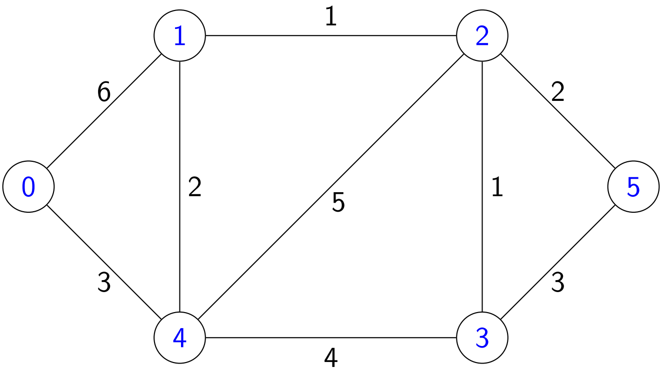
Dijkstra and Prim's examples (AP2)

Emma Rollón

UPC

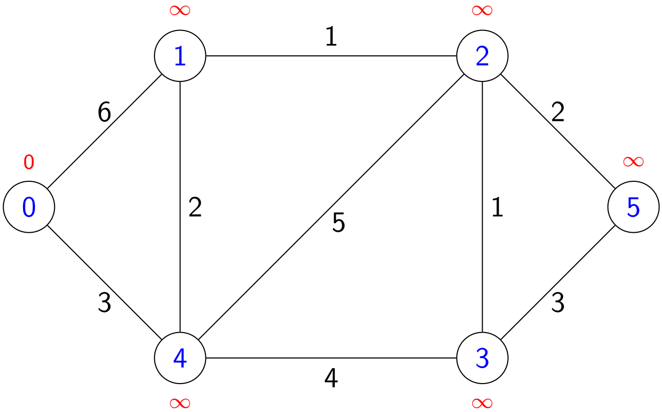
Dijkstra: Shortest Paths Algorithm



Input: a weighted undirected graph (with positive weights), a source node
Goal: to compute the shortest paths from the source node to any other node in the graph

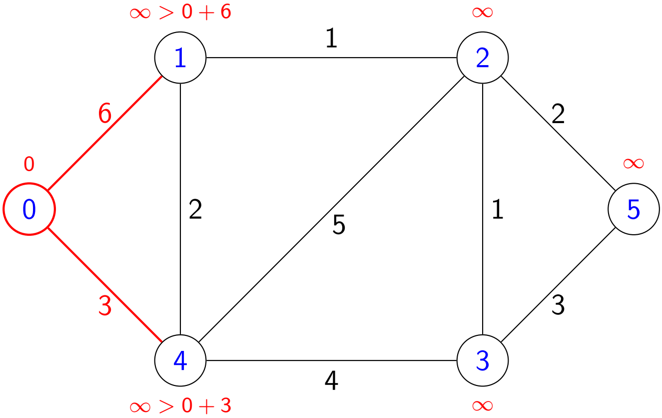
Idea: each node maintains its best known distance to the source node and, at each step, it selects the node with the lowest distance.

Dijkstra: Shortest Paths Algorithm



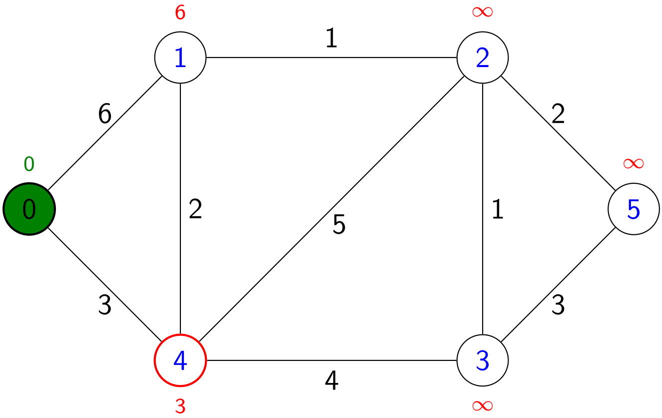
Step 1: associate each node with its best known distance to the source node (node 0 is the source node - the distance to itself is 0)

Dijkstra: Shortest Paths Algorithm



Step 2: select the node with the lowest distance; update the best known distance of its adjacent nodes (if necessary); and mark the selected node as visited.

