Incremental Single Source Shortest Paths in Sparse Digraphs

Shiri Chechik          Tianyi Zhang
Tel Aviv University          Tsinghua University
Partially dynamic SSSP

A weighted di-graph $G = (V, E)$ undergoes edge updates

- Decremental: all updates are deletions
- Incremental: all updates are insertions

**Goal.** Answer queries of distances from a source vertex $s \in V$

**Cost.** Total update time
Partially dynamic SSSP

A weighted digraph $G = (V, E)$ undergoes edge updates

- Decremental: all updates are deletions
- Incremental: all updates are insertions

**Goal.** Answer queries of distances from a source vertex $s \in V$

**Cost.** Total update time

<table>
<thead>
<tr>
<th>Edge insertions</th>
<th>Insert (1, 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Diagram 1]</td>
<td>![Diagram 2]</td>
</tr>
</tbody>
</table>

**Diagram 1:**
- **Source:** $s$
- **Vertices:** 1, 2, 3
- **Edges:** $s \rightarrow 1 \rightarrow 3$

**Diagram 2:**
- **Source:** $s$
- **Vertices:** 1, 2, 3
- **Edges:** $s \rightarrow 1 \rightarrow 3$

**Diagram 3:**
- **Source:** $s$
- **Vertices:** 1, 2, 3
- **Edges:** $s \rightarrow 1 \rightarrow 3$
Partially dynamic SSSP

A weighted directed graph $G = (V, E)$ undergoes edge updates

- **Decremental**: all updates are deletions
- **Incremental**: all updates are insertions

**Goal.** Answer queries of distances from a source vertex $s \in V$

**Cost.** Total update time

<table>
<thead>
<tr>
<th>Edge insertions</th>
<th>Insert $(1, 3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Diagram" /></td>
<td><img src="image.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Partially dynamic SSSP

A weighted directed graph $G = (V, E)$ undergoes edge updates

- Decremental: all updates are deletions
- Incremental: all updates are insertions

**Goal.** Answer queries of distances from a source vertex $s \in V$

**Cost.** Total update time
A weighted directed graph $G = (V, E)$ undergoes edge updates.

- Decremental: all updates are deletions.
- Incremental: all updates are insertions.

**Goal.** Answer queries of distances from a source vertex $s \in V$.

**Cost.** Total update time.

<table>
<thead>
<tr>
<th>Edge insertions</th>
<th>Insert $(1, 3)$</th>
<th>Insert $(s, 2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Picture</strong></td>
<td><img src="#" alt="Diagram" /></td>
<td><img src="#" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Partially dynamic SSSP

A weighted digraph $G = (V, E)$ undergoes edge updates

- Decremental: all updates are deletions
- Incremental: all updates are insertions

**Goal.** Answer queries of distances from a source vertex $s \in V$

**Cost.** Total update time

<table>
<thead>
<tr>
<th>Edge insertions</th>
<th>Insert $(1, 3)$</th>
<th>Insert $(s, 2)$</th>
<th>Insert $(s, 3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>Picture</td>
<td><img src="image4" alt="Diagram" /></td>
<td><img src="image5" alt="Diagram" /></td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Partially dynamic SSSP

A weighted directed graph $G = (V, E)$ undergoes edge updates:

- **Decremental**: all updates are deletions
- **Incremental**: all updates are insertions

**Goal.** Answer queries of distances from a source vertex $s \in V$

**Cost.** Total update time

<table>
<thead>
<tr>
<th>Edge insertions</th>
<th>Insert (1, 3)</th>
<th>Insert (s, 2)</th>
<th>Insert (s, 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Oblivious vs Adaptive

- Oblivious: edge updates are fixed at the beginning
- Adaptive: future edge updates may depend on queries
## History

Exact distances in partially dynamic SSSP (either decr or incr)

<table>
<thead>
<tr>
<th></th>
<th>Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classic</td>
<td>$O(mn)$ ($W=1$)</td>
<td>[ES’81]</td>
</tr>
<tr>
<td>APSP-hard</td>
<td>$\tilde{\Omega}(mn)$</td>
<td>[RZ’04]</td>
</tr>
<tr>
<td>k-cycle-hard</td>
<td>$\tilde{\Omega}(m^2)$</td>
<td>[PWW’20]</td>
</tr>
<tr>
<td>OMv3-hard</td>
<td>$\tilde{\Omega}(m^{(\omega+1)/2})$</td>
<td>[PWW’20]</td>
</tr>
</tbody>
</table>

