DSA 2021 Assignment -2

General Instructions:
All problems are compulsory.
Inbuilt algorithm Implementations can't be used in any of the problems.
For every implementation, pseudo-code and running time analysis should be present.

Problem 1: In a N-ary land, people really like the number 'n'. You are their chief algorithms expert and you are given the task of designing an n-ary heap for these people. An n-ary heap is like a binary heap, but (with one possible exception) non-leaf nodes have 'n' children instead of 2 children. You have to design an n-ary heap and answer the following questions:

a) How can we represent an n-ary heap in an array?
b) Let's say we have an n-ary heap which contains 'd' number of elements in total. Give the height of this heap in terms of 'n' and 'd'.
c) Give an efficient implementation of EXTRACT-MAX and INSERT in an n-ary Max heap. Analyse the running times of both of the functions in terms of 'd' (total size) and n.
d) Give an efficient implementation of DECREASE-KEY(A, i, k) which will give an error if k > A[i], but otherwise sets A[i] = k and then updates the n-ary max-heap structure appropriately. Analyse its running time in terms of 'd' (size) and n.

Assume array based implementation of the heap (for parts b, c and d) and array is always sufficiently large to accommodate all elements, if total elements in heap are 'd' then array[d+1] can be assumed to be 0 or -1. Also please assume elements to be stored in heap to be positive. Please write an explanation along with every pseudo code for parts c and d.

Marking Scheme: Total 15 marks: (2.5 + 2.5 + 5 + 5)

Problem 2: Dumbledore's Luncheon
Dumbledore is hosting a luncheon at Hogwarts, and Ron is incharge of the sitting plans. Dumledore's dining table can be thought of as an infinite row of seats with Dumbledore at one end. Each of Dumbledore's guests represents a character of English alphabet (a to z). Dumbledore wants his guests to be seated such that the resulting string formed from the alphabets should be lexicographically smallest (See examples for reference). He didn't tell Ron the total number of guests coming, but he will ask Ron to close the gate once all the guests arrive (marking the end of incoming guests).
The rules of seating are as follows:

- Seats are allotted in sequential order, a guest who arrives later must be further from Dumbledore than a guest who arrives earlier.
- An incoming guest must be assigned a seat, as soon as he arrives. He will take a seat.

- Once a guest takes their seat, you can't ask them to move to some other seat. Even if the seat in front of him is empty, they will stay at their occupied seat only.
- But you can make a guest disappear using your spells, Dumbledore has allowed you to make at most 'k' of his guests disappear. He can handle 'k' guests not in the party but you can't remove more than 'k' guests.
- No seat should remain empty between any two guests or between guests and Dumbledore. Since the table has infinite seats, a series of empty seats would be present in the end.

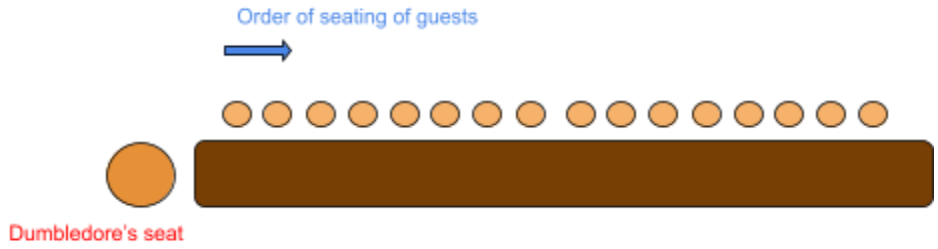
You are Ron, design a code using an efficient data structure that your magic wand will follow to keep track of which guest to assign which seat. Explain the running time and also explain why you have chosen the data structure which you have used.

Assumptions:

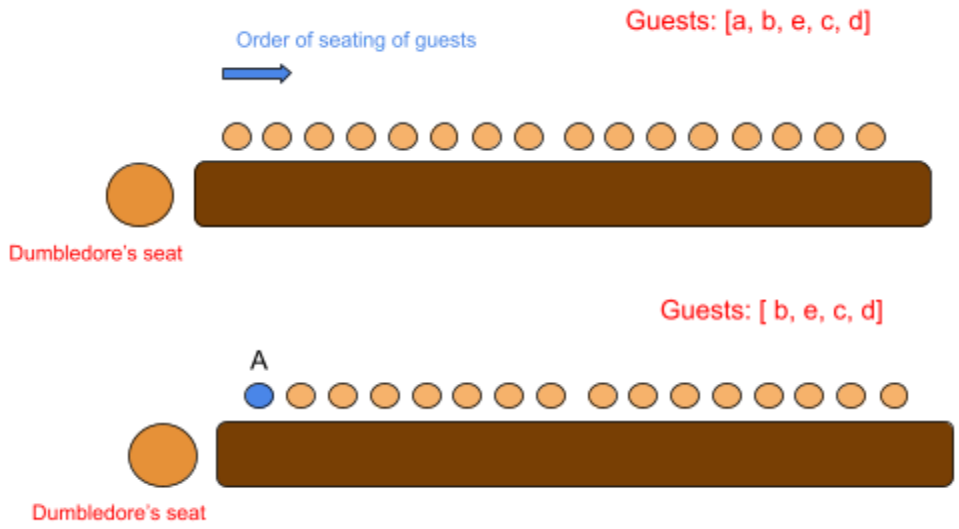
> You may assume that you have enough time when a guest comes so that you are able to make some existing guest disappear before assigning the new guest his seat.

