Accepted

user5565F submitted at Feb 19, 2024 06:54

Editorial | Solution

Runtime

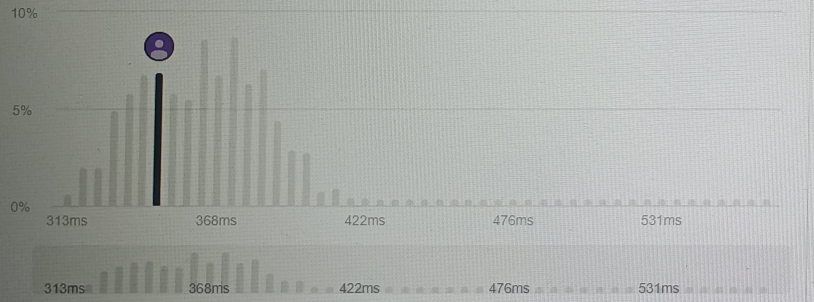
350 ms

Beats 71.89% of users with C

Memory

77.54 MB

Beats 96.31% of users with C



Code | C

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

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* deleteMiddle(struct ListNode* head) {
    struct ListNode *temp,*ptr,*ptr1;
    temp=head;
    ptr1=head;
    if(head==NULL||head->next==NULL)
        return NULL;
    while(temp!=NULL&&temp->next!=NULL)
    {
        temp=temp->next->next;
        ptr=ptr1;
        ptr1=ptr1->next;
    }
    ptr->next=ptr1->next;
    return head;

}
```

Saved to local

Testcase | Test Result

Accepted Runtime: 5 ms

• Case 1 • Case 2 • Case 3

Accepted

user5565F submitted at Feb 19, 2024 10:26

Editorial | Solution

Runtime

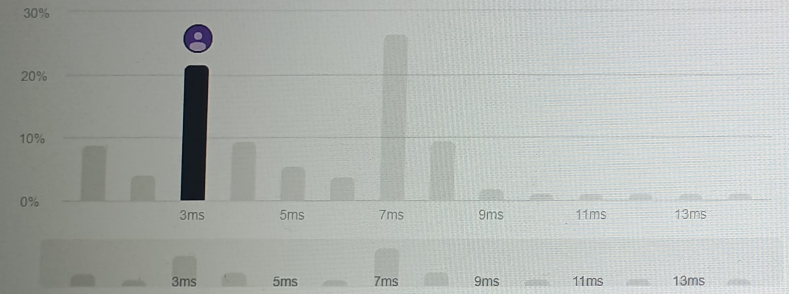
3 ms

Beats 86.96% of users with C

Memory

6.98 MB

Beats 56.98% of users with C



Code | C

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* oddEvenList(struct ListNode* head) {
    if(head==NULL || head->next==NULL)
        return head;
    struct ListNode *even=head;
    struct ListNode *evenhead=head->next;
    struct ListNode *odd=head->next;
    // temp=head;

    while(even->next!=NULL&&odd->next!=NULL)
    {
        even->next=even->next->next;
        even=even->next;
        odd->next=odd->next->next;
        odd=odd->next;
    }
    even->next=evenhead;
    return head;
}
```

Saved to local

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

19/02/2024

3. write a program

a) To construct Binary search tree

b) Traverse the tree using inorder, postorder, preorder.

c) display the elements in the tree.

```
struct node {
    int data;
    struct node * left;
    struct node * right;
};

struct node * create node (int data)
{
    struct node* new_node = (struct node*) malloc (sizeof
                            (struct node));
    new_node -> data = data;
    new_node -> left = new_node -> right = NULL;
    return new_node;
}

struct node * insert (struct node* root, int data)
{
    if (root == NULL)
        return create node(data);
    if (data < root -> data)
        root -> left = insert (root -> left, data);
    else if (data > root data)
        root -> right = inser (root -> right, data);
    return root;
}
```

```
void inorder(struct node *root)
{
    if (root!=NULL)
    {
        inorder(root->left);
        printf("%d", root->data);
        inorder(root->right);
    }
}

void preorder(struct node *root)
{
    if (root!=NULL)
    {
        printf("%d", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

void postorder(struct node *root)
{
    if (root!=NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d", root->data);
    }
}

void display(struct node *root)
{
    if (root!=NULL)
    {
        printf("Inorder:");
        inorder(root);
        printf("preorder");
        preorder(root);
        printf("postorder");
        postorder(root);
    }
}
```

```
int main()
{
    struct node *root = NULL;
    root = insert(root, 500);
    root = insert(root, 300);
    root = insert(root, 900);
    root = insert(root, 550);
    root = insert(root, 50);
    display(root);
}
```

o/p:

Inorder : 50 150 300 500 550 900

~~Postorder~~ Preorder : 500 300 150 50 900 550

~~Preorder~~ Postorder : 50 150 300 550 900 500

19.02.24

1. Delete the middle node of a linked list

```
struct ListNode * deleteMiddle(struct ListNode *head)
{
    struct ListNode *temp, *ptr, *ptr1;
    temp = head;
    ptr1 = head;
    if (head == NULL || head->next != NULL)
        return NULL;
    while (temp != NULL && temp->next != NULL)
    {
        temp = temp->next->next;
        ptr = ptr1;
        ptr1 = ptr1->next;
    }
    ptr->next = ptr1->next;
    return head;
}
```

9. Odd Even Linked List - Leetcode

```
Struct ListNode* oddEvenList(struct ListNode *head)
{
    if (head == NULL || head -> next == NULL)
        return head;
    struct ListNode *even = head, *evenhead = head->next, *odd = head;

    while (even -> next != NULL && odd -> next != NULL)
    {
        even -> next = even -> next -> next;
        even = even -> next;
        odd -> next = odd -> next -> next;
        odd = odd -> next;
    }
    even -> next = evenhead;
    return head;
}
```