) be shortest \( i \rightarrow j \) path with all intermediate nodes in \( [k] \)

- If \( k \) not an intermediate node of \( P \): \( P \) has all intermediate nodes in \( [k-1] \) \( \implies \)
  \[
  \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) \leq d_{ij}^{k-1} \leq \ell(P) = d_{ij}^{k}
  \]
- If \( k \) is an intermediate node of \( P \):
Optimal Substructure

**Theorem**

For all \(i, j, k \in [n]\):

\[
d^k_{ij} = \begin{cases} 
\ell(i, j) & \text{if } k = 0 \\
\min(d^{k-1}_{ij}, d^{k-1}_{ik} + d^{k-1}_{kj}) & \text{if } k \geq 1 
\end{cases}
\]

If \(k = 0\): \(\checkmark\)

If \(k \geq 1\): prove \(\leq\) and \(\geq\)

\(\leq\): Two feasible solutions

\(\geq\): Let \(P\) be shortest \(i \rightarrow j\) path with all intermediate nodes in \([k]\)

- If \(k\) not an intermediate node of \(P\): \(P\) has all intermediate nodes in \([k - 1]\) \(\implies\)
  \[
  \min(d^{k-1}_{ij}, d^{k-1}_{ik} + d^{k-1}_{kj}) \leq d^{k-1}_{ij} \leq \ell(P) = d^k_{ij}
  \]

- If \(k\) is an intermediate node of \(P\): divide \(P\) into \(P_1\) (subpath from \(i\) to \(k\)) and \(P_2\) (subpath from \(k\) to \(j\))

  \[
  \min(d^{k-1}_{ij}, d^{k-1}_{ik} + d^{k-1}_{kj}) \leq d^{k-1}_{ik} + d^{k-1}_{kj} \leq \ell(P_1) + \ell(P_2) = \ell(P) = d^k_{ij}
  \]
Floyd-Warshall Algorithm

Usually bottom-up, since so simple:

\[
M[i, j, 0] = \ell(i, j) \text{ for all } i, j \in [n]
\]

\[
\text{for}(k = 1 \text{ to } n)
\]

\[
\text{for}(i = 1 \text{ to } n)
\]

\[
\text{for}(j = 1 \text{ to } n)
\]

\[
M[i, j, k] = \min(M[i, j, k - 1], M[i, k, k - 1] + M[k, j, k - 1])
\]
Floyd-Warshall Algorithm

Usually bottom-up, since so simple:

\[
M[i, j, 0] = \ell(i, j) \text{ for all } i, j \in [n]
\]

\[
\text{for}(k = 1 \text{ to } n)
\]

\[
\text{for}(i = 1 \text{ to } n)
\]

\[
\text{for}(j = 1 \text{ to } n)
\]

\[
M[i, j, k] = \min(M[i, j, k - 1], M[i, k, k - 1] + M[k, j, k - 1])
\]

Correctness: obvious for \(k = 0\). For \(k \geq 1\):

\[
M[i, j, k] = \min(M[i, j, k - 1], M[i, k, k - 1] + M[k, j, k - 1])
\]

\[
= \min(d^{k-1}_{ij}, d^{k-1}_{ik} + d^{k-1}_{kj})
\]

\[
= d^{k}_{ij}
\]
Floyd-Warshall Algorithm

Usually bottom-up, since so simple:

\[
M[i, j, 0] = \ell(i, j) \text{ for all } i, j \in [n]
\]

for \(k = 1\) to \(n\)

\[
\text{for}(i = 1 \text{ to } n)
\]

\[
\text{for}(j = 1 \text{ to } n)
\]

\[
M[i, j, k] = \min(M[i, j, k - 1], M[i, k, k - 1] + M[k, j, k - 1])
\]

**Correctness:** obvious for \(k = 0\). For \(k \geq 1\):

\[
M[i, j, k] = \min(M[i, j, k - 1], M[i, k, k - 1] + M[k, j, k - 1])
\]

\[
= \min(d_{ik}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})
\]

\[
= d_{ij}^k
\]

**Running Time:** \(O(n^3)\)
Johnson’s Algorithm
Reweighting

Different Approach: Can we “fix” negative weights so Dijkstra from every node works?

