CS302 - Data Structures

using C++

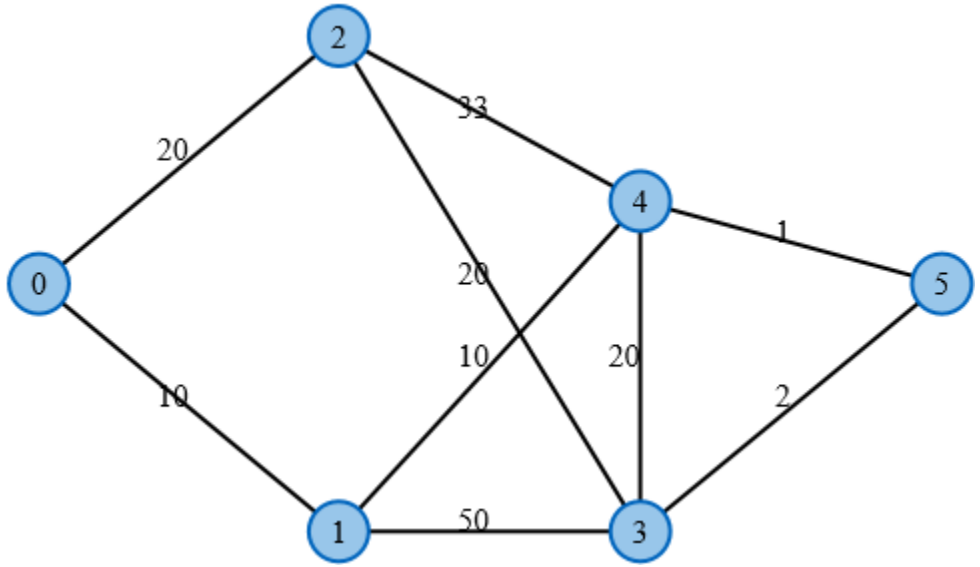
Topic: Minimum Spanning Tree

Kostas Alexis

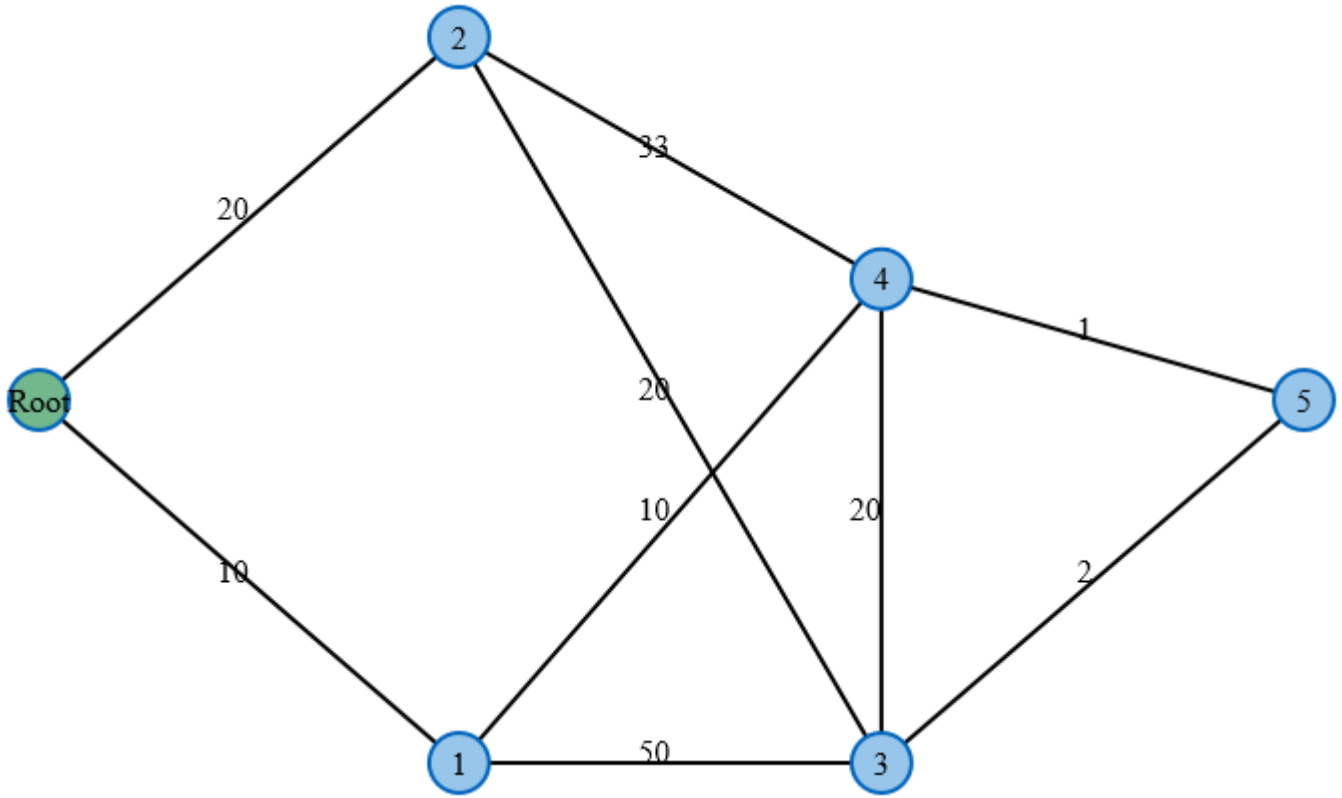
The Minimum Spanning Tree Algorithm

- A telecommunication company wants to connect all the blocks in a new neighborhood. However, the easiest possibility to install new cables is to bury them alongside existing roads. So the company decides to use hubs which are placed at road junctions. How can the installation cost be minimized if the price for connecting two hubs corresponds to the length of the cable attaching them?
- Considering the roads as a graph, the above example is an instance of the Minimum Spanning Tree problem.
- To solve such problems we can use Prim's Algorithm

