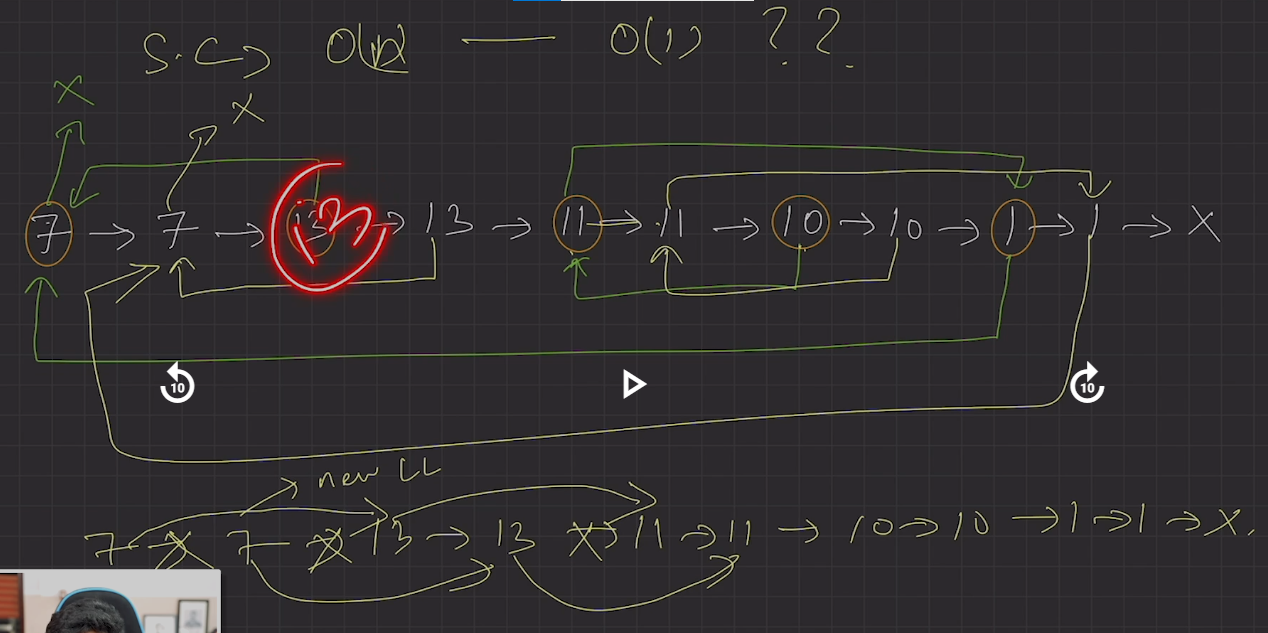
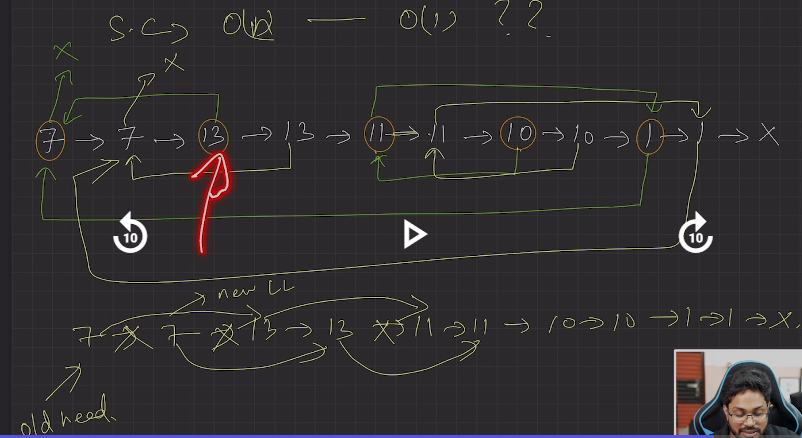
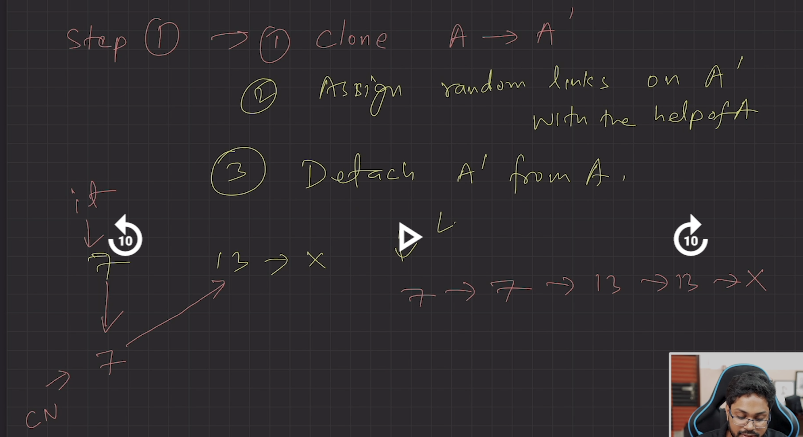
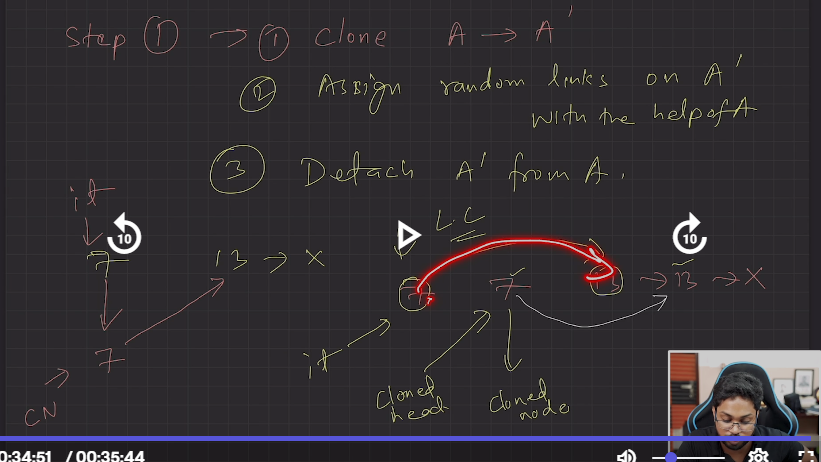
LinkedList-Assignments

