Problem 384: Shuffle an Array

Problem Information

Difficulty: Medium
Acceptance Rate: 59.42%
Paid Only: No
Tags: Array, Math, Design, Randomized

Problem Description

Given an integer array `nums`, design an algorithm to randomly shuffle the array. All permutations of the array should be **equally likely** as a result of the shuffling.

Implement the `Solution` class:

* `Solution(int[] nums)` Initializes the object with the integer array `nums`. * `int[] reset()` Resets the array to its original configuration and returns it. * `int[] shuffle()` Returns a random shuffling of the array.

Example 1:

Input ["Solution", "shuffle", "reset", "shuffle"] [[[1, 2, 3]], [], [], []] **Output** [null, [3, 1, 2], [1, 2, 3], [1, 3, 2]] **Explanation** Solution solution = new Solution([1, 2, 3]); solution.shuffle(); // Shuffle the array [1,2,3] and return its result. // Any permutation of [1,2,3] must be equally likely to be returned. // Example: return [3, 1, 2] solution.reset(); // Resets the array back to its original configuration [1,2,3]. Return [1, 2, 3] solution.shuffle(); // Returns the random shuffling of array [1,2,3]. Example: return [1, 3, 2]

Constraints:

* `1 <= nums.length <= 50` * `-106 <= nums[i] <= 106` * All the elements of `nums` are **unique**. * At most `104` calls **in total** will be made to `reset` and `shuffle`.

Code Snippets

C++:

```
class Solution {
public:
Solution(vector<int>& nums) {

}

vector<int> reset() {

}

vector<int> shuffle() {

}
};

/**
* Your Solution object will be instantiated and called as such:
* Solution* obj = new Solution(nums);
* vector<int> param_1 = obj->reset();
* vector<int> param_2 = obj->shuffle();
*/
```

Java:

```
class Solution {

public Solution(int[] nums) {

}

public int[] reset() {

}

public int[] shuffle() {

}
}

/**
* Your Solution object will be instantiated and called as such:
```

```
* Solution obj = new Solution(nums);
* int[] param_1 = obj.reset();
* int[] param_2 = obj.shuffle();
*/
```

Python3:

```
class Solution:

def __init__(self, nums: List[int]):


def reset(self) -> List[int]:


def shuffle(self) -> List[int]:



# Your Solution object will be instantiated and called as such:
# obj = Solution(nums)
# param_1 = obj.reset()
# param_2 = obj.shuffle()
```