5774. Process Tasks Using Servers

My Submissions (/contest/weekly-contest-243/problems/process-tasks-using-servers/submissions/)

Back to Contest (/contest/weekly-contest-243/)

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

You are given two **0-indexed** integer arrays `servers` and `tasks` of lengths `n` and `m` respectively. `servers[i]` is the **weight** of the i^{th} server, and `tasks[j]` is the **time needed** to process the j^{th} task **in seconds**.

You are running a simulation system that will shut down after all tasks are processed. Each server can only process one task at a time. You will be able to process the j^{th} task starting from the j^{th} second beginning with the 0^{th} task at second `0`. To process task `j`, you assign it to the server with the **smallest weight** that is free, and in case of a tie, choose the server with the **smallest index**. If a free server gets assigned task `j` at second `t`, it will be free again at the second `t + tasks[j]`.

If there are no free servers, you must wait until one is free and execute the free tasks **as soon as possible**. If **multiple** tasks need to be assigned, assign them in order of **increasing index**.

You may assign multiple tasks at the same second if there are multiple free servers.

Build an array `ans` of length `m`, where `ans[j]` is the **index** of the server the j^{th} task will be assigned to.

Return *the array* `ans`.

Example 1:

```
Input: servers = [3,3,2], tasks = [1,2,3,2,1,2]
Output: [2,2,0,2,1,2]
Explanation: Events in chronological order go as follows:
- At second 0, task 0 is added and processed using server 2 until second 1.
- At second 1, server 2 becomes free. Task 1 is added and processed using server 2 until second 3.
- At second 2, task 2 is added and processed using server 0 until second 5.
- At second 3, server 2 becomes free. Task 3 is added and processed using server 2 until second 5.
- At second 4, task 4 is added and processed using server 1 until second 5.
- At second 5, all servers become free. Task 5 is added and processed using server 2 until second 7.
```

Example 2:

```
Input: servers = [5,1,4,3,2], tasks = [2,1,2,4,5,2,1]
Output: [1,4,1,4,1,3,2]
Explanation: Events in chronological order go as follows:
- At second 0, task 0 is added and processed using server 1 until second 2.
- At second 1, task 1 is added and processed using server 4 until second 2.
- At second 2, servers 1 and 4 become free. Task 2 is added and processed using server 1 until second 4.
- At second 3, task 3 is added and processed using server 4 until second 7.
- At second 4, server 1 becomes free. Task 4 is added and processed using server 1 until second 9.
- At second 5, task 5 is added and processed using server 3 until second 7.
- At second 6, task 6 is added and processed using server 2 until second 7.
```

Constraints:

- `servers.length == n`
- `tasks.length == m`
- `1 <= n, m <= 2 * 10^5`
- `1 <= servers[i], tasks[j] <= 2 * 10^5`

Java

```java
class Solution {
    public int[] assignTasks(int[] servers, int[] tasks) {

    }
}
```

Custom Testcase | Use Example Testcases

Run | Submit