Problem | Editorial | Submissions | Comments

Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed	Attempts : Correct / Total
1112 / 1112	**1 / 2**
	Accuracy : 50%

Points Scored	Time Taken
1 / 1	**0.09**
Your Total Score: 3 ↑	

Solve Next

Count Linked List Nodes | Delete Alternate Nodes | Insert in Middle of Linked List

Kick start your career with GfG 160!

C++ (g++ 5.4) | Start Timer

```cpp
// } Driver Code Ends
/*
struct Node {
    int data;
    struct Node* next;

    Node(int x) {
        data = x;
        next = nullptr;
    }
};
*/
/*
    Print elements of a linked list on console
    Head pointer input could be NULL as well for empty list
*/

        class Solution {
  public:
    // Function to display the elements of a linked list in the same line
    void printList(Node *head) {
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data << " ";
            temp = temp->next;
        }

    }
};

// } Driver Code Ends
```

Custom Input | Compile & Run | Submit

Description | Accepted | Editorial | Solutions | Submissions

← All Submissions

Accepted 168 / 168 testcases passed

divyanshu919891 submitted at Feb 14, 2025 15:27

Editorial | Solution

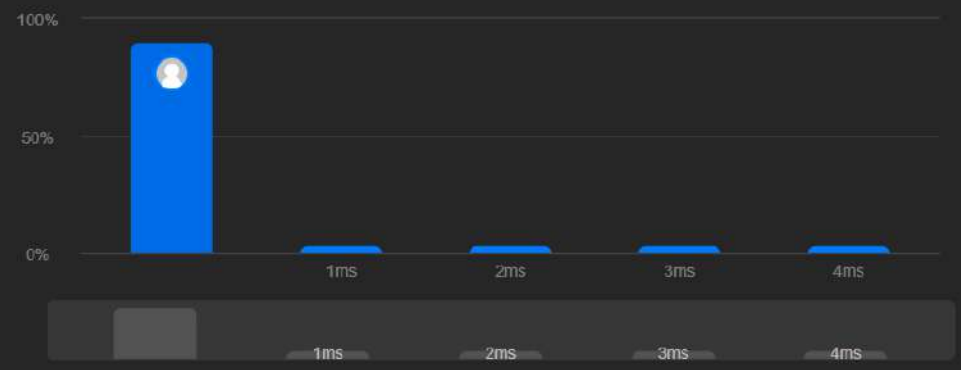
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

16.30 MB | Beats 35.36%



Code | C++

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
```

Code

C++ | Auto

```
 * };
 */
class Solution {
public:
  ListNode* deleteDuplicates(ListNode* head) {
    ListNode* curr = head;

    while (curr != nullptr) {
      while (curr->next && curr->val == curr->next->val)
        curr->next = curr->next->next;
      curr = curr->next;
    }

    return head;
  }
};
```

Saved

Ln 24, Col 3

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

Description | Editorial | Solutions | **Submissions**

	Status	Language	Runtime	Memory	Notes
2	Accepted Feb 13, 2025	C++	0 ms	13.5 MB	
1	Accepted Dec 24, 2024	C++	0 ms	13.3 MB	

Code

C++ | Auto

```cpp
class Solution {
 public:
  ListNode* reverseList(ListNode* head) {
    ListNode* prev = nullptr;

    while (head != nullptr) {
      ListNode* next = head->next;
      head->next = prev;
      prev = head;
      head = next;
    }

    return prev;
  }
};
```

Restored from local | Upgrade to Cloud Saving | Ln 1, Col 1

Testcase | Test Result

Case 1 | Case 2 | Case 3

head =

Source

Description | Accepted × | Editorial | Solutions | Submissions

← All Submissions

Accepted 70 / 70 testcases passed

divyanshu919891 submitted at Feb 14, 2025 15:30

Editorial | Solution

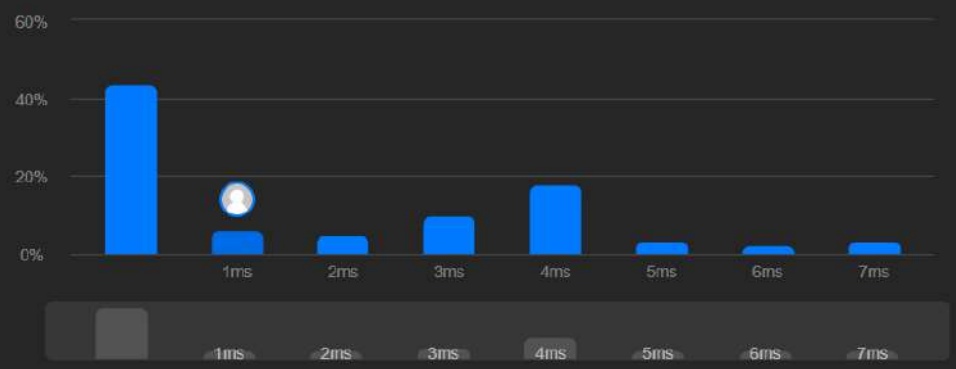
Runtime

1 ms | Beats 56.52%

Analyze Complexity

Memory

312.17 MB | Beats 17.83%



Code | C++

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
```

