Kruskal's and Prim's algorithm

1 Kruskal's algorithm to find a minimum weight spanning tree

The method consists of

- Sorting the edges by increasing weight;
- Constructing a spanning tree by adding one of the smallest available edges in each step.

An edge is available if it has not been selected before and it does not close a cycle with any of the previously selected edges.

Theorem 1 *Kruskal's algorithm yields a minimum weight spanning tree.*

Proof: Assume Kruskal's algorithm has selected the edges $e_1, \ldots, e_{v-1}$, in this order. These edges form a spanning tree T. Assume T' is a minimum weight spanning tree that shares the largest possible number of common edges with T. Sort the edges of T' by increasing order of weights, assume that we obtain the list $f_1, f_2, \ldots, f_{v-1}$. (Among edges of same weight we may assume that we always list the common elements of T and T' first, in the same order as in T.) If T and T' are not equal then there is an i such that $e_1 = f_1$, $e_2 = f_2$, $\ldots$, $e_{i-1} = f_{i-1}$, but $e_i \neq f_i$. Since e_i is not an edge of T', it closes a cycle in it. Assume this cycle is $(e_i, f_{i_1}, f_{i_2}, \ldots, f_{i_k})$. Here at least one of $f_{i_1}, f_{i_2}, \ldots, f_{i_k}$, say f_{i_j} has the property that is does not belong to $\{e_1, \ldots, e_{i-1}\}$ nor does it close a cycle with the set $\{e_1, \ldots, e_{i-1}\}$: in the contrary event we can replace each f_{i_j} with the corresponding walk between its endpoints with edges from $\{e_1, \ldots, e_{i-1}\}$ and concatenating this walks would yield a walk between the endpoints of e_1, using only edges from $\{e_1, \ldots, e_{i-1}\}$. Since we were allowed to choose e_i instead of f_{i_j} in step i, we have $w(e_i) \leq w(f_{i_j})$. Thus the tree $T'' := T' - f_{i_j} + e_i$ can not have larger weight than T'. Since T' has minimum weight, the same is true for T'' (and so $w(e_i) = w(f_{i_j})$). The tree T'' has one more edge in common with T, in contradiction with the choice of T'. This contradiction is avoided only if $T = T'$ and so T must be a minimum weight spanning tree. $\diamond$

2 Prim's algorithm

When a graph has a lot of edges, the first phase of Kruskal's algorithm might take long. Prim's algorithm consists of modifying Kruskal's algorithm by considering only those edges in each step that form a connected subgraph with the previously selected edges. We start with an arbitrarily selected vertex x_0. After $i-1$ steps we have selected a subtree on the vertex set $\{x_0, \ldots, x_{i-1}\}$. In step i we consider all edges of the form (x_j, y) where $1 \leq j \leq i-1$ and $y \notin \{x_0, \ldots, x_{i-1}\}$, and pick an edge (x_j, x_i) of minimum weight among them. (This determines the selection of x_i.)

Theorem 2 *Prim's algorithm yields a minimum weight spanning tree.*

Proof: The proof may be obtained by modifying the proof for Kruskal's algorithm as follows. Assume again that the output of our algorithm is T and that we added the edges $e_1, \ldots, e_{v-1}$, in this order. Let T' be a minimum weight spanning tree that has the largest possible number of common edges with T. If T is not minimum weight then $T \neq T'$ and there is a first edge e_i on the list that does not belong to T'. Removing e_i from T yields two components: the vertices $\{x_0, \ldots, x_{i-1}\}$ on the one side and $V \setminus \{x_0, \ldots, x_{i-1}\}$ on the other side. Since e_i does not belong to T', it closes a cycle in it. This cycle contains a second edge f connecting a vertex from $\{x_0, \ldots, x_{i-1}\}$ with a vertex from $V \setminus \{x_0, \ldots, x_{i-1}\}$. Since we chose e_i in step i, we must have $w(e_i) \leq w(f)$. Consider now the tree $T'' := T' - f + e_i$. It is minimum weight, and has one more edge in common with T. Again we reach a contradiction. ◇