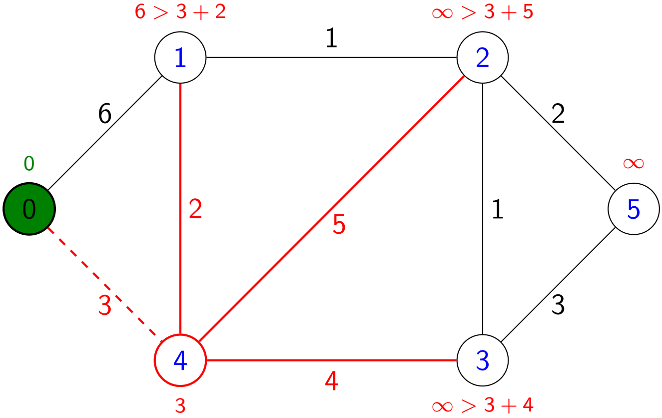
Dijkstra: Shortest Paths Algorithm



Step 3: repeat step 2 until all nodes have been visited.

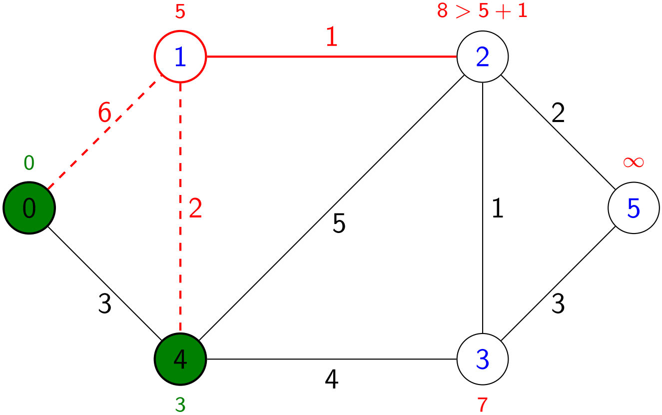
Can we assert that 3 is the definitive lowest distance from source node (node 0) to node 4?

Dijkstra: Shortest Paths Algorithm



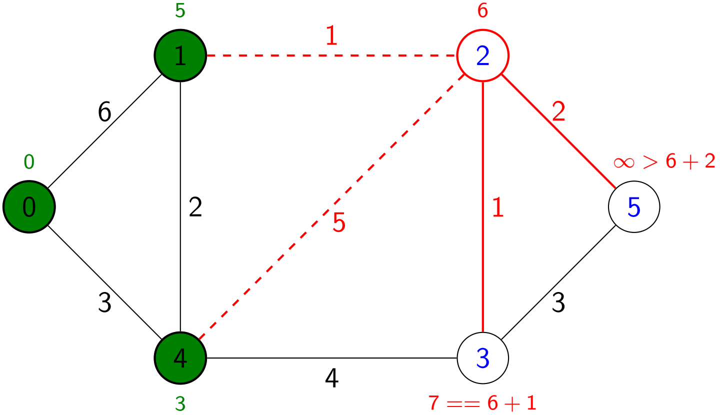
Step 3: repeat step 2 until all nodes have been visited.

think If an adjacent node has already been selected, could the weight of new discovered paths be smaller than the one it had?



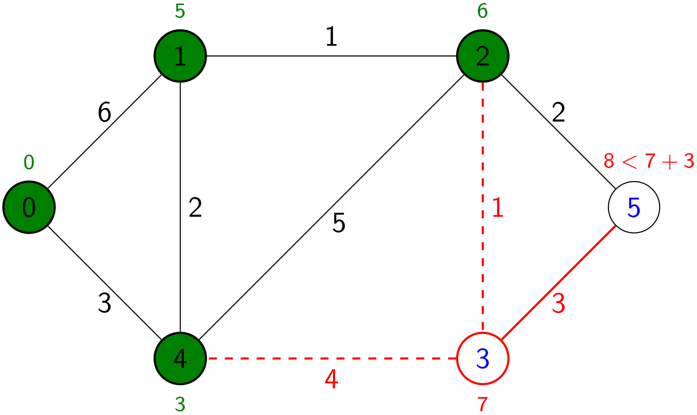
Step 3: repeat step 2 until all nodes have been visited.

