Week 4: Lab 3

Lab 3

A warning about GenAI: Many of the problems given in the first few labs are classical problems with well known solutions. They are well known because they are useful learning problems. This means tools like ChatGPT and Copilot are almost certain to have been trained on similar problems. This means they are easily able to solve many of these problems. While this is useful sometimes, using these solutions will prevent you from learning what the labs are trying to teach you. Even worse, later labs will build on prior labs and increase in novelty and difficulty. Generative AI tools will not be able to solve them, and neither will you if you didn't understand the prior labs. We strongly suggest solving these problems yourself and writing your own code. You are free to use Gen AI, but you should consider it to be the similar to Googling the answer or asking your lab facilitator--only use it if you are stuck and make sure you understand the solution it produces.

This lab focuses on numerical algorithms. We will be looking at algorithms to count primes since these algorithms often have interesting time complexities that come up frequently in the study of algorithms. A key theme for today's lab is that algorithms with 2 nested loops aren't necessarily always O(N^2).

Counting Primes in O(N^2)

Go to the DOMjudge server (http://domjudge.gozz.au/) select lab3 and look at "countprimesmall". To solve this problem you will need to write a naive algorithm to count the number of primes under a threshold. You are aiming for a complexity of O(N^2) if the threshold is N. Recall that a number is prime if there is no number (other than 1 and itself) that divides it.

Reminder: You can check if x divides y using the modulo operator.
Hint: You can brute force this. Try every number up to the threshold. Then, for a each number, check if it's a prime by trying all possible divisors from 2 to x-1, where x is the number we are checking for primality.

Counting Primes in O(N sqrt N)

Next, look at "countprimesmedium". This problem is the same but the bounds have increased. If you submit your O(N^2) solution you should get a TIMELIMIT verdict. A faster algorithm is needed. For this version of the question, you are aiming for a complexity of O(N sqrt N), that is, N times the square root of N. Try to speed up the inner loop that checks if a number is prime.

Hint 1: In the O(N^2) algorithm, each time we found a divisor d, we actually find two divisors, since x/d must also divide x. Can you use this obervation to end the loop early?