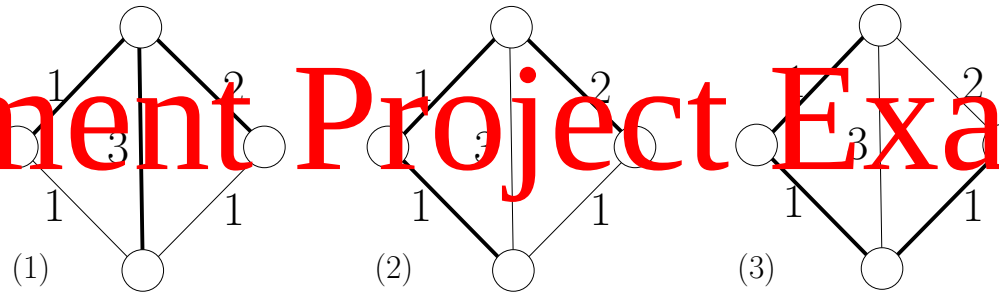
Distributed MST

Radu Nicolescu
Department of Computer Science
University of Auckland

14 Aug 2020

Spanning trees (ST)



For (1) and (2), consider that the top node is the root

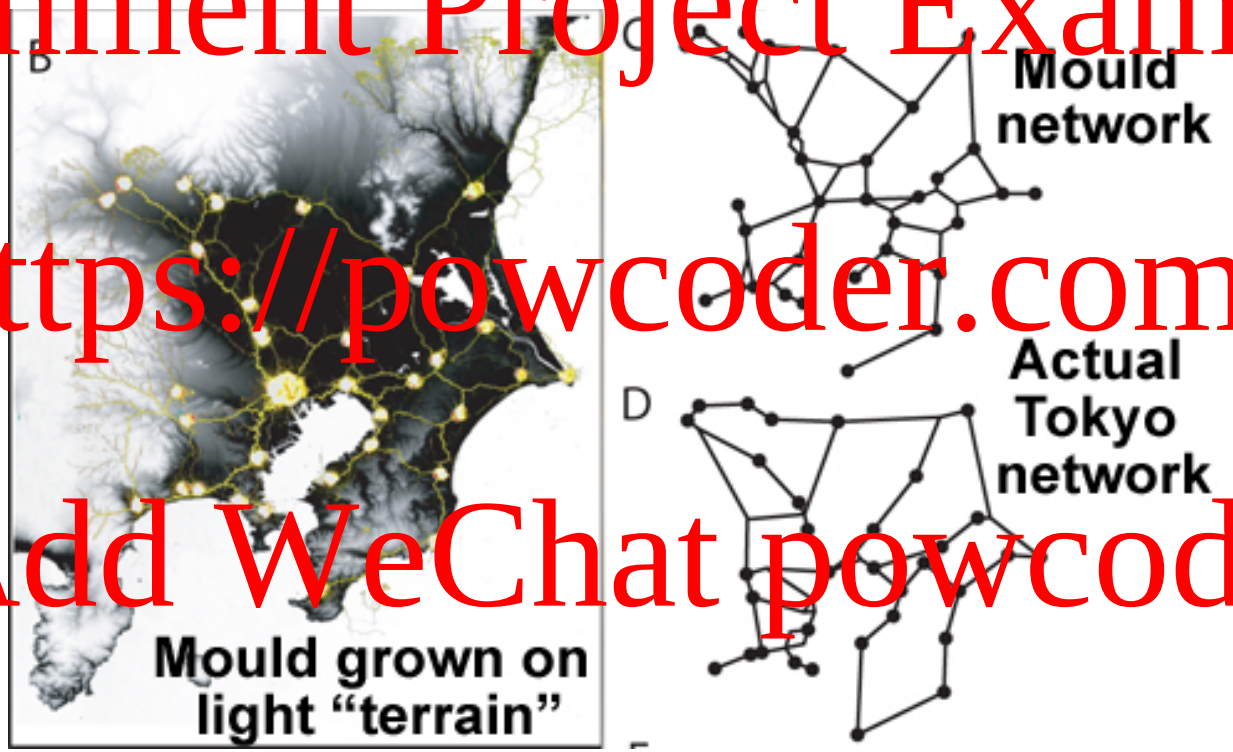
- (1) : min-height ST
  here also BFS ST (cf. sync Echo)
- (2) : shortest paths ST (cf. sync/async Bellman-Ford)
- (3) : minimum ST (cf. sync/async GHS)
  here also DFS ST (cf. sync/async Cidon)
- (1,2,3,...) : arbitrary ST (cf. async Echo)

Minimum spanning tree (MST) algorithms

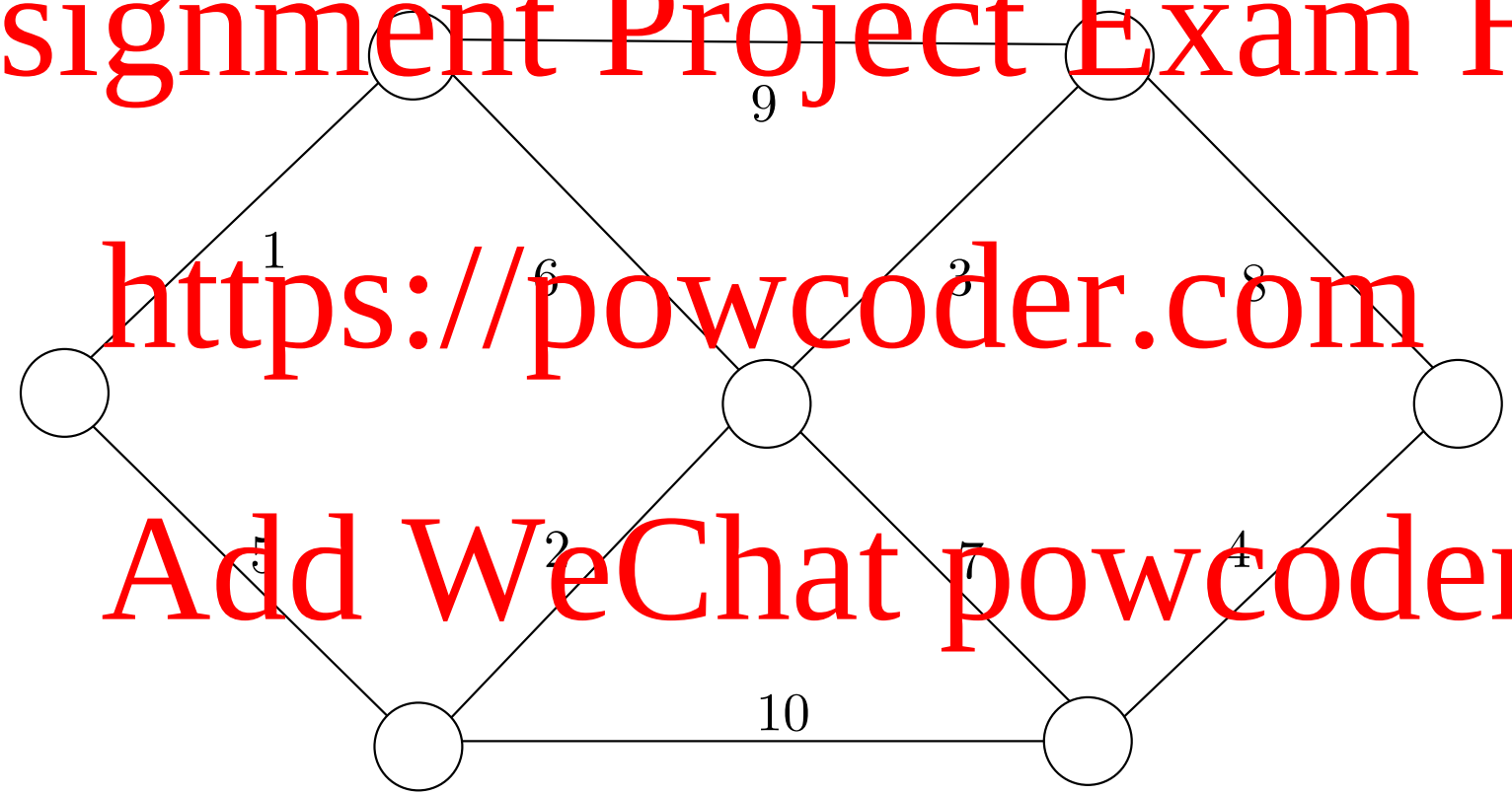
- If edges have different weights, then there is a unique MST
- Prim (1957), Jarník (1930), Dijkstra (1959): $O(M \log N)$ or $O(M + N \log N)$
- Kruskal (1956): $O(M \log N)$; Reverse-Kruskal: $O(M \log N \ldots)$
- Borůvka (1926); Choquet (1938); Florek, Łukasiewicz, Perkal, Steinhaus, and Zubrzycki (1951); Sollin (1965)[1]: $O(M \log N)$
- faster algorithms – almost linear: Chazelle (2000); randomization; integer weights; ...
- Distributed MST (sync,async): GHS - Gallager, Humblet and Spira (1983): $O(N \log N)$; improvements linear $O(N)$; or even sub-linear

[1] "Because Sollin was the only computer scientist in this list living in an English speaking country, this algorithm is frequently called Sollin's algorithm" (Wiki)

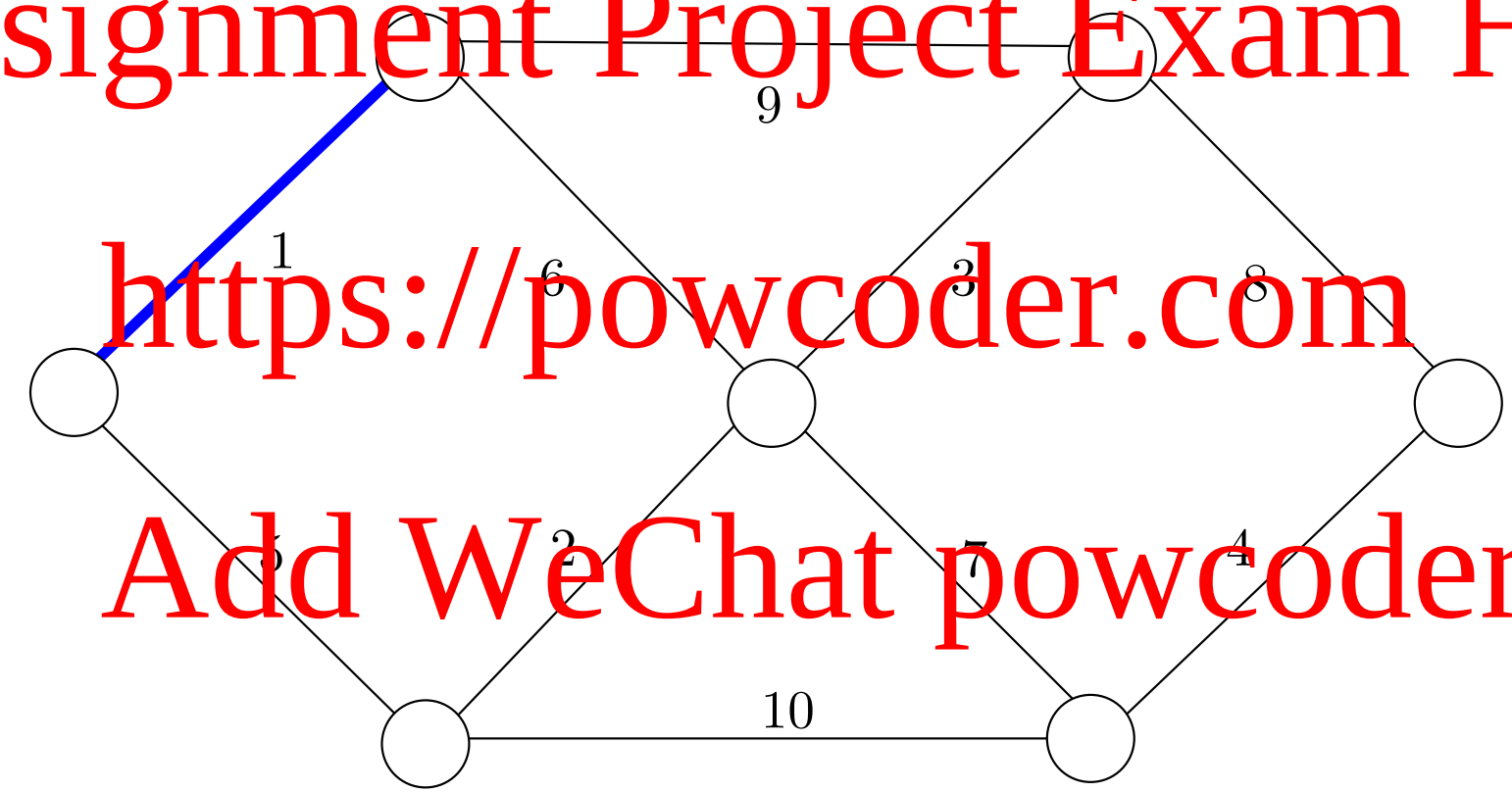
Side-bar: Minimum spanning networks in biology



