Single Source Shortest Paths with Positive Weights

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

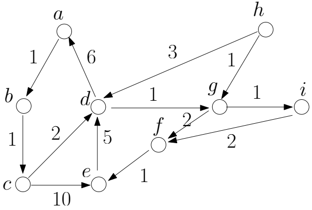
In this lecture, we will revisit the single source shortest path (SSSP) problem. Recall that we have already learned that the BFS algorithm solves the problem efficiently when all the edges have the same weight. Today we will see how to solve the problem in the more general situation where the edges 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).

Example



The integer on each edge indicates its weight. For example, $w(d, g) = 1$, $w(g, f) = 2$, and $w(c, e) = 10$.

Shortest Path

Consider a directed weighted graph defined by a directed graph $G = (V, E)$ and function w.

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

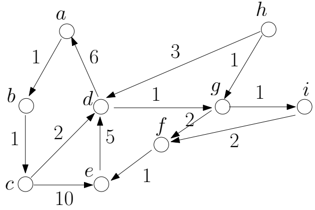
$$\sum_{i=1}^{\ell} w(v_i, v_{i+1}).$$

Recall that we may also denote the path as $v_1 \to v_2 \to ... \to v_{\ell+1}$.

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 ∞.

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.

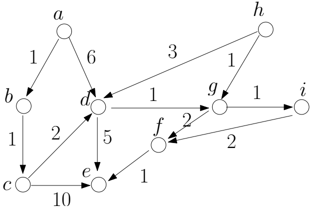
Given a vertex s in V, the goal of the SSSP problem is to find, for every other vertex $t \in V \setminus \{s\}$, a shortest path from s to t, unless t is unreachable from s.

A Subsequence Property

Lemma: If $v_1 \to v_2 \to ... \to v_{\ell+1}$ is a shortest path from v_1 to $v_{\ell+1}$, then for every i, j satisfying $1 \le i < j \le \ell + 1$, $v_i \to v_{i+1} \to ... \to v_j$ is a shortest path from v_i to v_j.

Proof: Suppose that this is not true. Then, we can find a shorter path to go from v_i to v_j. Using this path to replace the original path from v_i to v_j yields a shorter path from v_1 to $v_{\ell+1}$, which contradicts the fact that $v_1 \to v_2 \to ... \to v_{\ell+1}$ is a shortest path. $\square$

Example



Since $c \rightarrow d \rightarrow g \rightarrow f \rightarrow e$ is a shortest path, we know that any subsequence of this path is also a shortest path. For example, $c \rightarrow d \rightarrow g \rightarrow f$ must be a shortest path from c to f.

Next, we will first explain the Dijkstra's algorithm for solving the SSSP problem.

Utilizing the subsequence property, our algorithm will output 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 $parent(v) = \text{nil}$ for all vertices $v \in V$
2. Set $dist(s) = 0$, and $dist(v) = \infty$ for all other vertices $v \in V$
3. Set $S = V$
4. Repeat the following until S is empty:

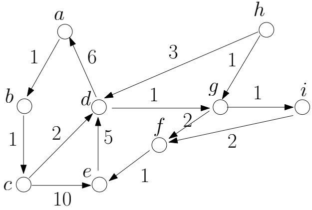
    5.1 Remove from S the vertex u with the smallest $dist(u)$.
    /* next we relax all the outgoing edges of u */

    5.2 for every outgoing edge (u, v) of u

    5.2.1 if $dist(v) > dist(u) + w(u, v)$ then
    set $dist(v) = dist(u) + w(u, v)$, and $parent(v) = u$

Example

Suppose that the source vertex is c.

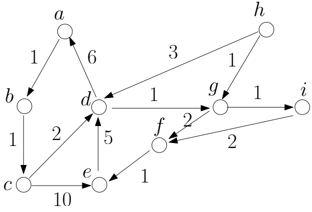


vertex v	$dist(v)$	$parent(v)$
a	∞	nil
b	∞	nil
c	0	nil
d	∞	nil
e	∞	nil
f	∞	nil
g	∞	nil
h	∞	nil
i	∞	nil

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

Example

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

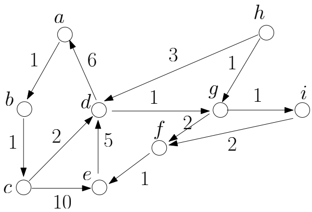


vertex v	$dist(v)$	$parent(v)$
a	∞	nil
b	∞	nil
c	0	nil
d	2	c
e	10	c
f	∞	nil
g	∞	nil
h	∞	nil
i	∞	nil

