lenchen.medium.com

LeetCode #380 Insert Delete GetRandom O(1) - Len Chen - Medium

Len Chen

2-3 minutes

Medium

Problem

Design a data structure that supports all following operations in *average* **O(1)** time.

1. `insert(val)`: Inserts an item val to the set if not already present.
2. `remove(val)`: Removes an item val from the set if present.
3. `getRandom`: Returns a random element from current set of elements. Each element must have the **same probability** of being returned.

Example:

// Init an empty set.
RandomizedSet randomSet = new RandomizedSet();// Inserts 1 to the set. Returns true as 1 was inserted successfully.
randomSet.insert(1);// Returns false as 2 does not exist in the set.
randomSet.remove(2);// Inserts 2 to the set, returns true. Set now contains [1,2].

randomSet.insert(2);// getRandom should return either 1 or 2 randomly.
randomSet.getRandom();// Removes 1 from the set, returns true. Set now contains [2].
randomSet.remove(1);// 2 was already in the set, so return false.
randomSet.insert(2);// Since 2 is the only number in the set, getRandom always return 2.
randomSet.getRandom();

Solution

Actually it's impossible to do insertion and deletion both in O(1) time. However, in this data structure, it only needs to support one more `getRandom` operation, which means the order or insert and delete is not important. Therefore we have a bypass strategy to make it happen.

Use a list to store all insert `val` and use a hash map to store indices of all `val`. While doing insert, do ordinary insertion to the list and save its corresponding index into hash map. It takes O(1) time obviously.

Once there is a deletion operation coming, we put the last element of the list to the index of the `val` which is going to be removed, update the index of original last element and erase the last one in list. By doing so, deletion can also be done in constant time.

Complexity

As problem describes, it takes ***O(1)* time averagely**. And it needs ***O(n)* extra space** for hash map and list, where `n` is the most element may be inserted.