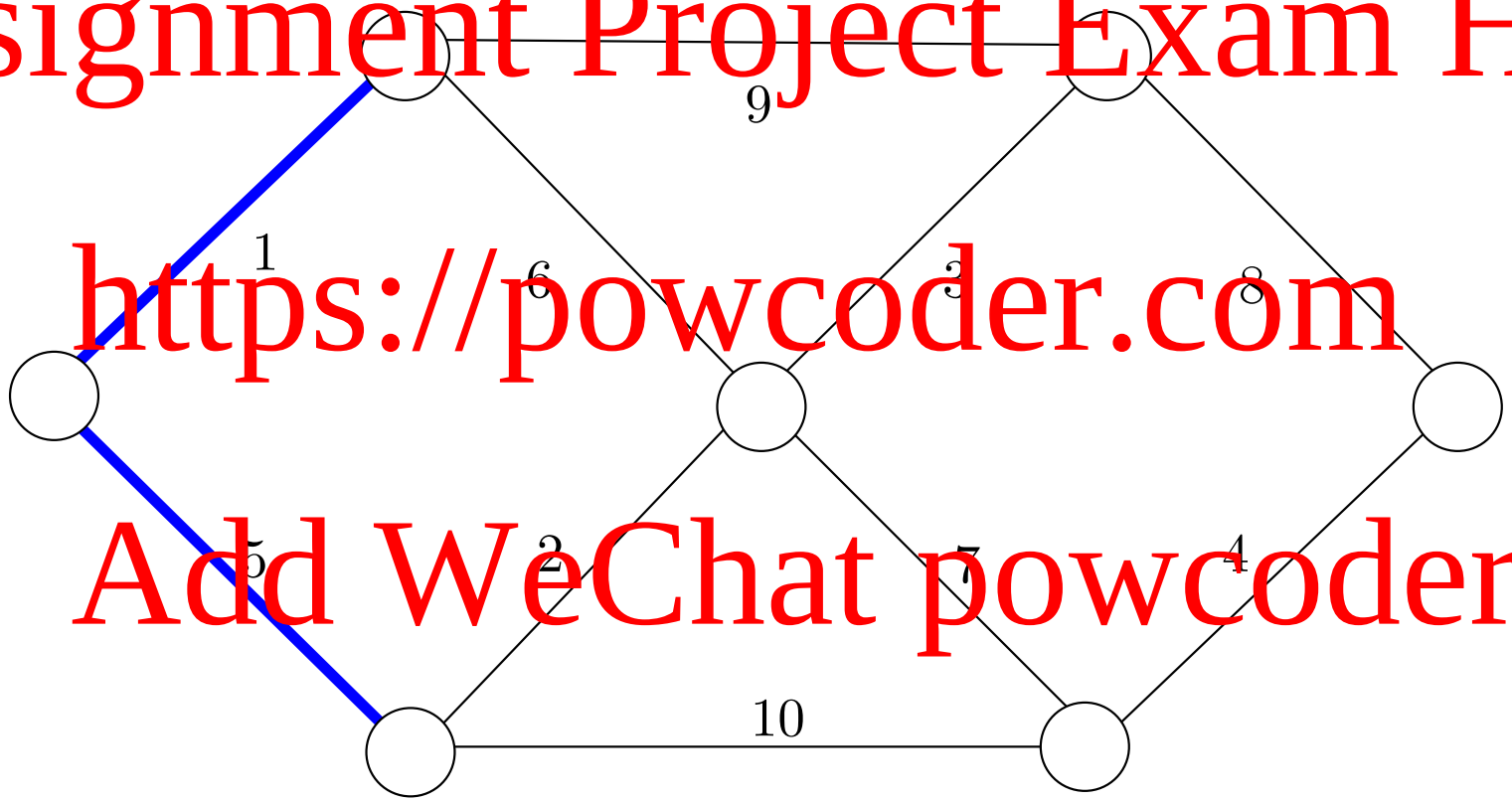
Prim



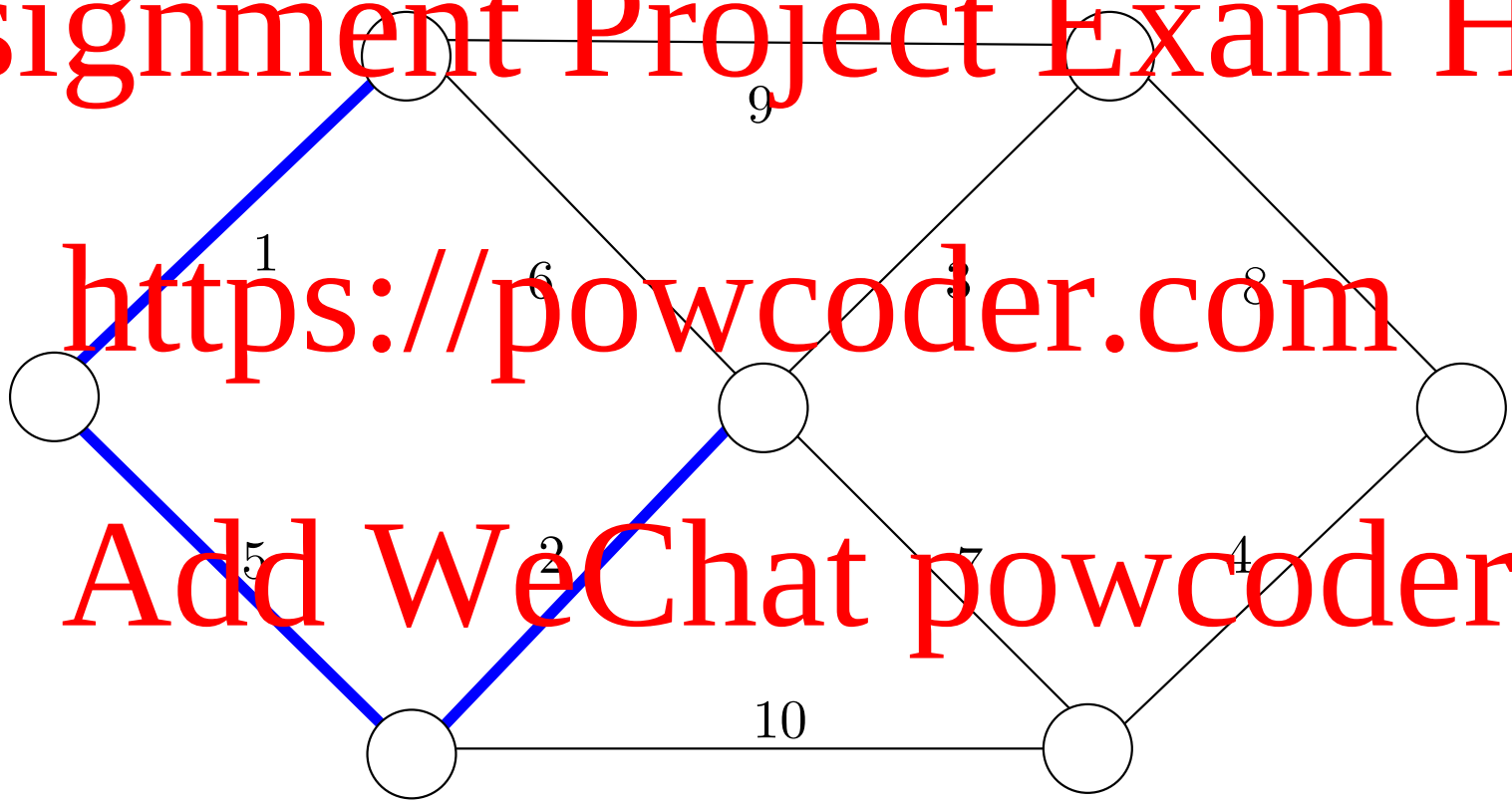
Prim



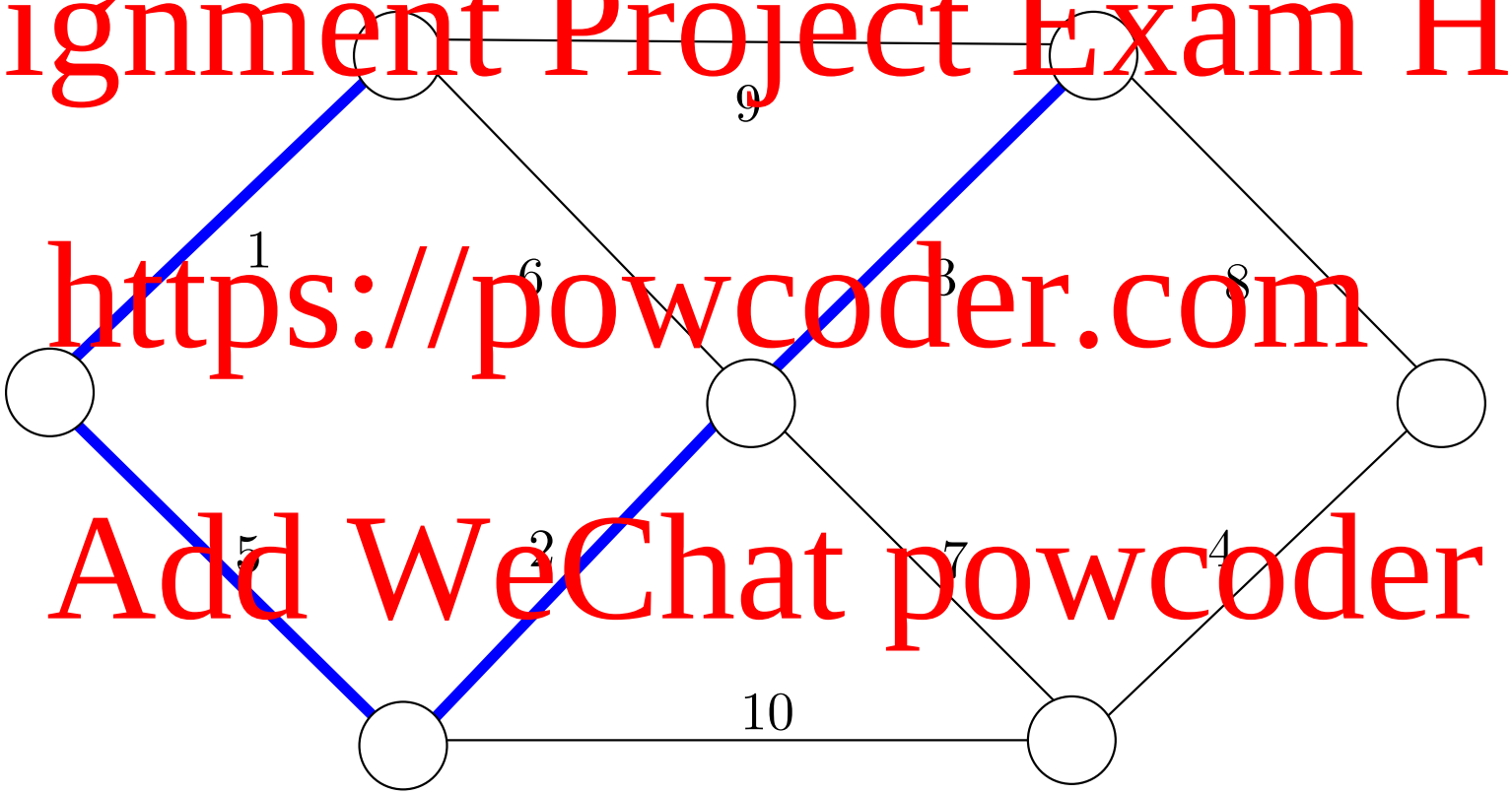
Prim



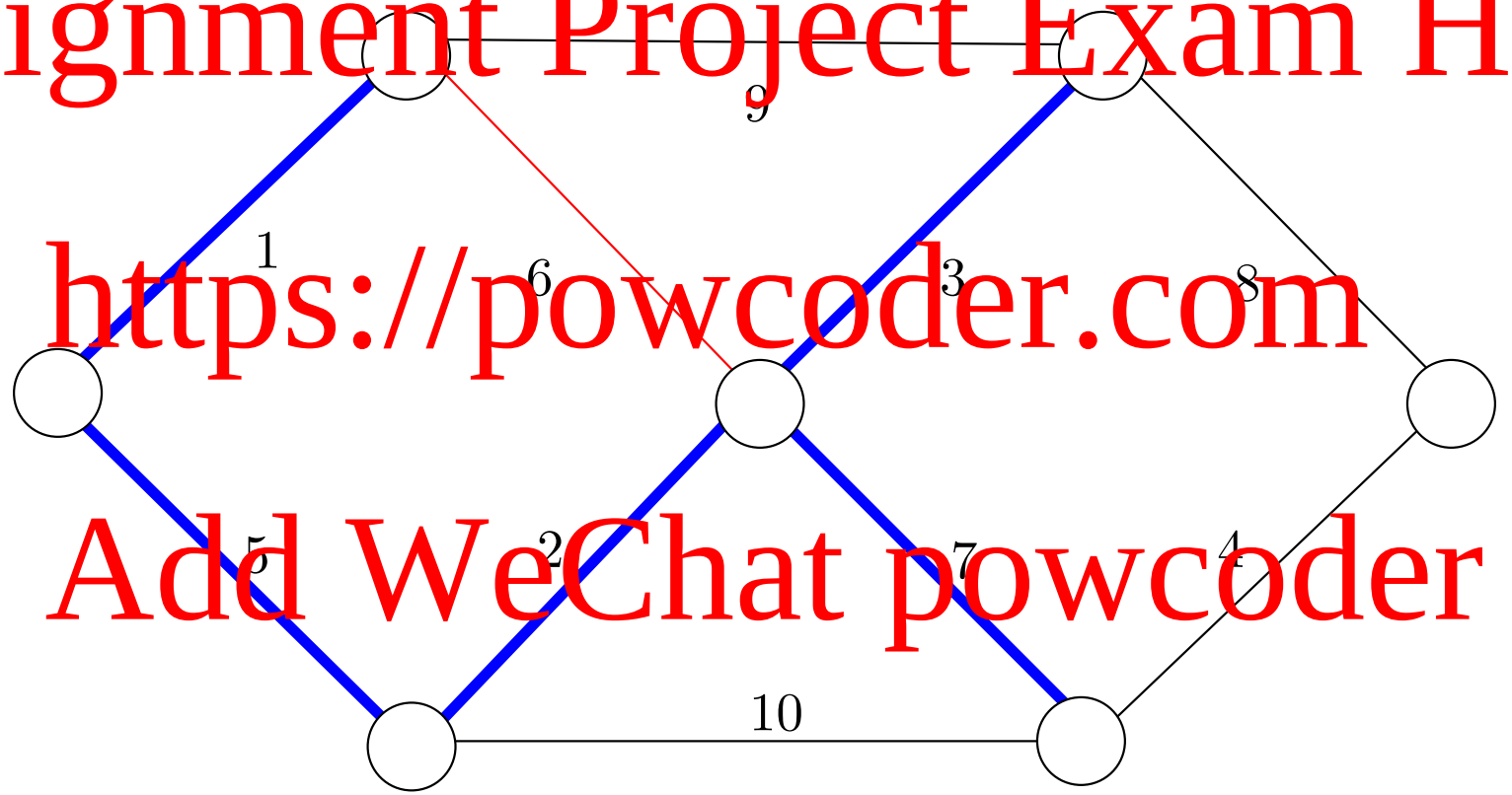
Prim



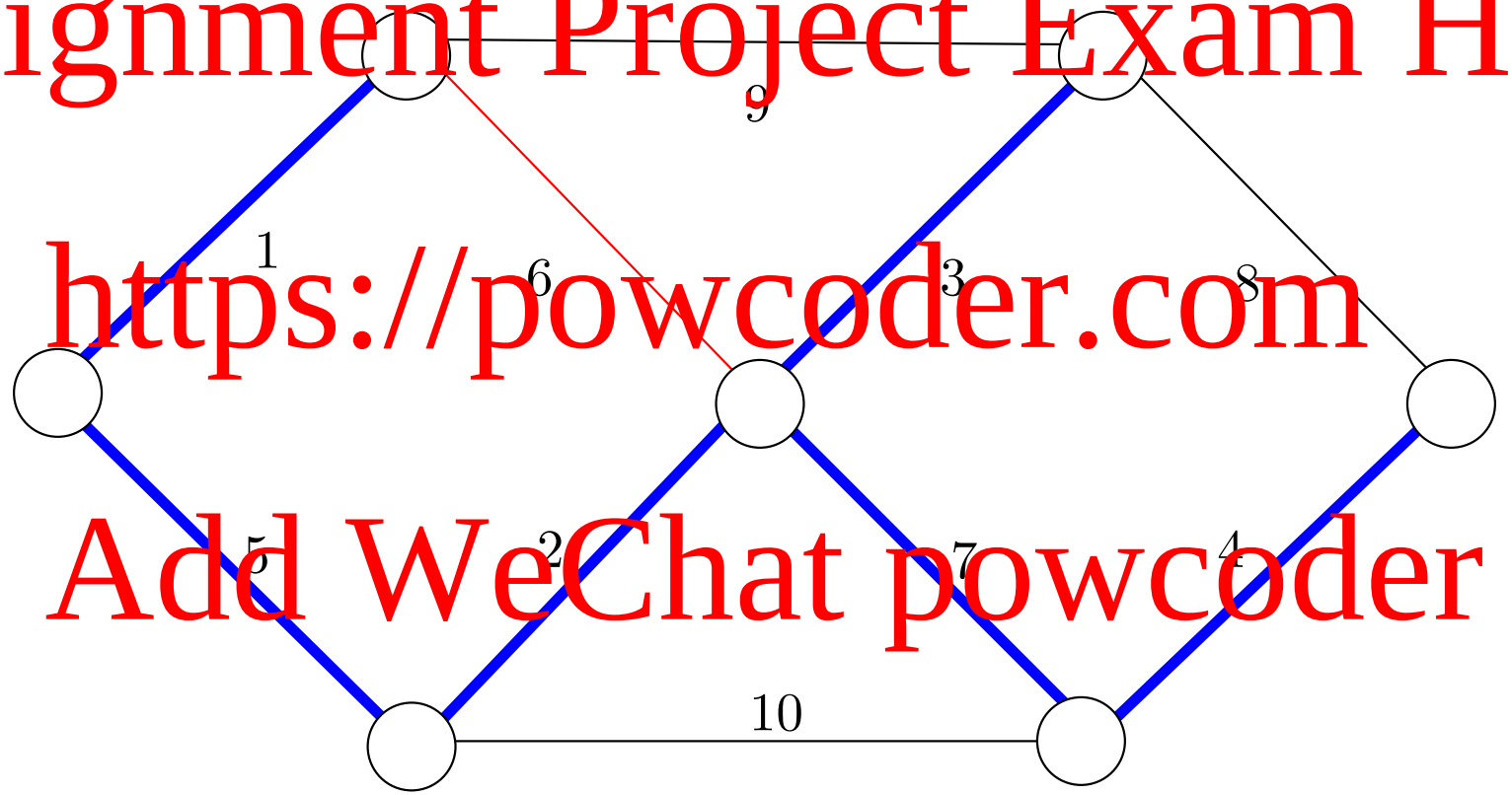
Prim



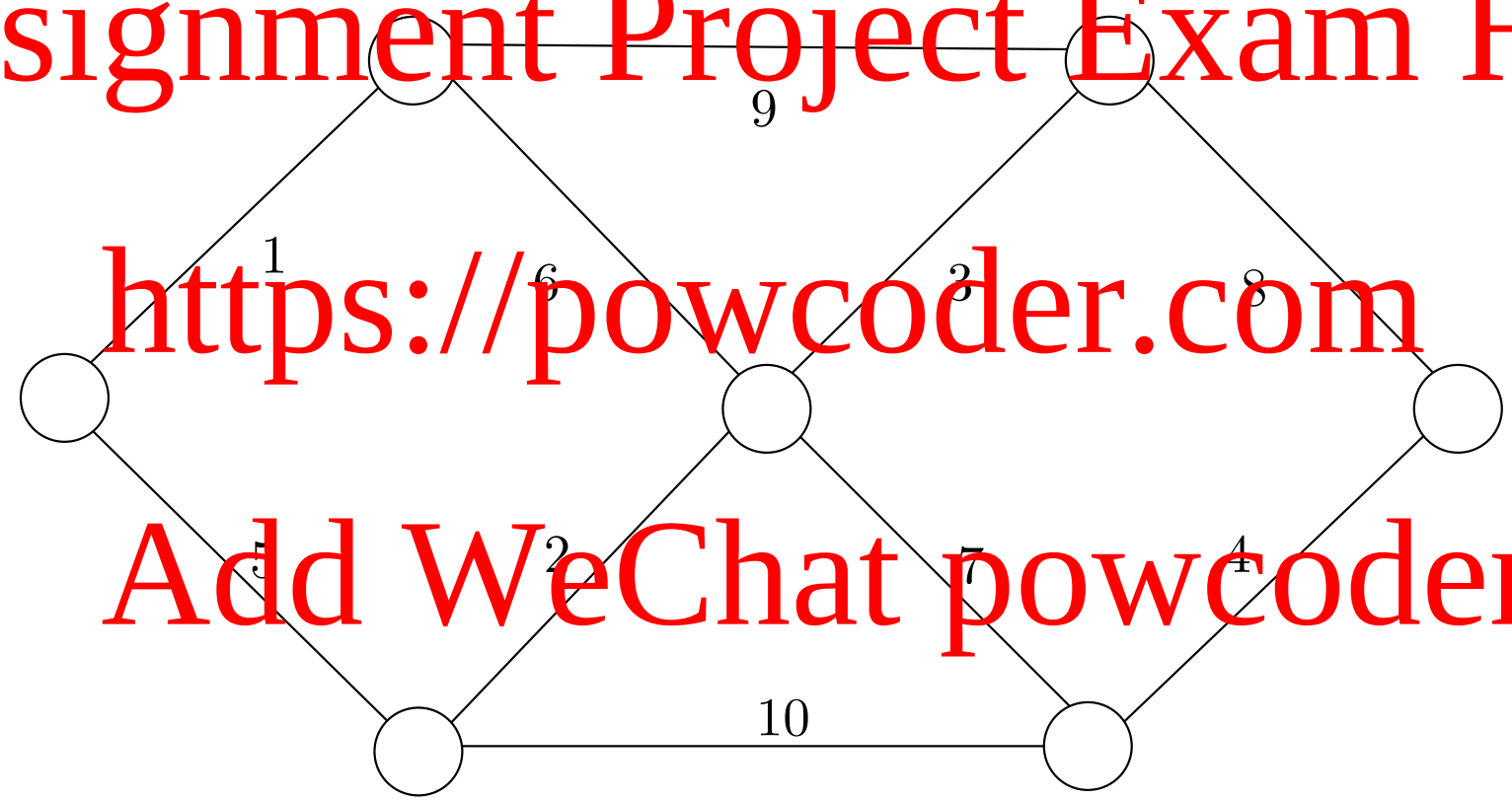
Prim



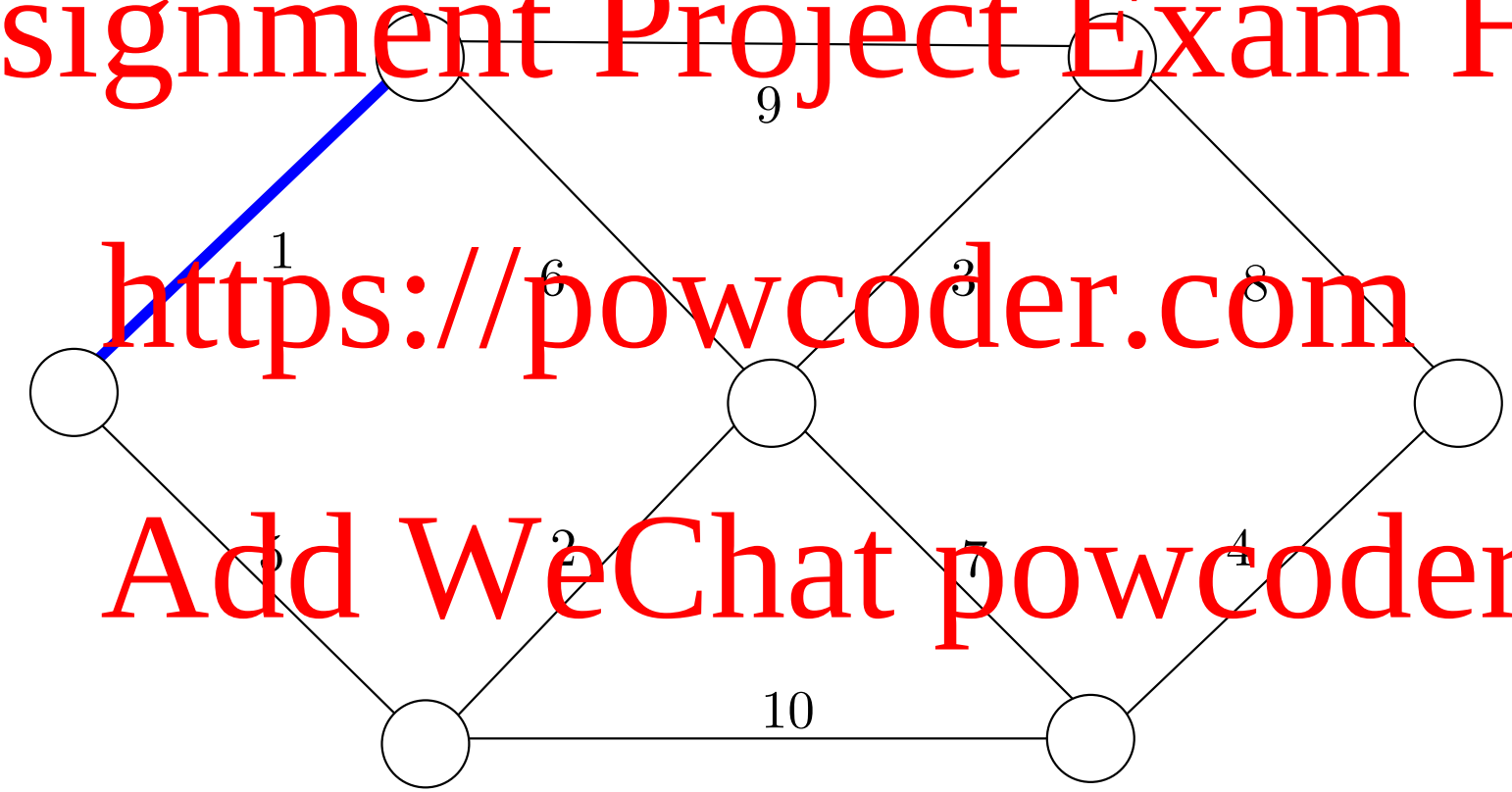
Prim



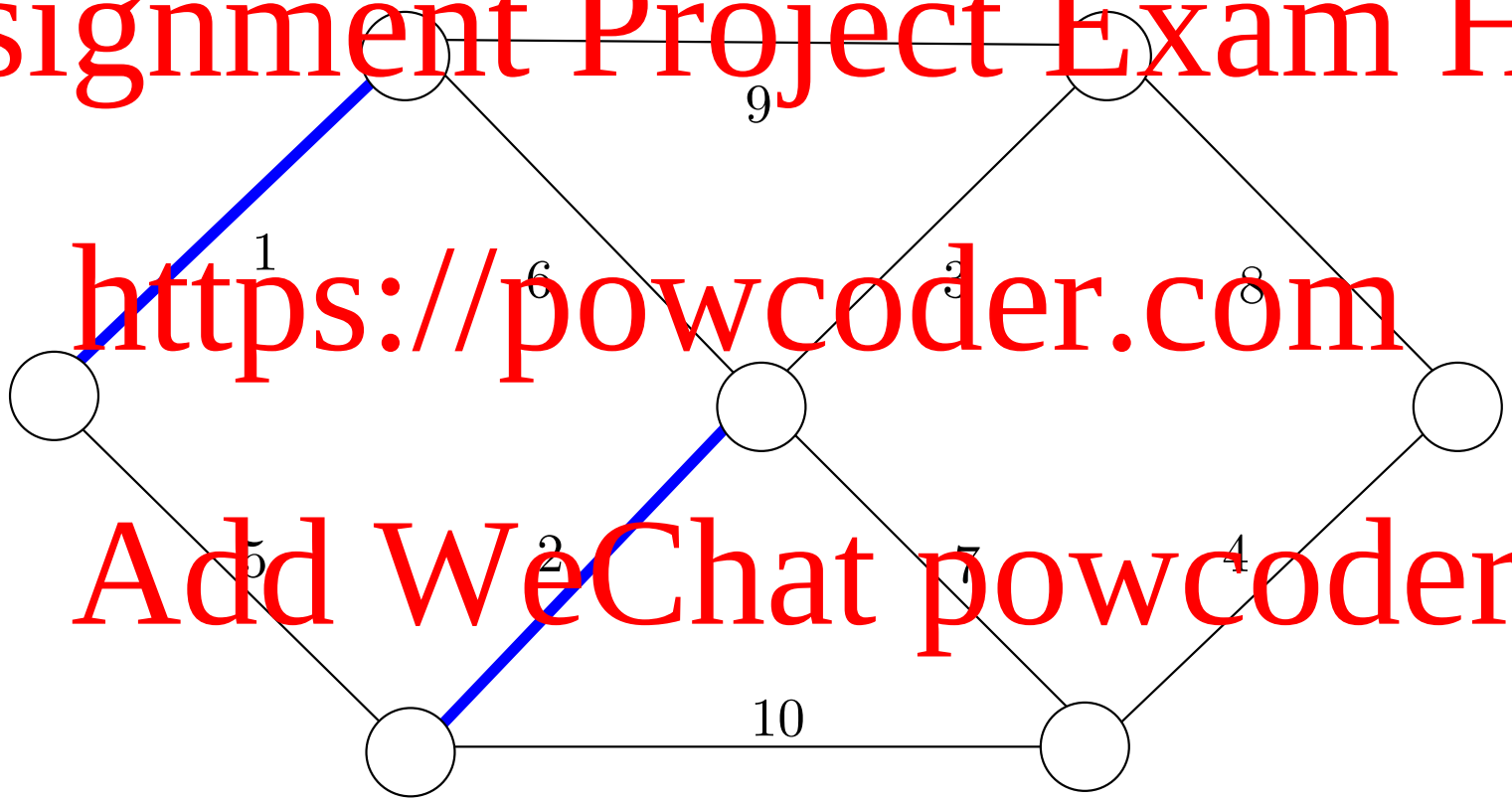
Kruskal



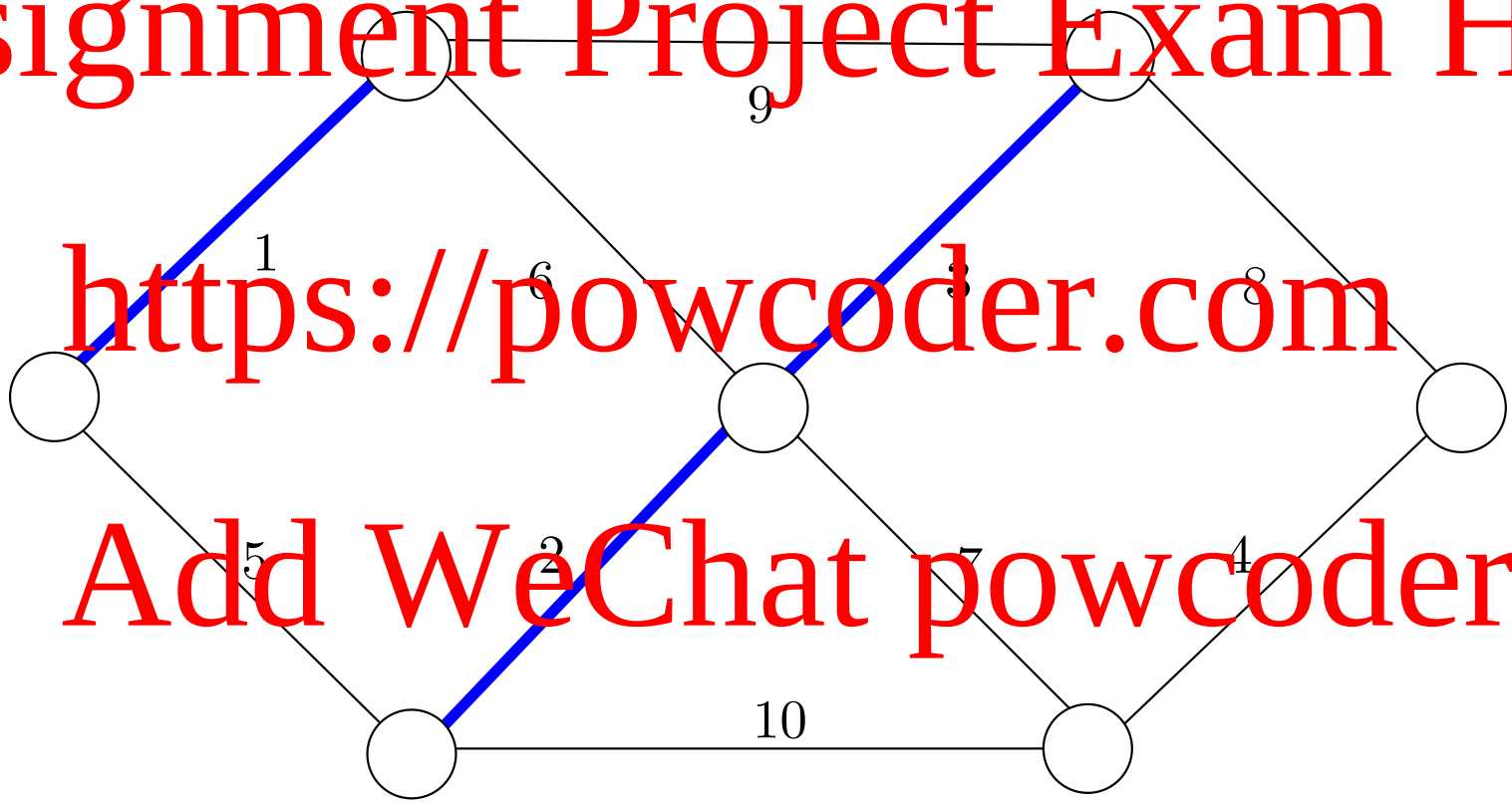
