Assignment 7.2

Comprehensive Report on LeetCode Experience

Introduction

LeetCode is a popular platform for coding practice and interview preparation, offering a wide range of problems that help enhance problem-solving and coding skills. Here, I summarize my experience with LeetCode by describing three problems I solved, the approaches I used, and the solutions I implemented in C++.

Problem 1: Two Sum

Description:

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice.

Approach:

Use a hash map to store the difference between the target and each element as the key, and the index of the element as the value.

Traverse the array and for each element,

check if it exists in the hash map.
If it exists, return the indices. If not, add the element to the hash map.
solution:

```
include < vector>
include < unorderedmap>
using namespace std

vector< int> twoSumvector< int> nums, int
target)
unorderedmap< int, int> hashmap
for (int i = 0 i < nums.size() + + i)
int complement = target - numsi
if (hashmap.find(complement) !=
hashmap.end())
return hashmapcomplement, i

hashmapnumsi = i

return
```

Explanation :

Initialize an unordered map to store elements and their indices.
Iterate through the array, calculating the complement for each element.

If the complement is found in the map, return the current index and the index of the complement.

Problem 2: Reverse Linked List

Description:

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

APPROACH :

Initialize three pointers: Prev as NULL, curr as head, and next as NULL.
Iterate through the list and reverse the links.
Update Prev to curr, curr to next, and next to curr-> next.
Return Prev as the new head.

solution:

```
struct ListNode
int val
ListNode next
ListNode(int x) : val(x), next(NULL)


ListNode reverseList(ListNode head)
ListNode Prev = NULL
ListNode curr = head
```

```
ListNode next = NULL
while (curr != NULL)
next = curr-> next
curr-> next = prev
prev = curr
curr = next

return prev
```

Explanation :

Initialize prev to NULL and curr to head.
Traverse the list while reversing the pointers.
Update curr and prev at each step to reverse the direction of the links.

problem 3: Longest substring without Repeating Characters

Description:

Given a string s, find the length of the longest substring without repeating characters.

Approach:

use a sliding window approach with two pointers.
maintain a set to keep track of characters in the current window.

move the right pointer to extend the window and the left pointer to shrink it when a duplicate character is found.
keep track of the maximum length of the window.

solution:

```
include < string>
include < unorderedset>
using namespace std

int lengthoflongestsubstringcstring s)
unorderedset< char>  charset
int left = 0, right = 0, maxlength = 0
while cright < s.length())
if ccharset.findcsright) = = charset.endc))
charset.insertcsright)
maxlength = maxcmaxlength, right - left +
1)

right+ +
  else
charset.erasecsleft)
left+ +


return maxlength
```

Explanation :

use an unordered set to keep track of characters in the current window.
Extend the window by moving the right pointer and add characters to the set.
If a duplicate is found, remove the leftmost character and move the left pointer.
Track the maximum window size without duplicates.

conclusion:
These problems demonstrate the use of hash maps for efficient lookup, pointer manipulation for linked list operations, and the sliding window technique for substring problems. LeetCode provides an excellent platform to practice and master such fundamental algorithms and data structures.