

## Experiment No. - 4

**Student Name:** Vivek Kumar

**Branch:** BE-CSE(LEET)

**Semester:** 6<sup>th</sup>

**Subject Name:** Competitive coding - II

**UID:** 21BCS8129

**Section/Group:** 20BCS-ST-801/B

**Date of Performance:** 07/03/2023

**Subject Code:** 20CSP-351

### 1. Aim/Overview of the practical:

#### Q.1 Longest Duplicate Substring.

<https://leetcode.com/problems/longest-duplicate-substring/>

### 2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

### 3. Objective:

- To understand the concept of B Search
- To implement the concept of Rabin Karp.

### 4. Code:

```
class Solution {
public:
    string longestDupSubstring(string S) {
        ans = "";
        power = vector<int>(S.length(), 1);
        int i;
        for (i = 1 ; i < S.length(); i++) {
            power[i] = (power[i - 1] * 26) % prime;
        }
        int low = 0, high = S.length();
        while (low <= high) {
            int mid = low + (high - low) / 2;
            string tmp = validate(mid, S);
            if (tmp.length() == 0) {
                high = mid - 1;
            } else {
                if (tmp.length() > ans.length()) {
                    ans = tmp;
                }
                low = mid + 1;
            }
        }

        return ans;
    }
};
```

private:

int prime = 19260817;

string ans;

Submitted By: Vivek Kumar

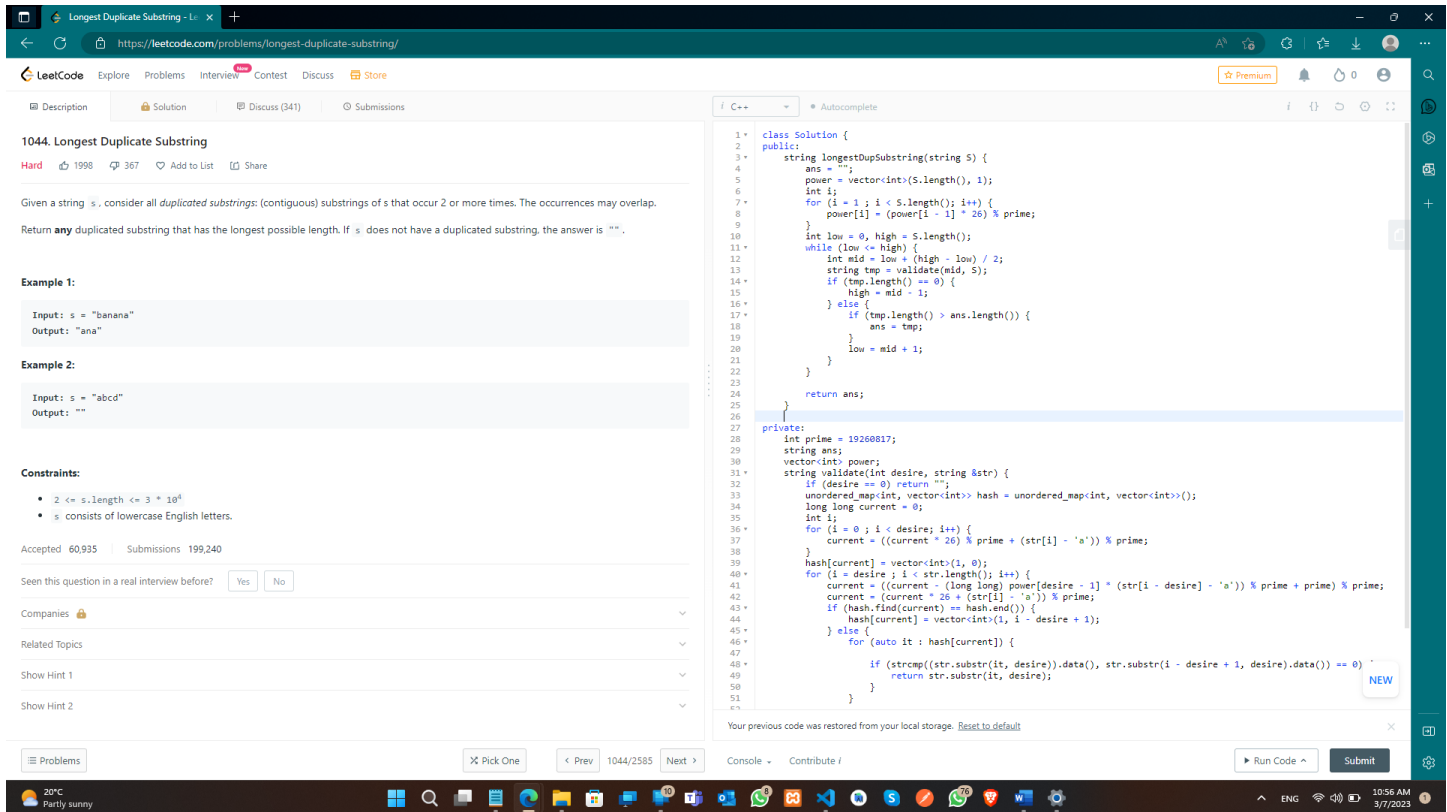
```
vector<int> power;
string validate(int desire, string &str) {
    if (desire == 0) return "";
    unordered_map<int, vector<int>> hash = unordered_map<int, vector<int>>();
    long long current = 0;
    int i;
    for (i = 0 ; i < desire; i++) {
        current = ((current * 26) % prime + (str[i] - 'a')) % prime;
    }
    hash[current] = vector<int>(1, 0);
    for (i = desire ; i < str.length(); i++) {
        current = ((current - (long long) power[desire - 1] * (str[i - desire] - 'a')) % prime + prime) %
prime;
        current = (current * 26 + (str[i] - 'a')) % prime;
        if (hash.find(current) == hash.end()) {
            hash[current] = vector<int>(1, i - desire + 1);
        } else {
            for (auto it : hash[current]) {

                if (strcmp((str.substr(it, desire)).data(), str.substr(i - desire + 1, desire).data()) == 0) {
                    return str.substr(it, desire);
                }
            }

            hash[current].push_back(i - desire + 1);
        }
    }

    return "";
}
};
```

