



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Name: Arpita

Section: 601-B

Subject Code: 22CSP-351

UID: 22BCS15627

Subject: Advanced Programming-II

DAY 1

Problem 1: Remove duplicates from sorted array

```
class Solution {
```

```
public:
```

```
    int removeDuplicates(vector<int>& nums) {
```

```
        if(nums.size()==0){
```

```
            return 0;
```

```
        }
```

```
        int k=1;
```

```
        for(int i=1;i<nums.size();i++){
```

```
            if(nums[i]!=nums[i-1]){
```

```
                nums[k]=nums[i];
```

```
                k++;
```

```
            }
```

```
        }
```

```
        return k;
```

```
    }
```

```
};
```

The screenshot shows a web browser with multiple tabs. The active tab is 'Remove Duplicates from Sorted Array' on the LeetCode website. The page displays the problem description for '26. Remove Duplicates from Sorted Array', which is marked as 'Solved'. The description states: 'Given an integer array `nums` sorted in **non-decreasing order**, remove the duplicates **in-place** such that each unique element appears only **once**. The **relative order** of the elements should be kept the **same**. Then return the *number of unique elements* in `nums`. Consider the number of unique elements of `nums` to be `k`, to get accepted, you need to do the following things: • Change the array `nums` such that the first `k` elements of `nums` contain the unique elements in the order they were present in `nums` initially. The remaining elements of `nums` are not important as well as the size of `nums`. • Return `k`.' The 'All Submissions' section shows a submission by 'ArpitaShashni' submitted on Jan 18, 2025, 17:24. The submission status is 'Accepted'. The runtime is '0 ms' (Beats 100.00%) and memory is '22.54 MB' (Beats 78.56%). The test result section shows 'Accepted' with 'Runtime: 0 ms' and 'Case 1' selected. The input for Case 1 is 'nums = [1,1,2]'.

Problem 2: Implementing insertion sort

public:

```
// Please change the array in-place
void insertionSort(vector<int>& arr) {
    int n = arr.size();
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}
```

The screenshot shows a web-based IDE interface for solving a problem. The code editor on the right contains the following C++ code:

```
1 // Driver Code Ends
2 class Solution {
3 public:
4     // Please change the array in-place
5     void insertionSort(vector<int>& arr) {
6         int n = arr.size();
7         for (int i = 1; i < n; i++) {
8             int key = arr[i];
9             int j = i - 1;
10            while (j >= 0 && arr[j] > key) {
11                arr[j + 1] = arr[j];
12                j--;
13            }
14            arr[j + 1] = key;
15        }
16    };
17 }
18 // Driver Code Ends
```

The output window on the left shows the following results:

- Test Cases Passed: 1115 / 1115
- Attempts: Correct / Total: 2 / 2
- Accuracy: 100%
- Time Taken: 0.02

A message at the bottom of the output window states: "You get marks only for the first correct submission if you solve the problem without viewing the full solution."

Problem 3: Contains Duplicate

class Solution {

public:

```
bool containsDuplicate(vector<int>& nums) {
    unordered_map<int,int> m;
    for(int i=0;i<nums.size();i++){
        m[nums[i]]++;
    }
    for(auto i:m){
        if(i.second>1){
```

```

        return true;
    }
}
return false;
}
};

```

The screenshot shows the LeetCode interface for the '217. Contains Duplicate' problem. The problem description states: 'Given an integer array `nums`, return `true` if any value appears at least twice in the array, and return `false` if every element is distinct.' Example 1 shows input `nums = [1,2,3,1]` and output `true`. Example 2 shows input `nums = [1,2,3,1]` and output `true`.