To break $O(mn)$, should consider $\text{(1 + } \epsilon\text{)-approximation}$

Assume digraph $G$ has $n$ vertices and $m$ edges ever appear in the graph

$\tilde{O}(\cdot)$ hides $\text{poly-log(nW)}$ factors, where $W$ is the largest integer weight
History

To break $O(mn)$, should consider $(1 + \epsilon)$-approximation

Decr-SSSP is a subroutine in many static algorithms, e.g. max-flow, sparsest cut

<table>
<thead>
<tr>
<th>Algorithm Type</th>
<th>Time Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best oblivious</td>
<td>$\tilde{O}(n^2)$, $\tilde{O}(mn^{2/3})$</td>
<td>[BPW’20]</td>
</tr>
<tr>
<td>Best adaptive</td>
<td>$\tilde{O}(m^{3/4}n^{5/4})$</td>
<td>[PW’20]</td>
</tr>
<tr>
<td>Best deterministic</td>
<td>$n^{8/3 + o(1)}$</td>
<td>[BPS’20]</td>
</tr>
</tbody>
</table>

Incr-SSSP is a natural sister problem of Decr-SSSP

<table>
<thead>
<tr>
<th>Algorithm Type</th>
<th>Time Complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best oblivious</td>
<td>$\tilde{O}(mn^{0.9})$</td>
<td>[HKN’14]</td>
</tr>
<tr>
<td>Best deterministic</td>
<td>$\tilde{O}(n^2)$</td>
<td>[PWW’20]</td>
</tr>
</tbody>
</table>

Assume digraph $G$ has $n$ vertices and $m$ edges ever appear in the graph $\tilde{O}(\cdot)$ hides $\text{poly-log}(nW)$ factors, where $W$ is the largest integer weight.
## Results

<table>
<thead>
<tr>
<th>Reference</th>
<th>Total update time</th>
<th>det / obl / ada</th>
</tr>
</thead>
<tbody>
<tr>
<td>[HKN’14]</td>
<td>$\tilde{O}(mn^{0.9})$</td>
<td>oblivious</td>
</tr>
<tr>
<td>[PWW’20]</td>
<td>$\tilde{O}(n^2)$</td>
<td>deterministic</td>
</tr>
<tr>
<td>New</td>
<td>$\tilde{O}(m^{5/3})$</td>
<td>deterministic</td>
</tr>
<tr>
<td>New</td>
<td>$\tilde{O}(mn^{1/2} + m^{1.4})$</td>
<td>adaptive</td>
</tr>
</tbody>
</table>

Our algorithm is the **sub-quadratic** when $m = o(n^{1.42})$
A deterministic algorithm
A basic procedure

- Similar to Dijkstra’s algorithm, but in a local & lazy manner

**Propagate** ($Q$):

while ($Q \neq \emptyset$)

$u \leftarrow$ dequeue $Q$

for each $(u, v) \in E$

if $d(v) - d(u) - \omega(u, v) \geq D$ or $v \in Q$

$d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$

$Q \leftarrow Q \cup \{v\}$

**Dijkstra:**

initialize dist labels $d(\cdot)$ for each $v \in V$

initialize a queue $Q \leftarrow V$

while ($Q \neq \emptyset$)

$u \leftarrow$ dequeue $Q$

For each $(u, v) \in E$

$d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$
A basic procedure

- Similar to Dijkstra’s algorithm, but in a local & lazy manner

Maintain dist labels $d(\cdot)$ for each $v \in V$

**Propagate**($Q$):
- while($Q \neq \emptyset$)
  - $u \leftarrow$ dequeue $Q$
  - for each $(u, v) \in E$
    - if $d(v) - d(u) - \omega(u, v) \geq D$ or $v \in Q$
    - $d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$
  - $Q \leftarrow Q \cup \{v\}$

