Prim's Algorithm

1 Algorithm

Build the MST by adding leaves one at a time from an arbitrary source vertex. Uses a similar greedy criterion to Dijkstra's, but this one only keys vertices by the weight of the lightest edge coming out from the partial MST.

```
def primMST(G=(V,E), s):
    initialize heap Q with all vertices to ∞, s to 0
    pred[s] = null
    while |Q| > 0:
        u = extract-min(Q)
        for (u, v) in E:
            if v undiscovered and w(u, v) < Q[v]:
                update v in Q with w(u, v)
                pred[v] = u

            mark u discovered
```

`pred` defines the MST as an inverted tree rooted at s.

2 Proof

At any time, the graph is partitioned into a cut $(S, V - S)$ where S is the set of all processed vertices. By the definition of the greedy criterion, vertices are added to S by choosing the cheapest edge that crosses this cut. By the safe edge lemma, the algorithm always adds safe edges and therefore correctly computes an MST.

3 Runtime

Extractions from the heap are $\Theta(\log n)$ each, while updating the neighbors of an extracted node in the heap is an $O(degree(u) \cdot \log n)$ operation.

$$T(n, m) = \sum_{u \in V} \Big(\log n + degree(u) \cdot \log n \Big) \tag{1}$$

$$= \log n \sum_{u \in V} (1 + degree(u)) \tag{2}$$

$$= \log n \cdot (n + 2m) \in \Theta(m \log n) \tag{3}$$