Find all Duplicate Numbers (easy)

We'll cover the following

Problem Statement

We are given an unsorted array containing 'n' numbers taken from the range 1 to 'n'. The array has some duplicates, **find all the duplicate numbers without using any extra space**.

Example 1:

```
Input: [3, 4, 4, 5, 5]
Output: [4, 5]
```

Example 2:

```
Input: [5, 4, 7, 2, 3, 5, 3]
Output: [3, 5]
```

Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
using namespace std;

#include <iostream>
#include <vector>

class FindAllDuplicate {
 public:
  static vector<int> findNumbers(vector<int> &nums) {
    vector<int> duplicateNumbers;
    // TODO: Write your code here
    return duplicateNumbers;
  }
};
```

TEST SAVE RESET

Solution

This problem follows the **Cyclic Sort** pattern and shares similarities with Find the Duplicate Number. Following a similar approach, we will place each number at its correct index. After that, we will iterate through the array to find all numbers that are not at the correct indices. All these numbers are duplicates.

Code

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```
using namespace std;

#include <iostream>
#include <vector>

class FindAllDuplicate {
 public:
  static vector<int> findNumbers(vector<int> &nums) {
    int i = 0;
    while (i < nums.size()) {
      if (nums[i] != nums[nums[i] - 1]) {
        swap(nums, i, nums[i] - 1);
      } else {
        i++;
      }
    }

    vector<int> duplicateNumbers;
    for (i = 0; i < nums.size(); i++) {
      if (nums[i] != i + 1) {
        duplicateNumbers.push_back(nums[i]);
      }
    }
```

```
24
25      return duplicateNumbers;
26    }
27
28  private:
```

RUN SAVE RESET

Close

Output 0.833s

```
Duplicates are: 5 4
Duplicates are: 3 5
```

Time complexity

The time complexity of the above algorithm is $O(n)$.

Space complexity

Ignoring the space required for storing the duplicates, the algorithm runs in constant space $O(1)$.