Minimum Spanning Trees using Kruskal's Algorithm with Path Compression and Union by Rank

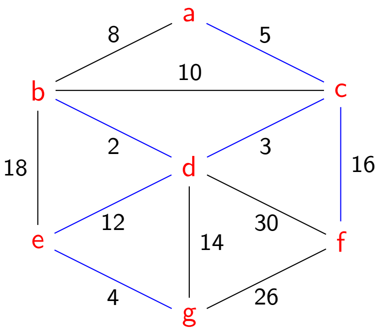
Alan R. Hahn

Clemson University

Dec. 2018

Definition

Given an undirected graph G with real valued weight on each edge, a Minimum Spanning Tree T of a graph G is a spanning tree of G such that the sum of the edge weights in T is minimum with respect to all spanning trees of G.



The blue subtree is the minimum spanning tree of G.

Consider an undirected graph G with real valued weight on each edge.

Kruskal's Algorithm

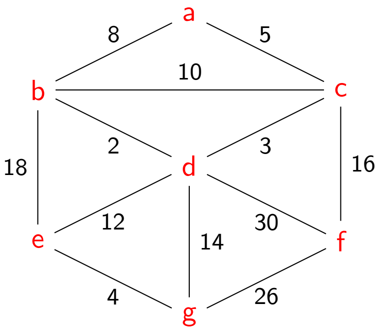
Apply the following step to the edges of G in non - decreasing order by edge weight:

Inclusion Step

If the edge $e = \{u, v\}$ is such that u and v are in the same blue subtree, leave e uncolored. Else, color e blue.

Inclusion Step

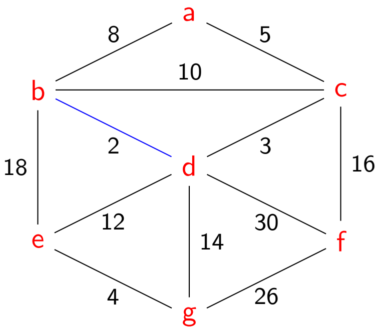
If the edge $e = \{u, v\}$ is such that u and v are in the same blue subtree, leave e uncolored. Else, color e blue.



After 0 steps

Inclusion Step

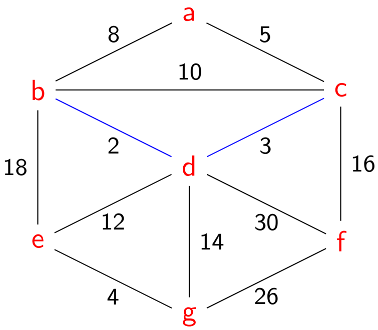
If the edge $e = \{u, v\}$ is such that u and v are in the same blue subtree, leave e uncolored. Else, color e blue.



After 1 step

Inclusion Step

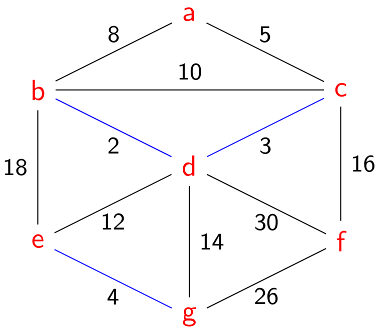
If the edge $e = \{u, v\}$ is such that u and v are in the same blue subtree, leave e uncolored. Else, color e blue.



After 2 steps

Inclusion Step

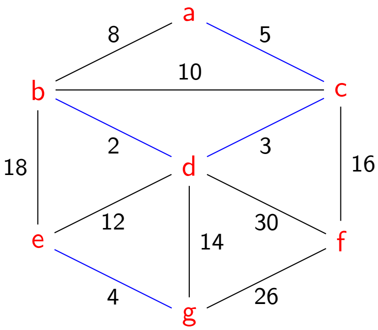
If the edge $e = \{u, v\}$ is such that u and v are in the same blue subtree, leave e uncolored. Else, color e blue.



After 3 steps

Inclusion Step

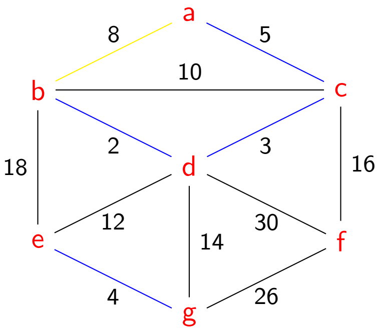
If the edge $e = \{u, v\}$ is such that u and v are in the same blue subtree, leave e uncolored. Else, color e blue.



After 4 steps

Inclusion Step

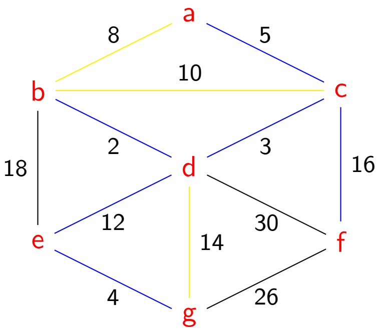
If the edge $e = \{u, v\}$ is such that u and v are in the same blue subtree, leave e uncolored. Else, color e blue.



After 5 steps

Inclusion Step

If the edge $e = \{u, v\}$ is such that u and v are in the same blue subtree, leave e uncolored. Else, color e blue.



