COS 330: Great Ideas in Theoretical Computer Science

Fall 2025

Problem Set 1

Module: Classic Algorithms

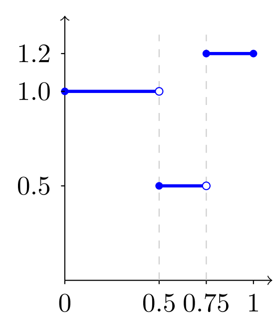
Below is a reminder of key aspects of the PSet:

- The only goal of this PSet is to help you develop your problem-solving skills in preparation for the exams. Your performance on this PSet will not directly contribute to your grade, but will indirectly improve your ability to do well on the exams.
- Because your performance does not directly impact your grade, you may use any resources you like (collaboration, AI, etc.) to help you complete the PSet.
- We suggest taking a serious stab at the PSet alone, to help self-evaluate where you're at. But, we also suggest collaborating with friends, visiting office hours, asking on Ed, and/or using AI tools to help when stuck. Even when able to compute the entire PSet on your own, you may still find any of these methods useful to discuss the PSet afterwards.
- Throughout the PSet, we've included some general tips to help put these into broader context. Exams will not have these, and future PSets may have fewer.

Problem 1 : A Bad ML Model

A piecewise constant function with k pieces is defined by k intervals $[a_1, b_1], [a_2, b_2], \ldots, [a_k, b_k]$ that partition $[0, 1]$ (i.e., $a_1 = 0$, $b_k = 1$, and $b_i = a_{i+1}$ for all $1 \leq i < k$), and each piece has a constant value c_i on interval $[a_i, b_i]$. Formally, $g(x) = c_i$ for all $x \in [a_i, b_i]$. Here is an example:

$$g(x) = \begin{cases} 1.0 & 0 \leq x < 0.5, \\ 0.5 & 0.5 \leq x < 0.75, \\ 1.2 & 0.75 \leq x \leq 1. \end{cases}$$



Consider a (not necessarily piecewise constant) function $f : [0, 1] \to \mathbb{R}$ given as n sample points (x_i, y_i), where $y_i = f(x_i)$ (assume all x_i are distinct). You want to approximate f with a piecewise constant function g using at most k pieces. Define the error of this approximation as the sum of squared errors at sample points, or formally $\sum_i (g(x_i) - y_i)^2$. Describe an $O(n^2 k)$ algorithm that finds the g that minimizes the error.

Problem Solving Tips

Recall our advice of "considering the smallest non-trivial example of the problem we're trying to solve". Here, that would be a case with $k = 1$ pieces. What would the optimal g be in that case? You'll need to know this to solve the general case.

Problem 2 : Climbing a 2D Mountain

Given an n by n matrix A of integers, a peak is an entry i, j such that $A_{i,j} \geq A_{i-1,j}$, $A_{i,j} \geq A_{i+1,j}$, $A_{i,j} \geq A_{i,j-1}$, and $A_{i,j} \geq A_{i,j+1}$ (if one of these entries doesn't exist, we ignore that condition).

(a) Prove that every matrix has at least one peak. Prove that for any n there are matrices that don't have more than one peak.

(b) Consider some column j of the matrix, and suppose $A_{i^\star,j}$ is the largest element in that column. Formally, this means that $A_{i^\star,j} \geq A_{i,j}$ for all $1 \leq i \leq n$. Suppose that $A_{i^\star,j} < A_{i^\star,j-1}$. Prove that there must be a peak in columns $1, \ldots, j-1$.

(c) Design an $O(n \log n)$-time algorithm to find a peak, show its correctness and analyze its running time.

Problem 3 : Signal Reconstruction

A sensor is taking measurements of a signal, which is a sequence of n integers between 1 and k. Each measurement is of the form (i, j, δ), which means that the absolute value of the difference between the i-th and j-th elements of the signal is δ. It's also known that the measurements don't have any cycles, i.e., the graph formed by the pairs (i, j) is acyclic. Your goal is to find any signal that is consistent with the measurements, or report that the measurements are inconsistent.

Formally, you're given n, k, and a list of m measurements, and you want to find a sequence a of length n such that for each measurement (i, j, δ), $|a_i - a_j| = \delta$ and $1 \leq a_i \leq k$ for all i. Your algorithm should take time $O(nk)$.

Problem 4 : Cell Towers

Suppose that there are m cell towers on a line with coordinates from 0 to n. The i-th tower is located at integer coordinate x_i.

(a) You want to build new cell towers in the middle of each pair of existing towers. More formally, for all pairs $1 \leq i < j \leq m$, you want to make sure there is a tower at coordinate $\lfloor (x_i + x_j)/2 \rfloor$. Describe an $O(m + n \log n)$-time algorithm to find how many new towers you need to build.

(Hint 1. You might want to determine for each coordinate c how many pairs of towers have c as their midpoint.)

(Hint 2. You might want to compute the quantity from the previous hint by writing two polynomials whose coefficients are related to the tower coordinates, and multiply them.)

(b) You want to find how many cell towers are equally spaced. More formally, you want to count the number of triples (i, j, k) such that $1 \leq i < j < k \leq m$ and $x_j - x_i = x_k - x_j$. Describe an $O(m + n \log n)$-time algorithm to find the number of such triples.

(Hint. The result from part (a) might be useful.)

Extra Credit

Recall that extra credits are quite challenging. We do not suggest attempting the extra credit problems to practice for the exam, but only to engage deeper with the course material. If you are interested in pursuing an IW/thesis in CS theory, the extra credits will give you a taste of what that might be like. You are welcome to discuss the extra credit problems with your TA/UCA coach.

Problem : A Unique Problem

You are given a sequence of non-negative integers $a_1, \ldots, a_n$. Your goal is to find if the sequence satisfies the following property: for every $1 \leq i \leq j \leq n$, the subsequence $a_i, a_{i+1}, \ldots, a_j$ contains at least one unique element. An element is unique in a subsequence if it appears exactly once in that subsequence. Design an $O(n \log n)$ algorithm to check if the sequence satisfies this property and prove its correctness and running time.