> When Dumbledore asks you to close the gate, you can still make some guests disappear. Dumbledore will have a look at your arrangement after you close the gate, it should be lexicographically smallest by then.

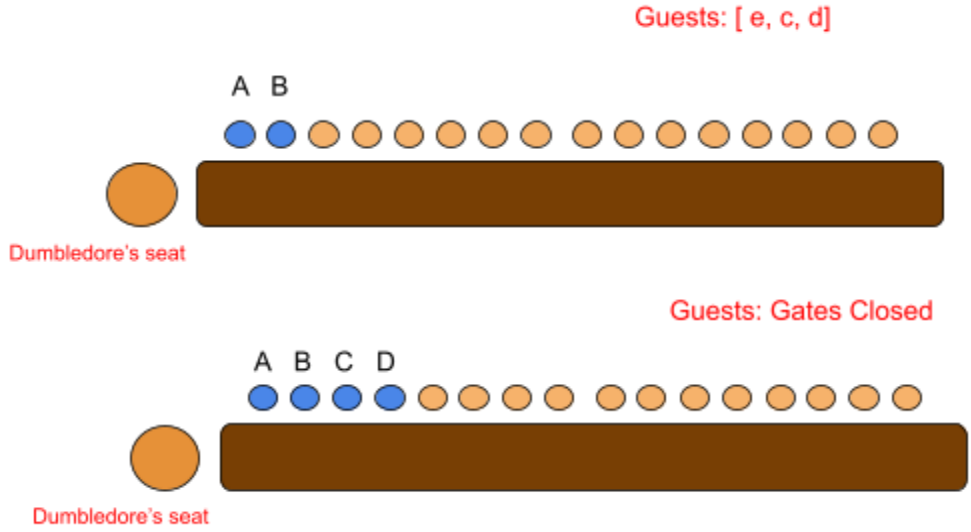
> When a guest arrives, you don't have access to the list of guests yet to arrive, you can't decide where to seat the current guest by accessing the list of remaining guests.



Let's say incoming guests are [a, b, e, c, d] and K = 1.



We can make [a, b] sit and make ‘e’ disappear. [a, b, c, d] is the lexicographically smallest sequence possible.



Marking Scheme: Total 15 points (5 + 5 + 5) (Algorithm explanation + Pseudo-code + runtime and data structure used)

Problem 3: In the country of WeirdLand, there is only one barber shop. In the shop there are 3 barbers, Alpha, Beta and Gamma, who are not always efficient as employees. All the customers are seated in a waiting area, each customer has a card which has a unique integral ID. For calling next person from the waiting area each barber has his own weird way:

Alpha: He will call either the person with least ID from the waiting area of the one with max ID. Min or Max is decided randomly by him (assume 50% probability for each min or max).
Beta: He will choose a 'k' and will call the kth smallest ID, (Kth ID when all IDs are arranged in ascending order). K is chosen randomly from the number of persons available in waiting area (1 <= K <= Total persons)
Gamma: He will always choose the median ID. If there are 'n' people in the waiting area, the median will be defined as (n+1)/2 th ID when all of them are arranged in ascending order. (For both odd and even 'n')

The Government has decided to digitise this shop and is asking for your suggestions on the data structure which should be ideal for this scenario.
Your task is to design a data structure which should support the query of each of the three barbers. The data structure should be highly efficient. You should properly explain which data structure you will use, what information will you store (and how), insertion and deletion of the IDs, query by each barber, and also discuss the time complexity of every operation (Insertion, deletion, and each query) (Write recurrence/ loop invariants and analyse worst case complexities).

Consider following Assumptions:
Each person will have a unique ID, IDs will be integral and can be represented using upto 10 digit integers.
The barbers will query sequentially, assuming no clash between two barbers choosing the same ID.
Once an ID is summoned by any barber, it should be removed from the waiting area.
Please come up with the most efficient solution using the data structures you have studied in class. More efficient solutions will fetch better marks. Avoid use of Inbuilt data structures.

Marking Scheme: Total 15 marks (10 + 5) (10 points for efficient implementation of all the required functions like, Insertion, Deletion, Queries and other Miscellaneous functions along with their time complexities + 5 points for correct data structure which will help to execute all the queries efficiently)

Problem 4:

There is a binary tree given to you. In which each of the node represents the student in the class. You are playing a game in which you need to team up with one of student in class. Team can be formed in such a way that in a team both of the students can't have same parent in tree and should be at the same level in binary tree.

Input: we have given you some pair of students you need to find that is the team valid or not.

Output: ‘Yes’ if the team is valid or ‘No’ if the team is invalid.

(a) Optimal solution of O(n) for checking the team validity.

