Aside: Slime Mould Solving Shortest Path
Dynamic Programming: Shortest Paths

Lecture 5
Feb 9, 2021

Most slides are courtesy Prof. Chekuri
Part I

Shortest Paths with Negative Length Edges
Single-Source Shortest Paths with Negative Edge Lengths

Single-Source Shortest Path Problems

**Input:** A directed graph \( G = (V, E) \) with arbitrary (including negative) edge lengths. For edge \( e = (u, v) \), \( \ell(e) = \ell(u, v) \) is its length.

1. Given nodes \( s, t \) find shortest path from \( s \) to \( t \).
2. Given node \( s \) find shortest path from \( s \) to all other nodes.
Single-Source Shortest Paths with Negative Edge Lengths

Single-Source Shortest Path Problems

**Input:** A directed graph $G = (V, E)$ with arbitrary (including negative) edge lengths. For edge $e = (u, v)$, $\ell(e) = \ell(u, v)$ is its length.

1. Given nodes $s, t$ find shortest path from $s$ to $t$.
2. Given node $s$ find shortest path from $s$ to all other nodes.
Negative Length Cycles

Definition

A cycle $C$ is a negative length cycle if the sum of the edge lengths of $C$ is negative.

![Graph with edge lengths](image)
Definition
A cycle $C$ is a negative length cycle if the sum of the edge lengths of $C$ is negative.
Shortest Paths and Negative Cycles

Given $G = (V, E)$ with edge lengths and $s, t$. Suppose

1. $G$ has a negative length cycle $C$, and
2. $s$ can reach $C$ and $C$ can reach $t$.

**Question:** What is the shortest distance from $s$ to $t$?

Possible answers: Define shortest distance to be:

1. undefined, that is $-\infty$
2. the length of a shortest simple path from $s$ to $t$. NP-Hard
Shortest Paths and Negative Cycles

Given $G = (V, E)$ with edge lengths and $s, t$. Suppose

1. $G$ has a negative length cycle $C$, and
2. $s$ can reach $C$ and $C$ can reach $t$.

**Question:** What is the shortest distance from $s$ to $t$?

Possible answers: Define shortest distance to be:

1. undefined, that is $-\infty$
   OR
2. the length of a shortest *simple* path from $s$ to $t$. 


Shortest Paths and Negative Cycles

Given $G = (V, E)$ with edge lengths and $s, t$. Suppose

1. $G$ has a negative length cycle $C$, and
2. $s$ can reach $C$ and $C$ can reach $t$.

Question: What is the shortest distance from $s$ to $t$?

Possible answers: Define shortest distance to be:

1. undefined, that is $-\infty$
   OR
2. the length of a shortest simple path from $s$ to $t$. NP-Hard!
Alterantively: Finding Shortest Walks

Given a graph $G = (V, E)$:

1. A path is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$. 

Define $\text{dist}(u, v)$ to be the length of a shortest walk from $u$ to $v$.

1. If there is a walk from $u$ to $v$ that contains a negative length cycle, then $\text{dist}(u, v) = -\infty$.

2. Else there is a path with at most $n - 1$ edges whose length is equal to the length of a shortest walk and $\text{dist}(u, v)$ is finite.

Helpful to think about walks.
Alternatively: Finding Shortest Walks

Given a graph $G = (V, E)$:

1. A path is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$.

2. A walk is a sequence of vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$. Vertices are allowed to repeat.

Define $\text{dist}(u, v)$ to be the length of a shortest walk from $u$ to $v$.

If there is a walk from $u$ to $v$ that contains a negative length cycle then $\text{dist}(u, v) = -\infty$.

Else there is a path with at most $n - 1$ edges whose length is equal to the length of a shortest walk and $\text{dist}(u, v)$ is finite.

Helpful to think about walks.
Alternatively: Finding Shortest Walks

Given a graph $G = (V, E)$:

1. A path is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$.

2. A walk is a sequence of vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$. Vertices are allowed to repeat.

Define $\text{dist}(u, v)$ to be the length of a shortest walk from $u$ to $v$. 
Given a graph $G = (V, E)$:

1. A path is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$.

2. A walk is a sequence of vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$. Vertices are allowed to repeat.

Define $dist(u, v)$ to be the length of a shortest walk from $u$ to $v$.

1. If there is a walk from $u$ to $v$ that contains negative length cycle then $dist(u, v) = -\infty$
Given a graph $G = (V, E)$:

1. A **path** is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$.

2. A **walk** is a sequence of vertices $v_1, v_2, \ldots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$. Vertices are allowed to repeat.

Define $\text{dist}(u, v)$ to be the length of a **shortest walk** from $u$ to $v$.

1. If there is a walk from $u$ to $v$ that contains negative length cycle then $\text{dist}(u, v) = -\infty$

2. Else there is a path with at most $n - 1$ edges whose length is equal to the length of a shortest walk and $\text{dist}(u, v)$ is finite

Helpful to think about walks
Shortest Paths with Negative Edge Lengths

Problems

Algorithmic Problems

Input: A directed graph \( G = (V, E) \) with edge lengths (could be negative). For edge \( e = (u, v) \), \( \ell(e) = \ell(u, v) \) is its length.

Questions:

1. Given nodes \( s, t \), either find a negative length cycle \( C \) that \( s \) can reach or find a shortest path from \( s \) to \( t \).
2. Given node \( s \), either find a negative length cycle \( C \) that \( s \) can reach or find shortest path distances from \( s \) to all reachable nodes.
3. Check if \( G \) has a negative length cycle or not.
Note: Negative cycle detection in undirected graph cannot be reduced to directed graph by bi-directing edges, why?
Note: Negative cycle detection in undirected graph cannot be reduced to directed graph by bi-directing edges, why?

Problem can be solved efficiently in undirected graphs but algorithms are different and more involved than those for directed graphs. Need min-cost matchings which we will see later in the course.
Why Negative Lengths?

Several Applications

1. Shortest path problems useful in modeling many situations — in some negative lengths are natural
2. Negative length cycle can be used to find arbitrage opportunities in currency trading
3. Important sub-routine in algorithms for more general problem: minimum-cost flow
What are the distances computed by Dijkstra’s algorithm?

The distance as computed by Dijkstra algorithm starting from $s$:

(A) $s = 0$, $x = 5$, $y = 1$, $z = 0$.

(B) $s = 0$, $x = 1$, $y = 2$, $z = 5$.

(C) $s = 0$, $x = 5$, $y = 1$, $z = 2$.

(D) IDK.
Dijkstra’s Algorithm and Negative Lengths

With negative length edges, Dijkstra’s algorithm can fail
Dijkstra’s Algorithm and Negative Lengths

With negative length edges, Dijkstra’s algorithm can fail.
Dijkstra’s Algorithm and Negative Lengths

With negative length edges, Dijkstra’s algorithm can fail.
With negative length edges, Dijkstra’s algorithm can fail.
Dijkstra’s Algorithm and Negative Lengths

With negative length edges, Dijkstra’s algorithm can fail
Dijkstra’s Algorithm and Negative Lengths

With negative length edges, Dijkstra’s algorithm can fail
Dijkstra’s Algorithm and Negative Lengths

With negative length edges, Dijkstra’s algorithm can fail.
With negative length edges, Dijkstra’s algorithm can fail.

