CMPSC 565: Design and Analysis of Algorithm (**Due 2024-11-15 10:00pm**)

Problem Set 4 Solutions

Instructor: Dr. Antonio Blanca TAs: Qiyu Chen, Jianqiang Li

Notice: Type your answers using LaTeX and make sure to upload the answer file on Gradescope before the deadline. Recall that for any problem or part of a problem, you can use the "I'll take 20%" option. For more details and the instructions read the syllabus.

Problem 1. Pattern matching by fingerprinting

In the pattern or string matching problem, the goal to determine if a pattern (or substring) $Y = y_1y_2\dots y_m$ is contained or occurs in a much longer string $X = x_1x_2\dots x_n$. You may assume Y and X are binary strings. There is a trivial algorithm that checks at every valid position i of X whether Y is a substring of X starting at position i. This takes $O(mn)$ time, which could be quite inefficient.

Alternative sophisticated deterministic algorithms are known to solve this problem in $O(m + n)$ time (e.g., Knuth-Morris-Pratt and Boyer-Moore). You will design here a much simpler randomized algorithm for this task with running time $O(m + n)$ and probability of success nearly 1.

Consider the following algorithm:

1. Pick a prime p from $\{2, \dots, T\}$ uniformly at random. You may assume that this step can be done efficiently and you don't need to factor it into your running time calculation.
2. Compute the fingerprint $F(Y, p) = Y \bmod p$; here we are viewing Y as an m-bit binary number.
3. for $j = 1$ to $n - m + 1$
   - 3.1 Let $X(j)$ be the m-bit number $x_jx_{j+1}\dots x_{j+m-1}$. Compute $F(X(j), p)$.
   - 3.2 if $F(Y, p) == F(X(j), p)$, assume a match was bound, output the index j and stop.
4. If no match was found, output -1.

(a) Show that if $T = mn^2$, the probability that the algorithm makes an error goes to 0 as n grows. Specifically, you can show that such probability is $O(\log(n)/n)$. *Hint: You should try combining the probability of failure of the fingerprinting algorithm from lectures with a union bound.*

(b) Argue that if implemented as above the algorithm has $O(mn)$ running time.

(c) To speed up the algorithm, it is possible to compute $F(X(j + 1), p)$ from $F(X(j), p)$ efficiently. Show that:

$$F(X(j + 1), p) = 2F(X(j), p) - (2^m x_j \bmod p) + (x_{j+m} \bmod p).$$

(d) Show how to use the fact above to implement the algorithm in $O(m + n)$ running time.

Problem 2. Binomial Parity

Show that if X is a Binomial$(n, 1/2)$ random variable with $n \geq 1$, then the probability that X is even is $1/2$.

Problem 3. Die rolling

Suppose we roll a standard fair die over and over. What is the expected number of rolls until the first pair of consecutive sixes appears? (*Hint: the answer is not 36.*)

Problem 4. Birthdays

Let us assume that every year has exactly 365 days (i.e., no leap years) and that if we choose a person uniformly at random, the probability that his/her birthday is any given day of the year is exactly 1/365 (i.e., it is uniformly distributed). What is the smallest n for which, in any group of n (uniformly) random people, the probability that two of them have the same birthday is at least 0.9?

Problem 5. Speeding on the Grass

Consider the following problem: given an unsorted array A of n elements, we wish to sample from the middle half of the array. i.e. we want a procedure that returns an element a of A such that $a = A'[i]$ where A' is a sorted version of A and $i \in \mathbb{N}$ where $n/4 \leq i \leq 3n/4$.

Consider the following algorithm for this problem: choose $k = 10 \log n$ elements from A u.a.r., sort them, and return the median of the sorted elements. What is the time complexity of this algorithm? What is the probability that one iteration of this algorithm returns an element that is not from the middle half of the array?

You can use these inequalities: $\binom{k}{k/2} \leq \frac{1}{2} \times 4^{k/2}$, $\sum_{i=k/2}^{k} (1/3)^i \leq (1/3)^{k/2} \frac{3}{2}$, $(3/4)^{k/2} \leq (1/2)^{k/5}$. Assume that it takes $O(\log n)$ time to compare two elements and $O(k \log n)$ time to select k random elements from an array of length n. Note that for the error probability an exact calculation is difficult and a reasonable upper bound is fine.

Problem 6. Quicksort

Quicksort is a simple and efficient sorting algorithm. The input is a list of n distinct numbers $x_1, x_2, \ldots x_n$. The algorithm starts by choosing a pivot element among the set $\{x_1, x_2, \ldots x_n\}$, say x, and then proceeds by comparing every other element to x to partition the elements into two sublists: those that are less than x and those that are greater than x. Then, Quicksort is called recursively to sort these sublists.

(a) Show that for some initial ordering and some poor pivot choices, Quicksort performs $\Omega(n^2)$ comparisons.

