Problem 380: Insert Delete GetRandom O(1)

Problem Information

Difficulty: Medium
Acceptance Rate: 55.16%
Paid Only: No
Tags: Array, Hash Table, Math, Design, Randomized

Problem Description

Implement the `RandomizedSet` class:

* `RandomizedSet()` Initializes the `RandomizedSet` object. * `bool insert(int val)` Inserts an item `val` into the set if not present. Returns `true` if the item was not present, `false` otherwise. * `bool remove(int val)` Removes an item `val` from the set if present. Returns `true` if the item was present, `false` otherwise. * `int getRandom()` Returns a random element from the current set of elements (it's guaranteed that at least one element exists when this method is called). Each element must have the **same probability** of being returned.

You must implement the functions of the class such that each function works in **average** `O(1)` time complexity.

Example 1:

Input ["RandomizedSet", "insert", "remove", "insert", "getRandom", "remove", "insert", "getRandom"] [[], [1], [2], [2], [], [1], [2], []] **Output** [null, true, false, true, 2, true, false, 2] **Explanation** RandomizedSet randomizedSet = new RandomizedSet(); randomizedSet.insert(1); // Inserts 1 to the set. Returns true as 1 was inserted successfully. randomizedSet.remove(2); // Returns false as 2 does not exist in the set. randomizedSet.insert(2); // Inserts 2 to the set, returns true. Set now contains [1,2]. randomizedSet.getRandom(); // getRandom() should return either 1 or 2 randomly. randomizedSet.remove(1); // Removes 1 from the set, returns true. Set now contains [2]. randomizedSet.insert(2); // 2 was already in the set, so return false. randomizedSet.getRandom(); // Since 2 is the only number in the set, getRandom() will always return 2.

Constraints:

* `-231 <= val <= 231 - 1` * At most `2 * ``105` calls will be made to `insert`, `remove`, and `getRandom`. * There will be **at least one** element in the data structure when `getRandom` is called.

Code Snippets

C++:

```
class RandomizedSet {
public:
RandomizedSet() {

}

bool insert(int val) {

}

bool remove(int val) {

}

int getRandom() {

}
};

/**
* Your RandomizedSet object will be instantiated and called as such:
* RandomizedSet* obj = new RandomizedSet();
* bool param_1 = obj->insert(val);
* bool param_2 = obj->remove(val);
* int param_3 = obj->getRandom();
*/
```

Java:

```
class RandomizedSet {
```

```
public RandomizedSet() {

}

public boolean insert(int val) {

}

public boolean remove(int val) {

}

public int getRandom() {

}
}

/**
* Your RandomizedSet object will be instantiated and called as such:
* RandomizedSet obj = new RandomizedSet();
* boolean param_1 = obj.insert(val);
* boolean param_2 = obj.remove(val);
* int param_3 = obj.getRandom();
*/
```

Python3:

```
class RandomizedSet:

def __init__(self):


def insert(self, val: int) -> bool:


def remove(self, val: int) -> bool:


def getRandom(self) -> int:
```

```

# Your RandomizedSet object will be instantiated and called as such:
# obj = RandomizedSet()
# param_1 = obj.insert(val)
# param_2 = obj.remove(val)
# param_3 = obj.getRandom()
```