Experience with LeetCode

LeetCode is an invaluable resource for honing coding skills, especially for preparing for technical interviews. With a vast array of problems categorized by difficulty, data structure, and algorithm, it provides a structured path to improve problem-solving capabilities. This report summarizes my experience on LeetCode and provides detailed explanations of three specific problems I solved, outlining the approach and solutions.

Problem 1-> Relative sort array (1122)

Link->Relative sort array

Problem ->

Given two arrays arr1 and arr2, the elements of arr2 are distinct, and all elements in arr2 are also in arr1.

Sort the elements of arr1 such that the relative ordering of items in arr1 are the same as in arr2. Elements that do not appear in arr2 should be placed at the end of arr1 in ascending order.

Approach ->

Brute force ->

Step 1- >Sort all element in arr1

Step2-> iterate in arr2 and for every same element in arr1 push it into answer vector and do it

Step 3-> now iterate in arr1 if any element in arr1 is not push in answer vector push it and return

Optimized ->

Step 1-> push freq of arr1 in map

Step2-> for every element in arr2 check freq in map and push that amount of elemnt in answer

Step 3-> now iterate in map and push all remaing elemnt in answer

```
class Solution {
public:
   vector<int> relativeSortArray(vector<int>& arr1, vector<int>& arr2) {
      ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
      unordered_map<int,int>mp;
      for(auto i:arr1)
```

```
    {
        mp[i]++; // insert all element in map
    }
    vector<int>ans;
    int j=0,a=0;
    for(auto i:arr2)  //  iterate in arr2
    {
        a=mp[i];
        mp.erase(i);
        j+=a;
        // keep track of element which are insted in this loop

        for(int j=0;j<a;j++)
        ans.push_back(i);  //  insert all element in answer vector
        // in order of arr2
    }
    for(auto i:mp)
    {
        a=i.second;
        for(int j=0;j<a;j++)
        ans.push_back(i.first);
        //insert remaing element which is not in arr2 in answer vector
    }
    sort(ans.begin()+j,ans.end());
    //sort only those element which are not in arr2 or
    //we can say sort only those which was inseted in last
    //for loop
    return v;
  }
};
```

Problem 2-> First Missing Positive (41)

Link -> first missing positive(hard)

Problem ->

Given an unsorted integer array nums. Return the smallest positive integer that is not present in nums.

You must implement an algorithm that runs in O(n) time and uses O(1) auxiliary space.

Approach ->

Brute force ->

Check from 0 -n+1 (n=size of array) if element is present in array so ok else it is smallest +ive missing

Optimize->

FIRST for loop->

make all negative number in array 0 as not needed.

SECOND FOR LOOP ->

iterate through every number and take its positve /abs value

let val=abs(nums[i])

if val<=size of array and val>=1

make val position value in array negative if it is zero make it -1*val

THIRD for loop->

itrate from start if number is not negative return that position

else if loop end return size +1 of array

```
class Solution {
public:
    int firstMissingPositive(vector<int>& nums) {
        ios_base::sync_with_stdio(false);cin.tie(NULL);
        int n=nums.size(),ans=1;
```

```
    for(int i=0;i<n;i++)
    {
        if(nums[i]<0)
        {
            nums[i]=0;
        }
    }
    for(int i=0;i<n;i++)
    {
        int a=abs(nums[i]);
        if(a<=n&&a>=1)
        {
            if(nums[a-1]>0)
            {
                nums[a-1]*=-1;
            }
            else if(nums[a-1]==0)
            {
                nums[a-1]=-1*(a);
            }
        }
    }
    for(int i=0;i<n;i++)
    {
        if(nums[i]>=0)
        {
            return i+1;
        }
    }
    return n+1;
}
```

};

Problem 3-> Distribute Coins in Binary Tree (979)

Link -> Distribution of coins

Problem ->

You are given the root of a binary tree with n nodes where each node in the tree has node.val coins. There are n coins in total throughout the whole tree.

In one move, we may choose two adjacent nodes and move one coin from one node to another. A move may be from parent to child, or from child to parent.

Return the minimum number of moves required to make every node have exactly one coin.

Approach ->

Brute force ->

Solve up every possible combination and check which fits best .

Optimized->

step 1-> start preorder traversal

strp 2-> let take variable c =0

setp3-> take it like that if node value is 1 so return c;

if node value is 0 so return c+1; -- (+ive indicate that we are asking for coins)

if node value is more than 1 return c-=(value-1) ; --- (ive indicate that extra coins are not required here so send them)

now what will happen that in parent when you encounter the return values -> so count = abs(ptr->left)+abs(ptr->right)

why abs ?

as they may be -ive value comming that means extra coins that we need to count how many of them are passed up which each take 1 step so we count all positve (need) and negative (not need) value.

last step is now after counting it take variable c= a + b (let a and b be return value from left and right).

this c will adjust itself to become the current need and now do step 2 for other node with this c value.

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
   int ans(TreeNode*ptr,int lvl,int &cnt)
   {
      if(ptr==NULL)
      return 0;
      int a=ans(ptr->left,lvl+1,cnt);
      int b=ans(ptr->right,lvl+1,cnt);
      cnt+=abs(a)+abs(b);
      if(ptr->val==0)
      a++;
      else if(ptr->val>1)
      a-=(ptr->val-1);
      return a+b;
   }
   int distributeCoins(TreeNode* root) {
```

```
        ios_base::sync_with_stdio(false);cin.tie(NULL);
        int cnt=0;
        ans(root,0,cnt);
        return cnt;
    }
};
```