An example of **Propagate**
Parameters: $D = 10$, $Q = \{w\}$

![Graph example](image)
A basic procedure

- Similar to Dijkstra’s algorithm, but in a **local & lazy** manner

```
maintain dist labels $d(\cdot)$ for each $v \in V$

**Propagate** ($Q$):
while ($Q \neq \emptyset$)
    $u \leftarrow$ dequeue $Q$
    for each $(u, v) \in E$
        if $d(v) - d(u) - \omega(u, v) \geq D$ or $v \in Q$
            $d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$
        $Q \leftarrow Q \cup \{v\}$
```

An example of **Propagate**
Parameters: $D = 10$, $Q = \{w\}$
A basic procedure

- Similar to Dijkstra’s algorithm, but in a **local & lazy** manner

**Propagate** ($Q$):

while ($Q \neq \emptyset$)

  $u \leftarrow$ dequeue $Q$

  for each $(u, v) \in E$

    if $d(v) - d(u) - \omega(u, v) \geq D$ or $v \in Q$

    $d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$

  $Q \leftarrow Q \cup \{v\}$

---

**Parameters:**

- $D = 10$
- $Q = \{w\}$

**Example:**

- $d(s) = 0$
- $d(w) = 20$
- $d(x) = 21$
- $d(y) = 25$
- $d(z) = 33$
A basic procedure

- Similar to Dijkstra’s algorithm, but in a **local & lazy** manner

Maintain dist labels $d(\cdot)$ for each $v \in V$

**Propagate**($Q$):
while($Q \neq \emptyset$)
  $u \leftarrow$ dequeue $Q$
  for each $(u, v) \in E$
    if $d(v) - d(u) - \omega(u, v) \geq D$ or $v \in Q$
      $d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$
    $Q \leftarrow Q \cup \{v\}$

An example of **Propagate**
Parameters: $D = 10$, $Q = \{w\}$
A basic procedure

- Similar to Dijkstra’s algorithm, but in a local & lazy manner

Maintain dist labels \( d(\cdot) \) for each \( v \in V \)

**Propagate** \( Q \):

while \( Q \neq \emptyset \)

\( u \leftarrow \text{dequeue } Q \)

for each \((u, v) \in E\)

if \( d(v) - d(u) - \omega(u, v) \geq D \) or \( v \in Q \)

\( d(v) \leftarrow \min \{d(u) + \omega(u, v), d(v)\} \)

\( Q \leftarrow Q \cup \{v\} \)

An example of **Propagate**

Parameters: \( D = 10 \), \( Q = \{w\} \)

Running time = **sum of degrees in the queue**

Each \( d(u) \) decreases by \( D \) if \( u \) was added to queue
A deterministic algorithm

For every $B$ insertions: $e_1, e_2, \ldots, e_B$

1. Run **Dijkstra** to refresh all distance labels $d(\cdot)$ at the beginning

2. For each insertion $e_i = (u, v)$, if $d(v) - d(u) - \omega(u, v) \geq D$, then
   
   - update $d(v) \leftarrow d(u) + \omega(u, v)$
   
   - run **Propagate**($\{v\}$)
A deterministic algorithm

For every $B$ insertions: $e_1, e_2, \cdots, e_B$

1. Run **Dijkstra** to refresh all distance labels $d(\cdot)$ at the beginning

2. For each insertion $e_i = (u, v)$, if $d(v) - d(u) - \omega(u, v) \geq D$, then
   - update $d(v) \leftarrow d(u) + \omega(u, v)$
   - run **Propagate**( $\{v\}$ )
A deterministic algorithm

For every $B$ insertions: $e_1, e_2, \ldots, e_B$

1. Run **Dijkstra** to refresh all distance labels $d(\cdot)$ at the beginning

2. For each insertion $e_i = (u, v)$, if $d(v) - d(u) - \omega(u, v) \geq D$, then
   
   - update $d(v) \leftarrow d(u) + \omega(u, v)$
   
   - run **Propagate**($\{v\}$)
A deterministic algorithm

For every $B$ insertions: $e_1, e_2, \cdots, e_B$

1. Run **Dijkstra** to refresh all distance labels $d(\cdot)$ at the beginning

2. For each insertion $e_i = (u, v)$, if $d(v) - d(u) - \omega(u, v) \geq D$, then

   - update $d(v) \leftarrow d(u) + \omega(u, v)$
   - run **Propagate**($\{v\}$)
A deterministic algorithm

Running time analysis:

- Focus on $\text{dist}(s, v)$ in $[L, 2L]$, so there are only $\log(nW)$ scales

- Total number of $\text{Dijkstra}$ calls is $\leq m/B$

- $d(v)$ drops by $D$ each time we scan $\text{adj}(v)$ during $\text{Propagate}$

  Total cost of $\text{Propagate}$ is at most $\sum_v L/D \cdot \deg(v) = Lm/D$

- Total update time $\approx m^2/B + Lm/D$

- How to choose $B$?
A wrong guess

- Total update time $\approx m^2/B + Lm/D$

- How to choose $B$?

