

Week 1 – assignment

Question-1 (Two Sum)

```
class Solution {
    public int[] twoSum(int[] numbers, int target) {
        HashMap <Integer,Integer> mpp = new HashMap<>();
        int [] arr = new int[2];
        arr[0] =arr[1]=-1;
        for(int i =0;i<numbers.length;i++){
            int moreNeeded = target- numbers[i];
            if(mpp.containsKey(moreNeeded)){
                arr[0] = mpp.get(moreNeeded)+1;
                arr[1] = i+1;
            }
            else{
                mpp.put(numbers[i],i);
            }
        }
        return arr;
    }
}
```

The screenshot displays a code editor interface for a Java solution to the Two Sum problem. The code is as follows:

```
1 class Solution {
2     public int[] twoSum(int[] numbers, int target) {
3         HashMap <Integer,Integer> mpp = new HashMap<>();
4         int [] arr = new int[2];
5         arr[0] =arr[1]=-1;
6         for(int i =0;i<numbers.length;i++){
7             int moreNeeded = target- numbers[i];
8             if(mpp.containsKey(moreNeeded)){
9                 arr[0] = mpp.get(moreNeeded)+1;
10                arr[1] = i+1;
11            }
12            else{
13                mpp.put(numbers[i],i);
14            }
15        }
16        return arr;
17    }
18 }
19
```

Below the code editor, the runtime and memory statistics are shown:

- Runtime: 9 ms | Beats 10.37%
- Memory: 45.94 MB | Beats 99.79%

The Test Result section shows the solution is Accepted with a runtime of 0 ms. The input for the test case is:

```
numbers = [2, 7, 11, 15]
target = 9
```

Question 2

```
class Solution {

    public int[] productExceptSelf(int[] nums) {
        int [] result = new int [nums.length];
        result[0] = 1;
        for(int i=1;i<nums.length;i++){
```

```

        result[i] = nums[i-1]*result[i-1];
    }
    int suffix =1;
    for(int i= nums.length-1;i>=0;i--){
        result[i] *= suffix;
        suffix *= nums[i];
    }
    return result;
}
}

```

Accepted 24 / 24 testcases passed
arnabdas1999 submitted at Mar 04, 2025 21:26

Runtime: 2 ms | Beats: 87.89%
Memory: 55.32 MB | Beats: 74.73%

Test Result: Accepted Runtime: 0 ms
Case 1: Input: nums = [1,2,3,4]

```

1 class Solution {
2     public int[] productExceptSelf(int[] nums) {
3         int [] result = new int[nums.length];
4         result[0] = 1;
5         for(int i=1;i<nums.length;i++){
6             result[i] = nums[i-1]*result[i-1];
7         }
8         int suffix =1;
9         for(int i= nums.length-1;i>=0;i--){
10            result[i] *= suffix;
11            suffix *= nums[i];
12        }
13        return result;
14    }
15 }

```

Question 3

```

class Solution {
    public void sortColors(int[] nums) {
        int low=0,mid=0,high=nums.length-1;
        while(mid<=high){
            if(nums[mid] == 0){
                int temp = nums[low];
                nums[low]=nums[mid];
                nums[mid] = temp;
                low++;
                mid++;
            }
            else if(nums[mid]==1){
                mid++;
            }
            else {
                int temp = nums[mid];
                nums[mid]=nums[high];
                nums[high] = temp;
            }
        }
    }
}

```

```

        }
        high--;
    }
}

```

leetcode.com/problems/product-of-array-except-self/

Problem List < > < > < >

Description Accepted x Editorial Solutions Submissions Testcase

All Submissions

Accepted 24 / 24 testcases passed

arnabdas1999 submitted at Mar 04, 2025 21:26

Editorial Solution

Runtime

2 ms Beats 87.89%

Analyze Complexity

Memory

55.32 MB Beats 74.73%

75%

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =

[1,2,3,4]

Output

Ln 15, Col 2 Saved

Run Submit

```

1 class Solution {
2     public int[] productExceptSelf(int[] nums) {
3         int [] result = new int[nums.length];
4         result[0] = 1;
5         for(int i=1; i<nums.length; i++){
6             result[i] = nums[i-1]*result[i-1];
7         }
8         int suffix = 1;
9         for(int i=nums.length-1; i>=0; i--){
10             result[i] *= suffix;
11             suffix *= nums[i];
12         }
13         return result;
14     }
15 }

```