## 5. Result/Output/Writing Summary:



**1044. Longest Duplicate Substring**

Hard 1998 367 Add to List Share

Given a string `s`, consider all *duplicated substrings*: (contiguous) substrings of `s` that occur 2 or more times. The occurrences may overlap.

Return **any** duplicated substring that has the longest possible length. If `s` does not have a duplicated substring, the answer is `""`.

**Example 1:**

Input: `s = "banana"`  
Output: `"ana"`

**Example 2:**


Input: `s = "abcd"`  
Output: `""`


**Constraints:**


- `2 <= s.length <= 3 * 104`
- `s` consists of lowercase English letters.


Accepted 60,935 Submissions 199,240

See this question in a real interview before? ☐ Yes ☐ No

Companies 

Related Topics 

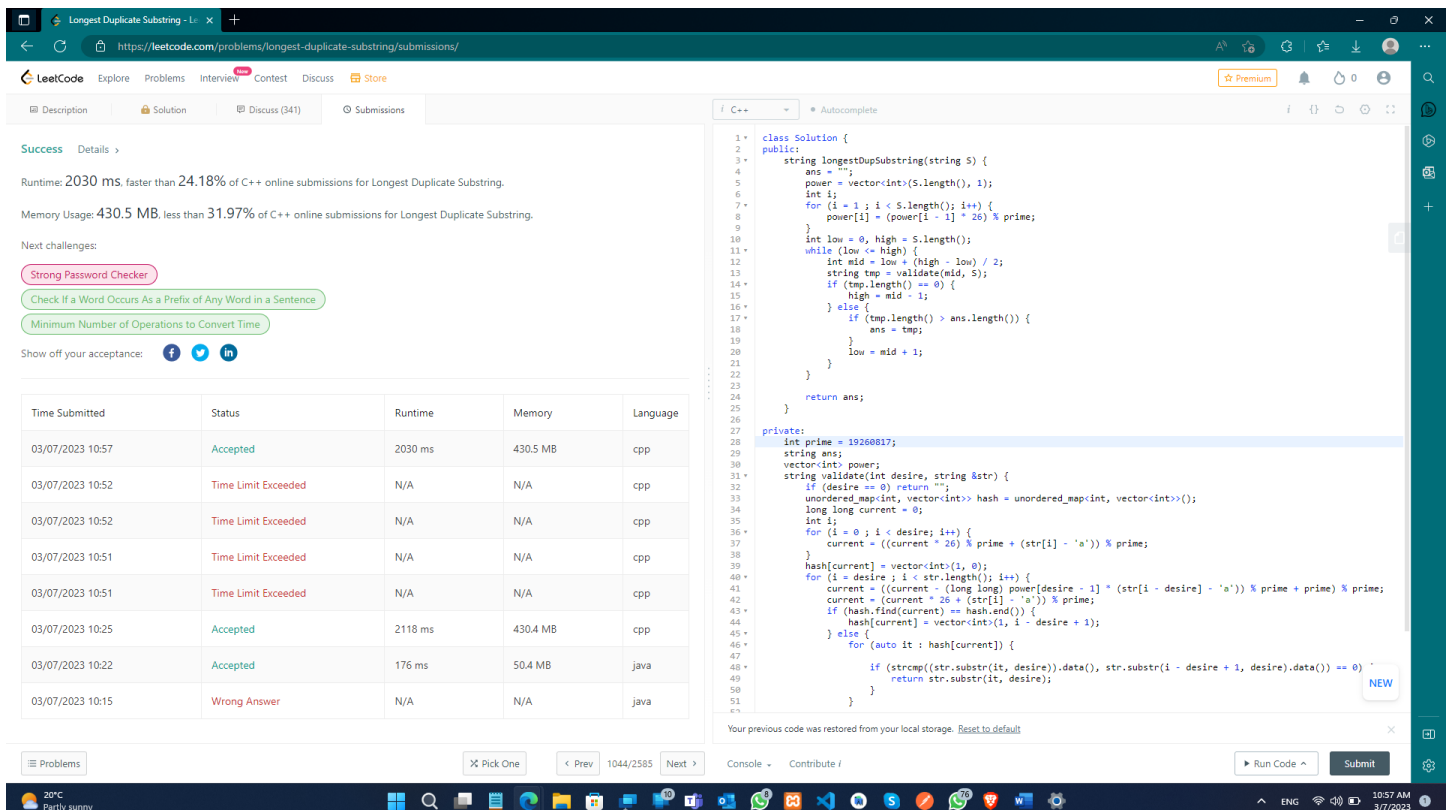
Show Hint 1 


Show Hint 2 

```

class Solution {
public:
    string longestDupSubstring(string S) {
        int ans = 0;
        power = vector<int>(S.length(), 1);
        int i;
        for (i = 1; i < S.length(); i++) {
            power[i] = (power[i - 1] * 26) % prime;
        }
        int low = 0, high = S.length();
        while (low < high) {
            int mid = low + (high - low) / 2;
            string tmp = validate(mid, S);
            if (tmp.length() == 0) {
                high = mid - 1;
            } else {
                if (tmp.length() > ans.length()) {
                    ans = tmp;
                }
                low = mid + 1;
            }
        }
        return ans;
    }
private:
    int prime = 19260817;
    string ans;
    vector<int> power;
    string validate(int desire, string &str) {
        if (desire == 0) return "";
        unordered_map<int, vector<int>> hash = unordered_map<int, vector<int>>();
        long long current = 0;
        int i;
        for (i = 0; i < desire; i++) {
            current = ((current * 26) % prime + (str[i] - 'a')) % prime;
        }
        hash[current] = vector<int>(1, 0);
        for (i = desire; i < str.length(); i++) {
            current = ((current - (long long) power[desire - 1] * (str[i - desire] - 'a')) % prime + prime) % prime;
            current = (current * 26 + (str[i] - 'a')) % prime;
            if (hash.find(current) == hash.end()) {
                hash[current] = vector<int>(1, i - desire + 1);
            } else {
                for (auto it : hash[current]) {
                    if (strcmp(str.substr(it, desire).data(), str.substr(i - desire + 1, desire).data()) == 0)
                        return str.substr(it, desire);
                }
            }
        }
    }
};

```






**Success** Details 

Runtime: 2030 ms, faster than 24.18% of C++ online submissions for Longest Duplicate Substring.

Memory Usage: 430.5 MB, less than 31.97% of C++ online submissions for Longest Duplicate Substring.

