App Academy

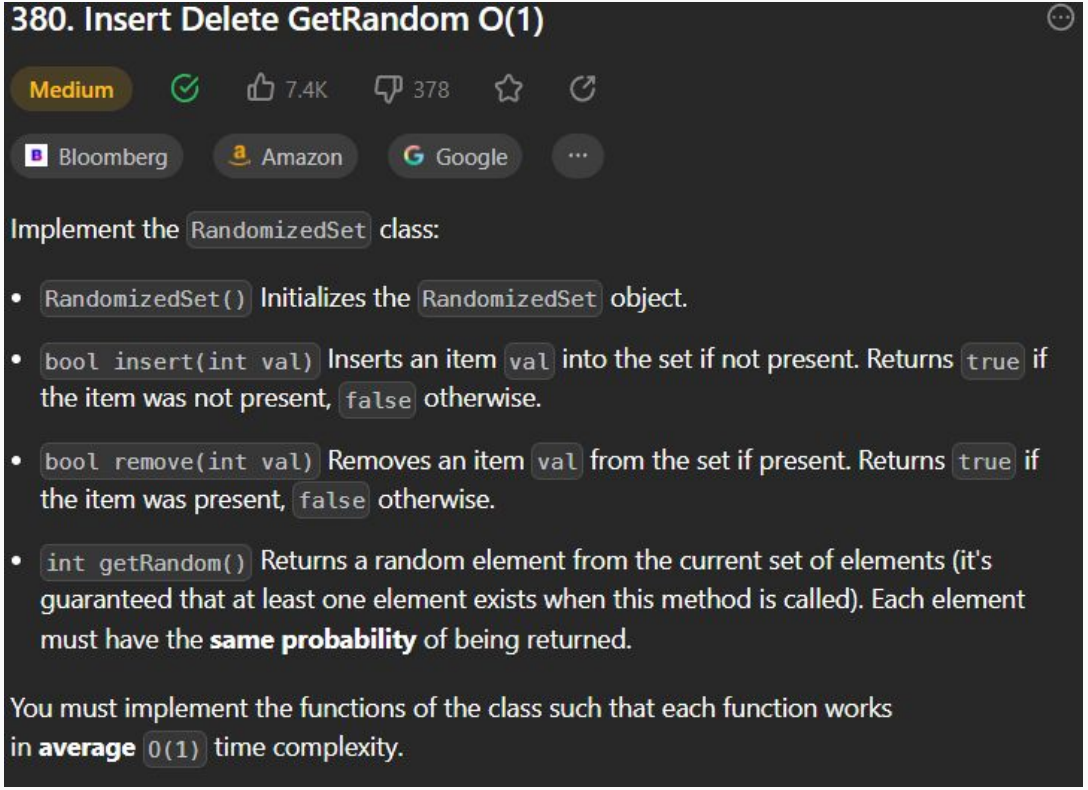
DS&A Interviewing Tactics
+ Design Problems

Succeeding in a DS&A interview

Six Steps (in four categories):

- Understand
  - 1. Ask clarifying questions (Do you understand the prompt?)
  - 2. Setup Sample I/O's/Identify edge cases
- Plan
  - 3. Identify approach(es) (What DSs combine to solve problem? Complexities?)
  - 4. Pseudocode! (This is the most important step)
- Code
  - 5. Implement your solution (CODE)
- Test and Analyze
  - 6. Test your I/O's and revisit Big-O Notation

Let's do a mock interview



380. Insert Delete GetRandom O(1)

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.

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

Understand the Problem

Clarification questions

- Can we assume that all values will be valid integers?
- Will there be more insertions, removals, or getRandoms done?
- How many calls can we expect to be made?
- Where are each of our parameters defined?
  - insert, remove, getRandom
- May I use the random.choice method provided in Python?

Time to plan

What data structure would be useful here?

- Let’s consider…
  - We need a way to store values
  - We need a way to find these values in O(1) time
  - We must also be able to remove any particular value in O(1) time
  - We must be able to randomly select a number from our set of numbers
- Potential data structures that could be useful…
  - set - allows us to insert and remove elements in O(1) time, disregards duplicates
  - array - allows us to index into any value in O(1) time
  - hash map - allows us to keep track of indices of values we insert into arrays

Can we just use a set?

- This sounds like the most intuitive thought!
- Using a set would make our insert and remove functions trivial
- But what about the getRandom function…?
    - We don't have a way to randomly select from a set, but we do have a way to randomly select from a list.

But how can we remove in O(1) with a list?

- What operations does an array / list have that run in O(1)?
  - lookup
  - push and pop
- Can we take advantage of the fact that popping removes an element in O(1)?
  - One can remember how heaps work! We can use the same idea here!
  - By swapping an element with the last element, we can remove the first element in O(1) regardless of initial location while keeping the second element

Let’s make sure we check all reqs

- insert
  - We can insert into our list of elements
  - O(1)
- remove
  - With our hash map, we can locate any elements in our list and remove them efficiently
  - O(1)
- getRandom
  - random.choice will have randomly select an number from our list
  - O(1)

Let’s code it out

Code it out

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

Walkthrough and Analyze

Walk through examples and revisit big O

- Potential follow ups:
    - What would happen if we allowed duplicates to be inserted?
- Big O
    - Was our original analysis consistent with what we end up with?
- Alternative solutions
    - Are there any alternative ways to solve the problem that you can think of?
    - Trade-offs?

Questions?

Let’s Practice!

- Review
  - LRU Cache