False assumption: Dijkstra’s algorithm is based on the assumption that if $s = v_0 \rightarrow v_1 \rightarrow v_2 \ldots \rightarrow v_k$ is a shortest path from $s$ to $v_k$ then $\text{dist}(s, v_i) \leq \text{dist}(s, v_{i+1})$ for $0 \leq i < k$. Holds true only for non-negative edge lengths.
Lemma

Let \( G \) be a directed graph with arbitrary edge lengths. If \( s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k \) is a shortest path from \( s \) to \( v_k \) then for \( 1 \leq i < k \):

1. \( s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_i \) is a shortest path from \( s \) to \( v_i \)
Lemma

Let $G$ be a directed graph with arbitrary edge lengths. If $s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$ is a shortest path from $s$ to $v_k$ then for $1 \leq i < k$:

1. $s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_i$ is a shortest path from $s$ to $v_i$
2. False: $\text{dist}(s, v_i) \leq \text{dist}(s, v_k)$ for $1 \leq i < k$. Holds true only for non-negative edge lengths.
Lemma

Let $G$ be a directed graph with arbitrary edge lengths. If $s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$ is a shortest path from $s$ to $v_k$ then for $1 \leq i < k$:

1. $s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_i$ is a shortest path from $s$ to $v_i$

2. **False**: $\text{dist}(s, v_i) \leq \text{dist}(s, v_k)$ for $1 \leq i < k$. Holds true only for non-negative edge lengths.

Cannot explore nodes in increasing order of distance! We need other strategies.
Shortest Paths: Sub-problems

What are the smaller sub-problems?

Lemma
Let $G$ be a directed graph with arbitrary edge lengths. If $s = v_0 \rightarrow v_1 \rightarrow v_2 \cdots \rightarrow v_k$ is a shortest path from $s$ to $v_k$ then for $1 \leq i < k$:

$s = v_0 \rightarrow v_1 \rightarrow v_2 \cdots \rightarrow v_i$ is a shortest path from $s$ to $v_i$. 

Sub-problem idea: paths of fewer hops/edges
What are the smaller sub-problems?

**Lemma**

Let $G$ be a directed graph with arbitrary edge lengths. If $s = v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$ is a shortest path from $s$ to $v_k$ then for $1 \leq i < k$:

$s = v_0 \rightarrow v_1 \rightarrow v_2 \ldots \rightarrow v_i$ is a shortest path from $s$ to $v_i$
What are the smaller sub-problems?

**Lemma**

Let $G$ be a directed graph with arbitrary edge lengths. If $s = v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k$ is a shortest path from $s$ to $v_k$ then for $1 \leq i < k$:

$s = v_0 \rightarrow v_1 \rightarrow v_2 \ldots \rightarrow v_i$ is a shortest path from $s$ to $v_i$

Sub-problem idea: paths of fewer hops/edges
Hop-based Recursion: Bellman-Ford Algorithm

Single-source problem: fix source $s$. Assume that all nodes can be reached by $s$ in $G$. (Remove nodes unreachable from $s$).

$d(v, k)$: shortest walk length from $s$ to $v$ using at most $k$ edges ($\infty$ if none exists).
Hop-based Recursion: Bellman-Ford Algorithm

Single-source problem: fix source $s$. Assume that all nodes can be reached by $s$ in $G$. (Remove nodes unreachable from $s$).

$d(v, k)$: shortest walk length from $s$ to $v$ using at most $k$ edges ($\infty$ if none exists).
Hop-based Recursion: Bellman-Ford Algorithm

Single-source problem: fix source $s$. Assume that all nodes can be reached by $s$ in $G$. (Remove nodes unreachable from $s$).

$d(v, k)$: shortest walk length from $s$ to $v$ using at most $k$ edges ($\infty$ if none exists).

Recursion for $d(v, k)$:
Hop-based Recursion: Bellman-Ford Algorithm

Single-source problem: fix source $s$. Assume that all nodes can be reached by $s$ in $G$. (Remove nodes unreachable from $s$).

$d(v, k)$: shortest walk length from $s$ to $v$ using at most $k$ edges ($\infty$ if none exists).

Recursion for $d(v, k)$:

$$d(v, k) = \min \begin{cases} 
\min_{u:(u,v) \in E} (d(u, k - 1) + \ell(u, v)) \\
 d(v, k - 1)
\end{cases}$$

Base case:
Hop-based Recursion: Bellman-Ford Algorithm

Single-source problem: fix source $s$. Assume that all nodes can be reached by $s$ in $G$. (Remove nodes unreachable from $s$).

$d(v, k)$: shortest walk length from $s$ to $v$ using at most $k$ edges ($\infty$ if none exists).

Recursion for $d(v, k)$:

$$d(v, k) = \min \left\{ \min_{u:(u,v) \in E} (d(u, k - 1) + \ell(u, v)), d(v, k - 1) \right\}$$

Base case: $d(s, 0) = 0$ and $d(v, 0) = \infty$ for all $v \neq s$. 
Example
A Basic Lemma

Lemma

Assume $s$ can reach all nodes in $G = (V, E)$. Then,

1. There is a negative length cycle in $G$ iff $d(v, n) < d(v, n - 1)$ for some node $v \in V$.
2. If there is no negative length cycle in $G$ then $\text{dist}(s, v) = d(v, n - 1)$ for all $v \in V$. 
Bellman-Ford Algorithm

for each $u \in V$ do

$d(u, 0) \leftarrow \infty$

$d(s, 0) \leftarrow 0$
Bellman-Ford Algorithm

for each $u \in V$ do
    $d(u, 0) \leftarrow \infty$
    $d(s, 0) \leftarrow 0$

for $k = 1$ to $n$ do
    for each $v \in V$ do
        $d(v, k) \leftarrow d(v, k - 1)$
    for each edge $(u, v) \in \text{In}(v)$ do
        $d(v, k) = \min\{d(v, k), d(u, k - 1) + \ell(u, v)\}$

for each $v \in V$ do
    $\text{dist}(s, v) \leftarrow d(v, n - 1)$

If $d(v, n) < d(v, n - 1)$
    Return "Negative Cycle in $G"$

Running time: $O(mn)$
Space: $O(m + n^2)$
Space can be reduced to $O(m + n)$.
Bellman-Ford Algorithm

for each \( u \in V \) do
    \( d(u, 0) \leftarrow \infty \)
    \( d(s, 0) \leftarrow 0 \)

for \( k = 1 \) to \( n \) do
    for each \( v \in V \) do
        for each \( u \in V \) do
            \( d(v, k) \leftarrow d(v, k - 1) \)
        for each edge \((u, v) \in \text{In}(v)\) do
            \( d(v, k) = \min\{d(v, k), d(u, k - 1) + \ell(u, v)\} \)

for each \( v \in V \) do
    \( \text{dist}(s, v) \leftarrow d(v, n - 1) \)
    If \( d(v, n) < d(v, n - 1) \)
    Return ‘‘Negative Cycle in \( G \)’’
Bellman-Ford Algorithm

for each $u \in V$ do
  $d(u, 0) \leftarrow \infty$
  $d(s, 0) \leftarrow 0$

for $k = 1$ to $n$ do
  for each $v \in V$ do
    for each $u \in V$ do
      $d(v, k) \leftarrow d(v, k - 1)$
    for each edge $(u, v) \in \text{In}(v)$ do
      $d(v, k) = \min\{d(v, k), d(u, k - 1) + \ell(u, v)\}$

for each $v \in V$ do
  $\text{dist}(s, v) \leftarrow d(v, n - 1)$
  If $d(v, n) < d(v, n - 1)$
    Return ‘‘Negative Cycle in $G$’’

