A Comparison of Prim’s and Kruskal’s Algorithms:

-Kruskal’s algorithm can generate disconnected components as well as it can work on disconnected components.
-Kruskal also invented another algorithm, sometimes called the reverse-delete algorithm, that always produces a minimum spanning tree when given as input a weighted graph with *distinct* edge weights.
This algorithm proceeds by successively deleting edges of maximum weight from a connected graph as long as doing so does not disconnect the graph.
-to find a minimum spanning tree of a graph with m edges and n vertices, Kruskal’s algorithm can be carried out using $O(m \log m)$ operations and Prim’s algorithm can be carried out using $O(m \log n)$ operations. Therefore, Kruskal’s algorithm may be preferred for graphs that are sparse, that is, where m is very small compared to $C(n, 2) = n(n - 1)/2$, the total number of possible edges in an undirected graph with n vertices. Otherwise, there is little difference in the complexity of these two algorithms.
-In Prim’s algorithm edges of minimum weight that are incident to a vertex already in the tree, and not
forming a circuit, are chosen; whereas in Kruskal’s algorithm edges of minimum weight that are not necessarily incident to a vertex already in the tree, and that do not form a circuit, are chosen. As in Prim’s algorithm, if the edges are not ordered, there may be more than one choice for the edge to add at a stage of this procedure. Consequently, the edges need to be ordered for the procedure to be deterministic.
-Prim’s algorithm assumes that all vertices are connected. But in a directed graph, every node may not be reachable from every other node. If the directed graph fail the requirement that all vertices are connected, Prim’s algorithm fails.
-In Kruskal’s algorithm, at each step, it is checked that if the edges form a cycle with the spanning-tree formed so far. But Kruskal’s algorithm may fail to correctly detect the cycles in a directed graph.