Assignment 7.2
Introduction to Leetcode

→ Leetcode is a popular online platform that offers a vast collection of coding problems designed to enhance one's problem-solving skills, particularly in the realm of algorithms and data structures.

Having solved over 450 question on leetcode, I have greatly enhanced my problem-solving abilities, algorithmic thinking, & technical interview praperdness. This report highlights three significant problems i encountered, the approaches, i employed & the solution i implemented, along with reflections on the positive experiences gained through this extensive practice.

Problem 1 - Two Sum -

→ problem Description -

The "two Sum" problem asks for two distinct indices of the numbers in an array that add up to a specific target. Given an array of integers 'nums' and an integers 'target', the goal is to return the indices of the two numbers such that they add up to 'target'.

Approach -

1. Brute Force Method - check all pairs of elements to find the pair that sums up to target. This method is simple but inefficent with a time complexity of $O(n^2)$.

2. HashMap Solution → Use a hash map to store the difference between the target & each element while iterating through the array. This method reduces time complexity to $O(n)$.

Code -

```
vector<int> twoSum (vector<int> &nums, int target)
{
    map<int, int>mp;
    vector<int>ans;
    for(int i=0; i<nums.size(); i++)
    {
        if(mp.find(target-nums[i]) != mp.end())
        {
            ans.push_back(mp[target-nums[i]]);
            ans.push_back(i);
            return ans;
        }
        mp[nums[i]] = i;
    }
    return ans;
}
```

Problem 2 - Longest substring without repeating characters.

Problem description -

Given a string 's', the task is to find the length of the longest substring without repeating characters. This problem is a classic example of efficient string manipulation.

Approach -

1. Sliding Window Technique - Use 2 pointers to represent the current window of characters, Expand the window by moving the right pointer pointer and shrink it by moving the left pointer & when a repeating characters is encountered.

Code -

```
int length (string s)
{
    int ans =0;
    int n= s.size();
    int i=0, j=0;
    map<char,int> mp;
    while(j<n)
    {
        mp[s[j]]++;
        if(mp[s[j]]>1)
        {
            while(i<j && mp[s[j]]>1)
            {
                mp[s[i]]--;
                i++;
            }
        }
        ans= max(ans, j-i+1);
        j++;
    }
    return ans;
}
```

Problem 3 - Merge Intervals -

problem Description -

Given a collection of intervals, the goal is to merge all overlapping interval, For example, given intervals =[[1,3], [2,6], [8,10], [15,18]], the result should be [[1,6], [8,10], [15,18]].

Approach -

1. Sort & merge - First, sort the intervals by their start times, Then, iterate through the intervals & merge them if they overlap.

2. Efficient merging -

Use a single list to store the merged intervals, updating the last interval in list if current interval overlaps.

Code ->

```
bool sortcol(vector<int> &a, vector<int> &b)
{ return a[0] < b[0];
}
vector<vector<int>> merge(vector<vector<int>> &inter) {
  vector<vector<int>> ans;
  sort( ~~ans~~ inter.begin(), inter.end(), sortcol);
 for(auto it : inter)
 {
     if (ans.empty())
     {  ans.push_back(it);
     else if (ans.back()[1] < it[0] || it[0] > ans.~~[illegible]~~back()[1])
     { ans.push_back(it);
     }
     else
     {  ans.back()[0] = min(it[0], ans.back()[0]);
       ans.back()[1] = max(it[1], ans.back()[1]);
     }
 }
 return ans;
```