CS 256: Algorithm Design and Analysis

Problem Set 3 (due 03/05/2025 at 9:59pm)

Instructor: Sam McCauley

Problem 1. Recall the "gas station" problem from class: we have n gas stations $d_1, \ldots, d_n$ along with a start location $d_0 = 0$ and a destination d_{n+1}. Our tank only fills up to 200 gallons of gas.

Let's say instead it is the 19th century, and rather than a sequence of gas stations you travel using a sequence of horses. There is a sequence of stables at locations $d_0 = 0, \ldots, d_n$. The horse at stable i can travel m_i miles before getting exhausted; you need to ensure you can make it to a new stable before then.

As before, you know this information ahead of time: you know $d_0 = 0, \ldots, d_{n+1}$ (you start at d_0 and want to reach destination d_{n+1}) and $m_0, \ldots, m_n$. The goal remains the same: we want to stop at the fewest number of stables possible.

Give a greedy algorithm to solve this problem and **prove that it is correct** using either a greedy-stays-ahead or exchange argument. You do not need to analyze the running time.

Hint. What should we be greedy about here? Recall the gas station problem we saw in class: why was waiting until the last possible gas station a good idea? How should we generalize for the horses?

Example. Here's an example instance with $n = 5$:

	0	1	2	3	4	5
d_i:	0	100	150	200	250	340
m_i:	160	160	60	60	100	

The optimal solution stops at d_0, d_1, d_4, d_5. (Notice that being greedy using d_i as we did in class doesn't work: we'd wind up stopping at d_0, d_2, d_3, d_4, d_5.)

Solution. $\square$

Problem 2. 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 2. In this 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 $1 \leq i \leq n$.

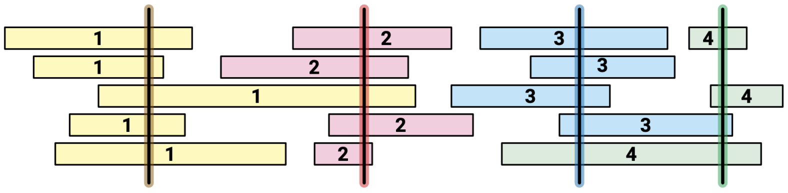

Figure 1: A set of intervals stabbed by four points. From Jeff Erickson's Algorithms book.

(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, we can restrict ourselves to stabbing sets that satisfy this property.

Solution. □

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.)

Solution. □

(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 3. We saw in class that Dijkstra's algorithm may not give a correct answer if the graph has edges with negative weights.

Consider some directed graph G where every edge with a negative weight is an outgoing edge from some vertex s. We will show that running Dijkstra's algorithm starting at s gives the correct answer on this graph. You can assume that $d(s, s) = 0$ (there is no negative-weight cycle).

(a) If you ask ChatGPT about this problem, it makes the following claim: Dijkstra's algorithm will explore (in other words: finish and fill in the entry in `d[]`) all neighbors of s before exploring any other nodes. Give a counterexample to this claim. (There are optional suggestions to help draw a graph, if it's helpful, commented in the Latex source code below.)

Solution.

□

(b) Prove that Dijkstra's algorithm gives the correct answer on G when starting with source vertex s.

Hint. The proof of correctness in this case looks a lot like the proof of correctness we saw in class (with no negative edges). I would recommend first writing out that proof as-is, and then modifying it where necessary to handle the negative edges adjacent to s.

Solution. □