

1. Jump Game II

```
class Solution {
```

```
public:
```

```
    int jump(vector<int>& nums) {  
        int n=nums.size();  
        int maxi=0;  
        int current=0;  
        int count=0;  
        for(int i=0;i<n-1;i++){  
            maxi=max(maxi,i+nums[i]);  
            if(i==current){  
                count++;  
                current=maxi;  
                if(current>=n-1)break;  
            }  
        }  
        return count;  
    }
```

```
};
```

The screenshot shows a code editor interface with a dark theme. On the left, a sidebar displays submission statistics for a C++ solution. It indicates that the solution is 'Accepted' and has passed all 313 testcases. The runtime is 45 ms, which beats 77.52% of other solutions, and the memory usage is 28.94 MB, beating 85.30%. Below the statistics is a graph showing the performance of the solution relative to others. The main editor area displays the C++ code for the 'threeSum' function. The code includes a class 'Solution' with a public method 'threeSum' that takes a vector of integers and returns a vector of integers. The function sorts the input array and then iterates through it to find three numbers that sum to zero. The bottom right of the editor shows the 'Testcase' section with the input 'nums = [2,3,1,1,4]' and an empty 'Output' field.

```
class Solution {  
public:  
    int jump(vector<int>& nums) {  
        int n=nums.size();  
        int maxi=0;  
        int current=0;  
        int count=0;  
        for(int i=0;i<n-1;i++){  
            maxi=max(maxi,i+nums[i]);  
            if(i==current){  
                count++;  
                current=maxi;  
                if(current>=n-1)break;  
            }  
        }  
        return count;  
    }  
};
```

```
1 class Solution {  
2 public:  
3     vector<vector<int>> threeSum(vector<int>& nums) {  
4  
5         int n = nums.size();  
6         sort(nums.begin(),nums.end());  
7         vector<vector<int>> finalArr;  
8         for(int i =0;i<n-2;i++){  
9             if(i>0 && nums[i]==nums[i-1])  
10                continue;  
11             int j = i+1;  
12             int c = n-1;  
13             // int sum = 0;  
14             while(j<c){  
15                 int sum = nums[i] + nums[j] + nums[c];  
16  
17                 // sum+=nums[i];  
18                 // sum+=nums[j];  
19                 // sum-=nums[c];  
20                 if(sum==0){  
21                     finalArr.push_back({nums[i],nums[j],nums[c]});  
22                     j++;  
23                 }  
24             }  
25         }  
26         return finalArr;  
27     }  
28 }
```

Testcase

Input

nums =
[2,3,1,1,4]

Output

2. Jump Game

```
class Solution {
```

```
public:
```

```
    bool canJump(vector<int>& nums) {
```

```
        int n=nums.size();
```

```
        int maxJump=0;
```

```
        for(auto i=0;i<n;++i){
```

```
            if(i>maxJump)return false;
```

```
            maxJump=max(maxJump,i+nums[i]);
```

```
        }
```

```
        return true;
```

```
    }
```

```
};
```

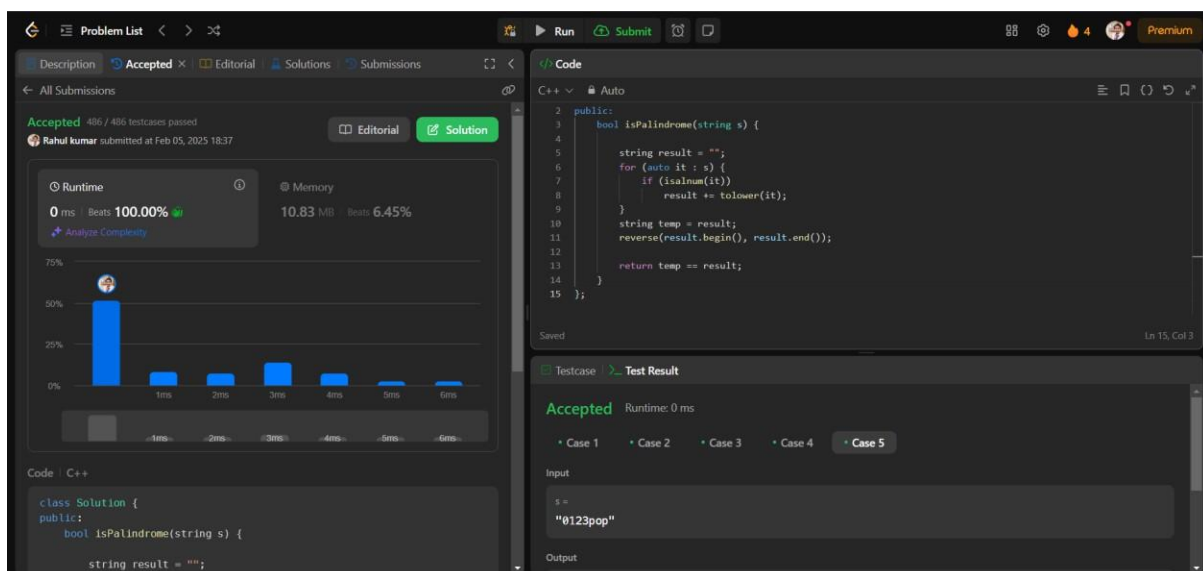
The screenshot displays a C++ IDE interface. On the left, the 'Accepted' status is confirmed with 11511/11511 test cases passed. The runtime is 7ms (Beats 16.28%) and memory usage is 6.54 MB (Beats 100.00%). A bar chart shows the performance relative to other submissions. The code editor on the right shows the following C++ code:

```
1 class Solution {
2 public:
3     bool isPalindrome(int x) {
4         long long sum = 0;
5         int g = x;
6         while(x>0 ){
7             sum = sum*10 + x%10;
8             x= x/10;
9         }
10
11         return g==sum;
12     }
13 }
14 ;
```

