ESO207: Data Structures and Algorithms

Theory Assignment 1

Due Date: 12th February, 2021

Total Number of Pages: 2 | Total Points 80

Instructions

1. The assignment contains 2 parts - Part 1 and Part 2. Please submit both parts in separate files titled `part1_roll.pdf` and `part2_roll.pdf` respectively (where `roll` is your roll no). Failure to do so will result in loss of marks.
2. For each question you must give the pseudocode of your algorithm and a brief description of the idea of your algorithm.
3. If an algorithm requires a certain time complexity or space complexity, then you must describe why your algorithm indeed works in that complexity bound.
4. The teaching assistant in charge of Part 1 is Madhusmita Sahoo (`madhusmita@cse.iitk.ac.in`) and in charge of Part 2 is Neeraj Matiyali (`neermat@cse.iitk.ac.in`). Contact them if you have any doubts.

Part 1

Problem1. (10 points) Let A be an array consisting of n elements such that there exists indices $i, j, k \in [0, n-1]$, where $0 \leq i < k < j \leq n-1$, and $A[0] < A[1] < ... < A[i-1] < A[i] > A[i+1] > ... > A[k] < A[k+1] < A[k+2] < ...A[j-1] < A[j] > A[j+1] > A[j+2].... > A[n-1]$. Design an $O(\log n)$ time algorithm to find i and j (the two local maxima) in the array. Describe your approach and prove the correctness of your algorithm.

Problem2. (10 points) Given a doubly linked list $a_1, a_2, \ldots, a_n$, rotating it from location p to location q to the right by k places gives the list $a_1, \ldots, a_{p-1}, a_{q-k+1}, \ldots, a_q, a_p, \ldots, a_{q-k}, a_{q+1}, \ldots, a_n$.

For example if $p = 3$, $q = 7$ and $k = 2$ then your algorithm should return L' .

Location	1	2	3	4	5	6	7	8	9	10
Before Rotation(L)	22	13	17	35	11	56	49	64	28	62
After Rotation(L')	22	13	**56**	**49**	**17**	**35**	**11**	64	28	62

Design an $O(n)$ time algorithm to rotate a sub-list from location p to location q to the right by k places using only $O(1)$ (not $O(k)$) extra space.

Problem3. (10 points) Suppose you are given two sorted arrays $A[0, \ldots, n-1]$ and $B[0, \ldots, n-1]$. Design an algorithm to find the median of the array obtained after merging the above two arrays (i.e. array of length $2n$). The time complexity of the algorithm should be $O(\log n)$. You can use only $O(1)$ extra space only. Prove the correctness of your algorithm.

Example :

A	12	17	23	34	65

B	40	53	59	61	66

Output : $(40 + 53)/2 = 46.5$

Problem4. (10 points) Suppose you are given a sorted array A storing n distinct positive integers, and three positive integers a,b and c. Design an $O(n)$ time algorithm to determine if there exist any two distinct integers $x, y \in A$ such that $a^2 = bx + cy$. Prove correctness of your algorithm.

Part 2

Problem5. (10 points) Prove the following statements:

(a) $\min(n^2, 10^{12}) = \mathcal{O}(1)$

(b) $n^2 + n \log n = \mathcal{O}(n^2)$

(c) $n^3 + 3n^2 + 8 \neq \mathcal{O}(n^2)$

(d) $4^n \neq \mathcal{O}(2^n)$

(e) $\log(n!) = \mathcal{O}(n \log n)$

Problem6. (10 points) Design an $\mathcal{O}(\log n)$ time algorithm that takes as input two arrays A and B sorted in ascending order and outputs an index k for which $A[k] = B[n-1-k]$. If such an index does not exist then your algorithm must return -1 to indicate failure. Describe your approach, provide the pseudocode and prove that your algorithm is correct and has a worst time complexity of $\mathcal{O}(\log n)$.

Example: For following input A and B ($n = 8$), the correct output is $k = 5$, since $A[5] = B[8-1-5] = B[2] = 10$.

index i	0	1	2	3	4	5	6	7
$A[i]$	-5	-2	-1	0	5	**10**	120	150
$B[i]$	-10	5	**10**	15	17	40	47	90

Problem7. (10 points) Consider two sets of points $A = [(a_x^0, a_y^0), \ldots (a_x^{n-1}, a_y^{n-1})]$ and $B = [(b_x^0, b_y^0), \ldots, (b_x^{n-1}, b_y^{n-1})]$ sampled from two parallel lines l_A and l_B, respectively. Design an $O(n \log n)$ time algorithm that takes A and B as input and returns two points (a_x^p, a_y^p) and (b_x^q, b_y^q) $(0 \leq p, q \leq n-1)$ from sets A and B that are closest to each other. Describe your algorithm and provide the pseudocode. Prove the correctness of your algorithm and show that it runs in $\mathcal{O}(n \log n)$ time.

Problem8. (10 points) You are given an unsorted array A. Design an algorithm that, given a query $[a, b]$ $(a, b \in \mathbb{R}, a \leq b)$, finds the following in $\mathcal{O}(\log n)$ time:

1. the total number of elements in A whose values lie in the interval $[a, b]$
2. the value in $[a, b]$ which occurs the most in A.

Example:

$$A = [1.5, 1.5, -2, 0, 2, 0, 0, 3.2, 0, 3, 2.4, -1, 1, 1, 1.7, 1.5, 1.2, -3, -2.1, -5]$$

- For query $[0.8, 3]$, there are 10 elements ($[1.5, 1.5, 2, 3, 2.4, 1, 1, 1.7, 1.5, 1.2]$) in the query interval, and 1.5 is the most frequent one.
- For query $[-0.2, 1.3]$, there are 7 elements ($[0, 0, 0, 0, 1, 1, 1.2]$) in the query interval, and 0 is the most frequent one.

Explain your approach and provide the pseudocode. You may use $\mathcal{O}(n \log n)$ time to preprocess the input data once. You can keep $\mathcal{O}(n \log n)$ additional space.