2349. Design a Number Container System

My Submissions (/contest/biweekly-contest-83/problems/design-a-number-container-system/submissions/) Back to Contest (/contest/biweekly-contest-83/)

Design a number container system that can do the following:

- **Insert** or **Replace** a number at the given index in the system.
- **Return** the smallest index for the given number in the system.

Implement the `NumberContainers` class:

- `NumberContainers()` Initializes the number container system.
- `void change(int index, int number)` Fills the container at `index` with the `number`. If there is already a number at that `index`, replace it.
- `int find(int number)` Returns the smallest index for the given `number`, or `-1` if there is no index that is filled by `number` in the system.

User Accepted:	6731
User Tried:	8685
Total Accepted:	6952
Total Submissions:	18142
Difficulty:	Medium

Example 1:

```
Input
["NumberContainers", "find", "change", "change", "change", "change", "find", "change", "find"]
[[], [10], [2, 10], [1, 10], [3, 10], [5, 10], [10], [1, 20], [10]]
Output
[null, -1, null, null, null, null, 1, null, 2]

Explanation
NumberContainers nc = new NumberContainers();
nc.find(10); // There is no index that is filled with number 10. Therefore, we return -1.
nc.change(2, 10); // Your container at index 2 will be filled with number 10.
nc.change(1, 10); // Your container at index 1 will be filled with number 10.
nc.change(3, 10); // Your container at index 3 will be filled with number 10.
nc.change(5, 10); // Your container at index 5 will be filled with number 10.
nc.find(10); // Number 10 is at the indices 1, 2, 3, and 5. Since the smallest index that is filled with 10 is 1, we return 1.
nc.change(1, 20); // Your container at index 1 will be filled with number 20. Note that index 1 was filled with 10 and then rep
nc.find(10); // Number 10 is at the indices 2, 3, and 5. The smallest index that is filled with 10 is 2. Therefore, we return 2
```

Constraints:

- $1 \le index, number \le 10^9$
- At most 10^5 calls will be made **in total** to `change` and `find`.

Discuss (https://leetcode.com/problems/design-a-number-container-system/discuss)

Java

```java
public class NumberContainers {
    Map<Integer, Integer> im;
    Map<Integer, TreeSet<Integer>> vm;

    public NumberContainers() {
        im = new HashMap<>();
        vm = new HashMap<>();
    }

    public void change(int index, int number) {
        // tr("change", index, number);
        vm.computeIfAbsent(number, x -> new TreeSet<>()).add(index);
        if (im.containsKey(index)) {
            int oldNumber = im.get(index);
            if (oldNumber != number) vm.get(oldNumber).remove(index);
        }
        im.put(index, number);
        // tr("value map", vm, "index map", im);
    }
```

```
20
21 ▾    public int find(int number) {
22          if (!vm.containsKey(number) || vm.get(number).size() == 0) return -1;
23          return vm.get(number).first();
24      }
25  }
```

☐ **Custom Testcase** Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/758689450/) More Details › (/submissions/detail/758689450/)

Share your acceptance!

3