Consider a Graph



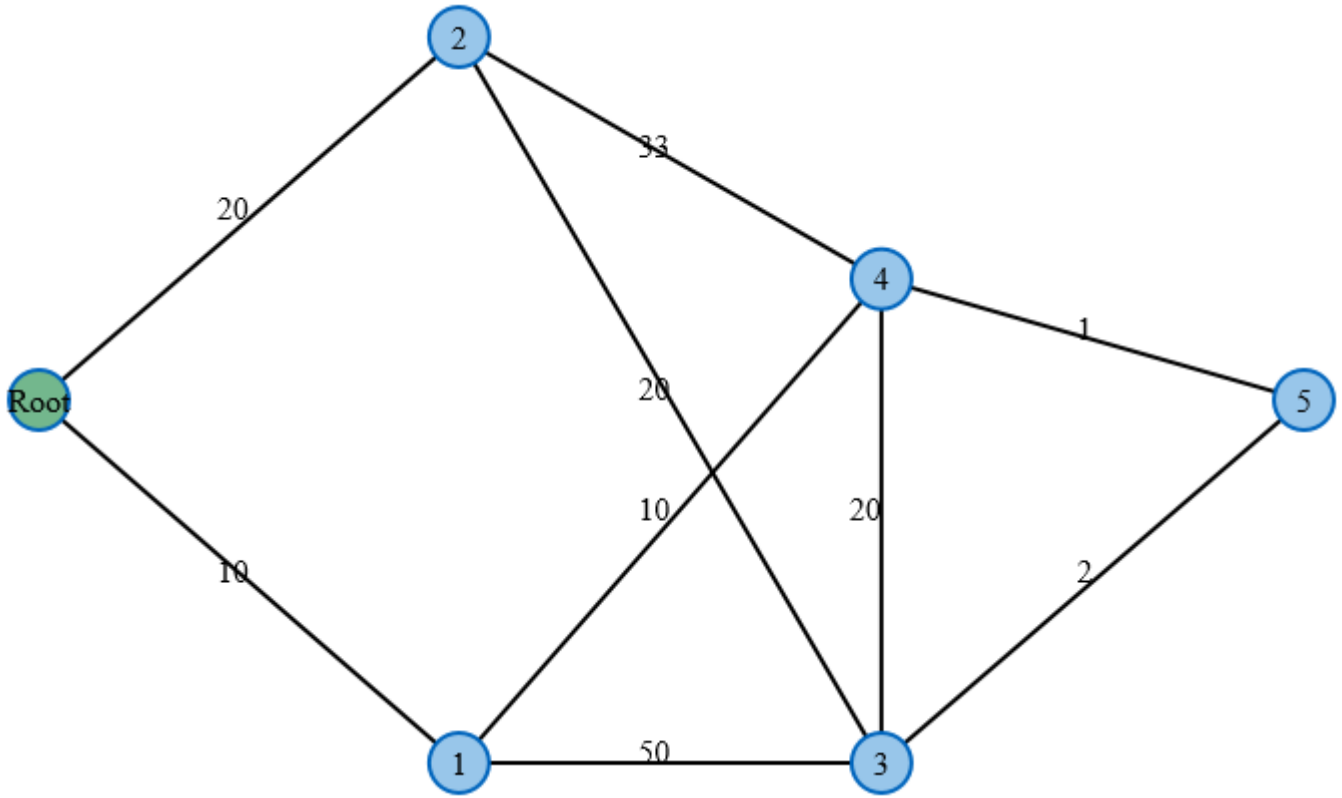
The Minimum Spanning Tree Algorithm



- Choose a node as the Root (e.g., “0”)

Based on the online tool: https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html

