**[Meeting Rooms III](https://leetcode.com/problems/meeting-rooms-iii/)**

You are given an integer n. There are n rooms numbered from 0 to n - 1.

You are given a 2D integer array meetings where meetings[i] = [starti, endi] means that a meeting will be held during the **half-closed** time interval [starti, endi). All the values of starti are **unique**.

Meetings are allocated to rooms in the following manner:

1. Each meeting will take place in the unused room with the **lowest** number.
2. If there are no available rooms, the meeting will be delayed until a room becomes free. The delayed meeting should have the **same** duration as the original meeting.
3. When a room becomes unused, meetings that have an earlier original **start** time should be given the room.

Return*the****number****of the room that held the most meetings.*If there are multiple rooms, return*the room with the****lowest****number.*

A **half-closed interval** [a, b) is the interval between a and b **including** a and **not including** b.

**Example 1:**

**Input:** n = 2, meetings = [[0,10],[1,5],[2,7],[3,4]]

**Output:** 0

**Explanation:**

- At time 0, both rooms are not being used. The first meeting starts in room 0.

- At time 1, only room 1 is not being used. The second meeting starts in room 1.

- At time 2, both rooms are being used. The third meeting is delayed.

- At time 3, both rooms are being used. The fourth meeting is delayed.

- At time 5, the meeting in room 1 finishes. The third meeting starts in room 1 for the time period [5,10).

- At time 10, the meetings in both rooms finish. The fourth meeting starts in room 0 for the time period [10,11).

Both rooms 0 and 1 held 2 meetings, so we return 0.

**Example 2:**

**Input:** n = 3, meetings = [[1,20],[2,10],[3,5],[4,9],[6,8]]

**Output:** 1

**Explanation:**

- At time 1, all three rooms are not being used. The first meeting starts in room 0.

- At time 2, rooms 1 and 2 are not being used. The second meeting starts in room 1.

- At time 3, only room 2 is not being used. The third meeting starts in room 2.

- At time 4, all three rooms are being used. The fourth meeting is delayed.

- At time 5, the meeting in room 2 finishes. The fourth meeting starts in room 2 for the time period [5,10).

- At time 6, all three rooms are being used. The fifth meeting is delayed.

- At time 10, the meetings in rooms 1 and 2 finish. The fifth meeting starts in room 1 for the time period [10,12).

Room 0 held 1 meeting while rooms 1 and 2 each held 2 meetings, so we return 1.

**Constraints:**

* 1 <= n <= 100
* 1 <= meetings.length <= 105
* meetings[i].length == 2
* 0 <= starti < endi <= 5 \* 105
* All the values of starti are **unique**.

Code :

#define ll long long

#define pi pair<ll,ll>

class Solution {

public:

    int mostBooked(int n, vector<vector<int>>& A) {

        vector<int> roomcnt(n,0);

        set<int> s;

        priority\_queue<pi,vector<pi>,greater<pi>> q;

        sort(A.begin(),A.end());

        int m=A.size();

        // store available rooms

        for(int i=0;i<n;i++){

            s.insert(i);

        }

        for(int i=0;i<m;i++){

            ll start=A[i][0];

            ll end=A[i][1];

            // storing available room in set

            while(q.size()>0 && q.top().first<=start){

                int room=q.top().second;

                q.pop();

                s.insert(room);

            }

            // delaying the current meeting

            if(s.size()==0){

                pair<ll,ll> p=q.top();

                q.pop();

                ll dif=end-start;

                start=p.first;

                end=start+dif;

                s.insert(p.second);

            }

            // lowest number of unsed room available

            auto it=s.begin();

            roomcnt[\*it]++;

            // assign meeting to lowest avaible room

            q.push({end,\*it});

            s.erase(\*it);

        }

        int ans=0;

        int maxi=0;

        // find room with maximum meetings

        for(int i=0;i<n;i++){

            if(maxi<roomcnt[i]){

                maxi=roomcnt[i];

                ans=i;

            }

        }

        return ans;

    }

};

Link : <https://leetcode.com/problems/meeting-rooms-iii/?envType=daily-question&envId=2024-02-18>