

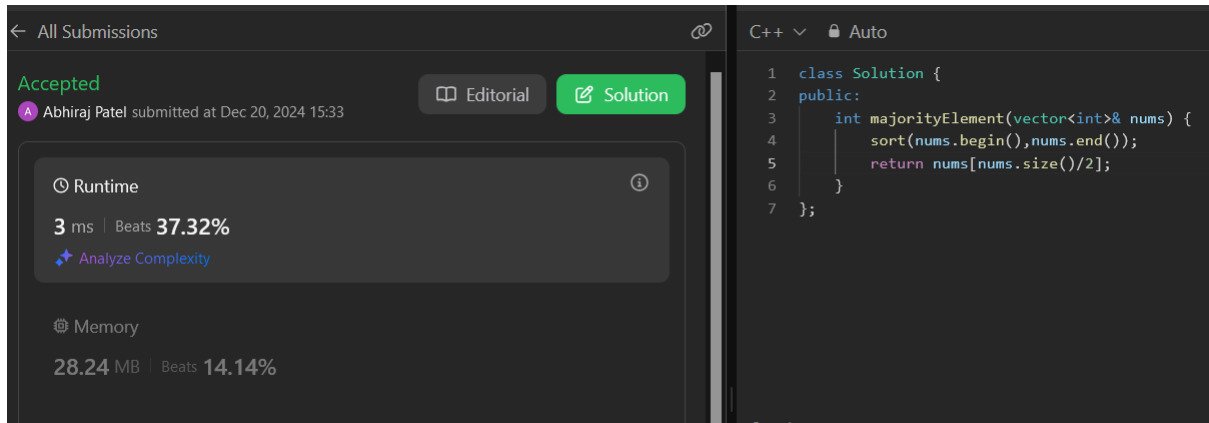
Winter Domain Camp Day-2

Name-Abhiraj Patel

UID-22BCS11329

Date-20-12-24

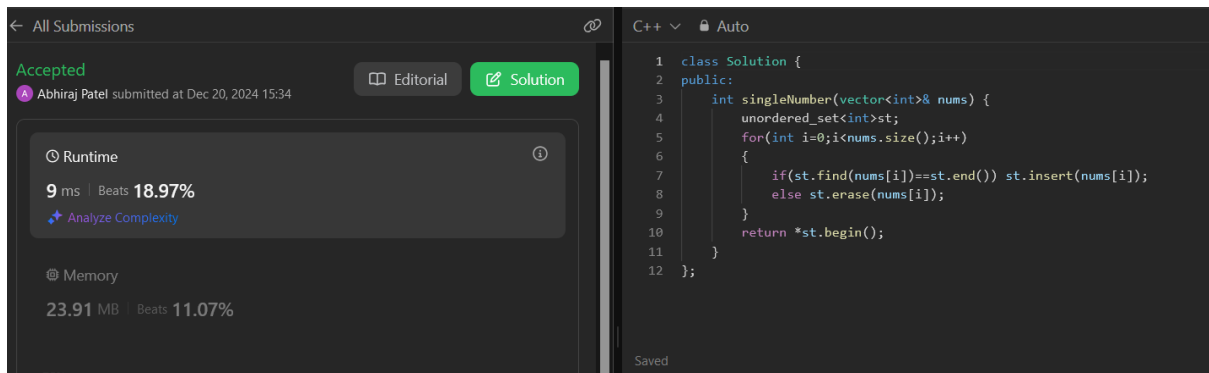
Q 1 : Majority Elements



The screenshot shows a LeetCode submission for the 'Majority Elements' problem. The submission is 'Accepted' and was submitted by 'Abhiraj Patel' at Dec 20, 2024 15:33. The runtime is 3 ms, beating 37.32% of submissions. The memory usage is 28.24 MB, beating 14.14% of submissions. The solution is written in C++ and uses a sorting approach.

```
1 class Solution {
2 public:
3     int majorityElement(vector<int>& nums) {
4         sort(nums.begin(), nums.end());
5         return nums[nums.size()/2];
6     }
7 };
```

