CSCI 3104 Spring 2022
Instructor: Profs. Chen and Layer

Problem Set 6

Due Date .... March 8
Name .... **Julia Troni**
Student ID **109280095**
Collaborators .... **Null**

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.

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

Using loop unrolling:

$$\begin{aligned} T(n) &= T(n-4) + 4 + cn \\ &= T(n-8) + c(n-4) + cn + 2 \cdot 4 \\ &= T(n-12) + c(n-8) + c(n-4) + cn + 3 \cdot 4 \\ &= T(n-16) + c(n-12) + c(n-8) + c(n-4) + cn + 4 \cdot 4 \\ &\ldots \\ &= T(n-4k) + c \cdot \sum_{i=0}^{k} (n-4i) + k \cdot 4 \end{aligned}$$

To find the number of times we need to unroll, we know we hit the base case when $n - 4k < 5$
So solve for k to get : $k > \frac{n-5}{4}$
So we unroll until $k = \lceil \frac{n-5}{4} \rceil$
Thus we have that:

$$\begin{aligned} T(n) &= T(n - 4 \cdot \lceil \frac{n-5}{4} \rceil) + c \cdot \sum_{i=0}^{\lceil \frac{n-5}{4} \rceil} (n-4i) + \cdot \lceil \frac{n-5}{4} \rceil \cdot 4 + 4 \cdot \lceil \frac{n-5}{4} \rceil \\ &= \Theta(1) + cn \cdot \lceil \frac{n-5}{4} \rceil - \frac{4c \lceil \frac{n-5}{4} \rceil \cdot (\lceil \frac{n-5}{4} \rceil + 1)}{2} + 4 \cdot \lceil \frac{n-5}{4} \rceil \\ &= \Theta(1) + cn \cdot \lceil \frac{n-5}{4} \rceil - 2c \lceil \frac{n-5}{4} \rceil \cdot (\lceil \frac{n-5}{4} \rceil + 1) + 4 \cdot \lceil \frac{n-5}{4} \rceil \end{aligned}$$

Thus we have that $T(n) \in \Theta(n^2)$

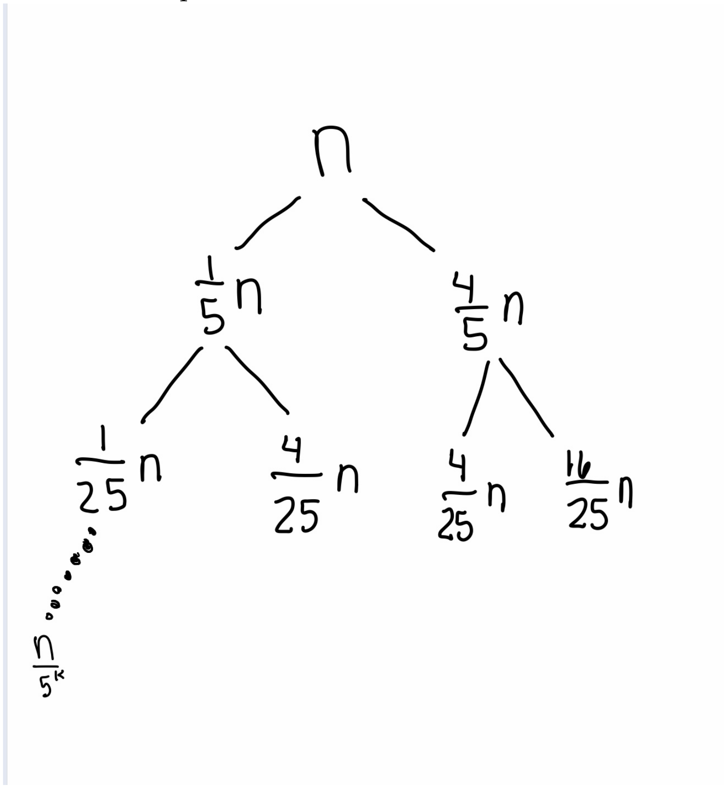
□

(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.