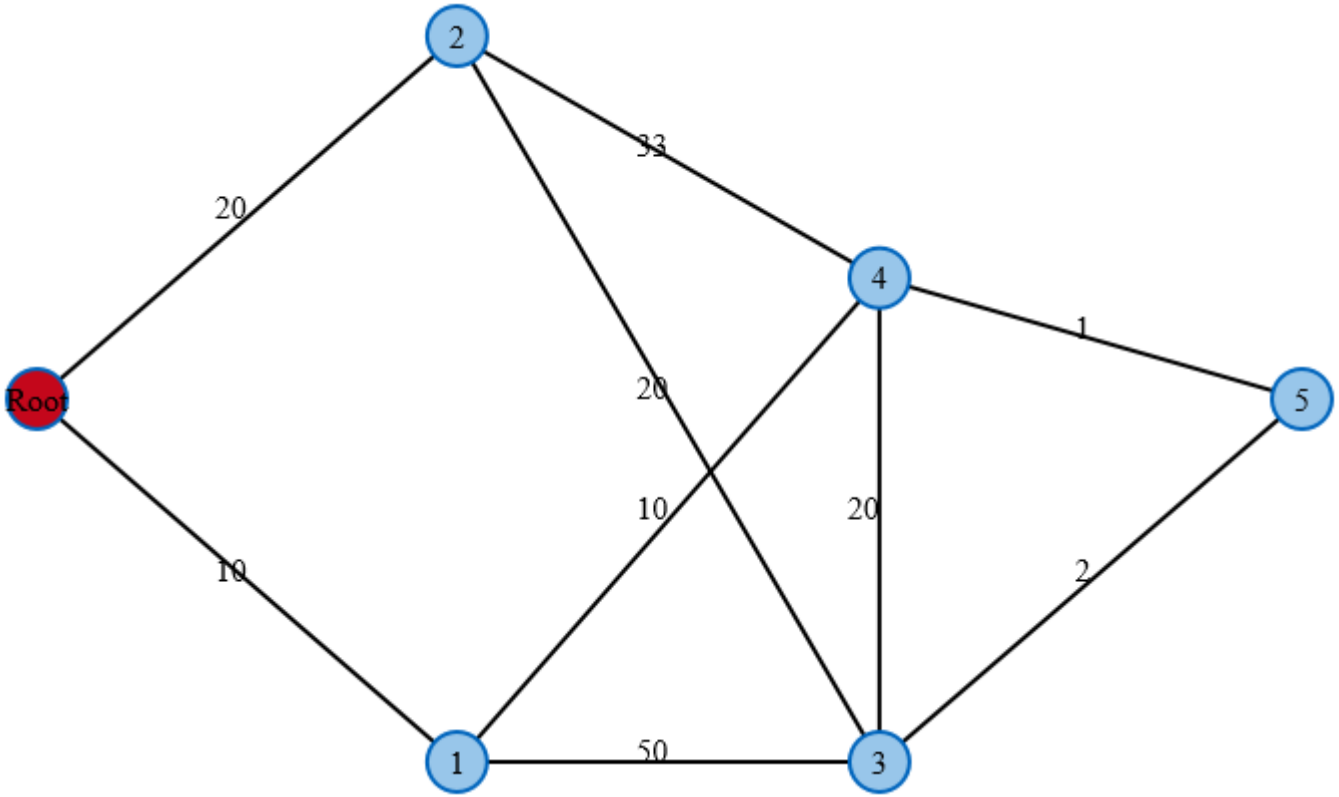
The Minimum Spanning Tree Algorithm



- Initialization
  - The MST to be constructed is still empty, hence the distance value for all vertices is set to INFINITY, and none of the vertices has a parent in the tree.
  - The start node is set as the root of the MST and is added to the queue. The distance value for root is 0 and its parent is set to itself.

Based on the online tool: https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html