Next challenges:

- [Strong Password Checker](#)
- [Check if a Word Occurs As a Prefix of Any Word in a Sentence](#)
- [Minimum Number of Operations to Convert Time](#)

Show off your acceptance:   

Time Submitted	Status	Runtime	Memory	Language
03/07/2023 10:57	Accepted	2030 ms	430.5 MB	cpp
03/07/2023 10:52	Time Limit Exceeded	N/A	N/A	cpp
03/07/2023 10:52	Time Limit Exceeded	N/A	N/A	cpp
03/07/2023 10:51	Time Limit Exceeded	N/A	N/A	cpp
03/07/2023 10:51	Time Limit Exceeded	N/A	N/A	cpp
03/07/2023 10:25	Accepted	2118 ms	430.4 MB	cpp
03/07/2023 10:22	Accepted	176 ms	50.4 MB	java
03/07/2023 10:15	Wrong Answer	N/A	N/A	java

```

class Solution {
public:
    string longestDupSubstring(string S) {
        int ans = 0;
        power = vector<int>(S.length(), 1);
        int i;
        for (i = 1; i < S.length(); i++) {
            power[i] = (power[i - 1] * 26) % prime;
        }
        int low = 0, high = S.length();
        while (low < high) {
            int mid = low + (high - low) / 2;
            string tmp = validate(mid, S);
            if (tmp.length() == 0) {
                high = mid - 1;
            } else {
                if (tmp.length() > ans.length()) {
                    ans = tmp;
                }
                low = mid + 1;
            }
        }
        return ans;
    }
private:
    int prime = 19260817;
    string ans;
    vector<int> power;
    string validate(int desire, string &str) {
        if (desire == 0) return "";
        unordered_map<int, vector<int>> hash = unordered_map<int, vector<int>>();
        long long current = 0;
        int i;
        for (i = 0; i < desire; i++) {
            current = ((current * 26) % prime + (str[i] - 'a')) % prime;
        }
        hash[current] = vector<int>(1, 0);
        for (i = desire; i < str.length(); i++) {
            current = ((current - (long long) power[desire - 1] * (str[i - desire] - 'a')) % prime + prime) % prime;
            current = (current * 26 + (str[i] - 'a')) % prime;
            if (hash.find(current) == hash.end()) {
                hash[current] = vector<int>(1, i - desire + 1);
            } else {
                for (auto it : hash[current]) {
                    if (strcmp(str.substr(it, desire).data(), str.substr(i - desire + 1, desire).data()) == 0)
                        return str.substr(it, desire);
                }
            }
        }
    }
};

