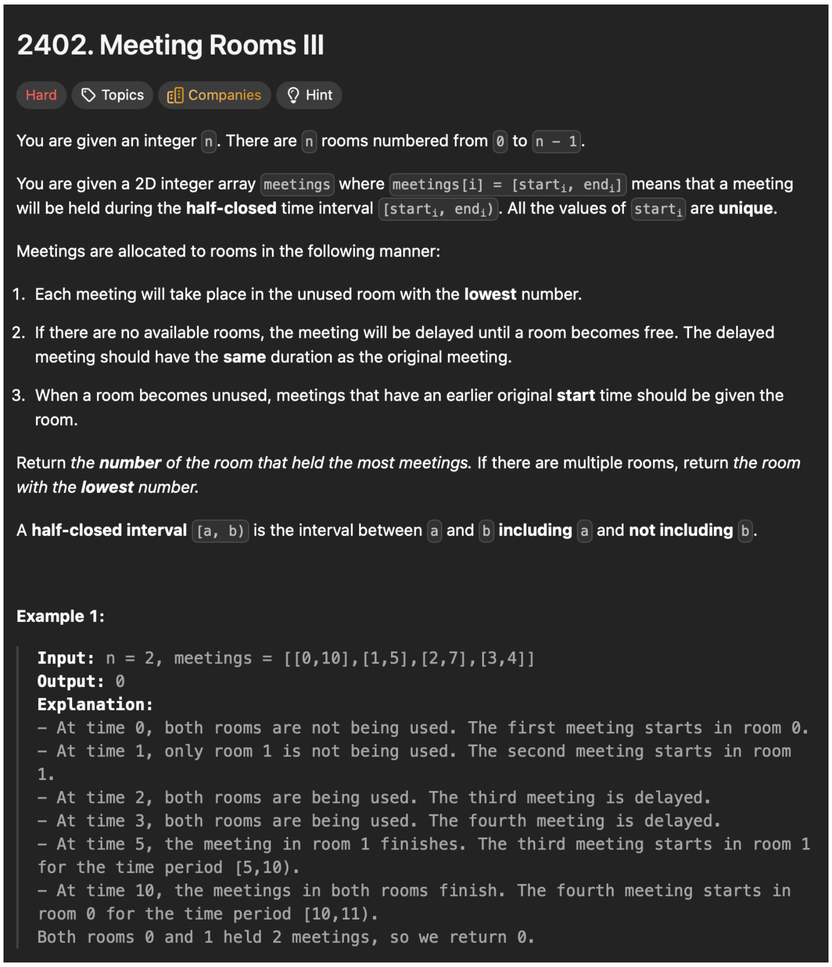


2402. Meeting Rooms III

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] = [start$_i$, end$_i$] means that a meeting will be held during the **half-closed** time interval [start$_i$, end$_i$). All the values of start$_i$ 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.
```

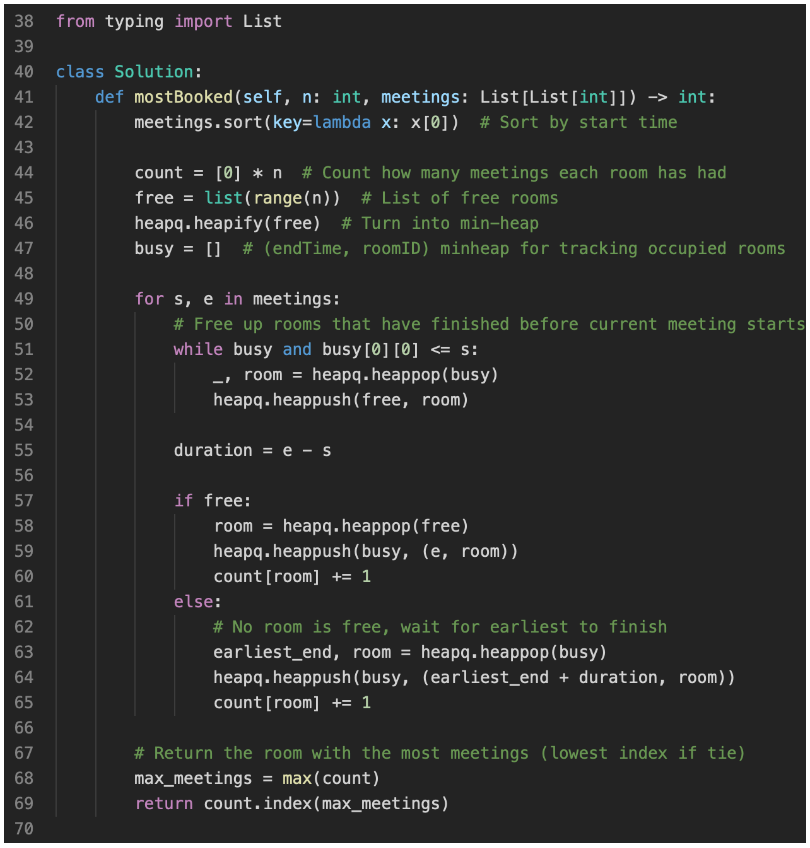
1. sort by start time

meetings.sort(key = lambda x: x[0])

2. minheap/priority queu

mh1: free rooms

mh2: end time of meetings happening.



```python
from typing import List

class Solution:
    def mostBooked(self, n: int, meetings: List[List[int]]) -> int:
        meetings.sort(key=lambda x: x[0])  # Sort by start time

        count = [0] * n  # Count how many meetings each room has had
        free = list(range(n))  # List of free rooms
        heapq.heapify(free)  # Turn into min-heap
        busy = []  # (endTime, roomID) minheap for tracking occupied rooms

        for s, e in meetings:
            # Free up rooms that have finished before current meeting starts
            while busy and busy[0][0] <= s:
                _, room = heapq.heappop(busy)
                heapq.heappush(free, room)

            duration = e - s

            if free:
                room = heapq.heappop(free)
                heapq.heappush(busy, (e, room))
                count[room] += 1
            else:
                # No room is free, wait for earliest to finish
                earliest_end, room = heapq.heappop(busy)
                heapq.heappush(busy, (earliest_end + duration, room))
                count[room] += 1

        # Return the room with the most meetings (lowest index if tie)
        max_meetings = max(count)
        return count.index(max_meetings)
```