The submission details show:

- Accepted: 76 / 76 testcases passed
- Submitted by: ArpitaShashni at Aug 28, 2024 01:54
- Runtime: 86 ms | Beats 5.26%
- Memory: 75.46 MB | Beats 21.33%

The test result for Case 1 shows:

- Accepted
- Runtime: 0 ms
- Input: `nums = [1,2,3,1]`

Problem 4: Two Sum

```
class Solution {
```

```
public:
```

```

    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int,int> m;
        for(int i=0;i<nums.size();i++){
            int a=target-nums[i];
            if(m.find(a)!=m.end()){
                return {m[a],i};
            }
            m[nums[i]]=i;
        }
        return {};
    }
};

```

The screenshot shows the LeetCode interface for the 'Two Sum' problem. On the left, the problem description is visible, including the input array [2, 7, 11, 15] and target 9. The submission result on the right shows 'Accepted' status with 63/63 testcases passed, a runtime of 0 ms, and a memory usage of 14.92 MB. The input field shows the array [2, 7, 11, 15].

Problem 5: Jump Game

class Solution {

public:

bool canJump(vector<int>& nums) {

int n=nums.size();

int x=nums[0];

for(int i=0;i<n;i++){

if(i>x){

return false;

}

x=max(x,nums[i]+i);

}

return true;

}

};

The screenshot shows a web browser with multiple tabs. The active tab is 'Jump Game - LeetCode'. The URL is 'leetcode.com/problems/jump-game/submissions/1516452709/'. The page displays the submission details for the 'Jump Game' problem. The submission is 'Accepted' with 173/173 testcases passed. The user 'ArpitaShashni' submitted it on Jan 22, 2025, at 09:52. The runtime is 3 ms, which beats 29.53% of other submissions. The memory usage is 52.15 MB, which beats 84.91% of other submissions. The test case input is [2, 3, 1, 1, 4].

Problem 6: Majority element

```
#include<unordered_map>
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int n=nums.size();
        unordered_map<int,int> m;
        for(int i=0;i<n;i++){
            m[nums[i]]++;
        }
        for(int i=0;i<n;i++){
            if(m[nums[i]]>n/2){
                return nums[i];
            }
        }
        return -1;
    }
};
```

169. Majority Element Solved ✓

Easy Topics Companies

Given an array `nums` of size `n`, return the majority element.

The majority element is the element that appears more than $\lfloor n / 2 \rfloor$ times. You may assume that the majority element always exists in the array.

Example 1:
Input: `nums = [3,2,3]`
Output: `3`

Example 2:
Input: `nums = [2,2,1,1,1,2,2]`
Output: `2`

20.3K 337 268 Online

All Submissions

Accepted 52 / 52 testcases passed
 ArpitaShashni submitted at Aug 29, 2024 21:49

Runtime 25 ms | Beats 7.14%
 Memory 28.20 MB | Beats 24.50%

Testcase | **Test Result**

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =
 [3,2,3]

Problem 7: Valid Palindrome

class Solution {

public:

```

    bool isal(char &c){
        if(c>='0' && c<='9'){
            return true;
        }
        if(c>='A' && c<='Z'){
            c=c-'A'+ 'a';
            return true;
        }
        if(c>='a' && c<='z'){
            return true;
        }
        return false;
    }

    bool isPalindrome(string s) {
        if(s==""){
            return true;
        }
        //s=toLowerCase(s);
        int n = s.length();
        int i = 0;
        int j = n;
        while(i<n && j>=0)
        {

```

```

        if (isal(s[i]) == 0)
        {
            i++;
        }
        else if (isal(s[j]) == 0)
        {
            j--;
        }
        else if (s[i] == s[j])
        {
            ++i;
            j--;
        }
        else
        {
            return false;
        }
    }
    return true;
}
};

```

The screenshot shows the LeetCode interface for the 'Valid Palindrome' problem (125). The problem description states: 'A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers. Given a string *s*, return *true* if it is a **palindrome**, or *false* otherwise.'

The submission details for user 'ArpitaShashni' (submitted at Sep 24, 2024 15:20) show:

- Status: Accepted (486 / 486 testcases passed)
- Runtime: 2 ms (Beats 39.72%)
- Memory: 8.75 MB (Beats 100.00%)

The test result for Case 1 shows:

- Input: *s* = "A man, a plan, a canal: Panama"
- Output: true
- Explanation: "amanaplanacanalpanama" is a palindrome.

