COS 330: Great Ideas in Theoretical Computer Science

Fall 2025

Problem Set 0

Module: Introduction

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 : Probability Practice

(a) Let X be a random variable such that there exists a single number c such that $\Pr[X = i] = \frac{c}{i^2}$ for all integers $i \geq 1$. Find the unique c such that X is indeed a well-defined random variable, and then compute $\mathbb{E}[X]$.

Aligning Expectations

You might need to find the value of some infinite series to solve this problem. You *don't* have to derive the value of the series yourself. Asking an AI assistant (or another online tool) to compute an infinite series for you is entirely appropriate.

(b) Let $X \sim \text{Unif}[a, b]$ be a uniform random variable on the interval $[a, b]$. That is, X is equally likely to be any real number between a and b. Find $\mathbb{E}[X]$.

You may use without proof the fact that the PDF of a uniform random variable on $[a, b]$ is $\frac{1}{b-a}$ on the interval $[a, b]$ and 0 otherwise. You may also use without proof the fact that the CDF of a uniform random variable on $[a, b]$ is 0 for $x < a$, 1 for $x > b$, and $\frac{x-a}{b-a}$ on $[a, b]$.

(c) A p-biased coin is one that lands heads with probability p, independently on every flip. Flip a p-biased coin n times. Each time the coin is Tails, throw it away. Each time the coin is heads, keep it independently with probability q (and throw it away otherwise). Let Y be the number of heads that are kept. Find $\mathbb{E}[Y]$.

Problem 2 : Climbing a Mountain

Given a sequence of n integers $a_1, a_2, \ldots, a_n$, a *peak* is an index $i \in \{2, \ldots, n-1\}$ such that $a_i \geq a_{i-1}$ and $a_i \geq a_{i+1}$.

(a) Prove that if $n > 2$, $a_1 < a_2$, and $a_n < a_{n-1}$, then the sequence has at least one peak.

(b) You are given a sequence of n ($n > 2$) integers such that $a_1 < a_2$ and $a_n < a_{n-1}$. Describe an $O(\log n)$-time algorithm to find a peak of a.

Problem 3 : Median Finding

The median of a list of n numbers (assume n is odd) is the $\lceil n/2 \rceil$-th smallest entry.[1] For example, the median of the list $[3, 1, 4, 1, 5]$ is 3.

Suppose you are given a list L of n numbers and you want to find a contiguous sublist of L with exactly k elements whose median is as large as possible. A contiguous sublist of L with k elements is a sublist formed by choosing a consecutive sequence of k elements from L. Assume k is always odd.

For example, if $L = [3, 1, 4, 1, 5]$ and $k = 3$, then the contiguous sublist $[4, 1, 5]$ has median 4, which is the largest possible median of any contiguous sublist of L.

(a) Suppose you are given the following simplified problem: given a list L of n numbers and a number x, decide whether there exists a contiguous sublist of L with k elements whose median is at least x. Describe an $O(n)$-time algorithm for this problem.

(b) Using your algorithm from part (a), describe an $O(n \log n)$-time algorithm to find a contiguous sublist of L with k elements whose median is as large as possible. (Hint: You might find it helpful to design a $O(n \log(\max(L) - \min(L)))$-time algorithm first.)

Problem Solving Tips

One trick to approaching a question like this is to ask yourself "why is this problem interesting?", or in other words, "why isn't the solution obvious?". We are asked to design an $O(n \log n)$-time algorithm

A good first step is to ask "what is a brute force solution that requires minimal creativity?" In this case, that would exhaust over $n-k$ possible contiguous sublists and compute the median

[1] That is, if you sorted the list, the entry at index $\lceil n/2 \rceil$.

of a list of size k from scratch. That would take total time $O(nk)$.

What do I learn from this? First, if this is an exam, I would definitely immediately write down "If $k = O(\log n)$, then the following brute force solution runs in time $O(kn) = O(n \log n)$ [and include a complete/correct analysis here]," and get my concrete partial progress. Second, I learn that the hard case is when k is large.

