Centralized Algos for constructing MST

Kruskal: Assumptions: simple graph, no parallel edges

1. Order edges in ascending order of edge weights
2. Add an edge to the "partial" tree if no cycles are formed.
3. Stop when n-1 edges have been added.
Prim Algorithm:

Let \( T \) be partial tree with start node.

\( T = (\{s_3\}, \emptyset) \)

1. for \( n-1 \) step
   - find the lowest weight edge \( e \) from \( T \) to \( T' \)
   - add \( e \) to \( T \).
MST Property

Let \( G \) be a weighted graph and let \( V_1 \) and \( V_2 \) be a partition of vertices of \( G \) into two disjoint, non-empty sets. Furthermore, let \( e \) be an edge in \( G \) with minimum weight from among those with one end point in \( V_1 \) and the other in \( V_2 \). There is a MST \( T \) that has \( e \) as one of its edges.
Proof by contradiction.
- Assume $e$ is not a part of any MST.
- Let $T$ be a MST
- adding $e$ forms a cycle with $e'$ in it.
- removing $e'$ we get another tree with lower weight. - contradicting the assumption that $T$ was MST
Barvinka's Algo.

**Input:** A weighted graph $G = (V, E)$ with $n$ vertices and $m$ edges.

**Output:** A MST $T$ for $G$.

1. Let $T$ be a subgraph of $G$, initially containing just the vertices in $V$.
2. While $T$ has fewer than $n-1$ edges do
   1. For each connected component $C_i$ of $T$ do:
     1. Find the smallest-weight edge $e(v, u)$ in $E$ with $v$ in $C_i$ and $u$ not in $C_i$.
     1.6 Add $e$ to $T$.
3. Return $T$. 
Time complexity? $O(n \log n)$

1. The number of rounds is $O(\log n)$ since at each step the number of components is halved in the worst case.
2. Selection of minimum weight edges across component edges can be done in $O(n)$ time at each round.

Converting all edge weights to uniform edge weights: $1 \rightarrow 1.00012\ldots$,
$2 \rightarrow 3.00043\ldots$,
$3 \rightarrow 3.50067\ldots$,
Barwke Algo - Distributed (Synchronous)

1. Let $T$ be partial tree - initially with only nodes & no tree edges.
2. While there is no more than one connected component defined by $T$ do
   a. for each connected component $C$ in parallel do
      i. choose the smallest edge $e$ joining $C$ to another component
      ii. add $e$ to the set $T$ of tree edges
Two implementation issues:
1. Identifying connected components
2. For each connected component, finding the minimum weight edge.