Code

C++ | Auto

```cpp
 */
class Solution {
 public:
  ListNode* deleteMiddle(ListNode* head) {
    ListNode dummy(0, head);
    ListNode* slow = &dummy;
    ListNode* fast = &dummy;

    while (fast->next != nullptr && fast->next->next != nullptr) {
      slow = slow->next;
      fast = fast->next->next;
    }

    slow->next = slow->next->next;
    return dummy.next;
  }
};
```

Saved

Ln 22, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 208 / 208 testcases passed

divyanshu919891 submitted at Feb 14, 2025 15:31

Editorial | Solution

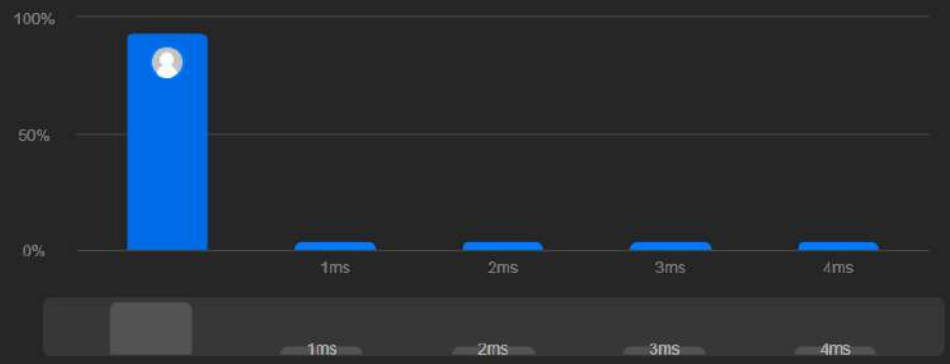
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

19.42 MB | Beats 62.56%



Code | C++

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
```

Code

C++ | Auto

```
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
 public:
  ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
    if (!list1 || !list2)
      return list1 ? list1 : list2;
    if (list1->val > list2->val)
      swap(list1, list2);
    list1->next = mergeTwoLists(list1->next, list2);
    return list1;
  }
};
```

Saved

Ln 21, Col 3

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

Description | Accepted × | Editorial | Solutions | Submissions

← All Submissions

Accepted 166 / 166 testcases passed

divyanshu919891 submitted at Feb 14, 2025 15:31

Editorial | Solution

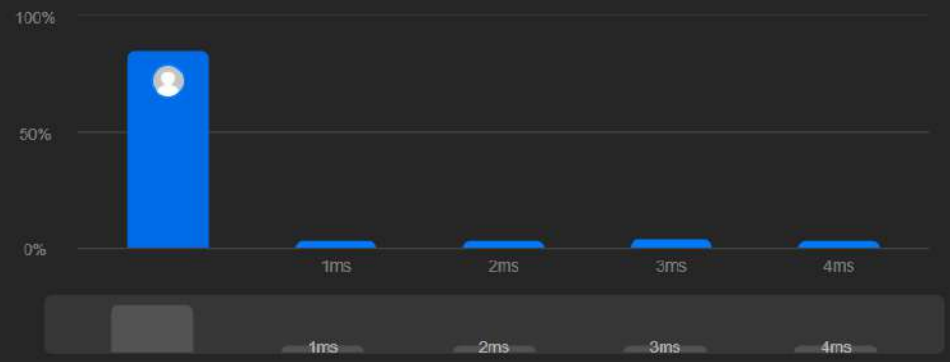
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

15.81 MB | Beats 18.26%



Code | C++

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
```

