CSCI 3104 Spring 2022
Instructor: Profs. Chen and Layer

# Problem Set 6

Due Date ..... March 8
Name ..... **Chengming Li**
Student ID ..... **109251991**
Collaborators ..... **N/A**

## Contents



## Instructions

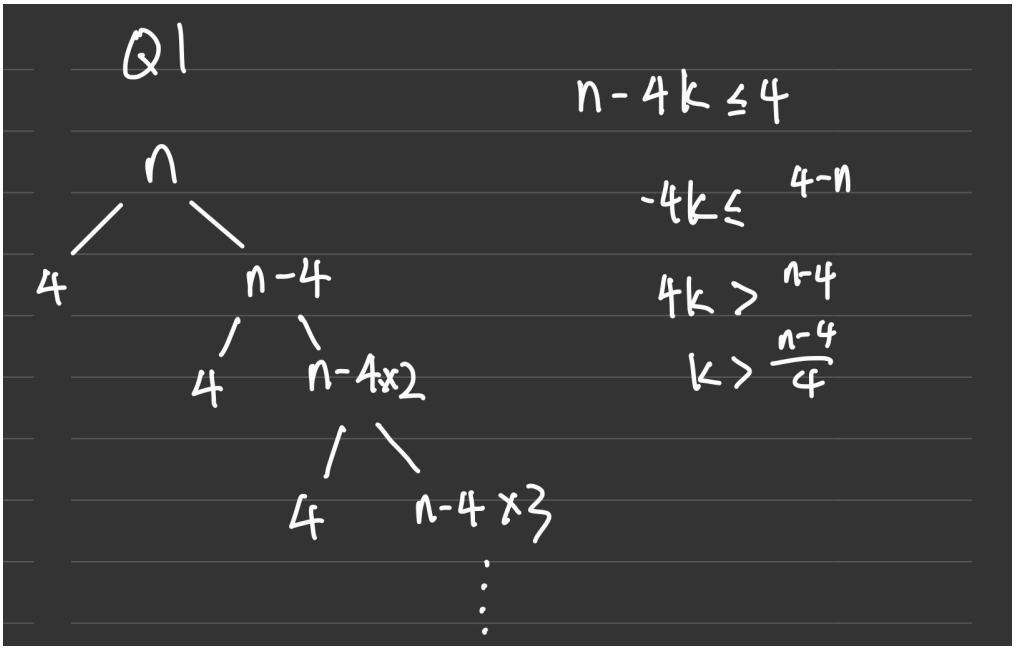
- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Useful links and references on LaTeXcan be found here on Canvas.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX 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.

# 1 Standard 17: Balanced versus unbalanced partitioning.

## 1.1 Problem 1

**Problem 1.** (a) Consider a modified Merge-Sort algorithm that at each recursion splits an array of size $n$ into two subarrays of sizes 4 and $n-4$, respectively. Write down a recurrence relation for this modified Merge-Sort algorithm and give its asymptotic solution.

*Answer.*




$$T(n) = \begin{cases} \Theta(1) & : n \leq 4. \\ T(n-4) + T(4) + \Theta(n) & : n > 4 \end{cases}$$

**Finding the depth of tree first**

$$\begin{aligned} n - 4k &\leq 4 \\ -4k &\leq 4 - n \\ 4k &> n - 4 \\ k &> \lceil \frac{n-4}{4} \rceil \\ k &\approx \frac{n-4}{4} \end{aligned}$$