Example:

1. start with $\text{dist}(s, v) = 2L = d(v)$

2.

3.
A wrong guess

- Total update time \( \approx \frac{m^2}{B} + \frac{Lm}{D} \)

- How to choose B?

**Example:**

1. start with \( \text{dist}(s, v) = 2L = d(v) \)

2. insert \( 10\epsilon L/D \) shortcuts along the path

3.
A wrong guess

- Total update time $\approx \frac{m^2}{B} + \frac{Lm}{D}$

- How to choose $B$?

Example:

1. start with $\text{dist}(s, v) = 2L = d(v)$

2. insert $10\epsilon L/D$ shortcuts along the path

3. $\text{dist}(s, v)$ gets below $(2 - 2\epsilon)L$, so $d(v)$ becomes a bad approximation
A wrong guess

- Total update time $\approx \frac{m^2}{B} + \frac{Lm}{D}$

- How to choose $B$? Choose $B = \epsilon L/D$?

Example:

1. start with $\text{dist}(s, v) = 2L = d(v)$

2. insert $10\epsilon L/D$ shortcuts along the path

3. $\text{dist}(s, v)$ gets below $(2 - 2\epsilon)L$, so $d(v)$ becomes a bad approximation

---

cannot take $10\epsilon L/D$ insertions
A counter example

• Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

- Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

- Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]

```plaintext
Before insertions:
d(tail) - d(head) = B

\[
\begin{align*}
5L & \quad 5L+1 & \quad 5L+B-2 & \quad 5L+B-1 & \quad x+0.1BD \\
\text{head} & \quad \rightarrow & \quad \rightarrow & \quad \rightarrow & \quad \rightarrow & \quad \rightarrow \\
\downarrow & \quad \downarrow & \quad \downarrow & \quad \downarrow & \quad \downarrow & \quad \downarrow \\
5 & 6 & 7 & 8 & 9 & 10 \\
\end{align*}
\]

\text{edge weight}:
\[ x-2 \quad x+0.1D-2 \quad x+0.1(B-2)D-2 \quad x+0.1(B-1)D-2 \quad x+0.1BD-2 \]

\text{1}
```
A counter example

- Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

- Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

- Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

• Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

• Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

- Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

• Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

• Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]
A counter example

- Construct the following gadget

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B \]

After insertions:
\[ d(\text{tail}) - d(\text{head}) = 0.1BD \]
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example

A gadget
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example
A counter example

- Use gadgets to construct a counter example

If we have $B$ gadgets and $2B$ insertions

head

.............

1 1 1 1

1

1

tail
A counter example

- Use gadgets to construct a counter example

If we have $B$ gadgets and $2B$ insertions

Before insertions:
\[ d(\text{tail}) - d(\text{head}) = B^2 \]

After insertions:
\[ d(\text{tail}) - d(\text{head}) = 0.1B^2D \]
A counter example

- Use gadgets to construct a counter example

If we have $B$ gadgets and $2B$ insertions

Before insertions:
$d(tail) - d(head) = B^2$

After insertions:
$d(tail) - d(head) = 0.1B^2D$

Error $= 0.1B^2D - B^2 \leq \epsilon L$

So $B = O(\sqrt{L/D})$

So update time $= \tilde{\Omega}(m^{5/3})$

It turns out to be tight
Total update time $= \tilde{O}(m^{5/3})$
A randomized algorithm

with $\tilde{O}(m^{1.5})$ total update time
Key idea

- Assume stretch \( d(u) - \text{dist}(s, u) > 10\epsilon L \) at some point

- Then, stretch \( d(w) - d(v) - \text{dist}(v, w) > D \) for many subpaths from \( v \) to \( w \)
Key idea

