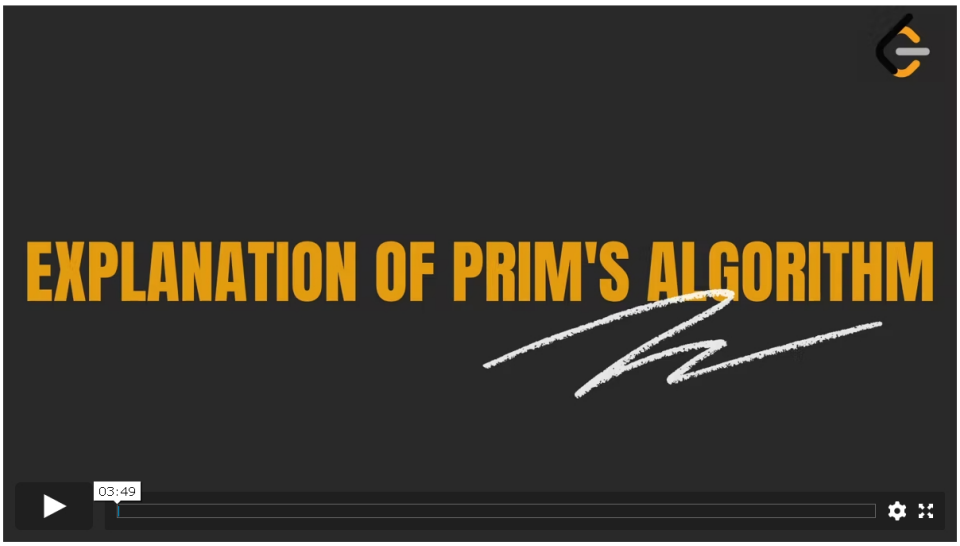
Prim's Algorithm

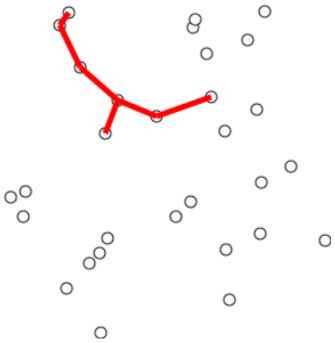
Report Issue

Video Explanation

After learning about Kruskal's Algorithm, let's look at another algorithm, "Prim's algorithm", that can be used to construct a "minimum spanning tree" of a "weighted undirected graph".



Visual Example



The above illustration demonstrates how Prim's algorithm works by adding vertices. In this example, the distance between two vertices is the edge weight. Starting from an arbitrary vertex, Prim's algorithm ***grows*** the minimum spanning tree by adding one vertex at a time to the tree. The choice of a vertex is based on the ***greedy*** strategy, *i.e.*, the addition of the new vertex incurs the minimum cost.

Proof of the Prim's Algorithm

You may wonder, does Prim's Algorithm guarantee that we get the correct "minimum spanning tree" at the end? The answer to this question is yes, as we will explain in the next video.

The difference between the “Kruskal’s algorithm” and the “Prim’s algorithm”

“Kruskal’s algorithm” expands the “minimum spanning tree” by adding edges. Whereas “Prim’s algorithm” expands the “minimum spanning tree” by adding vertices.

Complexity Analysis

V represents the number of vertices, and E represents the number of edges.

- Time Complexity: $O(E \cdot \log V)$ for Binary heap, and $O(E + V \cdot \log V)$ for Fibonacci heap.
    - For a Binary heap:
        - We need $O(V + E)$ time to traverse all the vertices of the graph, and we store in the heap all the vertices that are not yet included in our minimum spanning tree.
        - Extracting minimum element and key decreasing operations cost $O(\log V)$ time.
        - Therefore, the overall time complexity is $O(V + E) \cdot O(\log V) = O(E \cdot \log V)$.
    - For a Fibonacci heap:
        - Extracting minimum element will take $O(\log V)$ time while key decreasing operation will take amortized $O(1)$ time, therefore, the total time complexity would be $O(E + V \cdot \log V)$.
- Space Complexity: $O(V)$. We need to store V vertices in our data structure.