381. Insert Delete GetRandom O(1) - Duplicates allowed

Description

`RandomizedCollection` is a data structure that contains a collection of numbers, possibly duplicates (i.e., a multiset). It should support inserting and removing specific elements and also reporting a random element.

Implement the `RandomizedCollection` class:

- `RandomizedCollection()` Initializes the empty `RandomizedCollection` object.
- `bool insert(int val)` Inserts an item `val` into the multiset, even if the item is already present. Returns `true` if the item is not present, `false` otherwise.
- `bool remove(int val)` Removes an item `val` from the multiset if present. Returns `true` if the item is present, `false` otherwise. Note that if `val` has multiple occurrences in the multiset, we only remove one of them.
- `int getRandom()` Returns a random element from the current multiset of elements. The probability of each element being returned is **linearly related** to the number of the same values the multiset contains.

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

Note: The test cases are generated such that `getRandom` will only be called if there is **at least one** item in the `RandomizedCollection`.

Example 1:

```
Input
["RandomizedCollection", "insert", "insert", "insert", "getRandom", "remove", "getRandom"]
[[], [1], [1], [2], [], [1], []]
Output
[null, true, false, true, 2, true, 1]

Explanation
RandomizedCollection randomizedCollection = new RandomizedCollection();
randomizedCollection.insert(1);   // return true since the collection does not contain 1.
                                  // Inserts 1 into the collection.
randomizedCollection.insert(1);   // return false since the collection contains 1.
                                  // Inserts another 1 into the collection. Collection now contains [1,1].
randomizedCollection.insert(2);   // return true since the collection does not contain 2.
                                  // Inserts 2 into the collection. Collection now contains [1,1,2].
randomizedCollection.getRandom(); // getRandom should:
                                  // - return 1 with probability 2/3, or
                                  // - return 2 with probability 1/3.
randomizedCollection.remove(1);   // return true since the collection contains 1.
                                  // Removes 1 from the collection. Collection now contains [1,2].
randomizedCollection.getRandom(); // getRandom should return 1 or 2, both equally likely.
```

Constraints:

- $-2^{31} <= val <= 2^{31} - 1$
- At most $2 * 10^5$ calls **in total** will be made to `insert`, `remove`, and `getRandom`.
- There will be **at least one** element in the data structure when `getRandom` is called.