CSCI 3104 Spring 2022
Instructor: Profs. Chen and Layer

# Problem Set 6

Due Date ..... March 8
Name ..... **Your Name**
Student ID ..... **Your Student ID**
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.

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

$$
\begin{aligned}
T(n) &= 4 + T(n-4) + \Theta(n) \\
T(n) &= T(n-4) + \Theta(n) \\
&= T(n-5) + \Theta(n-1) + \Theta(n) \\
&= T(n-6) + \Theta(n-2) + \Theta(n-1) + \Theta(n) \\
&\cdot \\
&\cdot \\
&\cdot \\
T(n) &= \Theta(1) + ... + \Theta(n-2) + \Theta(n-1) + \Theta(n) \\
T(n) &= \sum_{i=1}^{n} \Theta(i) \\
T(n) &= \Theta(n^2)
\end{aligned}
$$

□

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

Considering the extremes of the two paths we get that the depth of the tree will either be $\log_5(n)$ or $log_{\frac{5}{4}}(n)$. Using this knowledge and the formula that says the run time of the algorithm=cost at each level * total run time we get that.

$$\begin{aligned} T(n) &= \Theta(n) * \Theta(\log_5(n)) \\ &= \Theta(n \log_5(n) \end{aligned}$$

Therefore $T(n) \in \Theta(n \log n)$ □

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

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

$Levels - 2, n-2$

$(n-2)/2.(n-2)/2$

$(n-2)/2-2, (n-2)/2-2$

$((n-2)/2-2)/2, ((n-2)/2-2)/2$ Therefore $T(n) \in (n \log(n)$ □

# 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.* {3,7,1,8,2,6,5,4}
First partition: Input:{3,7,1,8,2,6,5,4} Output: {3,2,1 |4| 7,6,5,8} Global: {3,2,1,4,7,6,5,8}
L QS partition: Input: {3,2,1} Output: {1|2,3} Global: {1,2,3,4,7,6,5,8}
R QS partition: Input: {7,6,5,8} Output: {7,6,5—8} Global: {1,2,3,4,7,6,5,8}
L QS partition: Input: {2,3} Output: {2|3} Global: {1,2,3,4,7,6,5,8}
R QS partition: Input:{7,6,5} Output: {5|7, 6} Global: {1,2,3,4,5,7,6,8}
L QS partition: do nothing
R QS partition: Input:{7,6} Output:{6|7} Global: {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.* We consider two layers of the recursion tree at once. Let us consider two functions, $T_1(n)$ $T_2(n)$, where $T_i(n)$ denotes the running time of the algorithm that starts in a layer of type $i$. Assume that this modification of parition with still be $\Theta(n)$, and that the algorithm will start off with picking the maximum pivot from level 0, we get a recurrence relation for each $T_i$ that involves the next $T_i$. We replace $\Theta(n)$ with a corresponding term $c$, where c is an arbitrary constant.

$$T_1(n) = T_2(n-1) + cn : minimumpivot$$
$$T_2(n) = 2T_1(n/2) + cn : medianpivot$$

We have determined that the recurrence relation of $T_1(n) = 2T_1(\frac{n}{2} - 3) + 2cn$ for our modified quick sort. Thus, we have that:

$$T_1(n) = 2T_1(\frac{n}{2} - 3) + 2cn$$
$$T_1(n) \leq 2T_1(n/2) + cn$$

We use the tree method to solve $T_1(n) \leq 2T_1(n/2) + cn$ to obtain an upper bound $T_1(n)$. We hit a base case when $n/2^k \leq 1$. Solving for k we have that:
$k = \log_2(n)$
We observe that at level i, the non-recursive work in a given node is $cn(1/2^i)$. There are $2^i$ such nodes at level $i$ so our total work at level $i$ is:
$cn * \frac{2^i}{2^i} = cn * 1$
Thus

$$T(n) = \sum_{i=0}^{\log_2(n)} cn * 1$$
$$= cn \sum_{i=0}^{\log_2(n)} 1$$
$$= cn * \log_2(n)$$

Therefore $T(n) \in \Theta(n \log_2(n))$ □