CMSC 351 Summer 2025 Homework 6

Due Monday 4 August 2025 by 11:59pm on Gradescope.

Directions:

- Homework must be done on printouts of these sheets and then scanned properly, or via Latex, or by downloading, writing on the PDF, and uploading. If you use Latex please do not change the Latex formatting.
- Do not use your own blank paper!
- The reason for this is that Gradescope will be following this template to locate the answers to the problems so if your answers are organized differently they will not be recognized.
- Tagging is automatic, you will not be able to manually tag.

---

1. Suppose we are using Counting Sort and we have the following non-cumulative `POS`:

i	0	1	2	3
Non-Cumulative `POS[i]`	1	2	3	1

→ 1+2+3+1 = 7

(a) How many elements are in the original list that we are sorting? [4 pts]

Number of elements:	7

(b) At the end, what is the sorted list? You will not need all the blanks below so only fill in as far as you need! [6 pts]

i	0	1	2	3	4	5	6	7	8	9
Sorted `A[i]`	0	1	1	2	2	2	3			

one 0 two 1's three 2's one 3

2. We are using Counting Sort to sort the following list of numbers: [6 pts]

i	0	1	2	3	4	5	6
Original A[i]	?	?	3	0	1	?	?

Fill in the blank entries in the following:

i	0	1	2	3
Non-Cumulative POS[i]	1	1	3	2
Cumulative POS[i]	1	2	5	7

3+ cum pos[1] = 5

2=1's + 0's so 1 1 and 1 0

3. Suppose we are using Counting Sort and we have the following cumulative POS:

Note: all the elements are integers.

i	0	1	2	3	4
Cumulative POS[i]	0	3	x	5	7

What value(s) could x be? b/c its cumulative POS[1] ≤ pos[2] ≤ pos[3] [5 pts]

$3 \leq x \leq 5 = 3, 4, 5$

List all the possible values for x	3, 4, 5

4. Suppose in the cumulative POS we have POS[i]==POS[i+1] for some i. Could the list contain both i and i+1? Explain briefly. [9 pts]

Solution:

No this is impossible because in the cumulative POS if there is no change between POS[i] and POS[i+1] that means that there was no change of total numbers counted between those two, so that means that no i+1 was observed in the list because otherwise, POS[i+1] would have incremented and changed therefore POS[i] wouldn't be equal to POS[i+1].

5. Suppose we run Radix Sort on the following list where the digit x is unknown and the process runs as follows: [6 pts]

Start	9x2	x62	7x6	152	2x4
After Iteration 1	9x2	x62	152	2x4	7x6
After Iteration 2	9x2	2x4	7x6	152	x62
After Iteration 3	152	2x4	x62	7x6	9x2

What are the possible digit values for x?

Possible Values for x	2, 3, 4

Scratch work; scratch work is not graded:

According to Iteration 2, x is < 5

According to Iteration 3, also $2 \leq x \leq 7$

so $2 \leq x < 5$

6. Your friend claims they ran Radix Sort on the following list where the digit x is unknown and the first two iterations ran as follows: [4 pts]

Start	863	911	341	1x8	42x	
After Iteration 1	911	341	42x	863	1x8	← x = 1 or 2
After Iteration 2	1x8	911	42x	341	863	

Why is this not possible?

Solution:

Well after the first iteration of "radix sort" here we know that x must be equal to 1 or 2 because it sorts correctly as a 1 and 2 when sorting the 1s place and comes out before the 863 after the first iteration (if x was >= 3, 863 would go before 42x because it appears earlier in the start array)
However, in the second iteration 1x8 comes out first, which at first glance could make sense if x = 1
(in fact, this sorting pass forces x = 1)
but 911 is the first entry in the input list for Iteration2, so even if x=1, 911 should be picked up THEN 1x8 after,
So the sort would be unstable and radix sort is supposed to be a stable sort.