- Time would be $O(n(m + n \log n)) = O(mn + n^2 \log n)$, better than Floyd-Warshall.

First attempt: Let $-\alpha$ be smallest length (most negative). Add $\alpha$ to every edge.

Does this work? No!

New length of path $P$ is $\ell(P) + \alpha$, so original shortest path might no longer be shortest path if it has many edges.

Some other kind of reweighting? Need new lengths $\hat{\ell}$ such that:

- Path $P$ a shortest path under lengths $\ell$ if and only if $P$ as shortest path under lengths $\hat{\ell}$.
- $\hat{\ell}(u, v) \geq 0$ for all $(u, v) \in E$.
Reweighting

Different Approach: Can we “fix” negative weights so Dijkstra from every node works?

- Time would be $O(n(m + n \log n)) = O(mn + n^2 \log n)$, better than Floyd-Warshall

First attempt: Let $-\alpha$ be smallest length (most negative). Add $\alpha$ to every edge.

- Does this work?
Reweighting

Different Approach: Can we “fix” negative weights so Dijkstra from every node works?

- Time would be $O(n(m + n \log n)) = O(mn + n^2 \log n)$, better than Floyd-Warshall

First attempt: Let $-\alpha$ be smallest length (most negative). Add $\alpha$ to every edge.

- Does this work? No!

Some other kind of reweighting? Need new lengths $\hat{\alpha}$ such that:

- Path $P$ a shortest path under lengths $\alpha$ if and only if $P$ as shortest path under lengths $\hat{\alpha}$

- $\hat{\alpha}(u, v) \geq 0$ for all $(u, v) \in E$
Reweighting

Different Approach: Can we “fix” negative weights so Dijkstra from every node works?

- Time would be $O(n(m + n \log n)) = O(mn + n^2 \log n)$, better than Floyd-Warshall

First attempt: Let $-\alpha$ be smallest length (most negative). Add $\alpha$ to every edge.

- Does this work? No!

- New length of path $P$ is $\ell(P) + \alpha |P|$, so original shortest path might no longer be shortest path if it has many edges.
Reweighting

Different Approach: Can we “fix” negative weights so Dijkstra from every node works?
- Time would be $O(n(m + n \log n)) = O(mn + n^2 \log n)$, better than Floyd-Warshall

First attempt: Let $-\alpha$ be smallest length (most negative). Add $\alpha$ to every edge.
- Does this work? No!
  - New length of path $P$ is $\ell(P) + \alpha|P|$, so original shortest path might no longer be shortest path if it has many edges.

Some other kind of reweighting? Need new lengths $\hat{\ell}$ such that:

- Path $P$ a shortest path under lengths $\ell$ if and only if $\hat{\ell}(u, v) \geq 0$ for all $(u, v) \in E$
Reweighting

Different Approach: Can we “fix” negative weights so Dijkstra from every node works?

- Time would be $O(n(m + n \log n)) = O(mn + n^2 \log n)$, better than Floyd-Warshall

First attempt: Let $-\alpha$ be smallest length (most negative). Add $\alpha$ to every edge.

- Does this work? No!

- New length of path $P$ is $\ell(P) + \alpha|P|$, so original shortest path might no longer be shortest path if it has many edges.

Some other kind of reweighting? Need new lengths $\hat{\ell}$ such that:

- Path $P$ a shortest path under lengths $\ell$ if and only $P$ a shortest path under lengths $\hat{\ell}$
- $\hat{\ell}(u, v) \geq 0$ for all $(u, v) \in E$
Vertex Reweighting

Neat observation: put weights at vertices!

- Let $h : V \rightarrow \mathbb{R}$ be node weights.
- Let $\ell_h(u, v) = \ell(u, v) + h(u) - h(v)$
Vertex Reweighting

Neat observation: put weights at vertices!

