Problems:

1 (text) Recurrence [10 points] Solve the following recurrence relation using repeated substitution.

$$T(n) = 5T(n/7) + 3n^2$$

Then solve it by using the Master Method, showing in detail which rule applies

Note: Show your work. You will get 4 points if you identify the pattern, 3 points if you do the proof work necessary to show what it resolves to, and 3 points for solving it using the master theorem.

$$T(n) = 5T\left(\frac{n}{7}\right) + 3n^2$$

$$= 5\left[5T\left(\frac{n}{7^2}\right) + 3\left(\frac{n}{7}\right)^2\right] + 3n^2 = 5^2 T\left(\frac{n}{7^2}\right) + 5 \cdot 3\frac{n^2}{7^2} + 3n^2$$

$$= 5^3 T\left(\frac{n}{7^3}\right) + 5^2 \cdot 3\frac{n^2}{7^2} + 5 \cdot 3\frac{n^2}{7^2} + 3\frac{n^2}{7^2} + 3n^2$$

$$= T(n) = 5^i T\left(\frac{n}{7^i}\right) + 3n^2 \sum_{j=0}^{i-1} \left(\frac{5}{49}\right)^j$$

* $\frac{n}{7^i} = 1$:

$$i = \log_7 n$$

$$T(n) = n^{\log_7 5} T(1) + 3n^2 \sum_{j=0}^{\log_7 n - 1} \left(\frac{5}{\quad}\right)^j$$

$$\sum_{j=0} (\tfrac{5}{7})$$

Since $\frac{5}{7} < 1$, geometric sum is $O(1)$:

$$= n^{\log_7 5} + O(n^2)$$

Since $\log_7 5 < 2$, dominant term

$$\boxed{O(n^2)}$$

Master Theorem:

$$a = 5,\ b = 7,\ f(n) = 3n^2$$

$$\log_7 5 \approx 0.827$$

Compare:

$$f(n) = \Omega(n^{\log_7 5 + \epsilon}) \text{ for some } \epsilon > 0$$

Check regularity condition:

$5f\left(\frac{n}{7}\right) = 5 \cdot 3\frac{n^2}{49} = \frac{15}{49}n^2 \leq cn^2$ for $c < 1$

Case 1 (f(n) dominates):

$\Theta(n^2)$

2 (text) Master Theorem [20 points] Apply the Master method to solve each of the following recurrences, or state that the Master method does not apply. Justify your answers. Note that the Master method covers all the cases

