CS 312: Algorithms **Fall 2018**

Homework 6

Your Name: _____

Collaborators and sources: _____

You may work in groups, but you must write solutions yourself. List collaborators on your submission.

If you are asked to design an algorithm, please provide: (a) either pseudocode or a precise English description of the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm and (e) justification for your running time analysis.

Submission instructions. This assignment is due by noon on Thursday, Nov 1 in Gradescope (as a pdf file). Please review the course policies on the course home page about Gradescope submissions.

1. **(16 points) Solving Recurrences.** *(Work independently)* Consider an algorithm whose running time $T(n)$ on an input of size n satisfies the following recurrence:

   $$T(n) \leq aT(n/b) + cn,$$

   where we assume the recurrence holds when $n \geq 2$, and that $T(2) \leq c$.

   (a) (2 points) How many nodes are there at level i of the recursion tree?

   (b) (2 points) What is the input size for a problem at level i of the recursion tree?

   (c) (2 points) How much work is done in a single function call at level i of the recursion tree? (Just as in class, count only the work done in the function itself, excluduing recursive calls.)

   (d) (2 points) What is the total work done at level i of the recursion tree?

   (e) (2 points) How many levels are in the recursion tree?

   (f) (2 points) If $a < b$, what is the running time of the algorithm? Give your answer in big-O form.

   (g) (2 points) If $a = b$, what is the running time of the algorithm? Give your answer in big-O form.

   (h) (2 points) If $a > b$, what is the running time of the algorithm? Give your answer in big-O form.

   Hint: remember the following fact about a geometric sum when $0 < r < 1$:

   $$\sum_{i=0}^{d} r^d = 1 + r + r^2 \ldots + r^d = \frac{1 - r^{d+1}}{1 - r} \leq \frac{1}{1 - r}$$

2. **(20 points) Database Medians**. You are working as a programmer for Mount Holyoke administration, and they ask you to determine the median GPA for all students. However, their system is really out of date, and student GPAs are stored in two different databases, one for students with last names that begin with A–L, and another for students with last names that begin with M–Z. Assume there are n students in each database, so there are $2n$ students total. You'd like to determine the median of this set of $2n$ values, which we will define here to be the nth smallest value.

   However, security is very tight, so the only way you can access these values is through queries to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the kth smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

   Give an algorithm that finds the median value using at most $O(\log n)$ queries.

3. **(10 points) Butterflies (Again?).** You have collected n butterfly specimens of different species. You want to know if there is a single species that accounts for more than half of the specimens. You don't know butterflies well enough to name the species of any single specimen, but you can carefully compare any two specimens and judge (correctly) if they are from the same species or not. Design an algorithm that returns "true" if there are more than $n/2$ specimens from one species, and returns "false" otherwise, using only $O(n \log n)$ pairwise comparisons.

   You do not need to prove correctness for this problem.

4. **(10 points) Proof by Induction for Recurrences.** (*Work independently.* Do this problem after the lecture on Monday, October 29.) Consider the following recurrence, which describes an algorithm that divides a problem of size n into two equal-sized subproblems, but then does $O(n \log n)$ outside of the recursive calls:

$$T(n) \leq 2T(n/2) + cn \log n,$$

   We again assume the recurrence holds for $n \geq 2$ and that $T(2) \leq c$. Prove by induction that $T(n) \leq cn(\log n)^2$. (Another way to say this is to say that $T(n)$ is $O(n \log^2 n)$.) You should assume that the logarithm is base 2, so that $\log(n/2) = \log n - 1$.

   Hint: see p. 213 of the book and lectures notes from Monday. The algebra for this proof is slightly more difficult but it follows the same pattern.