- Let $h : V \to \mathbb{R}$ be node weights.
- Let $\ell_h(u, v) = \ell(u, v) + h(u) - h(v)$

Let $P = (v_0, v_1, \ldots, v_k)$ be arbitrary (not necessarily shortest) path.

$\ell_h(P) = \sum_{i=0}^{k-1} \ell_h(v_i, v_{i+1})$ (telescoping)

$\ell_h(v_0) - h(v_k) + \sum_{i=0}^{k-1} (\ell(v_i, v_{i+1}) + h(v_i) - h(v_{i+1}))$ added to every $v_0 \to v_k$ path, so shortest path from $v_0$ to $v_k$ still shortest path!
### Vertex Reweighting

Neat observation: put weights at vertices!

- Let \( h : V \rightarrow \mathbb{R} \) be node weights.
- Let \( \ell_h(u,v) = \ell(u,v) + h(u) - h(v) \)

Let \( P = (v_0, v_1, \ldots, v_k) \) be arbitrary (not necessarily shortest) path.

\[
\ell_h(P) = \sum_{i=0}^{k-1} \ell_h(v_i, v_{i+1}) = \sum_{i=0}^{k-1} (\ell(v_i, v_{i+1}) + h(v_i) - h(v_{i+1}))
\]

added to every \( v_0 \rightarrow v_k \) path, so shortest path from \( v_0 \) to \( v_k \) still shortest path!
Vertex Reweighting

Neat observation: put weights at vertices!

- Let $h : V \rightarrow \mathbb{R}$ be node weights.
- Let $\ell_h(u, v) = \ell(u, v) + h(u) - h(v)$

Let $P = (v_0, v_1, \ldots, v_k)$ be arbitrary (not necessarily shortest) path.

$$\ell_h(P) = \sum_{i=0}^{k-1} \ell_h(v_i, v_{i+1}) = \sum_{i=0}^{k-1} (\ell(v_i, v_{i+1}) + h(v_i) - h(v_{i+1}))$$

$$= h(v_0) - h(v_k) + \sum_{i=0}^{k-1} \ell(v_i, v_{i+1})$$

(telescoping)
Vertex Reweighting

Neat observation: put weights at *vertices*!

- Let $h : V \rightarrow \mathbb{R}$ be node weights.
- Let $\ell_h(u, v) = \ell(u, v) + h(u) - h(v)$

Let $P = (v_0, v_1, \ldots, v_k)$ be arbitrary (not necessarily shortest) path.

\[
\ell_h(P) = \sum_{i=0}^{k-1} \ell_h(v_i, v_{i+1}) = \sum_{i=0}^{k-1} (\ell(v_i, v_{i+1}) + h(v_i) - h(v_{i+1}))
\]

\[
= h(v_0) - h(v_k) + \sum_{i=0}^{k-1} \ell(v_i, v_{i+1}) \quad \text{(telescoping)}
\]

\[
= \ell(P) + h(v_0) - h(v_k)
\]
Vertex Reweighting

Neat observation: put weights at *vertices*!

- Let $h: V \to \mathbb{R}$ be node weights.
- Let $\ell_h(u, v) = \ell(u, v) + h(u) - h(v)$

Let $P = (v_0, v_1, \ldots, v_k)$ be arbitrary (not necessarily shortest) path.

$$\ell_h(P) = \sum_{i=0}^{k-1} \ell_h(v_i, v_{i+1}) = \sum_{i=0}^{k-1} (\ell(v_i, v_{i+1}) + h(v_i) - h(v_{i+1}))$$

$$= h(v_0) - h(v_k) + \sum_{i=0}^{k-1} \ell(v_i, v_{i+1}) \quad \text{(telescoping)}$$

$$= \ell(P) + h(v_0) - h(v_k)$$

$h(v_0) - h(v_k)$ added to *every* $v_0 \to v_k$ path, so shortest path from $v_0$ to $v_k$ still shortest path!
Making lengths nonnegative

So vertex reweighting preserves shortest paths. Find weights to make lengths nonnegative?

