CS 161: Homework 1

Submit via Gradescope by 3pm (PST), Janurary 20, 2017.

Instructions: Please answer the following questions to the best of your ability. Unless otherwise indicated, provide full and rigorous proofs and include all relevant calculations. When writing proofs, please strive for clarity and brevity (in that order). Please see the course website for submission instructions (including Gradescope submission pass-code) and collaboration policy. Cite any sources that you use. You may discuss these problems with at most two other students, though you must write your own solutions, in your own words.

This homework contains some questions involving probability as a refresher, since we will start using some probability in the next few lectures. The second portion of this homework has questions that build on the lecture material on asymptotic notation, MergeSort, and the analysis of MergeSort.

1. **What do you want?** (5 points) This course might require a significant amount of your time. Before spending this effort, it is worth understanding what *you* hope to get out of the course. What skills do you hope to learn, what topics do you think we'll cover that will stick with you five, or ten years down the road? *Write at least three sentences describing how you expect to benefit from taking this class,* and please keep these things in mind throughout the rest of the course as a source of motivation if/when things get tough. [I will also keep what you write in mind. . . .]

2. **Probability Refresher**

   (a) (1 point) Let X denote a random variable defined as $\Pr[X = 0] = \Pr[X = 1] = 1/2$. What is $\mathbf{E}[X]$, the expectation of X? (Explain your answer with at most one sentence.)

   (b) (1 point) What is the variance of X, $\mathbf{Var}[X]$? (Explain your answer with at most one sentence.)

   (c) (1 point) Suppose $X_1, \ldots, X_n$ are n independent random variables, with $\Pr[X_i = 1] = \Pr[X_i = 0] = 1/2$. What is $\mathbf{E}[X_1 + X_2 + \ldots + X_n]$, the expected sum of the variables? (Explain with at most one sentence.)

   (d) (1 point) What is the variance of the sum, $\mathbf{Var}[X_1 + X_2 + \ldots + X_n]$? (Explain with at most one sentence.)

   (e) (1 point) Could $\mathbf{E}[X_1 + X_2 + \ldots + X_n]$ change if the random variables still satisfy $\Pr[X_i = 1] = \Pr[X_i = 0] = 1/2$, but are *not* independent? (If the answer is 'no', explain with one sentence; if the answer is 'yes, the expected sum could change' give a concrete example for $n = 2$ illustrating this.)

   (f) (1 point) Could $\mathbf{Var}[X_1 + X_2 + \ldots + X_n]$ change if the random variables still satisfy $\Pr[X_i = 1] = \Pr[X_i = 0] = 1/2$, but are *not* independent? (If the answer is 'no', explain with one sentence; if the answer is 'yes, the expected sum could change' give a concrete example for $n = 2$ illustrating this.)

3. **A Random Bit** Suppose you have a lopsided coin that lands 'heads' (H) with probability 1/3, and 'tails' (T) with probability 2/3. You and a friend are trying to decide who should pay for dinner, and you would like to decide in such a way that the probability each of you pays is 1/2. You suggest the following scheme: you flip the coin twice, and if you see 'HT' then you pay, if you see 'TH' then the friend pays, and if you see either 'HH' or 'TT', then you decide to keep flipping. The pseudo-code for this procedure is as follows:

```
while Have-Not-Paid
    flip coin twice;
    if outcomes = 'H','T'
          You pay
    elseif outcomes = 'T','H'
          Friend pays
    endif
endwhile
```

(a) (2 points) Prove that the probability that you are still flipping the coin (and haven't yet paid) after having flipped the coin 20 times is at most 0.01. [Hint: what is the probability that you will pay after 2 tosses? What is the probability that you will pay after 4 tosses? 6 tosses? etc.]

(b) (3 points) Prove that this is a fair procedure—namely that if you use the procedure until someone pays, then the probability you pay is equal to the probability that your friend pays, which is equal to 1/2. [Here, think of a "proof" as a compelling argument that would be sufficient to convince a skeptical friend that you aren't trying to use your fancy math skills and your shady biased coin to trick them into paying with a higher probability.]

