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

Prim's Algorithm

Prim's algorithm finds a Minimum Spanning Tree for a given graph. It takes in a graph G, a weight function w, and a starting vertex r. Instead of maintaining a set A of edges, it maintains a *key* and π attribute for each vertex. The *key* attribute for a vertex v contains the weight of edge (u, v), and the π attribute contains the node that immediately preceded that vertex in the spanning tree.

Prim's algorithm uses three Heap operations: INSERT, which inserts an element into the heap and re-heapifies, EXTRACT-MIN, which removes and returns the smallest element and re-heapifies, and DECREASE-KEY, which changes the value of a specific node and re-heapifies.

PRIM(G, w, r)

for each vertex $u \in G.V$
    $u.key = \infty$
    $u.\pi =$ NIL
$r.key = 0$
$H = \{\}$
for each vertex $u \in G.V$
    INSERT(H, u)
while $H \neq \{\}$
    $u =$ EXTRACT-MIN(H)
    for each vertex $v \in G.adj[u]$
        if $v \in H$ and $w(u, v) < v.key$
            $v.\pi = u$
            $v.key = w(u, v)$
            DECREASE-KEY$(H, v, w(u, v))$

We typeset the Prim's procedure above with the following LaTeX:

```
\begin{codebox}
\Procname{$\proc{Prim}(G, w, r)$}
\li \For each vertex $u \in G.V$
\Do
\li $\id{u.key} \gets \infty$
\li $\id{u.\pi} \gets \const{Nil}$
\End
\li $\id{r.key} \gets 0$
\li $H \gets \{\}$
\li \For each vertex $u \in G.V$
\Do
\li $\proc{Insert}(H, u)$
\End
\li \While $H \ne \{\}$
\Do
```

```
\li $u \gets \proc{Extract-Min}(H)$
\li \For each vertex $v \in G.adj[u]$
\Do
\li \If $v \in H$ and $w(u, v) < \id{v.key}$
\Then
\li $v.\pi \gets u$
\li $\id{v.key} \gets w(u, v)$
\li $\proc{Decrease-Key}(H, v, w(u, v))$
\end{codebox}
```