CS200 - Problem Set 9

1. Suppose a group of n people each order a different flavor of ice cream at an ice cream shop. Suppose the server didn't keep track of who ordered which flavor, and just hands the ice cream out randomly.

   (a) Let X be a random variable that is the number of people who got handed the correct flavor. Let X_i be the indicator random variable that takes value 1 if person i gets the correct flavor (and 0 otherwise.) Write X in terms of a sum of the X_i.

   (b) Use linearity of expectation to determine the average number of people who get the correct flavor.

2. (a) Prove that for all integers $n \geq 0$, and any $r \in \mathbb{R}$ such that $r \neq 1$

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}. \tag{1}$$

   (b) Assume r is a constant. Explain why if $r > 1$, then $\frac{r^{n+1}-1}{r-1} = O(r^n)$. (You should find k and C. k and C can depend on r, since r is a constant.)

   (c) Assume r is a constant. Explain why if $r < 1$, then $\frac{r^{n+1}-1}{r-1} = O(1)$. (Find k and C. k and C can depend on r, since r is a constant.)

   (d) In class, we found using the tree method that for a recurrence relation of the form $T(n) = aT(n/b) + O(n^d)$ and $T(1) = O(1)$,

$$T(n) = n^d \sum_{i=0}^{\log_b n} \left(\frac{a}{b^d}\right)^k. \tag{2}$$

   Using parts (a), (b), and (c) of this problem, explain how to get from this expression to our final result:

$$T(n) = \begin{cases} O(n^d \log n) & \text{if } a = b^d \\ O(n^d) & \text{if } a < b^d \\ O(n^{\log_b a}) & \text{if } a > b^d. \end{cases} \tag{3}$$

   (The final case will use properties of logs, and is challenging. Please check out the resources on logs, and the hints on the last page, but if you are still stuck don't spend too much time on it.)

3. Suppose you have devised three different recursive algorithms to solve the same problem:

   - Algorithm I: Uses three recursive calls, where the input to each recursive call is a quarter of the size of the original problem. It uses linear time to combine the outputs of the recursive calls into the final output.

- Algorithm II: Uses two recursive calls, where the input to each recursive call is a 2/3 of the size of the original problem. It uses constant time to combine the outputs of the recursive calls into the final output.
- Algorithm III: Uses nine recursive calls, where the input to each recursive call is a 1/3 of the size of the original problem. It uses quadratic (power of 2) time to combine the outputs of the recursive calls into the final output.

(a) For each algorithm, put a big-O bound on the time complexity.

(b) State which of the algorithms you would use to solve a problem with input size 1000000.

4. Let $A = \mathbb{N}$, and $B = \{1, 2\} \times \mathbb{N}$. Show $|A| = |B|$.

5. Let S be the the set of functions from $\mathbb{N}$ to $\{0, 1\}$. Prove S is uncountably infinite.

6. How long did you spend on this homework?

Hints for $a > b^d$ case. You should get a term of the form $a^{\log_b n}$. Use the change of base formula to rewrite $\log_b n$ in terms of base a. Then you will have a raised to a log base a, and you can simplify.

It is also useful that $\log_x y = \log_y y / \log_y x = 1 / \log_y x$.