5805. The Number of the Smallest Unoccupied Chair

My Submissions (/contest/biweekly-contest-57/problems/the-number-of-the-smallest-unoccupied-chair/submissions/)

Back to Contest (/contest/biweekly-contest-57/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

There is a party where `n` friends numbered from `0` to `n - 1` are attending. There is an **infinite** number of chairs in this party that are numbered from `0` to `infinity`. When a friend arrives at the party, they sit on the unoccupied chair with the **smallest number**.

- For example, if chairs `0`, `1`, and `5` are occupied when a friend comes, they will sit on chair number `2`.

When a friend leaves the party, their chair becomes unoccupied at the moment they leave. If another friend arrives at that same moment, they can sit in that chair.

You are given a **0-indexed** 2D integer array `times` where $\texttt{times[i]} = [\texttt{arrival}_i, \texttt{leaving}_i]$, indicating the arrival and leaving times of the $\texttt{i}^{\texttt{th}}$ friend respectively, and an integer `targetFriend`. All arrival times are **distinct**.

Return *the **chair number** that the friend numbered* `targetFriend` *will sit on*.

Example 1:

```
Input: times = [[1,4],[2,3],[4,6]], targetFriend = 1
Output: 1
Explanation:
- Friend 0 arrives at time 1 and sits on chair 0.
- Friend 1 arrives at time 2 and sits on chair 1.
- Friend 1 leaves at time 3 and chair 1 becomes empty.
- Friend 0 leaves at time 4 and chair 0 becomes empty.
- Friend 2 arrives at time 4 and sits on chair 0.
Since friend 1 sat on chair 1, we return 1.
```

Example 2:

```
Input: times = [[3,10],[1,5],[2,6]], targetFriend = 0
Output: 2
Explanation:
- Friend 1 arrives at time 1 and sits on chair 0.
- Friend 2 arrives at time 2 and sits on chair 1.
- Friend 0 arrives at time 3 and sits on chair 2.
- Friend 1 leaves at time 5 and chair 0 becomes empty.
- Friend 2 leaves at time 6 and chair 1 becomes empty.
- Friend 0 leaves at time 10 and chair 2 becomes empty.
Since friend 0 sat on chair 2, we return 2.
```

Constraints:

- `n == times.length`
- $\texttt{2 <= n <= 10}^4$
- `times[i].length == 2`
- $\texttt{1 <= arrival}_i \texttt{ < leaving}_i \texttt{ <= 10}^5$
- `0 <= targetFriend <= n - 1`
- Each $\texttt{arrival}_i$ time is **distinct**.

JavaScript

```
/**
 * @param {number[][]} times
 * @param {number} targetFriend
 * @return {number}
 */
var smallestChair = function(times, targetFriend) {

};
```

□ **Custom Testcase** Use Example Testcases

Run Submit

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) |

Privacy Policy (/privacy)

United States (/region)