Running time: $O(mn)$

Space: $O(m + n^2)$
Space can be reduced to $O(m + n)$. 
Bellman-Ford Algorithm

\[
\begin{align*}
&\text{for each } u \in V \text{ do} \\
&\quad d(u, 0) \leftarrow \infty \\
&\quad d(s, 0) \leftarrow 0 \\
&\text{for } k = 1 \text{ to } n \text{ do} \\
&\quad \text{for each } v \in V \text{ do} \\
&\quad\quad d(v, k) \leftarrow d(v, k - 1) \\
&\quad\quad \text{for each edge } (u, v) \in \text{In}(v) \text{ do} \\
&\quad\quad\quad d(v, k) = \min\{d(v, k), d(u, k - 1) + \ell(u, v)\} \\
&\text{for each } v \in V \text{ do} \\
&\quad \text{dist}(s, v) \leftarrow d(v, n - 1) \\
&\quad \text{If } d(v, n) < d(v, n - 1) \\
&\quad\quad \text{Return ‘‘Negative Cycle in } G’’
\end{align*}
\]

Running time: \( O(mn) \)
Bellman-Ford Algorithm

\[
\begin{align*}
\text{for each } u \in V & \text{ do} \\
& d(u, 0) \leftarrow \infty \\
& d(s, 0) \leftarrow 0 \\
\text{for } k = 1 \text{ to } n & \text{ do} \\
& \quad \text{for each } v \in V \text{ do} \\
& & \quad d(v, k) \leftarrow d(v, k - 1) \\
& & \quad \text{for each edge } (u, v) \in \text{In}(v) \text{ do} \\
& & & \quad d(v, k) = \min\{d(v, k), d(u, k - 1) + \ell(u, v)\} \\
\text{for each } v \in V & \text{ do} \\
& & \quad \text{dist}(s, v) \leftarrow d(v, n - 1) \\
& & \quad \text{If } d(v, n) < d(v, n - 1) \\
& & & \quad \text{Return ‘‘Negative Cycle in } G \text{’’}
\end{align*}
\]

Running time: \(O(mn)\) Space:
Bellman-Ford Algorithm

for each \( u \in V \) do \\
\( d(u, 0) \leftarrow \infty \) \\
\( d(s, 0) \leftarrow 0 \)

for \( k = 1 \) to \( n \) do \\
\hspace{1em} for each \( v \in V \) do \\
\hspace{2em} \( d(v, k) \leftarrow d(v, k - 1) \) \\
\hspace{2em} for each edge \( (u, v) \in \text{In}(v) \) do \\
\hspace{3em} \( d(v, k) = \min\{d(v, k), d(u, k - 1) + \ell(u, v)\} \)

for each \( v \in V \) do \\
\( \text{dist}(s, v) \leftarrow d(v, n - 1) \) \\
If \( d(v, n) < d(v, n - 1) \) \\
Return ‘‘Negative Cycle in \( G \)’’

Running time: \( O(mn) \) Space: \( O(m + n^2) \)
Bellman-Ford Algorithm

for each $u \in V$ do
  $d(u, 0) \leftarrow \infty$
  $d(s, 0) \leftarrow 0$

for $k = 1$ to $n$ do
  for each $v \in V$ do
    $d(v, k) \leftarrow d(v, k - 1)$
    for each edge $(u, v) \in \text{In}(v)$ do
      $d(v, k) = \min\{d(v, k), d(u, k - 1) + \ell(u, v)\}$

for each $v \in V$ do
  $\text{dist}(s, v) \leftarrow d(v, n - 1)$
  If $d(v, n) < d(v, n - 1)$
  Return ‘‘Negative Cycle in $G$’’

Running time: $O(mn)$  Space: $O(m + n^2)$
Space can be reduced to $O(m + n)$. 
Bellman-Ford with Space Saving

\[
\begin{align*}
&\text{for each } u \in V \text{ do} \\
&\quad d(u) \leftarrow \infty \\
&\quad d(s) \leftarrow 0 \\
&\text{for } k = 1 \text{ to } n - 1 \text{ do} \\
&\quad \text{for each } v \in V \text{ do} \\
&\quad\quad \text{for each edge } (u, v) \in \text{ln}(v) \text{ do} \\
&\quad\quad\quad d(v) = \min\{d(v), d(u) + \ell(u, v)\}
\end{align*}
\]
Bellman-Ford with Space Saving

for each $u \in V$ do
    $d(u) \leftarrow \infty$
    $d(s) \leftarrow 0$

for $k = 1$ to $n - 1$ do
    for each $v \in V$ do
        for each edge $(u, v) \in \text{In}(v)$ do
            $d(v) = \min\{d(v), d(u) + \ell(u, v)\}$

(* One more iteration to check if distances change *)

for each $v \in V$ do
    for each edge $(u, v) \in \text{In}(v)$ do
        if $(d(v) > d(u) + \ell(u, v))$
            Output ‘‘Negative Cycle’’

for each $v \in V$ do
    $\text{dist}(s, v) \leftarrow d(v)$

Exercise: Show that this algorithm achieves same result.
Correctness of the Bellman-Ford Algorithm

**Via induction show:** For each $v$, $d(v, k)$ is the length of a shortest walk from $s$ to $v$ with *at most* $k$ hops.
Via induction show: For each $v$, $d(v, k)$ is the length of a shortest walk from $s$ to $v$ with at most $k$ hops. 
And for each $1 \leq k \leq n - 1$, $d(v, k) \leq d(v, k - 1)$.
Correctness of the Bellman-Ford Algorithm

Via induction show: For each $v$, $d(v, k)$ is the length of a shortest walk from $s$ to $v$ with at most $k$ hops. And for each $1 \leq k \leq n - 1$, $d(v, k) \leq d(v, k - 1)$.

Lemma

Assume $s$ can reach all nodes in $G = (V, E)$. Then,

1. There is a negative length cycle in $G$ iff $d(v, n) < d(v, n - 1)$ for some node $v \in V$.

2. If there is no negative length cycle in $G$ then $\text{dist}(s, v) = d(v, n - 1)$ for all $v \in V$. 

Correctness of the Bellman-Ford Algorithm

Via induction show: For each $v$, $d(v, k)$ is the length of a shortest walk from $s$ to $v$ with at most $k$ hops. And for each $1 \leq k \leq n - 1$, $d(v, k) \leq d(v, k - 1)$.