**Total run time complexity:**

$$\frac{n-4}{4} \cdot n = \frac{n^2 - 4n}{4}$$

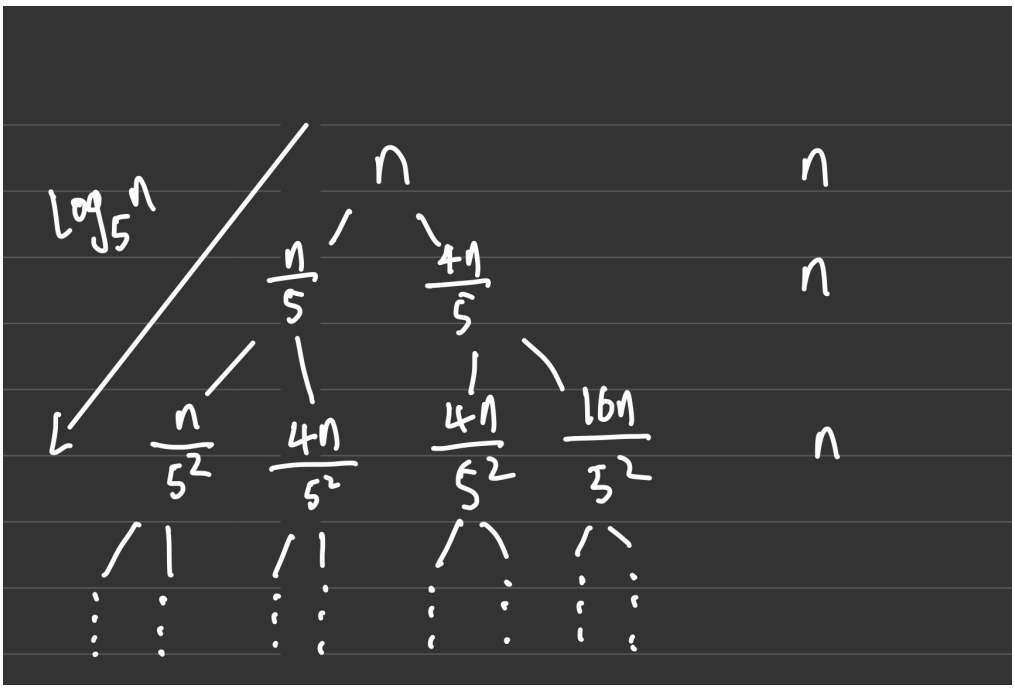
**Asymptotic Solution:**

$$\begin{aligned} \lim_{n\to\infty} \frac{\frac{n^2-4n}{4}}{n^2} &= \lim_{n\to\infty} \frac{2n-4}{8n} \\ &= 1/4 \end{aligned}$$

**This modified Merge Sort Algorithm has tight bound as $\Theta(n^2)$** □

(b) Consider a modified Merge-Sort algorithm that at each recursion splits an array of size $n$ into two subarrays of sizes $\frac{1}{5}n$ and $\frac{4}{5}n$, respectively. Write down a recurrence relation for this modified Merge-Sort algorithm and give its asymptotic solution.

*Answer.*




$$T(n) = \begin{cases} \Theta(1) & : n \leq 5. \\ T(\frac{n}{5}) + T(\frac{4n}{5}) + \Theta(n) & : n > 5 \end{cases}$$

**Finding the depth of tree first**

$$\begin{aligned} \frac{n}{5^k} &\leq 5 \\ \frac{n}{5} &\leq 5^k \\ log_5 \frac{n}{5} &\leq k \\ k &\geq log_5 \frac{n}{5} \\ k &\approx log_5 \frac{n}{5} \end{aligned}$$

