PRIM's MST algorithm

- Start with spanning tree containing arbitrary an vertex r and no edges
- Grow spanning tree by repeatedly adding minimal weight edge connecting vertex in current tree with a vertex not in the tree
- To find minimal edge connected to current tree we maintain a priority queue on vertices not in the tree:
  - The key/priority of a vertex v is the weight of minimal weight edge connecting v to the tree. We maintain pointer from adjacency list of v to v in the priority queue.
  - For each node v maintain $visit(v)$ such that edge $(v, visit(v))$ is the best edge connecting v to the current tree.

```
PRIM
/* initialize */
Pick arbitrary vertex r
For each vertex u ∈ V, u ≠ r: INSERT(PQ, u, ∞)
INSERT(PQ, r, 0), visit(r) = NULL
/* main loop */
WHILE PQ not empty
    u = DELETE-MIN(PQ)
    For each (u, v) ∈ E:
        IF v ∈ PQ and w(u, v) < key(v):
            visit[v] = u
            DECREASE-KEY(PQ, v, w(u, v))
Output edges (u, visit(u)) as part of MST.
```

Kruskal's MST algorithm

```
KRUSKAL
/* initialize */
For each vertex v ∈ V: MAKE-SET(v)
Sort edges of E in increasing order by weight
/* main loop */
FOR each edge e = (u, v) ∈ E in order of weight:
    IF FIND-SET(u) ≠ FIND-SET(v) THEN
        output edge e as part of MST
        UNION-SET(u, v)
```