Dijkstra: Shortest Paths Algorithm



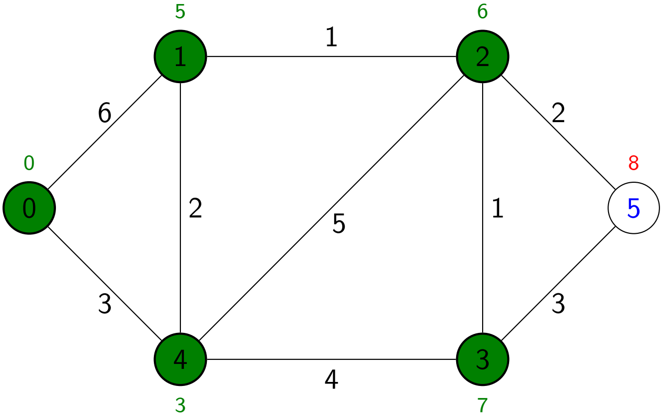
Step 3: repeat step 2 until all nodes have been visited.

Dijkstra: Shortest Paths Algorithm



Step 3: repeat step 2 until all nodes have been visited.

Dijkstra: Shortest Paths Algorithm



We don't need to treat the last node (we already know its best distance to the source node).

When all edge-weights are equal, which would be the order in which nodes are visited?

Dijkstra: Shortest Paths Algorithm

How to maintain the best distance to each node?

0	1	...	n - 1
d_0	d_1	$\ldots$	d_{n-1}

Vector

Access: $O(1)$

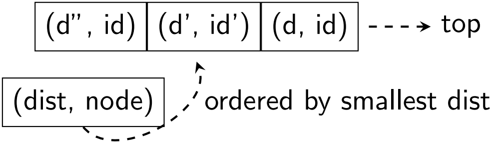
How to know if a node is already visited?

0	1	...	n - 1
t/f	t/f	$\ldots$	t/f

Vector

Access: $O(1)$

How to find the node with the lowest distance to source node?



Priority queue

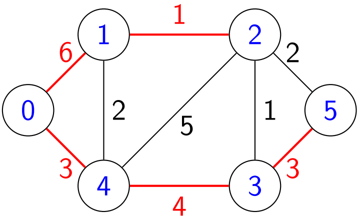
Access top value: $O(1)$

Remove top value: $O(\log e)$

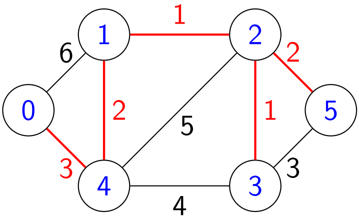
Insert new value: $O(\log e)$

Prim: Minimum Spanning Tree

Spanning tree:



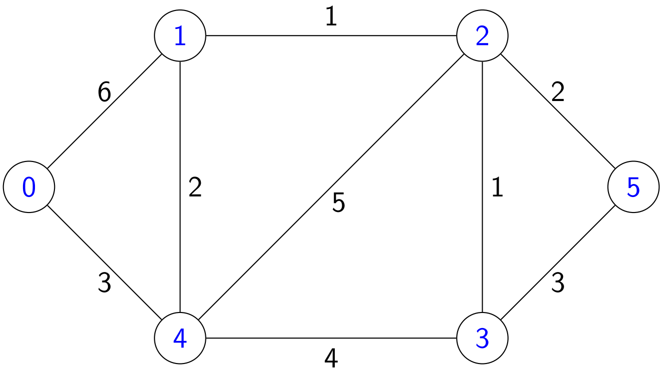
Minimum spanning tree:



A spanning tree is a subset of the edges of a connected undirected graph that connects all the vertices together without any cycles and with minimum possible number of edges.

A minimum spanning tree is a subset of the edges of a connected weighted undirected graph that connects all the vertices together without any cycles and with the minimum possible total edge weight.

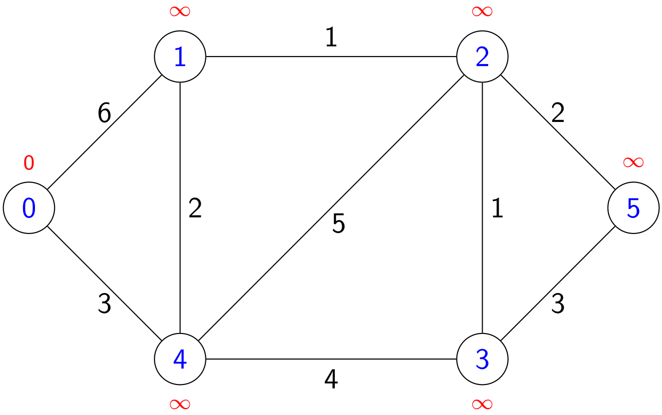
Prim: Minimum Spanning Tree



Input: a weighted undirected graph (with positive weights), a source node
Goal: to compute the minimum spanning tree where source is the root node of the tree

