

LEETCODE SOLUTIONS

1.TWO SUM

Progress - LeetCode x Remove Duplicates from Sorted Array x Two Sum - LeetCode x ds/leetcode at main · ramithh: x +

leetcode.com/problems/two-sum/submissions/1867293844/

Problem List < > x

Description Accepted x Editorial Solutions Submissions Submit Ctrl Enter

All Submissions

Accepted 63 / 63 testcases passed
RamithH submitted at Dec 28, 2025 11:19

2025 Rewind Remember the Journey, Carry It Forward!

Runtime 106 ms Beats 15.18%
Memory 8.77 MB Beats 34.41%

40%
20%
0%
16ms 32ms 48ms 64ms 80ms 96ms 112ms

Code C

```
1 /**  
2  * Note: The returned array must be malloced, assume caller calls free().  
3  */  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
[2,7,11,15]

target =
9

26.Remove duplicates from sorted array

Progress - LeetCode x Remove Duplicates from Sorted Array x ds/leetcode at main · ramithh: x +

leetcode.com/problems/remove-duplicates-from-sorted-array/submissions/1867294591/

Problem List < > x

Description Accepted x Editorial Solutions Submissions Submit Ctrl Enter

All Submissions

Accepted 362 / 362 testcases passed
RamithH submitted at Dec 28, 2025 11:20

2025 Rewind Remember the Journey, Carry It Forward!

Runtime 0 ms Beats 100.00%
Memory 12.48 MB Beats 12.25%

100%
50%
0%
4ms 102ms 204ms 296ms 398ms

Code C

```
1 int removeDuplicates(int* nums, int numsSize) {  
2     if (numsSize==0) return 0;  
3     int k=1;  
4  
5     for (int i=1;i<numsSize;i++){  
6         if (nums[i]!=nums[k-1]){  
7             nums[k]=nums[i];  
8             k++;  
9         }  
10    }  
11    return k;  
12 }
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =
[1,1,2]

Output

[1,2]

Expected

1 2

35.Search insert position

This screenshot shows the LeetCode interface for the 'Search Insert Position' problem. The submission is accepted, with a runtime of 0 ms and memory usage of 8.25 MB. The code is written in C and implements a linear search algorithm. The test result shows the input array [1, 3, 5, 6] and target 5, resulting in an output of 2.

Code:

```
1 int searchInsert(int* nums, int numsSize, int target) {
2     for (int i = 0; i < numsSize; i++) {
3         if (nums[i] >= target) {
4             return i;
5         }
6     }
7     return numsSize;
8 }
```

Testcase 1:

Input: nums = [1, 3, 5, 6], target = 5

Output: 2

203.Remove Linked List Elements

This screenshot shows the LeetCode interface for the 'Remove Linked List Elements' problem. The submission is accepted, with a runtime of 2 ms and memory usage of 12.72 MB. The code is written in C and implements a linked list removal algorithm. The test result shows the input array [1, 3, 5, 6] and target 5, resulting in an output of 2.

Code:

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     struct ListNode *next;
6  * };
7  */
8 struct ListNode* removeElements(struct ListNode* head, int val) {
9     struct ListNode dummy;
10    dummy.next = head;
11    struct ListNode* curr = &dummy;
12
13    while(curr->next != NULL){
14        if(curr->next->val == val){
15            struct ListNode* temp = curr->next;
16            curr->next = curr->next->next;
17            free(temp);
18        } else{
19            curr = curr->next;
20        }
21    }
22    return dummy.next;
23 }
24 }
```

Testcase 1:

Input: nums = [1, 3, 5, 6], target = 5

Output: 2

876.Middle of the Linked List

The screenshot shows the LeetCode interface for the problem "Middle of the Linked List". The submission is accepted, with 36/36 test cases passed. The user "RamithH" submitted it on Nov 30, 2025, at 23:43. A "BLACK FRIDAY" banner is visible. The runtime is 0 ms (beats 100.00%) and memory is 8.43 MB (beats 57.28%). A bar chart shows the performance relative to other submissions. The code is in C, implementing a two-pointer approach to find the middle node of a singly-linked list.

