

Problem 2349: Design a Number Container System

Problem Information

Difficulty: Medium

Acceptance Rate: 57.11%

Paid Only: No

Tags: Hash Table, Design, Heap (Priority Queue), Ordered Set

Problem Description

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.

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
replaced with 20. 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 <= index, number <= 109` * At most `105` calls will be made ****in total**** to `change` and `find`.

Code Snippets

C++:

```
class NumberContainers {
public:
    NumberContainers() {

    }

    void change(int index, int number) {

    }

    int find(int number) {

    }
};

/**
 * Your NumberContainers object will be instantiated and called as such:
 * NumberContainers* obj = new NumberContainers();
 * obj->change(index,number);
 * int param_2 = obj->find(number);
 */
```

Java:

```
class NumberContainers {

    public NumberContainers() {

    }

}
```

```

public void change(int index, int number) {

}

public int find(int number) {

}

}

/**
 * Your NumberContainers object will be instantiated and called as such:
 * NumberContainers obj = new NumberContainers();
 * obj.change(index,number);
 * int param_2 = obj.find(number);
 */

```

Python3:

```

class NumberContainers:

    def __init__(self):

    def change(self, index: int, number: int) -> None:

    def find(self, number: int) -> int:

    # Your NumberContainers object will be instantiated and called as such:
    # obj = NumberContainers()
    # obj.change(index,number)
    # param_2 = obj.find(number)

```