• Assume stretch $d(u) - \text{dist}(s, u) > 10\epsilon L$ at some point

• Then, stretch $d(w) - d(v) - \text{dist}(v, w) > D$

for many subpaths from v to w
Key idea

- Assume stretch $d(u) - \text{dist}(s, u) > 10\epsilon L$ at some point

- Then, stretch $d(w) - d(v) - \text{dist}(v, w) > D$ for many subpaths from v to w
Key idea

- Assume stretch $d(u) - \text{dist}(s, u) > 10\epsilon L$ at some point
- Then, stretch $d(w) - d(v) - \text{dist}(v, w) > D$
  for many subpaths from $v$ to $w$
Key idea

• Look at the interval $[0, 3L]$
Key idea

- Look at the interval \([0, 3L]\)
- Randomly sample an interval \([x, x + D/\epsilon]\)
Key idea

- Look at the interval $[0, 3L]$
- Randomly sample an interval $[x, x + D/\epsilon]$,
- Call $\text{Propagate}([w \mid d(w) \in [x, x + D/\epsilon]])$
Key idea

- Look at the interval \([0, 3L]\)
- Randomly sample an interval \([x, x + D/\epsilon]\)
- Call Propagate(\(\{w \mid d(w) \in [x, x + D/\epsilon]\}\))

\[
d(s) = 0 \quad x \quad x + D/\epsilon \quad d(u) = \Theta(L)
\]

\[d(w) \text{ decr by } D\]

all labels \(d(\ldots)\) decrease by more than \(D\)
**Key idea**

- Look at the interval $[0, 3L]$
- Randomly sample an interval $[x, x + D/\epsilon]$
- Call $\text{Propagate}(\{w \mid d(w) \in [x, x + D/\epsilon]\})$

$d(w)$ decr by $D$

all labels $d(\ldots)$ decrease by more than $D$

distances labels

$d(s) = 0$ \hspace{2cm} $x$ \hspace{2cm} $x + D/\epsilon$ \hspace{2cm} $d(u) = \Theta(L)$
Key idea

- Look at the interval \([0, 3L]\)
- Randomly sample an interval \([x, x + D/\epsilon]\)
- Call \textbf{Propagate}(\{w \mid d(w) \in [x, x + D/\epsilon]\})

\[
d(s) = 0 \quad x \quad x + D/\epsilon \quad d(u) = \Theta(L)
\]
Key idea

- Look at the interval $[0, 3L]$
- Randomly sample an interval $[x, x + D/\epsilon]$
- Call $\text{Propagate}\left(\{w \mid d(w) \in [x, x + D/\epsilon]\}\right)$

$d(s) = 0 \quad x \quad x + D/\epsilon \quad d(u) = \Theta(L)$

Distances labels

$d(w)$ decr by D

all labels $d(\ldots)$ decrease by more than D

$d(u)$ decr by $10\epsilon L$
Main algorithm

Pseudo-code

maintain dist labels $d(\cdot)$ for each $v \in V$

**Insert**($u, v$):
  If $d(v) - d(u) - \omega(u, v) \geq D$
    $d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$
    call **Propagate**( \{v\} )
  uniformly sample $x \in [0, 2L]$
  call **Propagate**( \{w | d(w) \in [x, x + D/\epsilon]\} )

Running time of
**Propagate**( \{w | d(w) \in [x, x + D/\epsilon]\} )
Main algorithm

Pseudo-code

maintain dist labels $d(\cdot)$ for each $v \in V$

$\textbf{Insert}(u, v)$:
If $d(v) - d(u) - \omega(u, v) \geq D$

$d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$
call $\textbf{Propagate}(\{v\})$

uniformly sample $x \in [0, 2L]$
call $\textbf{Propagate}(\{w | d(w) \in [x, x + D/e]\})$

Running time of $\textbf{Propagate}(\{w | d(w) \in [x, x + D/e]\})$

Scanning adjacency lists within
{$w | d(w) \in [x, x + D/e]$}

Time cost = $mD/eL$ each call
Main algorithm

Pseudo-code

maintain dist labels $d(\cdot)$ for each $v \in V$

**Insert**($u, v$):

If $d(v) - d(u) - \omega(u, v) \geq D$

$$d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$$

call **Propagate**( $\{v\}$ )

uniformly sample $x \in [0, 2L]$

call **Propagate**( $\{w \mid d(w) \in [x, x + D/\epsilon]\}$ )