```
1 /**  
2  * Definition for singly-linked list.  
3  * struct ListNode {  
4  *     int val;  
5  *     struct ListNode *next;  
6  * };  
7  */  
8 struct ListNode* middleNode(struct ListNode* head) {  
9     struct ListNode* first_ptr = head;  
10    struct ListNode* second_ptr = head;  
11    while(first_ptr != NULL && first_ptr->next != NULL){  
12        second_ptr = second_ptr->next;  
13        first_ptr = first_ptr->next->next;  
14    }  
15    return second_ptr;  
16 }
```

Testcase: Accepted, Runtime: 0 ms. Case 1 and Case 2 are both passed.

109.Convert sorted list to Binary Search Tree

The screenshot shows the LeetCode interface for the problem "Convert Sorted List to Binary Search Tree". The submission is accepted, with 32/32 test cases passed. The user "RamithH" submitted it on Dec 22, 2025, at 08:42. A "BLACK FRIDAY" banner is visible. The runtime is 7 ms (beats 59.23%) and memory is 21.53 MB (beats 27.31%). A bar chart shows the performance relative to other submissions. The code is in C, implementing a recursive approach to convert a sorted linked list to a binary search tree.

```
1 struct TreeNode* createBST(struct ListNode* low, struct ListNode* high) {  
2     if (low == high)  
3         return NULL;  
4  
5     struct ListNode* slow = low;  
6     struct ListNode* fast = low;  
7  
8     while (fast != high && fast->next != high) {  
9         slow = slow->next;  
10        fast = fast->next->next;  
11    }  
12  
13    struct TreeNode* node = (struct TreeNode*)malloc(sizeof(struct TreeNode));  
14    node->val = slow->val;  
15    node->left = createBST(low, slow);  
16    node->right = createBST(slow->next, high);  
17  
18    return node;  
19 }
```

Testcase: Accepted, Runtime: 0 ms. Case 1 and Case 2 are both passed. Input: head = [-10,-3,0,5,9]. Output: [0,-3,9,-10,null,5]. Expected: [0,-3,9,-10,null,5].

1669.Merge in between linked lists

The screenshot shows a LeetCode submission for problem 1669, "Merge in Between Linked Lists". The submission is accepted, with a runtime of 133 ms and memory usage of 37.38 MB. The code is written in C and implements a function to merge two linked lists between two specified nodes.

Problem Details:

- Problem: Merge in Between Linked Lists
- Accepted: 61 / 61 testcases passed
- Submitted: 1862006663/
- Editorial: [Link]
- Solution: [Link]

Performance Metrics:

- Runtime: 133 ms | Beat: 51.82%
- Memory: 37.38 MB | Beat: 78.31%

Code (C):

```
1 /**
2  * Definition for singly-linked list.
3  * struct ListNode {
4  *     int val;
5  *     struct ListNode *next;
6  * };
7  */
8
9 struct ListNode* mergeInBetween(struct ListNode* list1, int a, int b, struct ListNode* list2) {
10     struct ListNode* curr = list1;
11     for (int i = 0; i < a - 1; i++) {
12         curr = curr->next;
13     }
14     struct ListNode* beforeA = curr;
15     struct ListNode* afterB = list2;
16     struct ListNode* curr2 = list2;
17     while (curr2->next != NULL) {
18         curr2 = curr2->next;
19     }
20     beforeA->next = afterB;
21     curr2->next = curr->next;
22     return list1;
23 }
```

Testcase:

Input:

- list1 = [10,1,13,6,9,5]
- a = 3
- b = 4
- list2 = [1000000,1000001,1000002]

Output:

[10,1,13,1000000,1000001,1000002,5]

More challenges:

- 25. Reverse Nodes in k-Group
- 143. Reorder List
- 705. Design HashSet