leetcode tracking

**October 9, 2024**

Lc ques 921 C++

class Solution {

public:

    int minAddToMakeValid(string s) {

        int ans = 0;

        stack<char> st;

        if(s.length()==0) return 0;

        else{

            for(int i=0;i<s.length();i++){

                if(s[i] == '('){

                    st.push(s[i]);

                }

                else if(!st.empty() && st.top()=='(' && s[i]==')'){

                    st.pop();

                }

                else if(!st.empty() && s[i]==')'){

                    st.push(s[i]);

                }

                else if(st.empty() && s[i] == ')'){

                    st.push(s[i]);

                }

            }

            ans = st.size();

        }

        return ans;

    }

};

Data Structure Used - Stack

Cases to Ponder

- )()()

- ()()())

- ()((

- ))((

- (((()((

Lc ques 151 Python3

class Solution:

    def reverseWords(self, s: str) -> str:

        lst = s.split(" ")

        print(lst)

        lst = [x for x in lst if x != '']

        print(lst)

        answerstr = ''

        for x in reversed(lst):

            answerstr+=x

            answerstr+=" "

        answerstr = answerstr.rstrip().lstrip()

        return answerstr

DataStructure used - python list

Functions - reverse,split,strip

Cases to ponder

“ hello world ”

“ hello world ”

Lc ques 21 C++

class Solution {

public:

    int removeDuplicates(vector<int>& nums) {

        map<int,int> mp;

        vector<int> v;

        for(int i=0;i<nums.size();i++){

            mp[nums[i]] = i;

        }

        for(auto i:mp){

            v.push\_back(i.first);

        }

        int idx = 0;

        int count = 0;

        for(auto i:v){

            cout<<i<<endl;

            nums[idx++] = i;

        }

        count = v.size();

    return count;

    }

};

Ds - map,vector

I/p 1 1 2 3 4 4 4 5

O/p 1 2 3 4 5 1 4 4 // sorted unique elements

**October 10, 2024**

Lc ques 35

class Solution {

    int ans = 0;

    void binarySearch(vector<int> nums,int target,int s,int e){

        if(s<=e){

            int mid = s + (e-s)/2;

            if(target<nums[s]){

                ans = s;

            }

            else if(target>nums[e]){

                ans = e+1;

            }

            else if(target>nums[mid] && s!=e){

                binarySearch(nums,target,mid+1,e);

            }

            else if(target<nums[mid] && s!=e){

                binarySearch(nums,target,s,mid-1);

            }

            else if(target>nums[mid] && s==e){

                ans = s+1;

            }

            else if(target<nums[mid] && s==e){

                ans = e;

            }

            else if(target==nums[mid]){

                ans = mid;

            }

        }

    }

public:

    int searchInsert(vector<int>& nums, int target) {

        int s = 0;

        int e = nums.size()-1;

        if(target>nums[e]){

            return e+1;

        }

        else if(target<nums[s]){

            return s;

        }

        binarySearch(nums,target,s,e);

        cout<<ans<<endl;

        return ans;

    }

};

DataStructure - ?

Cases - item before start

Item after end

If start and end becomes equal // return ans

If they are not equal

Lc ques 66

class Solution:

    def plusOne(self, digits: List[int]) -> List[int]:

        ans = ''

        for i in digits:

            ans += str(i)

        ans = int(ans) +1

        ans = str(ans)

        res = [int(val) for val in str(ans)]

        return res

DataStructure used - Python lists,strings

Cases - 999

123

**October 11, 2024**

Lc ques 19

class Solution {

public:

    ListNode\* removeNthFromEnd(ListNode\* head, int n) {

        ListNode\* t1 = head,\*t2=head,\*t3;

        if(t1==NULL) return NULL;

        if(n==1 && t1->next==NULL){

            return NULL;

        }

        else{

            int count=0;

            while(t1!=NULL){

                t1=t1->next;

                count++;

            }

            if(count==n){

                head = head->next;

                return head;

            }

            int node = 1;

            int del = count-n;

            t1 = head;

            while(node!=del){

                t1=t1->next;

                node++;

            }

            t1->next = t1->next->next;

        }

    return head;

    }

};

