Kruskal's algorithm

```
0. T := new container for edges
1. L := edges sorted in non-decreasing order by weight
2. for each vertex v:
3.   v.cluster := make-cluster(v)
4. for each (u, v) in L:
5.   if u.cluster != v.cluster:
6.     T.add((u,v))
7.     merge u.cluster and v.cluster
8. return T
```

Kruskal's algorithm: correctness

Kruskal's algorithm maintains the loop invariants:

1. each cluster is a tree
2. $T \subseteq T_{min}$ for some MST T_{min}

Initially T is empty and clusters are single vertices, so trivially true.

Suppose (1) and (2) are true before line 4.

- on line 5, if $u.cluster \neq v.cluster$, then
- since u's cluster is a tree and v's cluster is a different tree,
- then the merged cluster (line 7) is a tree

Kruskal's algorithm: correctness

Suppose (1) and (2) are true before line 4.

- if $(u, v) \in T_{min}$, then choose $T'_{min} = T_{min}$ and done
- if $(u, v) \notin T_{min}$, then partition V into S and $V - S$ such that u's cluster $\subseteq S$, v's cluster $\subseteq V - S$, and no edge between S and $V - S$
- in T_{min} there is a unique simple path connecting u and v
- in T_{min} there is some edge (u', v') connecting S and $V - S$
- without (u', v'), T_{min} disconnected; (u, v) would reconnect
- (u, v) is the minimum-weight edge in L connecting two clusters
- $\therefore weight(u, v) \leq weight(u', v')$
- then choose $T'_{min} = T_{min} - \{(u', v')\} + \{(u, v)\}$ is an MST

Prim's algorithm

```
T := new container for edges
PQ := new min-heap()
start := pick a vertex
PQ.insert(0, start)
for each vertex v != start: PQ.insert(inf, v)
while not PQ.is-empty():
  u := PQ.extract-min()
  T.add((u.pred, u))
  for each v in u's adjacency list:
    if v in PQ and w(u, v) < prioroty(v):
      PQ.decrease-priority(v, w(u,v))
      v.pred := u
return T
```

Prim's algorithm: correctness

Prim's algorithm maintains the loop invariants:

1. T contains vertices in $V - PQ$
2. for each v in PQ, $priority(v)$ = minimum weight of any edge between v and T
3. $T \subseteq T_{min}$ for some MST T_{min}

Initially T is empty, PQ contains all of V, and all priorities are ∞, so trivially true.

Suppose (1), (2), and (3) are true before line 5.

- line 6 extracts u from PQ, line 7 adds edge $(u.pred, u)$ to T, so (1)
- lines 8-11 update priorities of vertices adjacent to u, so (2)

Prim's algorithm: correctness

Suppose (1), (2), and (3) are true before line 5. Let $p = u.pred$.

- if $(p, u) \in T_{min}$, then choose $T'_{min} = T_{min}$ and done
- if $(p, u) \notin T_{min}$, then in T_{min} there is a unique simple path connecting p and u
- in T_{min} there is some edge (x, y) where x no longer in PQ and y in PQ on a path from p to u
- without (x, y), T_{min} disconnected; (p, u) would reconnect
- u was just extracted from PQ, so $weight(p, u) = priority(u) \leq priority(y) = weight(x, y)$
- then choose $T'_{min} = T_{min} - \{(x, y)\} + \{(p, u)\}$ is an MST

General Theorem

Suppose

- $T \subseteq T_{min}$
- can partition V into S and $V - S$ (cut), such that
  - no edge between V and $V - S$
  - (u, v) is the cheapest edge (light edge) connecting V and $V - S$ (crosses the cut)

Then $T + \{(u, v)\} \subseteq T'_{min}$

- if $(u, v) \notin T_{min}$
- T_{min} has a unique simple path from u to v, via some edge (u', v') with $u' \in S$ and $v' \in V - S$
- T_{min} without (u', v') disconnected; (u, v) would would reconnect
- $weight(u, v) \leq weight(u', v')$
- Choose $T'_{min} = T_{min} - \{(u', v')\} + \{(u, v)\}$