CS7800: Advanced Algorithms. Fall 2017
Homework 8

Instructor: Jonathan Ullman TA: Albert Cheu

Due Friday, November 17th at 11:59pm
(Email to neu.cs7800@gmail.com)

Homework Guidelines

Collaboration Policy. Collaboration on homework problems is permitted, however it will serve you well on the exams if you solve the problems by yourself. If you choose to collaborate, you may discuss the homework with at most 2 other students currently enrolled in the class. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, indicate that on your submission. If asked, you must be able to explain your solution to the instructors.

Preparing and Submitting Solutions. You must type your solutions using LaTeX. Please use an 11-pt or larger font. Please submit both the source and PDF files using the naming conventions `lastname_hw8.tex` and `lastname_hw8.pdf`. Your name must be on the first page of the PDF.

Solution guidelines For problems that require you to provide an algorithm, you must give the following: (1) a precise description of the algorithm in English and, if helpful, pseudocode, (2) a proof of correctness, (3) an analysis of running time. You may use any facts from class in your analysis and you may use any algorithms from class as subroutines in your solution.

You should be as clear and concise as possible in your write-up of solutions. Communication of technical material is an important skill, so clarity is as important as correctness. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand. Points might be subtracted for solutions that are too long.

Review Problems (Do Not Hand In)

In order to study randomized algorithms, it's important to have a certain comfort with the basics of discrete probability: counting, events, independence, expectation, conditional probability, etc. I *highly* recommend that you look at Exercises 1-5 in Chapter 9 of Jeff Erickson's generally excellent algorithms lecture notes. I would not necessarily recommending spending the time to completely solve every problem, but I would make sure that you understand how to solve all of the problems and understand all of the concepts in the problems. If you find some of these concepts foreign or feel too rusty to attempt the problems, I would recommend reviewing Chapters 16-18 in Mathematics for Computer Science or the material at the end of the Randomized Algorithms chapter of Kleinberg and Tardos.

Assigned Problems

Problem 1 (Equivalence of Different Random Number Generators, 10 points)**.**

In class, we said that a randomized algorithm is one that "can flip coins." That is, it has access to a subroutine FairCoin that returns a bit $\{0,1\}$, where each of the two values has probability exactly $1/2$, and each call to FairCoin returns an independent value in $\{0,1\}$. However, in the algorithms we saw in class, we used slightly higher-level operations such as choosing random bits $\{0,1\}$ where the probability of 1 is $1/k$, or choosing random integers in a specified range $\{1,2,\ldots,k\}$. In this problem we will see how to obtain these functionalities using FairCoin.

(a) Consider the following procedure OneThird:
If (FairCoin$= 0$) then output 0 otherwise output $1 -$ OneThird.

- **(a1)** Prove that the probability OneThird outputs 1 is $1/3$.
- **(a2)** What is the probability that OneThird makes *exactly* i calls to FairCoin before terminating?
- **(a3)** What is the expected number of calls OneThird makes to FairCoin before terminating?

(b) Now suppose we *do not* have FairCoin but we *do* have a procedure OneThird that outputs a bit that is equal to 1 with probability exactly $1/3$.

- **(b1)** Design a procedure that implements FairCoin using access to the procedure OneThird.
- **(b2)** What is the probability that your procedure makes *exactly* i calls to OneThird before terminating?
- **(b3)** What is the expected number of calls your procedure makes to OneThird before terminating?

(c) Now suppose we have FairCoin back. Design a procedure Random(n) that returns a uniformly random integer from the set $\{1,2,\ldots,n\}$. Prove your procedure is correct and analyze its running time, using the assumption that a call to FairCoin takes one unit of time.

Problem 2 (The Game Show, 10 points)**.**

You have been selected to appear on the hit game show *BigMoneySuitcaseThrow!* There are n suitcases, containing distinct enormous sums of money $v_1,\ldots,v_n$. The values $v_1,\ldots,v_n$ can be

completely arbitrary and you have absolutely no idea what they are. The suitcases are permuted uniformly at random and then violently thrown at you one by one. You catch each suitcase, open it, and then have to make a snap decision: either take the money in that suitcase and end the game, or throw the suitcase into the *GiantPitOfRegrets* and move on to the next suitcase.

You come up with the following type of strategy: Throw out the first t suitcases for some $1 \leq t < n$. Let m be the largest amount you saw among the first t suitcases. Among the remaining $n - t$ suitcases, take the first one you see whose value is larger than m. Let $v^* = \max\{v_1, \ldots, v_n\}$ be the largest amount of money you could win.

(a) What is the probability that you get v^* as a function of t and n?[1]

(b) Find the choice of t that maximizes the probability from part **(a)**. What is the optimal choice of t and what is the probability of getting v^*? To simplify your analysis, you may assume $n \to \infty$ and take limits. (That is, your solution only needs the right value of t as a function of n as $n \to \infty$.)

[1] Be careful! Remember that the only thing that is random is the order in which the suitcases are thrown at you. The numbers $v_1, \ldots, v_n$ are not random, and your strategy does not make any random decisions.