```

**1. Aim/Overview of the practical:**

**Q.2 Missing Number**

<https://leetcode.com/problems/missing-number/>

**2. Apparatus / Simulator Used:**

- Windows 7 or above
- Google Chrome

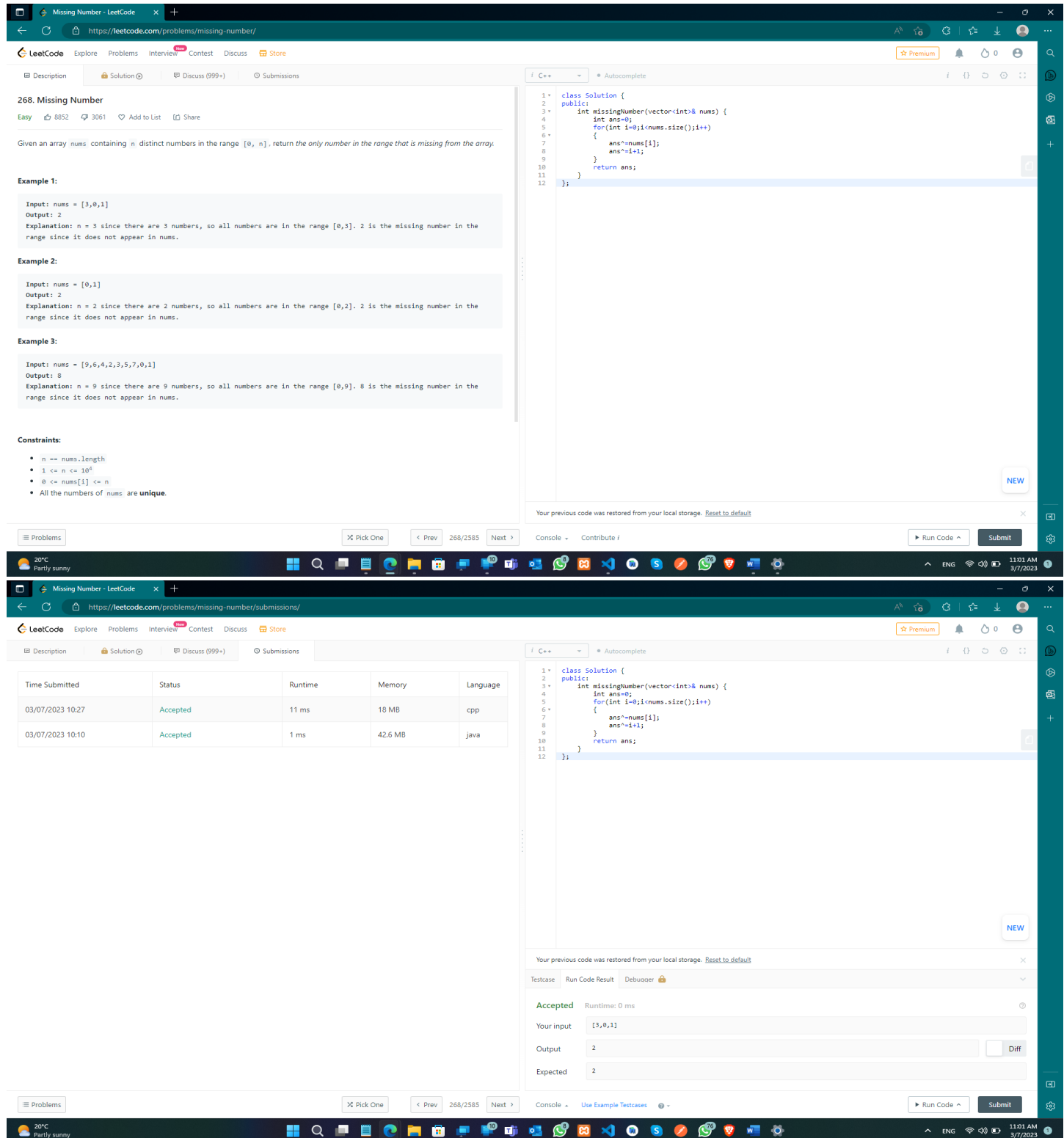
**3. Objective:**

- To understand the concept of Looping.
- To implement the concept of calculate the sum.

**4. Code:**

```
class Solution {  
public:  
    int missingNumber(vector<int>& nums) {  
        int ans=0;  
        for(int i=0;i<nums.size();i++)  
        {  
            ans^=nums[i];  
            ans^=i+1;  
        }  
        return ans;  
    }  
};
```

## 5. Result/Output/Writing Summary:



The screenshot displays the LeetCode website interface for the 'Missing Number' problem (Problem 268). The problem description states: "Given an array `nums` containing `n` distinct numbers in the range `[0, n]`, return the only number in the range that is missing from the array."

Example 1:  
Input: `nums = [3,0,1]`  
Output: 2  
Explanation: `n = 3` since there are 3 numbers, so all numbers are in the range `[0,3]`. 2 is the missing number in the range since it does not appear in `nums`.

Example 2:  
Input: `nums = [0,1]`  
Output: 2  
