Assignment 1

Randomized Algorithms

Due: Monday 12^{th}, by 11:55pm

Assignments are to be done individually. Submit a single pdf file to the classroom.

Problem 1 (2 points)**.**
There are n new year greeting cards that you have written up for n different people. There are n envelopes for the n cards to go into. You give your nephew the task of putting one card into each envelope, in return for a gift. The nephew, too eager to get his gift randomly places each card into an empty envelope. What is the expected number of people who receive the card meant for them ?

Problem 2 (2 points)**.**
There are n new year greeting cards that you have written up for n different people. There are n envelopes for the n cards to go into. You give your nephew the task of putting one card into each envelope, in return for a gift. The nephew, too eager to get his gift randomly places each card into an empty envelope. What is the probability that exactly r of the cards are in their right envelope? As $n \to \infty$ what is the probability that none of the cards are in the correct envelope?

Problem 3 (2 points)**.**
Design a randomized algorithm that finds the k^{th} smallest element in an array of n numbers. The expected running time of the algorithm should be $O(n)$.

Problem 4 (2 points)**.**
Suppose we have designed a new algorithm that can sort a given permutation of numbers. However, the algorithm is randomized, and we don?t know if the output is indeed a permutation. Design a randomized algorithm to check if the output is indeed a permutation, for an input of size n, and a given parameter $\epsilon > 0$. The algorithm must succeed with probability at least $\epsilon > 0$. Assume that you have access to a prime number generator that outputs a prime larger than n/ϵ and the elements of the input and output arrays in $O(1)$ time.

Problem 5. Consider the following variant of the quicksort algorithm: We repeat the pivot step until we find a pivot such that the number of elements on either side of the partition has at most 2/3 fraction of the elements. That is, we pick a pivot uniformly at random and compute the partition. If either side of the array has more than 2/3 fraction of the elements, we go back to picking a pivot uniformly at random. This process continues until we find a pivot such that either side has at most 2/3 fraction of the elements. What is the expected running time of this algortihm?