CMSC 351 Summer 2025 Homework 4

Due Tuesday 29 July 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. Consider the recurrence relation:

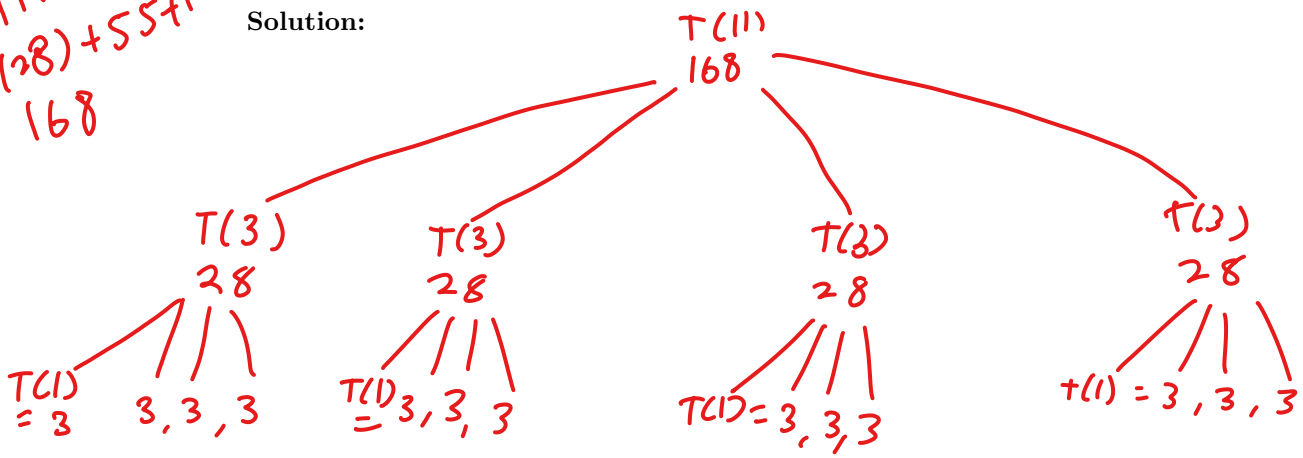
$$T(n) = 4\,T(\lfloor n/3 \rfloor) + 5n + 1 \quad \text{with} \quad T(0) = 2, T(1) = 3$$

(a) Calculate each of the following and simplify.

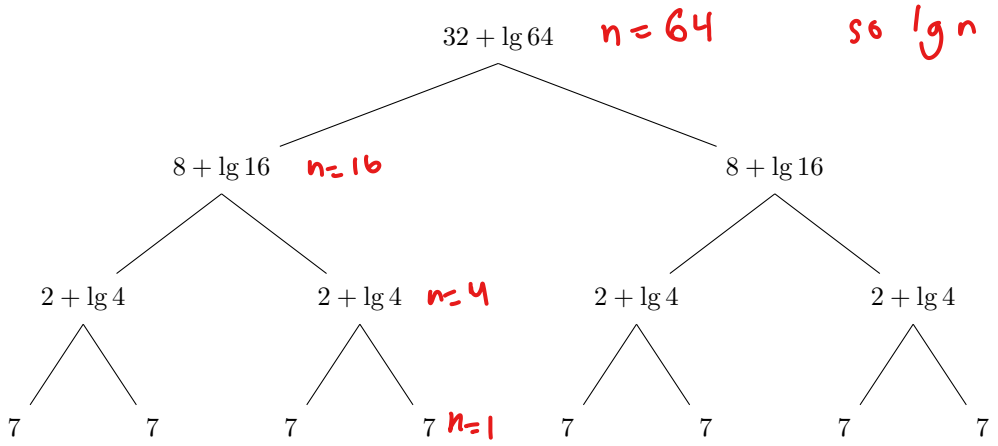
$T(2) =$	19	[2 pts]
$T(3) =$	28	[2 pts]
$T(11) =$	168	[3 pts]

(b) Draw the tree corresponding to $T(11)$. Simplify all node values. [5 pts]

Solution:



2. Here is a tree corresponding to $T(64)$ for an unknown recurrence relation: [8 pts]



Fill in the details for the corresponding recurrence relation:

$$T(n) = \boxed{2}\ T(n/4) + \boxed{\frac{n}{2} + \lg(n)}$$

$$T(1) = \boxed{7}$$

Put scratch work below; Scratch work is not graded but may be used for regrade partial credit:

