CSCI 3104 Fall 2022
Instructors: Prof. Grochow and Nagesh

# Problem Set 7

Due Date ..... October 24, 2022
Name ..... **Tyler Huynh**
Student ID ..... **109603994**
Collaborators ..... **List Your Collaborators Here**

## Contents



## Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Useful links and references on LaTeXcan be found here on Canvas.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.

- You **must** virtually sign the Honor Code (see Section Honor Code). Failure to do so will result in your assignment not being graded.

## Honor Code (Make Sure to Virtually Sign the Honor Pledge)

**Problem HC.** On my honor, my submission reflects the following:

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

In the specified region below, clearly indicate that you have upheld the Honor Code. Then type your name.

*Honor Pledge.* I, **Tyler Huynh** on my honor pledge that my submission is a reflection of my own understanding of the material, any and all collaborations/sources have been properly cited, I have not posted any material to external sources, and I have not copied other solutions as my own. □

# 19 Standard 19 - Solving Recurrences: Tree Method

**Problem 19.** Consider the recurrence $T(n)$ below. Using the **tree method**, determine a suitable function $f(n)$ such that $T(n) \in \Theta(f(n))$. Clearly show all steps. Note the following:

- You may assume, without loss of generality, that $n$ is a power of 3 (i.e., $n = 3^k$ for some integer $k \geq 0$).
- You may hand-draw your tree and embed it, provided it is legible and we do not have to rotate our screens to read it. However, **all your calculations must be typed**.

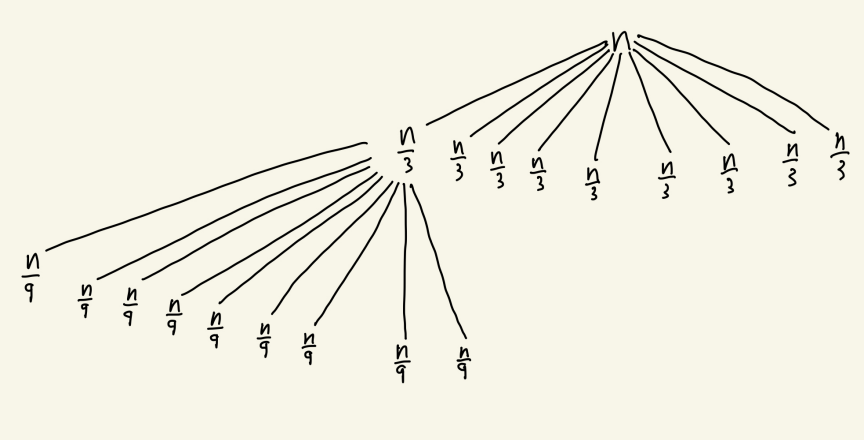
$$T(n) = \begin{cases} 1, & n < 3 \\ 9T(n/3) + n^2, & \text{otherwise.} \end{cases}$$

*Answer.* I will first begin by finding how many times we reach the base case of k:

$$\begin{aligned} \frac{n}{3^k} &< 3 \\ \log_3 n &< \log_3(3^k, 3) \\ \log_3 n &< k + \log_3 3 \\ k &> \log_3 n - \log_3 3 \\ k &> \log_3 n - 1 \end{aligned}$$

From the above we see that this will be the amount of time it will run, until we reach the base case of k.

I will now find the tree of T(n):




From the above for the children of $\frac{n}{3}$ will have 9 children, that will have a value $\frac{n}{9}$.

I will now find T(n):

$$T(n) = (\text{cost of base case}) \cdot (\text{number of times the base case is reached}) + \sum_{i=0}^{\log_3 n-1} 9^i * (\frac{n}{3^i})^2$$

$$= 9^k * 1 + \sum_{i=0}^{\log_3 n-1} 9^i * (\frac{n}{3^i})^2$$

$$= 9^k + \sum_{i=0}^{\log_3 n-1} 9^i * \frac{n^2}{3^{2i}}$$

$$= 9^k + \sum_{i=0}^{\log_3 n-1} 9^i * \frac{n^2}{9^i}$$

$$= 9^k + n^2 * (\log_3 n - 1)$$

$$= 9^k + n^2 \log_3 n - n^2$$

$$= 9^{\log_3 n-1} + n^2 \log_3 n - n^2$$

$$= \frac{9^{\log_3 n}}{9} + n^2 \log_3 n - n^2 \qquad \text{Log base change:} \qquad 9^{\log_3 n-1} = \frac{\log_9 n}{\log_9 3}$$