4. **Asymptotic Analysis.**

(a) For each of the following functions, indicate whether $f = O(g)$, $f = \Omega(g)$, or both ($f = \Theta(g)$). Rigorously prove your answers. (For example, by specifying some explicit constants $n_0, c > 0$ (or n_0, c_1, c_2 in the case that $f = \Theta(g)$) such that the definition of Big-Oh, Big-Omega, or Big-Theta is satisfied.) (1 point each)

(a)	$f(n) = n \log\left(n^3\right)$	$g(n) = n \log n$
(b)	$f(n) = 4n^2 + 2n \log n$	$g(n) = n^2$
(c)	$f(n) = 2^{2n}$	$g(n) = 3^n$
(d)	$f(n) = n \log n$	$g(n) = (\log n)^2$
(e)	$f(n) = 7^n$	$g(n) = n!$
(f)	$f(n) = n^{\log \log n}$	$g(n) = (\log n)^{\log n}$

(b) (2 points) Show that $\sum_{i=1}^{n} \log i = \Theta(n \log n)$.

(c) (2 points) Show that $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log n)$.

5. **MergeSort.** Recall in class that we analyzed the MergeSort algorithm by arguing that each Merge operation, when given as input two arrays whose lengths sum to m, requires $4m + 2 \leq 6m$ basic operations. Letting $T(n)$ denote the number of basic operations required by MergeSort to sort n numbers, we then argued that $T(n) \leq 2T(n/2) + 6n$, and $T(1) = 1$ and solved this recurrence relation by drawing out the computation/recursion tree, and analyzing the total cost of each level, to conclude that $T(n) \leq 6n(1 + \log n)$.

(a) (3 points) Give a tighter analysis of MergeSort, by directly analyzing the recurrence $T(n) = 2T(n/2) + 4n + 2$, and $T(1) = 1$ which is the recurrence we obtain if we use the actual number of operations that Merge requires, $4m + 2$, rather than bounding this crudely by $6m$. For simplicity, feel free to assume that the initial array size is a power of 2, namely $n = 2^s$ for some integer s. [Hint: write out the levels of the recursion, and calculate the total number of operations required at each level, then sum them: For example, the top level requires $4n + 2$, the next level requires $2\left(4\frac{n}{2} + 2\right) = 4n + 4$, etc.]

6. **MergeSort 3.0.** The standard MergeSort that we saw in class partitions the input into *two* pieces, recursively sorts the pieces, then Merges the two sorted parts. In this problem, we investigate the runtime if we instead partition the input into *three* pieces, and apply the same high-level idea. Below is the pseudo-code for the MergeSort3 algorithm. For simplicity, we will assume that the length of the input array is a power of 3, namely $n = 3^s$ for some integer $s = \log_3 n$:

```
MergeSort3,  Input array A
    n = length(A)
    B = MergeSort3(A[1:n/3]);
    C = MergeSort3(A[n/3+1:2n/3]);
    D = MergeSort3(A[2n/3:n]);
    return Merge(B, C, D)
```

(a) (3 points) Fill in the following pseudo-code for the Merge function so that it outputs a sorted list, assuming that the three input arrays are sorted (feel free to assume that all numbers are distinct):

```
Merge  Input arrays B, C, D
    m = length(B)+length(C)+length(D);
    S = emptyArray(m)
    ***
     :  (you write this part)
    ***
    return S
```

(b) (1 point) How many basic operations does your Merge function require if the *output*, S, has length m? For this problem, think of "basic operations" as comparisons, assignments, incrementing indices, and additions.

(c) (2 points) Analyze the runtime of MergeSort3 by writing out the computation tree, as we did in class for standard MergeSort. Feel free to give your answer in the form $cn \log n + O(n)$, for some specific constant c. (Namely, feel free to ignore the constant in front of the n, but do NOT ignore the constant in front of the $n \log n$.)

(d) (1 point) Is this better or worse than the runtime obtained in the tighter analysis of the standard MergeSort that you did in Problem 5? In one or two sentences, give an intuitive explanation for why the comparison ends up the way it does. [Note, you can still do this part, even if you did not do the previous part.)