

# Assignment - 3

V Pranav - 192311029

## 21. Merge Two Sorted List

### Code:

```
class Solution:
    def mergeTwoLists(self, list1: Optional[ListNode], list2:
Optional[ListNode]) -> Optional[ListNode]:
        cur = dummy = ListNode()
        while list1 and list2:
            if list1.val < list2.val:
                cur.next = list1
                list1, cur = list1.next, list1
            else:
                cur.next = list2
                list2, cur = list2.next, list2
        if list1 or list2:
            cur.next = list1 if list1 else list2
        return dummy.next
```

### Screenshot:

The screenshot shows the LeetCode interface for the 'Merge Two Sorted Lists' problem. The top navigation bar includes links for Explore, Problems, Contest, Discuss, Interview, and Store. The problem description is on the left, showing a success message with runtime (37 ms) and memory usage (16.5 MB) statistics. Below this are 'Next challenges' and a 'Show off your acceptance' section with social media icons. A table of recent submissions is at the bottom left. The code editor on the right shows the Python solution, and the test case results on the bottom right show it was accepted with a runtime of 38 ms.

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:39	Accepted	37 ms	16.5 MB	python3
06/05/2024 21:39	Accepted	41 ms	16.4 MB	python3

**Time Complexity:**  $O(n)$

## 22. Generate Parenthesis

### Code:

```
class Solution(object):
    def generateParenthesis(self, n):
        def backtrack(S='', left=0, right=0):
            if len(S) == 2 * n:
                result.append(S)
                return
            if left < n:
                backtrack(S + '(', left + 1, right)
            if right < left:
                backtrack(S + ')', left, right + 1)
        result = []
        backtrack()
        return result
```

### Screenshot for I/O:

**Success** Details >

Runtime: 43 ms, faster than 22.19% of Python3 online submissions for Generate Parentheses.

Memory Usage: 16.7 MB, less than 98.21% of Python3 online submissions for Generate Parentheses.

Next challenges:

- Letter Combinations of a Phone Number
- Valid Parentheses
- Check if a Parentheses String Can Be Valid

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:47	Accepted	43 ms	16.7 MB	python3

```
1 class Solution(object):
2     def generateParenthesis(self, n):
3         def backtrack(S='', left=0, right=0):
4             if len(S) == 2 * n:
5                 result.append(S)
6                 return
7             if left < n:
8                 backtrack(S + '(', left + 1, right)
9             if right < left:
10                backtrack(S + ')', left, right + 1)
11        result = []
12        backtrack()
13        return result
14
```

Testcase Run Code Result Debuqger

**Accepted** Runtime: 49 ms

Your input: 3

Output: ["((()))","(()())","(())()","()(())","()()()"]

Expected: ["((()))","(()())","(())()","()(())","()()()"]

Console Use Example Testcases Run Code Submit

**Time Complexity:**  $O(n)$

## 23. Merge K Sorted Lists

### Code:

class Solution:

```
def mergeKLists(self, lists: List[ListNode]) -> ListNode:
    if not lists:
        return None
    if len(lists) == 1:
        return lists[0]
    mid = len(lists) // 2
    left = self.mergeKLists(lists[:mid])
    right = self.mergeKLists(lists[mid:])
    return self.merge(left, right)

def merge(self, l1, l2):
    dummy = ListNode(0)
    curr = dummy
    while l1 and l2:
        if l1.val < l2.val:
            curr.next = l1
            l1 = l1.next
        else:
            curr.next = l2
            l2 = l2.next
        curr = curr.next
    curr.next = l1 or l2
    return dummy.next
```

### Screenshot:

The screenshot displays the LeetCode interface for the 'Merge k Sorted Lists' problem. The submission is marked as 'Success' with a runtime of 75 ms, faster than 71.53% of Python3 submissions, and a memory usage of 19.2 MB, less than 98.30% of Python3 submissions. The code is written in Python3 and uses a recursive divide-and-conquer approach. The test case shows an input of [[1,4,5],[1,3,4],[2,6]] and an output of [1,1,2,3,4,4,5,6].

Runtime: 75 ms, faster than 71.53% of Python3 online submissions for Merge k Sorted Lists.

Memory Usage: 19.2 MB, less than 98.30% of Python3 online submissions for Merge k Sorted Lists.