Problem 8: Jump Game 2

```

class Solution {
public:
    int jump(vector<int>& nums) {
        int n = nums.size();
    }
}

```

```

    if (n == 1) return 0;
    int jumps = 0, farthest = 0, currentEnd = 0;
    for (int i = 0; i < n - 1; i++) {
        farthest = max(farthest, i + nums[i]);
        if (i == currentEnd) {
            jumps++;
            currentEnd = farthest;
            if (currentEnd >= n - 1) break;
        }
    }
    return jumps;
}
};

```

The screenshot shows a web browser with multiple tabs. The active tab is 'Jump Game II - LeetCode'. The address bar shows the URL: `leetcode.com/problems/jump-game-ii/submissions/1528847549/`. The page displays the '45. Jump Game II' problem, which is marked as 'Solved'. The problem description states: 'You are given a 0-indexed array of integers nums of length n. You are initially positioned at nums[0]. Each element nums[i] represents the maximum length of a forward jump from index i. In other words, if you are at nums[i], you can jump to any nums[i + j] where: 0 <= j <= nums[i] and i + j < n. Return the minimum number of jumps to reach nums[n - 1]. The test cases are generated such that you can reach nums[n - 1]. Example 1: Input: nums = [2, 3, 1, 1, 4], Output: 3'. The submission details show 'Accepted' status with '110 / 110 testcases passed'. The submission was made by 'ArpitaShashni' on Feb 02, 2025 at 22:20. The performance metrics are 'Runtime: 0 ms | Beats 100.00%' and 'Memory: 20.45 MB | Beats 62.40%'. The input array shown is `nums = [2, 3, 1, 1, 4]`. The browser's taskbar at the bottom shows the system time as 2:14 PM on 2/5/2025.

Problem 9: 3Sum

```

class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        vector<vector<int>> ans;
        int n = nums.size();
        for (int i = 0; i < n - 2; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }
            int start = i + 1, end = n - 1;
            while (start < end) {

```



```

        int sum=nums[i]+nums[start]+nums[end];
        if(sum==0){
            ans.push_back({ nums[i],nums[start],nums[end]});
            while(start<end && nums[start]==nums[start+1]){
                start++;
            }
            while(start<end && nums[end]==nums[end-1]){
                end--;
            }
            start++;
            end--;
        }
        else if(sum>0){
            end--;
        }
        else{
            start++;
        }
    }
}
return ans;
}
};

```

The screenshot shows the LeetCode '3Sum' problem page. The problem description states: "Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0. Notice that the solution set must not contain duplicate triplets." Example 1 shows input nums = [-1, 0, 1, 2, -1, -4] and output [[-1, -1, 2], [-1, 0, 1]]. The user's submission is 'Accepted' with a runtime of 51 ms (Beats 55.97%) and memory of 29.07 MB (Beats 67.91%). The test case input is [-1, 0, 1, 2, -1, -4].

Problem 10: Set Matrix Zeroes

```

class Solution {
public:

```

```

void setZeroes(vector<vector<int>>& matrix) {
    int r=matrix.size();
    int c=matrix[0].size();
    vector<int> row(r,1);
    vector<int> col(c,1);
    for(int i=0;i<r;i++){
        for(int j=0;j<c;j++){
            if(matrix[i][j]==0){
                row[i]=0;
                col[j]=0;
            }
        }
    }
    for(int i=0;i<r;i++){
        for(int j=0;j<c;j++){
            if(row[i]==0||col[j]==0){
                matrix[i][j]=0;
            }
        }
    }
}
};

```

The screenshot shows a web browser with the LeetCode problem "73. Set Matrix Zeroes" open. The problem description states: "Given an $m \times n$ integer matrix `matrix`, if an element is 0, set its entire row and column to 0's. You must do it in place." An example shows a 3x3 matrix with a 0 at (1,1) being transformed into a matrix where the entire row 1 and column 1 are 0.

