

Experiment – 1.4

Student Name: Avinash Jena

UID: 20BCS2690

Branch: CSE

Section/Group: 707/B

Subject Name: Competitive Coding II

Subject Code: 20CSP-351

Aim: Missing Number

Objective:

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]`

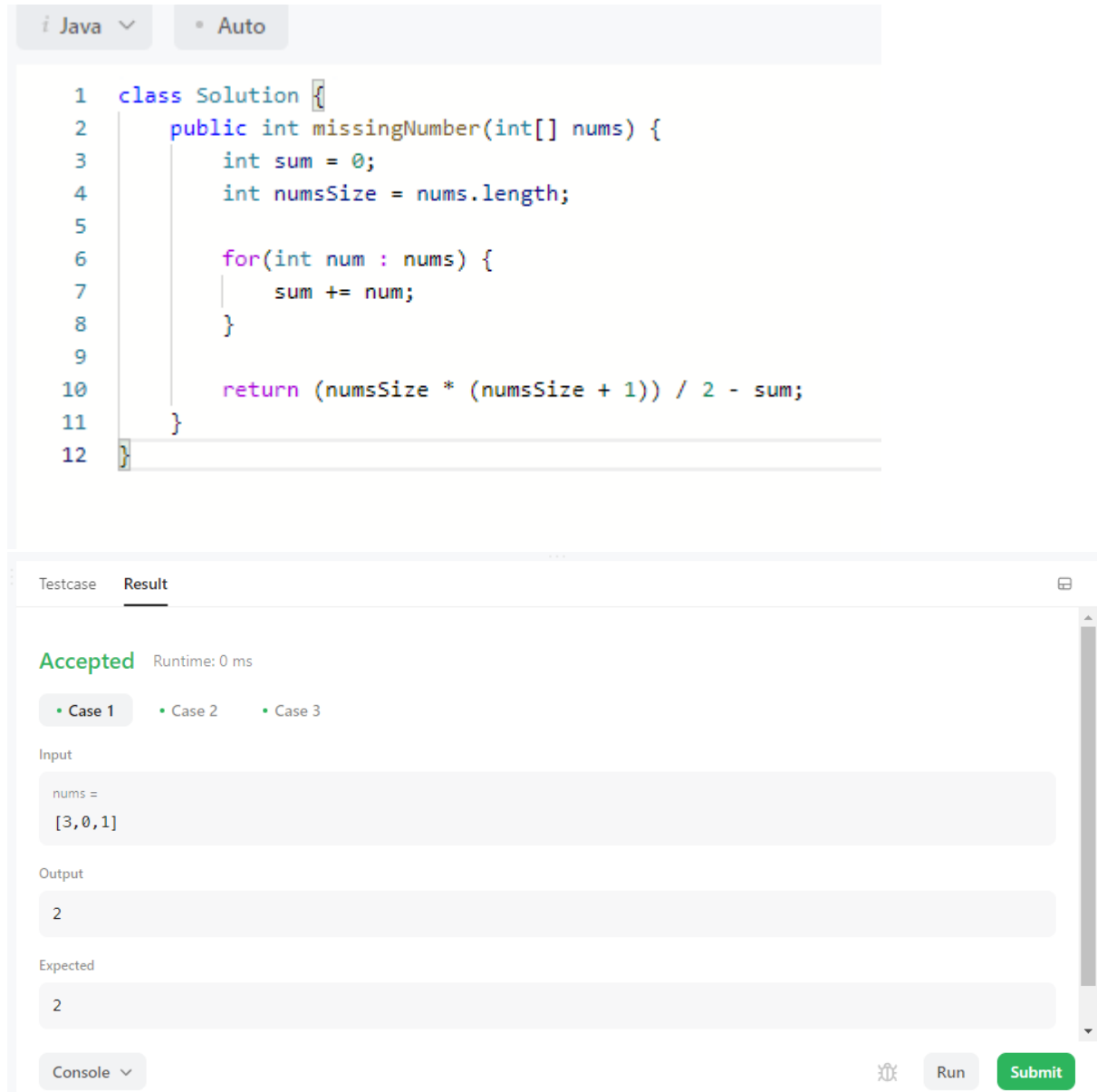
Output: 2

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`.

