# Lecture 15: All-Pairs Shortest Paths

Michael Dinitz

October 19, 2021
601.433/633 Introduction to Algorithms

# Announcements

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

# Introduction

Setup:

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

Last time: All distances from source node $\mathbf{v} \in \mathbf{V}$.

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 $\mathbf{G} = (\mathbf{V}, \mathbf{E})$
- Length $\ell(\mathbf{x}, \mathbf{y})$ on each edge $(\mathbf{x}, \mathbf{y}) \in \mathbf{E}$
- Length of path $\mathbf{P}$ is $\ell(\mathbf{P}) = \sum_{(\mathbf{x},\mathbf{y}) \in \mathbf{P}} \ell(\mathbf{x}, \mathbf{y})$
- $\mathbf{d}(\mathbf{x}, \mathbf{y}) = \mathbf{min}_{\mathbf{x} \to \mathbf{y} \text{ paths } \mathbf{P}} \ell(\mathbf{P})$

Last time: All distances from source node $\mathbf{v} \in \mathbf{V}$.

Today: Distances between all pairs of nodes!

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

# Introduction

Setup:

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

Last time: All distances from source node $\mathbf{v} \in \mathbf{V}$.

Today: Distances between all pairs of nodes!

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

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

# Introduction

Setup:

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

Last time: All distances from source node $\mathbf{v} \in \mathbf{V}$.

Today: Distances between all pairs of nodes!

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

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

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

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

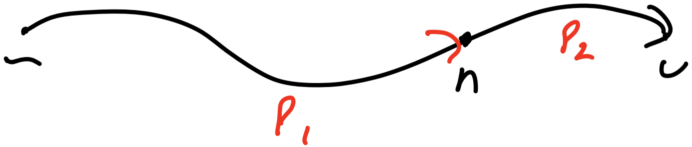
# Floyd-Warshall: A Different Dynamic Programming Approach

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

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 $\mathbf{P_1}$ from **u** to **n** and a path $\mathbf{P_2}$ from **n** to **v**, neither of which uses **n** (internally).



# Floyd-Warshall: A Different Dynamic Programming Approach

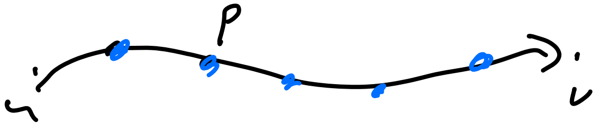
To simplify notation, let $V = \{1, 2, \ldots, n\}$ and $\ell(i, j) = \infty$ if $(i, j) \notin E$

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




$\mathbf{u} \to \mathbf{v}$ path $\mathbf{P}$: "intermediate nodes" are all nodes in $\mathbf{P}$ other than $\mathbf{u}, \mathbf{v}$.

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

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

# Formalizing Subproblems

$\mathbf{u} \to \mathbf{v}$ path $\mathbf{P}$: "intermediate nodes" are all nodes in $\mathbf{P}$ other than $\mathbf{u}, \mathbf{v}$.

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

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

$$\mathbf{d_{ij}^k} = \begin{cases} & \text{if } \mathbf{k} = \mathbf{0} \\ & \text{if } \mathbf{k} \geq \mathbf{1} \end{cases}$$

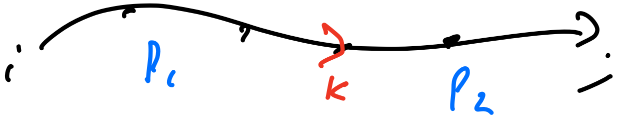
# Formalizing Subproblems

$\mathbf{u} \rightarrow \mathbf{v}$ path $\mathbf{P}$: "intermediate nodes" are all nodes in $\mathbf{P}$ other than $\mathbf{u}, \mathbf{v}$.

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

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


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




if k not used: $d_{ij}^{k-1}$

k is used: $d_{ik}^{k-1} + d_{kj}^{k-1}$

# Formalizing Subproblems

$u \to v$ path $P$: "intermediate nodes" are all nodes in $P$ other than $u, v$.

