Question: https://leetcode.com/problems/maximum-number-of-events-that-can-be-attended/

So we sort events in increasing order by their starting time, and if same starting time for two events then sort them according to ending time.

As we all know a tree set data structure stores data like binary tree structure so we can get next available time slot for a event once we are done with a particular time.

So we loop through the sorted events and see if the starting time of the event is available or not. If not then we check the next possible time available from tree set, now if that time is in the range of event that means we can attend it.

Code:

class Solution {

public int maxEvents(int[][] events) {

TreeSet<Integer> set = new TreeSet();

for(int i=1;i<=100000;i++)

set.add(i);

//Fill the set with all the numbers from 1 to 100000(Max time limit for the events)

int count=0;

Arrays.sort(events,(a,b)-> a[1]-b[1]==0 ? a[0]-b[0] : a[1]-b[1]);

//Sort the array based on end time

for(int i=0;i<events.length;i++){

//As the array is sorted know e can follow a greedy approach and fill the start time if found

if(set.contains(events[i][0])){

//If we encounter the start time of the event in the list remove

//it from the set and increment the count

set.remove(events[i][0]);

count++;

}

else{

//If not find the just next higher element for the current event and

//check whether it lies in the range of start to end time

Integer high = set.higher(events[i][0]);

if(high!=null && high<=events[i][1]){

count++;//Increment the count

set.remove(high); // Remove it from the set

}

}

}

return count;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>