MST Algorithms

MST-KRUSKAL

MST-KRUSKAL(G, w)
1. A ←∅
2. for each vertex v ∈ V[G]
3. do MAKE-SET(v)
4. sort the edges of E into nondecreasing order by weight w
5. for each edge (u, v) ∈ E, taken in nondecreasing order by weight
6. do if FIND-SET(u) ≠ FIND-SET(v)
7. then A ← A ∪ {(u, v)}
8. UNION(u, v)
9. return A

The implementation of Kruskal’s algorithm uses a disjoint-set data structure to maintain several disjoint sets of elements. The definitions of disjoint-set operations are listed in page 499 of the textbook.

MST-PRIM

MST-PRIM(G, w, r)
1. for each vertex u ∈ V[G]
2. do key[u] ← ∞
3. π[u] ← NIL
4. key[r] ← 0
5. Q ← V[G]
6. while Q ≠ ∅
7. do u ← EXTRACT-MIN(Q)
8. for each v ∈ Adj[u]
9. do if v ∈ Q and w(u, v) < key[v]
10. then π[v] ← u
11. key[v] ← w(u, v)
12. DECREASE-KEY(Q, v, key[v])

The implementation of Prim’s algorithm uses a min-priority queue Q, containing vertices v using key[v] as the key-value. The performance of Prim’s algorithm depends on how we implement Q.