I found a minimum spanning tree with prim's algorithm. A full graph is generated, with an (undirected) edge between any set of two vertices. The weight of the edge equals the distance, for task1, it is the Euclidean distance between two vectors and for task 3, it is the cosine distance between them.

A minimum spanning tree is found with default Prim algorithm, starting with the first node. The minimum spanning tree generated by two algorithms are almost same, with the exact same edges(vert) but different numeric weight on edges. The total weight of MST generated with Euclidean norm as edge weight is 245.48829 while with cosine similarity is 63.6918858.

The reason that both generates same MST is that distance is cardinal, given any two pairs of distance d1 and d2, we can always give a definite relationship either d1 is longer or d2 is longer regardless of how we represent the distance. So the weight between any two edges in the original graph has a definite relationship on which is greater and the numeracies of the weight is less relevant, thus resulting in the same MST.

If we view MST as an undirected graph, it is unique in this dictionary. The last part in my program checks if two edges have exact same weight

```
boolean unique=true;
for (Edge edge1 : tree.getEdges()){
        for (Edge edge2 : tree.getEdges()){
                if ((edge1 != edge2) && edge1.getWeight()==edge2.getWeight())
                {System.out.println(edge1.toString()+edge1.toString());unique=false;}

        }

}
System.out.println("Unique MST:"+unique);
```

Here we don't have any two edges with the exact same weight. Depending on the accuracy of calculation, one may have different result.

Starting from different vertices in Prim's algorithm results in MST that "looks" different, but if we disregard the direction of edge in a tree, all MST are in fact the same. To illustrate this, we can make edges MST undirected and compare all the graph (comparing each node to see if they have links to the same set of nodes). The choice root seems to be affected when starting from different nodes as Prim enqueues all incoming edges. Specifically, the root chosen is the starting node and it builds on with vertices closest to it.