Greedy Algorithms for Minimum Spanning Tree

Harvey J. Greenberg
University of Colorado at Denver
http://www.cudenver.edu/~hgreenbe/
(url changed December 1, 1998)

March 28, 1998

The glossary defines a spanning tree for a connected graph with non-negative weights on its edges, and one problem: find a *max weight spanning tree*. Remarkably, the greedy algorithm results in a solution. Here we present similar greedy algorithms due to Prim [3] and Kruskal [2], respectively, for the problem: find a *min weight spanning tree*. Graham and Hell [1] gives a history of the problem, which originated with the work of Czekanowski in 1909. The material here is based on Rosen [4].

The Algorithms

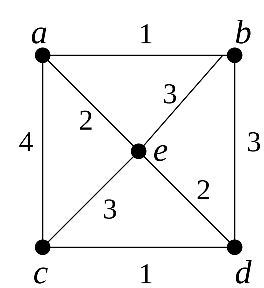
We are given a connected graph, $G = [V, E]$, with n vertices and m edges with non-negative weights, $w(e_i)$. We first sort the edges such that $w(e_1) \leq \ldots \leq w(e_m)$. (This takes $O(m \log m)$ time.) The output is a spanning tree, T, whose total weight is a minimum.

For each algorithm, T is initialized with e_1 (an edge with minimum weight) and its two endpoints. The number of vertices in T is denoted $v(T)$ (which is initialized at 2).

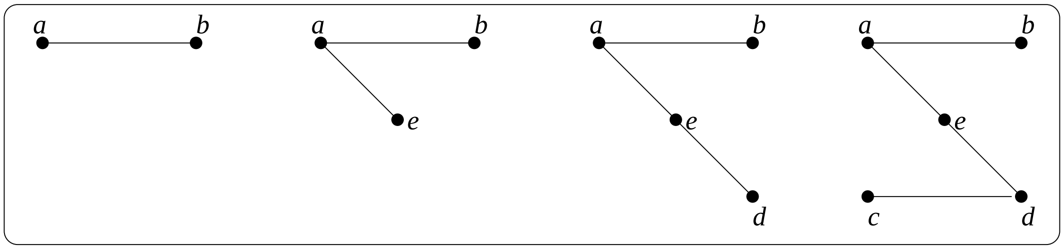
Prim's Algorithm. DO WHILE $v(T) < n$: interrogate edges (in order) until one is found that is incident with a vertex in T and does not form a simple circuit in T. Then, add this edge and its endpoint to T (thereby increasing $v(T)$ by 1).

Kruskal's Algorithm. DO WHILE $v(T) < n$: interrogate edges (in order) until one is found that does not form a simple circuit in T. Then, add this edge and its endpoint(s) to T (thereby increasing $v(T)$ by 1 or 2).

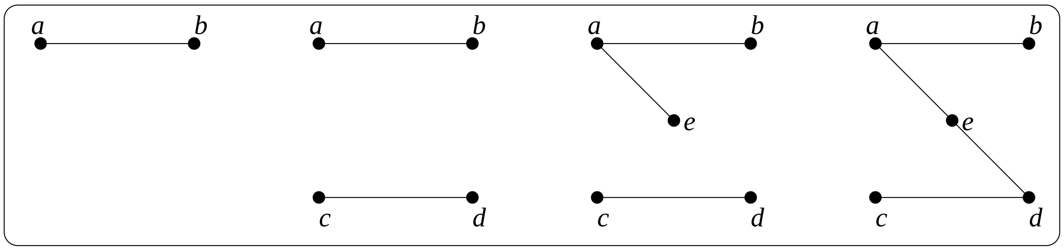
The algorithms differ in that Prim's requires that the next edge added be incident with a vertex in the (partial) tree, T, whereas Kruskal's just adds the next edge that does not form a circuit. To illustrate, we present the progression of Prim's and Kruskal's algorithms for the following graph:



Prim



Kruskal

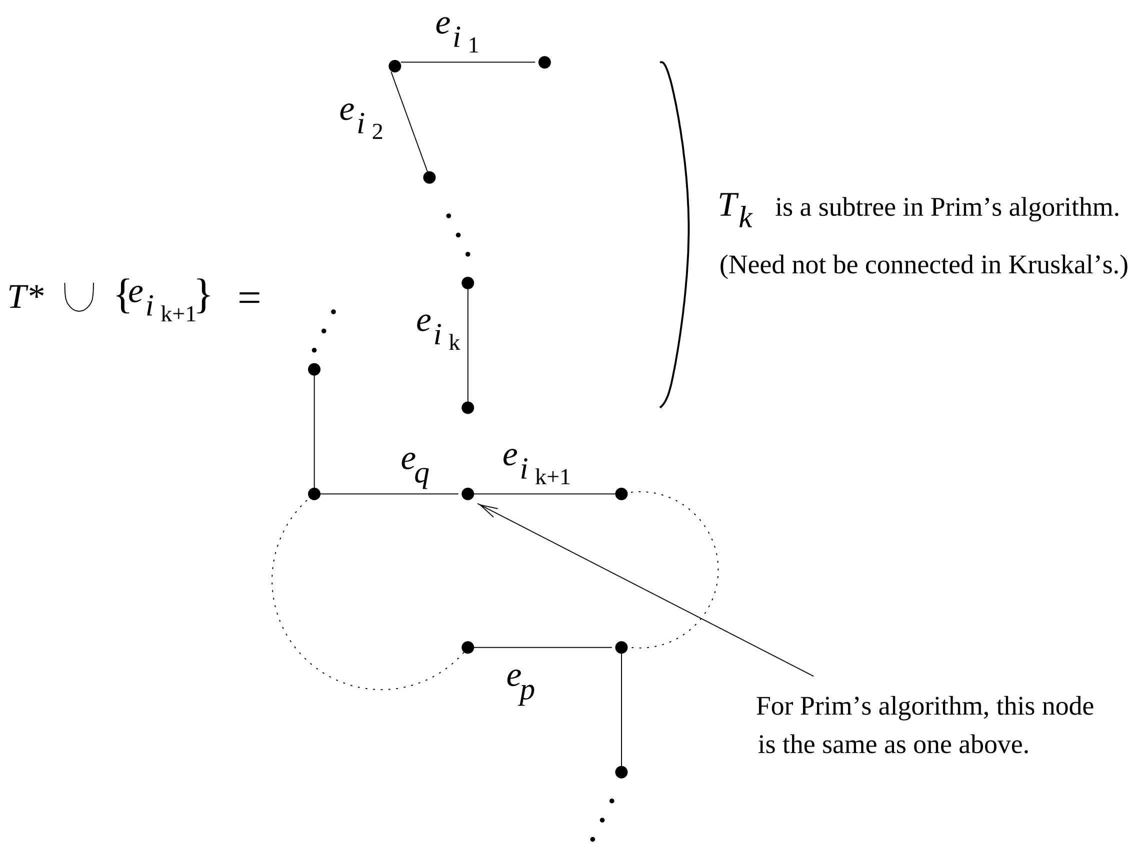


They arrive at the same minimum spanning tree whose total weight is 6.

Proof of Optimality

Every graph with n vertices, $n-1$ edges and no circuits must be a tree; in particular, the graph must be connected, and the algorithms result in a spanning tree (whose minimality is shown below). If the original graph is not connected, Prim's algorithm will find a minimum spanning tree in the component containing e_1, then it will fail to add any more edges. Kruskal's algorithm will find a minimum spanning tree for each component. In the following proof of optimality, we assume G is connected, and the algorithm added the edges in the order $e_{i_1}, e_{i_2}, \ldots, e_{i_{n-1}}$ (note: $i_1 = 1$).

For each subset, $\{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$, let T_k denote the associated subgraph consisting of those edges plus their endpoints. (In the case of Prim's algorithm, T_k is connected, so it is a tree with $v(T_k) = k+1$.) Choose k to be the maximum integer with the property that a minimum spanning tree exists that contains T_k. ($k = 0$ means no minimum spanning tree contains e_1, but the following proof shows this cannot happen.) Let T^* be a minimum spanning tree, with total weight $w(T^*)$, that contains T_k (for the maximum k), but $k < n-1$ (i.e., $T^* \neq T_{n-1}$). Since $e_{i_{k+1}} \notin T^*$, $T^* \cup \{e_{i_{k+1}}\}$ has a simple circuit containing $e_{i_{k+1}}$. We let e_p be any edge in this circuit that was a candidate for Kruskal's algorithm, and we let e_q be a candidate for Prim's algorithm. Here is a picture to clarify the notation:



We then let e' denote e_p or e_q, according to which algorithm is executed, and we consider an exchange of $e_{i_{k+1}}$ for e'. The circuit cannot contain only edges in T_k because that would make $e_{i_{k+1}}$ ineligible, so e' is not one of the previously selected edges in T_k, which means the exchange results in a new spanning tree, T', with total weight, $w(T') = w(T^*) + w(e_{i_{k+1}}) - w(e')$. Since e' was a candidate, the rules for adding an edge in either algorithm imply $w(e_{i_{k+1}}) \le w(e')$, so $w(T') \le w(T^*)$. Since T^* is a minimum spanning tree, equality must hold, so we have $w(T') = w(T^*)$, which means T' is also a minimum spanning tree. However, $T' \supseteq T_{k+1}$, which contradicts the maximality of k.

The implication of this is that either greedy algorithm (Prim or Kruskal) for the minimum spanning tree problem produces an optimal solution.

References

[1] R.L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1):43–57, 1985.

[2] J.B. Kruskal. On the shortest spanning tree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[3] R.C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technology Journal*, 36:1389–1401, 1957.

[4] K.H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill, Inc., New York, NY, third edition, 1995.