Lemma

Assume $s$ can reach all nodes in $G = (V, E)$. Then,

1. There is a negative length cycle in $G$ iff $d(v, n) < d(v, n - 1)$ for some node $v \in V$.

2. If there is no negative length cycle in $G$ then $\text{dist}(s, v) = d(v, n - 1)$ for all $v \in V$.

Exercise: Prove algorithm correctness from above two.
Proof of Lemma

Proposition

Suppose there is no negative length cycle in $G$ then
\[ d(v, h) \geq d(v, n - 1) \] for all $h \geq n$ and for all $v \in V$.

Proof.

By contradiction. Suppose for some $v$, $d(v, h) < d(v, n - 1)$ for an $h \geq n$. 
Proposition

Suppose there is no negative length cycle in $G$ then $d(v, h) \geq d(v, n - 1)$ for all $h \geq n$ and for all $v \in V$.

Proof.

By contradiction. Suppose for some $v$, $d(v, h) < d(v, n - 1)$ for an $h \geq n$. Choose smallest such $h$. $P : s-v$ walk with $h$ edges of length $d(v, h)$. 
Proof of Lemma

Proposition

Suppose there is no negative length cycle in $G$ then $d(v, h) \geq d(v, n - 1)$ for all $h \geq n$ and for all $v \in V$.

Proof.

By contradiction. Suppose for some $v$, $d(v, h) < d(v, n - 1)$ for an $h \geq n$. Choose smallest such $h$. $P : s-v$ walk with $h$ edges of length $d(v, h)$. $P$ has a cycle $C$. 
Proof of Lemma

Proposition

Suppose there is no negative length cycle in $G$ then

$$d(v, h) \geq d(v, n - 1)$$

for all $h \geq n$ and for all $v \in V$.

Proof.

By contradiction. Suppose for some $v$, $d(v, h) < d(v, n - 1)$ for an $h \geq n$. Choose smallest such $h$. $P : s-v$ walk with $h$ edges of length $d(v, h)$. $P$ has a cycle $C$. $P' :$ walk after removing $C$ from $P$. $k$ : #edges on $P'$
Proof of Lemma

Proposition

Suppose there is no negative length cycle in $G$ then $d(v, h) \geq d(v, n - 1)$ for all $h \geq n$ and for all $v \in V$.

Proof.

By contradiction. Suppose for some $v$, $d(v, h) < d(v, n - 1)$ for an $h \geq n$. Choose smallest such $h$. $P : s-v$ walk with $h$ edges of length $d(v, h)$. $P$ has a cycle $C$. $P'$ : walk after removing $C$ from $P$. $k : \#edges$ on $P'$ $\ell(P') = \ell(P) - \ell(C) \leq \ell(P)$. 
Proof of Lemma

Proposition

Suppose there is no negative length cycle in $G$ then $d(v, h) \geq d(v, n - 1)$ for all $h \geq n$ and for all $v \in V$.

Proof.

By contradiction. Suppose for some $v$, $d(v, h) < d(v, n - 1)$ for an $h \geq n$. Choose smallest such $h$. $P : s-v$ walk with $h$ edges of length $d(v, h)$. $P$ has a cycle $C$. $P'$ : walk after removing $C$ from $P$. $k : \#$edges on $P'$

$\ell(P') = \ell(P) - \ell(C) \leq \ell(P)$.

Case I $k \leq (n - 1)$:
Proof of Lemma

**Proposition**

Suppose there is no negative length cycle in $G$ then $d(v, h) \geq d(v, n - 1)$ for all $h \geq n$ and for all $v \in V$.

**Proof.**

By contradiction. Suppose for some $v$, $d(v, h) < d(v, n - 1)$ for an $h \geq n$. Choose smallest such $h$. $P: s-v$ walk with $h$ edges of length $d(v, h)$. $P$ has a cycle $C$. $P'$: walk after removing $C$ from $P$. $k$ : #edges on $P'$ $\ell(P') = \ell(P) - \ell(C) \leq \ell(P)$.

Case I $k \leq (n - 1)$: $d(v, k) \leq \ell(P') \leq \ell(P) < d(v, n - 1)$, a contradiction.
Proof of Lemma

Proposition

Suppose there is no negative length cycle in \( G \) then

\[
d(v, h) \geq d(v, n - 1) \quad \text{for all } h \geq n \text{ and for all } v \in V.
\]

Proof.

By contradiction. Suppose for some \( v \), \( d(v, h) < d(v, n - 1) \) for an \( h \geq n \). Choose smallest such \( h \). \( P : s-v \) walk with \( h \) edges of length \( d(v, h) \).

\( P \) has a cycle \( C \). \( P' : \) walk after removing \( C \) from \( P \). \( k : \) #edges on \( P' \)

\[
\ell(P') = \ell(P) - \ell(C) \leq \ell(P).
\]

Case I \( k \leq (n - 1) \): \( d(v, k) \leq \ell(P') \leq \ell(P) < d(v, n - 1) \), a contradiction.

Case II \( k > (n - 1) \):
Proof of Lemma

Proposition

Suppose there is no negative length cycle in $G$ then
\[ d(v, h) \geq d(v, n - 1) \] for all $h \geq n$ and for all $v \in V$.

Proof.

By contradiction. Suppose for some $v$, $d(v, h) < d(v, n - 1)$ for an $h \geq n$. Choose smallest such $h$. $P: s-v$ walk with $h$ edges of length $d(v, h)$. $P$ has a cycle $C$. $P'$: walk after removing $C$ from $P$. $k$ : #edges on $P'$

$\ell(P') = \ell(P) - \ell(C) \leq \ell(P)$.

Case I $k \leq (n - 1)$: $d(v, k) \leq \ell(P') \leq \ell(P) < d(v, n - 1)$, a contradiction.

Case II $k > (n - 1)$: $k < h \Rightarrow$ a contradiction to the choice of $h$. □
Proof of Lemma

Proposition

Suppose there is no negative length cycle in $G$ then $d(v, h) \geq d(v, n - 1)$ for all $h \geq n$ and for all $v \in V$. 
Proof of Lemma cond

**Proposition**

If $G$ has a negative length cycle reachable from $s$ then there is some $v$ such that $d(v, n) < d(v, n - 1)$.

**Proof.**
Proof of Lemma cond

Proposition

*If $G$ has a negative length cycle reachable from $s$ then there is some $v$ such that $d(v, n) < d(v, n - 1)$.*

Proof.

Suppose not.
Proof of Lemma cond

Proposition

If $G$ has a negative length cycle reachable from $s$ then there is some $v$ such that $d(v, n) < d(v, n - 1)$.

Proof.

Suppose not.

Let $C = v_1 \rightarrow \ldots \rightarrow v_h \rightarrow v_1$ be negative length cycle reachable from $s$. $d(v_i, n - 1)$ is finite for $1 \leq i \leq h$ since $C$ is reachable from $s$. 