DS - Linked List

**October 12, 2024**

Lc ques 219

class Solution {

public:

    bool containsNearbyDuplicate(vector<int>& nums, int k) {

        unordered\_map<int,int> mp;

        int flag = 0;

        for(int i=0;i<nums.size();i++){

            if(mp.count(nums[i])){

                if(abs(mp.at(nums[i])-i) <= k)

                flag++;

                else mp[nums[i]] = i;

            }

            else

            mp[nums[i]] = i;

        }

    if(flag>0) return true;

    else return false;

    }

};

DS - Map

Case - 1,0,1,1 , k = 1

**October 13, 2024**

Lc ques 2807

class Solution {

public:

    ListNode\* insertGreatestCommonDivisors(ListNode\* head) {

        if(head==NULL || head->next==NULL){

            return head;

        }

        else{

            ListNode \* first=head,\* second=head->next;

            while(second!=NULL){

                ListNode\* gcdnode = new ListNode(gcd(first->val,second->val));

                first->next = gcdnode;

                gcdnode->next = second;

                first = second;

                second = second->next;

            }

        }

    return head;

    }

};

DS - Linked List

October 14, 2024

2530

long long maxKelements(vector<int>& nums, int k) {

    if (nums.size() == 0) return 0;

    long long score = 0;

    priority\_queue<int> maxi;

    for (int num : nums) {

        maxi.push(num);

    }

    for (int i = 0; i < k; i++) {

        int maxelem = maxi.top();

        maxi.pop();

        score += maxelem;

        int newElem = (maxelem + 2) / 3;

        maxi.push(newElem);

    }

    return score;

    }

Pq

October 15, 2024

2938

long long minimumSteps(string s) {

        if(s.length()==0 || s.length()==1)

        return 0;

        if(s.length()==2 && s[0]=='0')

        return 0;

        long long count = 0;

        long long count1 = 0;

        long long count0 = 0;

        vector<long long> ans;

        for(int i=s.length()-1;i>=0;i--){

            if(s[i]=='1' && count0 !=0){

                count1++;

                count = count1\*count0;

                ans.push\_back(count);

                count1 = 0;

            }

            if(s[i]=='0'){

                count0++;

            }

        }

        count = 0;

        for(auto i:ans){

            count+=i;

        }

        return count;

    }

October 16, 2024

bool makeRotated(vector<int> nums,int n){

        int flag = 0;

        vector<int> arr;

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

            arr.push\_back(nums[i]);

        }

        for(int i=0;i<=n;i++){

            arr.push\_back(nums[i]);

        }

        for(int i=0;i<arr.size()-1;i++){

            if(arr[i]>arr[i+1]){

                return false;

            }

        }

        return true;

    }

public:

    bool check(vector<int>& nums) {

        int flag = 0;

        int idx = 0;

        bool ans;

        for(int i=0;i<nums.size()-1;i++){

            if(nums[i]>nums[i+1]){

                flag++;

                idx = i;

                ans = makeRotated(nums,i);

                break;

            }

        }

        cout<<flag<<endl;

        if(flag==0)

            return true;

return ans;

}

189

void rotate(vector<int>& nums, int k) {

        stack<int> st;

        vector<int> v;

        if(nums.size()==1 || nums.size()==0 || nums.size() == k){

            return;

        }

        if(nums.size()<k){

            k = k % nums.size();

        }

        for(int i=nums.size()-1;i>=nums.size()-k;i--){

            st.push(nums[i]);

        }

        while(!st.empty()){

            v.push\_back(st.top());

            st.pop();

        }

        for(int i=0;i<nums.size()-k;i++){

            v.push\_back(nums[i]);

        }

        for(int i=0;i<v.size();i++){

            nums[i] = v[i];

        }

    }

268

int missingNumber(vector<int>& nums) {

        SOL1

        // map<int,int> mp;

        // int ans=0;

        // for(int i=0;i<nums.size();i++){

        //     mp[nums[i]] = i;

        // }

        // for(int i=0;i<nums.size()+1;i++){

        //     if(!mp.contains(i)){

        //         ans = i;

        //     }

        // }

        //return ans;

        SOL 2

        // sort(nums.begin(),nums.end());

        // int num=0;

        // for(int n:nums){

        //     if(n!=num)

        //         return num;

        //     else num++;

        // }

        // return num;

        SOL 3

        // sort(nums.begin(),nums.end());

        // for(int i=0;i<nums.size();i++){

        //     if((nums[i]^i) !=0) return i;