Code:

```
class Solution {  
    public int missingNumber(int[] nums) {  
        int sum = 0;  
        int numsSize = nums.length;  
  
        for(int num : nums) {  
            sum += num;  
        }  
  
        return (numsSize * (numsSize + 1)) / 2 - sum;  
    }  
}
```

Output:



The screenshot displays a Java code editor at the top and a test result panel below it. The code editor shows a Java class named `Solution` with a method `missingNumber` that calculates the missing number in an array. The test result panel shows the code was accepted with a runtime of 0 ms. It includes tabs for 'Testcase' and 'Result', with 'Result' selected. Under 'Result', there are tabs for 'Case 1', 'Case 2', and 'Case 3', with 'Case 1' selected. The input is `nums = [3,0,1]`, the output is `2`, and the expected result is `2`. At the bottom right, there are buttons for 'Run' and 'Submit'.

```
1 class Solution {  
2     public int missingNumber(int[] nums) {  
3         int sum = 0;  
4         int numsSize = nums.length;  
5  
6         for(int num : nums) {  
7             sum += num;  
8         }  
9  
10        return (numsSize * (numsSize + 1)) / 2 - sum;  
11    }  
12 }
```

Testcase Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =
[3,0,1]

Output

2

Expected

2

Console Run Submit

Aim: Longest Duplicate Substring**Objective:**

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: $s = \text{"banana"}$

Output: "ana"

Example 2:

Input: $s = \text{"abcd"}$

Output: ""

Constraints:

$2 \leq s.length \leq 3 * 10^4$

s consists of lowercase English letters.

Code:

```
class Solution {
    private static final long q = (1 << 31) - 1;
    private static final long R = 256;

    public String longestDupSubstring(String S) {
        int left = 2;
        int right = S.length() - 1;
        int start = 0;
        int maxLen = 0;

        while (left <= right) {
            int len = left + (right - left) / 2;
            boolean found = false;

            Map<Long, List<Integer>> map = new HashMap<>();
            long hash = hash(S, len);
            map.put(hash, new ArrayList<>());
            map.get(hash).add(0);
            long RM = 1l;
            for (int i = 1; i < len; i++) RM = (R * RM) % q;

            loop:
            for (int i = 1; i + len <= S.length(); i++) {
                hash = (hash + q - RM * S.charAt(i - 1) % q) % q;
                hash = (hash * R + S.charAt(i + len - 1)) % q;
                if (!map.containsKey(hash)) {
                    map.put(hash, new ArrayList<>());
                } else {
                    for (int j : map.get(hash)) {
                        if (compare(S, i, j, len)) {
                            found = true;
                            start = i;
                            maxLen = len;
                            break loop;
                        }
                    }
                }
            }
        }
    }
}
```

```
        map.get(hash).add(i);
    }
    if (found) left = len + 1;
    else right = len - 1;
}

return S.substring(start, start + maxLen);
}

private long hash(String S, int len) {
    long h = 0;
    for (int j = 0; j < len; j++) h = (R * h + S.charAt(j)) % q;
    return h;
}

private boolean compare(String S, int i, int j, int len) {
    for (int count = 0; count < len; count++) {
        if (S.charAt(i++) != S.charAt(j++)) return false ;
    }
    return true ;
}
}
```

Output:

```
Java Auto

1 class Solution {
2     private static final long q = (1 << 31) - 1;
3     private static final long R = 256;
4
5     public String longestDupSubstring(String S) {
6         int left = 2;
7         int right = S.length() - 1;
8         int start = 0;
9         int maxLen = 0;
10
11         while (left <= right) {
12             int len = left + (right - left) / 2;
13             boolean found = false;
14
15             Map<Long, List<Integer>> map = new HashMap<>();
16             long hash = hash(S, len);
17             map.put(hash, new ArrayList<>());
18             map.get(hash).add(0);
19             long RM = 1L;
20             for (int i = 1; i < len; i++) RM = (R * RM) % q;
21
22             loop:
23             for (int i = 1; i + len <= S.length(); i++) {
24                 hash = (hash + q - RM * S.charAt(i - 1) % q) % q;
25                 hash = (hash * R + S.charAt(i + len - 1)) % q;
26                 if (!map.containsKey(hash)) {
27                     map.put(hash, new ArrayList<>());
28                 } else {
29                     for (int j : map.get(hash)) {
30                         if (compare(S, i, j, len)) {
31                             found = true;
32                             start = i;
33                             maxLen = len;
34                             break loop;
35                         }
36                     }
37                 }
38                 map.get(hash).add(i);
39             }
40             if (found) left = len + 1;
41             else right = len - 1;
42         }
43     }
44 }
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Testcase

Result

Accepted Runtime: 0 ms

• Case 1

• Case 2

Input

```
s =  
"banana"
```

Output

```
"ana"
```

Expected

```
"ana"
```

Console ▾

 Run Submit