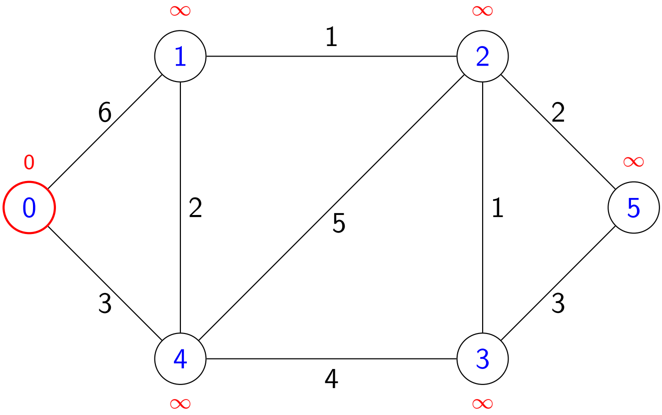
Idea: select the lowest weighted edge among those connecting nodes already in the tree and nodes not yet in the tree.

Prim: Minimum Spanning Tree



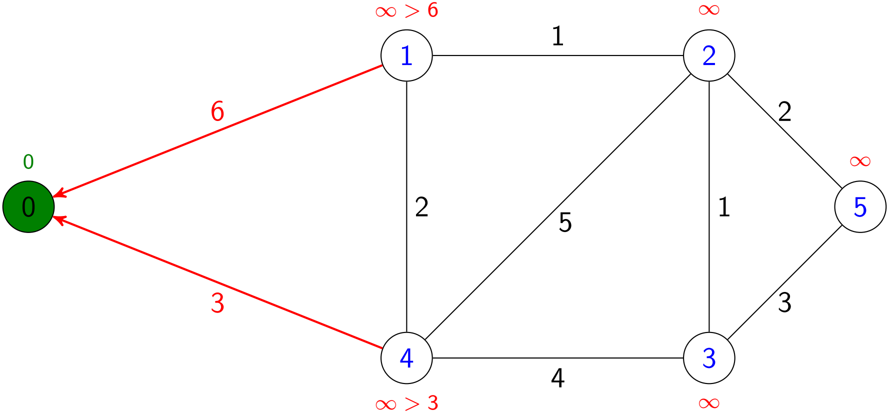
Step 1: associate each node with its lowest weight edge connecting the node with one already in the tree (at this stage, none of the nodes are in the tree but node 0 is the root node - fake edge with 0 cost).

Prim: Minimum Spanning Tree



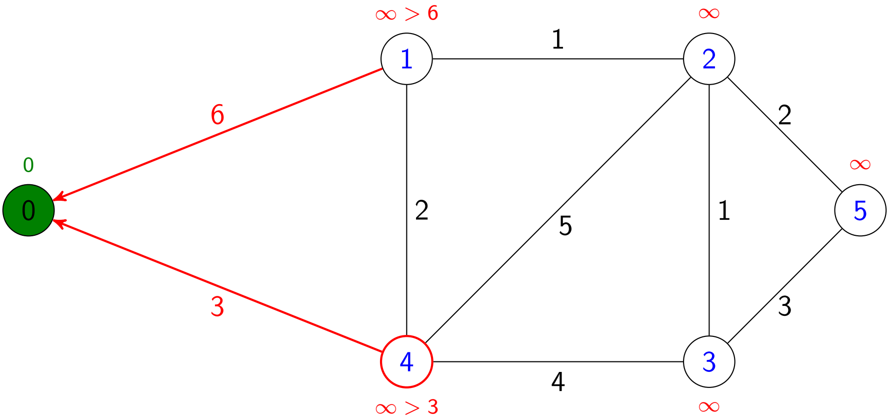
Step 2.a: pick the node with the lowest weighted edge crossing between green nodes (nodes already in the MST) and white nodes (nodes not yet in the MST). That minimum weighted edge is in the MST (initially, the 0-cost edge is a fake edge).

