CMSC 351 Fall 2023 Homework 5

Due Wednesday Oct 11, 2023 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.
- 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, do not manually tag.

---

1. For each of the following recurrence relations fill in the corresponding a, b, and $f(n)$ according to the Master Theorem and then fill in the resulting Θ time complexity. If no conclusion can be drawn write NA in the last column. [40 pts]

   Note: For the last one it will be helpful to graph $f(n)$ to understand its nature better.

Relation	a	b	$f(n)$	Θ of?
(a) $T(n) = 2T(n/2) + 3n \lg n$	2	2	$3n \lg n$	$n \lg^2 n$
(b) $T(n) = 2T(n/4) + \sqrt[3]{n} + 1$	2	4	$\sqrt[3]{n} + 1$	$\sqrt{n}$
(c) $T(n) = 30T(n/5) + n^3 + \lg n$	30	5	$n^3 + \lg n$	n^3
(d) $T(n) = 17T(n/4) + n \lg n$	17	4	$n \lg n$	$n^{\log_4 17}$
(e) $T(n) = 9T(n/3) + 5n^2(1 + \sin n)$	9	3	$5n^2(1 + \sin n)$	NA

Put scratch work below. Scratch work is not graded but note that you should know how to explain your answer because you may be expected to do this on an exam.

(a) Case 2f) $a = 2, b = 2 \implies \log_b a = 1, c = 1, k = 1 \implies T(n) = \theta(n \lg^2 n)$

(b) Case 1) $a = 2, b = 4 \implies \log_b a = \frac{1}{2}, c = \frac{1}{3} \implies \theta(\sqrt{n})$

(c) Case 3) $a = 30, b = 5 \implies \log_b a \approx 2.11, c = 3 \implies \theta(n^3)$

(d) Case 1) $\log_b a \approx 2.04$. Note that $f(n) = O(n^2)$ and $\log_b a > c$ so $\theta(n^{\log_4 17})$

(e) Plotting the graph, we can't say anything about $T(n)$ except it is $\Omega(n^0)$ but $\log_b a > 0 \; \forall \, n$, so NA

2. Consider the recurrence relation which you may assume only has positive values: [10 pts]

$$T(n) = 8T(n/2) - T(n/10) + n^3 \lg^4 n$$

The Master Theorem cannot be used to find the Θ for $T(n)$ but we can be sneaky and use it to find the "best" $\mathcal{O}$ for $T(n)$. Be sneaky and do so.

Note: By "best" we mean the smallest possible function from our standard list.

Solution:

While we cannot find θ for $T(n)$, we can consider the expression

$$8T(n/2) + n^3 \lg^4 n$$

which we will denote $G(n)$. In this case, $n^3 \lg^4 n = \theta(n^3 \lg^4 n) \implies c = 3, k = 4$ and $a = 8, b = 2 \implies \log_b a = 3$. Note that $\log_b a = c$. Therefore, we have satisfied the conditions for case iii. of the Master Theorem with respect to $G(n)$. Therefore

$$G(n) = \theta(n^{\log_b a} \lg^{k+1} n) = \theta(n^3 \lg^5 n)$$

Recall that $\theta \implies O, \Omega$, and so we now also know

$$G(n) = O(n^3 \lg^5 n)$$

Observe that since the recurrence relation only has positive values, we know $T(n/10) > 0 \; \forall \; n$, and so it is also true that if $G(n)$ is bounded above by some function, then $T(n) = G(n) -$ some positive number is also bounded above by the same function. Therefore, we conclude

$$T(n) = O(n^3 \lg^5 n)$$

3. We are sorting the following list using MergeSort: [8 pts]

32, 16, 48, 24, 10, 50, 30, 20

At some point during merge sort, the array is:

16, 24, 48, 32, 10, 20, 30, 50

As merge sort continues to execute, **next time there is an ACTUAL CHANGE in the array**, what will the array look like?

Index	0	1	2	3	4	5	6	7
Original array	32	16	48	24	50	10	40	30
At some point	16	24	32	48	50	10	40	30
Next time there is a change	16	24	32	48	10	50	40	30

Put scratch work below. Scratch work is not graded but note that you should know how to explain your answer because you may be expected to do this on an exam.

