Algorithm 3: Kruskal's Algorithm

Input: graph $G = (V, E)$, weight function $w : E \to \mathbb{R}$

Output: minimal spanning tree $T = (V, F)$ of G

```
1 F ← ∅
2 L ← E
3 Sort the edges in L increasingly by weight
4 while L ≠ ∅ do
5 |   e ← pop_front(L)
6 |   if (V, F ∪ {e}) is acyclic then
7 |   |   F ← F ∪ {e}
```

Algorithm 4: Prim's Algorithm

Input: connected graph $G = (V, E)$ given as adjacency list, weight function $w : E \to \mathbb{R}$, root node $r \in V$

Output: minimal spanning tree $T = (V, \{(v, \text{pred}(v)) \mid v \in V \setminus \{r\}\})$ of G

```
1  foreach v ∈ V do
2  |  pred(v) ← 0
3  |  dist(v) ← ∞ // distance from tree
4  Q ← V // priority queue
5  dist(r) ← 0
6  while Q ≠ ∅ do
7  |  v ← extract_min(Q) // vertex with minimal
   |     distance
8  |  foreach u ∈ Adj(v) do
9  |  |  if u ∈ Q and w(u, v) < dist(u) then
10 |  |  |  pred(u) ← v
11 |  |  |  dist(u) ← w(u, v)
```