CYCLIC SORT

C++ Assignments | Cyclic sort | Week 13

1.What is the worst case time complexity of cycle sort?

a) O(n)

b) O(log n)

c) O(n log n)

d) O(n*n)

ANSWER :_ D

2.You have a set of integers s , which originally contains all the numbers from 1 to n .
Unfortunately, due to some error, one of the numbers in s got duplicated to another number
in the set, which results in repetition of one number and loss of another number.
You are given an integer array nums representing the data status of this set after the error.
Find the number that occurs twice and the number that is missing and return them in the form
of an array. [Leetcode 645]

Example 1:
Input: nums = [1,2,2,4]
Output: [2,3]

Example 2:
Input: nums = [1,1]
Output: [1,2]

```
#include <iostream>
#include <vector>
using namespace std;

vector<int> findErrorNums(vector<int>& nums) {
    int n = nums.size();
    int duplicate = -1, missing = -1;

    // Iterate through the array
    for (int num : nums) {
        int index = abs(num) - 1;
        if (nums[index] < 0) {
            // If the current number is negative, it means
index+1 has been seen before (duplicate)
            duplicate = abs(num);
        } else {
            // Mark index+1 as visited by making nums[index]
negative
            nums[index] = -nums[index];
        }
    }

    // Find the missing number
    for (int i = 0; i < n; ++i) {
        if (nums[i] > 0) {
            missing = i + 1;
            break;
        }
    }

    return {duplicate, missing};
}

int main() {
    vector<int> nums1 = {1, 2, 2, 4};
    vector<int> result1 = findErrorNums(nums1);
    cout << "Input: [1, 2, 2, 4]" << endl;
    cout << "Output: [" << result1[0] << ", " << result1[1]
<< "]" << endl;

    vector<int> nums2 = {1, 1};
    vector<int> result2 = findErrorNums(nums2);
    cout << "\nInput: [1, 1]" << endl;
    cout << "Output: [" << result2[0] << ", " << result2[1]
<< "]" << endl;

    return 0;
}
```

3.Given an integer array nums of length n where all the integers of nums are in the range [1,
n] and each integer appears once or twice, return an array of all the integers that appears
twice.
You must write an algorithm that runs in O(n) time and uses only constant extra space.

[Leetcode 442]

Example 1:
Input: nums = [4,3,2,7,8,2,3,1]
Output: [2,3]

```
Example 2:
Input: nums = [1,1,2]
Output: [1]

#include <iostream>
#include <vector>
using namespace std;

vector<int> findDuplicates(vector<int>& nums) {
    vector<int> result;

    // Iterate through the array
    for (int num : nums) {
        int index = abs(num) - 1;
        if (nums[index] < 0) {
            // If the current number is negative, it means
index+1 has been seen before (duplicate)
            result.push_back(abs(num));
        } else {
            // Mark index+1 as visited by making nums[index]
negative
            nums[index] = -nums[index];
        }
    }

    return result;
}

int main() {
    vector<int> nums1 = {4, 3, 2, 7, 8, 2, 3, 1};
    vector<int> result1 = findDuplicates(nums1);
    cout << "Input: [4, 3, 2, 7, 8, 2, 3, 1]" << endl;
    cout << "Output: [";
    for (int num : result1) {
        cout << num << " ";
    }
    cout << "]" << endl;

    vector<int> nums2 = {1, 1, 2};
    vector<int> result2 = findDuplicates(nums2);
    cout << "\nInput: [1, 1, 2]" << endl;
    cout << "Output: [";
    for (int num : result2) {
        cout << num << " ";
    }
    cout << "]" << endl;

    return 0;
}
```