Explanation: `n = 2` since there are 2 numbers, so all numbers are in the range `[0,2]`. 2 is the missing number in the range since it does not appear in `nums`.

Example 3:  
Input: `nums = [9,6,4,2,3,5,7,0,1]`  
Output: 8  
Explanation: `n = 9` since there are 9 numbers, so all numbers are in the range `[0,9]`. 8 is the missing number in the range since it does not appear in `nums`.

Constraints:

- `n == nums.length`
- `1 <= n <= 104`
- `0 <= nums[i] <= n`
- All the numbers of `nums` are **unique**.

The solution code is written in C++:

```
class Solution {
public:
    int missingNumber(vector<int>& nums) {
        int ans=0;
        for(int i=0;i<nums.size();i++)
        {
            ans+=nums[i];
            ans+=i+1;
        }
        return ans;
    }
};
```

The bottom part of the screenshot shows the 'Submissions' tab with a table of results:

Time Submitted	Status	Runtime	Memory	Language
03/07/2023 10:27	Accepted	11 ms	18 MB	cpp
03/07/2023 10:10	Accepted	1 ms	42.6 MB	java

The 'Testcase' section shows the input `[3,0,1]`, the output `2`, and the expected output `2`, with a status of 'Accepted' and runtime of 0 ms.

## Learning outcomes (What I have learnt):

- Learned the concept of cheapest flights within k stops.
- Learnt about Array in Vector and Its iteration.

Submitted By: Vivek Kumar