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

The tree is depicted below



Note that the cost at each level (as long as no branches have reached their base case) is n. Thus we do at most $O(n)$ work in each layer
The upper bound of the number of possible layers is the depth of the deepest base case path. The largest recursion at each level is $\frac{n}{5}$, so the longest depth of the tree will occur when we reach base case for the $T(\frac{n}{5})$ term.
We hit that base case when $\frac{n}{5^k} \leq 5$
So solve for k to get : $k \geq log_5 n - 1$
So the path must terminate by depth $k = \lceil\ log_5 n - 1\rceil$
Thus the total run time is $T(n) \leq n \cdot \lceil\ log_5 n - 1\rceil$

Thus we have that $T(n) \in O(nlog_5 n)$ $\square$

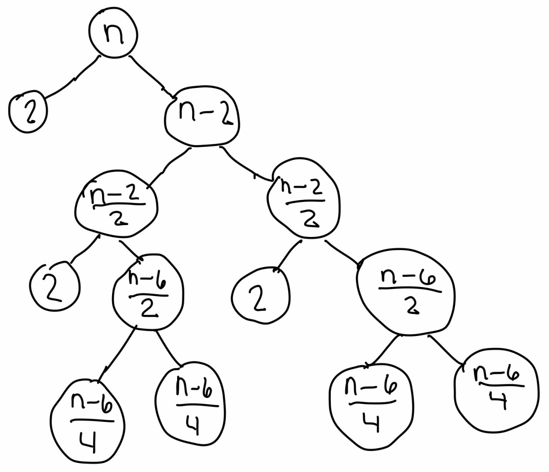
(c) Suppose that we modify the Merge-Sort algorithm in such a way that on alternating levels of the recursion, the partitioning is either a $(2, n-2)$ split or a $(n/2,\ n/2)$ split. Write down a recurrence relation for this modified Merge-Sort algorithm and give its asymptotic solution. Then, give a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

Answer. The recurrences are

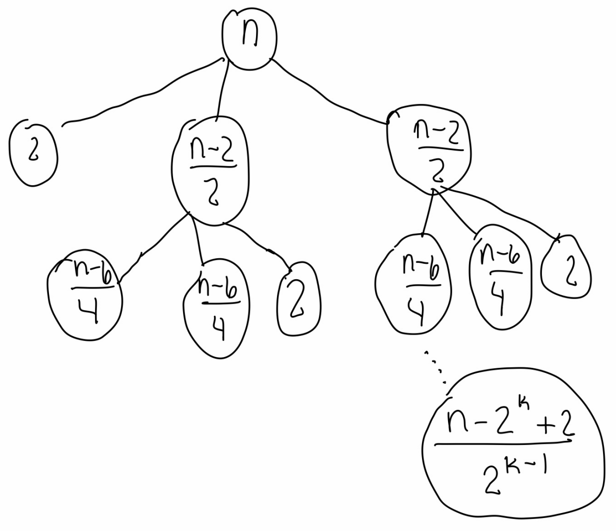
$$T_1(n) = \begin{cases} \Theta(1) & : n \leq 2, \\ T_2(n-2) + 2 + \Theta(n) & : n > 2. \end{cases}$$

$$T_2(n) = \begin{cases} \Theta(1) & : n \leq 2, \\ 2T_1(\frac{n}{2}) + \Theta(n) & : n > 2. \end{cases}$$

The tree is below:



I will now examine every other level of the tree to identify the base case



Observe that at level i of the tree the amount of nonrecursive work is $2 \cdot (n - 2^i + 2) = 2(n+2) - 2 \cdot 2^i$

Let us now determine the number of levels of the tree. Let k denote the number of levels of the tree. We will solve for k. The leaf nodes correspond to the base cases. We hit base cases when $\frac{n-2^k+2}{2^{k-1}} \leq 2$
Solving for k we get $log_2(n+2) - 1 \leq k$
So the number of levels of the tree is $k = \lceil log_2(n+2) - 1 \rceil$

So the total work is

$$\begin{aligned}
T(n) &\leq \sum_{i=0}^{\lceil log_2(n+2)-1 \rceil} (2(n+2) - 2 \cdot 2^i) \\
&\leq 2(n+2) \sum_{i=0}^{\lceil log_2(n+2)-1 \rceil} (1) - 2 \cdot \sum_{i=0}^{\lceil log_2(n+2)-1 \rceil} (2^i) \\
&\leq 2(n+2) \cdot (\lceil log_2(n+2) - 1 \rceil) - 2 \cdot \frac{(n+2)}{2} \\
&\leq 2(n+2) \cdot (\lceil log_2(n+2) - 1 \rceil) - (n+2)
\end{aligned}$$

