Insert Delete GetRandom O(1)

Try to solve the Insert Delete GetRandom O(1) problem.

We'll cover the following

Statement

Implement a **Random Set** data structure that can perform the following operations:

- **Constructor()**: This initializes the **Random Set** object.
- **Insert()**: This function takes an integer, **data**, as its parameter and, if it does not already exist in the set, add it to the set, returning TRUE. If the integer already exists in the set, the function returns FALSE.
- **Delete()**: This function takes an integer, **data**, as its parameter and, if it exists in the set, removes it, returning TRUE. If the integer does not exist in the set, the function returns FALSE.
- **GetRandom()**: This function takes no parameters. It returns an integer chosen at random from the set.

> **Note:** Your implementation should aim to have a running time of $O(1)$ (on average) for each operation.

Constraints:

- $-2^{31} \leq \textbf{data} \leq 2^{31}$
- No more than 2×10^5 calls will be made to the **Insert()**, **Delete()** and **GetRandom()** functions.
- There will be at least one element in the data structure when the **GetRandom()** function is called.

Examples

Understand the problem

Let's take a moment to make sure you've correctly understood the problem. The quiz below helps you check if you're solving the correct problem:

Insert Delete GetRandom O(1)

1 What is the output for the following series of commands?

```
Delete(20), Insert(20), Insert(20), GetRandom()
```

A) FALSE, TRUE, TRUE, 20

B) FALSE, TRUE, FALSE, 20

C) TRUE, TRUE, TRUE, 20

D) FALSE, FALSE, TRUE, 20

Submit Answer

Question 1 of 3
0 attempted

Reset Quiz

Figure it out!

We have a game for you to play. Rearrange the logical building blocks required to implement the **Delete** operation.

Drag and drop the cards to rearrange them in the correct sequence.

We store our data in an array and use a hash map to track the location at which each data element is stored in the array.

We use the hash map to look up the location of the element

to delete.

Swap the last element in the array with the one to be deleted.

In the hash map, update the location of the element we just moved.

Delete the key-value pair of the target data element from the hash map and then delete the target element from our array.

Reset | Show Solution | Submit

Try it yourself

Implement your solution in the following coding playground:

Java

usercode > RandomSet.java

```java
import java.util.*;
class RandomSet{

    /** Initialize your data structure here. */
    public RandomSet() {
        // Write your code here
    }

    /** Inserts a value to the dataset. Returns true if the dataset did not already contain the specified
    public boolean insert(int val) {
        // write your code here

    /** Deletes a value from the dataset. Returns true if the dataset contained the specified value. */
    public boolean delete(int val) {
        // Write your code here
        return true;
    }

    /** Get a random value from the dataset. */
    public int getRandomData() {
        // Write your code here
        return -1;
    }

}
```

Powered by AI | Submit

Test Cases | Results

Case 1 | Case 2 | Case 3

Input #1

["RandomSet","Insert","GetRandom"]

Input #2

[[],[25],[]]

Insert Delete GetRandom O(1)

Back

Solution: Implement L...

Next

Solution: Insert Delete...

Mark as Completed