$$= \frac{9^{\frac{\log_9 n}{\log_9 3}}}{9} + n^2 \log_3 n - n^2$$

$$= \frac{9^{\frac{\log_9 n}{1/2}}}{9} + n^2 \log_3 n - n^2$$

$$= \frac{9^{2\log_9 n}}{9} + n^2 \log_3 n - n^2$$

$$= \frac{9^{\log_9 n^2}}{9} + n^2 \log_3 n - n^2$$

$$= \frac{n^2}{9} + n^2 \log_3 n - n^2$$

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

From the above we can see that the largest component would be $(n^2 \log n)$, such that our final runtime complexity of T(n) will be:

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

□

# 20 Standard 20 - QuickSort

## 20(a). Part (a)

**Problem 20.** (a) Write down a recurrence relation that models the **best case** running time of Quicksort, i.e. the case where PARTITION selects the **median** element at each iteration. What is the best-case running time of QuickSort? Be sure to write **both the recurrence relation and the runtime.**

*Answer.* **Referenced Levet notes**
Our recurrence relation below:

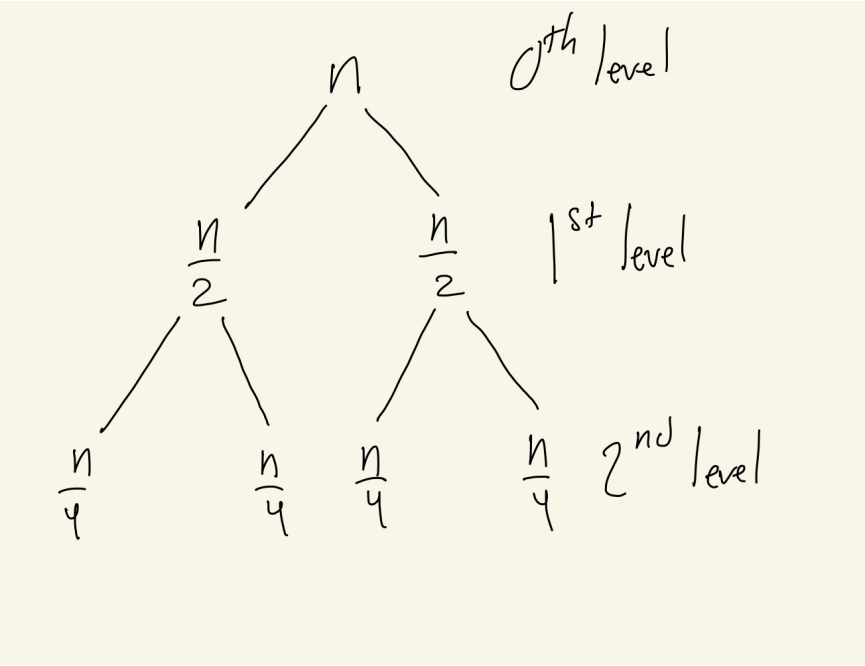
$$T(n) = \begin{cases} \Theta(1) & : n \leq 2, \\ 2T(\frac{n}{2}) + \Theta(n) & : n > 2. \end{cases}$$

The best-case running time of QuickSort is.... when the algorithm chooses the median in a list as the pivot, i.e. also splitting it in half. Below I will show how I come to this conclusion:

I will solve my recurrence relation to find the runtime of T(n) by using the tree method:
I will first begin by finding how many times until we reach the base case of k:

$$\begin{aligned} \frac{n}{2^k} &\leq 2 \\ \log_2 n &\leq \log_2(^k, 2) \\ \log_2 n &\leq k + \log_2 2 \\ k &\geq \log_2 n - \log_2 2 \\ k &\geq \log_2 n - 1 \end{aligned}$$

From the above we see that this will be the amount of times it will run, until we reach the base case of k.
I will now find the tree of T(n):



I will now solve for T(n):

$$
\begin{aligned}
T(n) &= (\text{cost of base case}) \cdot (\text{number of times the base case is reached}) + \sum_{i=0}^{\log_2 n-1} 2^i * \frac{n}{2^i} \\
&= \Theta(1) * 2^k + \sum_{i=0}^{\log_2 n-1} 2^i * \frac{n}{2^i} \\
&= \Theta(1) * 2^{\log_2 n-1} + \sum_{i=0}^{\log_2 n-1} n \\
&= \Theta(1) * 2^{\log_2 n-1} + n(\log_2 n - 1) \\
&= \Theta(1) * 2^{\log_2 n-1} + n \log_2 n - n \\
&= \Theta(1) * 2^{\log_2 n} * 2^{-}1 + n \log_2 n - n \\
&= \Theta(1) * n * \frac{1}{2} * n \log_2 n - n \\
&= \frac{\Theta(1)}{2} + n \log_2 n - n \\
T(n) &= \Theta(n \log n)
\end{aligned}
$$