Thus, we have that $T(n) \in O(nlogn)$. Thus, this merge sort algorithm does not signicicantly change the running time of merge sort, as traditional merge sort is $T(n) \in \Theta(nlogn)$ $\square$

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	global array
first partition	x=4, p=0, r=7	A=[3,7,1,8,2,6,5,4]	A=[3,1,2, **4**, 7, 6, 5, 8]	A=[3,1,2, 4, 7, 6, 5, 8]
LQuicksort partition	x=2, p=0, r=2	A=[3,1,**2**]	A=[1, **2**, 3]	A=[1, 2, 3, 4, 7 ,6, 5, 8]
RQuicksort partition(A,4,7)	x=8, p=4, r=7	A=[7,6,5,**8**]	A=[7,6,5,**8**]	A=[1, 2, 3, 4, 7 ,6, 5, 8]
L(LQuicksort) partition	do nothing			A=[1, 2, 3, 4, 7 ,6, 5, 8]
L(RQuicksort) partition	do nothing			A=[1, 2, 3, 4, 7 ,6, 5, 8]
R(LQuicksort) partition	x=5, p=4, r=6	A=[7,6,**5**]	A=[**5**,6,7]	A=[1, 2, 3, 4, 5 ,6, 7, 8]
R(RQuicksort) partition	do nothing			A=[1, 2, 3, 4, 5 ,6, 7, 8]
RL(LQuicksort) partition	do nothing			A=[1, 2, 3, 4, 5 ,6, 7, 8]
RL(RQuicksort) partition	x=7, p=5, r=6	A=[6,**7**]	No change A=[6,**7**]	A=[1, 2, 3, 4, 5 ,6, 7, 8]
RLR(LQuicksort) partition	do nothing			A=[1, 2, 3, 4, 5 ,6, 7, 8]
RLR(RQuicksort) partition	do nothing			A=[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.

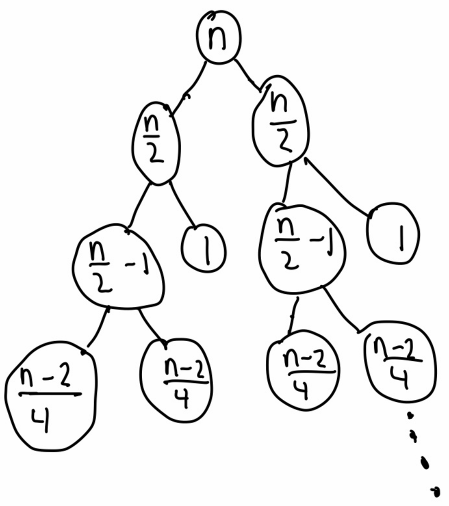
Write down a recurrence relation for the running time of this version of QUICKSORT given an array n distinct elements and solve it asymptotically, i.e. give your answer as $\Theta(f(n))$ for some function $f(n)$. Show your work.

Answer. The recurrences are

$$T_1(n) = \begin{cases} \Theta(1) & : n \le 2, \\ T_2(n-1) + 1 + \Theta(n) & : n > 2. \end{cases}$$

$$T_2(n) = \begin{cases} \Theta(1) & : n \le 2, \\ 2T_1(\frac{n}{2}) + \Theta(n) & : n > 2. \end{cases}$$

The tree is below:



Observe that at level i of the tree the amount of nonrecursive work is $c(\frac{n}{2^i})$ and there are 2^i nodes at level i
So the nonrecursive work at level i is : $c(\frac{n}{2^i}) \cdot 2^i = cn$

Let us now determine the number of levels of the tree. Let k denote the number of levels of the tree. We will solve for k. The leaf nodes correspond to the base cases. We hit base cases when $\frac{n}{2^k} \le 2$
Solving for k we get $log_2 n - 1 \le k$
So the number of levels of the tree is $k = \lceil log_2(n) - 1 \rceil$

So the total work is

$$\begin{aligned}
T(n) &\leq \sum_{i=0}^{\lceil log_2(n)-1\rceil} (cn) \\
&\leq cn \cdot \lceil log_2(n) - 1\rceil
\end{aligned}$$

Thus, we have that $T(n) \in O(nlogn)$

$\square$