Prim’s Algorithm
The idea behind Prim’s algorithm is simple, a spanning tree means all vertices must be connected. So the two disjoint subsets of vertices must be connected to make a Spanning Tree. And they must be connected with the minimum weight edge to make it a Minimum Spanning Tree.

Algorithm
1) Create a set mstSet that keeps track of vertices already included in MST.
2) Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.
3) While mstSet doesn’t include all vertices

    a. Pick a vertex u which is not there in mstSet and has a minimum key value.

    b. Include u to mstSet.

    c. Update the key value of all adjacent vertices of u. To update the key values, iterate through all adjacent vertices. For every adjacent vertex v, if the weight of edge u-v is less than the previous key value of v, update the key value as the weight of u-v

The idea of using key values is to pick the minimum weight edge from the cut. The key values are used only for vertices that are not yet included in MST, the key value for these vertices indicate the minimum weight edges connecting them to the set of vertices included in MST.

Time Complexity
If one implements Prim's algorithm using only an adjacency matrix, its time complexity is $O(N^2)$ - where N is the number of nodes in a graph. The simple implementation yields high time complexity in this case.

One can improve time complexity by opting in for a more complex implementation, consisting of Fibonacci or a Binary Heap alongside the adjacency matrix. In that case, the time complexity could be O(E log N), meaning that Prim's algorithm can run as fast as Kruskal's!

Kruskal’s Algorithm

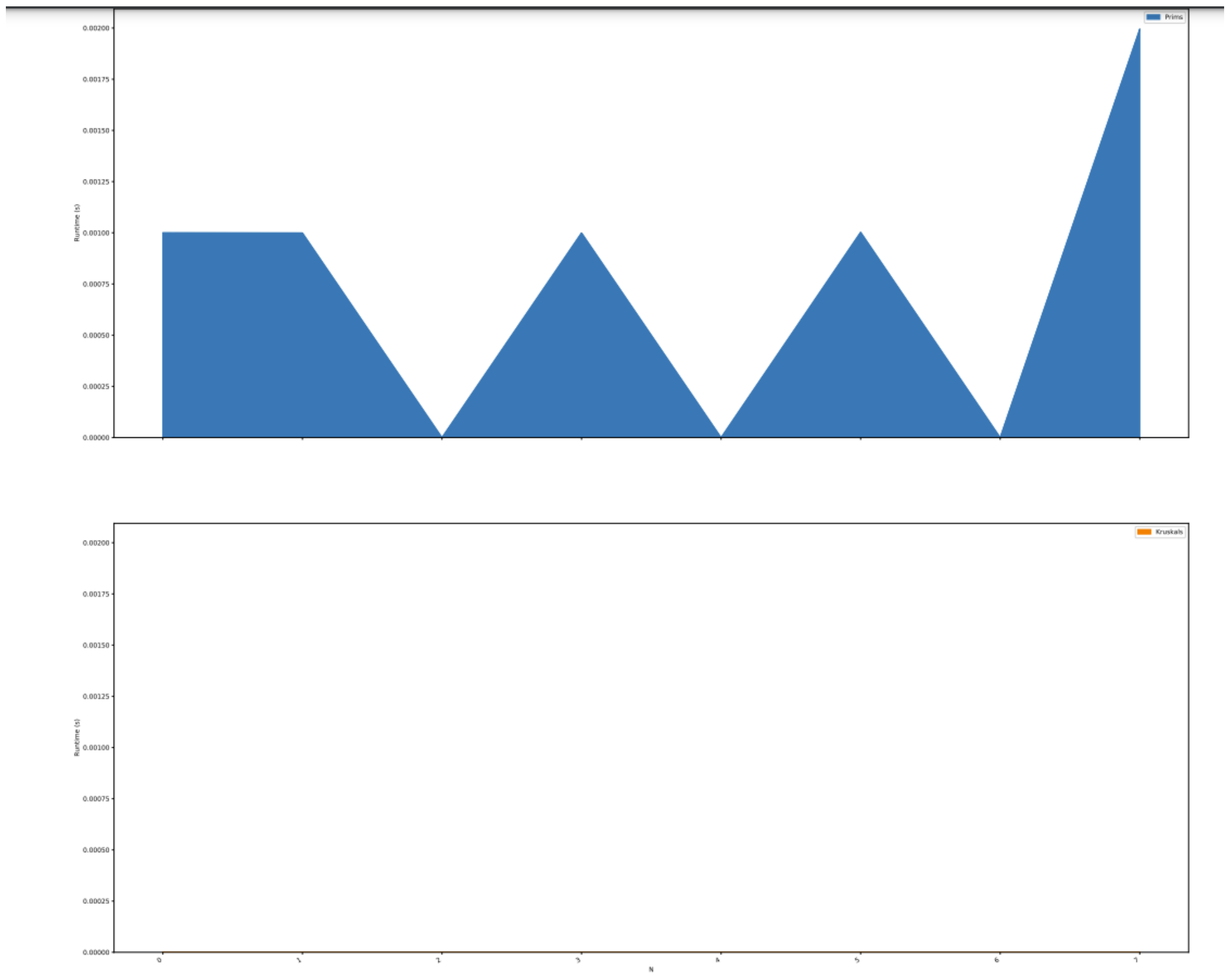
The idea behind Kruskal's algorithm is pretty simple, there are essentially two steps that you repeat until you get the MST:

1. Sort all edges by their weight in the ascending order
2. Pick the edge with the smallest weight and try to add it to the tree
3. If it forms a cycle, skip that edge

Repeat these steps until you have a connected tree that covers all nodes

Time complexity
Let's assume that you have a graph with E edges and N nodes. You will find the MST using Kruskal's algorithm in time of O(E log E), which is equivalent to O(E log N).

Runtime v/s n



Conclusion

Prim's algorithm is not only efficient but flexible when it comes to finding the Minimum Spanning Tree of a graph. The Python implementation is also really simple. MSTs are useful structures that can be applied in a wide variety of fields, making Prim's and Kruskal’s algorithms an incredibly important feature to use while solving diverse problems..