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

- 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* $\mathbf{i}, \mathbf{j}, \mathbf{k} \in [\mathbf{n}]$:

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

# Optimal Substructure

**Theorem**

*For all* $\mathbf{i}, \mathbf{j}, \mathbf{k} \in [\mathbf{n}]$*:*

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

If $\mathbf{k} = \mathbf{0}$: ✓

# Optimal Substructure

**Theorem**

*For all* $\mathbf{i}, \mathbf{j}, \mathbf{k} \in [\mathbf{n}]$*:*

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

If $\mathbf{k} = \mathbf{0}$: ✓

If $\mathbf{k} \geq \mathbf{1}$: prove $\leq$ and $\geq$

# Optimal Substructure

## Theorem

*For all* $\mathbf{i}, \mathbf{j}, \mathbf{k} \in [\mathbf{n}]$*:*

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

If $\mathbf{k} = \mathbf{0}$: ✓

If $\mathbf{k} \geq \mathbf{1}$: prove $\leq$ and $\geq$

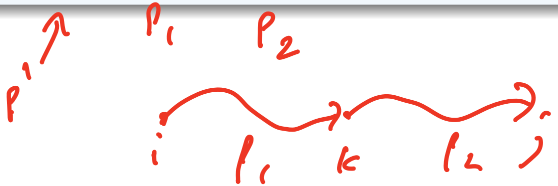
$\leq$:

# Optimal Substructure

**Theorem**

*For all* $\mathbf{i}, \mathbf{j}, \mathbf{k} \in [\mathbf{n}]$*:*

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



If $\mathbf{k} = \mathbf{0}$: ✓

If $\mathbf{k} \geq \mathbf{1}$: prove $\leq$ and $\geq$

$\leq$: Two feasible solutions

# Optimal Substructure

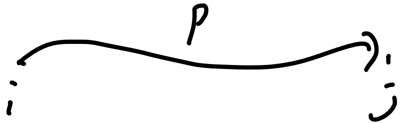
## Theorem

*For all* $\mathbf{i, j, k} \in [\mathbf{n}]$*:*

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

If $\mathbf{k} = \mathbf{0}$: ✓

If $\mathbf{k} \geq \mathbf{1}$: prove $\leq$ and $\geq$




$\leq$: Two feasible solutions

$\geq$: Let $\mathbf{P}$ be shortest $\mathbf{i} \to \mathbf{j}$ path with all intermediate nodes in $[\mathbf{k}]$

- If $\mathbf{k}$ not an intermediate node of $\mathbf{P}$:

# Optimal Substructure

**Theorem**

*For all* $\mathbf{i}, \mathbf{j}, \mathbf{k} \in [\mathbf{n}]$*:*

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

If $\mathbf{k} = \mathbf{0}$: ✓

If $\mathbf{k} \geq \mathbf{1}$: prove $\leq$ and $\geq$

$\leq$: Two feasible solutions

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

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

# Optimal Substructure

**Theorem**

*For all* $\mathbf{i}, \mathbf{j}, \mathbf{k} \in [\mathbf{n}]$*:*

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

If $\mathbf{k} = \mathbf{0}$: ✓

If $\mathbf{k} \geq \mathbf{1}$: prove $\leq$ and $\geq$

$\leq$: Two feasible solutions

$\geq$: Let $\mathbf{P}$ be shortest $\mathbf{i} \to \mathbf{j}$ path with all intermediate nodes in $[\mathbf{k}]$

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

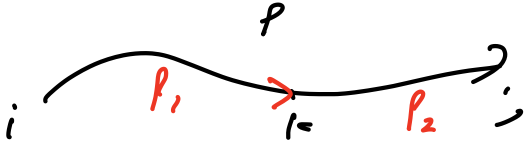
# Optimal Substructure

## Theorem

*For all* $\mathbf{i}, \mathbf{j}, \mathbf{k} \in [\mathbf{n}]$*:*

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

If $\mathbf{k} = \mathbf{0}$: ✓




If $\mathbf{k} \geq \mathbf{1}$: prove $\leq$ and $\geq$

$\leq$: Two feasible solutions

$\geq$: Let $\mathbf{P}$ be shortest $\mathbf{i} \to \mathbf{j}$ path with all intermediate nodes in $[\mathbf{k}]$

