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.

$$T_n = \begin{cases} 1 & : n \leq 4 \\ T(4) + T(n-4) + O(n) & : \text{otherwise} \end{cases} \tag{1}$$

If k is the maximum depth of recursion, then $n - 4k \leq 4$. Solving for the smallest such k, $k = \frac{n-4}{4}$. At level i, we perform $O(1) + O(n)$ work. Thus, the total work is at most

$$\begin{aligned} \sum_{i=0}^{\frac{n-4}{4}} O(1) + O(n) &= O\left(\frac{n-4}{4}\right) + O\left(n \cdot \frac{n-4}{4}\right) \\ &= O(n^2) \end{aligned}$$

□

(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.

$$T_n = \begin{cases} 1 & : n \leq 4 \\ T(4/5 \cdot n) + T(1/5 \cdot n) + O(n) & : \text{otherwise} \end{cases} \tag{2}$$

Let k denote the maximum depth of recursion. Since the larger portion of the input on which we recurse is the $4n/5$ portion, we have that $n \cdot \left(\frac{4}{5}\right)^k \leq 4$, and $k = \log_{5/4} \frac{n}{4}$. At level i we do at most $O(n)$ work, so our total work is at most $O(n) \cdot \log_{5/4} \frac{n}{4} = O(n \log n)$.

□

(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. Define two mutual recurrence relations describing the complexity on alternating depths of the tree

$$T_1(n) \in \begin{cases} \Theta(1) & : n \leq 2 \\ 2T_2(n/2) + \Theta(n) & : \text{otherwise} \end{cases}$$

$$T_2(n) \in \begin{cases} \Theta(1) & : n \leq 2 \\ T_1(2) + T_1(n-2) + \Theta(n) & : \text{otherwise} \end{cases}$$

By noting that $T_1(2) \in \Theta(1)$ and $\Theta(1) + \Theta(n) = \Theta(n)$ (the sum of a function in $\Theta(1)$ and a function in $\Theta(n)$ is a function in $\Theta(n)$) we can simplify the second relation to

$$T_2(n) \in \begin{cases} \Theta(1) & : n \leq 2 \\ T_1(n-2) + \Theta(n) & : \text{otherwise} \end{cases}$$

For $n > 2$, we have

$$\begin{aligned} T_1(n) =& 2T_2(n/2) + \Theta(n) \\ & 2(T_1(n/2-1)) + \Theta(n) \end{aligned}$$

We reach a base case after k unrolls when $\frac{n}{2^{k/2}} - 1 \leq 2$ or $k \geq 2\log_2(n/3) \in \Theta(\log n)$ and at each level of the tree we perform $\Theta(n)$ units of non-recursive work.

Our total complexity is then $\Theta(n \log n)$.

□

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). page 5. on Week 8's note

Answer. ""

input	output			global
$[3, 7, 1, 8, 2, 6, 5, \mathbf{4}]$	$[3, 1, 2]$	$[\mathbf{4}]$	$[7, 6, 5, 8]$	$[3, 1, 2, \mathbf{4}, 7, 6, 5, 8]$
$[3, 1, \mathbf{2}]$	$[1]$	$[\mathbf{2}]$	$[3]$	$[1, \mathbf{2}, 3, 4, 7, 8, 6, 5]$
$[7, 6, 5, \mathbf{8}]$	$[7, 6, 5]$	$[\mathbf{8}]$	$[]$	$[1, 2, 3, 4, 7, 6, 5, \mathbf{8}]$
$[7, 6, \mathbf{5}]$	$[]$	$[\mathbf{5}]$	$[6, 7]$	$[1, 2, 3, 4, 5, \mathbf{5}, 6, 7]$
$[6, \mathbf{7}]$	$[6]$	$[\mathbf{7}]$	$[]$	$[1, 2, 3, 4, 5, 6, \mathbf{7}]$

□

p r

i j

3, 7, 1, 8, 2, 6, 5, (4)

since $3 < 4 \Rightarrow i = i+1$, and swap (S_i, S_j)

i j

3, 7, 1, 8, 2, 6, 5, 4

since $1 < 4 \Rightarrow i = i+1$, and swap (S_i, S_j)

i j

3, 1, 7, 8, 2, 6, 5, 4

since $2 < 4 \Rightarrow i = i+1$, and swap (S_i, S_j)

i j

3, 1, 2, 8, 7, 6, 5, 4

since $j = r-1$, we stop partition after exchange (S_{i+1}, S_r)

3, 1, 2, 4, 7, 6, 5, 8

$\Rightarrow$ We finish the first partition. using 4 as a pivot.

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. Perhaps the simplest way to write this recurrence relation is to consider two layers of the recursion tree at once, but let's derive it slowly. Note that there are really two different kinds of layers: (1) layers in which the algorithm chooses the best possible pivot (median), and (2) layers in which the algorithm chooses the worst possible pivot (maximum element). Let us have 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 (for $i = 1, 2$). We get a recurrence relation for each T_i that involves the next T_i:

$$T_1(n) = 2T_2(n/2) + \Theta(n) = 2\left(T_1\left(\frac{n}{2}-1\right) + \Theta(n)\right) + \Theta(n) \quad (3)$$

$$T_2(n) = T_1(n-1) + \Theta(n). = 2T_1\left(\frac{n}{2}-1\right) + 3\Theta(n) \quad (4)$$

The derivations of these are exactly as in class, except that the next layer of the tree is of the other type so the recursive calls are on the other numbered T. (We comment on the rounding issues later.)

We can still use substitution to unroll these recurrence relations. So we have (inserting an explicit constant for our calculations)

$$T_1(n) = 2T_2(n/2) + cn$$

Applying (2), we have

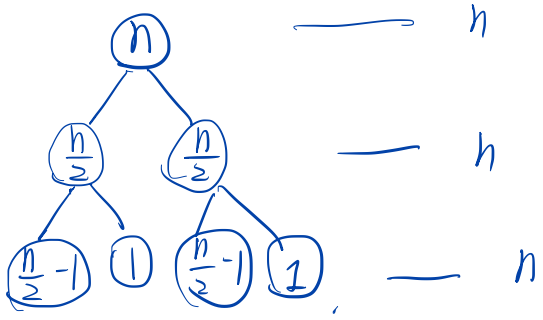
$$T_1(n) = 2T_1(n/2 - 1) + 2cn$$
$$T_1(n) \leq 2T_1(n/2) + \Theta(n).$$

We use the tree method to solve $T_1(n) \leq 2T_1(n/2) + \Theta(n)$ to obtain an upper bound for $T_1(n)$.

- We hit a base case when $n/2^k \leq 1$. Solving for k, we obtain that $k = \lceil \log_2(n) \rceil$.
- At level i of the recursion tree, our non-recursive work at a given node is $cn/2^i$. There are 2^i such nodes. So our total work at level i is cn.
- Thus, $T(n) = cn \cdot \lceil \log_2(n) \rceil$. So $T(n) \in \Theta(n \log(n))$.

$T_1(n) = 2T_1\left(\frac{n}{2}-1\right) + 3\Theta(n)$

$\frac{n}{2^k} \leq 1 \quad k = \lceil \log_2(n) \rceil \approx \log_2 n$



$T_1(n) = 2^k + 3k\,\Theta(n)$

$= 2^{\log_2 n} + 3\log_2 n\,\Theta(n) = n^{\log_2 2} + \Theta(n \log n) = \Theta(n \log n)$

$\geq n.$

□