The bottom panel shows the 'Test Result' for 'Case 1', which is 'Accepted' with a runtime of 0 ms.

4. Valid Palindrome

```
class Solution {  
public:  
    bool isPalindrome(string s) {  
  
        string result = "";  
        for (auto it : s) {  
            if (isalnum(it))  
                result += tolower(it);  
        }  
        string temp = result;  
        reverse(result.begin(), result.end());  
  
        return temp == result;  
    }  
};
```



6. Majority Element

```
class Solution {
```

```
public:
```

```
    int majorityElement(vector<int>& nums) {
```

```
        map<int, int> freq;
```

```
        int n = nums.size();
```

```
        int majority = 0;
```

```
        for (auto it : nums) {
```

```
            freq[it]++;
```

```
        }
```

```
        for (auto it : freq) {
```

```
            if (it.second > n / 2)
```

```
                majority = it.first;
```

```
        }
```

```
        return majority;
```

```
    }
```

```
};
```

The screenshot shows a C++ IDE interface. On the left, the 'Problem List' and 'All Submissions' tabs are visible. The 'Solution' tab is active, showing submission details for 'Rupesh_b...' submitted at Dec 28, 2023 22:39. The submission is 'Accepted' with 61/61 testcases passed. The runtime is 11 ms, beating 46.70% of other submissions. The memory usage is 11.27 MB, beating 99.98% of other submissions. A bar chart shows the distribution of runtime and memory usage across different test cases. The right panel displays the C++ code for the 'majorityElement' function, which uses a map to count the frequency of each element in the input vector. The code is as follows:

```
1 class Solution {
2 public:
3
4     vector<int> twoSum(vector<int>& nums, int target) {
5         unordered_map<int, int> numMap;
6         int n = nums.size();
7
8         for (int i = 0; i < n; i++) {
9             int complement = target - nums[i];
10            if (numMap.count(complement)) {
11                return {numMap[complement], i};
12            }
13            numMap[nums[i]] = i;
14        }
15
16        return {};
17    }
18
19 };
20
```

The bottom status bar indicates 'Accepted Runtime: 0 ms'.

7. Contains Duplicate

// code

```
class Solution {
```

```
public:
```

```
    bool containsDuplicate(vector<int>& nums) {
```

```
        set<int> duplicate(nums.begin(), nums.end());
```

```
        if (duplicate.size() == nums.size())
```

```
            return false;
```

```
        return true;
```

```
    }
```

```
};
```

The screenshot shows a coding platform interface with the following components:

- Problem List:** Shows the problem is "Accepted" with 76/76 testcases passed. The user "Rahul kumar" submitted it on Feb 04, 2025 at 11:06.
- Runtime and Memory:** Runtime is 79 ms (Beats 5.29%), Memory is 77.18 MB (Beats 7.95%).
- Code Editor:** Contains the C++ code for the solution:

```
1 class Solution {
2 public:
3     bool containsDuplicate(vector<int>& nums) {
4
5         set<int> duplicate(nums.begin(), nums.end());
6
7         if (duplicate.size() == nums.size())
8             return false;
9         return true;
10    }
11};
```
- Test Result:** Shows the solution is "Accepted" with a runtime of 0 ms. The input is `nums = [1,2,3,4]` and the output is empty.

