Discover. Learn. Empower. ENGINEERING

WORKSHEET-9

Student Name: Arpit Srivastava

UID: 22BCS10378

Branch: CSE

Section/Group: NTPP-603-B

Semester: 6th

Date of Performance: 20/3/25

Subject Name: AP-2

Subject Code: 22CSP-351

1. Two Sum

Solved

Easy | Topics | Companies | Hint

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Accepted 63 / 63 testcases passed

iMPERIAL07 submitted at Jan 07, 2025 14:45

Editorial | Solution

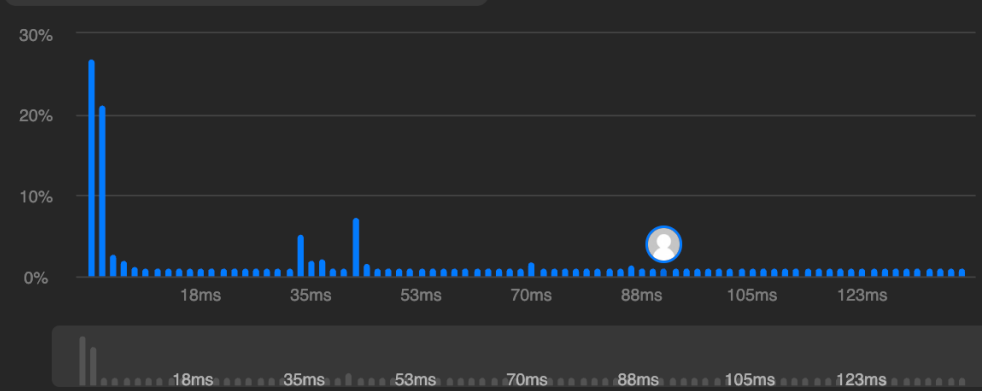
Runtime

91 ms | Beats 11.02%

Analyze Complexity

Memory

14.04 MB | Beats 77.52%



Code | C++

```cpp
class Solution {
public:
    vector<int> twoSum(vector<int> &nums, int target) {
        for (int i = 0; i < nums.size(); i++) {
            for (int j = i + 1; j < nums.size(); j++) {
                if (nums[j] == target - nums[i]) {
                    return {i, j};
                }
            }
        }
        // Return an empty vector if no solution is found
        return {};
    }
};
```

View less

7. Reverse Integer

Solved

Medium | Topics | Companies

Given a signed 32-bit integer `x`, return `x` *with its digits reversed*. If reversing `x` causes the value to go outside the signed 32-bit integer range `[-2^31, 2^31 - 1]`, then return `0`.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1:

```
Input: x = 123
Output: 321
```

Accepted 1045 / 1045 testcases passed

iMPERIAL07 submitted at Jan 07, 2025 14:53

Editorial | Solution

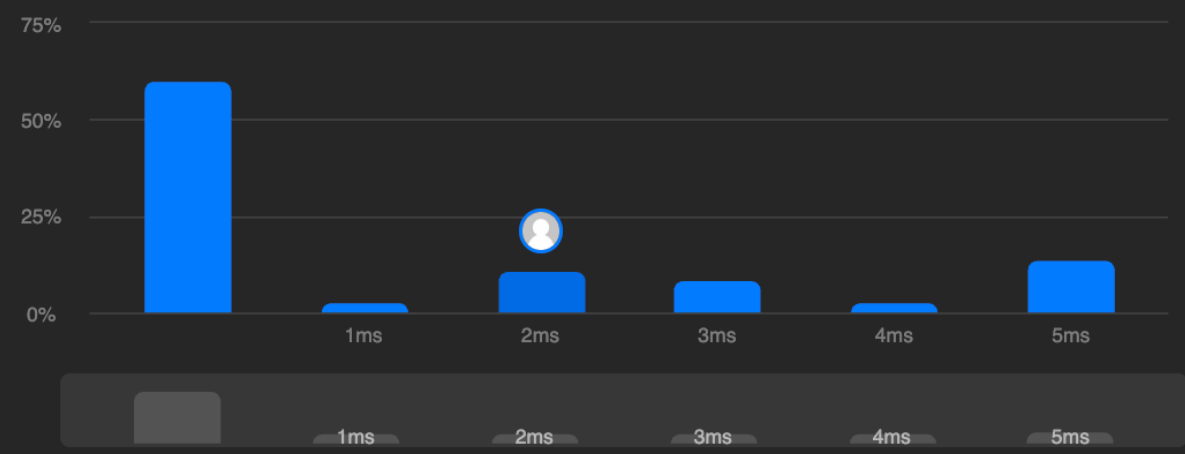
Runtime

2 ms | Beats 39.49%

Analyze Complexity

Memory

8.61 MB | Beats 20.76%



Code | C++

```cpp
class Solution {
public:
    int reverse(int x) {
```

Code | C++

```cpp
class Solution {
public:
    int reverse(int x) {
        int ans = 0; // Initialize the reversed number to 0
        while (x != 0) {
            int digit = x % 10; // Get the last digit of x

            // Check for overflow/underflow before updating ans
            if ((ans > INT_MAX / 10) || (ans < INT_MIN / 10)) {
                return 0; // Return 0 if reversing x would cause overflow/underfl
            }

            ans = ans * 10 + digit; // Append the digit to the reversed number
            x = x / 10; // Remove the last digit from x
        }
        return ans; // Return the reversed number
    }
};
```

