CS 6901 (Applied Algorithms) – Lecture 6

Antonina Kolokolova

Sep 29, 2016

1 Greedy algorithms

It is not immediately clear that Kruskal's algorithm yields a spanning tree at all, let alone a minimum cost spanning tree. We will now prove that it does in fact produce an optimal spanning tree.

To show this, we will use the technique that applies to correctness proofs of all algorithms of this type: *greedy* algorithms. An algorithm is greedy if it ranks its input objects in some way (either before or during the execution), then looks at each object in that order and decides whether to include it in the solution or now. At that stage, once it considered an object and made a decision (whether to take it or leave it), it never looks back and changes its decision. This is what makes these algorithms "greedy" – just keep taking the best of what is left. Greedy algorithms tend to be fairly fast (e.g., $O(m \log m)$ time), but in many cases only work for restricted inputs. However, for minimal spanning tree problem, Kruskal's algorithm, in spite of being greedy, still achieves the optimal. We will show this today.

To prove correctness of a greedy algorithm we need to show that every decision it makes does not ruin its chances of getting an optimal solution. There could be many different optimal solutions; we have to show that after each step it can still reach at least one of them.

1.1 Correctness of Kruskal's Algorithm

In case of Kruskal's algorithm, we reason that after each execution of the loop, the set T of edges can be expanded to an optimal spanning tree using edges that have not yet been considered; this will be our loop invariant. Hence after termination, since all edges have been considered, T must itself be a minimum cost spanning tree.

We can formalize this reasoning as follows:

Definition 1 *A set T of edges of G is* promising after stage i *if T can be expanded to a optimal spanning tree for G using edges from $\{e_{i+1}, e_{i+2}, \ldots, e_m\}$. That is, T is promising after stage i if there is an optimal spanning tree T_{opt} such that $T \subseteq T_{opt} \subseteq T \cup \{e_{i+1}, e_{i+2}, \ldots, e_m\}$.*

Lemma 1 *For $0 \leq i \leq m$, let T_i be the value of T after i stages, that is, after examining edges $e_1, \ldots, e_i$. Then the following predicate $P(i)$ holds for every i, $0 \leq i \leq m$:*

$P(i)$: T_i is promising after stage i.

Proof:
We will prove this by induction. $P(0)$ holds because T is initially empty. Since the graph is connected, there exists *some* optimal spanning tree T_{opt}, and
$T_0 \subseteq T_{opt} \subseteq T_0 \cup \{e_1, e_2, \ldots, e_m\}$.

For the induction step, let $0 \leq i < m$, and assume $P(i)$. We want to show $P(i+1)$. Since T_i is promising for stage i, let T_{opt} be an optimal spanning tree such that
$T_i \subseteq T_{opt} \subseteq T_i \cup \{e_{i+1}, e_{i+2}, \ldots, e_m\}$. If e_{i+1} is rejected, then $T_i \cup \{e_{i+1}\}$ contains a cycle and $T_{i+1} = T_i$. Since $T_i \subseteq T_{opt}$ and T_{opt} is acyclic, $e_{i+1} \notin T_{opt}$. So
$T_{i+1} \subseteq T_{opt} \subseteq T_{i+1} \cup \{e_{i+2}, \ldots, e_m\}$.

Now consider the case that $T_i \cup \{e_{i+1}\}$ does *not* contain a cycle, so we have $T_{i+1} = T_i \cup \{e_{i+1}\}$. If $e_{i+1} \in T_{opt}$, then we have $T_{i+1} \subseteq T_{opt} \subseteq T_{i+1} \cup \{e_{i+2}, \ldots, e_m\}$.
So assume that $e_{i+1} \notin T_{opt}$. Then according to the Exchange Lemma below (letting T_1 be T_{opt} and T_2 be T_{i+1}), there is an edge $e_j \in T_{opt} - T_{i+1}$ such that $T'_{opt} = T_{opt} \cup \{e_{i+1}\} - \{e_j\}$ is a spanning tree. Clearly $T_{i+1} \subseteq T'_{opt} \subseteq T_{i+1} \cup \{e_{i+2}, \ldots, e_m\}$. It remains to show that T'_{opt} is optimal. Since $T_{opt} \subseteq T_i \cup \{e_{i+1}, e_{i+2}, \ldots, e_m\}$ and $e_j \in T_{opt} - T_{i+1}$, we have $j > i+1$. So (because we sorted the edges) $c(e_{i+1}) \leq c(e_j)$, so $c(T'_{opt}) = c(T_{opt}) + c(e_{i+1}) - c(e_j) \leq c(T_{opt})$. Since T_{opt} is optimal, we must in fact have $c(T'_{opt}) = c(T_{opt})$, and T'_{opt} is optimal.

This completes the proof of the above lemma, except for the Exchange Lemma.

Lemma 2 *(Exchange Lemma) Let G be a connected graph, let T_1 be any spanning tree of G, and let T_2 be be a set of edges not containing a cycle. Then for every edge $e \in T_2 - T_1$ there is an edge $e' \in T_1 - T_2$ such that $T_1 \cup \{e\} - \{e'\}$ is a spanning tree of G.*

Proof:
Let T_1 and T_2 be as in the lemma, and let $e \in T_2 - T_1$. Say that $e = [u, v]$. Since there is a path from u to v in T_1, $T_1 \cup \{e\}$ contains a cycle C, and it is easy to see that C is the only cycle in $T_1 \cup \{e\}$. Since T_2 is acyclic, there must be an edge e' on C that is not in T_2, and hence $e' \in T_1 - T_2$. Removing a single edge of C from $T_1 \cup \{e\}$ leaves the resulting graph acyclic but still connected, and hence a spanning tree. So $T_1 \cup \{e\} - \{e'\}$ is a spanning tree of G. $\square$

We have now proven Lemma 4. We therefore know that T_m is promising after stage m; that is, there is an optimal spanning tree T_{opt} such that $T_m \subseteq T_{opt} \subseteq T_m \cup \emptyset = T_m$, and so $T_m = T_{opt}$. We can therefore state:

Theorem 1 *Given any connected edge weighted graph G, Kruskals algorithm outputs a minimum spanning tree for G.*