Kruskal



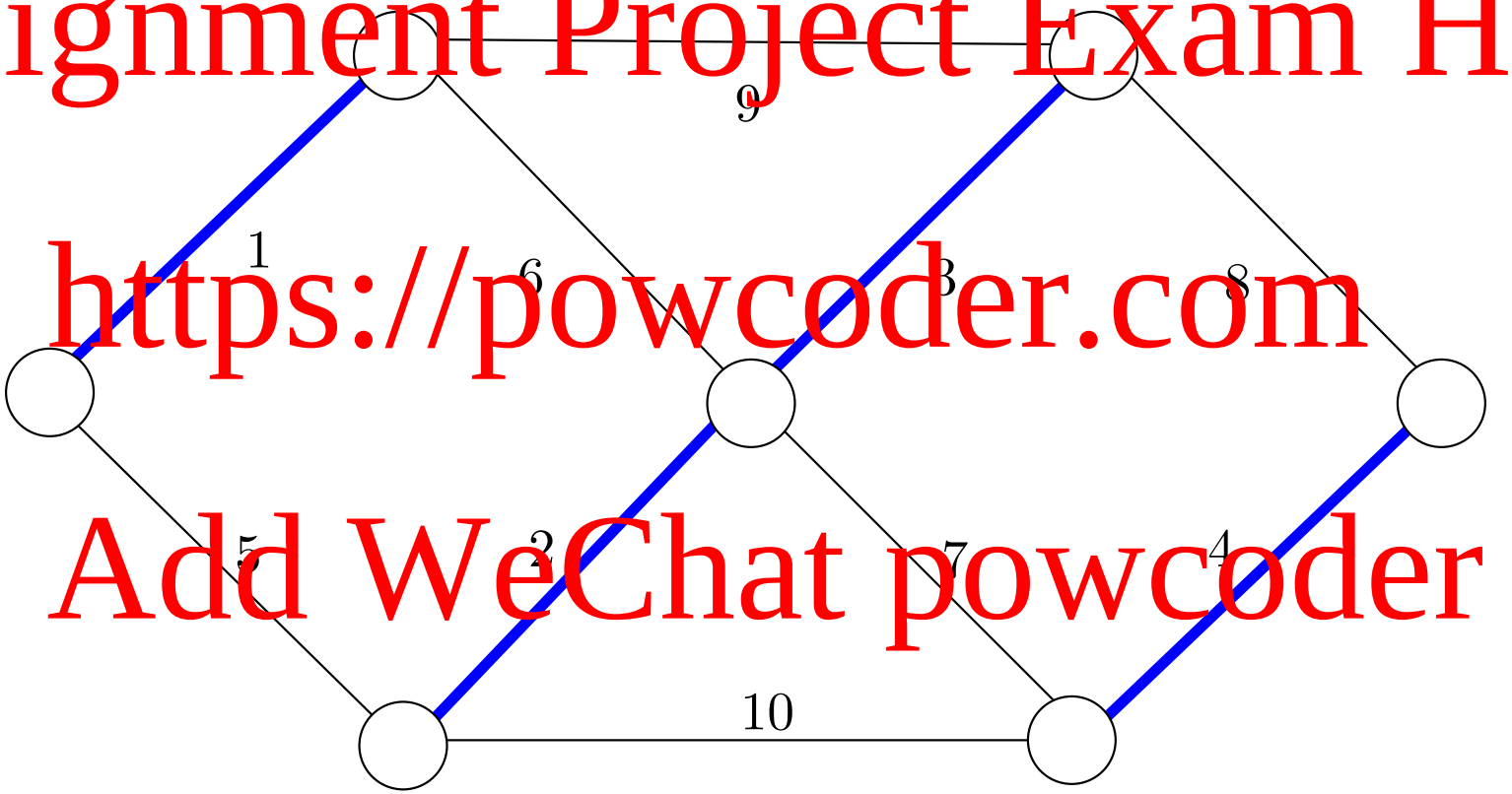
Kruskal



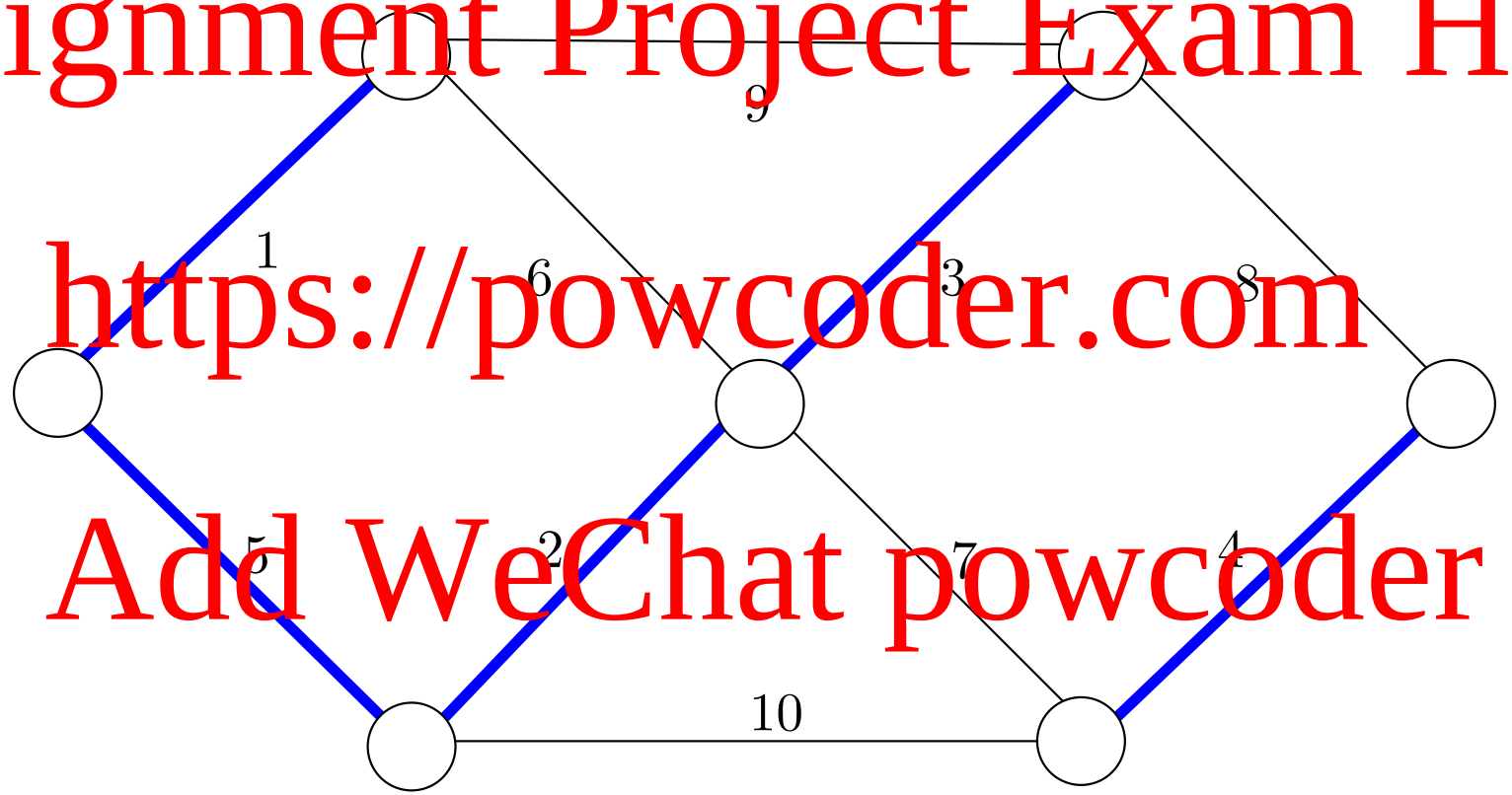
Kruskal



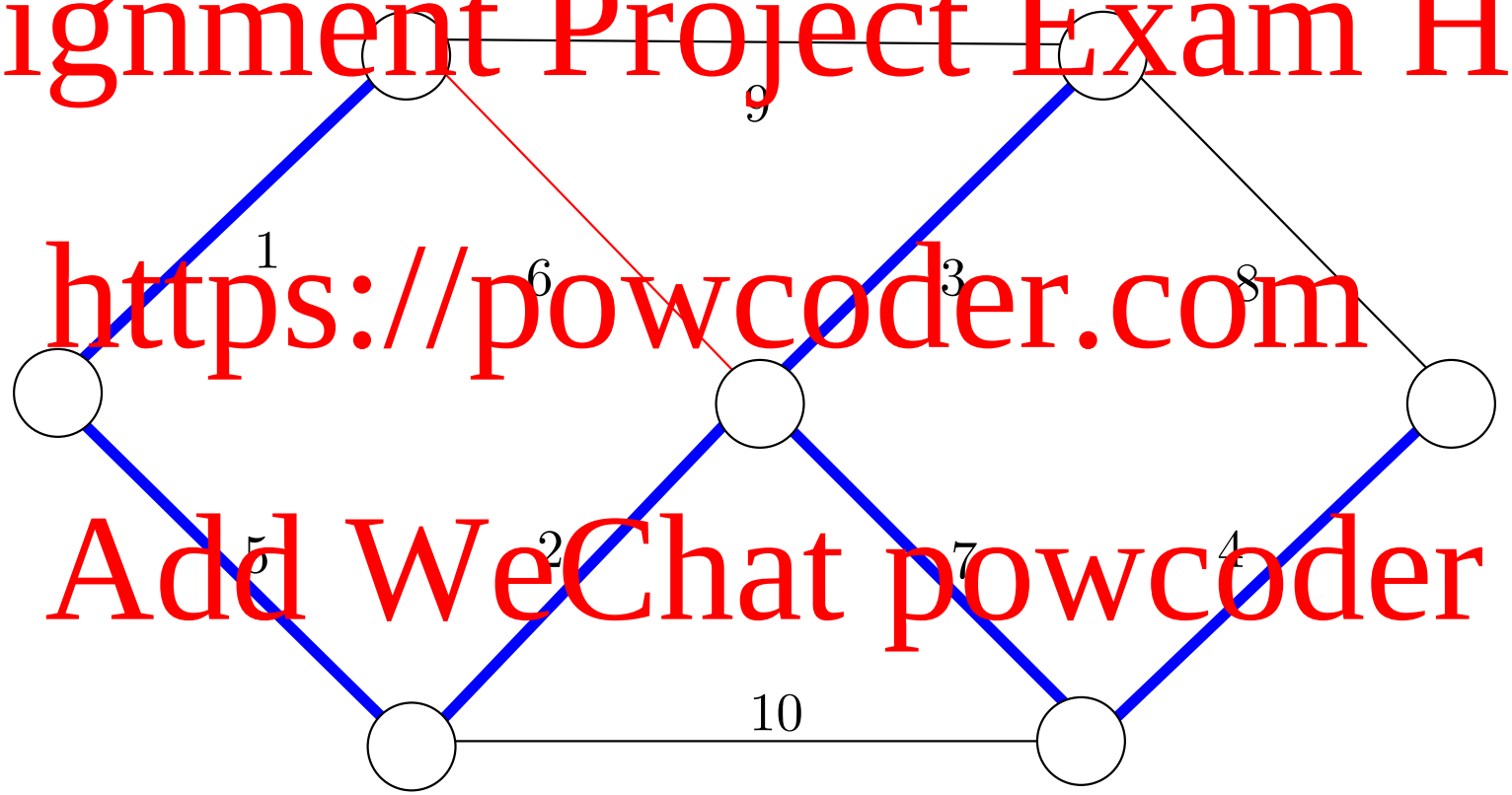
Kruskal



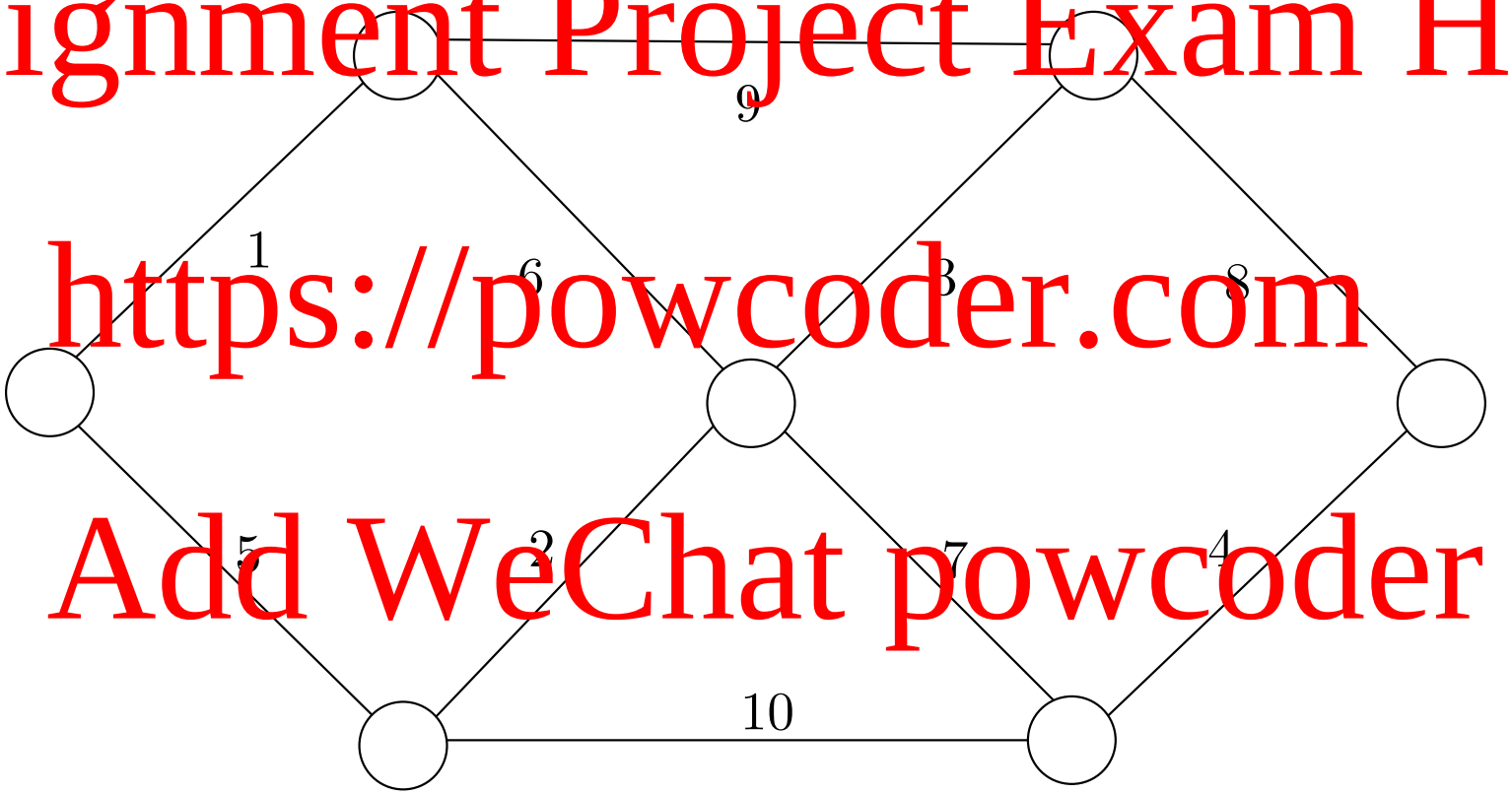
Kruskal



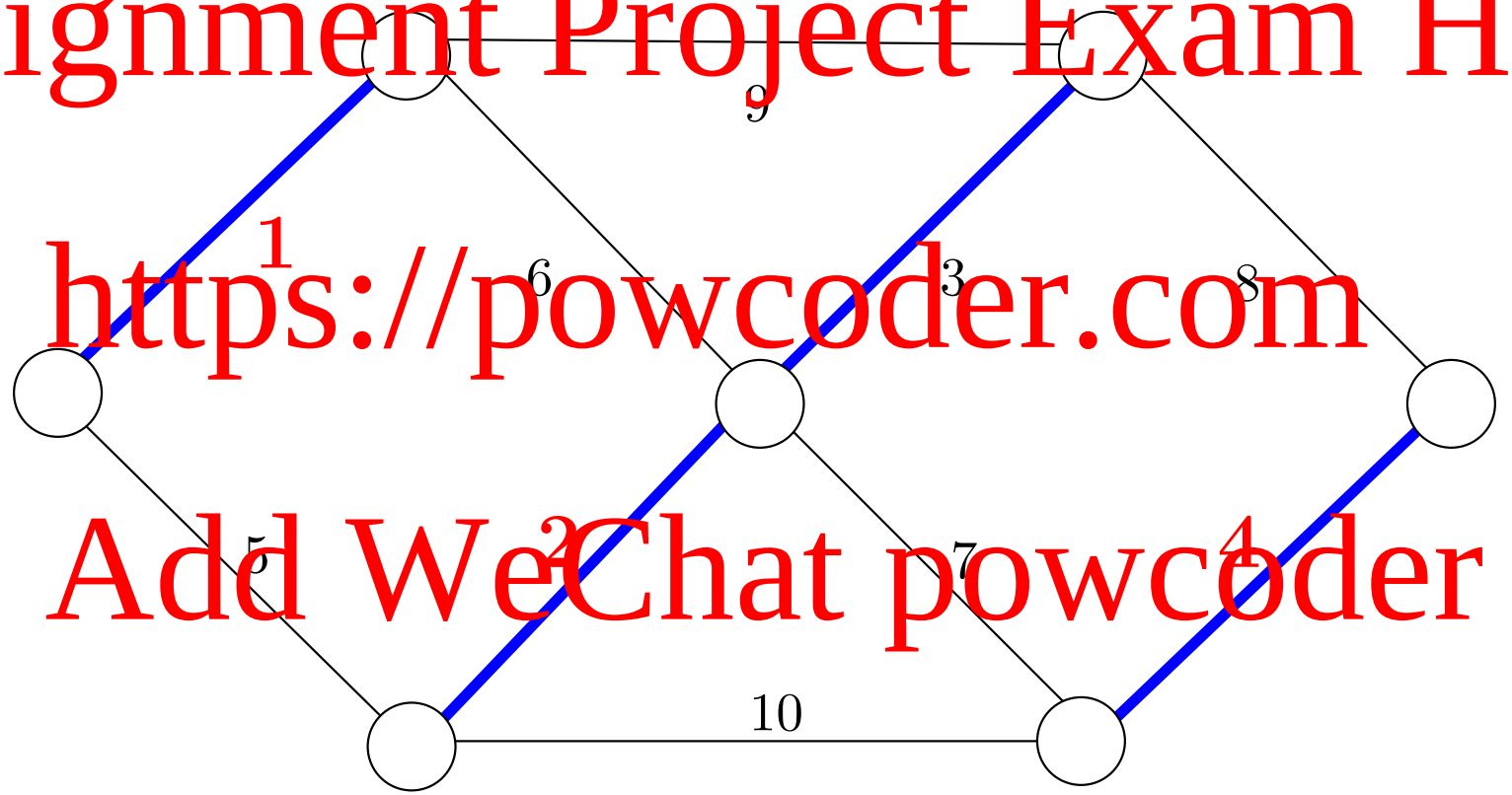
Kruskal



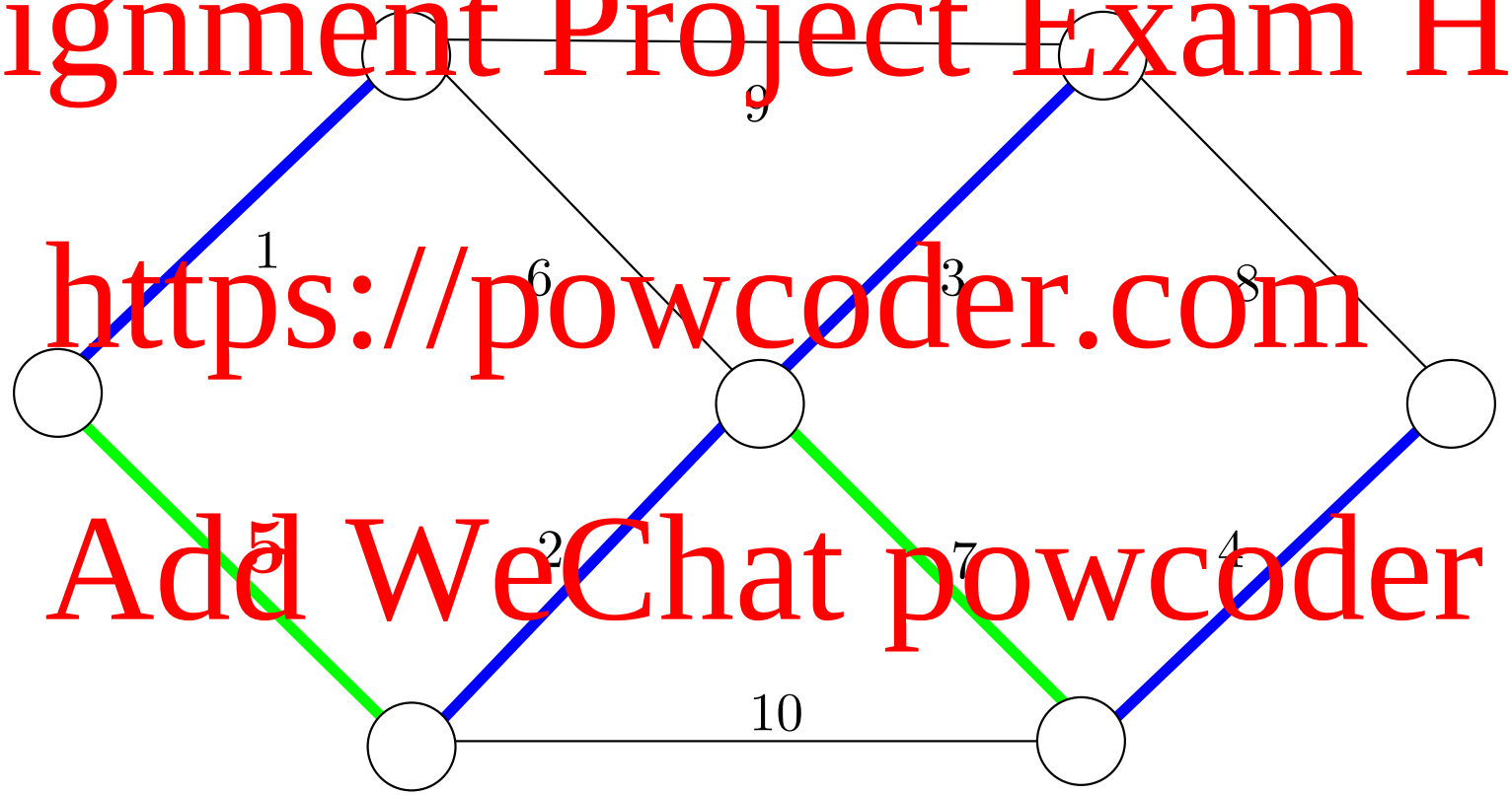
Borůvka – red: Common MWOE (min-weight out edge)



Borůvka – red: Common MWOE (min-weight out edge)