Ruta (UIUC)  
CS473  
Spring 2021
Proof of Lemma cond

**Proposition**

If $G$ has a negative length cycle reachable from $s$ then there is some $v$ such that $d(v, n) < d(v, n - 1)$.

**Proof.**

Suppose not.

Let $C = v_1 \rightarrow \ldots \rightarrow v_h \rightarrow v_1$ be negative length cycle reachable from $s$.

$d(v_i, n - 1)$ is finite for $1 \leq i \leq h$ since $C$ is reachable from $s$.

By assumption $d(v, n - 1) \leq d(v, n)$ for all $v \in C$; this means
Proof of Lemma cond

Proposition

If $G$ has a negative length cycle reachable from $s$ then there is some $v$ such that $d(v, n) < d(v, n - 1)$.

Proof.

Suppose not. Let $C = v_1 \rightarrow \ldots \rightarrow v_h \rightarrow v_1$ be negative length cycle reachable from $s$. $d(v_i, n - 1)$ is finite for $1 \leq i \leq h$ since $C$ is reachable from $s$. By assumption $d(v, n - 1) \leq d(v, n)$ for all $v \in C$; this means

$$d(v_i, n - 1) \leq d(v_i, n) \leq d(v_{i-1}, n - 1) + \ell(v_{i-1}, v_i) \quad \text{for } 2 \leq i \leq h$$

and

$$d(v_1, n - 1) \leq d(v_1, n) \leq d(v_n, n - 1) + \ell(v_n, v_1).$$
Proof of Lemma cond

Proposition

If $G$ has a negative length cycle reachable from $s$ then there is some $v$ such that $d(v, n) < d(v, n - 1)$.

Proof.

Suppose not.
Let $C = v_1 \rightarrow \ldots \rightarrow v_h \rightarrow v_1$ be negative length cycle reachable from $s$.
$d(v_i, n - 1)$ is finite for $1 \leq i \leq h$ since $C$ is reachable from $s$.
By assumption $d(v, n - 1) \leq d(v, n)$ for all $v \in C$; this means

$$d(v_i, n - 1) \leq d(v_i, n) \leq d(v_{i-1}, n - 1) + \ell(v_{i-1}, v_i)$$
for $2 \leq i \leq h$

and

$$d(v_1, n - 1) \leq d(v_1, n) \leq d(v_n, n - 1) + \ell(v_n, v_1).$$

Adding up all these inequalities results in the inequality $0 \leq \ell(C)$ which contradicts the assumption that $\ell(C) < 0$. 

□
Proof of Lemma contd

Exercise: Finish proof of lemma using the two propositions.
Finding the Paths and a Shortest Path Tree

How do we find a shortest path tree in addition to distances?

- For each \( v \) the \( d(v) \) can only get smaller as algorithm proceeds.
- If \( d(v) \) becomes smaller it is because we found a vertex \( u \) such that \( d(v) > d(u) + \ell(u, v) \) and we update \( d(v) = d(u) + \ell(u, v) \). That is, we found a shorter path to \( v \) through \( u \).

For each \( v \) have a \( \text{prev}(v) \) pointer and update it to point to \( u \) if \( v \) finds a shorter path via \( u \). At end of algorithm \( \text{prev}(v) \) pointers give a shortest path tree oriented towards the source \( s \).
Finding the Paths and a Shortest Path Tree

How do we find a shortest path tree in addition to distances?

- For each $v$ the $d(v)$ can only get smaller as algorithm proceeds.
- If $d(v)$ becomes smaller it is because we found a vertex $u$ such that $d(v) > d(u) + \ell(u, v)$ and we update $d(v) = d(u) + \ell(u, v)$. That is, we found a shorter path to $v$ through $u$.
- For each $v$ have a $\text{prev}(v)$ pointer and update it to point to $u$ if $v$ finds a shorter path via $u$.
- At end of algorithm $\text{prev}(v)$ pointers give a shortest path tree oriented towards the source $s$. 
Given directed graph $G$ with arbitrary edge lengths, does it have a negative length cycle?
Negative Cycle Detection

Given directed graph $G$ with arbitrary edge lengths, does it have a negative length cycle?

Bellman-Ford checks whether there is a negative cycle $C$ that is reachable from a specific vertex $s$. There may be negative cycles not reachable from $s$. 
Given directed graph $G$ with arbitrary edge lengths, does it have a negative length cycle?

1. Bellman-Ford checks whether there is a negative cycle $C$ that is reachable from a specific vertex $s$. There may be negative cycles not reachable from $s$.

2. Run Bellman-Ford $|V|$ times, one from each node $u$?
Add a new node $s'$ and connect it to all nodes of $G$ with zero length edges. Bellman-Ford from $s'$ will fill find a negative length cycle if there is one. Exercise: why does this work?

Negative cycle detection can be done with one Bellman-Ford invocation.
Part II

Shortest Paths in DAGs
Single-Source Shortest Path Problems

- **Input**: A directed acyclic graph $G = (V, E)$ with arbitrary (including negative) edge lengths. For edge $e = (u, v)$, $\ell(e) = \ell(u, v)$ is its length.

1. Given nodes $s, t$ find shortest path from $s$ to $t$.
2. Given node $s$ find shortest path from $s$ to all other nodes.
Shortest Paths in a DAG

Single-Source Shortest Path Problems

Input: A directed acyclic graph $G = (V, E)$ with arbitrary (including negative) edge lengths. For edge $e = (u, v)$, $\ell(e) = \ell(u, v)$ is its length.

1. Given nodes $s, t$ find shortest path from $s$ to $t$.
2. Given node $s$ find shortest path from $s$ to all other nodes.

Simplification of algorithms for DAGs

1. No cycles and hence no negative length cycles! Hence can find shortest paths even for negative edge weights.
2. Can order nodes using topological sort.
Algorithm for DAGs

1. Want to find shortest paths from $s$. Ignore nodes not reachable from $s$.
2. Let $s = v_1, v_2, v_{i+1}, \ldots, v_n$ be a topological sort of $G$. 

Observation:
1. Shortest path from $s$ to $v_i$ cannot use any node from $v_i+1, \ldots, v_n$, since no path from $s$ to $v_i$ uses any of them.
2. Can find shortest paths in topological sort order.
Algorithm for DAGs

1. Want to find shortest paths from $s$. Ignore nodes not reachable from $s$.

2. Let $s = v_1, v_2, v_{i+1}, \ldots, v_n$ be a topological sort of $G$.

Observation:

1. Shortest path from $s$ to $v_i$ cannot use any node from $v_{i+1}, \ldots, v_n$. 
Algorithm for DAGs

1. Want to find shortest paths from $s$. Ignore nodes not reachable from $s$.
2. Let $s = v_1, v_2, v_{i+1}, \ldots, v_n$ be a topological sort of $G$

Observation:

1. shortest path from $s$ to $v_i$ cannot use any node from $v_{i+1}, \ldots, v_n$, since no path from $s$ to $v_i$ uses any of them.
2. can find shortest paths in topological sort order.
Algorithm for DAGs

\[
\text{for } i = 1 \text{ to } n \text{ do} \\
\quad d(s, v_i) = \infty \\
\quad d(s, s) = 0 \\
\text{for } i = 1 \text{ to } n - 1 \text{ do} \\
\quad \text{for each edge } (v_i, v_j) \text{ in } \text{Adj}(v_i) \text{ do} \\
\quad \quad d(s, v_j) = \min\{d(s, v_j), d(s, v_i) + \ell(v_i, v_j)\} \\
\text{return } d(s, \cdot) \text{ values computed}
\]
Algorithm for DAGs

\[
\begin{align*}
\text{for } i = 1 \text{ to } n \text{ do} \\
& \quad d(s, v_i) = \infty \\
& \quad d(s, s) = 0 \\
\text{for } i = 1 \text{ to } n - 1 \text{ do} \\
& \quad \text{for each edge } (v_i, v_j) \text{ in } \text{Adj}(v_i) \text{ do} \\
& \quad \quad d(s, v_j) = \min\{d(s, v_j), d(s, v_i) + \ell(v_i, v_j)\} \\
\text{return } d(s, \cdot) \text{ values computed}
\end{align*}
\]

Correctness by induction: If by the end of \(i\)th round \(d(s, v_j)\) is the shortest path length from \(s\) to \(v_j\) for each \(1 \leq j \leq i\), then after \((i + 1)\)th round \(d(s, v_{i+1})\) is the shortest path length from \(s\) to \(v_{i+1}\).
Algorithm for DAGs

\begin{algorithm}
\begin{algorithmic}
\For{$i = 1$ to $n$}
\State $d(s, v_i) = \infty$
\State $d(s, s) = 0$
\EndFor
\For{$i = 1$ to $n-1$}
\For{each edge $(v_i, v_j)$ in $\text{Adj}(v_i)$}
\State $d(s, v_j) = \min\{d(s, v_j), d(s, v_i) + \ell(v_i, v_j)\}$
\EndFor
\EndFor
\Return $d(s, \cdot)$ values computed
\end{algorithmic}
\end{algorithm}

Correctness by induction: If by the end of $i$th round $d(s, v_j)$ is the shortest path length from $s$ to $v_j$ for each $1 \leq j \leq i$, then after $(i + 1)$th round $d(s, v_{i+1})$ is the shortest path length from $s$ to $v_{i+1}$. Use observation in the previous slide.
Algorithm for DAGs

\[
\begin{align*}
\text{for } i = 1 & \text{ to } n \text{ do} \\
& d(s, v_i) = \infty \\
& d(s, s) = 0 \\
\text{for } i = 1 & \text{ to } n - 1 \text{ do} \\
& \text{for each edge } (v_i, v_j) \text{ in Adj}(v_i) \text{ do} \\
& d(s, v_j) = \min\{d(s, v_j), d(s, v_i) + \ell(v_i, v_j)\} \\
\text{return } d(s, \cdot) \text{ values computed}
\end{align*}
\]

Correctness by induction: If by the end of \(i\)th round \(d(s, v_j)\) is the shortest path length from \(s\) to \(v_j\) for each \(1 \leq j \leq i\), then after \((i + 1)\)th round \(d(s, v_{i+1})\) is the shortest path length from \(s\) to \(v_{i+1}\). Use observation in the previous slide.

Running time: \(O(m + n)\) time algorithm! Works for negative edge lengths and hence can find longest paths in a DAG.
Part III

All Pairs Shortest Paths
Shortest Path Problems

Input A (undirected or directed) graph \( G = (V, E) \) with edge lengths (or costs). For edge \( e = (u, v) \), \( \ell(e) = \ell(u, v) \) is its length.

1. Given nodes \( s, t \) find shortest path from \( s \) to \( t \).
2. Given node \( s \) find shortest path from \( s \) to all other nodes.
3. Find shortest paths for all pairs of nodes.
Single-Source Shortest Path Problems

Input  A (undirected or directed) graph $G = (V, E)$ with edge lengths. For edge $e = (u, v)$, $\ell(e) = \ell(u, v)$ is its length.

1. Given nodes $s, t$ find shortest path from $s$ to $t$.
2. Given node $s$ find shortest path from $s$ to all other nodes.
**Single-Source Shortest Paths**

### Single-Source Shortest Path Problems

**Input** A (undirected or directed) graph $G = (V, E)$ with edge lengths. For edge $e = (u, v)$, $\ell(e) = \ell(u, v)$ is its length.

1. Given nodes $s, t$ find shortest path from $s$ to $t$.
2. Given node $s$ find shortest path from $s$ to all other nodes.

**Dijkstra’s algorithm** for non-negative edge lengths. Running time: $O((m + n) \log n)$ with heaps and $O(m + n \log n)$ with advanced priority queues.

**Bellman-Ford algorithm** for arbitrary edge lengths. Running time: $O(nm)$. 
All-Pairs Shortest Paths

All-Pairs Shortest Path Problem

**Input** A (undirected or directed) graph \( G = (V, E) \) with edge lengths. For edge \( e = (u, v) \), \( \ell(e) = \ell(u, v) \) is its length.

1. Find shortest paths for all pairs of nodes.

- For non-negative lengths: \( O(nm \log n) \) with heaps and \( O(nm + n^2 \log n) \) using advanced priority queues.
- For arbitrary edge lengths: \( O(n^2 m) \).
- \( \Theta(n^4) \) if \( m = \Omega(n^2) \).
All-Pairs Shortest Path Problem

Input A (undirected or directed) graph $G = (V, E)$ with edge lengths. For edge $e = (u, v)$, $\ell(e) = \ell(u, v)$ is its length.

1. Find shortest paths for all pairs of nodes.

Apply single-source algorithms $n$ times, once for each vertex.

1. Non-negative lengths. $O(nm \log n)$ with heaps and $O(nm + n^2 \log n)$ using advanced priority queues.

2. Arbitrary edge lengths: $O(n^2 m)$. \( \Theta(n^4) \) if $m = \Omega(n^2)$. 

Can we do better?
## All-Pairs Shortest Paths

### All-Pairs Shortest Path Problem

**Input** A (undirected or directed) graph $G = (V, E)$ with edge lengths. For edge $e = (u, v)$, $\ell(e) = \ell(u, v)$ is its length.

1. **Find shortest paths for all pairs of nodes.**

Apply single-source algorithms $n$ times, once for each vertex.

1. **Non-negative lengths.** $O(nm \log n)$ with heaps and $O(nm + n^2 \log n)$ using advanced priority queues.

2. **Arbitrary edge lengths:** $O(n^2m)$. $\Theta(n^4)$ if $m = \Omega(n^2)$.

Can we do better?
All-Pairs: Recursion on index of intermediate nodes

1. Number vertices arbitrarily as $v_1, v_2, \ldots, v_n$

2. $dist(i, j, k)$: length of shortest walk from $v_i$ to $v_j$ among all walks in which the largest index of an intermediate node is at most $k$ (could be $-\infty$ if there is a negative length cycle).