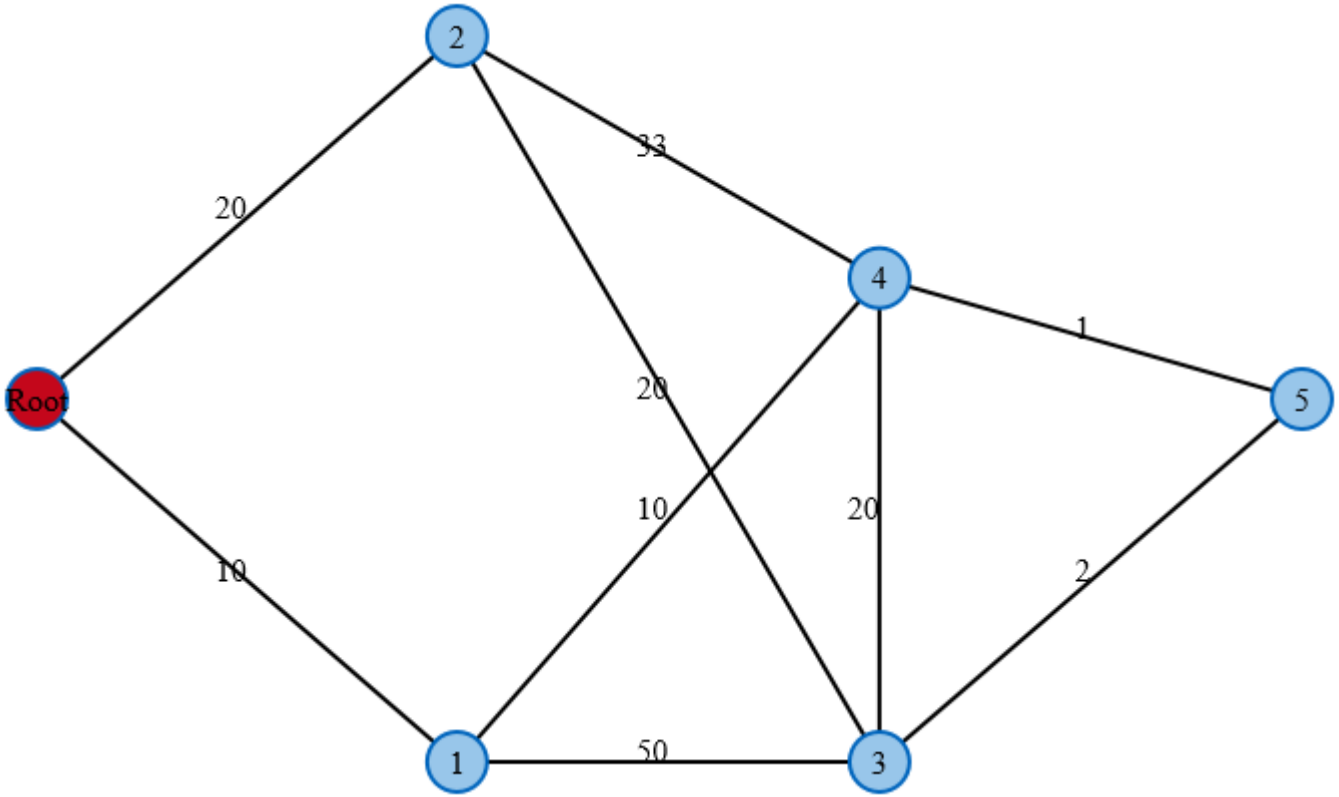
The Minimum Spanning Tree Algorithm



- Process next node
  - The node with the smallest distance value is removed from the queue, and the edge connecting it to the MST is added to the resulting set. The only exception is the root node which has a parent value set to itself.
  - As this node is being further processed, it is colored red.
  - The edges added to the tree are shown as green.

Based on the online tool: https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html