The code editor shows a C++ solution:

```

1 class Solution {
2 public:
3     void setZeroes(vector<vector<int>>& matrix) {
4         int r=matrix.size();
5         int c=matrix[0].size();
6         vector<int> row(r,1);
7         vector<int> col(c,1);
8         for(int i=0;i<r;i++){
9             for(int j=0;j<c;j++){

```

The test result shows "Accepted" with a runtime of 0 ms. The input matrix is `[[1,1,1],[1,0,1],[1,1,1]]`.

Problem 11: Longest substring without repeating characters

```

class Solution {
public:
    int lengthOfLongestSubstring(string s) {

```

```

unordered_map<char,int> m;
int ans=0,left=0;
for(int i=0;i<s.length();i++){
    char c=s[i];
    if(m.count(c) && m[c]>=left){
        left=m[c]+1;
    }
    m[c]=i;
    ans=max(ans,i-left+1);
}
return ans;
}
};

```

The screenshot shows a web browser with multiple tabs. The active tab is 'Longest Substring Without Repeating Characters' on LeetCode. The page displays the problem description, which asks to find the length of the longest substring without repeating characters. It includes two examples: Example 1 with input 's = "abcabcbb"' and output '3', and Example 2 with input 's = "bbbbbb"' and output '1'. The problem is marked as 'Solved' and 'Medium' difficulty. On the right, the 'Accepted' solution is shown, indicating it passed 987 / 987 test cases. The solution's performance is listed as 16 ms (Beats 36.56%) and 12.23 MB (Beats 43.55%). The 'Testcase' section shows the input 's = "abcabcbb"' and the output 'Accepted'.

Problem 12: Finding duplicate number

```

class Solution {
public:
    int findDuplicate(vector<int>& nums) {
        int slow=nums[0];
        int fast=nums[0];
        do{
            slow=nums[slow];
            fast=nums[nums[fast]];
        }while(slow!=fast);
        slow=nums[0];
        while(slow!=fast){
            slow=nums[slow];

```

```

        fast=nums[fast];
    }
    return fast;
}
};

```

The screenshot shows a web browser with multiple tabs. The active tab is 'Find the Duplicate Number - Le'. The browser address bar shows 'leetcode.com/problems/find-the-duplicate-number/description/'. The page displays the problem description for '287. Find the Duplicate Number', which is marked as 'Solved'. The problem description states: 'Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive. There is only one repeated number in nums, return this repeated number. You must solve the problem without modifying the array nums and using only constant extra space. Example 1: Input: nums = [1,3,4,2,2] Output: 2'. The right sidebar shows the submission status: 'Accepted 59 / 59 testcases passed', 'ArpitaShashni submitted at Jan 14, 2025 18:39', 'Runtime: 0 ms | Beats 100.00%', 'Memory: 65.16 MB | Beats 41.16%', and a 'Testcase' section showing 'Case 1' with 'nums = [1,3,4,2,2]'. The bottom of the browser shows a Windows taskbar with various icons and a system clock indicating '2:17 PM 2/5/2025'.

DAY 2

Problem 1: Print linked list

class Solution {

public:

void printList(Node *head) {

if(head==NULL){

return;

}

Node* temp=head;

while(temp!=NULL){

cout<<temp->data<<" ";

temp=temp->next;

}

}

};

The screenshot shows the GeeksforGeeks website interface. On the left, the 'Output Window' displays 'Problem Solved Successfully' with 1112/1112 test cases passed, 2/2 attempts, and 100% accuracy. The time taken is 0.07. A message states: 'You get marks only for the first correct submission if you solve the problem without viewing the full solution.' Below this is a 'Solve Next' button. On the right, the C++ code editor shows a solution for printing a linked list. The code defines a Node structure and a Solution class with a printList method. The method iterates through the linked list and prints each node's data. The code is in C++ (g++ 5.4) and includes a 'Driver Code Ends' comment.