Add *new node* $s$ to graph, edges $(s, v)$ for all $v \in V$ of length 0

Note $h(u) \leq 0$ for all $u \in V$

Want to show that $\ell(h(u), v) \geq 0$ for all edges $(u, v)$.

Triangle inequality:

$h(v) = d(s, v) \leq d(s, u) + \ell(u, v) = h(u) + \ell(u, v)$

$\ell(h(u), v) = \ell(u, v) + h(u) - h(v) \geq \ell(u, v) + h(u) - (h(u) + \ell(u, v)) = 0$
Making lengths nonnegative

So vertex reweighting preserves shortest paths. Find weights to make lengths nonnegative?

Add *new node* $s$ to graph, edges $(s, v)$ for all $v \in V$ of length 0

- Run Bellman-Ford from $s$, then for all $u \in V$ set $h(u)$ to be $d(s, u)$
- Note $h(u) \leq 0$ for all $u \in V$
Making lengths nonnegative

So vertex reweighting preserves shortest paths. Find weights to make lengths nonnegative?

Add *new node* $s$ to graph, edges $(s,v)$ for all $v \in V$ of length 0

- Run Bellman-Ford from $s$, then for all $u \in V$ set $h(u)$ to be $d(s,u)$
- Note $h(u) \leq 0$ for all $u \in V$

Want to show that $\ell_h(u,v) \geq 0$ for all edges $(u,v)$.

- Triangle inequality: $h(v) = d(s,v) \leq d(s,u) + \ell(u,v) = h(u) + \ell(u,v)$
Making lengths nonnegative

So vertex reweighting preserves shortest paths. Find weights to make lengths nonnegative?

Add new node $s$ to graph, edges $(s, v)$ for all $v \in V$ of length 0

- Run Bellman-Ford from $s$, then for all $u \in V$ set $h(u)$ to be $d(s, u)$
- Note $h(u) \leq 0$ for all $u \in V$

Want to show that $\ell_h(u, v) \geq 0$ for all edges $(u, v)$.

- Triangle inequality: $h(v) = d(s, v) \leq d(s, u) + \ell(u, v) = h(u) + \ell(u, v)$

\[
\ell_h(u, v) = \ell(u, v) + h(u) - h(v) \geq \ell(u, v) + h(u) - (h(u) + \ell(u, v)) = 0
\]
Johnson’s Algorithm

- Add vertex $s$ to graph, edge $(s, u)$ for all $u \in V$ with $\ell(s, u) = 0$
- Run Bellman-Ford from $s$, set $h(u) = d(s, u)$
- Remove $s$, run Dijkstra from every node $u \in V$ to get $d_h(u, v)$ for all $u, v \in V$
- If want distances, set $d(u, v) = d_h(u, v) - h(u) + h(v)$ for all $u, v \in V$

**Correctness:** From previous discussion.
Johnson’s Algorithm

- Add vertex $s$ to graph, edge $(s, u)$ for all $u \in V$ with $\ell(s, u) = 0$
- Run Bellman-Ford from $s$, set $h(u) = d(s, u)$
- Remove $s$, run Dijkstra from every node $u \in V$ to get $d_h(u, v)$ for all $u, v \in V$
- If want distances, set $d(u, v) = d_h(u, v) - h(u) + h(v)$ for all $u, v \in V$

**Correctness:** From previous discussion.

**Running Time:**
Johnson’s Algorithm

- Add vertex \( s \) to graph, edge \((s, u)\) for all \( u \in V \) with \( \ell(s, u) = 0 \)
- Run Bellman-Ford from \( s \), set \( h(u) = d(s, u) \)
- Remove \( s \), run Dijkstra from every node \( u \in V \) to get \( d_h(u, v) \) for all \( u, v \in V \)
- If want distances, set \( d(u, v) = d_h(u, v) - h(u) + h(v) \) for all \( u, v \in V \)

**Correctness:** From previous discussion.

**Running Time:** \( O(n) + O(mn) + O(n(m + n \log n)) = O(mn + n^2 \log n) \)