Next challenges: [Ugly Number II](#) [Smallest Subarrays With Maximum Bitwise OR](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:49	Accepted	75 ms	19.2 MB	python3

```
1 class Solution:
2     def mergeKLists(self, lists: List[ListNode]) -> ListNode:
3         if not lists:
4             return None
5         if len(lists) == 1:
6             return lists[0]
7         mid = len(lists) // 2
8         left = self.mergeKLists(lists[:mid])
9         right = self.mergeKLists(lists[mid:])
10        return self.merge(left, right)
11
12    def merge(self, l1, l2):
13        dummy = ListNode(0)
14        curr = dummy
15        while l1 and l2:
16            if l1.val < l2.val:
17                curr.next = l1
18                l1 = l1.next
19            else:
20                curr.next = l2
21                l2 = l2.next
22            curr = curr.next
23        curr.next = l1 or l2
24        return dummy.next
```

Testcase: Run Code Result: Debugger

Accepted Runtime: 37 ms

Your input: [[1,4,5],[1,3,4],[2,6]]

Output: [1,1,2,3,4,4,5,6] [Diff](#)

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

Console [Use Example Testcases](#) [Run Code](#) [Submit](#)

**Time Complexity:  $O(n)$**

## 24. Swap Nodes in Pairs

### Code:

```
class Solution:
    def swapPairs(self, head: Optional[ListNode]) -> Optional[ListNode]:
        lst=[]
        while head:
            lst.append(head.val)
            head=head.next
        ans=[]
        for i in range(0,len(lst),2):
            val=lst[i:i+2]
            ans+=val[::-1]
        final=ListNode(0)
        tmp=final
        for i in ans:
            tmp.next=ListNode(i)
            tmp=tmp.next
        return final.next
```

### Screenshot:

The screenshot shows the LeetCode interface for the problem 'Swap Nodes in Pairs'. The left sidebar displays the problem status as 'Success' with details: Runtime 30 ms (faster than 87.31% of Python3 submissions) and Memory Usage 16.4 MB (less than 98.21% of Python3 submissions). It also lists next challenges: 'Reverse Nodes in k-Group' and 'Swapping Nodes in a Linked List'. The main content area shows the Python solution code, which is identical to the code provided in the 'Code' section. The right sidebar shows the test case results, indicating the solution is 'Accepted' with a runtime of 39 ms. The input is [1,2,3,4] and the output is [2,1,4,3], matching the expected output.

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:50	Accepted	30 ms	16.4 MB	python3

Testcase	Run Code Result	Debugger
Accepted	Runtime: 39 ms	

**Time Complexity:  $O(n)$**

## 25. Reverse Nodes in k-Group

### Code:

```
class Solution:
    def getLength(self, head):
        length = 0
        while head:
            length += 1
            head = head.next
        return length

    def reverseGroup(self, head, k, length):
        if length < k:
            return head
        curr = head
        prev = None
        next = None
        count = 0
        while curr and count < k:
            next = curr.next
            curr.next = prev
            prev = curr
            curr = next
            count += 1
        if next:
            head.next = self.reverseGroup(next, k, length - k)
        return prev

    def reverseKGroup(self, head: Optional[ListNode], k: int) -> Optional[ListNode]:
        length = self.getLength(head)
        return self.reverseGroup(head, k, length)
```

### Screenshot:

The screenshot shows a LeetCode submission page for the problem "Reverse Nodes in k-Group". The page is divided into two main sections: a left sidebar with problem details and a right section with the code editor and test results.

**Left Sidebar:**

- Success Details >
- Runtime: 50 ms, faster than 13.83% of Python3 online submissions for Reverse Nodes in k-Group.
- Memory Usage: 17.4 MB, less than 94.50% of Python3 online submissions for Reverse Nodes in k-Group.
- Next challenges:
  - Swapping Nodes in a Linked List
  - Reverse Nodes in Even Length Groups
- Show off your acceptance: [Facebook] [Twitter] [LinkedIn]
- Submission table:

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:51	Accepted	50 ms	17.4 MB	python3

**Right Section:**

- Code editor showing the Python solution for "Reverse Nodes in k-Group".
- Testcase tab selected, showing "Accepted" result with runtime 34 ms.
- Input: [1,2,3,4,5], 2
- Output: [2,1,4,3,5]
- Expected: [2,1,4,3,5]
- Buttons: Run Code, Submit