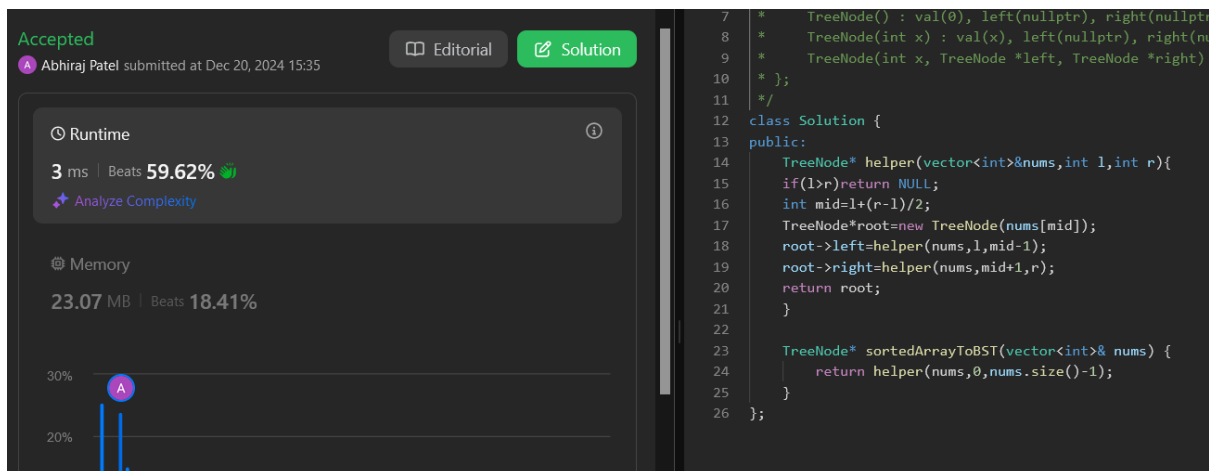
Question 2. Single Number



The screenshot shows a LeetCode submission for the 'Single Number' problem. The submission is 'Accepted' and was submitted by 'Abhiraj Patel' at Dec 20, 2024 15:34. The runtime is 9 ms, beating 18.97% of submissions. The memory usage is 23.91 MB, beating 11.07% of submissions. The solution is written in C++ and uses an unordered set to find the single number.

```
1 class Solution {
2 public:
3     int singleNumber(vector<int>& nums) {
4         unordered_set<int> st;
5         for(int i=0; i<nums.size(); i++)
6         {
7             if(st.find(nums[i]) == st.end()) st.insert(nums[i]);
8             else st.erase(nums[i]);
9         }
10        return *st.begin();
11    }
12 };
```

Question 3 [Convert Sorted Array to Binary Search Tree](#)



The screenshot shows a LeetCode submission for the 'Convert Sorted Array to Binary Search Tree' problem. The submission is 'Accepted' and was submitted by 'Abhiraj Patel' at Dec 20, 2024 15:35. The runtime is 3 ms, beating 59.62% of submissions. The memory usage is 23.07 MB, beating 18.41% of submissions. The solution is written in C++ and uses a recursive helper function to build the BST.

```
7 *   TreeNode() : val(0), left(nullptr), right(nullptr) {}
8 *   TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
9 *   TreeNode(int x, TreeNode *left, TreeNode *right) {}
10 * };
11 */
12 class Solution {
13 public:
14     TreeNode* helper(vector<int>& nums, int l, int r) {
15         if(l > r) return NULL;
16         int mid = l + (r - l) / 2;
17         TreeNode* root = new TreeNode(nums[mid]);
18         root->left = helper(nums, l, mid - 1);
19         root->right = helper(nums, mid + 1, r);
20         return root;
21     }
22
23     TreeNode* sortedArrayToBST(vector<int>& nums) {
24         return helper(nums, 0, nums.size() - 1);
25     }
26 };
```

Question 4. Remove Element

All Submissions

Accepted

Abhiraj Patel submitted at Dec 20, 2024 15:37

Editorial

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

22.57 MB | Beats 32.51%

C++

Auto

```
1 class Solution {
2 public:
3     int removeDuplicates(vector<int>& nums) {
4         int n=nums.size();
5         int k=1;
6         for(int i=1;i<n;i++)
7         {
8             if(nums[i]!=nums[i-1])
9             {
10                 nums[k]=nums[i];
11                 k++;
12             }
13         }
14         return k;
15     }
16 }
```

Saved

Question 5. Remove Linked List Elements

Accepted

Abhiraj Patel submitted at Dec 20, 2024 11:45

Editorial

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

20.17 MB | Beats 32.34%

Analyze Complexity

100%

50%

0%

1ms

2ms

3ms

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

```
ListNode(int x) : val(x), next(nullptr) {}
ListNode(int x, ListNode *next) : val(x), next(next) {}
*/
class Solution {
public:
    ListNode* removeElements(ListNode* head, int val) {
        while(head&&head->val==val)
        {
            head=head->next;
        }
        if(!head) return NULL;
        ListNode*temp=head;
        while(temp&&temp->next)
        {
            if(temp->next->val==val)
            {
                temp->next=temp->next->next;
            }
            else temp=temp->next;
        }
        return head;
    }
};
```