        // }

        // if((nums[nums.size()-1]^nums.size())!=0){

        //     return nums.size();

        // }

        // return 0;

    }

485

int findMaxConsecutiveOnes(vector<int>& nums) {

        int count = 0;

        // vector<int> v;

        int maxi = 0;

        for(int i=0;i<nums.size();i++){

            if(nums[i]==1){

                count++;

                maxi = max(maxi,count);

            }

            else{

                // v.push\_back(count);

                count = 0;

            }

        }

        // v.push\_back(count);

        return maxi;

        // return \*max\_element(v.begin(),v.end());

    }

169

int moorevoting(vector<int> nums){

        int idx=0,cnt=1;

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

            if(nums[idx]==nums[i]){

                cnt++;

            }

            else{

                cnt--;

            }

            if(cnt==0){

                idx = i;

                cnt=1;

            }

        }

        return nums[idx];

    }

    bool MajorityOrNot(vector<int> nums,int elem){

        int cnt=0;

        for(int i=0;i<nums.size();i++)

            if(nums[i]==elem) cnt++;

        if(cnt>nums.size()/2) return 1;

        else return 0;

    }

public:

    int majorityElement(vector<int>& nums) {

        //moore voting algo

        int wannabeMajority = moorevoting(nums);

        //check if majority

        // bool isIt = MajorityOrNot(nums,wannabeMajority);

        /\*we don't need to check whether the elem is Maj or Not as

        question me likha h assume karo it's always present\*/

        return wannabeMajority;

    }

October 17, 2024

121

int maxi(int a,int b){

        if(a>b) return a;

        else return b;

    }

    int mini(int a,int b){

        if(b>a) return a;

        else return b;

    }

public:

    int maxProfit(vector<int>& prices) {

        int flag = 0;

        int max = 0;

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

            if(prices[i-1]<prices[i]){ // check if arr is desc ordered

                flag++;

            }

        }

        if(flag==0)

        return 0;

        else{

            int min = 100000;

            for(int i=0;i<prices.size();i++){

                min = mini(min,prices[i]);

                max = maxi(max,prices[i]-min);

            }

        }

    return max;

    }

11

int mini(int a,int b){

        if(a>b) return b;

        else return a;

    }

    int maxi(int a,int b){

        if(a>b) return a;

        else return b;

    }

public:

    int maxArea(vector<int>& height) {

        if(height.size()==0 || height.size()==1) return 0;

        int left=0,right = height.size()-1;

        int maxCap = 0,minHeight = 0;

        for(int i=0;i<height.size();i++){

            minHeight = mini(height[left],height[right]);

            maxCap = maxi(maxCap,(right-left)\*minHeight);

            if(minHeight==height[right]) right--;

            else left++;

        }

        return maxCap;

    }

October 18, 2024

22

vector<string> ans;

    void recursion(int numOpen,int numClose,string s,int n){

        if(s.length()==n\*2){

            ans.push\_back(s);

            return;

        }

        if(numOpen<n){

            s+='(';

            recursion(numOpen+1,numClose,s,n);

            s.pop\_back();

        }

        if(numOpen>numClose){

            s+=')';

            recursion(numOpen,numClose+1,s,n);

            s.pop\_back();

        }

    }

public:

    vector<string> generateParenthesis(int n) {

        string s = "";

        recursion(0,0,s,n);

    return ans;

    }

34

vector<int> ans = {-1, -1};

    int f = INT\_MAX,l = -1;

    int max(int a,int b){

        if(a>b) return a;

        return b;

    }

    int min(int a,int b){

        if(a>b) return b;

        return a;

    }

    void bs(vector<int> n,int t,int st,int end){

        if(st<=end){

            int mid = st + (end-st)/2;

            if(t==n[mid]){

                l = max(l,mid);

                f = min(f,mid);

                bs(n,t,mid+1,end);

                bs(n,t,st,mid-1);

            }

            else if(t>n[mid]){

                bs(n,t,mid+1,end);

            }

            else if(t<n[mid]){

                bs(n,t,st,mid-1);

            }

        }

    }

public:

    vector<int> searchRange(vector<int>& nums, int target) {

        bs(nums,target,0,nums.size()-1);

        if(f!=INT\_MAX){

            ans[0] = f;

        }

        if(l!=-1){

            ans[1] = l;

        }

        return ans;

    }