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

# of leaves in subtree; size of subproblem.

3. For each of the following descriptions of an algorithm write down the corresponding recurrence relation.

   (a) An algorithm which processes a list of length n by recursively processing three lists of half the length and then exactly an additional asymptotically linear amount of work on the original list. [4 pts]

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

   (b) An algorithm which processes a list of length n by recursively processing one list of half the length and then at most an additional asymptotic logarithmic amount of work on the original list. [4 pts]

   $T(n) =$ $(1) \cdot T\left(\frac{n}{2}\right) + \lg n - 1$

   (c) An algorithm which processes a list of length n by recursively processing one list of length one less and then at least an additional asymptotic quadratic amount of work on the original list. [4 pts]

   $T(n) =$ $(1) \cdot T(n-1) + n^4$

   (d) An algorithm which processes a list of length n by recursively processing four lists of length three less and then at most an additional asymptotic constant amount of work on the original list. [4 pts]

   $T(n) =$ $4 \cdot T(n-3) + 1$

$T(n) = 2^n T(\frac{n}{2}) + n^n$

$e^{n \ln(x)} = f(x)$

$e^{n \ln 2} T(\frac{n}{2}) + e^{n \ln n}$ $\Rightarrow f(2) T(\frac{n}{2}) + f(n)$

$\Omega(e^n)$

4. For each of the following recurrence relations fill in the Θ time complexity given by the Master Theorem. If the Master Theorem does not apply write NA. [20 pts]

Relation	Θ of?
(a) $T(n) = 3T(n/2) + n^2$	$\Theta(n^2)$
(b) $T(n) = 4T(n/2) + n^2$	$\Theta(n^2 \lg 2)$
(c) $T(n) = 2T(n/4) + n^{0.51}$	$\Theta(n^{0.51})$
(d) $T(n) = 2^n T(n/2) + n^n$	N/A
(e) $T(n) = 16T(n/4) + n$	$\Theta(n^2)$

c2

c3

$\log_2 3 = 1.\sim$ f c=2

c3?

$f(n) = n^2 = \Omega(n^2)$ ✓

$\log_2 3 < 2$ ✓

Put scratch work below; Scratch work is not graded but may be used for regrade partial credit.

b) $T(n) = 4T(\frac{n}{2}) + n^2$ obs. $\log_2 4 = 2$

$T(n) = \Theta(n^{\log_2 4} \lg 2)$

$= \Theta(n^2 \lg 2)$

c) $f(n) = n^{0.51} = \Omega(n^{0.5})$

$\log_b a = \log_4 2 = \frac{1}{2} : 0.5 < 0.51$

so $T(n) = \Theta(n^{0.51})$

e) $T(n) = 16T(\frac{n}{4}) + n$ $\quad f(n) = n$

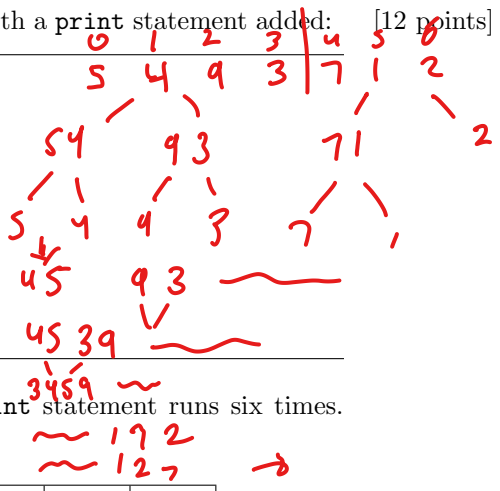
$\log_4 16 = 2 > 1$ $\quad c = 1$

so c1: $T(n) = \Theta(n^{\log_4 16})$

$= \Theta(n^2)$

5. Here is the pseudocode for the `MergeSort` part of Merge Sort with a `print` statement added: [12 points]

```
function MergeSort(arr,start,end)
    if start < end:
        middle = (start + end ) // 2
        MergeSort(arr,start,middle)
        MergeSort(arr,middle+1,end)
        Merge(arr,start,middle,middle+1,end)
        print(arr)
    end if
end function
```



Suppose we call `MergeSort([5,4,9,3,7,1,2],0,6)`. The `print` statement runs six times. What is printed after each time?