Borůvka – red: Common MWOE (min-weight out edge)



Prim and Kruskal as particular cases of Boruvka

- Kruskal is a restricted Boruvka, where we only use the overall lowest-cost MWOE (not necessarily common) - thus, at any given step, we only merge 2 trees (one single 2-way merge)
- Prim is a restricted Boruvka, where we only use the MWOE of the chosen tree - thus, at any given step, we merge the chosen tree with one node (a "level 0" tree)
- Boruvka deals with all MWOEs at the same "step" (called "level" in the distributed case) - thus, we can perform several multi-way merges "simultaneously" (at the same "step")
- Quotation marks indicate that many things may happen at the same "step" or "level" ...
- The distributed MST versions exploit this ability of Boruvka

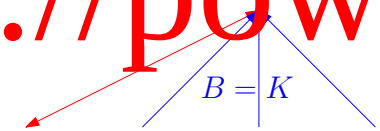
All unrooted trees (edge shapes) with 4 & 5 nodes

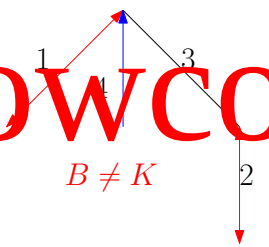
- Looking for MWEs (minimal working examples) where Kruskal is essentially different from Boruvka: 4 or 5 nodes?

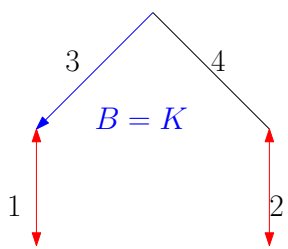
- Here the selected roots have minimum eccentricities
- WLG (without loss of generality), are these all that need consideration?

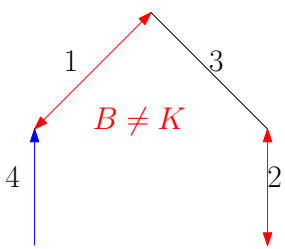
Five nodes – Kruskal essentially different from Boruvka?