**Total run time complexity:**

$$log_5 \frac{n}{5} \cdot n = nlog_5 \frac{n}{5}$$

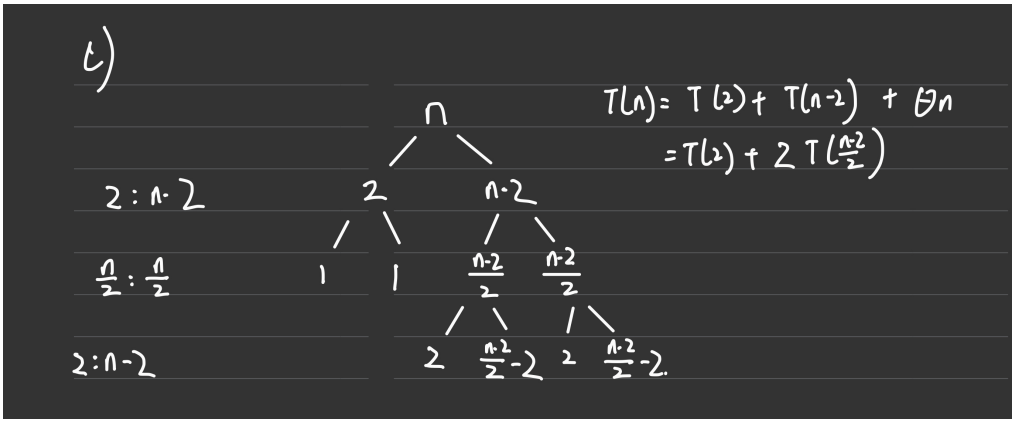
**Asymptotic Solution:**

$$\begin{aligned} \lim_{n\to\infty} \frac{nlog_5\left(\frac{n}{5}\right)}{nlog_5\left(n\right)} &= \lim_{n\to\infty} \frac{\frac{1}{ln(5)\left(\frac{n}{5}\right)}}{\frac{1}{ln(5)(n)}} \\ &= \lim_{n\to\infty} \frac{ln\left(5\right)n}{ln\left(5\right)\left(\frac{n}{5}\right)} \\ &= 5 \end{aligned}$$

**This modified Merge Sort Algorithm has tight bound as $\Theta(nlog_5\left(n\right))$** □

(c) Suppose that we modify the Merge-Sort algorithm in such a way that on alternating levels of the recursion, the partitioning is either a $(2, n-2)$ split or a $(n/2,\ n/2)$ split. Write down a recurrence relation for this modified Merge-Sort algorithm and give its asymptotic solution. Then, give a verbal explanation of how this Merge-Sort algorithm changes the running time of Merge-Sort.

*Answer.*




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

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

$$\begin{aligned} T(n) &= T(2) + T(n-2) + \Theta(n) \\ &= T(2) + 2T(\frac{n-2}{2}) + 2\Theta(n) \end{aligned}$$

**For finding the asymptotic solution, we could ignore the term T(2) and left with $2T(\frac{n-2}{2})$. Finding the depth of tree first**

$$\begin{aligned} \frac{n-2}{2^{0.5k}} &\leq 2 \\ n-2 &< 2^{0.5k+1} \\ log_2\,(n-2) &\leq (0.5k+1) \\ k &\geq 2(log_2\,(n-2)-1) \\ k &\approx 2(log_2\,(n-2)-1) \end{aligned}$$

**Total run time is:**

$$2(log_2\,(n-2)-1) \cdot n$$

**Asymptotic Solution is:**

$$2(log_2\,(n-2)-1) \cdot n \in \Theta(nlog_2(n))$$

**This algorithm doesn't change the run time of Merge Sort a lot since they have the same tight bound as $nlog_2(n)$. But it does increase the run time by factor of 2 due to an extra level with**

**(2,n-2) splitting in each original algorithm level if we focus on the details of run time of this modified merge sort algorithm.**
**In this algorithm, the size of array will reduce whenever its recurrence has $T_{(n/2,n/2)}(n)$ at the layer $(n/2, n/2)$, and then the depth of tree will go further down. But, when its recurrence is $T_{(2,n-2)}(n)$, the depth of tree will not go further down and it delays the run-time of whole algorithm. And the factor of 2, I mentioned above, comes from this recurrence.** □

# 2 Standard 18: Quicksort.

## 2.1 Problem 2

**Problem 2.** Given an input array {3, 7, 1, 8, 2, 6, 5, 4}. Consider the deterministic QuickSort algorithm and show the input array, the output array, and the global array at every partition as in the example in Section 2.1.1 of the course notes for week 8 (see Week 8 under "Modules" of the course canvas).

*Answer.*

| Procedure | arguments | input | output | return | global array |
|---|---|---|---|---|---|
| first | x=4 p=0 r=7 | A=[3, 7, 1, 8, 2, 6, 5, **4**] | $[3,1,2 \mid \mathbf{4} \mid 7,6,5,8]$ | r = 3 | $[3,1,2,4,7,6,5,8]$ |
| L QS | x=2 p=0,r=2 | A= $[3,1,\mathbf{2}]$ | $[1,\mathbf{2},3]$ | r = 1 | $[1,2,3,4,7,6,5,8]$ |
| R QS | x=8 p=4,r=7 | A= $[7,6,5,\mathbf{8}]$ | $[7,6,5,\mathbf{8}]$ | r = 7 | $[1,2,3,4,7,6,5,8]$ |
| L QS | x=5 p=4,r=6 | A=$[7,6,\mathbf{5}]$ | $[\mathbf{5},6,7]$ | r = 5 | $[1,2,3,4,5,6,7,8]$ |
| R QS | x=8 p=8,r=7(violated) | A = [8] | $N/A$ | r = NA | $[1,2,3,4,5,6,7,8]$ |