- If $\mathbf{k}$ not an intermediate node of $\mathbf{P}$: $\mathbf{P}$ has all intermediate nodes in $[\mathbf{k}-\mathbf{1}]$ $\Longrightarrow$ $\min(\mathbf{d}_{ij}^{k-1}, \mathbf{d}_{ik}^{k-1} + \mathbf{d}_{kj}^{k-1}) \leq \mathbf{d}_{ij}^{k-1} \leq \ell(\mathbf{P}) = \mathbf{d}_{ij}^{k}$
- If $\mathbf{k}$ is an intermediate node of $\mathbf{P}$: divide $\mathbf{P}$ into $\mathbf{P}_1$ (subpath from $\mathbf{i}$ to $\mathbf{k}$) and $\mathbf{P}_2$ (subpath from $\mathbf{k}$ to $\mathbf{j}$)

$$\min(\mathbf{d}_{ij}^{k-1}, \mathbf{d}_{ik}^{k-1} + \mathbf{d}_{kj}^{k-1}) \leq \mathbf{d}_{ik}^{k-1} + \mathbf{d}_{kj}^{k-1} \leq \ell(\mathbf{P}_1) + \ell(\mathbf{P}_2) = \ell(\mathbf{P}) = \mathbf{d}_{ij}^{k}$$

# Floyd-Warshall Algorithm

Usually bottom-up, since so simple:

$\mathbf{M[i,j,0]} = \ell(\mathbf{i,j})$ for all $\mathbf{i,j} \in [\mathbf{n}]$
for($\mathbf{k} = \mathbf{1}$ to $\mathbf{n}$)
    for($\mathbf{i} = \mathbf{1}$ to $\mathbf{n}$)
        for($\mathbf{j} = \mathbf{1}$ to $\mathbf{n}$)
            $\mathbf{M[i,j,k]} = \mathbf{min(M[i,j,k-1], M[i,k,k-1] + M[k,j,k-1])}$

# Floyd-Warshall Algorithm

Usually bottom-up, since so simple:

$\mathbf{M}[\mathbf{i},\mathbf{j},\mathbf{0}] = \ell(\mathbf{i},\mathbf{j})$ for all $\mathbf{i},\mathbf{j} \in [\mathbf{n}]$
for($\mathbf{k} = \mathbf{1}$ to $\mathbf{n}$)
    for($\mathbf{i} = \mathbf{1}$ to $\mathbf{n}$)
        for($\mathbf{j} = \mathbf{1}$ to $\mathbf{n}$)
            $\mathbf{M}[\mathbf{i},\mathbf{j},\mathbf{k}] = \mathbf{min}(\mathbf{M}[\mathbf{i},\mathbf{j},\mathbf{k}-\mathbf{1}], \mathbf{M}[\mathbf{i},\mathbf{k},\mathbf{k}-\mathbf{1}] + \mathbf{M}[\mathbf{k},\mathbf{j},\mathbf{k}-\mathbf{1}])$

**Correctness:** obvious for $\mathbf{k} = \mathbf{0}$. For $\mathbf{k} \geq \mathbf{1}$:

$$\begin{aligned}
\mathbf{M}[\mathbf{i},\mathbf{j},\mathbf{k}] &= \mathbf{min}(\mathbf{M}[\mathbf{i},\mathbf{j},\mathbf{k}-\mathbf{1}], \mathbf{M}[\mathbf{i},\mathbf{k},\mathbf{k}-\mathbf{1}] + \mathbf{M}[\mathbf{k},\mathbf{j},\mathbf{k}-\mathbf{1}]) && \text{(def of algorithm)} \\
&= \mathbf{min}(\mathbf{d}_{\mathbf{ij}}^{\mathbf{k}-\mathbf{1}}, \mathbf{d}_{\mathbf{ik}}^{\mathbf{k}-\mathbf{1}} + \mathbf{d}_{\mathbf{kj}}^{\mathbf{k}-\mathbf{1}}) && \text{(induction)} \\
&= \mathbf{d}_{\mathbf{ij}}^{\mathbf{k}} && \text{(optimal substructure)}
\end{aligned}$$

# Floyd-Warshall Algorithm

Usually bottom-up, since so simple:

$M[i,j,0] = \ell(i,j)$ for all $i,j \in [n]$
for($k = 1$ to $n$)
  for($i = 1$ to $n$)
    for($j = 1$ 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 $k \geq 1$:

$$
\begin{aligned}
M[i,j,k] &= \min(M[i,j,k-1], M[i,k,k-1] + M[k,j,k-1]) && \text{(def of algorithm)} \\
&= \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) && \text{(induction)} \\
&= d_{ij}^{k} && \text{(optimal substructure)}
\end{aligned}
$$

**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 $\mathbf{O(n(m + n \log n)) = O(mn + n^2 \log n)}$, better than Floyd-Warshall

# Reweighting

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

- Time would be $\mathbf{O(n(m + n \log n)) = O(mn + n^2 \log n)}$, 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!