Problem 2: Remove duplicates from a sorted array

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if(head==NULL){
            return NULL;
        }
        ListNode* temp=head;
        while(temp!=NULL && temp->next!=NULL){
            if((temp->val==temp->next->val)){
                ListNode* data=temp->next;
                temp->next=data->next;
                delete data;
            }
            else{
                temp=temp->next;
            }
        }
        return head;
    }
};
```

83. Remove Duplicates from Sorted List Solved ✓

Easy Topics Companies

Given the `head` of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well.

Example 1:

```

graph LR
    1((1)) --> 1_2((1))
    1_2 --> 2((2))
    1_2 --> 1_3((1))
    1_3 --> 2
  
```

↓

```

graph LR
    1_4((1)) --> 2
  
```

9.1K 105 91 Online

Accepted 168 / 168 testcases passed

ArpitaShashni submitted at Aug 01, 2024 00:38

Runtime: 9 ms Beats 1.47% Memory: 16.83 MB Beats 11.36%

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head =

[1,1,2]

Problem 3: Reverse a linked list

```

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     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* reverseList(ListNode* head) {
        if(head==NULL || head->next==NULL){
            return head;
        }
        else{
            ListNode* temp1=NULL;
            ListNode* temp2=head;
            while(temp2!=NULL){
                ListNode* temp3=temp2->next;
                temp2->next=temp1;
                temp1=temp2;
                temp2=temp3;
            }
        }
    }
};

```

CONT_22CSH-359 - PROJECT B... x Chandigarh University Manage... x Introducing ChatGPT | OpenAI... x AbhayKejriwal04/22BCS14663... x Reverse Linked List - LeetCode... x +

leetcode.com/problems/reverse-linked-list/submissions/1339911953/

Problem List < >

Accepted

Code Accepted

All Submissions

Accepted 28 / 28 testcases passed

ArpitaShashni submitted at Aug 01, 2024 00:23

Editorial Solution

Runtime

4 ms | Beats 1.48%

Analyze Complexity

Memory

12.88 MB | Beats 99.98%

150%

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head =

[1, 2, 3, 4, 5]

22.4K 263 311 Online

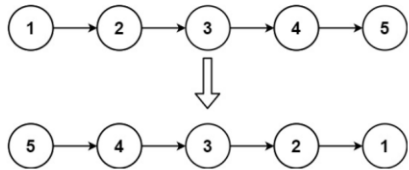
206. Reverse Linked List

Solved

Easy Topics Companies

Given the **head** of a singly linked list, reverse the list, and return the *reversed list*.

Example 1:



Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     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* deleteMiddle(ListNode* head) {
        if(head->next==NULL){
            return NULL;
        }
        ListNode* temp=head;
        int n=0;
        while(temp!=NULL){
            n++;
            temp=temp->next;
        }
    }
};
```

```

    }
    temp=head;
    int a=n/2;
    int x=0;
    while(temp!=NULL){
        x++;
        if(x==a){
            break;
        }
        temp=temp->next;
    }
    ListNode* c=temp->next;
    temp->next=c->next;
    delete c;
    return head;
}
};

```

Problem 5: Merge two sorted linked lists

```

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     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* solve(ListNode* &list1,ListNode* &list2){
        if(list1->next==NULL){
            list1->next=list2;
            return list1;
        }
        ListNode* curr1=list1;
        ListNode* curr2=list2;
        ListNode* next1=list1->next;
        ListNode* next2=list2->next;
        while(next1!=NULL && curr2!=NULL){
            if(curr1->val<=curr2->val && curr2->val<=next1->val){
                curr1->next=curr2;
                next2=curr2->next;
                curr2->next=next1;
                curr1=curr2;
                curr2=next2;
            }
            else{
                curr1=next1;
                next1=next1->next;
                if(next1==NULL){
                    curr1->next=curr2;
                    return list1;
                }
            }
        }
        return list1;
    }
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if(list1==NULL){
            return list2;
        }
        if(list2==NULL){
            return list1;
        }
        if(list1->val<=list2->val){
            return solve(list1,list2);
        }
        else{
            return solve(list2,list1);
        }
    }
}

```

```
};
```

Problem 6: Remove duplicates from sorted lists 2

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     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* deleteDuplicates(ListNode* head) {
        if (!head || !head->next) return head;