8. Find the Duplicate Number

```
class Solution {  
public:  
    int findDuplicate(vector<int>& nums) {  
        int slow = nums[0];  
        int fast = nums[0];  
        do {  
            slow = nums[slow];  
            fast = nums[nums[fast]];  
        } while (slow != fast);  
        slow = nums[0];  
        while (slow != fast) {  
            slow = nums[slow];  
            fast = nums[fast];  
        }  
        return slow;  
    }  
};
```

The screenshot displays a coding platform interface for the problem "287. Find the Duplicate Number". The problem description states: "Given an array of integers `nums` containing $n + 1$ integers where each integer is in the range $[1, n]$ inclusive. There is only **one** repeated number in `nums`, return *this* repeated number. You must solve the problem **without** modifying the array `nums` and using only constant extra space." Three examples are provided: Example 1: Input: `nums = [1,3,4,2,2]`, Output: 2; Example 2: Input: `nums = [3,1,3,4,2]`, Output: 3; Example 3: Input: `nums = [3,3,3,3,3]`, Output: 3. The submission results show the code was "Accepted" with 59/59 testcases passed, submitted by "Rahul kumar" on Dec 24, 2024 at 11:49. Performance metrics include Runtime: 0 ms (Beats 100.00%) and Memory: 65.13 MB (Beats 41.16%). A bar chart shows the distribution of runtime times across testcases, with most times clustered below 10ms. The interface also includes tabs for "Description", "Editorial", "Solutions", and "Submissions", and a "Testcase" section at the bottom.

9. Insertion Sort

// code

```
class Solution {  
  
public:  
  
    void insertionSort(vector<int>& arr) {  
  
        int n=arr.size()  
  
        for(int i=1;i<n;i++){  
  
            int key=arr[i];  
  
            int j=i-1;  
  
            while(j>=0&&arr[j]>key){  
  
                arr[j+1]=arr[j];  
  
                j--;  
  
            }  
  
            arr[j+1]=key;  
  
        }  
  
    }  
  
};
```

The screenshot displays a C++ IDE interface. On the left, the 'Output Window' shows 'Compilation Results' for a problem named 'Y.O.G.I. (AI Bot)'. It indicates 'Problem Solved Successfully' with 1115 test cases passed out of 1115, 1 attempt correct out of 1, 100% accuracy, 2 points scored out of 2, and a time taken of 0.03 seconds. The 'Solve Next' section lists 'Bubble Sort', 'Selection Sort', and 'Counting Sort'. On the right, the code editor shows the C++ implementation of Insertion Sort, which matches the code provided in the previous block. The code includes necessary headers, uses the std namespace, and defines the Solution class with the insertionSort method.

10. Two Sum

```
class Solution {  
public:  
    vector<int> twoSum(vector<int>& nums, int target) {  
        vector<pair<int, int>> indexedNums;  
        for (int i = 0; i < nums.size(); i++) {  
            indexedNums.push_back({nums[i], i});  
        }  
        sort(indexedNums.begin(), indexedNums.end());  
        int left = 0, right = nums.size() - 1;  
        while (left < right) {  
            int sum = indexedNums[left].first + indexedNums[right].first;  
            if (sum == target) {  
                return {indexedNums[left].second, indexedNums[right].second};  
            } else if (sum < target) {  
                left++;  
            } else {  
                right--;  
            }  
        }  
        return {};  
    }  
};
```


Accepted 313 / 313 testcases passed

Rupesh_bansal submitted at Dec 18, 2024 12:55

Editorial

Solution


Runtime

45 ms | Beats 77.52%

Analyze Complexity

Memory

28.94 MB | Beats 85.30%



C++

Auto

```
1 class Solution {
2 public:
3     vector<vector<int>> threeSum(vector<int>& nums) {
4
5         int n = nums.size();
6         sort(nums.begin(), nums.end());
7         vector<vector<int>> finalArr;
8         for(int i = 0; i < n - 2; i++){
9             if(i > 0 && nums[i] == nums[i - 1])
10                continue;
11             int j = i + 1;
12             int c = n - 1;
13             // int sum = 0;
14             while(j < c){
15                 int sum = nums[i] + nums[j] + nums[c];
16
17                 // sum+=nums[i];
18                 // sum+=nums[j];
19                 // sum+=nums[c];
20                 if(sum == 0){
21                     finalArr.push_back({nums[i], nums[j], nums[c]});
22                     j++;
23                 }
24             }
25         }
26         return finalArr;
27     }
28 }
```

Saved

Ln 1, Col 1

Testcase

Test Result