□

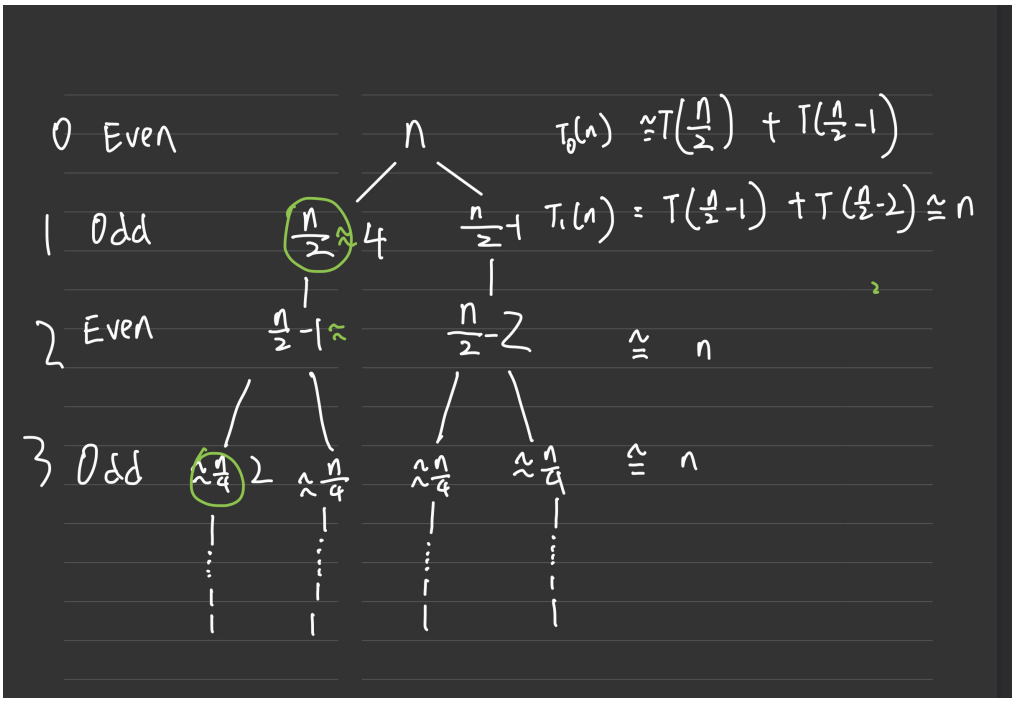
## 2.2 Problem 3

**Problem 3.** Suppose that we modify PARTITION$(A, s, e)$ so that it chooses the median element of $A[s..e]$ in calls that occur in nodes of even depth of the recursion tree of a call QUICKSORT$(A[1, \ldots, n], 1, n)$, and it chooses the minimum element of $A[s..e]$ in calls that occur in nodes of odd 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 an even number), its children are at level 1, etc.

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.

*Answer.*




**Above is my general idea about what's going on in this algorithm.**

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

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

$$\begin{aligned} T(n) &= T(\frac{n}{2}) + T(\frac{n}{2} - 1) + \Theta(n) \\ &= T(\frac{n}{2} - 1) + T(\frac{n}{2} - 2) + \Theta(n) \\ &= ..... \end{aligned}$$

**Finding the depth of tree first**

$$\begin{aligned} \frac{n}{2^{0.5k}} &\leq 1 \\ n &< 2^{0.5k} \\ log_2(n) &\leq (0.5k) \\ k &\geq 2log_2(n) \\ k &\approx 2log_2(n) \end{aligned}$$

**At each level, I approximate the expression (i.e. ignoring the constant part). As seen in level 1, there are two nodes. So the total run time complexity at this level is $2 \cdot \frac{n}{2} = n$**
**And as seen in level 2, 3..., Run time complexity at each level approximately equal to n.**
**Total run time complexity is:**

$$2log_2(n) \cdot n = 2nlog_2(n)$$

**Tight bound:**

$$\lim_{n \to \infty} \left( \frac{2nlog_2(n)}{nlog_2(n)} \right) = 2$$

**This algorithm has bound as $\Theta(nlog_2(n))$** □