Code

C++ | Auto

```
*/
class Solution {
public:
 ListNode* deleteDuplicates(ListNode* head) {
   ListNode dummy(0, head);
   ListNode* prev = &dummy;

   while (head != nullptr) {
     while (head->next && head->val == head->next->val)
       head = head->next;
     if (prev->next == head)
       prev = prev->next;
     else
       prev->next = head->next;
     head = head->next;
   }

   return dummy.next;
 }
};
```

Saved

Ln 29, Col 3

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

Description | Editorial | Solutions | **Submissions**

	Status	Language	Runtime	Memory	Notes
2	Accepted Feb 13, 2025	C++	9 ms	11.8 MB	
1	Accepted Dec 24, 2024	C++	10 ms	11.9 MB	

Code | Accepted ×

C++ | Auto

```cpp
class Solution {
 public:
  bool hasCycle(ListNode* head) {
    ListNode* slow = head;
    ListNode* fast = head;

    while (fast != nullptr && fast->next != nullptr) {
      slow = slow->next;
      fast = fast->next->next;
      if (slow == fast)
        return true;
    }

    return false;
  }
};
```

Saved

Ln 1, Col 1

Testcase | Test Result

Case 1 Case 2 Case 3 +

head =

Source

Description | Accepted | Editorial | Solutions | Submissions

← All Submissions

Accepted 44 / 44 testcases passed

divyanshu919891 submitted at Feb 14, 2025 15:33

Editorial | Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

11.12 MB | Beats 73.55%

Code | C++

```cpp
class Solution {
 public:
  ListNode* reverseBetween(ListNode* head, int left, int right) {
    if (left == 1)
```

Code

C++ | Auto

```cpp
class Solution {
 public:
  ListNode* reverseBetween(ListNode* head, int left, int right) {
    if (left == 1)
      return reverseN(head, right);

    head->next = reverseBetween(head->next, left - 1, right - 1);

    return head;
  }

 private:
  ListNode* reverseN(ListNode* head, int n) {
    if (n == 1)
      return head;

    ListNode* newHead = reverseN(head->next, n - 1);
    ListNode* headNext = head->next;
    head->next = headNext->next;
```

Saved

Ln 24, Col 3

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

Description | Accepted × | Editorial | Solutions | Submissions

← All Submissions

Accepted 232 / 232 testcases passed

divyanshu919891 submitted at Feb 14, 2025 15:33

Editorial | Solution

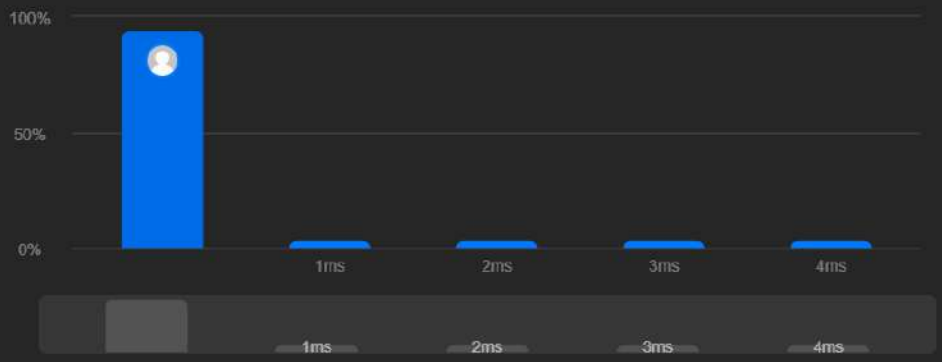
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

16.48 MB | Beats 31.82%



Code | C++

```cpp
class Solution {
 public:
  ListNode* rotateRight(ListNode* head, int k) {
    if (!head || !head->next || k == 0)
```

Code

C++ | Auto

```cpp
class Solution {
 public:
  ListNode* rotateRight(ListNode* head, int k) {
    if (!head || !head->next || k == 0)
      return head;

    ListNode* tail;
    int length = 1;
    for (tail = head; tail->next; tail = tail->next)
      ++length;
    tail->next = head;  // Circle the list.

    const int t = length - k % length;
    for (int i = 0; i < t; ++i)
      tail = tail->next;
    ListNode* newHead = tail->next;
    tail->next = nullptr;

    return newHead;
```

