COSC 311: ALGORITHMS
HW2: RECURRENCES AND PROOFS

Due Friday, October 6, 12pm

Please type up your solutions. If you discuss any of the problems with your classmates, please note at the top of your submission with whom you consulted.

1 Practice with Recurrences

1) An erroneous inductive proof.
Consider the following recurrence:

$$T(n) = \begin{cases} 3T(n/3) + n & n > 1 \\ 1 & n = 1 \end{cases}$$

where n is a power of 3.

(a) Here is an *incorrect* theorem and proof about this recurrence:

Theorem 1. $T(n) \in O(n)$.

Proof. Our proof will be by induction.
Base case: $n = 3$. Then we have:

$$T(3) = 3T(1) + 3 = 3 \cdot 1 + 3 = 6 \leq cn$$

for $c = 2$.
Inductive hypothesis: For some $n \geq 3$, $T(n) \leq cn$.
Inductive step: Assume the inductive hypothesis holds. We will show that $T(3n) \leq 3cn$. We have:

$$\begin{aligned} T(3n) &= 3T(3n/3) + 3n && (1) \\ &= 3T(n) + 3n && (2) \\ &\leq 3cn + 3n && (3) \\ &= O(n). && (4) \end{aligned}$$

Hence $T(n) \in O(n)$. □

Where is the mistake in the proof?

(b) What is the correct asymptotic class for this recurrence? Prove by induction that your guess is correct.

2) Does the Master Theorem apply?
For each of the following recurrences, check whether the Master Theorem applies. If it does, give the asymptotic class of $T(n)$. If not, explain why not.

(a) $T(n) = 27T(n/3) + n^2$

(b) $T(n) = 2T(n/4) + n$

(c) $T(n) = 4T(n/2) + n \sin n$

(d) $T(n) = 8T(n/2) + 4n^3$

(e) $T(n) = 3T(n/3) + \frac{n}{\lg n}$

2 Selection

In the *selection* problem, our input is an array `a` of ints (in no particular order). Our goal is to find the kth smallest element in the array (where $k = 0$ is the smallest element). For example, if our array is `a = <3 7 9 1 6 8>` and $k = 3$, the algorithm should give 7 as the output since 7 is the 3rd smallest element (again, where $k = 0$ is the smallest element).

Clearly, one way to solve the selection problem is to sort the array and then return `a[k]`. But as we have seen, sorting requires $\Omega(n \lg n)$ time and we would like to solve the selection problem faster than that.

3) Modified quicksort.
Design an algorithm called quickselect that uses a modified version of (deterministic) quicksort to solve the selection problem in average time $\Theta(n)$.

(a) Write pseudocode for your algorithm.

(b) Assume your choice of pivot always produces a partition that has pn elements to the left of the pivot (i.e., smaller than the pivot) and $(1 - p)n$ elements to the right of the pivot (i.e., bigger than the pivot). Write a recurrence for the runtime of your algorithm, $T(n)$ (assume that $p \geq 1/2$).

(c) Prove by induction that $T(n) = O(n)$.

4) Worst case linear-time selection.
Unfortunately, in the worst case the quickselect algorithm from problem (4) takes time $O(n^2)$ in the worst case. We would like to design a deterministic selection algorithm that takes time $O(n)$ in the worst case to find the kth smallest element. Consider the following selection algorithm:

1. Divide the n elements into $\lfloor n/5 \rfloor$ groups of 5 elements each, and possibly one additional

group with $n \mod 5$ elements.

2. For each group, find its median. Do this by insertion sorting the elements in the group and choosing the middle element from the sorted list.

3. Recursively find the median of the $\lceil n/5 \rceil$ medians (if $\lceil n/5 \rceil$ is even, use the smaller of the two median elements). Call this median-of-medians x.

4. Partition the original input array (as in quicksort), using x as the pivot. Let q be the index of x in the partitioned array (so there are $q-1$ elements less than x and $n-q$ elements greater than x).

5. If $k = q$, then x is the kth smallest element and we are done. Otherwise recursively find the kth smallest element on the "small" side if $k < q$, or find the $(k-q)$th smallest element on the "big" side if $k > q$.

(a) For each step of the above algorithm, give the runtime of that step (if the step involves a recursive call, express that step's runtime in terms of the runtime of the smaller subproblem). Put the pieces together to write a recurrence that expresses the runtime of the entire algorithm, $T(n)$.

(b) Prove inductively that $T(n) = O(n)$.

3 Divide and Conquer

5) Probing binary trees.
Consider an n-node complete binary tree, where $n = 2^d - 1$ for some d. Each node of the tree x stores some value v_x. You can assume that all of the values are distinct, i.e., $v_x \neq v_y$ for all $x \neq y$. A node x is considered a *local minimum* if its value v_x is less than the value v_y for all nodes y that are connected to x by an edge.

You can look up the value of a node x by *probing* the node. Assume that probing is a constant-time operation. Your goal in this problem is to come up with an algorithm that finds a local minimum in time $O(\lg n)$.

(a) Give a precise description of your algorithm. You may write pseudocode or state the steps of your algorithm as in problem (5) above.

(b) Prove, using induction, that your algorithm is correct.

(c) Write a recurrence for the runtime of your algorithm, $T(n)$. Use the Master Theorem to show that $T(n) = O(\lg n)$.

6) Database queries.
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 want to determine the median of this set of $2n$ values, which we will define to be the nth smallest value.

The only way you can access these values is through queries to the database. 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.

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

(b) Prove, using induction, that your algorithm is correct.

(c) Write a recurrence for the runtime of your algorithm, $T(n)$. Use the Master Theorem to show that $T(n) = O(\log n)$.

4 Submission

This assignment is due on Friday, October 6, at 12pm. Please type up your solutions and bring a hard copy of your typed responses to submit in class.