- B step 1: collects all red (Common MWOE) and blue (one way MWOE) edges
- K collects in order: red edges (costs 1 ,2), then costs 3, 4
- Figure codes: $B \neq K$: essential differences, $B = K$: NO essential differences



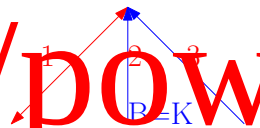


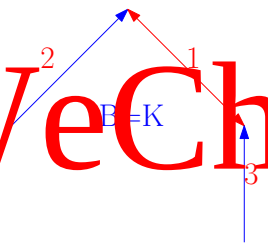


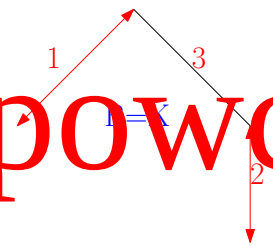


Four nodes

- On 4 nodes, Kruskal is NOT essentially different from Boruvka







Further discussions

Prove, for Borůvka's algorithm:

- In any multi-way merge there is always one Common MWOE
- The Common MWOE is unique

Prove that these hold for:

- Level 0 trees, i.e. initial nodes (size 1 trees)
- Level k trees, for any $k \geq 0$

$\exists!$

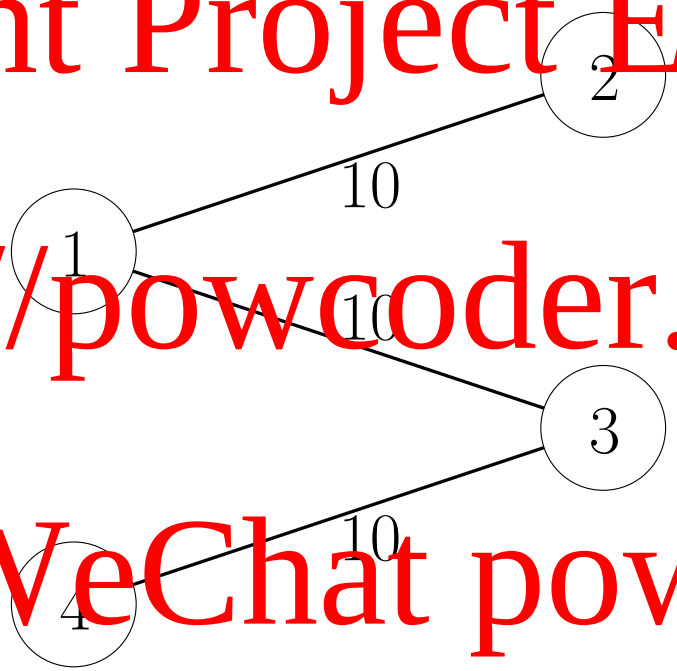
Distributed MST (Sync)

- Based on Borůvka; see Lynch §4.4
- Nodes have unique IDs
- Nodes know their adjacent neighbours

- Nodes know the graph size, N (required for synchronisation)
  - could be obtained by a preliminary phase, based on Echo

- Edges have unique weights or same weight edges can be ordered
  - e..g, by lexicographical comparisons, where each edge $\{i, j\}$ is represented by an ordered triple $(w, v, v'), v < v'$, where $w =$ edge weight, $v, v' =$ ID's of i, j

Sync MST – Comparing weights with equal weights



$$\{1,2\} = (10,1,2) < \{1,3\} = (10,1,3) < \{4,3\} = (10,3,4)$$

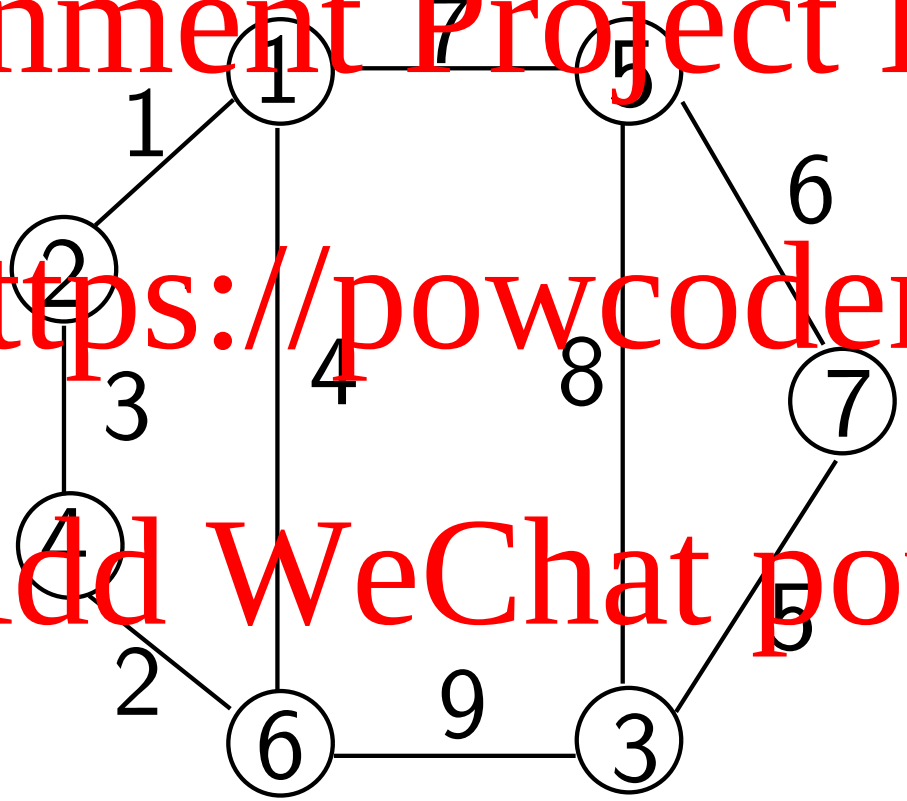
Distributed MST (Sync)

- Time complexity: $O(N \log N)$
- Message complexity: $O((N + M) \log N)$
- Levels: $O(\log N)$; each level defines a spanning forest
- Level 0 components are the individual nodes
- Level $k+1$ builds larger components by merging 2 or more level k components into new components
- Thus, level $k \geq 0$ has component subtrees of size 2^k at least
- Each component has a distinguished leader; the leader ID identifies the component
- For connected graphs, the algorithm ends with a MST (unique, if edges have different weights)

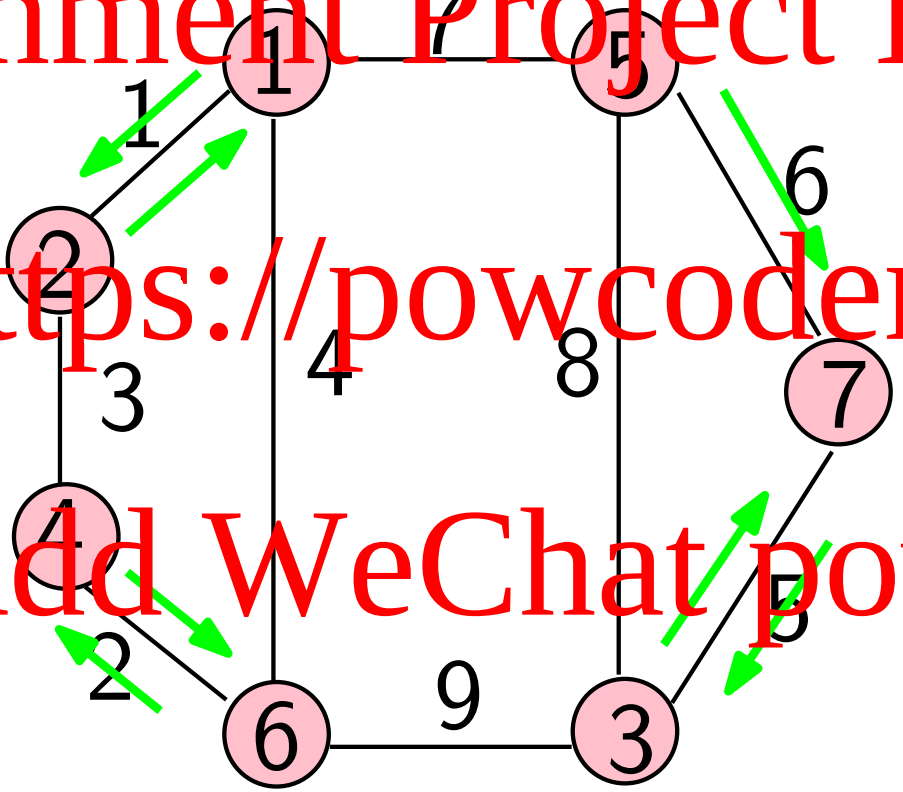
Distributed MST (Sync)

- Details may vary, with slightly different performance...
- The following diagrams use a set of suggestive, but not exactly necessary, messages
  - *initiate*
  - *test*
  - *accept*, *reject*
  - *report*
  - *connect* (implies a *change-root*), *connect!*
- To ensure correct component identification, each level $k > 1$ takes a predefined number of steps, $O(N)$ – nodes may need to stay idle until this count is completed
  - depending on the actual algorithm details, this may happen in different ways

Sync MST – Level 0 Components

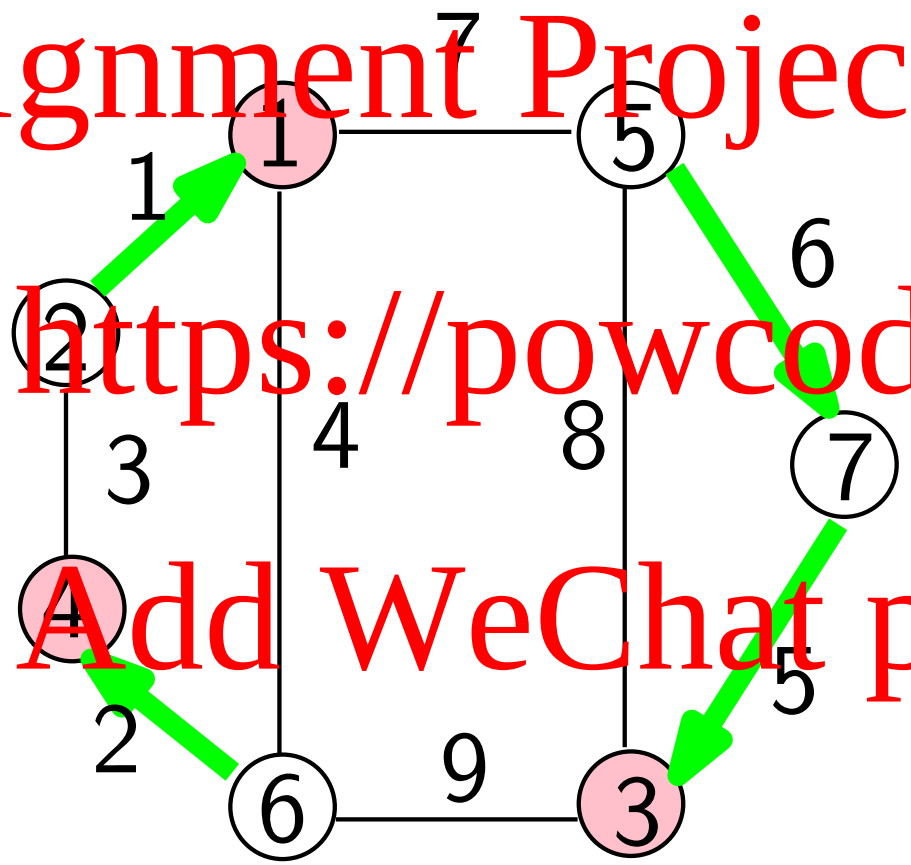


Sync MST – Level 0 Components

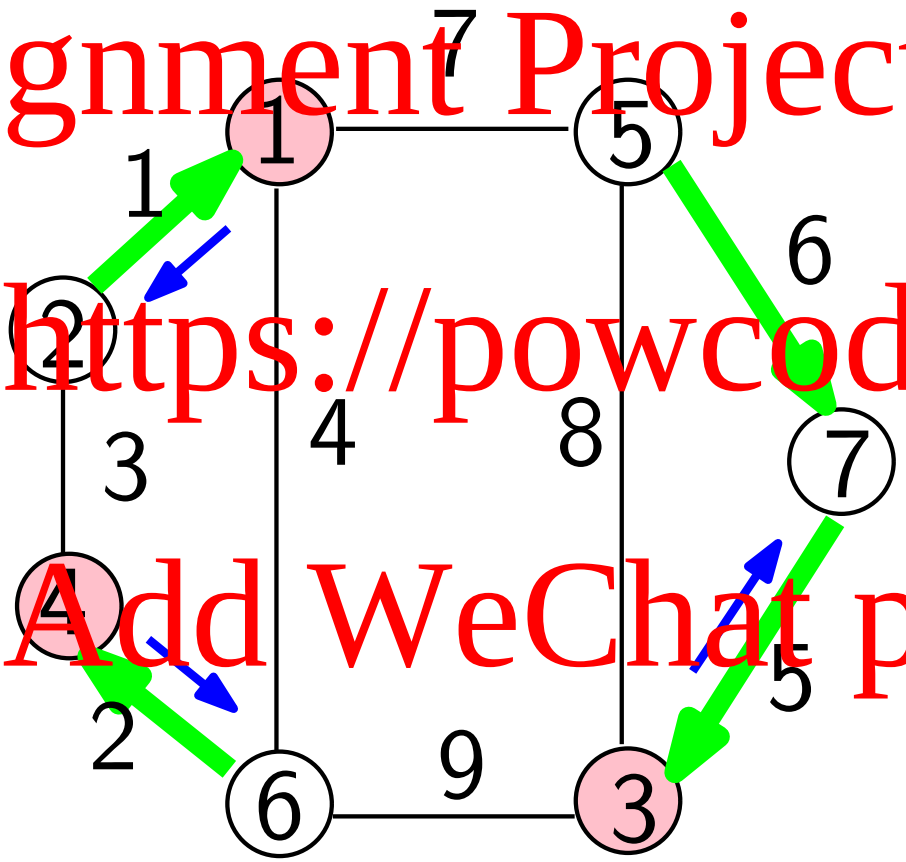


connect!

