Single Source Shortest Paths with Positive Weights

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
In this lecture, we will revisit the **single source shortest path** (SSSP) problem. Recall that we have already learned that BFS solves the problem efficiently when all the edges have the **same** weight. Today, we will see how to solve the problem in a more general situation where the edges can have arbitrary positive weights.
Weighted Graphs

Let \( G = (V, E) \) be a directed graph. Let \( w \) be a function that maps each edge in \( E \) to a positive integer value. Specifically, for each \( e \in E \), \( w(e) \) is a positive integer value, which we call the weight of \( e \).

A directed weighted graph is defined as the pair \((G, w)\).
The integer on each edge indicates its weight. For example, $w(d, g) = 1$, $w(g, f) = 2$, and $w(c, e) = 10$. 
Consider a directed weighted graph defined by a directed graph \( G = (V, E) \) and function \( w \).

Consider a path in \( G \): \((v_1, v_2), (v_2, v_3), \ldots, (v_\ell, v_{\ell+1})\), for some integer \( \ell \geq 1 \). We define the length of the path as

\[
\sum_{i=1}^{\ell} w(v_i, v_{i+1}).
\]

Recall that we may also denote the path as \( v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_{\ell+1} \).

Given two vertices \( u, v \in V \), a shortest path from \( u \) to \( v \) is a path from \( u \) to \( v \) that has the minimum length among all the paths from \( u \) to \( v \).

If \( v \) is unreachable from \( u \), then the shortest path distance from \( u \) to \( v \) is \( \infty \).
Example

The path $c \rightarrow e$ has length 10.

The path $c \rightarrow d \rightarrow g \rightarrow f \rightarrow e$ has length 6.

The first path is a shortest path from $c$ to $e$. 
**Single Source Shortest Path (SSSP) with Positive Weights**

Let \((G, w)\) with \(G = (V, E)\) be a directed weighted graph, where \(w\) maps every edge of \(E\) to a positive value.

Given a vertex \(s\) in \(V\), the goal of the **SSSP problem** is to find, for every other vertex \(t \in V \setminus \{s\}\), a shortest path from \(s\) to \(t\), unless \(t\) is unreachable from \(s\).
Next, we will first explain the Dijkstra’s algorithm for solving the SSSP problem, which outputs a **shortest path tree** that encodes all the shortest paths from the source vertex $s$. 
The Edge Relaxation Idea

For every vertex $v \in V$, we will — at all times — maintain a value $dist(v)$ that represents the length of the shortest path from $s$ to $v$ found so far.

At the end of the algorithm, we will ensure that every $dist(v)$ equals the precise shortest path distance from $s$ to $v$.

A core operation in our algorithm is called edge relaxation:

- Given an edge $(u, v)$, we relax it as follows:
  - If $dist(v) < dist(u) + w(u, v)$, do nothing;
  - Otherwise, reduce $dist(v)$ to $dist(u) + w(u, v)$. 
Dijkstra’s Algorithm

1. Set $\text{parent}(v) = \text{nil}$ for all vertices $v \in V$
2. Set $\text{dist}(s) = 0$, and $\text{dist}(v) = \infty$ for all other vertices $v \in V$
3. Set $S = V$
4. Repeat the following until $S$ is empty:
   5.1 Remove from $S$ the vertex $u$ with the smallest $\text{dist}(u)$.
      /* next we relax all the outgoing edges of $u$ */
   5.2 for every outgoing edge $(u, v)$ of $u$
      5.2.1 if $\text{dist}(v) > \text{dist}(u) + w(u, v)$ then
          set $\text{dist}(v) = \text{dist}(u) + w(u, v)$, and $\text{parent}(v) = u$
Example

Suppose that the source vertex is $c$.

\[
S = \{a, b, c, d, e, f, g, h, i\}.
\]
Example

Relax the out-going edges of $c$ (because $\text{dist}(c)$ is the smallest in $S$):

$$\begin{array}{|c|c|c|}
\hline
\text{vertex } v & \text{dist}(v) & \text{parent}(v) \\
\hline
a & \infty & \text{nil} \\
b & \infty & \text{nil} \\
c & 0 & \text{nil} \\
d & 2 & c \\
e & 10 & c \\
f & \infty & \text{nil} \\
g & \infty & \text{nil} \\
h & \infty & \text{nil} \\
i & \infty & \text{nil} \\
\hline
\end{array}$$