**Time Complexity:  $O(n)$**

## 26. Remove Duplicate from Sorted Array

### Code:

```
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:
        i,j=0,1
        while i<=j and j<len(nums):
            if nums[i]==nums[j]:
                j+=1
            else:
                nums[i+1]=nums[j]
                i+=1
        return i+1
```

### Screenshot:

The screenshot shows a LeetCode submission interface. On the left, the problem description is visible, showing a 'Success' status with a runtime of 72 ms and memory usage of 17.9 MB. Below this, a table lists the submission details:

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:53	Accepted	72 ms	17.9 MB	python3
06/05/2024 21:52	Accepted	78 ms	17.8 MB	python3

On the right, the code editor shows the Python solution. The code is as follows:

```
1 class Solution:
2     def removeDuplicates(self, nums: List[int]) -> int:
3         i,j=0,1
4         while i<=j and j<len(nums):
5             if nums[i]==nums[j]:
6                 j+=1
7             else:
8                 nums[i+1]=nums[j]
9                 i+=1
10        return i+1
11
```

Below the code editor, the test case results are shown. The status is 'Accepted' with a runtime of 34 ms. The input is [1,1,2], the output is [1,2], and the expected output is [1,2].

**Time Complexity:  $O(n)$**

## 27. Remove Element

### Code:

```
class Solution:
    def removeElement(self, nums: List[int], val: int) -> int:
        c=nums.count(val)
        while c!=0:
            nums.remove(val)
            c=c-1
        return len(nums)
```

### Screenshot:

The screenshot shows the LeetCode interface for the 'Remove Element' problem. The left sidebar displays the problem status as 'Success' with details: Runtime: 35 ms, faster than 72.96% of Python3 online submissions; Memory Usage: 16.5 MB, less than 42.49% of Python3 online submissions. It also lists next challenges: 'Remove Linked List Elements' and 'Move Zeroes'. A table shows two accepted submissions on 06/05/2024 at 21:54, with runtimes of 35 ms and 43 ms, and memory usage of 16.5 MB and 16.4 MB respectively. The main area shows the Python code for the solution, which uses a while loop to remove the target value. The right sidebar shows the test case results, indicating the solution is 'Accepted' with a runtime of 44 ms. The input is [3,2,2,3] and the output is [2,2].

Success Details >

Runtime: 35 ms, faster than 72.96% of Python3 online submissions for Remove Element.

Memory Usage: 16.5 MB, less than 42.49% of Python3 online submissions for Remove Element.

Next challenges:

Remove Linked List Elements Move Zeroes

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:54	Accepted	35 ms	16.5 MB	python3
06/05/2024 21:54	Accepted	43 ms	16.4 MB	python3

```
1 class Solution:
2     def removeElement(self, nums: List[int], val: int) -> int:
3         c=nums.count(val)
4         while c!=0:
5             nums.remove(val)
6             c=c-1
7         return len(nums)
```

Testcase Run Code Result Debugger

Accepted Runtime: 44 ms

Your input [3,2,2,3]  
3

Output [2,2] ☐ Diff

Expected [2,2]

Console Use Example Testcases

Run Code Submit

**Time Complexity:  $O(n)$**

## 28. Find the Index of the First Occurrence in a String

### Code:

```
class Solution(object):
    def strStr(self, haystack, needle):
        for i in range(len(haystack) - len(needle) + 1):
            if haystack[i : i+len(needle)] == needle:
                return i
        return -1
```

### Screenshot:

The screenshot shows a submission on a coding platform. The left sidebar displays the problem title, success status, runtime (35 ms), memory usage (16.5 MB), and next challenges. The main area shows the Python code for the solution. The right sidebar shows the test case results, including the input, output, and expected values.

**Success** Details >

Runtime: 35 ms, faster than 54.35% of Python3 online submissions for Find the Index of the First Occurrence in a String.

Memory Usage: 16.5 MB, less than 84.44% of Python3 online submissions for Find the Index of the First Occurrence in a String.