Running time of **Propagate**( $\{w \mid d(w) \in [x, x + D/\epsilon]\}$ )

- Scanning adjacency lists within $\{w \mid d(w) \in [x, x + D/\epsilon]\}$
- Time cost = $mD/\epsilon L$ each call
- Propagation for **decr-by-D vertices**
- Total Time cost = $mL/D$
Main algorithm

Pseudo-code

Maintain dist labels $d(\cdot)$ for each $v \in V$

**Insert**($u, v$):
- If $d(v) - d(u) - \omega(u, v) \geq D$
  - $d(v) \leftarrow \min\{d(u) + \omega(u, v), d(v)\}$
  - Call **Propagate**($\{v\}$)
- Uniformly sample $x \in [0, 2L]$
- Call **Propagate**($\{w \mid d(w) \in [x, x + D/\epsilon]\}$)

**Running time of**

**Propagate**($\{w \mid d(w) \in [x, x + D/\epsilon]\}$)

Scanning adjacency lists within $\{w \mid d(w) \in [x, x + D/\epsilon]\}$

Time cost = $mD/\epsilon L$ each call

Propagation for **decr-by-D vertices**

Total Time cost = $mL/D$

Total update time = $m^2D/\epsilon L + mL/D = m^{1.5}$
Proof of correctness

- Main difficulty: propagation might stop early
Proof of correctness

- Main difficulty: propagation might stop early
Proof of correctness

• Main difficulty: propagation might **stop early**
Proof of correctness

- Main difficulty: propagation might stop early
Proof of correctness

- Main difficulty: propagation might stop early
Proof of correctness

- Main difficulty: propagation might stop early
Proof of correctness

- Main difficulty: propagation might stop early
Proof of correctness

- Main difficulty: propagation might stop early
Proof of correctness

- Main difficulty: propagation might stop early

\[ d(w) - \text{dist}(w) \geq 10\epsilon L \]

\[ \text{dist}(w) \leq 2L \]
Proof of correctness

- Main difficulty: propagation might stop early

\[ d(w) - \text{dist}(w) \]

\[ \text{dist}(w) \leq 2L \]

\[ \geq 10\epsilon L \]

Propagation stops here!
Proof of correctness

- Where does Propagation succeed?
Proof of correctness

- Where does **Propagation** succeed?

\[ d(w) - \text{dist}(w) \leq 2L \]
Proof of correctness

- Where does Propagation succeed?

\[ d(w) - \text{dist}(w) \leq 2L \]
Proof of correctness

- Where does Propagation succeed?

$$d(w) - \text{dist}(w)$$

$$(k+1)D$$
$$kD$$
$$3D$$
$$2D$$
$$D$$

$$(\text{dist}(w) \leq 2L)$$
Proof of correctness

• Where does Propagation succeed?

\[ \begin{align*}
    d(w) - \text{dist}(w) \\
    \text{(k+1)D} \\
    \text{kD} \\
    \vdots \\
    \text{3D} \\
    \text{2D} \\
    \text{D}
\end{align*} \]

\[ \text{dist}(w) \leq 2L \]
Proof of correctness

- **Propagation** could succeed at these places

\[
d(w) - \text{dist}(w) \leq 2L
\]
Proof of correctness

- **Propagation** could succeed at these places

\[ d(w) - \text{dist}(w) \leq 2L \]
Proof of correctness

- **Propagation** could succeed at **these places**

\[ d(w) - \text{dist}(w) \]

\[ (i+1) \times D \leq \text{dist}(w) \leq 2L \]
Proof of correctness

- **Propagation** could succeed at these places

\[ d(w) - \text{dist}(w) \leq 2L \]
Proof of correctness

- **Propagation** could succeed at these places

\[ d(w) - \text{dist}(w) \leq 2L \]

\[ \text{all } d(\ldots) \text{ decreas by } D \]
Proof of correctness

- **Propagation** could succeed at these places

\[ d(w) - \text{dist}(w) \leq 2L \]
Proof of correctness

- **Propagation** could succeed at these places

\[ d(w) - \text{dist}(w) \leq 2L \]

All \( d(\ldots) \) decr by \( D \)

Would reach \( u \) as it never goes below \((i+1)*D\)
Thank you!