7. Suppose Radix Sort is used to sort the following list of strings. Show the state of the list after each iteration of the underlying sort. [10 pts]

Start	DIG	DIE	BID	DAD	BAD
After Iteration 1	BID	DAD	BAD	DIE	DIG
After Iteration 2	DAD	BAD	BID	DIE	DIG
After Iteration 3	BAD	BID	DAD	DIE	DIG

8. Suppose $b = 2$ is a fixed base and j is a fixed positive integer. Suppose that our lists of length n contain integers between 0 and n^j inclusive. What is the time complexity of using Radix Sort with underlying Counting Sort? Show work. [10 pts]

Solution:

Radix sort runs in (d(n + (b - 1))) with d being the amount of stable sorts we are doing on the list.
Each underlying Counting Sort pass is (n+(b-1)) = (n)
Total = (d*n)

so,
we need to be able to represent integers between 0 and n^j.
Our base is 2, so we have 2 total positive integers available (~0 and 1 inclusive)

in binary to represent a number according to the notes the range is 0 and 2^n-1
- in this example, d=n
so: 2^d - 1 >= x
so d > lg(x)
d = floor(lg(n^j)) + 1
d = j * floor(lg(n)) + 1 (find time complexity now)
(d) = j * floor(lg(n))   (drop + 1)
(d) = floor(lg(n))   (j is a constant)
(d) = (lg(n))   (floor is insignificant)
(d) = lg(n)
so, (d * n) = (lg(n) * n)
= (n*lgn)

(Also, if I'm not mistaken it says that Radix Sort + Counting Sort time complexity would be (nlgn) with base 2 and list of length n in the radixsort.pdf in section 4.2)

Usually for n digit sbm, it is $n \cdot n$ so $\theta(n^2)$
but here, we have K, so sub K

9. Suppose A is an n-digit number and B is a k-digit number where k is a fixed constant. What is the Θ asymptotic time complexity of calculating AB via schoolbook multiplication? Explain. [5 pts]

Solution: $\theta(n \cdot K) = \theta(\cancel{K}n) = \underline{\theta(n)}$

10. We are using Karatsuba's algorithm to calculate (2539)(12046). In what follows, SDM means single-digit multiplication.

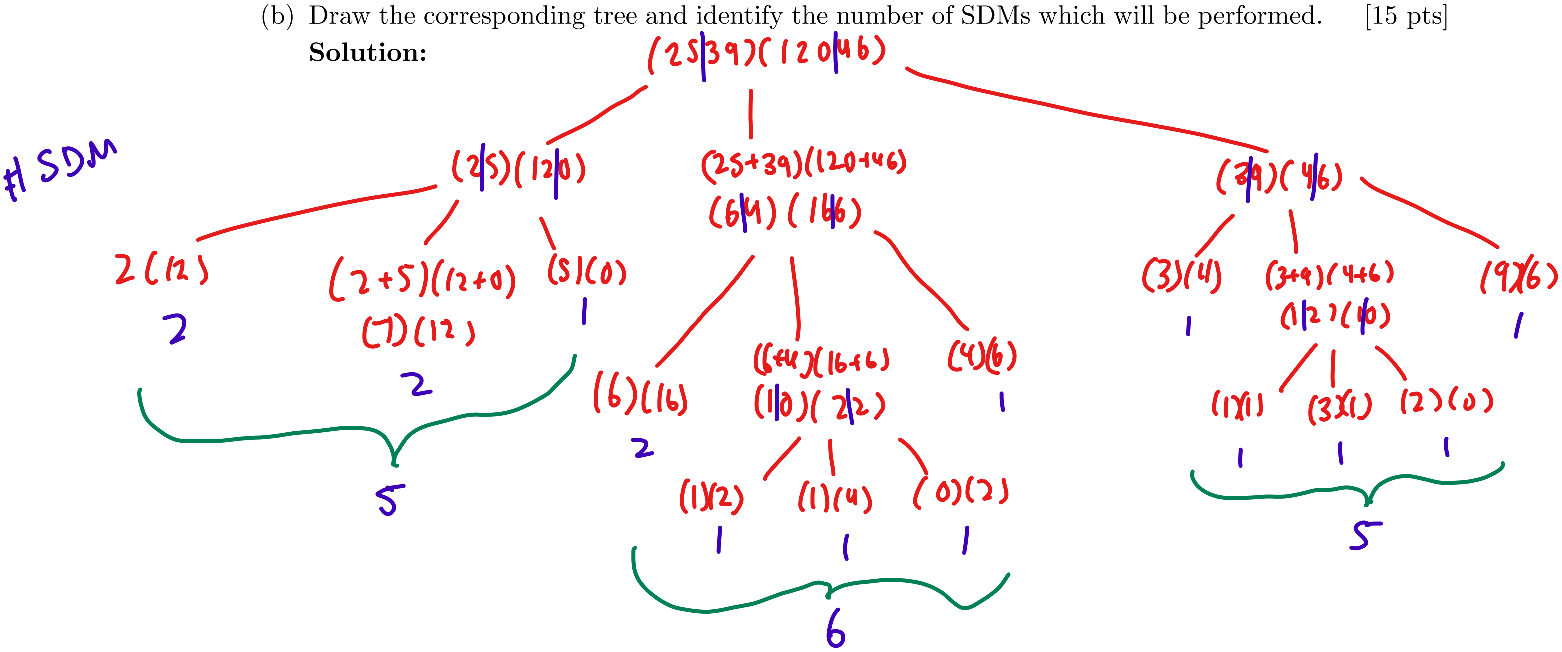
(a) Using schoolbook multiplication how many SDMs will be performed? [5 pts]

