PRIM's MST algorithm

- Start with an arbitrary vertex r. Grow MST by repeatedly adding the smallest edge connecting a vertex in the tree with a vertex not in the tree
- To find the smallest edge we use a priority queue containing the *vertices* not in the tree yet:
  - The key/priority $d[v]$ of a vertex v is the weight of the smallest edge connecting v to the tree (implementation note: the priority of a vertex will be stored in an array $d[v]$ and also in the priority queue $(v, d[v])$; in order to be able to decrease the prioriy of a vertex we store a pointer to v's location in the priority queue)
  - For each vertex v we store the edge that connects it to the tree; we call the other vertex of this edge by $pred(v)$

```
PRIM(G)
// initialize
Pick arbitrary vertex r and set d[r] = 0, PQ.INSERT(r, 0), pred(r) = NULL
For each vertex u ∈ V (u ≠ r): d[u] = ∞, PQ.INSERT(u, ∞)

while PQ not empty
    u = PQ.DELETE-MIN() // u is vertex closest to the tree
    For each adjacent edge (u, v)
        IF v in PQ and w_uv < d[v]
            PQ.DECREASE-KEY(v, w_uv)
            pred[v] = u
Output the edges (u, pred(u)) as the MST.
```

Analysis: $O(|E| \lg |V|)$

Kruskal's MST algorithm

```
KRUSKAL(G)
// initialize
For each vertex v ∈ V: MAKE-SET(v)
Sort edges of E in increasing order by weight

for each edge e = (u, v) in order of weight
    if FIND-SET(u) ≠ FIND-SET(v)
        output edge e as part of MST
        UNION-SET(u, v)
```

Analysis: $O(|E| \lg |V|)$