1429 First Unique Number

1429. First Unique Number

难度 中等 12

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","showFirstUniq
[[[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
```

执行用时： 336 ms ，在所有 C++ 提交中击败了 76.09% 的用户

内存消耗： 146.6 MB ，在所有 C++ 提交中击败了 19.57% 的用户

炫耀 一下：

```
1 /*
2   Author by guoguo
3
```

```cpp
  实际上这个题目 很类似 LRU,  本质上是 优化 add 的时间复杂度

  我的思路是维护一个 双向链表，越靠近头节点，意思越是第一个 unique number
  然后采用 Map 进行 映射删除,  如果发现之前有类似的，直接映射删除
*/
class Node{
public:
    int val;
    Node* next;
    Node* prev;
    Node(int v) : val(v), next(nullptr), prev(nullptr){}
};

class DoublyLinkedList{
public:
    Node* head;
    Node* tail;

    DoublyLinkedList(){
        head = new Node(0);
        tail = new Node(0);
        head->next = tail;
        tail->prev = head;
    }

    void addLast(Node* node){
        node->prev = tail->prev;
        node->next = tail;

        tail->prev->next = node;
        tail->prev = node;
    }

    void del(Node* node){
        if(node == nullptr)
            return;
        node->next->prev = node->prev;
        node->prev->next = node->next;
    }
};

class FirstUnique {
public:
    unordered_map<int, Node*> map;
    DoublyLinkedList dll;
    FirstUnique(vector<int>& nums) {
        for(int num : nums){
            add(num);
        }
```

```
    }

    int showFirstUnique() {
        if(dll.head->next == dll.tail)
            return -1;
        return dll.head->next->val;
    }

    void add(int value) {
        if(map.count(value) != 0){
            dll.del(map[value]);
            map[value] = nullptr;

            return;
        }

        Node* newNode = new Node(value);
        map.insert({value, newNode});
        dll.addLast(newNode);
    }
};


/**
 * Your FirstUnique object will be instantiated and called as such:
 * FirstUnique* obj = new FirstUnique(nums);
 * int param_1 = obj->showFirstUnique();
 * obj->add(value);
 */
```