Design and Analysis of Algorithms

CS 302 Fall 2020

# Homework # 2

Each problem carries 10 marks.

Give an **efficient algorithm** for each of the following problems.

**Q1)** Given a set N of integers and an integer y, determine if there exit two elements in N whose absolute difference is equal to y and also print those numbers. The algorithm should take O(n lg n) time. Justify why your algorithm runs in O(n lg n) time. [10 Marks]

e.g. Let N = 3 , 7, 2, 1, 4, 10

y = 1

there are three pairs of elements in N whose absolute difference is 1

Pair 1 = |3 - 2| = |-1| = 1

Pair 2 = |3 - 4|= |-1| = 1

Pair 3 = |2 -1| = 1

**Q2)** Given two sorted arrays X[ ] and Y[ ] of sizes M and N where M ≥ N, devise an algorithm to merge them into a new sorted array C[ ] using O( N lg M) comparison operations. Suppose arrays M and N are indexed from 1 to M and from 1 to N respectively.

*Hint*: use binary search.

**Q3)** Consider an array of distinct numbers sorted in increasing order. The array has been rotated (clockwise) k number of times. Given such an array, find the value of k. The solution should be efficient and use divide and conquer approach.

**Examples:**
Input : arr[] = {15, 18, 2, 3, 6, 12}
Output: 2
Explanation : Initial array must be {2, 3, 6, 12, 15, 18}. We get the given array after rotating the initial array twice.

Input : arr[] = {7, 9, 11, 12, 5}
Output = 4

**Q4)** Given an array of integers which is initially increasing and then decreasing, find the maximum value in the array.

**Examples :**

Input: arr[] = {8, 10, 20, 80, 100, 200, 400, 500, 3, 2, 1}
Output: 500

Input: arr[] = {1, 3, 50, 10, 9, 7, 6}
Output: 50

Corner case (No decreasing part)
Input: arr[] = {10, 20, 30, 40, 50}
Output: 50

Corner case (No increasing part)
Input: arr[] = {120, 100, 80, 20, 0}
Output: 120

**Q5)** Suppose you have an unsorted array A of colors *red*, *white* and *blue.* You want to sort this array so that all *reds* are before all *whites*, followed by all *blues*. Only operations available to you for this purpose are: equality comparison A[i] = = c where c is one of the three colors, and swap(i, j) which swaps the colors at indices i and j in A. How to sort this array in O(n) worst case time and O(1) additional space. Assume that some satellite data is also there with these colors so counting the number of reds, whites and blues will not solve the problem.

**Q 6)** Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range [a….b] in O(1) time. Your algorithm should use O(n + k) preprocessing time.