**Questions: 1**

In a shop, there are N counters. M people come to bill at the counters at a time given by ����[�]. Each person will greedily choose the counter with the least wait time; that is the number of people already present. If there is no person then he gets the billing immediately, else he stands in the queue.

For each person, print the time at which they leave the counter after successfully billing.

Notes

• It takes 1 unit of time for the counter to process a person's bill.

• The counter processes the next person immediately after the current person leaves.

• The times are given in increasing order of arrival at the counters. In formal terms ����[�]≤����[�+1].

Function description

Complete the function solve. This function takes the following 3 parameters and returns the required answer:

• N: Represents the number of counters

• M: Represents the number of persons

• time: Represents an array containing the entry time of the people

Input format for custom testing

Note: Use this input format if you are testing against custom input or writing code in a language where we don’t provide boilerplate code.

• The first line contains N denoting the number of counters.

• The second line contains M denoting the number of persons.

• The third line contains an array time, indicating the entry time of the people.

**Solution:**

static int[] solve(int n, int m, int[] t){

PriorityQueue<Integer> h = new PriorityQueue<>();

for (int i = 0; i < n; i++) {

h.add(1);

}

int[] ans = new int[t.length];

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

int avl = h.poll();

int k = Math.max(avl, t[i] + 1);

ans[i] = k;

h.add(k + 1);

}

return ans;

}

**Question:2**

Problem name

Train booking app

Problem statement

You are developing a train booking app. There are N trains that are labeled from 1 to N. Each train has a total capacity of 10000 seats. There are K bookings done. You are given an array of train bookings where booking[i] = [firsti , lasti ,seatsi ] represents a booking for train number firsti through lasti (both inclusive) with seatsi number of seats booked in each train on in the ith booking.

For example, booking[i]= [2, 5, 4] denotes that each of the trains with train number [2, 3, 4, 5] have 4 seats booked in the ith booking.

Note: Booking\_Length denotes the length of each booking which is 3 i.e [firsti , lasti ,seatsi].

A user wants to return an array capacity of length N, where capacity[i] is the total number of seats that are unbooked for train i. Determine the array.

Function description

Complete the Solve() function provided in the editor below that takes the following 4 arguments and returns an array capacity of length N, where capacity[i] is the total number of seats that are unbooked for train i:

• N: Represents the total number of trains

• K: Represents the total number of bookings

• Booking\_Length: Represents the length of each Booking (i.e 3)

• Booking: Represents an array of K bookings where booking[i] = [firsti , lasti ,seatsi ] represents a booking for train number firsti through lasti ( both inclusive) with seatsi number of seats booked in each train on the ith day

Input format for custom testing

Note: Use this input format if you are testing against custom input or writing code in a language where we don’t provide boilerplate code.

• The first line contains an integer N denoting the total number of trains.

• The second line contains an integer K denoting the total number of bookings.

• The third line contains an integer Booking\_Length denoting the length of each Booking (i.e. 3).

• The next K lines contain the bookings. Each line has 3 space-separated integers denoting booking[i] = [firsti , lasti ,seatsi ] represents a booking for train number firsti through lasti ( both inclusive) with seatsi number of seats booked in each train on ith day.

**Solution:**

static int[] solve(int N, int K, int Booking\_Length, int[][] Booking){

if (N > 100 || K > 100 || N < 1 || K < 1) {

throw new RuntimeException("Invalid Input Constraints");

}

int[] res = new int[N + 1];

for (int[] booking : Booking) {

int i = booking[0];

int j = booking[1];

K = booking[2];

res[i - 1] += K;

res[j] -= K;

}

for (int i = 1; i < N; i++) {

res[i] += res[i - 1];

}

int[] ans = new int[N];

for (int i = 0; i < N; i++) {

ans[i] = 10000 - res[i];

}

return ans;

}

**Alternative solution for above program:**

static int[] solve(int N, int K, int Booking\_Length, int[][] Booking){

// Write your code here

int[] result = new int[N];

Map<Integer, Integer> trainWithBookings = new LinkedHashMap<>();

int i=1;

while(i<=N){

trainWithBookings.put(i++, 0);

}

//for(int totalTrain = 0; totalTrain< N;totalTrain++)

for(int j=0;j<K;j++){

int start = Booking[j][0];

int end = Booking[j][1];

int bookingCount = Booking[j][2];

while(start <= end){

trainWithBookings.put(start, trainWithBookings.get(start) + bookingCount);

start++;

}

}

int l=0;

while(l<N){

result[l] = 10000 - trainWithBookings.get(l+1);

l++;

}

return result;

}