CS91T: Computational Geometry **Fall 2023**

Homework 5

David Yang and Nick Fettig

1. **Let's (temporarily) define the *quality* of a polygon triangulation to be the minimum interior angle of any triangle used. A high-quality triangulation must avoid sliver triangles, so maximizing quality seems related to the goal of minimizing cost (in the sense of total length of diagonals). Using an explicit example, prove that these two goal are not equivalent.**

2. **On n parallel railway tracks there are n trains going at constant speeds $v_1, \dots, v_n$. At time $t = 0$, the trains are at positions $k_1, \dots, k_n$. Using the algorithm for computing intersections of half-planes, design an $O(n \log n)$-time algorithm that lists all trains that are ever in the lead.**

3. **(From Dasgupta, Papadimitriou, and Vazirani) A certain string-processing language offers a primitive operation which splits a string into two pieces. Since this operation involves copying the original string, it takes n units of time for a string of length n, regardless of the location of the cut. Suppose, now, that you want to break a string into many pieces. The order in which the breaks are made can affect the total running time. For example, if you want to cut a 20-character string at positions 3 and 10, then making the first cut at position 3 incurs a total cost of $20 + 17 = 37$, while doing position 10 first has a better cost of $20 + 10 = 30$. Give an $O(m^3)$-time dynamic programming algorithm that, given the locations of m cuts in a string of length n, finds the minimum cost of making those m cuts.**

4. **Prove that $\Omega(n \log n)$ is a lower bound on the worst-case time complexity of finding all optimal solutions to a three-dimensional linear program with n constraints. Your proof should the technique of reduction from sorting, just like our $\Omega(n \log n)$ lower bound on finding the convex hull of n points in the plane. That is, you should explain how, given an unsorted array A of length n, you could sort A by constructing a three-dimensional linear program with $O(n)$ constraints (in $O(n)$ time), finding all solutions to that LP, and then post-processing the solutions (in $O(n)$ time) to get a sorted version of A.**