# Reweighting

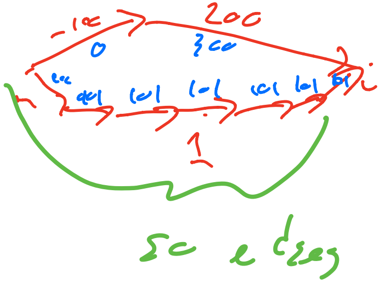
Different Approach: Can we "fix" negative weights so Dijkstra from every node works?

- Time would be $\mathbf{O(n(m + n\log n)) = O(mn + n^2 \log n)}$, 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 $\mathbf{P}$ is $\ell(\mathbf{P}) + \alpha|\mathbf{P}|$, so original shortest path might no longer be shortest path if it has many edges.



# Reweighting

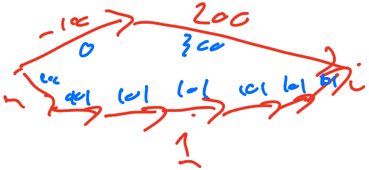
Different Approach: Can we "fix" negative weights so Dijkstra from every node works?

- Time would be $\mathbf{O(n(m + n\log n)) = O(mn + n^2 \log n)}$, 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 $\mathbf{P}$ is $\ell(\mathbf{P}) + \alpha|\mathbf{P}|$, 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:

# Reweighting

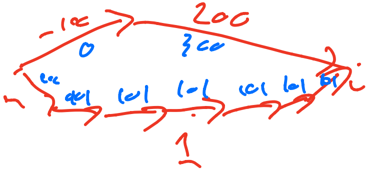
Different Approach: Can we "fix" negative weights so Dijkstra from every node works?

- Time would be $\mathbf{O(n(m + n \log n)) = O(mn + n^2 \log n)}$, 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 $\mathbf{P}$ is $\ell(\mathbf{P}) + \alpha|\mathbf{P}|$, so original shortest path might no longer be shortest path if it has many edges.



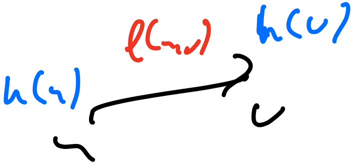

Some other kind of reweighting? Need new lengths $\hat{\ell}$ such that:

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

# Vertex Reweighting

Neat observation: put weights at *vertices*!

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



# Vertex Reweighting

Neat observation: put weights at *vertices*!

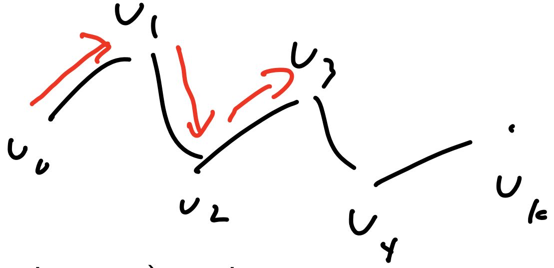
- Let $h : V \to \mathbb{R}$ be node weights.
- Let $\ell_h(u, v) = \ell(u, v) + h(u) - h(v)$

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

# Vertex Reweighting

Neat observation: put weights at *vertices*!

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




Let $\mathbf{P} = \langle \mathbf{v_0}, \mathbf{v_1}, \ldots, \mathbf{v_k} \rangle$ be arbitrary (not necessarily shortest) path.

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

# Vertex Reweighting

Neat observation: put weights at *vertices*!

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

Let $\mathbf{P} = \langle \mathbf{v_0}, \mathbf{v_1}, \dots, \mathbf{v_k} \rangle$ be arbitrary (not necessarily shortest) path.

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

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

# Vertex Reweighting

Neat observation: put weights at *vertices*!