From the above we can see that the best case running time of the algorithm Quicksort will be:

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

□

## 20(b). Part (b)

(b) Write down a recurrence relation that models the **worst case** running time of Quicksort, i.e. the case where PARTITION selects the **last** element at each iteration. What is the worst-case running time of QuickSort? Be sure to write **both the recurrence relation and the runtime.**

*Answer.*

$$T(n) = \begin{cases} \Theta(1) & : n \leq 2, \\ T(n-1) + T(1) + \Theta(n) & : n > 2. \end{cases}$$

The worst-case running time of QuickSort is.... when the partition algorithm chooses the last element in the list at each iteration, below I will show how I came to this conclusion:

I will solve my recurrence relation to find the runtime of T(n) by using the unrolling method:
I will first begin by finding how many times until we reach the base case of k:

$$\begin{aligned} n - k &\leq 2 \\ n &\leq 2 + k \\ n - 2 &\leq k \end{aligned}$$

From the above this will be how many times it will run, until we reach the base case of k. I will now find how many times T(n) will run:
I will change $T(n)$ and $\Theta(n)$into $c_1$ and $c_2 n$

$$\begin{aligned} T(n) &= T(n-1) + c_1 + c_2 n \\ T(n-1) &= T(n-2) + c_1 + c_2(n-1) \\ T(n-2) &= T(n-3) + c_1 + c_2(n-2) \end{aligned}$$

I will now find T(n):

$$\begin{aligned} T(n) &= (\text{cost of base case}) \cdot (\text{number of times the base case is reached}) + \sum_{i=0}^{n-2} T(1) + c_2(n-i) \\ &= \Theta(1) * 2^k + \sum_{i=0}^{n-2} c_1 + c_2(n-i) \\ &= \Theta(1) * 2^{n-2} + \sum_{i=0}^{n-2} c_1 + c_2(\sum_{i=0}^{n-2} n - \sum_{i=0}^{n-2} i) \\ &= \Theta(1) * \frac{2^n}{4} + c_1(n-2) + c_2[n(n-2) - (\frac{(n-2)(n-2+1)}{2})] \\ &= \Theta(1) * \frac{2^n}{4} + c_1(n-2) + c_2[n^2 - 2n - (\frac{(n-2)(n-1)}{2})] \\ &= \Theta(1) * \frac{2^n}{4} + c_1(n-2) + c_2[n^2 - 2n - (\frac{n^2 - 3n + 2}{2})] \\ &= \Theta(1) * \frac{2^n}{4} + c_1(n-2) + c_2[n^2 - 2n - (\frac{n^2 - 3n}{2} + 1)] \\ T(n) &= \Theta(n^2) \end{aligned}$$

From the above we can see that the highest component is $n^2$, such that our final answer for the worst case runtime scenario of T(n):

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

□

## 20(c). Part (c) (Also credit towards S17 or S19)

(c) Suppose that we modify PARTITION so that it chooses the median element as the pivot in calls that occur in nodes of the recursion tree of a call to QUICKSORT whose depth in the recursion tree is divisible by 3, and it chooses the maximum element as the pivot in calls that occur in nodes **all** other depth of this recursion tree.

Assume that the running time of this modified PARTITION is still $\Theta(n)$ on any subarray of length $n$. You may assume that the root of a recursion tree starts at level 0 (which is divisible by 3), its children are at level 1, etc. For example, the modified PARTITION chooses the median element at the root of the recursion tree, in the next two layers of the recursion tree it chooses the max, and in level 3 of the recursion tree it chooses the median again, and so on.

**Your job** is to write down a recurrence relation for the running time of this version of QUICKSORT given an array $n$ distinct elements and solve it asymptotically, i.e. give your answer as $\Theta(f(n))$ for some function $f(n)$. Show your work.

(If you solve your recurrence using unrolling, you can get credit towards S17. If you solve your recurrence using the tree method you can get credit towards S19. In either case, this problem also counts towards credit for S20.)

*Answer.* □