

# CS 473 (Spring 2025)

## Homework 5 (due Mar 13 Thu 10am)

**Instructions:** As in previous homeworks.

**Problem 5.1:** In this problem, we will explore other simple families of hash functions and see how close they are to being universal. Let  $U \geq m$ .

- (a) (35 pts) Let  $p$  be a random prime in  $[m/2, m]$ . Let  $b$  be a random number in  $\{0, \dots, p-1\}$ . Define  $h_p : \{0, \dots, U-1\} \rightarrow \{0, \dots, m-1\}$  by

$$h_p(x) = x \bmod p.$$

Prove that for any fixed  $x, y \in \{0, \dots, U-1\}$  with  $x \neq y$ , we have  $\Pr_p[h_p(x) = h_p(y)] \leq O((\log U)/m)$ .

(Thus, this hash function family is “almost” universal with an extra logarithmic factor.)

[*Hint:* By the prime number theorem, the number of primes in  $[m/2, m]$  is  $\Theta(m/\log m)$ . Given a number  $z \leq U$ , how many prime factors in  $[m/2, m]$  can  $z$  have?]

- (b) (35 pts) Suppose  $U = 2^w$  and  $m = 2^\ell$ . We can view numbers in  $\{0, \dots, U-1\}$  as binary vectors in  $\{0, 1\}^w$ , and numbers in  $\{0, \dots, m-1\}$  as binary vectors in  $\{0, 1\}^\ell$ .

Pick a random 0-1 matrix  $A$  of dimension  $\ell \times w$  (each of its  $\ell w$  entries is an independently chosen random bit). Define  $h_A : \{0, 1\}^w \rightarrow \{0, 1\}^\ell$  by

$$h_A(x) = (Ax) \bmod 2.$$

All arithmetic operations are done modulo 2; for example, addition is equivalent to exclusive-or.

Prove that for any fixed  $x, y \in \{0, 1\}^w$  with  $x \neq y$ , we have  $\Pr_A[h_A(x) = h_A(y)] = 1/m$ .

(Thus, this hash function family is universal, and its computation avoids integer division and requires only bitwise exclusive-or, which is cheaper. However, the evaluation time isn't  $O(1)$ .)

[*Hint:* suppose  $x$  and  $y$  differs at the  $i$ -th bit. Focus on choices over the  $i$ -th column of  $A$ ...]

- (c) (30 pts) Continuing part (b), let  $b$  be a random 0-1 vector of dimension  $\ell$ . Define  $h_{A,b} : \{0, 1\}^w \rightarrow \{0, 1\}^\ell$  by

$$h_{A,b}(x) = (Ax + b) \bmod 2.$$

Prove that for any fixed  $x, y \in \{0, 1\}^w$  with  $x \neq y$  and for any fixed  $s, t \in \{0, 1\}^\ell$ , we have  $\Pr_{A,b}[(h_{A,b}(x) = s) \wedge (h_{A,b}(y) = t)] = 1/m^2$ .

(Thus, this hash function family is 2-universal.)

[*Hint:* use (b).]

**Problem 5.2:** We are given a subset  $S$  of  $\{1, \dots, N\}$  and a number  $k \leq N$ . We want to compute a new set  $f_k(S) = \{a_1 + \dots + a_k : a_1, \dots, a_k \text{ are } k \text{ distinct elements of } S\}$ .

(Note that this is a little different from one of the midterm 1 practice problems, which is about computing the set  $S + \dots + S$  with  $k$  terms, where an element may be used more than once in the sum.)

- (a) (50 pts) First consider a related problem: given  $\ell$  disjoint sets  $S_1, \dots, S_\ell \subseteq \{1, \dots, N\}$  and a number  $k \leq \ell \leq N$ , compute a new set  $g_k(S_1, \dots, S_\ell) = \{a_1 + \dots + a_k : a_1, \dots, a_k \text{ are elements chosen from } k \text{ different sets among } S_1, \dots, S_\ell\}$ .

Give a (deterministic) algorithm that solves this problem in  $O(\ell^3 N \log N)$  time or better.

[Hint: use divide-and-conquer, like in the midterm 1 practice question. Recall that for  $A, B \subseteq \{1, \dots, M\}$ , we can compute  $A + B = \{a + b : a \in A, b \in B\}$  in  $O(M \log M)$  time by convolution/FFT. It may be helpful to extend the problem to computing  $g_k(S_1, \dots, S_\ell)$  for all  $k \leq \ell$  simultaneously.]

- (b) (50 pts) Now design and analyze a Monte Carlo algorithm to compute  $f_k(S)$ . The running time should be of the form  $O(k^c N \log^{c'} N)$  for some constants  $c, c'$ , and the overall error probability should be at most  $1/N$ .

[Hint: partition  $S$  into  $\ell$  disjoint sets randomly or by using a hash function, and then use part (a). How should we set  $\ell$ ?]

**Problem 5.3:** Consider the following geometric problem: given a set  $P$  of  $n$  points in 2D, with integer coordinates from  $\{0, 1, \dots, U - 1\}$ , 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  $|P| \leq 100$  then return answer by brute force
2. partition  $P$  into subsets  $P_1, \dots, P_{20}$  each with at most  $\lceil n/20 \rceil$  points
3. let  $S = \{(i, j) \mid 1 \leq i < j \leq 20\}$
4.  $r = \infty$
5. for each  $(i, j) \in S$  in random order do
6.     if  $\delta(P_i \cup P_j) < r$  then
7.          $r = \text{CLOSEST-PAIR}(P_i \cup P_j)$
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?)