CS5800 Algorithms *Out: 20 September 2022*

Problem Set 2

Ravi Sundaram *Due: 30 September 2022*

Problem 1 (Recurrences) $20 = 4 + 4 + 8 + 4$

Using techniques used in class determine the asymptotic runtime of the following recurrences. If you use the Master Theorem you must state which case of master theorem you are using and show its preconditions hold.
For each of the subparts you may assume T(1) = O(1)

1. $T(n) = 3T(n/2) + n$ (This is the recursion for Karatsuba's Algorithm)
2. $T(n) = 3T(n/2) + n^2$
3. $T(n) = 2T(n/4) + T(n/2) + n$ (Hint: Think of the number of nodes at each level. You can determine it by solving a recurrence).
4. $T(n) = 5T(n/4) + n$

Problem 2 (Sorting special arrays) 20

Consider the problem of sorting an array $A[1, ..., n]$ of integers. We presented an $O(n \log n)$-time algorithm in class and, also, proved a lower bound of $\Omega(n \log n)$ for any comparison-based algorithm.

1. Come up with an efficient sorting algorithm for a ***boolean***[1] array $B[1, ..., n]$.
2. Come up with an efficient sorting algorithm for an array $C[1, ..., n]$ whose elements are taken from the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.
3. Come up with an efficient sorting algorithm for an array $D[1, ..., n]$ whose elements are distinct ($D[i] \neq D[j]$, for every $i \neq j \in \{1, ..., n\}$) and are taken from the set $\{1, 2, ..., 100n\}$.
4. In case you designed linear-time sorting algorithms for the previous subparts, does it mean that the lower bound for sorting of $\Omega(n \log n)$ is wrong? Explain.

Problem 3 (Lower Bounds) 15

Show an $\Omega(\log n)$ lower bound for finding elements in a sorted array (in a comparison based model).

Problem 4 (Peak Detection) $45 = 10 + 5 + 30$

For this problem you will be required to complete the following 3 tasks for the "Problem Statement" below:

1. Description of algorithm (10 points) - Provide an explanation of your algorithm and how it uses divide and conquer to solve the problem.
2. Runtime analysis (5 points) - Provide analysis for why your algorithm is $\Theta(nlog(n))$.

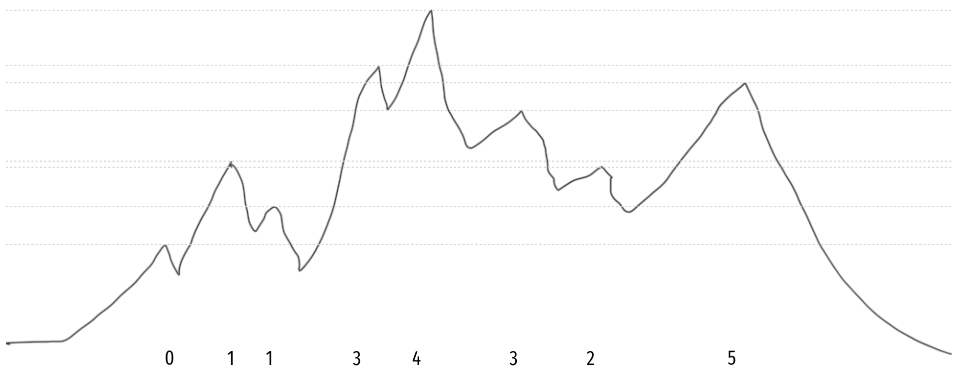
[1] In a boolean array $B[1, ..., n]$, each element $B[i]$ (for $i = 1, ..., n$) is either 0 or 1.

3. Code (30 points) - submitted to the following Hackerrank link:
   https://www.hackerrank.com/cs5800-fall22-pset2.

The description of the algorithm and runtime analysis should be individually typed and submitted in your submission similar to questions 1-3. Still only one submission of the code is required per group. Everyone in the group should list the username of the submission to be used for grading at the top of their answer.

Problem Statement: **IMPORTANT** You are NOT allowed to use built in language functions which trivialize the task of computing exponents. Any submissions which use this and avoid the task at hand will be given a 0.

Peak detection in signals has many applications, from scanning for irregularities in the heart to identifying outliers in data. A robust algorithm to for peak detection is very useful. Define the *left peak function* , denoted $p_\ell(s)$, s as the total number of times that a peak is taller than one of its left neighboring peaks. The *right peak function*, $p_r(s)$, is defined analogously. The signal s in image below has 8 peaks, $p_\ell = 19$, the contribution of each peak is listed below.



Design and analyze a divide and conquer algorithm that computes the left function of a signal s with n peaks. The input $A[1, \ldots, n]$ consists of the heights of each peak in left to right order; assume all peaks have unique heights. Your solution should have a running time of $\Theta(n \log n)$.

Input Format: First line is **n**, followed by n unique integers separated by spaces.

5

3 4 1 8 2 .

Constraints: The integers will satisfy $2 \leq n \leq 2^{64}$ and $1 \leq$ Peak height $\leq 2^{64}$

Output Format: An integer output of left peak funciton.

5

Explanation:There are 5 peaks, p1=3, p2=4, p3=1, p4=8, p5=2.

Left peak function :

Number of peaks to the left of p1, shorter than p1 = 0

Number of peaks to the left of p2, shorter than p2 = 1

Number of peaks to the left of p3, shorter than p3 = 0

Number of peaks to the left of p4, shorter than p4 = 3

Number of peaks to the left of p5, shorter than p5 = 1

Output of left peak function = 0+1+0+3+1 = 5