CS 256: Algorithm Design and Analysis

Greedy Algorithms Practice (Optional!)

Instructor: Sam McCauley

Greedy Stays Ahead and Exchange Argument

Problem 1. Let X be a set of n intervals on the real line. We say that a set P of points stabs X (or is *the stabbing set*) if every interval in X contains at least one point in P. See Figure 1. In this and the next question, we will design and analyze an efficient algorithm to compute minimum set of points that stabs X. Assume that your input consists of pairs (ℓ_i, r_i), representing the left and right endpoints of ith interval in X, for $1 \leq i \leq n$.

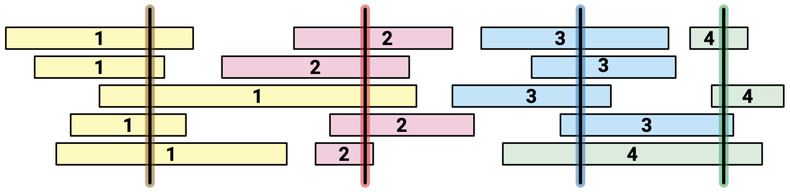

Figure 1: A set of intervals stabbed by four points. From Jeff Erickson's Algorithms book.

(a) *Structure of optimal.* If we were to choose stabbing points anywhere on the intervals, the possibilities would be endless. So first, we prove a structural property about any optimal solution to the problem, as this will guide our algorithm design. Show that for any set of intervals, there exists an optimal solution (a stabbing set of minimum size) where every stabbing point is the right endpoint r_i of some interval. This means that without loss of generality (WLOG), we can restrict ourselves to stabbing sets that satisfy this property.

Consider the following greedy strategy: sort the intervals in increasing order of their right endpoints. Take the first interval from this list, and add its right endpoint to the stabbing set. Remove all intervals that contain this point. Repeat until no intervals remain. Clearly, this produces a valid stabbing set, but is it optimal?

(b) *Greedy is optimal.* Prove that the above greedy strategy is optimal, i.e., it produces a stabbing set of minimum size. (*Hint.* Use (a) and compare to an optimal solution where every stabbing point is a right endpoint of some interval.)

(c) *Running time of greedy.* Analyze the running time of the greedy strategy and show that it can be implemented in $O(n \log n)$ time.

Solution. □

Problem 2 (K-T 4.9)**.** One of the basic motivations behind the Minimum Spanning Tree problem is the goal of designing a spanning network for a set of nodes with minimum *total* cost. Here we explore another type of objective: designing a spanning network for which the *most expensive* edge is as cheap as possible.

Specifically, let $G = (V, E)$ be a connected graph with n vertices, m edges, and positive edge costs that you may assume are all distinct. Let $T = (V, E')$ be a spanning tree of G; we define the *bottleneck edge* of T to be the edge of T with the greatest cost.

A spanning tree T of G is a *minimum-bottleneck spanning tree* if there is no spanning tree T' of G with a cheaper bottleneck edge.

- Is every minimum-bottleneck tree of G a minimum spanning tree of G? Prove or give a counterexample.

  Solution. □

- Is every minimum spanning tree of G a minimum-bottleneck tree of G? Prove or give a counterexample.

  Solution. □