1 Shortest Path Algorithm

**Problem 1.** Given graph $G = (V, E)$ and length (or weights) $w(u, v)$ on each edge, find (length of) shortest path from source $s$ (to target $t$)

**Definition 2.** length of path = $\sum_{(u,v)} w(u, v)$

2 Dijkstra's Algorithm

Dijkstra's algorithm solves the single-source shortest-paths problem on a weighted graph for the case in which all edge weights are nonnegative.

Basic idea: The algorithm maintains a set $S$ of vertices whose final shortest-path weights from the source $s$ have already been determined. Then, repeatedly selects the vertex $u \in V - S$ with the minimum shortest-path estimate, adds $u$ to $S$ and relaxes all edges leaving from $u$.

First, initializing algorithm:

$dijkstra (G, s, t) $(V, E) = G$
for $v \in V$

$d[v] = \infty, \pi[v] = \bot$

d[$s$] = 0, $q = \{(0, s)\}$

Here $d[\cdot]$ is the shortest distance to source $s$ and $q$ stores elements going to be processed which is represents as (distance to $s$, vertex). And $\pi[v]$ is the previous vertex of $v$.

3 Depth-First Search/Breadth-First Search

The following part of Dijkstra's algorithm is find to the vertex $c$ in $V - S$ with shortest distance to $s$; add $c$ in $S$; then update all the vertices’ distance in $V - S$ which is connected to $c$. The process can be summarized as:
while $q \neq \emptyset$
  \[ c = q.\text{extractMin}() \quad O(|V|) \]
  \[ \text{if } c == t \]
    \[ \text{return } (d, \pi) \]
  \[ \text{for each edge } (c, v) \in E \]
  \[ \text{relax } (c, v, q) \]
\[ \text{return } (d, \pi) \]

The following algorithm describes the relax process:

\[ \text{relax } (u, v, q) \]
  \[ \text{if } d[u] + w(u, v) < d[v] \]
    \[ q.\text{update} (v, d[v] + w(u, v)) \]
    \[ d[v] = d[u] + w(u, v) ; \]
    \[ \pi[v] = u \]
    \[ \text{return } \text{true} \]
\[ \text{return } \text{false} \]

The following algorithm describes the update process:

\[ (d[v], v) \in q \]
\[ q.\text{update} (v, \text{oldd}, \text{newd}) \]
\[ q.\text{delete} (\text{oldd}, v) \]
\[ q.\text{insert} (\text{newd}, v) \]

Figure 3.1: Process of relaxing
Proposition 3. At all times $\forall u, d[u] \geq d(s, u)$.

Proof: (by mathematical induction)
True at start ($d[u] = \infty$).
Suppose true at start of relax, so $d[v] \geq d(s, v)$. Afterwards,

$$d[v] = d[u] + w(u, v) \geq d(s, u) + w(u, v) \geq d(s, v)$$

the conclusion is proved.

Proposition 4. Let $s = c_1, c_2, \cdots, c_n$ be the sequence of vertices obtained by extractMin(). Then $d(s, c_i) \leq d(s, c_{i+1})$. If $v \notin \{c_1, \cdots, c_k\}$, then $d(s, v) = \infty$.

Proof: (by contradiction)
Let $v$ be the closest element to $s$ such that the above conclusion fails.

- Case 1 $d(s, v) < 0$ but $v \in \{c_1, \cdots, c_k\}$
  - consider shortest path $s \to v' \to v$, so $d(s, v') < d(s, v)$
  - By induction, $v' \in \{c_1, \cdots, c_k\}$
  - by reading code, must have relax($v', v$)
  - At that time, would insert $v$ into $q$.

- Case 2 $v = c_i$
  - case a: $d(s, c_i) > d(s, c_{i+1})$
    * consider shortest path $s \to v' \to c_{i+1}$
    * Now $v' = c_j$ for some $j$. By ordering $j < c_i$, after $c_j$, $d[c_{i+1}] = d(s, c_{i+1})$
    * Then $d[c_{i+1}] < d(s, c_i) \leq d[c_i]$
    * so $c_{i+1}$ extracted before $c_i$.
  - case b: $d(s, c_i) < d(s, c_{i+1})$
    * the proof process is same

- Case 3: $d[v] \neq d(s, v)$ when $v$ extended, look at shortest path $s \to v' \to v$.
  By ordering, $v'$ extended before $v$ and $d[v'] = d(s, v')$ when it is extracted.
  This relax would update $d[v] = d[v'] + w(v', v) = d(s, v') + w(v', v) = d(s, v)$