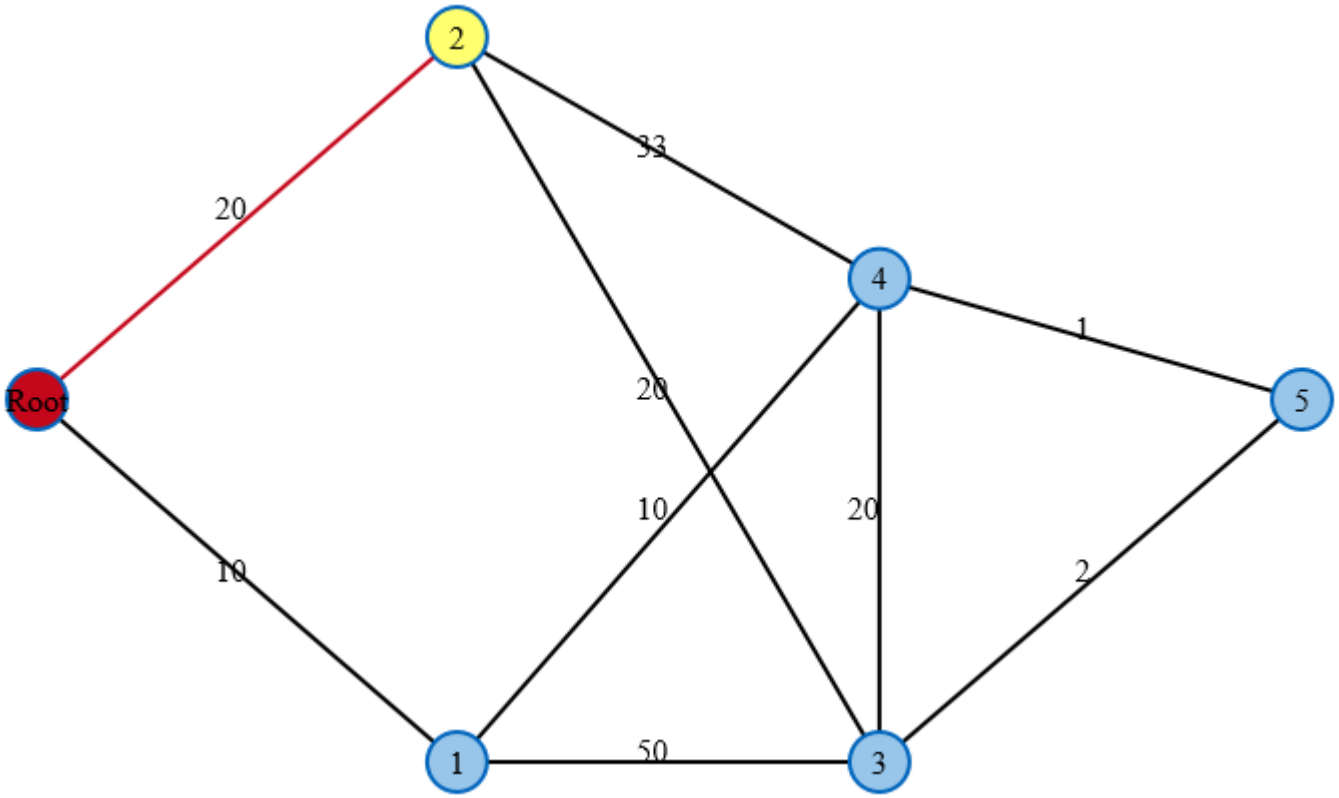
The Minimum Spanning Tree Algorithm



- Considering the neighbor
  - Each neighboring node is processed for possible modifications
  - There are two cases in which the node is not further processed. First is the case when the node has been added to the MST before. Second is the case when the node is in the queue, but the previous edge connecting it to the MST has a lower weight.

Based on the online tool: https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html