After all steps, terminate and return the blue subtree which is the minimum spanning tree of G.

Requirements

- Sort the list of edges E of G
- Check whether two vertices share a blue subtree

The Partition Class

A class to represent a Partition of a given input set S. To identify a given part of a partition, an arbitrary but unique element will be maintained within each part as a representative element.

- *initialize_partition(S): Initialize the partition as a collection of one element parts, one for each s in S*
- *find(s): Return the representative element of the part containing the element s*
- *link(x, y): Union the two parts whose representative elements are x and y, and choose a new representative element to represent the one part*

$$S = \{a, b, c, d, e, f, g\}$$

initialize_partition(S)

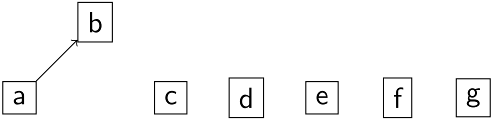
$$[a : a,\ \ b : b,\ \ c : c,\ \ d : d,\ \ e : e,\ \ f : f,\ \ g : g]$$

$\boxed{a}$ $\boxed{b}$ $\boxed{c}$ $\boxed{d}$ $\boxed{e}$ $\boxed{f}$ $\boxed{g}$

$$S = \{a, b, c, d, e, f, g\}$$

$$\text{link}(a, b):$$

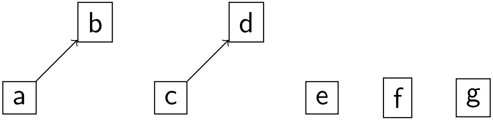
$$[a : b,\ \ b : b,\ \ c : c,\ \ d : d,\ \ e : e,\ \ f : f,\ \ g : g]$$



$$S = \{a, b, c, d, e, f, g\}$$

$$\text{link}(c, d):$$

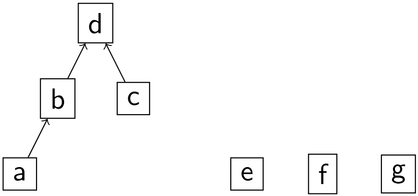
$$[a : b,\ b : b,\ c : d,\ d : d,\ e : e,\ f : f,\ g : g]$$



$$S = \{a, b, c, d, e, f, g\}$$

link(b, d):

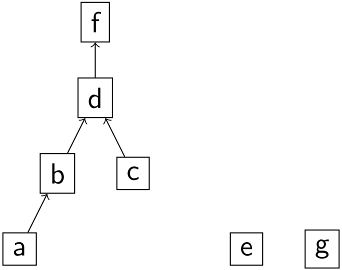
$$[a : b,\; b : d,\; c : d,\; d : d,\; e : e,\; f : f,\; g : g]$$



$$S = \{a, b, c, d, e, f, g\}$$

$$\text{link}(d, f)$$

$$[a : b,\ b : d,\ c : d,\ d : f,\ e : e,\ f : f,\ g : g]$$



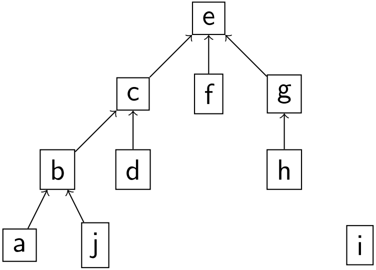
find(a) returns f

Two Heuristics to improve runtime

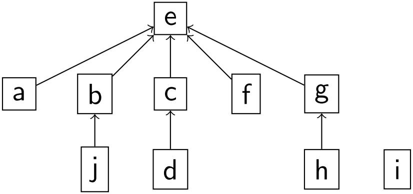
- Path Compression: For an input vertex a and representative element s, find(a) returns s and updates to s the parent of every node on the path from a to s

- Union by Rank: Changes the map and the link method so that along with the parent node, the map of a given vertex s returns a non-negative integer function called the rank of s. For each element its rank is initialized to 0. Given two representatives x and y, if without loss of generality $\text{rank}(x) > \text{rank}(y)$, link chooses as the representative of the union of the parts which contain x and y to be x. If $\text{rank}(x) = \text{rank}(y)$, one of $\text{rank}(x)$, $\text{rank}(y)$ is increased by one and the element chosen to have higher rank is then the representative of the union of the parts which contain x and y

Path Compression



find(a) returns e and updates the parent node of every vertex along the path from a to e:

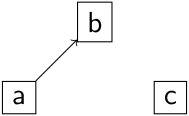


Union by Rank

$[a : [a, 0],\ \ b : [b, 0],\ \ c : [c, 0]]$

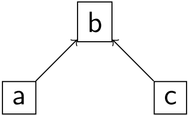
$\text{link}(a, b)$

$[a : [b, 0],\ \ b : [b, 1],\ \ c : [c, 0]]$



$\text{link}(b, c)$

$[a : [b, 0],\ \ b : [b, 1],\ \ c : [b, 0]]$



References

- Kumar, S., Spezzano, F., Subrahmanian, V., & Faloutsos, C. (2016). Edge weight prediction in weighted signed networks. In *Data mining (icdm), 2016 ieee 16th international conference on* (pp. 221-230).
- Tarjan, R. E. (1983). *Data structures and network algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.