Prim 算法是一种贪婪算法，可在加权连通图里搜索最小生成树。

贪心算法（又称贪婪算法）是指，在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，他所做出的是在某种意义上的局部最优解。

算法从将起始顶点添加到 T 中开始，然后持续地将顶点从未加入该树的顶点添加到 T 中。每次选择顶点是选择与 T 中顶点边权值最小的那个顶点（不是只和刚加进 T 中的那个点比较，而是和 T 中所有的点比较）。

```
public MST getMinimumSpanningTree(int startingVertex) {
        // cost[v] stores the cost by adding v to the tree
        double[] cost = new double[getSize()];
        for (int i = 0; i < cost.length; i++) {
            cost[i] = Double.POSITIVE_INFINITY; // Initial cost
        }
        cost[startingVertex] = 0; // Cost of source is 0

        int[] parent = new int[getSize()]; // Parent of a vertex
        parent[startingVertex] = -1; // startingVertex is the root
        double totalWeight = 0; // Total weight of the tree thus far

        List<Integer> T = new ArrayList<>();

        // Expand T
        while (T.size() < getSize()) {
            // Find smallest cost v in V - T
            int u = -1; // Vertex to be determined
            double currentMinCost = Double.POSITIVE_INFINITY;
            for (int i = 0; i < getSize(); i++) {
                if (!T.contains(i) && cost[i] < currentMinCost) {
                    currentMinCost = cost[i];
                    u = i;
                }
            }

            T.add(u); // Add a new vertex to T
            totalWeight += cost[u]; // Add cost[u] to the tree

            // Adjust cost[v] for v that is adjacent to u and v in V - T
            for (Edge e : neighbors.get(u)) {
                if (!T.contains(e.v) && cost[e.v] > ((WeightedEdge)e).weight) {
                    cost[e.v] = ((WeightedEdge)e).weight;
                    parent[e.v] = u;
                }
            }
        } // End of while
```

```
    return new MST(startingVertex, parent, T, totalWeight);
}
```