(b) Assume now that the pivot is chosen uniformly at random from $\{x_1, x_2, \ldots x_n\}$. Show that the expected number of comparisons is $O(n \log n)$.

Hint: Let $y_1, y_2, \ldots y_n$ be the input but sorted, say increasingly. Let X be random variable for the number of comparisons Quicksort *does. We want to compute $E[X]$. Define X_{ij} to be the indicator random variable that is 1 if y_i and y_j are compared during the execution of the algorithm and 0 otherwise. Write down X as a function of the X_{ij}'s to simplify the expectation calculation.*

Problem 7. Coupon collecting by pairs

Consider the following variation of the coupon collector's problem. Each box of cereal contains one of $2n$ different coupons. The coupons are organized into n pairs, so that coupons 1 and 2 are a pair, coupons 3 and 4 are a pair, and so on. Once you obtain one coupon from every pair, you can obtain a prize. Assuming that the coupon in each box is chosen independently and uniformly at random from the $2n$ possibilities, what is the expected number of boxes you have to buy before you can claim the prize?

Problem 8. Generalization of Coupon Collector Problem

Let $U = \{x_1, x_2, x_3, \ldots, x_n\}$ be a set of n coupons. Define S_d as the set that contains all subsets of U with size at most d. In other words, S_d consists of every subset of U that has at most d elements. For example, when $d = n$, S_n is the power set of U, which includes all possible subsets of U.

Consider a process where each drawing selects a subset of coupons from S_d uniformly at random, meaning that each draw picks a subset of U of size at most d with equal probability. A coupon is collected if it has appeared in at least one of the drawn subsets. The goal is to collect all coupons in U, meaning each coupon in U must appear in at least one of the draw subsets. Specially, when $d = 1$. this is the standard coupon collector problem.

Given the parameter d, we are interested in determining the number of drawings needed to collect all n coupons in the following cases:

1. When $d = O(1)$, prove that the number of drawings required to collect all coupons is $O(n \log n)$.
2. When $d = O(\log n)$, prove that the number of drawings required to collect all coupons is $O(n)$.
3. When $d = \Theta(n)$, prove that the number of drawings required to collect all coupons is $O(\log n)$.

Hint: Use the fact that $\binom{n}{d} \le |S_d| = \sum_{i=0}^{d} \binom{n}{i} \le \binom{n}{d} \frac{n-d+1}{n-2d+1}$ when $d \le n/2$.

Problem 9. Close Games

Consider a balls-and-bins experiment with $2n$ balls but only two bins. As usual, each ball independently selects one of the two bins, both bins equally likely. The expected number of balls in each bin is n. In this problem, we explore the question of how big their difference is likely to be. Let the random variables X_1 and X_2 denote the number of balls in the two bins, respectively. Prove that for any $\varepsilon > 0$ there is a constant $c > 0$ such that the probability $\Pr\left[X_1 - X_2 \ge c\sqrt{n}\right] \le \varepsilon$.

Hint: use Chebyshev's inequality.

Problem 10. Cliche Cliques

Let $G \sim G(n,p)$ and let X be the random variable corresponding to the number of cliques of size 4 in $G = (V,E)$. Let Ω_4 be the set of all the subsets of size 4 of V. That is, $\Omega_4 = \{C \subset V : |C| = 4\}$ and so $|\Omega_4| = \binom{n}{4}$. Here $G(n,p)$ is a distribution of n node graphs generated by including each possible edge independently with probability p.

1. Show that if $pn^{2/3} \to \infty$, then $E[X] \to \infty$ and that if $pn^{2/3} \to 0$, then $E[X] \to 0$.

   Hint: Observe that $X = \sum_{C \in \Omega_4} X_C$ where X_C is the 0/1 random variable for whether C is a clique or not in G.

2. Use part (a) and Markov's inequality to show that $\Pr[G \text{ has a 4 clique}] \to 0$ when $pn^{2/3} \to 0$ (here $n \to 0$).

3. Show that the variance of X satisfies:

$$\mathrm{Var}(X) = \sum_{C \in \Omega_4} \mathrm{Var}(X_C) + \sum_{C,D \in \Omega_4 : D \ne C} \mathrm{Cov}(X_C, X_D),$$

   where the covariance is defined as $\mathrm{Cov}(X_C, X_D) = E[X_C X_D] - E[X_C]E[X_D]$.

4. Show that $\sum_{C \in \Omega_4} \mathrm{Var}(X_C) = O(n^4 p^6)$.

5. Show that $\sum_{C,D \in \Omega_4 : D \ne C} \mathrm{Cov}(X_C, X_D) = O(n^6 p^{11}) + O(n^5 p^9)$.

6. Use part 4., 5. and Chebyshev's inequality to show that $\Pr[G \text{ has a 4 clique}] \to 1$ when $pn^{2/3} \to \infty$ (here $n \to \infty$).

   Hint: Use Chebyshev's inequality to prove that it is sufficient that $\frac{\mathrm{Var}(X)}{E[X]^2} \to 0$.