![Graph with labeled edges]

$\begin{align*}
dist(i, j, 0) &= \\
dist(i, j, 1) &= \\
dist(i, j, 2) &= \\
dist(i, j, 3) &= 
\end{align*}$
All-Pairs: Recursion on index of intermediate nodes

1. Number vertices arbitrarily as $v_1, v_2, \ldots, v_n$

2. $\text{dist}(i, j, k)$: length of shortest walk from $v_i$ to $v_j$ among all walks in which the largest index of an *intermediate node* is at most $k$ (could be $-\infty$ if there is a negative length cycle).

\[
\begin{align*}
\text{dist}(i, j, 0) &= 100 \\
\text{dist}(i, j, 1) &= \\
\text{dist}(i, j, 2) &= \\
\text{dist}(i, j, 3) &= \\
\end{align*}
\]
All-Pairs: Recursion on index of intermediate nodes

1. Number vertices arbitrarily as $v_1, v_2, \ldots, v_n$

2. $dist(i, j, k)$: length of shortest walk from $v_i$ to $v_j$ among all walks in which the largest index of an intermediate node is at most $k$ (could be $-\infty$ if there is a negative length cycle).

$dist(i, j, 0) = 100$
$dist(i, j, 1) = 9$
$dist(i, j, 2) =$
$dist(i, j, 3) =$
1. Number vertices arbitrarily as $v_1, v_2, \ldots, v_n$
2. $\text{dist}(i, j, k)$: length of shortest walk from $v_i$ to $v_j$ among all walks in which the largest index of an intermediate node is at most $k$ (could be $-\infty$ if there is a negative length cycle).

\begin{align*}
\text{dist}(i, j, 0) &= 100 \\
\text{dist}(i, j, 1) &= 9 \\
\text{dist}(i, j, 2) &= 8 \\
\text{dist}(i, j, 3) &= \\
\end{align*}
All-Pairs: Recursion on index of intermediate nodes

1. Number vertices arbitrarily as $v_1, v_2, \ldots, v_n$

2. $\text{dist}(i, j, k)$: length of shortest walk from $v_i$ to $v_j$ among all walks in which the largest index of an intermediate node is at most $k$ (could be $-\infty$ if there is a negative length cycle).

$$
\begin{align*}
\text{dist}(i, j, 0) &= 100 \\
\text{dist}(i, j, 1) &= 9 \\
\text{dist}(i, j, 2) &= 8 \\
\text{dist}(i, j, 3) &= 5
\end{align*}
$$
For the following graph, \( \text{dist}(i, j, 2) \) is...

(A) 9
(B) 10
(C) 11
(D) 12
(E) 15
All-Pairs: Recursion on index of intermediate nodes

\[ \text{dist}(i, k, k - 1) \rightarrow k \rightarrow \text{dist}(k, j, k - 1) \]

\[ \text{dist}(i, j, k - 1) \]

\[ \text{dist}(i, j, k) = \min \begin{cases} 
\text{dist}(i, j, k - 1) \\
\text{dist}(i, k, k - 1) + \text{dist}(k, j, k - 1) 
\end{cases} \]

Base case: \(\text{dist}(i, j, 0) = \ell(i, j)\) if \((i, j) \in E\), otherwise \(\infty\)

Correctness: If \(i \rightarrow j\) shortest walk goes through \(k\) then \(k\) occurs only once on the path — otherwise there is a negative length cycle.
**All-Pairs: Recursion on index of intermediate nodes**

\[
dist(i, k, k - 1) \quad k \quad dist(k, j, k - 1)
\]

\[
dist(i, j, k - 1)
\]

\[
dist(i, j, k) = \min \left\{ \begin{array}{ll}
\dist(i, j, k - 1) \\
\dist(i, k, k - 1) + \dist(k, j, k - 1)
\end{array} \right\}
\]

**Base case:** \( \dist(i, j, 0) = \ell(i, j) \) if \((i, j) \in E\), otherwise \(\infty\)
All-Pairs: Recursion on index of intermediate nodes

\[
dist(i, k, k - 1) \quad k \quad dist(k, j, k - 1)
\]

\[
dist(i, j, k - 1)
\]

\[
dist(i, j, k) = \min \left\{ \begin{array}{l}
\dist(i, j, k - 1) \\
\dist(i, k, k - 1) + \dist(k, j, k - 1)
\end{array} \right\}
\]

Base case: \( \dist(i, j, 0) = \ell(i, j) \) if \( (i, j) \in E \), otherwise \( \infty \)

Correctness: If \( i \rightarrow j \) shortest walk goes through \( k \) then \( k \) occurs only once on the path — otherwise there is a negative length cycle.
All-Pairs: Recursion on index of intermediate nodes

If \( \text{dist}(k, k, k - 1) < 0 \) then \( G \) has a negative length cycle containing \( k \).
If \( \text{dist}(k, k, k - 1) < 0 \) then \( G \) has a negative length cycle containing \( k \).
Now if \( i \) can reach \( k \) and \( k \) can reach \( j \) then \( \text{dist}(i, j, k) = -\infty \).

Therefore, recursion below is valid only if \( \text{dist}(k, k, k - 1) \geq 0 \).

\[
\text{dist}(i, j, k) = \min \left\{ \begin{array}{l}
\text{dist}(i, j, k - 1) \\
\text{dist}(i, k, k - 1) + \text{dist}(k, j, k - 1)
\end{array} \right\}
\]
All-Pairs: Recursion on index of intermediate nodes

If $\text{dist}(k, k, k - 1) < 0$ then $G$ has a negative length cycle containing $k$.

Now if $i$ can reach $k$ and $k$ can reach $j$ then $\text{dist}(i, j, k) = -\infty$.

Therefore, recursion below is valid only if $\text{dist}(k, k, k - 1) \geq 0$.

$$\text{dist}(i, j, k) = \min \left\{ \text{dist}(i, j, k - 1), \text{dist}(i, k, k - 1) + \text{dist}(k, j, k - 1) \right\}$$

We can detect this during the algorithm or wait till the end.
Floyd-Warshall Algorithm
for All-Pairs Shortest Paths

for $i = 1$ to $n$ do
  for $j = 1$ to $n$ do
    $\text{dist}(i, j, 0) = \ell(i, j)$ (* $\ell(i, j) = \infty$ if $(i, j) \notin E$, 0 if $i = j$ *)
Floyd-Warshall Algorithm
for All-Pairs Shortest Paths

\[
\text{for } i = 1 \text{ to } n \text{ do} \\
\quad \text{for } j = 1 \text{ to } n \text{ do} \\
\quad \quad \text{dist}(i, j, 0) = \ell(i, j) \quad (\ast \ell(i, j) = \infty \text{ if } (i, j) \notin E, \ 0 \text{ if } i = j \ast) \\
\]

\[
\text{for } k = 1 \text{ to } n \text{ do} \\
\quad \text{for } i = 1 \text{ to } n \text{ do} \\
\quad \quad \text{for } j = 1 \text{ to } n \text{ do} \\
\quad \quad \quad \text{dist}(i, j, k) = \min \left\{ \text{dist}(i, j, k - 1), \text{dist}(i, k, k - 1) + \text{dist}(k, j, k - 1) \right\}
\]
Floyd-Warshall Algorithm
for All-Pairs Shortest Paths