$S = \{a, b, d, e, f, g, h, i\}$. Note that $c$ has been removed!
Example

Relax the out-going edges of $d$ (because $\text{dist}(d)$ is the smallest in $S$):

$$
S = \{a, b, e, f, g, h, i\}.
$$
Example

Relax the out-going edges of $g$:

\[
S = \{a, b, e, f, h, i\}.
\]
Example

Relax the out-going edges of $i$:

\[ S = \{a, b, e, f, h\}. \]

\[ S = \{a, b, e, f, h\}. \]
Example

Relax the out-going edges of $f$:

$$
\begin{array}{cccc}
\text{vertex } v & \text{dist}(v) & \text{parent}(v) \\
 a & 8 & d \\
b & \infty & \text{nil} \\
c & 0 & \text{nil} \\
d & 2 & c \\
e & 6 & f \\
f & 5 & g \\
g & 3 & d \\
h & \infty & \text{nil} \\
i & 4 & g \\
\end{array}
$$

$S = \{a, b, e, h\}$. 
Example

Relax the out-going edges of $e$:

$S = \{a, b, h\}$.
Example

Relax the out-going edges of $a$:

$S = \{b, h\}$. 

<table>
<thead>
<tr>
<th>vertex $v$</th>
<th>$dist(v)$</th>
<th>$parent(v)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>8</td>
<td>$d$</td>
</tr>
<tr>
<td>$b$</td>
<td>9</td>
<td>$a$</td>
</tr>
<tr>
<td>$c$</td>
<td>0</td>
<td>$\text{nil}$</td>
</tr>
<tr>
<td>$d$</td>
<td>2</td>
<td>$c$</td>
</tr>
<tr>
<td>$e$</td>
<td>6</td>
<td>$f$</td>
</tr>
<tr>
<td>$f$</td>
<td>5</td>
<td>$g$</td>
</tr>
<tr>
<td>$g$</td>
<td>3</td>
<td>$d$</td>
</tr>
<tr>
<td>$h$</td>
<td>$\infty$</td>
<td>$\text{nil}$</td>
</tr>
<tr>
<td>$i$</td>
<td>4</td>
<td>$g$</td>
</tr>
</tbody>
</table>
Example

Relax the out-going edges of $b$:

\[ S = \{ h \}. \]
Example

Relax the out-going edges of \( h \):

\[
\begin{array}{c|c|c}
\text{vertex } v & \text{dist}(v) & \text{parent}(v) \\
\hline
a & 8 & d \\
b & 9 & a \\
c & 0 & \text{nil} \\
d & 2 & c \\
e & 6 & f \\
f & 5 & g \\
g & 3 & d \\
h & \infty & \text{nil} \\
i & 4 & g \\
\end{array}
\]

\( S = \{ \} \).

All the shortest path distances are now final.
Constructing the Shortest Path Tree

For every vertex \( v \), if \( u = \text{parent}(v) \) is not nil, then make \( v \) a child of \( u \).

**Example**
Running Time

It will be left as an exercise for you to implement Dijkstra’s algorithm in $O(|V| + |E| \cdot \log |V|)$ time. You have already learned all the data structures for this purpose. Now it is time to practice using them.
Correctness

**Lemma:** When vertex $v$ is removed from $S$, $\text{dist}(v)$ equals precisely the shortest path distance — denoted as $\text{spdist}(v)$ — from $s$ to $v$.

The correctness of Dijkstra’s algorithm follows from the lemma.
Correctness

We will prove the claim by induction on the sequence of vertices removed.

- Base case:
  This is obviously true for the first vertex removed, which is $s$ itself with $dist(s) = 0$.

- Inductive:
  Assume the claim is true with respect to all the vertices already removed. Let $v$ be the next node to be removed. We need to prove $dist(v) = spdist(v)$.
Correctness

Consider an arbitrary shortest path $\pi$ from $s$ to $v$. Let $u$ be the vertex right before $v$ on $\pi$.

Claim: $u$ must have been removed from $S$.

Our target lemma follows from the above claim because, by our inductive assumption, $\text{dist}(u) = \text{spdist}(u)$ when $u$ was removed. Then, the algorithm relaxed the edge $(u, v)$, which must have set $\text{dist}(v) = \text{spdist}(u) + w(u, v) = \text{spdist}(v)$. 
Stronger claim: All the nodes on $\pi$ from $s$ to $u$ must have been removed.
Correctness

We will prove the stronger claim by contradiction.

Suppose the statement is not true. When \( v \) is to be removed from \( S \), another vertex on \( \pi \) — let it be \( v' \) — still remains in \( S \). Define \( p \) as the vertex right before \( v' \) on \( \pi \).
Correctness

By the inductive assumption, $\text{dist}(p) = \text{spdist}(p)$ when $p$ was removed. Hence, after relaxing the edge $(p, v')$, we have $\text{dist}(v') = \text{spdist}(p) + w(p, v') = \text{spdist}(v')$.

But this means $\text{dist}(v') = \text{spdist}(v') < \text{spdist}(v) \leq \text{dist}(v)$!

Hence, $v'$ should be the next vertex to be removed from $S$, contradicting the definition of $v$. 

---

Yufei Tao

Single Source Shortest Paths with Positive Weights