4 dig. x 5 dig = 20

Schoolbook SDM count =	20

(b) Draw the corresponding tree and identify the number of SDMs which will be performed. [15 pts]

Solution:



$5 + 6 + 5 = \boxed{16}$

by Piazza note: $T(n) = m\,T(\frac{n}{k}) + \Theta(n)$

$T_{Karat.}(n) = 3T(\frac{n}{2}) + \Theta(n)$

11. Suppose we have two n-digit numbers A and B. Suppose an integer k divides n and suppose we break each of A and B into k blocks of n/k digits each as follows:

$$A = A_{k-1}...A_3A_2A_1A_0$$
$$B = B_{k-1}...B_3B_2B_1B_0$$

In the above, each A_i and each B_j is a block of n/k digits.

If we used some non-Karatsuba-like method we would need k^2 multiplications (each A_i times each B_j). Suppose then we reduced the number of multiplications to m by some Karatsuba-like approach.

the integers m that the time complexity would be slower than Karatsuba are the ones where m < k^(1.585)

(a) For which integers m would the Θ time complexity be faster than Karatsuba? Explain. [5 pts]

Solution:

$T_{Karat.}(n) = 3T(\frac{n}{2}) + \Theta(n)$

by MT + notes

$= \Theta(n^{\log_2 3})$

$= \Theta(n^{\lg 3})$

$\approx \Theta(n^{1.585})$

need $\Theta(n^{\log_k m}) < \Theta(n^{\log_2 3})$

$= \log_k m < \log_2 3$

< 1.585

$m < k^{1.585}$

so for example:

(answers given in (k, m) tuple format)

ex:

(k, m)

(2, 1 or 2)

(3, 1-5)

$1 < 2^{1.585} \approx 3$

$2 < \approx 3$

$? < 3^{1.585} \approx 5.7$

$? = \{1, 2, 3, 4, 5\}$

(b) For which integers m would the Θ time complexity be slower than Karatsuba? Explain. [5 pts]

Solution:

need $\log_k m > \log_2 3$

need $= m > k^{1.585}$

ex:

k	$k^{1.585}$	m =
2	≈ 3	$m > 3$ $\{4, 5, 6 \ldots\}$
3	≈ 5.7	$m > 5.7$
4	≈ 9	$m > 9$ $\{10, 11, 12, 13, \ldots\}$

$m > 3$

$m > 5.7$ $\{6, 7, 8, 9 \ldots\}$

the integers m that the time complexity would be slower than Karatsuba are the ones where m > k^(1.585)

(c) For which integers m would the Θ time complexity be the same as Karatsuba? Explain. [5 pts]

Solution:

need $\log_k m = \log_2 3$

$\hookrightarrow m = k^{\log_2 3}$

use log simplification

ex:

$2^{\log_2 3} = \underline{3}$

b/c log + exp. are inverse

ex:

k	$k^{\log_2 3}$	m
2	3	3
3	≈ 5.7	X
4	9	9
5	≈ 12.81	X
6	≈ 17.11	x
8	27	27
16	81	81

$4^{\frac{\log(3)}{\log(2)}} = 9$

even #'s? X

All the k's that work with m = k^(lg3) are powers of 2, specifically for all k=2^n && k >=2, m = k^(lg3).
Or basically integers m come up as a solution when k is a power of 2