380.O(1)时间插入、删除和获取随机元素

> 计算学部十大打卡活动——“龙舞编程新春会”编程打卡（2024-1-27）
>
> 力扣——380.O(1)时间插入、删除和获取随机元素

一、题目

- 实现 `RandomizedSet` 类：
  - `RandomizedSet()` 初始化 `RandomizedSet` 对象
  - `bool insert(int val)` 当元素 `val` 不存在时，向集合中插入该项，并返回 `true` ；否则，返回 `false` 。
  - `bool remove(int val)` 当元素 `val` 存在时，从集合中移除该项，并返回 `true` ；否则，返回 `false` 。
  - `int getRandom()` 随机返回现有集合中的一项（测试用例保证调用此方法时集合中至少存在一个元素）。每个元素应该有 **相同的概率** 被返回。

  你必须实现类的所有函数，并满足每个函数的 **平均** 时间复杂度为 `O(1)` 。

  示例：

  > 输入
  > ["RandomizedSet", "insert", "remove", "insert", "getRandom", "remove", "insert", "getRandom"]
  > [[], [1], [2], [2], [], [1], [2], []]
  > 输出
  > [null, true, false, true, 2, true, false, 2]
  >
  > 解释
  > RandomizedSet randomizedSet = new RandomizedSet();
  > randomizedSet.insert(1); // 向集合中插入 1 。返回 true 表示 1 被成功地插入。
  > randomizedSet.remove(2); // 返回 false ，表示集合中不存在 2 。
  > randomizedSet.insert(2); // 向集合中插入 2 。返回 true 。集合现在包含 [1,2] 。
  > randomizedSet.getRandom(); // getRandom 应随机返回 1 或 2 。
  > randomizedSet.remove(1); // 从集合中移除 1 ，返回 true 。集合现在包含 [2] 。
  > randomizedSet.insert(2); // 2 已在集合中，所以返回 false 。
  > randomizedSet.getRandom(); // 由于 2 是集合中唯一的数字，getRandom 总是返回 2 。

  提示：

  - $-2^{31} <= val <= 2^{31} - 1$
  - 最多调用 `insert` 、 `remove` 和 `getRandom` 函数 $2 * 10^5$ 次
  - 在调用 `getRandom` 方法时，数据结构中 **至少存在一个** 元素。

二、题解

根据类中函数描述进行编写：

```
class RandomizedSet {
public:
    //初始化对象
    RandomizedSet() {
        srand((unsigned)time(NULL));
```

```
    }

    //插入
    bool insert(int val) {
        if (indices.count(val)) {//如果对象已经在哈希表中，返回false
            return false;
        }
        int index = nums.size();
        nums.emplace_back(val);//在数组末尾插入
        indices[val] = index;//存入哈希表
        return true;
    }
    //删除
    bool remove(int val) {
        if (!indices.count(val)) {//如果对象不存在，无法删除
            return false;
        }
        int index = indices[val];
        int last = nums.back();
        nums[index] = last;
        indices[last] = index;
        nums.pop_back();
        indices.erase(val);
        return true;
    }
    //随机返回
    int getRandom() {
        int randomIndex = rand()%nums.size();
        return nums[randomIndex];
    }
private:
    vector<int> nums;
    unordered_map<int, int> indices;
};
```

复杂度分析

时间复杂度：$O(1)$

空间复杂度：$O(n)$