This may or may not ultimately lead me directly to a solution. For example, my instinct from here was actually to try and re-use information from one median computation for the next one, but I soon realized it's a reasonably advanced data structures problem to get right, but that's OK! Even though I threw all this work out and pivoted to the outline above instead, I got more confident/comfortable with the problem along the way.

Problem 4 : Solving 2-SAT

2-SAT is the problem of deciding whether a SAT formula with m clauses over n variables, where each clause has exactly 2 Boolean variables, has a satisfying assignment. For example, the formula (x_1 or x_2) and ($\neg x_1$ or x_3) and ($\neg x_2$ or $\neg x_3$) is satisfied by the assignment $x_1 = \text{true}, x_2 = \text{false}, x_3 = \text{true}$.

Consider the following randomized algorithm to find a satisfying assignment.

1. Start with an arbitrary assignment X_0 to the n variables.
2. Repeat until termination:
3. if the current assignment X_i satisfies all clauses, return it;
4. otherwise pick an unsatisfied clause uniformly at random, pick one of its two variables uniformly at random, and flip its value.

Assume the formula is satisfiable and fix a satisfying assignment a. Let d_i be the number of variable positions where X_i agrees with a, so $0 \leq d_i \leq n$.

(a) Show that for any step i where X_i does not satisfy the formula, we have $\Pr[d_{i+1} = d_i + 1] \geq \frac{1}{2}$. (Hint: In any clause that is unsatisfied by X_i, at least one of its two variables disagrees with a; sometimes both do.)

(b) Suppose a token starts at position 0. Time proceeds in steps $t = 0, 1, 2, \ldots$:

- If the token is at a position s with $1 \leq s \leq n-1$, flip a fair coin: on heads move to $s+1$; on tails move to $s-1$.
- If the token is at position 0, the next move goes to 1 with probability 1 (so the token never goes below 0). This is called a reflecting barrier at 0.

Let T be the number of steps until the token first reaches position n when starting from 0. Show that $\mathbb{E}[T] = n^2$. (Hint: it might help to write a formula for the expected number of additional steps to hit n when the token is at position s, and then expand it.)

What to do when you're stuck

You might get stuck on this problem (or any problem). That's normal. When you're stuck, it's good to be at least a little bit stubborn and try to push through (after all, that's your only option on an exam). But it's also good to recognize when you'll learn more with help getting unstuck.

Basic options are all good here: work in a PSet group, visit office hours, or try Ed. AI is now starting to be decent, and is totally fair game to try.

When doing so, you could ask it to give you a complete solution to the problem (for this problem, and comparably difficult problems in this course, it would give it to you), but that won't help you learn that much. The most helpful way for AI to help you would be for it to give you some hints, or advice on how to get started. Prompting an AI assistant effectively isn't easy, so here are some tips:

- Be very detailed with your prompt. For example, the prompt "Give me some advice on how to get started. Do not give me the solution to the problem" will likely still output the whole solution, but just not a detailed solution. A better prompt would be "Give me some advice on how to get started. Don't give me advice on the whole approach. Keep the advice short".
- Give the AI context about you and the class. For example, you can say "Assume that my background is only undergrad introductory probability" (you can be a lot more detailed than that, but this will already help a lot). If you don't do that, the AI assistant might give you solutions that use advanced tools you won't really understand. For example, in this problem, if the AI uses the word "martingale" then it is using an approach that is too advanced for this class.

(c) Using parts (a) and (b), complete the analysis of the algorithm and conclude that the expected number of iterations the algorithm needs to find a satisfying assignment (when one exists) is $O(n^2)$.

(d) Let's sanity check the result from part (c). Suppose we alter the algorithm in the following subtle way:

1. Start with an arbitrary assignment x_0 to the n variables.
2. Repeat until termination:
3. if the current assignment x_i satisfies all clauses, return it;
4. otherwise pick *any* clause uniformly at random, pick one of its two variables uniformly at random, and flip its value.

Why doesn't the same analysis go through?