Problem Set 4

mst3k

data structures and algorithms 2 - spring 2024

due february 21, 2024 at 11:59p

Collaboration Policy: You are encouraged to collaborate with up to 4 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: list your collaborators

Sources: list your sources

problem 1 *Solving Recurrences*

Prove a (as tight as possible) O (big-Oh) asymptotic bound on the following recurrences. You may use any base cases you'd like.

1. For the following recurrence, it may be helpful to draw out the tree. However, you should prove the asymptotic bound using induction.
   - $T(n) = T(\frac{n}{3}) + T(\frac{n}{9}) + n$

     Solution:

2. For the following recurrence relations, indicate: (i) which case of the Master Theorem applies (if any); (ii) justification for why that case applies (if one does) i.e., what is a, $f(n)$, ε, etc; (iii) the asymptotic growth of the recurrence (if any case applies).
   - $T(n) = 4T(\frac{n}{2}) + 42n^2$

     Solution:
   - $T(n) = 4T(\frac{n}{2}) + 7n^2 \log n + 5n$

     Solution:

problem 2 *Disney+*

The executives at Disney+, like those at Netflix, are worried about password sharing and have asked you to look into the problem. They come to you with a list of n total login instances and provide an *equivalence tester* that tells you, in constant time, if two logins were produced from the same account. Specifically, due to privacy concerns, they do not share full login details and will only allow your algorithm to compare the login equivalence of two of the items in the list at a time.

Design an algorithm to determine if there exists a set of at least $\lceil \frac{n}{2} \rceil$ logins that were from the same account. Your algorithm must solve this problem in $\Theta(n \log n)$ total invocations of the *equivalence tester* Disney provided.

Solution:

PROBLEM 3 *Castle Hunter*

We are currently developing a new board game called *Castle Hunter*. This game works similarly to *Battleship*, except instead of trying to find your opponent's ships on a two dimensional board, you're trying to find and destroy a castle in your opponent's one dimensional board. Each player will decide the layout of their terrain, with castles placed on each hill. Specifically, each castle is placed such that they are higher than the surrounding area, i.e. they are on a local maximum, because hill tops are easier to defend. Each player's board will be a list of n floating point values. To guarantee that a local maximum exists somewhere in each player's list, we will force the first two elements in the list to be (in order) 0 and 1, and the last two elements to be (in order) 1 and 0.

To make progress, you name an index of your opponent's list, and she/he must respond with the value stored at that index (i.e., the altitude of the terrain). To win you must correctly identify that a particular index is a local maximum (the ends don't count), i.e., find one castle. An example board is shown in Figure 1. [We will require that all values in the list, excepting the first and last pairs, be unique.]

0	1	4	23	18	14	15	13	1	0
0	1	2	3	4	5	6	7	8	9

Figure 1: An example board of size $n = 10$. You win if you can identify any one local maximum (a castle); in this case both index 3 and index 6 are local maxima.

Devise a strategy which will guarantee that you can find a local maximum in your opponent's board using no more than $O(\log n)$ queries, prove your run time and correctness.

Solution: