

Accepted

anannya_sharma submitted at Jan 10, 2024 23:52

Editorial

Solution

Runtime

179 ms

Beats 62.49% of users with C++

Memory

114.48 MB

Beats 92.94% of users with C++



Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head =

[1,2,2,1]

```
11 class Solution {
12 public:
13     bool isPalindrome(ListNode* head) {
14         ListNode*slow=head , *fast=head;
15         while(fast and fast->next){
16             slow = slow-> next;
17             fast=fast->next->next;
18         }
19         if(fast!=NULL and fast->next==NULL){
20             slow=slow->next;
21         }
22         ListNode* prev= NULL;
23         while(slow and slow-> next){
24             ListNode* temp=slow->next;
25             slow->next=prev;
26             prev=slow;
27             slow=temp;
28         }
29
30         if(slow!=NULL){slow->next =prev;}
31         fast=head;
32         while(slow and fast){
33             if(slow->val!=fast->val)
34                 return false;
35             slow=slow->next;
36             fast=fast->next;
37         }
38         return true;
39     }
40 }
41 };
```

Accepted

anannya_sharma submitted at Jan 11, 2024 00:01

Editorial

Solution

Runtime

11 ms

Beats 10.65% of users with C++

Memory

15.14 MB

Beats 46.06% of users with C++

30%

Testcase Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

list1 =

[1,2,4]

list2 =

[1,3,4]

Output

[1,1,2,3,4,4]

```

1  /**
2   * Definition for singly-linked list.
3   * struct ListNode {
4   *     int val;
5   *     ListNode *next;
6   *     ListNode() : val(0), next(nullptr) {}
7   *     ListNode(int x) : val(x), next(nullptr) {}
8   *     ListNode(int x, ListNode *next) : val(x), next(next) {}
9   * };
10 */
11 class Solution {
12 public:
13     ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
14         ListNode* dummy = new ListNode(0);
15         ListNode* curr = dummy;
16         while(list1!=NULL && list2!=NULL){
17             if(list1->val<=list2->val){
18                 curr->next=list1;
19                 list1=list1->next;
20             }
21             else{
22                 curr->next=list2;
23                 list2=list2->next;
24             }
25             curr= curr->next;
26         }
27         curr->next=list1!=NULL ? list1:list2;
28         return dummy->next;
29     }
30 }
31 
```

Accepted

anannya_sharma submitted at Jan 11, 2024 00:30

Editorial

Solution

Runtime

174 ms

Beats 20.42% of users with C++

Memory

74.38 MB

Beats 31.53% of users with C++

10%

☒ Testcase [Test Result](#)**Accepted**

Runtime: 3 ms

[Case 1](#) [Case 2](#) [Case 3](#)

Input

head =

[4,2,1,3]

Output

[1,2,3,4]

```
11 class Solution {
12 public:
13     ListNode* Mid(ListNode* head) {
14         ListNode* slow = head;
15         ListNode* fast = head;
16
17         while(fast->next!=NULL && fast->next->next!=NULL){
18             slow=slow->next;
19             fast=fast->next->next;
20         }
21         return slow;
22     }
23
24     ListNode* mergeSortedList(ListNode*p1 , ListNode*p2){
25         if (p1==NULL or p2==NULL){
26             return(p1==NULL)?p2:p1;
27         }
28
29         ListNode* ans=new ListNode(0);
30         ListNode* curr=ans;
31
32         while(p1!=NULL and p2!=NULL){
33             if (p1->val < p2->val){
34                 curr->next=p1;
35                 p1=p1->next;
36             }
37             else{
38                 curr->next=p2;
39                 p2=p2->next;
40             }
41         }
42     }
```

Saved to local

Accepted

anannya_sharma submitted at Jan 11, 2024 00:30

Editorial

Solution

Runtime

174 ms

Beats 20.42% of users with C++

Memory

74.38 MB

Beats 31.53% of users with C++

10%

☒ Testcase ☒ Test Result

Accepted Runtime: 3 ms

☒ Case 1 ☐ Case 2 ☐ Case 3

Input

head =
[4,2,1,3]

Output

[1,2,3,4]

```
40         p2=p2->next;
41     }
42     curr=curr->next;
43 }
44 if (p1!=NULL or p2!=NULL){
45     curr->next=(p1!=NULL)? p1:p2;
46 }
47 }
48 return ans->next;
49 }
50
51 ListNode* sortList(ListNode*head){
52     if(head == NULL or head->next==NULL)return head;
53     ListNode* mid= Mid(head);
54     ListNode* newhead =mid->next;
55     mid->next=NULL;
56
57     ListNode*left_half=sortList(head);
58     ListNode* right_half=sortList(newhead);
59     return mergeSortedList(left_half , right_half);
60 }
61
62
63
64
65 };
```

Accepted

anannya_sharma submitted at Jan 11, 2024 07:56

Editorial

Solution

Runtime

9 ms

Beats 51.14% of users with C++

30%

Memory

8.05 MB

Beats 45.35% of users with C++

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1

Case 2

Case 3

Input

[3,2,0,-4]

1

Output

tail connects to node index 1

```
6      *      ListNode(int x) : val(x), next(NULL) {}
7      *  };
8      */
9  class Solution {
10 public:
11     ListNode *detectCycle(ListNode *head) {
12         if(head==NULL || (head->next == NULL ))
13             return NULL;
14
15         ListNode* slow= head;
16         ListNode* fast= head;
17
18         while(fast!=NULL && fast->next!=NULL){
19             slow=slow->next;
20             fast= fast->next->next;
21
22             if(slow==fast)
23                 break;
24         }
25
26         if(slow!=fast)
27             return NULL;
28
29         ListNode* P= head;
30
31         while(P!=slow){
32             P=P->next;
33             slow=slow->next;
34         }
35         return P;
```

Output Window



Compilation Results

Custom Input

[Suggest Feedback](#)

Problem Solved Successfully ✓

Test Cases Passed:

1115 / 1115

Total Points Scored:

2 / 2

Your Total Score:

2 ↑

Total Time Taken:

0.21

```
98  
99 };  
100 */  
101 class Solution  
102 {  
103     public:  
104     Node* reverseDLL(Node * head)  
105     {  
106         if(!head->next){  
107             return head;  
108         }  
109         Node* current = head;  
110         Node* temp=NULL;  
111         while(current){  
112             temp=current->prev;  
113             current->prev = current->next;  
114             current->next= temp;  
115             current = current->prev;  
116         }  
117         return temp->prev;  
118     }  
119 };
```

Accepted

anan... submitted at Jan 11, 2024 20:30

Editorial

Solution

Runtime

12 ms

Beats 88.69% of users with C++

Testcase Test Result

Accepted Runtime: 5 ms

Case 1

Case 2

Case 3

Input

lists =

[[1,4,5],[1,3,4],[2,6]]

Output

```
18     ListNode* left = mergeKListsHelper(lists, start, mid);
19     ListNode* right = mergeKListsHelper(lists, mid + 1,
20     return merge(left, right);
21 }
22
23
24 ListNode* merge(ListNode* l1, ListNode* l2) {
25     ListNode* dummy = new ListNode(0);
26     ListNode* curr = dummy;
27
28     while (l1 && l2) {
29         if (l1->val < l2->val) {
30             curr->next = l1;
31             l1 = l1->next;
32         } else {
33             curr->next = l2;
34             l2 = l2->next;
35         }
36         curr = curr->next;
37     }
38
39     curr->next = l1 ? l1 : l2;
40
41     return dummy->next;
42 }
43 };
```

Accepted

anan... submitted at Jan 11, 2024 20:30

Editorial

Solution

Runtime

12 ms

Beats 88.69% of users with C++

Testcase > Test Result

Accepted Runtime: 5 ms

Case 1 Case 2 Case 3

Input

```
1 class Solution {
2 public:
3     ListNode* mergeKLists(vector<ListNode*>& lists) {
4         if (lists.empty()) {
5             return nullptr;
6         }
7         return mergeKListsHelper(lists, 0, lists.size() - 1);
8     }
9
10    ListNode* mergeKListsHelper(vector<ListNode*>& lists, int start, int end) {
11        if (start == end) {
12            return lists[start];
13        }
14        if (start + 1 == end) {
15            return merge(lists[start], lists[end]);
16        }
17        int mid = start + (end - start) / 2;
18        ListNode* left = mergeKListsHelper(lists, start, mid);
19        ListNode* right = mergeKListsHelper(lists, mid + 1, end);
20        return merge(left, right);
21    }
22 }
```