Sync MST – Level 1 Components

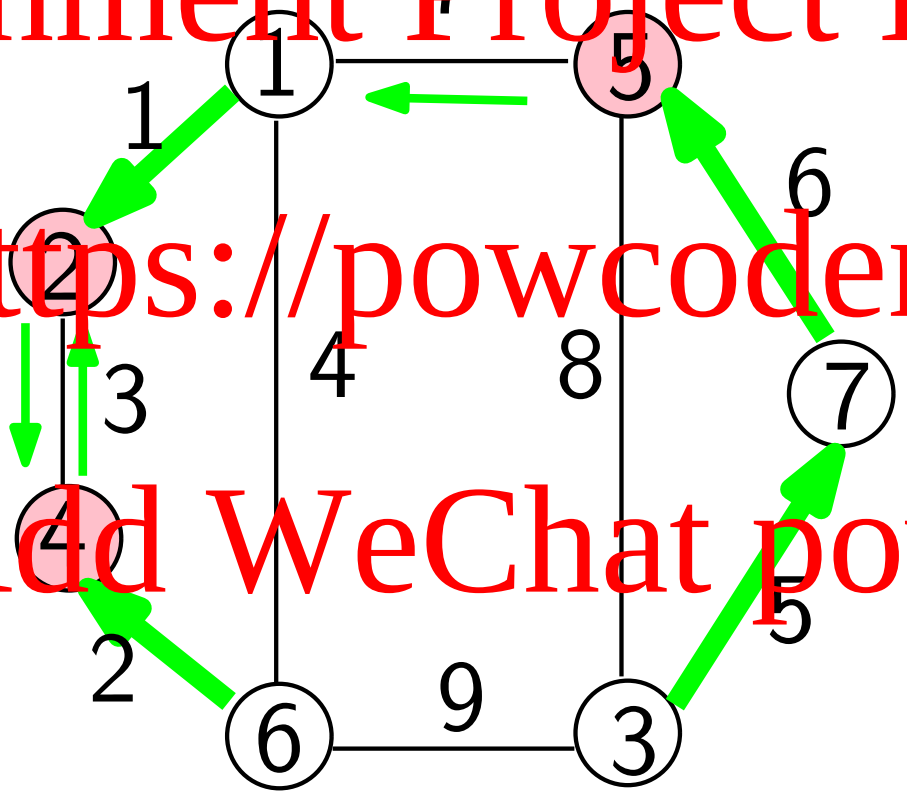


Sync MST – Level 1 Components



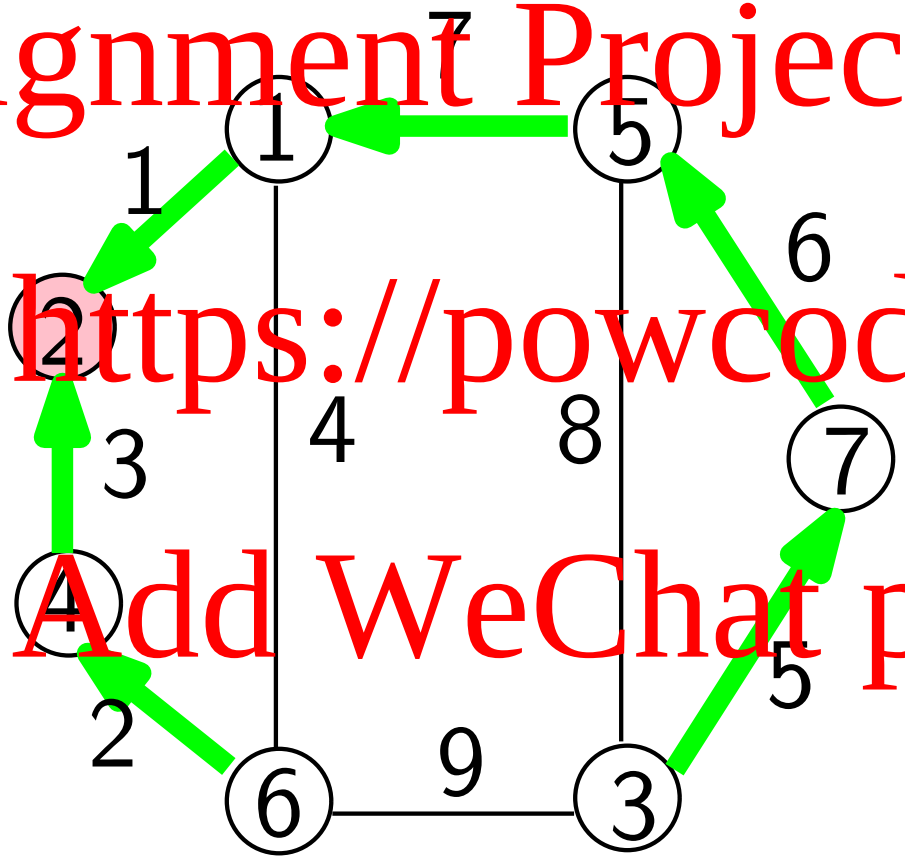
initiate

Sync MST – Level 1 Components

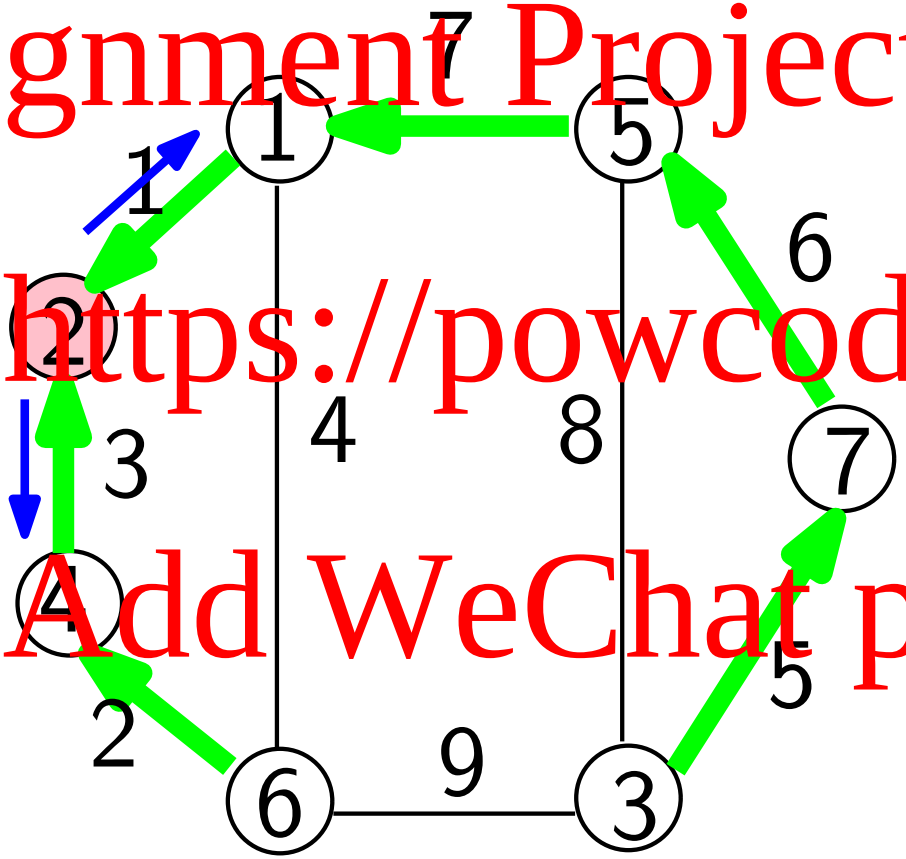


connect!

Sync MST – Level 2 Components

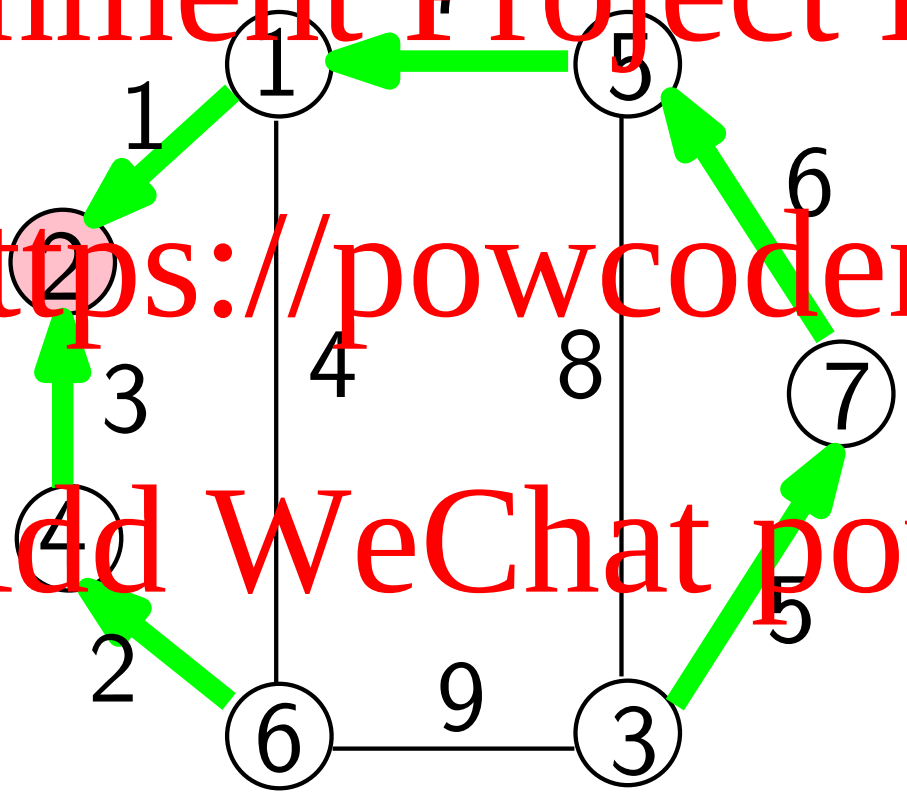


Sync MST – Level 2 Components



initiate

Sync MST – Level 2 Components



Sync MST – Memento

- Distributed Sync MST is based on Borůvka
  - Borůvka can be viewed as extension of both Prim and Kruskal
  - At each given step: Prim grows one single tree; Borůvka grows all trees simultaneously
  - At each given step: Kruskal merges exactly two trees; Borůvka merges all trees simultaneously (2-way or n-way unions)
  - The steps of sequential Borůvka correspond to more complex levels (phases) in distributed Sync MST

Sync MST – Memento

- The MST is unique, if weights are pairwise distinct (but this can always be arranged, e.g. by lexicographic comparisons)
- There is one single common MWOE, aka the core, in all tree unions in the algorithm (obvious for 2-way unions, but needs proof for more)
- The leader of a union of trees is one of the two endpoints of the core, i.e. of the single common MWOE (this also ensures that the root stays reasonably close to the leaves)
- A level k union tree has size $\geq 2^k$, thus tree sizes grow exponentially and there are at most $\log N$ phases
- To ensure required synchronicity, each level requires $O(N)$ steps (this is ok, as there are only few levels)

Sync MST – Memento

- The accept and reject messages are not really needed; i.e. the test messages may hold enough info for this decision
- The report messages are evaluated on-the fly, at each node, and only the best MWOE's details are forwarded up, towards the leader
- The report messages leave behind a route to the node holding the best MWOE of that subtree
- The connect messages are routed on the tree path which goes from the leader to the node holding the overall best MWOE
- The connect messages reshape the tree, by transporting the leadership, i.e. resetting parent and child pointers along their path

Async MST – GHS and variants

Specific difficulties of the async version - not present in the sync version:

- Two neighbours may be part of the same component tree, but they have not learned this yet (logical error)
- Not all component trees may have a guaranteed size $> 2^k$; some may grow much faster than others (complexity issue)
- More generally, component trees may be at different levels... (logical and complexity issues)

Read more in Lynch's textbook:

- §4.4 : sync GHS
- §15.5 : async GHS, plus summary revision of sync GHS
- §15.3 : async STtoLeader (on unrooted STs)