Minimum Spanning Tree (MST)

Definition of a minimum spanning tree (MST):

- input: set of unique points P. P can be 2D or 3D.
- output: an undirected acyclic graph of edges created from points in P. The graph is spanning, meaning that all points in P are included.
- Since the graph is acyclic, this means there are no "loops" (cycles). Thus the graph is actually a tree.
- "Undirected" means that the edges have no implicit direction (from : to)
- The tree edges are weighted by their lengths.
- The tree is minimal if the generated tree has a minimum overall sum of edge lengths (distances).
- Note that it is trivial to create a spanning tree that is not minimal: Select a point, and then create edges from that point to every other point. If there are K points, then there are (K-1) edges. We aren't too interested in these trees!

Prim's Algorithm

1. Create set of edges E for all points in P. Find distance of each edges in E. Sort edge list in increasing order of distance.
2. Mark all points in P as unused.
3. Initialize a tree data structure T to empty. (You will add edges to this structure).
4. Add smallest edge to T. Mark its end points as used.
5. While T is not spanning (ie. there exists a point p in P that is not yet included in T) {
    FInd smallest edge E[i] that has an endpoint P[j] that is still unused (not yet in T).
    Add E[i] to T.
    Mark P[j] as used.
   }

Comments:

- You need a data structure to save the edges. It will be of size K*(K-1), and includes (squared) length of the edge.
- Use the squared distance of each edge. Square roots are not useful and are a waste of CPU.
- There are other algorithms for MST. Kruskal's algorithm is a close variation of Prim's. There are also other ways to perform loop checking.
- Uses of MST: Local area network design. It creates a network that connects all nodes, while minimizing wire length, and hence cost and time delays (latency).

References:

Prim's Algorithm: https://en.wikipedia.org/wiki/Prim%27s_algorithm