        ListNode* dummy = new ListNode(0);
        dummy->next = head;
        ListNode* prev = dummy;

        while (head) {
            if (head->next && head->val == head->next->val) {
                while (head->next && head->val == head->next->val)
                    head = head->next;
            }
            prev->next = head;
            prev = head;
            head = head->next;
        }
    }
};
```

```

        prev->next = head->next;
    } else {
        prev = prev->next;
    }
    head = head->next;
}
return dummy->next;
}
};

```

The screenshot displays the LeetCode interface for problem 82, "Remove Duplicates from Sorted List II". On the left, the problem description states: "Given the head of a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list. Return the linked list **sorted** as well." An example shows a linked list with nodes 1, 2, 3, 3, 4, 4, 5 being transformed into a linked list with nodes 1, 2, 5. The input is given as an array [1, 2, 3, 3, 4, 4, 5] and the output as [1, 2, 5].

On the right, the submission details are shown. The submission is marked as "Accepted" and was submitted by "ArpitaShashni" on Feb 02, 2025. The performance metrics are: Runtime: 0 ms (Beats 100.00%), Memory: 15.41 MB (Beats 99.67%). The test result section shows "Accepted" with a runtime of 0 ms for both Case 1 and Case 2. The input for the test case is [1, 2, 3, 3, 4, 4, 5].

Problem 7: Detect a cycle in a linked list

/**

* Definition for singly-linked list.

* struct ListNode {

* int val;

* ListNode *next;

* ListNode(int x) : val(x), next(NULL) {}

* };

*/

class Solution {

public:

bool hasCycle(ListNode *head) {

 ListNode* slow=head;

 ListNode* fast=head;

 if(head==NULL){

 return false;

 }

```

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

```

The screenshot shows a web browser with multiple tabs. The active tab is 'Linked List Cycle - LeetCode'. The URL is 'leetcode.com/problems/linked-list-cycle/submissions/1511416437/'. The page displays the '141. Linked List Cycle' problem, which is marked as 'Solved'. The problem description states: 'Given head, the head of a linked list, determine if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. **Note that pos is not passed as a parameter.** Return true if there is a cycle in the linked list. Otherwise, return false.' An example diagram shows a linked list with nodes 3, 2, 0, and -4, where the node with value 0 points back to the node with value 2, creating a cycle. The submission details show it was accepted by 'ArpitaShashni' on Jan 17, 2025, with a runtime of 12 ms (beats 40.43%) and memory usage of 11.68 MB (beats 96.43%). The test result for Case 1 shows the input 'head = [3,2,0,-4]' and the output 'Accepted'.

Problem 8: Reverse linked list 2

```

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     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* reverse(ListNode* head){
        ListNode* curr = head;
        ListNode* prev = NULL;
        ListNode* next = NULL;
        while (curr != NULL) {
            next = curr->next;
            curr->next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }
public:
    ListNode* reverseBetween(ListNode* head, int left, int right) {
        ListNode *revs = NULL, *revs_prev = NULL;
        ListNode *revend = NULL, *revend_next = NULL;
        int i = 1;
        ListNode* curr = head;
        while (curr && i <= right) {
            if (i < left)
                revs_prev = curr;
            if (i == left)
                revs = curr;
            if (i == right) {
                revend = curr;
                revend_next = curr->next;
            }
            curr = curr->next;
            i++;
        }
        revend->next = NULL;
        revend = reverse(revs);
        if (revs_prev)
            revs_prev->next = revend;
        else
            head = revend;
        revs->next = revend_next;
        return head;
    }
};

```

92. Reverse Linked List II Solved

Medium Topics Companies

Given the head of a singly linked list and two integers left and right where left ≤ right, reverse the nodes of the list from position left to position right, and return the reversed list.

Example 1:

Diagram showing a linked list with nodes 1, 2, 3, 4, 5. The nodes from position 2 to 4 are reversed, resulting in a linked list with nodes 1, 4, 3, 2, 5.

Accepted 44 / 44 testcases passed
ArpitaShashni submitted at Feb 05, 2025 23:51

Runtime: 0 ms | Beats 100.00%
Memory: 11.20 MB | Beats 73.26%

Testcase Test Result
Accepted Runtime: 0 ms
Case 1 Case 2
Input: head = [1, 2, 3, 4, 5]

Problem 9: Rotate a list

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     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* rotateRight(ListNode* head, int k) {
        if (head == NULL || head->next == NULL || k == 0) {
            return head;
        }
        ListNode* temp = head;
        int len = 0;
        while (temp != NULL) {
            len++;
            temp = temp->next;
        }
        k = k % len;
        if (k == 0) {
            return head;
        }
    }
};
```

```

    }
    ListNode* tail = head;
    for (int i = 1; i < len - k; i++) {
        tail = tail->next;
    }
    ListNode* newHead = tail->next;
    tail->next = NULL;
    temp = newHead;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = head;

    return newHead;
}
};