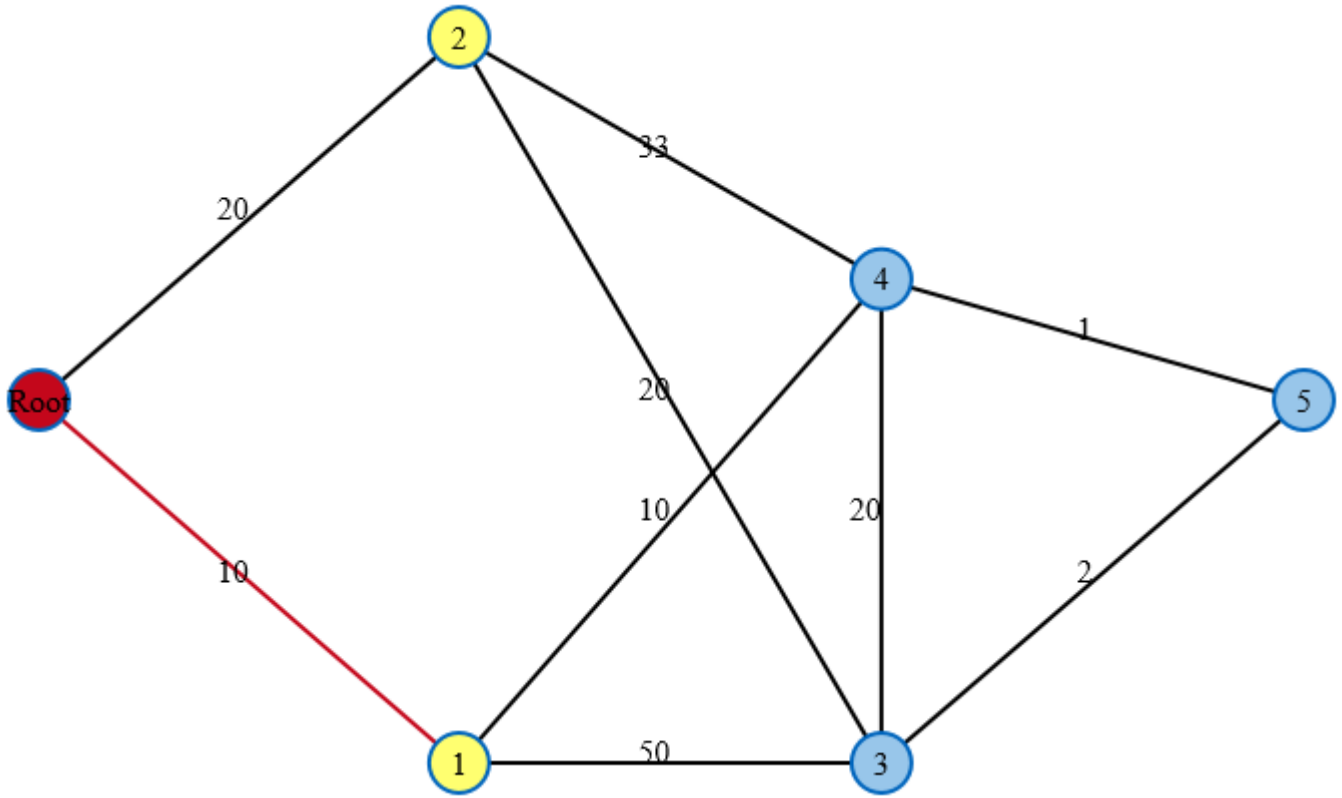
The Minimum Spanning Tree Algorithm



- Not yet visited neighboring nodes
  - If the node is not already added to the MST, then it must be processed later. In this case, the node is added to the queue.

Based on the online tool: https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html