a. $T(n) = 5T(\frac{n}{9}) + n^7$

b. $T(n) = 3T(\frac{n}{3}) + 7n$

c. $T(n) = 15T(\frac{n}{4}) + 100$

d. $T(n) = 9T(\frac{n}{3}) + n^2$

e. $T(n) = 6T(\frac{n}{5}) + n^3$

(a.) $T(n) = 5T(n/9) + n^7$

$a = 5,\ b = 9,\ f(n) = n^7$

$\log_9 5 = \frac{\log 5}{\log 9} \approx 0.73$

since n^7 dominates $n^{0.73}$ → Master Theorem

Case 1

$$T(n) = \Theta(n^7)$$

(b) $T(n) = 3T(n/3) + 7n$

$a = 3,\ b = 3,\ f(n) = \Theta n$

$\log_3 3 = 1$

since $f(n) = \Theta(n^{\log_3 3})$ → Master Theorem

Case 2

$$T(n) = \Theta(n \log n)$$

(c) $T(n) = 15T(n/4) + 100$

$a = 15,\ b = 4,\ f(n) = \Theta(1)$

$\log_4 15 \approx 1.95$

since $f(n) = O(n^{\log_4 15 - \epsilon})$ → Master Theorem

Case 3

$$T(n) = \Theta(n^{\log_4 15})$$

(d.) $T(n) = 9T(n/3) + n^2$

$a = 9, \quad b = 3, \quad f(n) = \Theta(n^2)$

$\log_3 9 = 2$

since $f(n) = \Theta(n^{\log_3 9})$ → Master Theorem

Case 2

$$T(n) = \Theta(n^2 \log n)$$

(e.) $T(n) = 6T(n/5) + n^3$

$a = 6,\ b = 5,\ f(n) = \Theta(n^3)$

$\log_5 6 \approx 1.11$

Since n^3 dominates $n^{1.11}$ → Master Theorem Case 1

$$T(n) = \Theta(n^3)$$

3 (text) Radix Sort [5 points] Lexicographical ordering means order of the dictionaries to sequences of ordered symbols therefore (a < b < c < d < e < f < … < m < n < o < … < y < z). The same logic applies to Uppercase letters. Non-alphabetical characters have higher precedence that alphabetical characters

Illustrate the operation of Radix-Sort on the following list of strings using lexicographic ordering.

BOY,RIS,GUA,ATL,SUC,CES,QUI,PUT,ANT,VAU,HUI,VAL,SAN,SAN,CAQ,VIC,CAL,LA_GUA, TOL,META,CAU,BOL,AMA,MAG,CHO,NSA,GUA,CAS,COR

Radix Sort on strings:

Step 1: Sort by 3rd letter

A: GUA, GUA, LA_GUA, META, AMA, NSA
C: SUC, VIC
G: MAG

I: QUI, HUI
L: ATL, VAL, CAL, TOL, BOL
N: SAN
O: CHO
Q: CAQ
R: COR
S: RIS
T: PUT, ANT
U: VAU
Y: BOY

GUA, GUA, LA-GUA, META, AMA, NSA, SUC, VIC, MAG, QUI, HUI, ATL, VAL, CAL, TOL, BOL, SAN, CHO, CAQ, COR, RIS, PUT, ANT, VAU, BOY

Step 2: Sort by 2nd letter

A: LA_GUA, MAG, VAL, CAL, SAN, CAQ, VAU, CAU
E: META, CES
H: CHO
I: VIC, RIS

N: ANT
O: TOL, BOL, COR, BOY
S: NSA
T: ATL
U: GUA, GUA, SUC, QUI, MUI, PUT

LA_GUA, MAG, VAL, CAL, SAN, CAQ, VAU, CAU, META, CES, CHO, VIC, RIS, ANT, TOL, BOL, COR, BOY, NSA, ATL, GUA, GUA, SUC, QUI, MUI, PUT

Step 3: Sort by 1st letter

A: AMA, ANT, ATL
B: BOL, BOY
C: CAL, CAQ, CAS, CAU, CES, CHO, COR
M: MUI
L: LA_GUA
M: MAGA, META
N: NSA
P: PUT
Q: QUI

R: RIS
S: SAN, SUC
T: TOL
V: VAL, VAV, VAC

Final sorted order:

AMA, ANT, ATL, BOL, BOY, CAL, CAQ, CAS, CAV, CES, CHO, COR, MUI, LA-GUA, MAGA, META, NSA, PUT, QUI, RIS, SAN, SUC, TOL, VAL, VAV, VAC

7 (text) Algorithm Analysis [15 points]
For each of the algorithms you wrote for problems 4-6, explain their time complexity and space complexity using Big-O notation. Explain how you arrived at your answer.

Problem 5:

The time complexity is $O(n^2)$ because I had 2 loops that go through the data. I think the first loop runs

n times and inside it, the second loop can run up to n times too. So that makes it n×n, which is $O(n^2)$

The space complexity is $O(1)$ because I didn't use any extra large data structures. I just used a few variables.

Problem C:

The time complexity is $O(n \log n)$ because the first thing I did was sort the input. I believe sorting takes about n log n time in the worst case (for example if we use quick sort or merge sort.) After sorting I had a loop that just goes through the array once to finish solving the problem, which is $O(n)$. But when we add $O(n \log n)$ and $O(n)$ the $O(n \log n)$ part is the one that matters most because as n gets

bigger n log n grows faster than n. So, I believe the time complexity is O(n log n)

The space complexity is O(n) because I needed extra space to store either the sorted array or another array or list that I made during the process. Even though I'm not making extra space for every step, the extra space I use is depends on how big the input is, so it grows with n.