MA2C03 Mathematics
School of Mathematics, Trinity College
Hilary Term 2016
Lecture 38 (February 3, 2016)

David R. Wilkins

We now analyze Kruskal's Algorithm in order to show that if that algorithm is applied to a given connected graph, then the spanning tree generated by that algorithm minimizes cost.

First we recall the specification of the algorithm.

We are given a connected graph. Let V denote the set of vertices of the given graph, and let E denote the set of edges of the given graph. There is a cost function $c\colon E \to \mathbb{R}$ defined on the set of edges of the given graph. The cost of any spanning tree is defined to be the sum of the costs of the edges of that spanning tree. The objective is to find a spanning tree of the connected graph whose cost is less than or equal to that of every other spanning tree of the connected graph.

To implement Kruskal's Algorithm we order the edges of the given connected graph in a finite sequence, or *queue*, so that, given any pair of edges e and e' for which $c(e) < c(e')$, the edge e precedes the edge e' in the queue.

We start with a acyclic subgraph of the given connected graph consisting of all the vertices of the given graph. We build up an acyclic subgraph through the addition of edges. This initial subgraph has no edges. We then take edges in order from the the front of the queue. Having taken an edge from the front of the queue, we determine whether or not addition of that edge to the current acyclic subgraph would create a circuit in the resultant graph. If a circuit would be created, then we discard the edge. Otherwise we add the edge to the acyclic subgraph so as to create a larger acyclic subgraph. We continue until the queue has been exhausted.

Proposition 34.1

Let a connected graph be given, together with a cost function defined on its set of edges, and let Kruskal's Algorithm be applied to determine a spanning tree K. Let T be a spanning tree in the connected graph that does not coincide with the spanning tree K resulting from application of Kruskal's Algorithm. Then there exists a spanning tree T', where the cost of T' does not exceed that of T, such that T' includes more edges of K than does T.

Proof

Let us refer to the spanning tree of the given connected graph constructed using Kruskal's Algorithm as the *Kruskal spanning tree*.

Let the given connected graph have $m + 1$ vertices. Then any spanning tree for this graph has m edges and $m + 1$ vertices (see Theorem 32.3). Moreover any connected subgraph of the given connected graph with m edges and $m + 1$ vertices is a spanning tree for that graph (see Corollary 33.1).

Let the edges of the Kruskal spanning tree be denoted by $e_1, e_2, \ldots, e_m$, where the order of these edges is the order established by the queue constructed in applying the algorithm.

Now let T be an spanning tree for the given connected graph that is distinct from the Kruskal spanning tree. Both spanning trees have the same number of edges. It follows that the Kruskal spanning tree must have at least one edge that does not belong to the spanning tree T. Therefore there exists some integer k satisfying $0 \leq k \leq m$ such that e_i is an edge of T whenever $i < k$ but e_k is not an edge of T.

Because T is connected, there is a path in T between any two vertices of T. It follows that if the edge e_k is added to T, then the resultant graph contains a circuit. This circuit must include the edge e_k. But it cannot be contained within the Kruskal spanning tree, because the Kruskal spanning tree is acyclic. Therefore the circuit must include at least one edge e' that does not belong to the Kruskal spanning tree.

Now there cannot exist any circuit in the subgraph of the given connected graph consisting of the vertices of that graph, the edge e' and the edges e_i for $i < k$, because that subgraph is contained in the spanning tree T. Had it been the case that $c(e') < c(e_k)$, then e' would have preceded e_k in the queue in the application of Kruskal's algorithm, and it would accordingly have been added to the Kruskal spanning tree. Because e' was not added to the Kruskal spanning true, it must be the case that $c(e') \geq c(e_k)$.

Suppose we modify the spanning tree T by adding the edge e_k and then removing the edge e' to obtain a subgraph T' of the given connected graph. Once the edge e_k is added, the resultant graph contains a circuit which includes the edge e'. The removal of e' then breaks the circuit, leaving behind a connected graph T' with m edges and $m+1$ vertices. This connected graph T' must be a spanning tree of the given connected graph. Its cost cannot exceed that of the graph T, because $c(e') \geq c(e_k)$. Moreover T' includes edges the e_i of the Kruskal spanning tree corresponding to all positive integers i satisfying $i \leq k$. Moreover the spanning tree T' contains more edges of the Kruskal spanning tree than does the spanning tree T, because an edge e' that does not belong to the Kruskal spanning tree has been replaced by an edge e_k that does. The result follows. ∎

Theorem 34.1

Let a connected graph be given, together with a cost function defined on its set of edges, and let Kruskal's Algorithm be applied to determine a spanning tree K. Then the cost of the spanning tree generated by Kruskal's Algorithm is less than or equal to that of every other spanning tree for the given connected graph.

Proof

The number of possible spanning trees of the connected graph is finite. There is therefore a well-defined real number that is the minimum of the costs of all spanning trees for the given connected graph. Moreover there exists a spanning tree T with this minimum cost that has the maximum possible number of edges in common with the spanning tree K generated by Kruskal's Algorithm. But then T must coincide with K.

Indeed if it were the case that T did not coincide with K, then Proposition 34.1 would guarantee the existence of a spanning tree T', where the cost of T' does not exceed that of T, such that T' includes more edges of K than does T. The cost of T' would then also be the minimum of the costs of all spanning trees for the given connected graph. But the existence of such a spanning tree T' would contradict the choice of T as the spanning tree of minimum cost with the maximum possible number of edges in common with K. We conclude therefore that T must coincide with K, and therefore the cost of K is less than or equal to every other spanning tree. The result follows. ∎