(b) How many tree traversal is required for the solution of the above problem.

Marking Scheme: Total 15 marks-> 10 for part (a)(Algorithm explanation + Pseudo-code + runtime and data structure used)+ 5 for part (b)(explanation is required)

Problem 5:

Suppose there is a list s1->s2->s3->…….->s_{n-1}->$s_{n.}$ given to you. You need to modify this list in such a way that it will be s1->s_n->s2->s_{n-1}->s3->s_{n-2}…….

Note: algorithm should be inplace without modifying the values of the nodes.

Marking Scheme: Total 10 marks (Algorithm explanation + Pseudo-code + runtime and data structure used)

Problem 6:

x Number of players are participating in a sporting event.They are standing in a row.
Each player has some match value associated with them. Now each player wants to make one friend on their right side having greater match value as compared to him and it should be the nearest player as well to him.
Your task is to print for each player their friend’s index and his match value. If a player is not able to make such a friend print -1.

a) Find an Algorithm of Complexity O(n^2) to solve this problem.
b) Find an efficient way to compute the friends in O(n) Complexity.

Assume 1-based indexing.

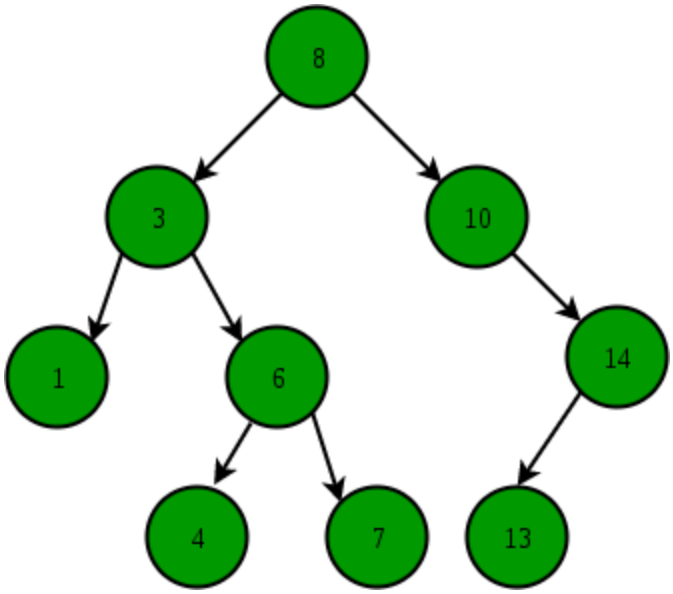
Eg 1:- for the input containing the match values for x = 9 players:-[2, 5, 9, 3, 1, 12, 6, 8, 7]
Output:-
i)Output Array (index of friend for each player):-[2, 3, 6, 6, 6, -1, 8, -1, -1]
ii) Output match value with index of friends
For player at index1 the nearest friend is at index 2 having match value 5
For player at index2 the nearest friend is at index 3 having match value 9
For player at index3 the nearest friend is at index 6 having match value 12
For player at index4 the nearest friend is at index 6 having match value 12
For player at index5 the nearest friend is at index 6 having match value 12
For player at index6 there is no nearest friend found
For player at index7 the nearest friend is at index 8 having match value 8
For player at index 8 there is no nearest friend found
For player at index9 there is no nearest friend found

Marking Scheme:-Total 15 marks Part a:- 5 marks ; Part b:-10 marks
Rubrics:- For part a:-(2.5+2.5) (Algorithm Explanation(2.5)+Pseudo code(2.5))
For part b:-(5+2.5+2.5)(Algorithm Explanation with data Structure used(5)+Pseudo Code(2.5)+Runtime Analysis(2.5))

Problem 7:-
Two monkeys Charlie and Leo are standing at two different nodes of a tree.
This tree is very weird, its root is situated at the highest point and grows in a downward direction.
Also it consists of several nodes and branches connecting two different nodes.But at max 2 branches can originate from a node.
Now ,there are some tasks which they have to perform using some algorithms.Since they are very lazy they want you to complete those tasks.
Task 1:- They want to move to the root node.So they want to calculate the individual distance each of them have to travel in reaching to the root.
Task2:-They want to meet with each other .So they want to compute the distance between each other before they start moving.
Can you help them in achieving these tasks?
Assumption:- Assume Tree as binary Tree.
You will be given the root Node of the tree and the reference nodes containing the address of the nodes where charlie and leo are present

Example of the tree:-

Input:-Here the root node is 8 and charlie is on node 1 and Leo is on node 7.
Output:- So the distance between Root to charlie:-2
For Task 1:-Distance between root to Leo:- 3
For Task 2:-Distance between leo and Charlie:- 3



Marking Scheme:- Total Marks 15 marks (Task 1:- 5marks; Task 2:-10 marks)
Rubrics:-For Task 1:-(2+2+1) (Algorithm Explanation(2)+Pseudo code(2)+Time Complexity(1))
For Task 2:-(4+4+2)(Algorithm Explanation(4)+Pseudo Code(4)+Runtime Analysis(2))