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

Let $\mathbf{P} = \langle \mathbf{v_0}, \mathbf{v_1}, \ldots, \mathbf{v_k} \rangle$ be arbitrary (not necessarily shortest) path.

$$\begin{aligned}
\ell_{\mathbf{h}}(\mathbf{P}) = \sum_{i=0}^{k-1} \ell_{\mathbf{h}}(\mathbf{v_i}, \mathbf{v_{i+1}}) &= \sum_{i=0}^{k-1} (\ell(\mathbf{v_i}, \mathbf{v_{i+1}}) + \mathbf{h}(\mathbf{v_i}) - \mathbf{h}(\mathbf{v_{i+1}})) \\
&= \mathbf{h}(\mathbf{v_0}) - \mathbf{h}(\mathbf{v_k}) + \sum_{i=0}^{k-1} \ell(\mathbf{v_i}, \mathbf{v_{i+1}}) && \text{(telescoping)} \\
&= \ell(\mathbf{P}) + \mathbf{h}(\mathbf{v_0}) - \mathbf{h}(\mathbf{v_k})
\end{aligned}$$

# Vertex Reweighting

Neat observation: put weights at *vertices*!

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

Let $\mathbf{P} = \langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_k \rangle$ be arbitrary (not necessarily shortest) path.

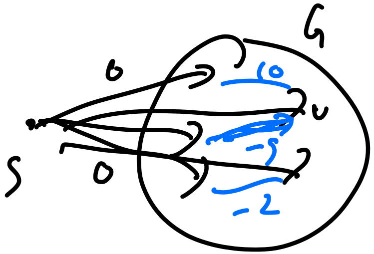
$$\begin{aligned}
\ell_{\mathbf{h}}(\mathbf{P}) &= \sum_{i=0}^{k-1} \ell_{\mathbf{h}}(\mathbf{v}_i, \mathbf{v}_{i+1}) = \sum_{i=0}^{k-1} \left(\ell(\mathbf{v}_i, \mathbf{v}_{i+1}) + \mathbf{h}(\mathbf{v}_i) - \mathbf{h}(\mathbf{v}_{i+1})\right) \\
&= \mathbf{h}(\mathbf{v}_0) - \mathbf{h}(\mathbf{v}_k) + \sum_{i=0}^{k-1} \ell(\mathbf{v}_i, \mathbf{v}_{i+1}) \qquad \text{(telescoping)} \\
&= \ell(\mathbf{P}) + \mathbf{h}(\mathbf{v}_0) - \mathbf{h}(\mathbf{v}_k)
\end{aligned}$$

$\mathbf{h}(\mathbf{v}_0) - \mathbf{h}(\mathbf{v}_k)$ added to *every* $\mathbf{v}_0 \to \mathbf{v}_k$ path, so shortest path from $\mathbf{v}_0$ to $\mathbf{v}_k$ still shortest path!

# Making lengths nonnegative

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

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



# Making lengths nonnegative

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

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

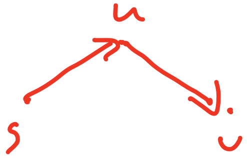
- Run Bellman-Ford from $\mathbf{s}$, then for all $\mathbf{u} \in \mathbf{V}$ set $\mathbf{h(u)}$ to be $\mathbf{d(s, u)}$
- Note $\mathbf{h(u)} \leq \mathbf{0}$ for all $\mathbf{u} \in \mathbf{V}$

# Making lengths nonnegative

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

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

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




Want to show that $\ell_{\mathbf{h}}\mathbf{(u, v)} \geq \mathbf{0}$ for all edges $\mathbf{(u, v)}$.

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

# Making lengths nonnegative

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

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

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

Want to show that $\ell_{\mathbf{h}}(\mathbf{u}, \mathbf{v}) \geq \mathbf{0}$ for all edges $(\mathbf{u}, \mathbf{v})$.

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

$$\ell_{\mathbf{h}}(\mathbf{u}, \mathbf{v}) = \ell(\mathbf{u}, \mathbf{v}) + \mathbf{h}(\mathbf{u}) - \mathbf{h}(\mathbf{v}) \geq \ell(\mathbf{u}, \mathbf{v}) + \mathbf{h}(\mathbf{u}) - (\mathbf{h}(\mathbf{u}) + \ell(\mathbf{u}, \mathbf{v})) = \mathbf{0}$$

# Johnson's Algorithm

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

**Correctness:** From previous discussion.

# Johnson's Algorithm

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

**Correctness:** From previous discussion.

**Running Time:**

$O(n)$

$O(mn)$

$O(m)$ to write $\ell_h$'s

$n \cdot O(m + n \log n) = O(nm + n^2 \log n)$

$O(n^2)$

# 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:** $O(n) + O(mn) + O(n(m + n \log n)) = O(mn + n^2 \log n)$