Description | Solution | Discuss (532) | Submissions

1429. First Unique Number

Medium 334 17 Add to List Share

You have a queue of integers, you need to retrieve the first unique integer in the queue.

Implement the `FirstUnique` class:

- `FirstUnique(int[] nums)` Initializes the object with the numbers in the queue.
- `int showFirstUnique()` returns the value of **the first unique** integer of the queue, and returns **-1** if there is no such integer.
- `void add(int value)` insert value to the queue.

Example 1:

```
Input:
["FirstUnique","showFirstUnique","add","showFirstUnique","add","showFirstUnique","ad

[[[2,3,5]],[],[5],[],[2],[],[3],[]]
Output:
[null,2,null,2,null,3,null,-1]
Explanation:
FirstUnique firstUnique = new FirstUnique([2,3,5]);
firstUnique.showFirstUnique(); // return 2
firstUnique.add(5);            // the queue is now [2,3,5,5]
firstUnique.showFirstUnique(); // return 2
firstUnique.add(2);            // the queue is now [2,3,5,5,2]
firstUnique.showFirstUnique(); // return 3
firstUnique.add(3);            // the queue is now [2,3,5,5,2,3]
firstUnique.showFirstUnique(); // return -1
```

Example 2:

```
Input:
["FirstUnique","showFirstUnique","add","add","add","add","add","showFirstUnique"]
[[[7,7,7,7,7,7]],[],[7],[3],[3],[7],[17],[]]
Output:
[null,-1,null,null,null,null,null,17]
Explanation:
FirstUnique firstUnique = new FirstUnique([7,7,7,7,7,7]);
firstUnique.showFirstUnique(); // return -1
firstUnique.add(7);            // the queue is now [7,7,7,7,7,7,7]
firstUnique.add(3);            // the queue is now [7,7,7,7,7,7,7,3]
firstUnique.add(3);            // the queue is now [7,7,7,7,7,7,7,3,3]
firstUnique.add(7);            // the queue is now [7,7,7,7,7,7,7,3,3,7]
firstUnique.add(17);           // the queue is now [7,7,7,7,7,7,7,3,3,7,17]
firstUnique.showFirstUnique(); // return 17
```

Example 3:

```
Input:
["FirstUnique","showFirstUnique","add","showFirstUnique"]
[[[809]],[],[809],[]]
Output:
[null,809,null,-1]
Explanation:
FirstUnique firstUnique = new FirstUnique([809]);
firstUnique.showFirstUnique(); // return 809
firstUnique.add(809);          // the queue is now [809,809]
firstUnique.showFirstUnique(); // return -1
```

Constraints:

- `1 <= nums.length <= 10^5`
- `1 <= nums[i] <= 10^8`
- `1 <= value <= 10^8`
- At most `50000` calls will be made to `showFirstUnique` and `add`.

Accepted 64,031 | Submissions 124,952

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Amazon | 6 Docusign | 4

Related Topics

Array Hash Table Design Queue Data Stream

Hide Hint 1

Use doubly Linked list with hashmap of pointers to linked list nodes. add unique number to the linked list. When add is called check if the added number is unique then it have to be added to the linked list and if it is repeated remove it from the linked list if exists. When showFirstUnique is called retrieve the head of the linked list.

Hide Hint 2

Use queue and check that first element of the queue is always unique.

Hide Hint 3

Use set or heap to make running time of each function O(logn).

Java | Autocomplete

```java
class FirstUnique {

  private Set<Integer> setQueue = new LinkedHashSet<>();
  private Map<Integer, Boolean> isUnique = new HashMap<>();

  public FirstUnique(int[] nums) {
    for (int num : nums) {
      this.add(num);
    }
  }

  public int showFirstUnique() {
    // If the queue contains values, we need to get the first one from it.
    // We can do this by making an iterator, and getting its first item.
    if (!setQueue.isEmpty()) {
      return setQueue.iterator().next();
    }
    return -1;
  }

  public void add(int value) {
    // Case 1: This value is not yet in the data structure.
    // It should be ADDED.
    if (!isUnique.containsKey(value)) {
      isUnique.put(value, true);
      setQueue.add(value);
    // Case 2: This value has been seen once, so is now becoming
    // non-unique. It should be REMOVED.
    } else if (isUnique.get(value)) {
      isUnique.put(value, false);
      setQueue.remove(value);
    }
  }
}
```