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

Example

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

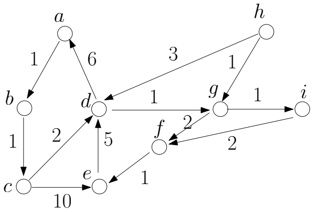


vertex v	$dist(v)$	$parent(v)$
a	8	d
b	∞	nil
c	0	nil
d	2	c
e	10	c
f	∞	nil
g	3	d
h	∞	nil
i	∞	nil

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

Example

Relax the out-going edges of g:

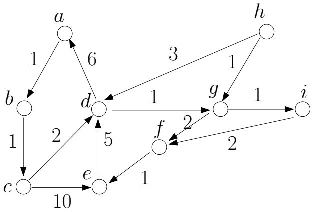


vertex v	$dist(v)$	$parent(v)$
a	8	d
b	∞	nil
c	0	nil
d	2	c
e	10	c
f	5	g
g	3	d
h	∞	nil
i	4	g

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

Example

Relax the out-going edges of i:

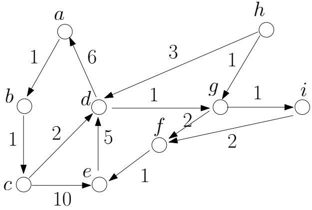


vertex v	$dist(v)$	$parent(v)$
a	8	d
b	∞	nil
c	0	nil
d	2	c
e	10	c
f	5	g
g	3	d
h	∞	nil
i	4	g

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

Example

Relax the out-going edges of f:

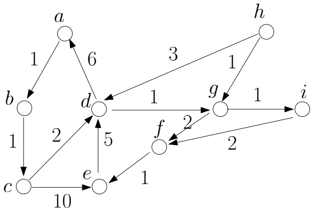


vertex v	$dist(v)$	$parent(v)$
a	8	d
b	∞	nil
c	0	nil
d	2	c
e	6	f
f	5	g
g	3	d
h	∞	nil
i	4	g

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

Example

Relax the out-going edges of e:

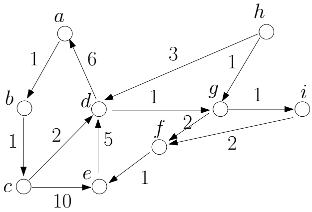


vertex v	$dist(v)$	$parent(v)$
a	8	d
b	∞	nil
c	0	nil
d	2	c
e	6	f
f	5	g
g	3	d
h	∞	nil
i	4	g

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

Example

Relax the out-going edges of a:

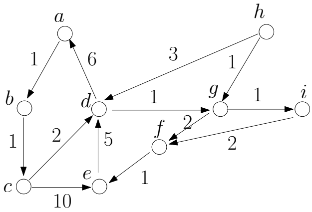


vertex v	$dist(v)$	$parent(v)$
a	8	d
b	9	a
c	0	nil
d	2	c
e	6	f
f	5	g
g	3	d
h	∞	nil
i	4	g

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

Example

Relax the out-going edges of b:

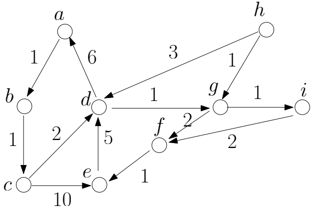


vertex v	$dist(v)$	$parent(v)$
a	8	d
b	9	a
c	0	nil
d	2	c
e	6	f
f	5	g
g	3	d
h	∞	nil
i	4	g

$S = \{h\}$.

Example

Relax the out-going edges of h:



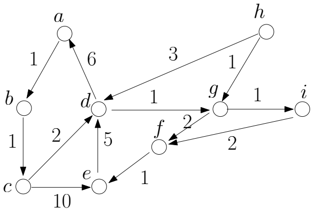
vertex v	$dist(v)$	$parent(v)$
a	8	d
b	9	a
c	0	nil
d	2	c
e	6	f
f	5	g
g	3	d
h	∞	nil
i	4	g

$S = \{\}$.
All the shortest path distances are now final.

Constructing the Shortest Path Tree

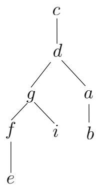
For every vertex v, if $u = parent(v)$ is not nil, then make v a child of u.

Example



vertex v	$parent(v)$
a	d
b	a
c	nil
d	c
e	f
f	g
g	d
h	nil
i	g

shortest path tree



Correctness and Running Time

It will be left as an exercise for you to prove that Dijkstra's algorithm is correct (solution provided).

Just as equally instructive is 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.