Prim: Minimum Spanning Tree



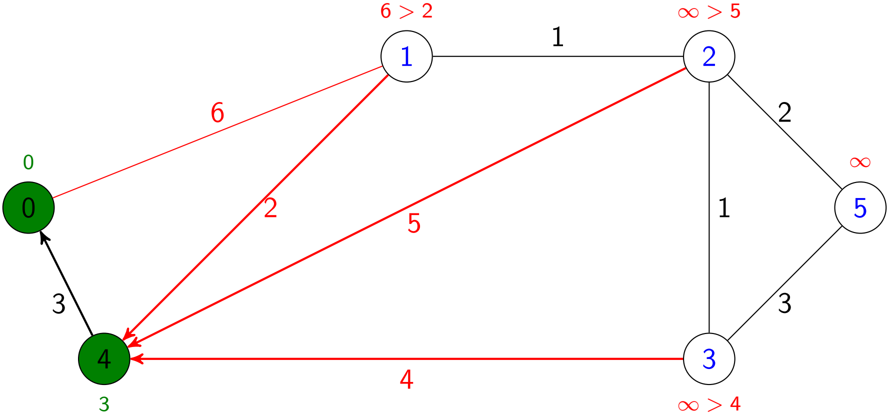
Step 2.b: put the selected node in the set of nodes already in the tree (green nodes); update the cost of its adjacent nodes not yet in the tree with the cost of its minimum weighted edge crossing to green nodes.

Prim: Minimum Spanning Tree



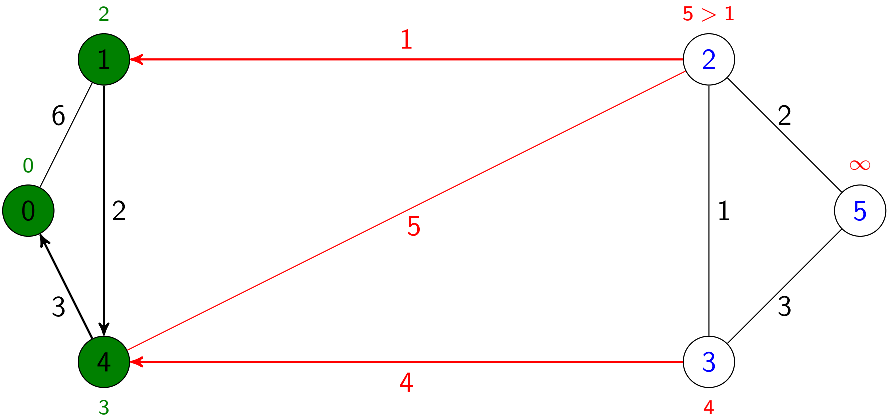
Step 3: repeat step 2.a (edge from node 4 to node 0 with cost 3 is in the MST) and ...

Prim: Minimum Spanning Tree



Step 3: ... and also step 2.b until all nodes have been visited.