Next challenges: [Shortest Palindrome](#) [Repeated Substring Pattern](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:55	Accepted	35 ms	16.5 MB	python3

```
1 class Solution(object):
2     def strStr(self, haystack, needle):
3         for i in range(len(haystack) - len(needle) + 1):
4             if haystack[i : i+len(needle)] == needle:
5                 return i
6         return -1
7
```

Testcase Run Code Result Debugger

**Accepted** Runtime: 30 ms

Your input: "sadbutsad", "sad"

Output: 0

Expected: 0

Console Use Example Testcases Run Code Submit

**Time Complexity:  $O(n)$**



## 29. Divide Two Integers

### Code:

class Solution:

```
    def divide(self, dividend: int, divisor: int) -> int:
        sign = -1 if (dividend >= 0 and divisor < 0) or (dividend < 0 and
divisor >= 0) else 1
        dividend = abs(dividend)
        divisor = abs(divisor)
        result = len(range(0, dividend-divisor+1, divisor))
        if sign == -1:
            result = -result
        minus_limit = -(2**31)
        plus_limit = (2**31 - 1)
        result = min(max(result, minus_limit), plus_limit)
        return result
```

### Screenshot:

The screenshot displays the LeetCode interface for the 'Divide Two Integers' problem. The top navigation bar includes links for Explore, Problems, Contest, Discuss, Interview, and Store. The problem status is 'Success', and the runtime is 28 ms, faster than 94.87% of Python3 submissions. The memory usage is 16.6 MB, less than 67.82% of Python3 submissions. The next challenges are 'New 21 Game' and 'Subtract the Product and Sum of Digits of an Integer'. The submission table shows a successful submission on 06/05/2024 at 21:57 with a runtime of 28 ms and memory of 16.6 MB. The code editor on the right shows the Python solution, and the test case section shows the input (10, 3) and output (3).

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:57	Accepted	28 ms	16.6 MB	python3

**Time Complexity:  $O(n)$**

## 30. Substring with Concatenation of All Words

### Code:

```
class Solution:
    def calc(self, i):
        cnt = 0
        ind = i
        while (ind < i+self.pl):
            news = self.s[ind : ind + self.n]
            if news in self.dic :
                self.dic[news] -= 1
                if self.dic[news] == 0 :
                    cnt += 1
                    ind += self.n
            else :
                return False
        if cnt == len(self.dic) :
            return True
        else :
            return False
    def findSubstring(self, s: str, words: List[str]) -> List[int]:
        self.s = s
        self.n = len(words[0])
        d = {}
        for x in words :
            if x in d :
                d[x] += 1
            else :
                d[x] = 1
        self.pl = len(words)*len(words[0])
        ans = []
        i = 0
        while(i< len(s) - self.pl + 1):
            self.dic = {x : d[x] for x in d}
            if self.calc(i) :
                ans += [i]
                i += 1
            else :
                i += 1
        return ans
```

### Screenshot:

Explore Problems Contest Discuss Interview Store

Description Solution Discuss (999+) Submissions

Success Details

Runtime: 4473 ms, faster than 32.55% of Python3 online submissions for Substring with Concatenation of All Words.

Memory Usage: 17.1 MB, less than 93.73% of Python3 online submissions for Substring with Concatenation of All Words.

Next challenges: Minimum Window Substring

Show off your acceptance: f t in

Time Submitted	Status	Runtime	Memory	Language
06/05/2024 21:58	Accepted	4473 ms	17.1 MB	python3

Python3

```
1 cnt = 0
2 ind = i
3 while (ind < i+self.pl):
4     news = self.s[ind : ind + self.n]
5     if news in self.dic :
6         self.dic[news] -= 1
7         if self.dic[news] == 0 :
8             cnt += 1
9             ind += self.n
10    else :
11        return False
12 if cnt == len(self.dic) :
13     return True
14 else :
15     return False
16 def findSubstring(self, s: str, words: List[str]) -> List[int]:
17     self.s = s
18     self.n = len(words[0])
19     d = {}
20     for x in words :
21         if x in d :
22             d[x] += 1
23         else :
24             d[x] = 1
25     self.pl = len(words)*len(words[0])
26     ans = []
27     i = 0
28     while(i< len(s) - self.pl + 1):
29         self.dic = {x : d[x] for x in d}
30         if self.calc(i) :
31             ans += [i]
32             i += 1
33         else :
34             i += 1
35     return ans
```

Accepted Runtime: 39 ms

Your input: "barfoothefoobarman", ["foo", "bar"]

Output: [0, 9]

Expected: [0, 9]

Console Use Example Testcases Run Code Submit