View less

9. Palindrome Number

Solved

Easy | Topics | Companies | Hint

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise.*

Example 1:

```
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
```

Accepted 11511 / 11511 testcases passed

iMPERIAL07 submitted at Jan 07, 2025 14:54

Editorial | Solution

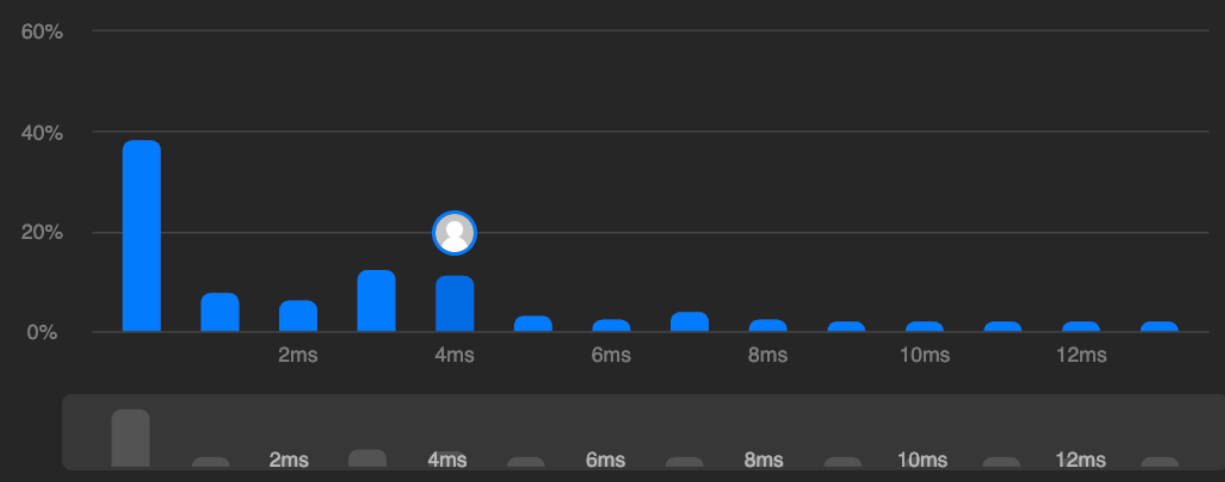
Runtime

4 ms | Beats 34.14%

Analyze Complexity

Memory

8.39 MB | Beats 99.70%



Code | C++

```cpp
class Solution {
public:
    bool isPalindrome(int x) {
        if (x < 0) {
            return false;
        }

        long long reversed = 0;
        long long temp = x;

        while (temp != 0) {
            int digit = temp % 10;
            reversed = reversed * 10 + digit;
            temp /= 10;
        }

        return (reversed == x);
    }
};
```

View less

53. Maximum Subarray

Solved

Medium | Topics | Companies

Given an integer array `nums`, find the subarray with the largest sum, and return *its sum*.

Example 1:

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.
```

Accepted 210 / 210 testcases passed

iMPERIAL07 submitted at Apr 03, 2025 15:09

Editorial | Solution

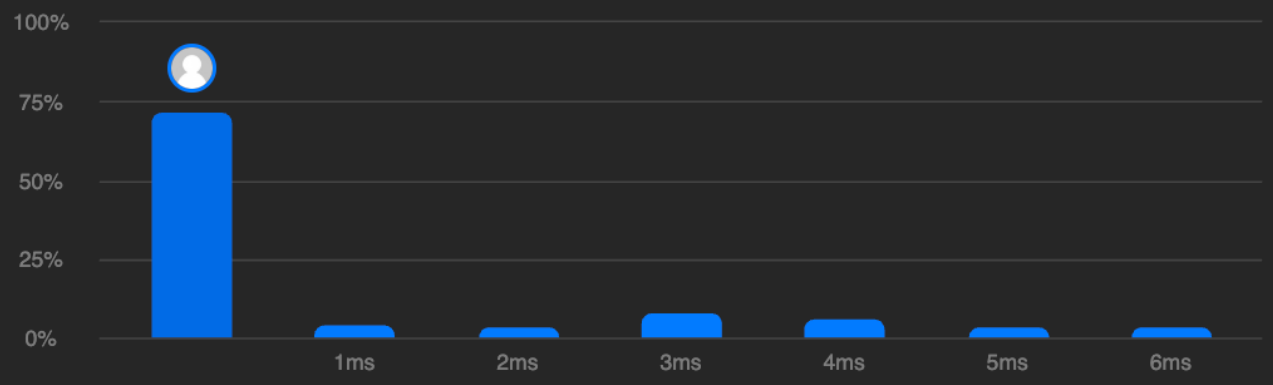
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

71.85 MB | Beats 18.84%



Code | C++

