Problem 1. Solving Recurrences.

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) How many nodes are there at level $i$ of the recursion tree?

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

(c) 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, excluding recursive calls.)

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

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

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

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

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

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

$$\sum_{i=0}^{d} r^i = 1 + r + r^2 \ldots + r^d = \frac{1 - r^{d+1}}{1 - r} \leq \frac{1}{1 - r}$$
Problem 2. Proving by Induction.

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 $\log(n/2) = \log n - 1$. 
Problem 3. (Optional) Finding Median.

Work on the following problem from homework 4: You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains \( n \) numerical values - so there are \( 2n \) values total - and you may assume that no two values are the same. You'd like to determine the median of this set of \( 2n \) values, which we will define here to be the \( n^{th} \) smallest value.

However, 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 \( k^{th} \) 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 \( \mathcal{O}(\log n) \) queries.