4. Here is the pseudo-code for Mergesort and merge with print statements added

```
mergeSort( int [] arr, int start, int end )
    if sublist has at least 2 elements
        mergeSort Left half
        mergeSort Right half
        print "HI"; // this line was added
        merge the two sorted halves
    end if
end

merge(A, start1 ,end1 , start2 ,end2 ) // start2 = end1 + 1
    temparray = array of same size as arr
    i = start1 , j = start2 , k = start1
    while i <= end1 and j <= end2
        slot arr[i] OR arr[j] in kth correct slot in temparray
        increment i , j , k as appropriate
    end while
    while i <= end1
        flush left subarray of arr into temparray
    end while
    while j <= end2
        flush right subarray of arr into temparray
    end while
    // copy temparray into arr (NO print statement here)
    for i = start1 to i = end2
        print "HELLO"; // this line was added
        arr[i] = temp[i]
    end for
end
```

(a) When we run the code above on an array of 4 elements, how many times will HI and HELLO be output? [4 pts]

Number of times `HI` will be output	3
Number of times `HELLO` will be output	8

Put scratch work below. Scratch work is not graded but note that you should know how to explain your answer because you may be expected to do this on an exam.

(b) When we run the code above on an array of 8 elements, how many times will HI and HELLO be output? [4 pts]

Number of times `HI` will be output	7
Number of times `HELLO` will be output	24

Put scratch work below. Scratch work is not graded but note that you should know how to explain your answer because you may be expected to do this on an exam.

(c) When we run the code above on an array of $n = 2^k$ elements, how many times will HI and HELLO be output? Give your answer in terms of k. [4 pts]

Number of times `HI` will be output	$2^k - 1$
Number of times `HELLO` will be output	$k * 2^k$

Explain your two answers in the table above (only for an array of 2^k elements). [10 pts]

Explain:

For "HI", it will be printed once on the list of 2^k elements, twice for the 2 lists of 2^{k-1} elements, 4 times for the 4 lists of 2^{k-2} elements ... until the exponent is 0 in which the lists are of length 1. Thus, we obtain the summation

$$\sum_{i=1}^{k} 2^{k-i} = 2^k - 1$$

For "HELLO", note that it is printed once for a list of length 2^k, twice for the 2 lists of length 2^{k-1} up until we get lists of length 1 and so we get the summation

$$\sum_{i=1}^{k} 2^k = k * 2^k$$

5. Assume that any operation/simple statement (comparison, addition, subtraction, multiplication, division, return statement) takes EXACTLY 1 unit of time.

(a) Consider the following pseudocode: [5 pts]

```
funtion fooOne(n)
    if n < 3:
        return 0;
    else
        return 2 * foo1(n - 1) + 3;
    end if
end function
```

Write down a recurrence relation for $T(n)$, the running time of `fooOne`. **DO NOT SOLVE**.

Recurrence Relation for `fooOne`	$T(n) =$ $T(n-1)+4, T(2)=2$

(b) The following is a recurrence relation for the running time of a method: [5 pts]

$$T(n) = 2T(n/3) + 3$$

Suppose we code the following function (don't worry about what it does):

```
function fooTwo(n)
    if( n < 2 )
        return 0;
    else
        return EXPRESSION
    end if
end function
```

What could the `EXPRESSION` be so the function satisfies the recurrence relation?

`EXPRESSION`	fooTwo(n/3) * fooTwo(n/3)

6. Consider a **1-indexed** array of many (as many as we need) elements implementing a heap. [10 pts]
Consider the heap node nd with index $2^k - 1$ for some k.

We denote $GPS(nd)$ the sibling of the grandparent of nd.

Answer each of the following. Do not use the floor operator in your answers and simplify your answers as much as possible.

Solution:

Number of nodes at the same level as nd in the heap?	2^{k-1}
What is the index of $GPS(nd)$ as a function of k?	$2^{k-2} - 2$

Put scratch work below. Scratch work is not graded but note that you should know how to explain your answer because you may be expected to do this on an exam.