Saved

Ln 21, Col 3

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

Description | Accepted | Editorial | Solutions | Submissions

← All Submissions

Accepted 30 / 30 testcases passed

divyanshu919891 submitted at Feb 14, 2025 15:34

Editorial | Solution

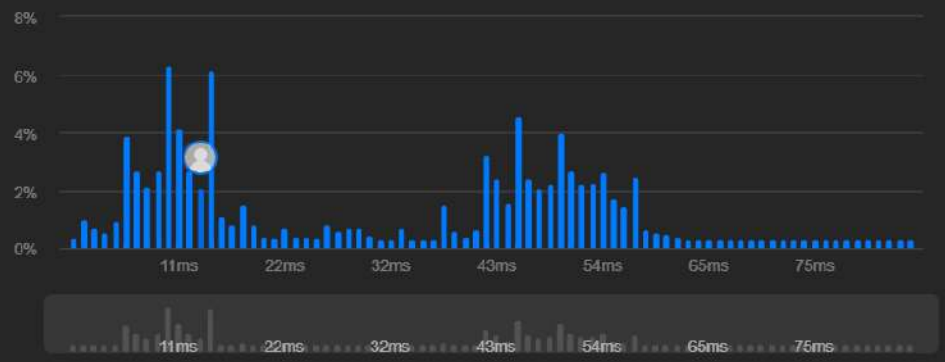
Runtime

13 ms | Beats 74.55%

Analyze Complexity

Memory

56.98 MB | Beats 90.48%



Code | C++

```
class Solution {
 public:
  ListNode* sortList(ListNode* head) {
    const int length = getLength(head);
```

Code

C++ | Auto

```
class Solution {
 public:
  ListNode* sortList(ListNode* head) {
    const int length = getLength(head);
    ListNode dummy(0, head);

    for (int k = 1; k < length; k *= 2) {
      ListNode* curr = dummy.next;
      ListNode* tail = &dummy;
      while (curr != nullptr) {
        ListNode* l = curr;
        ListNode* r = split(l, k);
        curr = split(r, k);
        auto [mergedHead, mergedTail] = merge(l, r);
        tail->next = mergedHead;
        tail = mergedTail;
      }
    }

```

Saved

Ln 57, Col 3

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

Description | Accepted | Editorial | Solutions | Submissions

← All Submissions

Accepted 18 / 18 testcases passed

divyanshu919891 submitted at Feb 14, 2025 15:35

Editorial | Solution

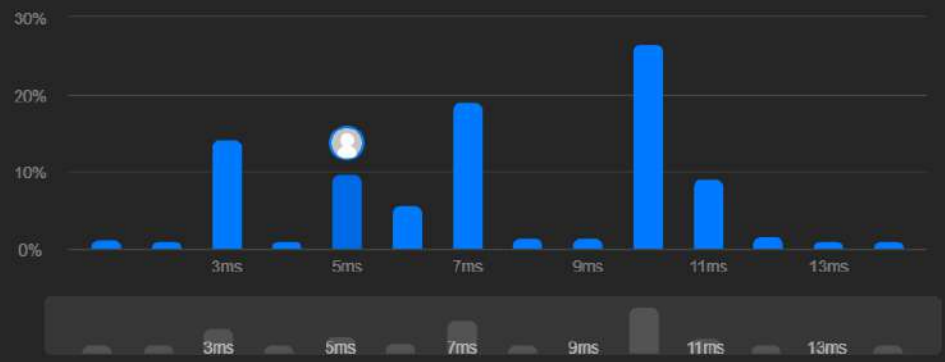
Runtime

5 ms | Beats 83.55%

Analyze Complexity

Memory

11.19 MB | Beats 98.67%



Code | C++

```cpp
class Solution {
 public:
  ListNode* detectCycle(ListNode* head) {
    ListNode* slow = head;
```

Code

C++ | Auto

```cpp
class Solution {
 public:
  ListNode* detectCycle(ListNode* head) {
    ListNode* slow = head;
    ListNode* fast = head;

    while (fast != nullptr && fast->next != nullptr) {
      slow = slow->next;
      fast = fast->next->next;
      if (slow == fast) {
        slow = head;
        while (slow != fast) {
          slow = slow->next;
          fast = fast->next;
        }
        return slow;
      }
    }

```

Saved

Ln 22, Col 3

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input