CS5002 Discrete Structures	Prof. Amor
Fall 2024	November 7, 2024

Homework # 9

Assigned: Thursday November 7, 2024
Due: Tuesday November 12, 2024 @ 11:59:00 pm

Instructions:

- Homework is due on Tuesday at 11:59 pm Boston Time. Homeworks received up to 24 hours late will be accepted with no penalty. *NO* assignment will be accepted after that.
- We expect that you will study with friends and often work out problem solutions together, but *you must write up your own solutions, in your own words.* **Cheating will not be tolerated**. Professors, TAs, and peer tutors will be available to answer questions but will not do your homework for you. One of our course goals is to teach you how to think on your own.
- Assignments should be typed using Word or LaTeX, or hand-written *neatly.* Do write and submit your answers as if they were a professional report. There will be point deductions if the submission isn't neat (is disordered, difficult to read, scanned upside down, etc. . . .). When submitting to gradescope be sure to indicate the page containing your answer to each problem.
- *To get full credit, SHOW YOUR WORK!*

Problem 1 [20pts: 10,10]**: Search Algorithms**

A company database has 10,000 customers sorted by last name, 30% of whom are known to be good customers. Under typical usage of this database, 60% of lookups are for the good customers. Two design options are considered to store the data in the database:

1. Put all the names in a single array and use binary search.
2. Put the good customers in one array and the rest of them in a second array. Only if we do not find the query name on a binary search of the first array do we do a binary search of the second array.

Given these options, answer the following.

i. Calculate the expected worst-case performance for each of the two structures above, given typical usage. Which of the two structures is the best option?

ii. Suppose that over time the usage of the database changes, and so a greater and greater fraction of lookups are for good customers. At what point does the answer to part **i** change?

iii. Under typical usage again, suppose that instead of binary search we had used linear search. What is the expected worst-case performance of each of the two structures and which is the better option? Where is the cross-over this time?

Problem 2 [22pts: 4,8,10]**: Merge Sort**

1. If mergesort divided its input array into five pieces instead of two, calling mergesort on each piece and combining with a linear-time 5-way merge, what would its recurrence be?
2. Find $T(n)$ for $n = 16$ if $T(1) = 1$ and $T(N) = 4T(N/2) + 1$.
3. Using the method of substituting the recurrence into itself repeatedly to find an equation in terms of T(1), find a non-recursive formula for $T(N)$ from the preceding problem. You can approximate and ignore constants.

Problem 3 [10 pts]**: Growth of Functions**

Organize the following functions into six columns. Items in the same column should have the same asymptotic growth rates (they are big-O and big-Θ of each other). If a column is to the left of another column, all its growth rates should be slower than those of the column to its right.

n^2, $n!$, $n \log_2 n$, $3n$, $5n^2 + 3$, 2^n, 10000, $n \log_3 n$, 100, $100n$

Problem 4 [20 pts (8,2,10)]**: Recurrences & Induction**

Solve the following recurrence using the substition method. Assume a base case of $T(1) = 1$.

$T(n) = T(n-1) + 2$.

1. Find a non-recursive formula for $T(N)$ in terms of $T(1)$. You will need to establish a pattern for what the recurrence looks like after the k-th iteration.
2. Give the big-O expression for your recurrence.

3. Prove that your pattern is correct using Induction.

Problem 5 [28 pts (8, 10, 10)]**: More Recurrences**

Solve the following recurrence using the substition method. Assume a base case of $T(1) = 1$.

1. Find a non-recursive formula for $T(N)$ in terms of $T(1)$. You will need to establish a pattern for what the recurrence looks like after the k-th iteration. You *do not* need to formally prove that your patterns are correct via induction, but you will lose points if your patterns are not correct. Your solutions may include integers, n raised to a power, and/or logarithms of n. For example, a solution of the form $8^{\log_2 n}$ is unacceptable; this should be simplified as $n^{\log_2 8} = n^3$.

2. Give the big-O expression for each of the recurrences.

i. $T(n) = 2\,T(n-1) + 1$.

ii. $T(n) = 9\,T(n/3) + n^2$.

iii. $T(n) = 6\,T(n/3) + n$.