# Day1

## 1. Two Sum

CPP

class Solution {

public:

vector<int> twoSum(vector<int>& nums, int target)

{map<int,int> m;

vector<int> v;

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

{

if(m.find(target-nums[i])!=m.end())

{

v.push\_back(m[target-nums[i]]);

v.push\_back(i);

return v;

}

m[nums[i]]=i;

}

return v;

}

};

## 26. Remove Duplicates from Sorted Array

class Solution {

public:

int removeDuplicates(vector<int>& nums) {

if(nums.empty())

return 0;

int i=0,j=0;

while(++j<nums.size())

{

if(nums[j]!=nums[i])

nums[++i] =nums[j];

}

return i+1;

}

};

## 27. Remove Element

class Solution {

public:

int removeElement(vector<int>& nums, int val) {

int i=0,nind=0;

while(i<nums.size())

{

if(nums[i]!=val)

nums[nind++]=nums[i];

i++;

}

for(i=0;i<nind;i++)

cout<<nums[i]<<" ";

return nind;

}

};

## 46. Permutations

class Solution {

public:

vector<vector<int>> permute(vector<int>& nums) {

if(nums.size()<=1)

return{nums};

vector <vector<int>> result;

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

vector<int> v(nums.begin(),nums.end());

v.erase(v.begin()+i);

auto res=permute(v);

for(int j=0;j<res.size();j++)

{

vector<int> vec=res[j];

vec.insert(vec.begin(),nums[i]);

result.push\_back(vec);

}

}

return result;

}

};

## 189. Rotate Array

class Solution {

public:

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

k%=nums.size();

reverse(nums,0,nums.size()-1);

reverse(nums,0,k-1);

reverse(nums,k,nums.size()-1);

}

void reverse(vector<int>& nums,int start,int end)

{

while(start<end)

{

int temp=nums[start];

nums[start]=nums[end];

nums[end]=temp;

start++;

end--;

}

}

};

## 268. Missing Number

class Solution {

public:

void swap(int \* a,int \*b)

{

int t;

t=\*a;

\*a=\*b;

\*b=t;

}

int missingNumber(vector<int>& nums) {

int i,j,sum=0;

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

sum+=nums[i];

int val=nums.size()\*(nums.size()+1)/2 -sum;

return val;

}

};

## 283. Move Zeroes

class Solution {

public:

void moveZeroes(vector<int>& nums) {

int i=0,count=0;

while(i<nums.size())

{

if(nums[i]!=0)

nums[count++]=nums[i];

i++;

}

while(count<nums.size())

nums[count++]=0;

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

cout<<nums[i]<<" ";

}

};

## 287. Find the Duplicate Number

class Solution {

public:

int findDuplicate(vector<int>& nums) {

int tortoise = nums[0];

int hare = nums[0];

do {

tortoise = nums[tortoise];

hare = nums[nums[hare]];

} while (tortoise != hare);

// Find the "entrance" to the cycle.

tortoise = nums[0];

while (tortoise != hare) {

tortoise = nums[tortoise];

hare = nums[hare];

}

return hare;

}

};

## 442. Find All Duplicates in an Array

class Solution {

public List<Integer> findDuplicates(int[] nums) {

List<Integer> arr = new ArrayList<>();

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

{

int index = Math.abs(nums[i])-1;

if(nums[index]<0)

arr.add(index+1);

else

nums[index]=-nums[index];

}

return arr;

}

}

## 509. Fibonacci Number

class Solution {

public:

int fib(int n) {

if(n < 2) return n;

int prev1 = 1, prev2 = 0;

for(int i = 2; i <= n; i++) {

// Add up 2 most recent nums

int temp = prev1;

prev1 += prev2;

prev2 = temp;

}

return prev1;

}

};

## 560. Subarray Sum Equals K

class Solution {

public int subarraySum(int[] nums, int k) {

int count=0,sum=0;

HashMap<Integer,Integer> map=new HashMap<>();

map.put(0,1);

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

{

sum+=nums[i];

if(map.containsKey(sum-k))

count+=map.get(sum-k);

map.put(sum,map.getOrDefault(sum,0)+1);

}

return count;

}

}

## 974. Subarray Sums Divisible by K

class Solution {

public:

int subarraysDivByK(vector<int>& nums, int k) {

int mod[k];

memset(mod, 0, sizeof(mod));

int cumSum = 0;

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

cumSum += nums[i];

mod[((cumSum % k) + k) % k]++;

}

int result = 0;

for (int i = 0; i < k; i++)

if (mod[i] > 1)

result += (mod[i] \* (mod[i] - 1)) / 2;

result += mod[0];

return result;

}

};

## 1590. Make Sum Divisible by P

class Solution {

public:

int minSubarray(vector<int>& nums, int p) {

int N = nums.size();

vector<int> pre(N);

int cur = 0;

for (int i=0; i<N; i++) {

cur += nums[i];

cur %= p;

pre[i] = cur;

}

if (pre.back()%p == 0) return 0;

int ans = INT\_MAX;

int target = pre.back()%p;

unordered\_map<int, int> mp;

mp[0] = -1;

for (int i=0; i<N; i++) {

int x = pre[i]-target;

if (x < 0) x += p;

if (mp.count(x) and (i!=N-1 or mp[x]!=-1)) {

ans = min(ans, i-mp[x]);

}

mp[pre[i]] = i;

}

return ans == INT\_MAX ? -1 : ans;

}

};