

384. Shuffle an Array

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:

- $1 \leq \text{nums.length} \leq 50$
- $-10^6 \leq \text{nums}[i] \leq 10^6$
- All the elements of `nums` are **unique**.
- At most 10^4 calls **in total** will be made to `reset` and `shuffle`.