```

61. Rotate List Solved ✓

Medium Topics Companies

Given the `head` of a linked list, rotate the list to the right by `k` places.

Example 1:

rotate 1

rotate 2

Input: head = [1,2,3,4,5], k = 2
Output: [4,5,1,2,3]

10.2K 99 101 Online

Accepted 232 / 232 testcases passed
 ArpitaShashni submitted at Jan 17, 2025 18:35

Runtime: 0 ms | Beats 100.00%
 Memory: 16.40 MB | Beats 65.26%

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

head = [1,2,3,4,5]

Problem 10: Merge k sorted lists

```

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     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) {}
 * };

```

```

*/
#include<queue>
class Solution {
public:
    class compare{
    public:
        bool operator()(ListNode* a,ListNode* b){
            return a->val > b->val;
        }
    };
    ListNode* mergeKLists(vector<ListNode*> & lists) {
        priority_queue<ListNode*,vector<ListNode*>,compare> minheap;
        int k=lists.size();
        if(k==0){
            return NULL;
        }
        for(int i=0;i<k;i++){
            if(lists[i]!=NULL){
                minheap.push(lists[i]);
            }
        }
        ListNode* head=NULL;
        ListNode* tail=NULL;
        while(minheap.size()>0){
            ListNode* temp=minheap.top();
            minheap.pop();
            if(head==NULL){
                head=temp;
                tail=temp;
                if(head->next!=NULL){
                    minheap.push(tail->next);
                }
            }
            else{
                tail->next=temp;
                tail=temp;
                if(tail->next!=NULL){
                    minheap.push(tail->next);
                }
            }
        }
        return head;
    }
};

```


Problem 11: Sort list

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     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* getMiddle(ListNode* head) {
        ListNode* slow = head;
        ListNode* fast = head->next;
        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;
        }
        return slow;
    }
    ListNode* merge(ListNode* left, ListNode* right) {
        if (!left) return right;
        if (!right) return left;
        ListNode* dummy = new ListNode(0);
```

```

ListNode* curr = dummy;
while (left && right) {
    if (left->val < right->val) {
        curr->next = left;
        left = left->next;
    } else {
        curr->next = right;
        right = right->next;
    }
    curr = curr->next;
}
if (left) curr->next = left;
if (right) curr->next = right;
return dummy->next;
}

ListNode* sortList(ListNode* head) {
    if (!head || !head->next) return head;
    ListNode* mid = getMiddle(head);
    ListNode* rightHead = mid->next;
    mid->next = nullptr;
    ListNode* left = sortList(head);
    ListNode* right = sortList(rightHead);
    return merge(left, right);
}
};

```

The screenshot displays the LeetCode interface for the '148. Sort List' problem. On the left, the problem description states: 'Given the head of a linked list, return the list after sorting it in ascending order.' An example shows a linked list with nodes 4, 2, 1, 3 being transformed into a sorted list with nodes 1, 2, 3, 4. The input is given as 'head = [4,2,1,3]'. On the right, the submission results for user 'ArpitaShashni' are shown, indicating the solution was 'Accepted' with 30/30 testcases passed. Performance metrics include a runtime of 54 ms (beating 20.91%) and memory usage of 75.76 MB (beating 22.31%). The test result section shows 'Accepted' with a runtime of 0 ms for Case 1.