```cpp
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int n = nums.size();
        int maxSum = INT_MIN;
        int currSum = 0;

        for(int i=0; i<n; i++) {
            currSum += nums[i];

            if(currSum > maxSum) {
                maxSum  = currSum;
            }

            if(currSum < 0) {
                currSum = 0;
            }
        }
        return maxSum;
    }
};
```

View less

1710. Maximum Units on a Truck

Solved

Easy | Topics | Companies | Hint

You are assigned to put some amount of boxes onto **one truck**. You are given a 2D array `boxTypes`, where `boxTypes[i] = [numberOfBoxes$_i$, numberOfUnitsPerBox$_i$]`:

- `numberOfBoxes$_i$` is the number of boxes of type `i`.
- `numberOfUnitsPerBox$_i$` is the number of units in each box of the type `i`.

You are also given an integer `truckSize`, which is the **maximum** number of **boxes** that can be put on the truck. You can choose any boxes to put on the truck as long as the number of boxes does not exceed `truckSize`.

Return *the **maximum** total number of **units** that can be put on the truck.*

Example 1:

```
Input: boxTypes = [[1,3],[2,2],[3,1]], truckSize = 4
Output: 8
Explanation: There are:
- 1 box of the first type that contains 3 units.
- 2 boxes of the second type that contain 2 units each.
- 3 boxes of the third type that contain 1 unit each.
You can take all the boxes of the first and second types, and one box of the third type.
The total number of units will be = (1 * 3) + (2 * 2) + (1 * 1) = 8.
```

Accepted 76 / 76 testcases passed

iMPERIAL07 submitted at Apr 03, 2025 15:41

Editorial Solution

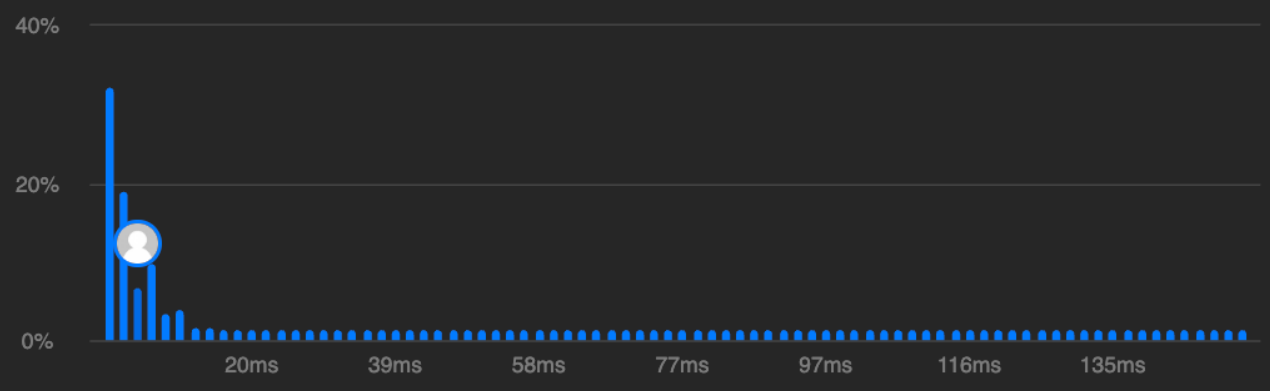
Runtime

5 ms | Beats 48.62%

Analyze Complexity

Memory

20.98 MB | Beats 22.73%



Code | C++

```
class Solution {
public:
    int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
        sort(boxTypes.begin(),boxTypes.end(),[](const vector<int>& a,const vector
            return a[1] > b[1];
        });
        int mx = 0;
        for(auto i:boxTypes)
        {
            /*if(truckSize > 0)
            {
                truckSize = truckSize - i[0];
                mx = mx + i[1]*i[0];
                if(truckSize <= 0)
                {
                    mx = mx - abs(truckSize)*i[1];
                }
            }*/

             if (truckSize == 0) break;


            int boxesToTake = min(i[0], truckSize);
            mx += boxesToTake * i[1];
            truckSize -= boxesToTake;
        }
        return mx;
    }
};
```