Theory (Optional) Problems

Help Center

The following problems are for those of you looking to challenge yourself beyond the required problem sets and programming questions. They are completely optional and will not be graded. While they vary in level, many are pretty challenging, and we strongly encourage you to discuss ideas and approaches with your fellow students on the "Theory Problems" discussion forum.

1. [Posted March 16, 2015.] Consider a connected undirected graph G with not necessarily distinct edge costs. Consider two different minimum-cost spanning trees of G, T and T'. Is there necessarily a sequence of minimum-cost spanning trees $T = T_0, T_1, T_2, \ldots, T_r = T'$ with the property that each consecutive pair T_i, T_{i+1} of MSTs differ by only a single edge swap? Prove the statement or exhibit a counterexample.
2. [Posted March 16, 2015.] Consider the following algorithm. The input is a connected undirected graph with edge costs (distinct, if you prefer). The algorithm proceeds in iterations. If the current graph is a spanning tree, then the algorithm halts. Otherwise, it picks an arbitrary cycle of the current graph and deletes the most expensive edge on the cycle. Is this algorithm guaranteed to compute a minimum-cost spanning tree? Prove it or exhibit a counterexample.
3. [Posted March 16, 2015.] Consider the following algorithm. The input is a connected undirected graph with edge costs (distinct, if you prefer). The algorithm proceeds in phases. Each phase adds some edges to a tree-so-far and reduces the number of vertices in the graph (when there is only 1 vertex left, the MST is just the empty set). In a phase, we identify the cheapest edge e_v incident on each vertex v of the current graph. Let $F = \{e_v\}$ be the collection of all such edges in the current phase. Obtain a new (smaller) graph by contracting all of the edges in F --- so that each connected component of F becomes a single vertex in the new graph --- discarding any self-loops that result. Let T denote the union of all edges that ever get contracted in a phase of this algorithm. Is T guaranteed to be a minimum-cost spanning tree? Prove it or exhibit a counterexample.
4. [Posted March 16, 2015.] Recall the definition of a minimum bottleneck spanning tree from Problem Set #1. Give a linear-time (i.e., $O(m)$) algorithm for computing a minimum bottleneck spanning tree of a connected undirected graph. [Hint: make use of a non-trivial linear-time algorithm discussed in Part 1.]
5. [Posted April 12, 2015.] Consider a connected undirected graph G with edge costs, which need not be distinct. Prove the following statement or provide a counterexample: for every MST T of G, there exists a way to sort G's edges in nondecreasing order of cost so that Kruskal's algorithm outputs the tree T.
6. [Posted April 12, 2015.] Consider a connected undirected graph G with distinct edge costs that are positive integers between 1 and n^3, where n is the number of vertices of G. How fast can you compute the MST of G?
7. [Posted April 12, 2015.] Read about matroids. Prove that the greedy algorithm correctly computes a maximum-weight basis. For the matroid of spanning trees of a graph, this algorithm becomes Kruskal's algorithm. Can you formulate an analog of Prim's MST algorithm for matroids?

Created Mon 3 Dec 2012 1:52 AM IST

Last Modified Mon 30 Mar 2015 5:58 PM IST