CS302 - Problem Set 1

Please familiarize yourself with the sections of the syllabus on the honor code and problem sets before starting this homework.

1. This problem will help you to review and recall logarithms, exponentiation, summation, trees, recurrence relations, and geometric series. Solutions for several parts of this question are on the final page so you can check that you are on the right track, but try to do as much as possible on your own first. A couple of log tricks that will be helpful: $x = y^{\log_y x}$, and $(x^y)^z = (x^z)^y = x^{yz} = x^{zy}$.

   Suppose a recursive algorithm whose runtime $T(n)$ on input size n is given by the recurrence relation:

   $$T(1) = O(1) \qquad T(n) = aT(n/b) + O(n^d), \tag{1}$$

   where a, b, and d are constants.

   (a) **[6 points]** What do a, b, and d represent?

   (b) **[3 points]** If we describe the structure of the algorithm as a tree where each call to the algorithm is represented by a node, the initial call of the algorithm is the root, and each recursive call produces a child node from the node that called it, how many levels will this tree have (from the root to the leaves)? Explain briefly.

   (c) **[3 points]**If we put the root at top and leaves at the bottom, and label the root as level 0, how many nodes are there in level k below the root? Explain briefly.

   (d) **[3 points]** What is the size of the input to these nodes/algorithm calls at level k? Explain briefly.

   (e) **[3 points]** At a single node at level k, how much work is done by that instance of the algorithm, excluding any work done in the recursive calls it makes. Explain briefly.

   (f) **[3 points]** How much total work is done, excluding the work done in further recursive calls, at all nodes at level k? Explain briefly.

   (g) **[3 points]** Using the previous answer, how much work is done over the whole algorithm? In other words, find an expression for $T(n)$ (Leave in summation notation.)

   (h) **[3 points]** Use the formula for geometric series to evaluate the sum from the previous question.

   (i) **[6 points]** Use big-O notation (i.e. take the asymptotic limit of large n) and rules for logarithms to simplify the results from the previous question for $a/b^d \neq 1$. Explain your steps.

   (j) **[2 points]** Put everything together into one "master" formula for evaluating these types of recurrence relations.

2. **[6 points]** Let $T(n)$ be the number of bit strings of length n where the number of 1's is a multiple of 3. Create a recurrence relation for $T(n)$. Explain your reasoning.

   See next page for hints if you are really stuck, and if needed, warm up with a couple of simpler problems: numbers of bit strings with an even number of 1's, number of bit strings with two consecutive 1's.

3. The elements of a bi-tonic list either only increase, only decrease, or first increase and then decrease.

   (a) **[9 points]** Write pseudocode for a recursive algorithm that finds the maximum value of a bi-tonic list of distinct integers.

   (b) **[11 points]** Prove your algorithm is correct.

   (c) **[3 points]** Create a recurrence relation for the runtime of your algorithms.

   (d) **[6 points]** Evaluate the relation using your result from problem 1, and also provide an intuitive explanation for why the runtime is what it is.

Hints:

1. The solution to part (g) is $n^d \sum_{k=0}^{\log_b n} \left(a/b^d\right)^k$.
   The solution to part (j) is

$$T(1) = O(1)$$

$$T(n) = \begin{cases} n^d \log_b n & \text{if } a = b^d \\ n^d & \text{if } a < b^d \\ n^{\log_b a} & \text{if } a > b^d \end{cases} \tag{2}$$

2.
   - Consider the possibilities for the final bit of the string, and how many strings there are in each case.
   - It is helpful to introduce additional functions: let $T_1(n)$ be the number of strings whose number of 1's is 1 mod 3 (has remainder 1 when divded by 3), and let $T_2(n)$ be the number of strings whose number of 1's is 2 mod 3 (has remainder 2 when divded by 3).
   - Note that $T_1(n) + T_2(n) + T_3(n) = 2^n$.