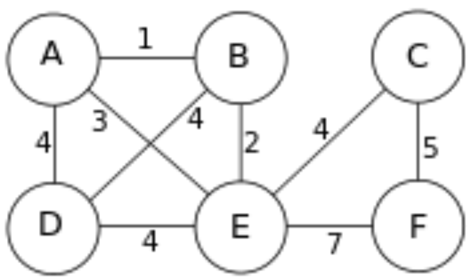
Case 0: If all vertices are not connected, only Kruskal's algorithm provides a minimum spanning forest. Prim's algorithm would have to be individually applied to each vertex-group to achieve the same result.

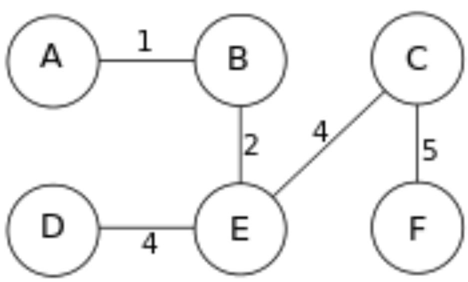
Case 1: If each edge has a distinct weight then there will be only one, unique minimum spanning tree (MST), found by both Kruskal and Prim's algorithm.

Case 2: If each edge has the same weight, then there will be multiple MSTs found by each algorithm.

Case 3: If each edge does not have a distinct weight or the same weight, there may exist more than one unique MST. While Kruskal and Prim's algorithms can find the same solution, they can also find different solutions.

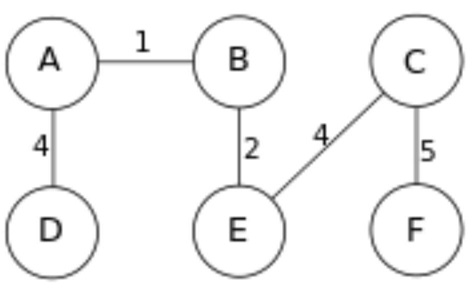


For example,
Given the graph on the left.



Kruskal's algorithm could find the MST on the left through the following steps:

1. Add A—B
2. Add B—E
3. Arbitrarily choose D—E over A—D
4. Add E—C
5. Add C—F



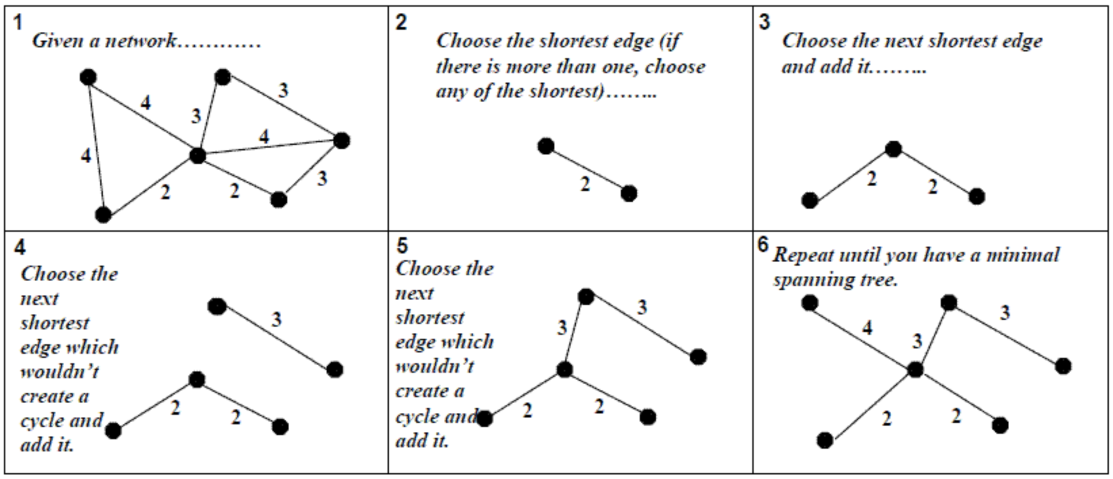
Prim's algorithm could find the MST on the left through the following steps:

1. Arbitrarily choose F
2. Add F—C
3. Add C—E
4. Add E—B
5. Add B—A
6. Add A—D

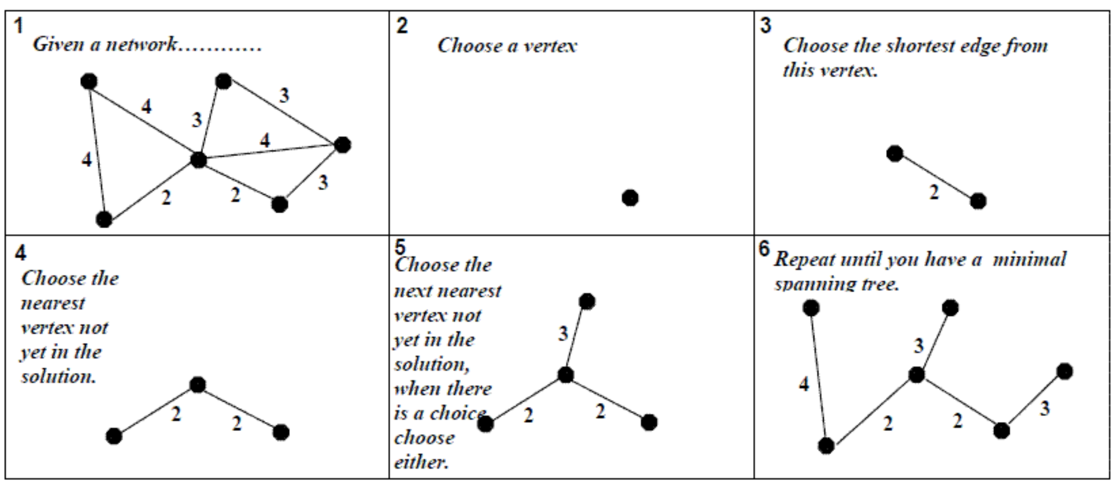
Picture Source - https://en.wikipedia.org/wiki/Minimum_spanning_tree

Another representation of Kruskal’s verus Prim’s resulting in different MSTs.
Source - http://stackoverflow.com/questions/1195872/kruskal-vs-prim

Kruskal’s Algorithm



Prim’s Algorithm



In summary, in Case 3, Kruskal and Prim’s algorithm can result in different solutions. This is largely influenced by arbitrary decisions such as which edge between two edges of the same weight Kruskal chooses and the initial vertex Prim’s chooses. It is also influenced by the difference in the process: Kruskal’s builds up the solution edge to edge while Prim’s builds up the solution vertex to vertex.