Question: https://leetcode.com/problems/insert-delete-getrandom-o1/

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

So we need to perfor insertion, deletion and getrandom in constant time complexity.

So let’s explore data structures which can help us achieve this.

We can achieve insertion and deletion in O(1) complexity using HashMap, but we can’t achieve getRandom in O(1) complexity using HashMap.

Similarly, we can achieve insertion and getRandom in O(1) complexity using Array, but we can’t achieve deletion in O(1) complexity using Array.

So what we can do is maintain both Array and HashMap.

Since the Array needs a specific fixed length for memory consumption, so we will use ArrayList.

* **Insertion**-If the element exists in HashMap return false, else insert both in ArrayList and HashMap.
* **Deletion**-If the element doesnot exist in HashMap then return false else perform remove() in HashMap and swap that element with the last element in arrayList and remove the last element from the ArrayList.
* **GetRandom**-Get a random position using rand() and return ar[rand].

Code:  
public class RandomizedSet {

ArrayList<Integer> nums;

HashMap<Integer, Integer> locs;

java.util.Random rand = new java.util.Random();

/\*\* Initialize your data structure here. \*/

public RandomizedSet() {

nums = new ArrayList<Integer>();

locs = new HashMap<Integer, Integer>();

}

/\*\* Inserts a value to the set. Returns true if the set did not already contain the specified element. \*/

public boolean insert(int val) {

boolean contain = locs.containsKey(val);

if ( contain ) return false;

locs.put( val, nums.size());

nums.add(val);

return true;

}

/\*\* Removes a value from the set. Returns true if the set contained the specified element. \*/

public boolean remove(int val) {

boolean contain = locs.containsKey(val);

if ( ! contain ) return false;

int loc = locs.get(val);

if (loc < nums.size() - 1 ) { // not the last one than swap the last one with this val

int lastone = nums.get(nums.size() - 1 );

nums.set( loc , lastone );

locs.put(lastone, loc);

}

locs.remove(val);

nums.remove(nums.size() - 1);

return true;

}

/\*\* Get a random element from the set. \*/

public int getRandom() {

return nums.get( rand.nextInt(nums.size()) );

}

}

Github Link :<https://lnkd.in/ecwtJeaz>