CS:5360 Fall 2018 Homework 2

Due: Thu, 9/20

Notes: (a) It is possible that solutions to some of these problems are available to you via text-books on randomized algorithms or on-line lecture notes, etc. If you use any such sources, please acknowledge these in your homework fully *and* present your solutions in your own words. You will benefit most from the homework, if you seriously attempt each problem on your own first, before seeking other sources. (b) As mentioned in the syllabus, it is okay to form groups of two in solving and submitting homework solutions. But, my advice from (b) still applies: you will benefit most from the homework, if you seriously attempt each problem on your own first, before seeking help from your group partner. (c) Discussing these problems with any of your classmates is okay, provided you and your classmates are not being too specific about solutions. In any case, make sure that you take no written material away from these discussions *and* (as in (b)) you present your solutions in your own words. When discussing homework with classmates please be aware of guidelines on "Academic Dishonesty" as mentioned in the course syllabus.

1. Suppose we roll a standard die 1000 times. Let X denote the sum of the numbers that appeared over the 1000 rolls. (a) Use Markov's inequality to upper bound $Pr(X \geq 5000)$. (b) Use Chebyshev's inequality to upper bound $Pr(X \geq 5000)$.

2. As mentioned in class, there is a simple randomized Las Vegas algorithm that solves the SELECTION problem. This is very similar to the randomized quicksort algorithm discussed in class. Here is an informal description of this algorithm. Suppose we want to find an element of rank k in the given list L (of n distinct elements). We pick a *pivot* index $p \in [1 \ldots n]$ uniformly at random and construct sublists $L_1 = \{u \in S \mid u < L[p]\}$ and $L_2 = \{v \in S \mid v > L[p]\}$. If $L[p]$ has rank k, we are done. Otherwise, we recurse on one of L_1 *or* L_2 looking for an element of appropriate rank.

   (a) State this algorithm precisely in pseudocode.

   (b) Show that the expected running time of this algorithm is $O(n)$ by using the following analysis approach.

   - For $i = 1, 2, \ldots$, let Y_i be the random variable that denotes the number of recursive calls made during which the size of the input is in the range

     $$\left(\left(\frac{2}{3}\right)^i n, \left(\frac{2}{3}\right)^{i-1} n\right].$$

     So for example, Y_1 is the number of recursive calls that are made while the size the input is more than $(2/3)n$. Similarly, Y_2 is the number of recursive calls made when the input size is more than $4n/9$, but at most $2n/3$. Show that $E[Y_i] = O(1)$ for all $i = 1, 2, \ldots$.
   - Let T_i be the random variable that denotes the total running time of all the recursive calls made during which the size of the input is in the range $((2/3)^i n, (2/3)^{i-1} n]$. Calculate $E[T_i]$.
   - Let T be the random variable that denotes the total running time of the algorithm. Express T in terms of T_i's and then calculate $E[T]$.

(c) Show that the expected running time of the algorithm is $O(n)$ by mimicking the approach used for analyzing randomized quicksort in class. Specifically, let X_{ij} denote the number of times y_i and y_j are compared by the algorithm. (Recall the definitions of y_i and y_j from lecture notes.) Let $X = \sum_i \sum_j X_{ij}$ denote the total number of comparisons made by the algorithm. Show that $E[X] = O(n)$.

3. The following problem models a simple distributed system wherein "agents" contend for resources but "backoff" in the face of contention. This is a situation that arises in wireless networks when wireless nodes contend for access to the wireless medium to send messages.

   The system evolves in round. Every round, balls are thrown independently and uniformly at random into n bins. Any ball that lands in a bin by itself is *served* and removed from consideration. The remaining balls are all thrown again in the next round. We begin with n balls in the first round and we finish when every ball is served.

   (a) If there are b balls at the start of a round, what is the expected number of balls at the start of the next round?

   (b) Let x_j be the expected number of balls left after j rounds. Show that $x_{j+1} \leq x_j^2/n$.

   (c) Use this to argue that if in every round the number of balls served was *exactly* the expected number of balls to be served, then all balls would be served in $O(\log \log n)$ rounds.

4. Suppose we flip a fair coin n times to obtain n random bits. Consider all $m = \binom{n}{2}$ pairs of bits in some order. Let Y_i be the exclusive-or of the ith pair of bits and let $Y = \sum_{i=1}^{m} Y_i$ denote the number of Y_i's that equal 1.

   (a) Show that each Y_i is 0 with probability 1/2 and 1 with probability 1/2.

   (b) Show that the Y_i's are not mutually independent.

   (c) Show that the Y_i's are pairwise independent.

   (d) Calculate $Var[Y]$ using that fact that if Y_i's are pairwise independent then $Var[Y] = \sum_{i=1}^{m} Var[Y_i]$.

   (e) Finally, use Chebyshev's Inequality to calculate an upper bound on $Pr(|Y - E[Y]| \geq n)$.