\[
\text{for } i = 1 \text{ to } n \text{ do }
\]
\[
\quad \text{for } j = 1 \text{ to } n \text{ do }
\]
\[
\quad \quad \text{dist}(i, j, 0) = \ell(i, j) (* \ell(i, j) = \infty \text{ if } (i, j) \notin E, 0 \text{ if } i = j *)
\]

\[
\text{for } k = 1 \text{ to } n \text{ do }
\]
\[
\quad \text{for } i = 1 \text{ to } n \text{ do }
\]
\[
\quad \quad \text{for } j = 1 \text{ to } n \text{ do }
\]
\[
\quad \quad \quad \text{dist}(i, j, k) = \min \left\{ \text{dist}(i, j, k - 1), \right. \\
\quad \quad \quad \text{dist}(i, k, k - 1) + \text{dist}(k, j, k - 1) \left\}
\]

\[
\text{for } i = 1 \text{ to } n \text{ do }
\]
\[
\quad \text{if } \left( \text{dist}(i, i, n) < 0 \right) \text{ then }
\]
\[
\quad \quad \text{Output that there is a negative length cycle in } G
\]
Floyd-Warshall Algorithm
for All-Pairs Shortest Paths

for $i = 1$ to $n$ do
  for $j = 1$ to $n$ do
    $dist(i, j, 0) = \ell(i, j)$ (* $\ell(i, j) = \infty$ if $(i, j) \notin E$, 0 if $i = j$ *)

for $k = 1$ to $n$ do
  for $i = 1$ to $n$ do
    for $j = 1$ to $n$ do
      $dist(i, j, k) = \min \left\{ dist(i, j, k - 1), dist(i, k, k - 1) + dist(k, j, k - 1) \right\}$

for $i = 1$ to $n$ do
  if ($dist(i, i, n) < 0$) then
    Output that there is a negative length cycle in $G$

Running Time:
$\Theta(n^3)$, Space: $\Theta(n^3)$. Correctness: via induction and recursive definition.
Floyd-Warshall Algorithm
for All-Pairs Shortest Paths

for $i = 1$ to $n$ do
  for $j = 1$ to $n$ do
    $\text{dist}(i, j, 0) = \ell(i, j)$ (* $\ell(i, j) = \infty$ if $(i, j) \notin E$, 0 if $i = j$ *)

for $k = 1$ to $n$ do
  for $i = 1$ to $n$ do
    for $j = 1$ to $n$ do
      $\text{dist}(i, j, k) = \min \left\{ \text{dist}(i, j, k - 1), \right.$
      $\left. \text{dist}(i, k, k - 1) + \text{dist}(k, j, k - 1) \right\}$

for $i = 1$ to $n$ do
  if ($\text{dist}(i, i, n) < 0$) then
    Output that there is a negative length cycle in $G$

Running Time: $\Theta(n^3)$, Space: $\Theta(n^3)$. 

Floyd-Warshall Algorithm
for All-Pairs Shortest Paths

for $i = 1$ to $n$ do
  for $j = 1$ to $n$ do
    $dist(i, j, 0) = \ell(i, j)$ (* $\ell(i, j) = \infty$ if $(i, j) \notin E$, 0 if $i = j$ *)

for $k = 1$ to $n$ do
  for $i = 1$ to $n$ do
    for $j = 1$ to $n$ do
      $dist(i, j, k) = \min\left\{dist(i, j, k - 1), \right.
      \left. dist(i, k, k - 1) + dist(k, j, k - 1)\right\}$

for $i = 1$ to $n$ do
  if ($dist(i, i, n) < 0$) then
    Output that there is a negative length cycle in $G$

Running Time: $\Theta(n^3)$, Space: $\Theta(n^3)$.
Correctness: via induction and recursive definition
**Floyd-Warshall Algorithm: Finding the Paths**

**Question:** Can we find the paths in addition to the distances?

1. Create a $n \times n$ array `Next` that stores the next vertex on shortest path for each pair of vertices.
2. With array `Next`, for any pair of given vertices $i, j$ can compute a shortest path in $O(n)$ time.
Floyd-Warshall Algorithm

Finding the Paths

for $i = 1$ to $n$ do
    for $j = 1$ to $n$ do
        $dist(i, j, 0) = \ell(i, j)$
        (* $\ell(i, j) = \infty$ if $(i, j)$ not edge, 0 if $i = j$ *)
        $Next(i, j) = -1$

for $k = 1$ to $n$ do
    for $i = 1$ to $n$ do
        for $j = 1$ to $n$ do
            $dist(i, j, k) = dist(i, j, k - 1)$
            if ($dist(i, j, k - 1) > dist(i, k, k - 1) + dist(k, j, k - 1)$) then
                $dist(i, j, k) = dist(i, k, k - 1) + dist(k, j, k - 1)$
                $Next(i, j) = k$

for $i = 1$ to $n$ do
    if ($dist(i, i, n) < 0$) then
        Output that there is a negative length cycle in $G$
Exercise: Given $\textit{Next}$ array and any two vertices $i, j$ describe an $O(n)$ algorithm to find a $i$-$j$ shortest path.
Johnson’s Algorithm

- Bellman-Ford gives $O(nm)$ time algorithm to solve single-source shortest paths when $G$ has no negative lengths.
- To compute APSP running Bellman-Ford $n$ times will give a run time of $O(n^2m)$.
- However, if $G$ has no negative length cycle, after computing shortest paths from one vertex using Bellman-Ford, one can use “reduced” costs to convert the graph into one with non-negative edge lengths. And then one can run $n$ Dijkstra’s on this new graphs to solve APSP. This gives a run time of $O(nm + n^2 \log n)$ for APSP.

See notes for more details.
### Summary of results on shortest paths

#### Single Source Shortest Paths

<table>
<thead>
<tr>
<th>Condition</th>
<th>Algorithm</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>No negative edges</td>
<td>Dijkstra</td>
<td>$O(n \log n + m)$</td>
</tr>
<tr>
<td>Edge lengths can be negative</td>
<td>Bellman Ford</td>
<td>$O(nm)$</td>
</tr>
</tbody>
</table>

#### All Pairs Shortest Paths

<table>
<thead>
<tr>
<th>Condition</th>
<th>Algorithm</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>No negative edges</td>
<td>$n \times$ Dijkstra</td>
<td>$O(n^2 \log n + nm)$</td>
</tr>
<tr>
<td>No negative cycles</td>
<td>$n \times$ Bellman Ford</td>
<td>$O(n^2 m) = O(n^4)$</td>
</tr>
<tr>
<td>No negative cycles</td>
<td>BF + $n \times$ Dijkstra</td>
<td>$O(nm + n^2 \log n)$</td>
</tr>
<tr>
<td>No negative cycles</td>
<td>Floyd-Warshall</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td>Unweighted</td>
<td>Matrix multiplication</td>
<td>$O(n^{2.38}), O(n^{2.58})$</td>
</tr>
</tbody>
</table>