# 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 numbers appearing twice, **find all these 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++

```python
def find_all_duplicates(nums):
  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

```python
def find_all_duplicates(nums):
  i = 0
  while i < len(nums):
    j = nums[i] - 1
```

```
    if nums[i] != nums[j]:
      nums[i], nums[j] = nums[j], nums[i]  # swap
    else:
      i += 1

  duplicateNumbers = []
  for i in range(len(nums)):
    if nums[i] != i + 1:
      duplicateNumbers.append(nums[i])

  return duplicateNumbers


def main():
  print(find_all_duplicates([3, 4, 4, 5, 5]))
  print(find_all_duplicates([5, 4, 7, 2, 3, 5, 3]))


main()

```

Run Save Reset

## 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 $O(1)$.

Back
Find the Duplicate Number (easy)

Next
Problem Challenge 1

Completed