CS 473, Spring 2023
Homework 5 (due Mar 8 Wed 9pm)

Instructions: As in Homework 1.

Problem 5.1: In this problem, we will investigate a simpler family of hash functions that satisfies a weaker version of universality (with some extra logarithmic factors).

Let m be a given integer. Let $p_1, \ldots, p_k$ be the list of all prime numbers between $m/2$ and m. You may assume that this list has been precomputed and you may use the known fact that $k = \Theta(m/\log m)$ (this follows from the well-known "Prime Number Theorem").

Pick a random index $j \in \{1, \ldots, k\}$ and define $h_j : \{0, 1, \ldots, U-1\} \to \{0, 1, \ldots, m-1\}$ by

$$h_j(x) = x \bmod p_j.$$

(a) (40 pts) For any fixed $x, y \in \{0, 1, \ldots, U-1\}$ with $x \neq y$, prove that $\Pr_j[h_j(x) = h_j(y)] \leq O(\frac{\log U}{m})$.

(Hint: given a number $z \leq U$, how many prime factors can z have that are at least $m/2$?)

(b) (60 pts) Recall the 3SUM problem: Given three sets of integers A, B, and C with $|A| + |B| + |C| = n$, we want to decide whether there exist elements $a \in A$, $b \in B$, and $c \in C$ such that $a + b = c$. Prof. X claims to have discovered an $O(n^{1.99})$-time algorithm to solve the special case of the problem when $A, B, C \subseteq \{0, 1, \ldots, n^4\}$. Show how to use Prof. X's algorithm to solve the more general case of the problem when $A, B, C \subseteq \{0, 1, \ldots, n^{100}\}$ by a Monte Carlo $O(n^{1.99})$-time algorithm with error probability at most 0.01.

(Hint: use part (a). The property that $h_j(a) + h_j(b)$ is equal to $h_j(a+b)$ or $h_j(a+b) + p_j$ may be helpful...)

Problem 5.2:

(a) (30 pts) Consider the following problem: given a directed graph $G = (V, E)$ with n vertices, decide whether G contains a directed cycle of length 5. Give a deterministic algorithm that solves this problem in $O(n^{2.81})$ time.

(b) (70 pts) Consider the following problem: given an *undirected* graph $G = (V, E)$ with n vertices, decide whether G contains a *simple* cycle of length 5. (A cycle is simple if no vertices appear more than once in the cycle.) Give a Monte Carlo algorithm that solves this problem in $O(n^{2.81})$ time with error probability at most 0.01.

(Hint: pick a random ordering of the vertices and use (a variant of) your algorithm for part (a).)

Problem 5.3: Consider the following geometric problem: given a set P of n points in 2D, with integer coordinates from $\{0, 1, \ldots, U-1\}$, find a *closest pair*, i.e., two points $p, q \in P$ ($p \neq q$) with the smallest Euclidean distance. Let $\delta(P)$ denote the distance of the closest pair.

We have seen an $O(n \log n)$-time divide-and-conquer algorithm from class. In this question, we give a different, faster randomized algorithm (which has the added advantage that it can be extended to higher dimensions).

(a) (35 pts) First give an $O(n)$-expected-time (Las Vegas) algorithm for the easier *decision problem*: given a value r, decide whether $\delta(P) < r$.

(Hints: Build a uniform grid where each cell is an $(r/2) \times (r/2)$ square. Use hashing. How many points can a grid cell have? For each grid cell, how many grid cells are of distance at most r?)

(b) (65 pts) Now, consider the following recursive Las Vegas algorithm to compute $\delta(P)$:

```
CLOSEST-PAIR(P):
1. if |P| ≤ 100 then return answer by brute force
2. partition P into subsets P_1, ..., P_20 each with at most ⌈n/20⌉ points
3. let S = {(i, j) | 1 ≤ i < j ≤ 20}
4. r = ∞
5. for each (i, j) ∈ S in random order do
6.     if δ(P_i ∪ P_j) < r then
7.         r = CLOSEST-PAIR(P_i ∪ P_j)
8. return r
```

Explain why the algorithm is always correct, and analyze its expected running time by solving a recurrence.

(Hints: Where is part (a) used? What is $|S|$? What is the expected number of times line 7 is performed, using facts from class?)