First Time	4	5	9	3	7	1	2
Second Time	4	5	3	9	7	1	2
Third Time	3	4	5	9	7	1	2
Fourth Time	3	4	5	9	1	7	2
Fifth Time	3	4	5	9	1	2	7
Sixth Time	1	2	3	4	5	7	9

6. Suppose $n \geq 1$ and the list A consists of the integers 1 through 2^n in reverse order. We call:

`MergeSort(A,0,len(A)-1)`

using the pseudocode from Question 5. How many `print` statements will occur (but not including)...

$(n-1 \text{ merges}) - 1 = n - 2 = 2^n - 2$

...the code ends?	$2^n - 2$	[4 pts]
...the first half of the list changes?	$2^{n-1} - 2$	[2 pts]
...the second half of the list changes?	$2^{n-1} - 3$	[4 pts]
...the fourth quarter of the list changes?	$2^n - 4$	[6 pts]

$2(T(\frac{n}{4}) - 1)$
$= 2(\frac{n}{4}) - 2$
$= \frac{n}{2} - 2$
$\frac{2^n}{2} - 2$
$= 2^{n-1} - 2$

- first half,
- third q
- sub q $\rightarrow (\frac{n}{2} - 1) + (\frac{n}{4} - 1)$
$= (2^{n-1} - 1) + (2^{n-2} - 1) + 2(\frac{n}{8} - 1)$

$(\frac{n}{2} - 1) + 2(\frac{n}{4} - 1) = 2(2^{n-2} - 1) = 2^{n-1} - 2$

$\frac{2^n}{2} - 1 = (2^{n-1} - 1) + 2(\frac{2^n}{2^2} - 1)$ $\quad (2^{n-1} - 1) + (2^{n-1} - 2)$
$= 2^{n-1} - 3$

$2\frac{2^k}{2^3} = 2^{n-3} - 1 + 1 = 2^{n-2} + (2^{n-1} - 1) + (2^{n-2} - 1)(2^{n-2} - 2)$

$T(n) = 2n \cdot T(\frac{n}{2}) + \Theta(n)$

7. In Merge Sort we say a *merge* occurs when two lists are merged back together. Suppose that when merging two lists the number of seconds it takes equals twice the length of the longest list. Suppose we have a list of length n.

(a) If n is a power of 2 how long will the merging process take? Give your answer in terms of n. [3 pts]

Merge Time When n is a Power of 2	$\Theta(n \lg n)$

$f(n) = \Theta(n^1)$
$\log_2 2 = 1$ ✓

(b) Explain your answer to (a). [5 pts]

Solution:

$T(n) = \Theta(n \log n)$ but when n = power of 2,

it does $\log_2 n$ iterations of merge() instead of $\log n$

notes say that each level costs $\Theta(n)$ and there are $\lg n$ levels when $n = 2^k$

$\Theta(n \log_2 n) = \Theta(n \lg n)$

so $f(n) = \Theta(n) = \Theta(n^1)$

$= \log_2 2 = 1$ ✓ $T(n) = \Theta(n^{\log_2 2} \cdot \lg n)$

$= \Theta(n^1 \lg n)$

$= \Theta(n \lg n)$

(c) For any n (not necessarily a power of 2) we know that $n \leq 2^{\lceil \lg n \rceil}$. Use this to find an upper bound on the time it will take for the merging process. Give your answers in terms of n. [3 pts]

Merge Time Upper Bound	$2^{\lceil \lg n \rceil} \cdot \lg(2^{\lceil \lg n \rceil})$

(d) Explain your answer to (c). [5 pts]

Solution:

case 2) $n \in 2^{\lceil \lg n \rceil}$

$f(n) = \Theta(2^{\lceil \lg n \rceil})$

for merge, split list into 2 sublists.

Sublist size will be $n/2$ each, so $T(n) = 2T(\frac{n}{2}) + n$

$n = f(n) = \Theta(n^1)$ and $\log_2 2 = 1$ so $T(n) = \Theta(n^{\log_b a} \lg n)$

$= 2^{\lceil \lg n \rceil} \cdot \lg(2^{\lceil \lg n \rceil})$

$a = 2$
$b = 2$
$\log_b a = 1$ $n^{\log_b a} = n$
$f(n) = \Theta(n^{\log_b a})$
$= \Theta(n)$