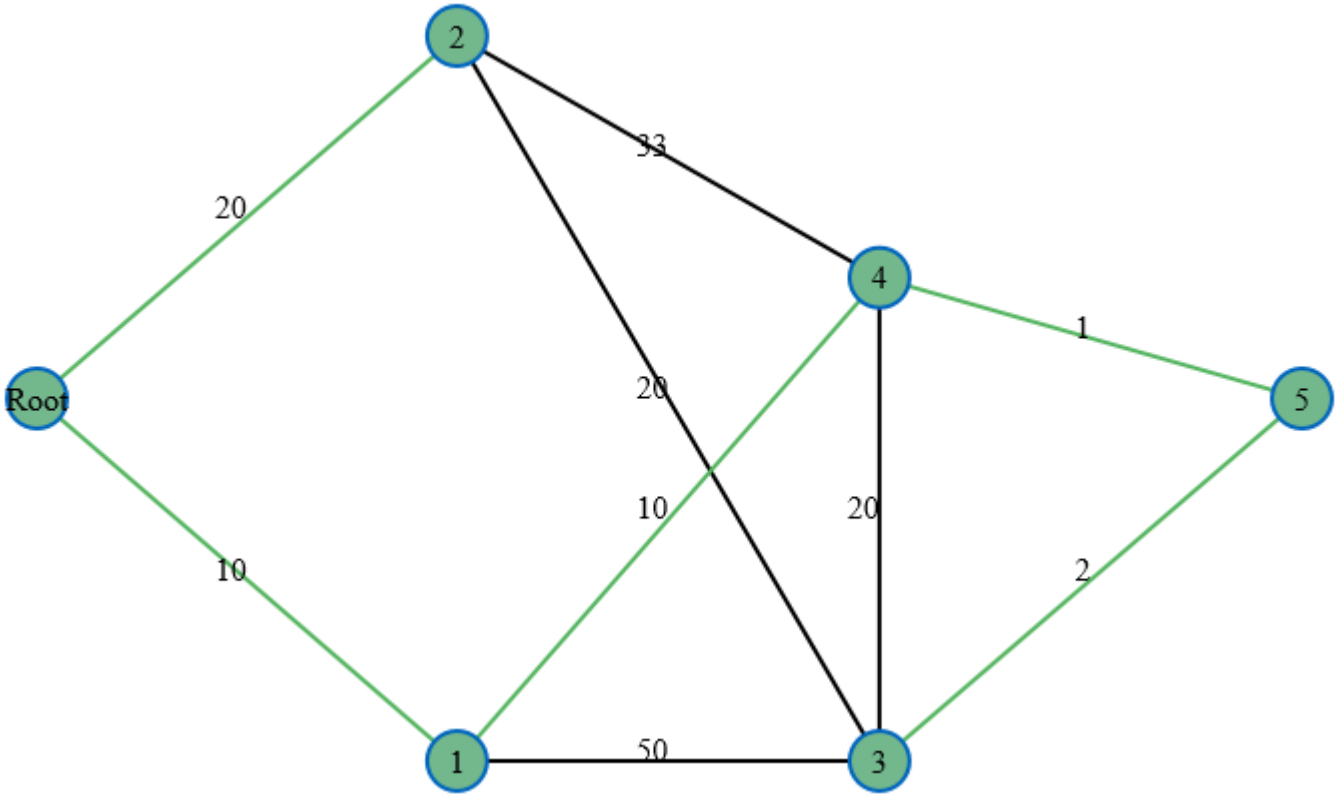
The Minimum Spanning Tree Algorithm



- […]

Based on the online tool: https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html

The Minimum Spanning Tree Algorithm



- End of the algorithm
  - The queue is empty, all nodes are processed, and the MST is computed

Based on the online tool: https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html

Minimum Spanning Tree

- Given a connected, undirected graph, a spanning tree of that graph is an acyclic subgraph that connects all vertices.
- Obviously, a graph may have many spanning trees.
- Weights may be assigned to each edge of the graph, then the total weight of a subgraph is the sum of its edge weights.
- The spanning tree with the weight less than or equal to all other spanning trees is called the minimum spanning tree (MST).
- More generally, any undirected graph has a minimum spanning forest (MSF), which is a union of minimum trees for its connected components.

Prim’s Algorithm

- Prim's algorithm is a greedy algorithm (a problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum) that efficiently finds the minimum spanning tree for the connected weighted undirected graphs.

Prim's Algorithm

- The algorithm initializes a tree with a single vertex chosen arbitrarily, and in each round the tree is grown by one edge. This edge has the minimum weight among all the edges that connect the tree to the vertices not yet on the tree. The process is continued until all the vertices are added to the tree.

Prim's Algorithm

- The most efficient way to implement the algorithm is using a queue.
- This queue keeps track of all the nodes that are visited but not yet added to the tree. The order of the nodes in the queue is determined using the distance of each node to the tree.
