Lab Worksheet 2

Question 1.

(a) An entry of 2D array is called "peak" if its value is the largest among its neighbors (i.e. peak element is the largest among its right, left, up, down elements and itself). Given that you are given a matrix of size $m \times n$, devise an algorithm that returns one possible peak within $O(m\log n)$ time. (Hint: Try binary search)

(b) Given an $n \times n$ 2D array where every row and column is sorted in non-decreasing order. Find the kth smallest element in the given 2D array which has time complexity equal to or better than $O(n\log(n))$.

Question 2. Consider a problem where you need to search String with length of at most m among maximum n number of Strings. For instance, consider a situation where you need to query whether String A is in the set of above strings. If you brute force, then it will take $O(mn)$ time as you need to go through n strings and check whether every letter match up with the desired string, which takes $O(m)$ time. It is very inefficient indeed! Therefore, your job is to come up with $O(m)$ solution for this problem.

Remark) Huffman Coding works in a similar way; one can prove that Greedy solution for Huffman coding always yield the most optimal prefix coding:)

Question 3. In Statistics, the median of an array A of N integers is a value which divides array into two equal parts: the higher half and the lower half. In the case when N is odd, the median is clearly defined: the middle value of the sorted version of array. In the case when N is even, the median is the average of the two middle values, rounded down so that we always have an integer median. In this problem, your task is to compute median values

as integers are added into array A one by one. Since this is an incremental process, we call these "continuous medians". To simplify this problem, you just need to output the sum of the medians of A at the end. Your algorithm should attain $O(N\log(N))$ time.

Question 4. (Amortized Analysis) In the Set Union problem we have n elements, that each are initially in n singleton sets, and we want to support the following operations:

(1) Union(A,B): Merge the two sets A and B into one new set $C = A \cup B$, destroying the old sets.

(2) SameSet(x, y): Return true, if x and y are in the same set, and false otherwise. We implement it in the following way. Initially, give each set a distinct color. When merging two sets, recolor the smaller (in size) one with the color of the larger one (break ties arbitrarily). Note, to recolor a set you have to recolor all the elements in that set. To answer SameSet queries, check if the two elements have the same color. (Assume that you can check the color an element in $O(1)$ time, and to recolor an element you also need $O(1)$ time. Further assume that you can know the size of a set in $O(1)$ time.)

Show that on average, the worse case time complexity for Union method is $O(\log(n))$. That means, show that any sequence of m union operations take $O(m\log(n))$ time. (Note: We start with m singleton sets).

Question 5. We have learnt fabulous BST & bBST data structure in CS2040 lecture. In fact, one important application of balanced BST is that it can be used to prove that time complexity of comparison based sorting is at least $O(n\log(n))$ time.

Part (a) Draw a decision tree for sorting an Array $A := [1, 2, 3]$. Is decision tree for comparison sorting always balanced? If so, could you prove that it is always balanced where balance factor for every node is smaller or equal to 1?

Part (b) Show that comparison based sorting could not perform better than $O(n\log(n))$ time complexity. (Hint: you might find Sterling's approximation useful).

Question 6. Given that you have max heap of size M and N, devise an algorithm that merges two max heap within $O(M + N)$ time.

Question 7. Given two equally sized arrays (A, B) of size N, a sum combination is made by adding one element from array A and another element of array B. Display the maximum K valid sum combinations from all the possible sum combinations within $O(N\log(N))$ time.