Q)copy list with random pointer(deep copy) (page no-101 anand\_sunil\_part2)









Delete N nodes after M nodes of a linked list (GFG)

(page no-102 anand\_sunil\_part2)

#include <iostream>

#include <string.h>

#include <vector>

#include <algorithm>

using namespace std;

class Node

{

    public:

    int data;

    Node \*next;

    Node(int x){

        data = x;

        next = NULL;

    }

};

void linkdelete(Node  \*&head, int &m, int &n){

    if(head==NULL){

            return;

        }

    Node\*it=head;

   int i=m-1;

    while(i>0){

        cout<<"it->data"<<it->data<<endl;

        if(it==NULL){

            return;

        }

        if(it->next==NULL){

            return;

        }

        it=it->next;

        i--;

    }

    Node\* connectNode=it;

    i=n;

    while(i>0){

        if(connectNode->next==NULL){

            return ;

        }

        Node\*temp=connectNode->next;

        connectNode->next=temp->next;

        temp->next=NULL;

        delete temp;

        i--;

    }

    linkdelete(connectNode->next, m, n);

}

void print(Node  \*&head){

    Node\*temp=head;

    while(temp!=NULL){

        cout<<temp->data<<"->";

        temp=temp->next;

    }

}

int main()

{

    //9 1 3 5 9 4 10 1

    Node\* head=new Node(9);

    Node\* second=new Node(1);

    Node\* third=new Node(3);

    Node\* fourth=new Node(5);

    Node\* fifth=new Node(9);

    Node\* sixth=new Node(4);

    Node\* seventh=new Node(10);

    Node\* eight=new Node(1);

    head->next=second;

    second->next=third;

    third->next=fourth;

    fourth->next=fifth;

    fifth->next=sixth;

    sixth->next=seventh;

    seventh->next=eight;

    eight->next=NULL;

int m=3;

    int n=2;

    linkdelete(head, m, n);

    print(head);

    return 0;

}



[2058. Find the Minimum and Maximum Number of Nodes Between Critical Points](https://leetcode.com/problems/find-the-minimum-and-maximum-number-of-nodes-between-critical-points/)(leetcode)

(page no-103)\_( anand\_sunil\_part2)

#include <iostream>

#include <string.h>

#include <vector>

#include <algorithm>

#include <limits.h>

using namespace std;

class Node

{

public:

    int data;

    Node \*next;

    Node(int x)

    {

        data = x;

        next = NULL;

    }

};

void print(Node \*&head)

{

    Node \*temp = head;

    while (temp != NULL)

    {

        cout << temp->data << "->";

        temp = temp->next;

    }

}

void nodesBetweenCriticalPoints(Node \*&head, vector<int> &ans)

{

    Node \*prev = head;

    Node \*curr = prev->next;

    Node \*next = curr->next;

    int count = 2;

    while ((curr != NULL) && (next != NULL))

    {

        if ((prev->data > curr->data && next->data > curr->data) || (prev->data < curr->data && next->data < curr->data))

        {

            ans.push\_back(count);

        }

        prev = curr;

        curr = prev->next;

        next = curr->next;

        count++;

    }

}

int main()

{

    // 6,8,4,1,9,6,6,10,6

    Node \*head = new Node(6);

    Node \*second = new Node(8);

    Node \*third = new Node(4);

    Node \*fourth = new Node(1);

    Node \*fifth = new Node(9);

    Node \*sixth = new Node(6);

    Node \*seventh = new Node(6);

    Node \*eight = new Node(10);

    Node \*nineth = new Node(6);

    head->next = second;

    second->next = third;

    third->next = fourth;

    fourth->next = fifth;

    fifth->next = sixth;

    sixth->next = seventh;

    seventh->next = eight;

    eight->next = nineth;

    nineth->next = NULL;

    vector<int> ans;

    nodesBetweenCriticalPoints(head, ans);

    for (int i = 0; i < ans.size(); i++)

    {

        cout << ans[i] << " ";

    }

    cout << endl;

    vector<int> finalans = {-1, -1};

    if (ans.size() > 1)

    {

        int max = ans[ans.size() - 1] - ans[0];

        int min = INT\_MAX;

        for (int i = 0; i < ans.size() - 1; i++)

        {

            if (ans[i + 1] - ans[i] < min && i + 1 < ans.size())

            {

                min = ans[i + 1] - ans[i];

            }

        }

        finalans[0] = min;

        finalans[1] = max;

        cout<<"min ="<<min<<endl;

        cout<<"max ="<<max<<endl;

    }

    print(head);

    return 0;

}

