**Minimal spanning Trees:**

* Given undirected, connected graph a spanning tree of graph G is tree that spans G => include every vertex of G. and is a subgraph of G(every edge in the tree belongs to G)
* Cost of spanning tree = sum of weights of all edges.
  + MST is spanning tree with lowest cost among other spanning trees, there can be multiple MST.
* Reason for finding MST?
  + Applications: network desgin: telephone, electrical, TV cable, computer, etc…
* **Two famous algorithms for finding MST: Kruskal, Prim’s Algorithm.**

**Kruskal’s Algorithm:**

* Build spanning tree by adding edges one by one into a growing spanning tree.
* Follow greedy approach -> at each iteration it finds an edge with lowest cost.
* How to check if 2 vertices are connected or not? 🡨 why ?
  + Could use DFS
  + Disjoint sets: sets whose intersection is empty set => do not have elements in common.
* Most time-consuming job here is sorting edges depending on their weights.
* Time complexity of disjoint-set operation will be O(E log V).

Steps:

* Sort edges in order of their weights (smallest -> largest)
* Add them in lowest to higher order to the MST.
* Only add edges that do not form a cycle.

From <https://www.tutorialspoint.com/data_structures_algorithms/kruskals_spanning_tree_algorithm.htm>:

* It treats each node as individual tree. It conencts to another iff it has least cost among other options and does not violate MST properties

**Prim’s Algorithm:**

* Similar to Kruskal in that it uses greedy approach to find minimum spanning tree. Grow spanning tree from starting position by adding **vertex**.
* Starting from any vertex, select another vertex that will give lowest weight edge.
* Time complexity is O((V+E) log V) since each vertex is inserted in priority queue only once and insertion in priority queue take logarithmic time.

Steps:

1. Maintain two disjoint sets of vertices. One containing vertices in growing spanning tree and other not in it.
2. Select cheapest vertex that is connected to growing spanning tree and not in growing spanning tree set, then add it to the set. Can be done using Priority Queues, insert vertices that are connected to growing spanning tree (not already in), into priority queue.
3. Check for cycles to do that mark nodes which have been already selected and insert only those nodes in Priority Queue that are not marked.

From <https://www.tutorialspoint.com/data_structures_algorithms/prims_spanning_tree_algorithm.htm>

* “in contrast with Kruskal’s it treats nodes as single tree” but didn’t they say kruskal treats graph as graph and every node as an individual tree? What is the difference between single tree and individual tree?
* Keep on adding new nodes to spanning tree from given graph
* Prim’s algorithm:

1. Remove loops, parallel edges(keep lower cost one)
2. Choose any arbitrary node as root.
3. Check outgoing edges and select one with less cost

Questions:

* Only for weighted trees graphs?
* If Kruskal, Prim does the same thing what is the reason for having to different algorithm??

Applications:

* Designs of networks
* Used in alg approximating travelling salesman problem, minimum-cost weighted perfect matching, etc…
* Cluster Analysis
* Handwriting recognition
* Image segmentation

References:

* <https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/>