CS3000: Algorithms & Data — Summer '23 — Laney Strange

Dijkstra's Algorithm

The algorithm below works on a weighted, directed graph G, represented in a adjacency list. This algorithm takes in the graph G, a starting vertex s, and a weight function w as parameters. It returns nothing, but instead updates the d (distance) and π (predecessor) values of every vertex.

Dijkstra's is a greedy algorithm whose job it is to find the shortest path from a source vertex s to all vertices reachable from s. It is one of a handful of single-source shortest-paths (SSSP) algorithms, and works only when the graph contains non-negative weight edges. It uses a helper function, Relax, to update the d and π values on the vertices. It also uses a helper function Initialize-Single-Source, to initialize the d and π values before the algorithm begins.

```
Dijkstra(G, s, w)
Initialize-Single-Source(G, s)
S = ∅
H = ∅
for each vertex u ∈ G.V
    Insert(H, u)
while H ≠ ∅
    u = Extract-Min(H)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        Relax(u, v, w)
        if the call of Relax decreased v.d
            Decrease-Key(H, v, v.d)
```

```
Initialize-Single-Source(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = Nil
s.d = 0
```

```
Relax(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

We typeset the procedures above with the following LaTeX:

```
\begin{codebox}
\Procname{$\proc{Dijkstra}(G, s, w)$}
\li $\proc{Initialize-Single-Source}(G, s)$
\li $S \gets \emptyset$
```

```
\li $H \gets \emptyset$
\li \For each vertex $u \in G.V$
\Do
\li $\proc{Insert}(H, u)$
\End
\li \While $H \ne \emptyset$
\Do
\li $u \gets \proc{Extract-Min}(H)$
\li $S \gets S \cup \{u\}$
\li \For each vertex $v \in G.Adj[u]$
\Do
\li $\proc{Relax}(u, v, w)$
\li \If the call of $\proc{Relax}$ decreased $v.d$
\Then
\li $\proc{Decrease-Key}(H, v, v.d)$
\end{codebox}

\begin{codebox}
\Procname{$\proc{Initialize-Single-Source}(G, s)$}
\li \For each vertex $v \in G.V$
\Do
\li $v.d \gets \infty$
\li $v.\pi \gets \const{Nil}$
\End
\li $s.d \gets 0$
\end{codebox}


\begin{codebox}
\Procname{$\proc{Relax}(u, v, w)$}
\li \If $v.d > u.d + w(u, v)$
\Then
\li $v.d \gets u.d + w(u, v)$
\li $v.\pi \gets u$
\end{codebox}
```