- Hence, in each round the vertex that is connected to the tree by the cheapest edge is removed from the queue.

Prim's Algorithm

- The algorithm may be described step-by-step. First is the initialization part:
  - an arbitrary node is selected as the root of the tree.
  - a distance array keeps track of minimum weighted edge connecting each vertex to the tree. The Root node has distance zero, and for all other vertices there is no edge to the tree, so their distance is set to infinity.
  - the root node is added to the queue.
- Next, the algorithm iterates over the queue. In each round, the node in front of the queue is extracted. Initially this may be only the root node. Then all the neighbouring vertices of the removed node are considered with their respective edges. Each neighbour is checked whether it is in the queue and its connecting edge weighs less than the current distance value for that neighbour. If this is the case, the cost of this neighbour can be reduced to the new value. Otherwise, if the node has not yet been visited then it most be inserted into the queue, so the edges going out of it may be considered later.
- The above iteration continues until no more nodes are included in the queue. Then the algorithm is finished and as the result it returns all the edges used for building the minimum spanning tree.

Prim's Algorithm Pseudocode

$ReachSet = \{0\};$ // You can use any node...
$UnReachSet = \{1, 2, \ldots, N-1\};$
$SpanningTree = \{\};$

while ($UnReachSet \neq \emptyset$)
{
Find edge $e = (x, y)$ such that:
1. $x \in ReachSet$
2. $y \in UnReachSet$
3. e has smallest cost

SpanningTree = SpanningTree ∪ $\{e\}$;

$ReachSet = ReachSet \cup \{y\};$
$UnReachSet = UnReachSet - \{y\};$
}

Thank you