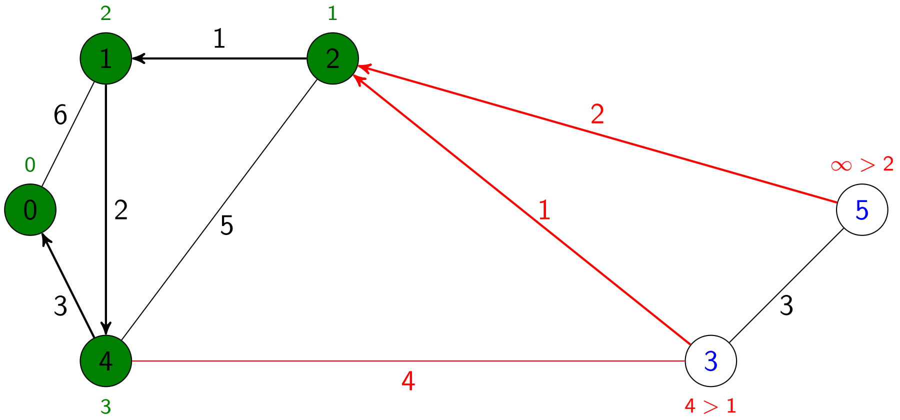
Prim: Minimum Spanning Tree



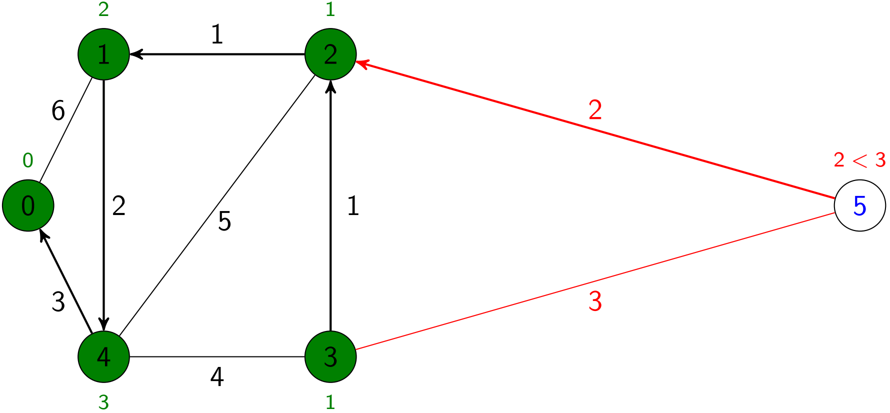
Step 3: repeat step 2.a and 2.b until all nodes have been visited.

If an adjacent node has already been included in the tree, could we find an edge with lower weight?

Prim: Minimum Spanning Tree

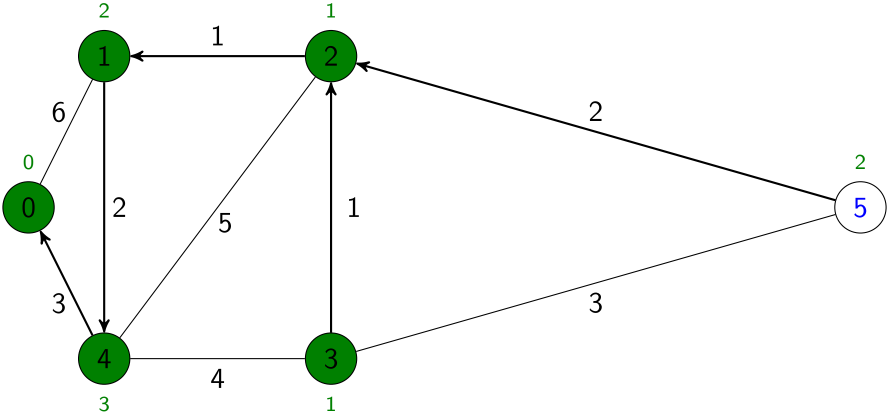


Step 3: repeat step 2.a and 2.b until all nodes have been visited.



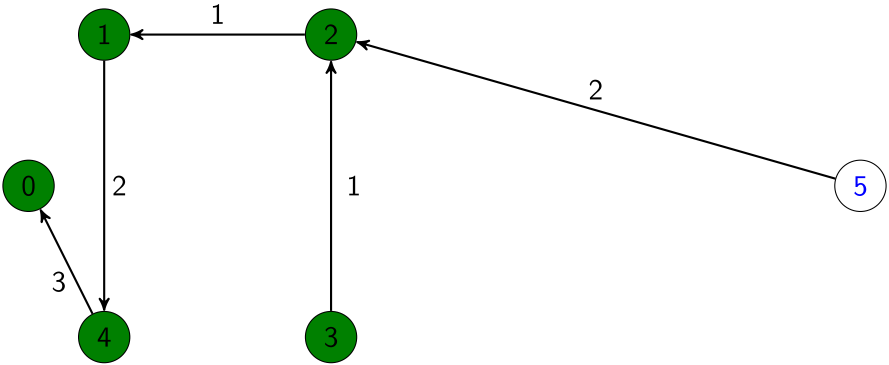
Step 3: repeat step 2.a and 2.b until all nodes have been visited.

Prim: Minimum Spanning Tree



We don't need to treat the last node (we already know its best weighted edge connecting it to the spanning tree).

Prim: Minimum Spanning Tree



The algorithm obtains this MST with cost 9.

Prim: Minimum Spanning Tree

How to know the lowest weighted edge to nodes alredy in the tree?

0	1	...	n - 1
w_0	w_1	$\dots$	w_{n-1}

0	1	...	n - 1
p_0	p_1	$\dots$	p_{n-1}

How to know if a node is already visited?

0	1	...	n - 1
t/f	t/f	$\dots$	t/f

How to find the node with the lowest weighted edge connecting it to one of the spanning tree nodes?