***Week – 3 (14.04.2021 – 18.04.2021)***

***RANDOM CODES***

1. ***Single Number:***

class Solution {

public:

int singleNumber(vector<int>& nums) {

int i;

map<int,int> m;

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

m[nums[i]]++;

for(auto e : m)

if(e.second == 1) return e.first;

return 0;

}

};

1. ***Set Mismatch:***

class Solution {

public:

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

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

vector<int> res;

int i, missing, dup;

map<int,int> m;

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

m[i] = 0;

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

m[nums[i]]++;

for(auto e : m)

{

if(e.second == 0) missing = e.first;

if(e.second == 2) dup = e.first;

}

res.push\_back(dup);

res.push\_back(missing);

return res;

}

};

1. ***Smallest Integer Divisible by K:***

class Solution {

public:

int smallestRepunitDivByK(int K) {

if(K%2 == 0 || K%5 == 0) return -1;

int i=1;

long long int num = 1;

while(num%K != 0)

{

num = (num\*10 + 1)%K;

i++;

}

return i;

}

};

1. ***Smallest Range I:***

class Solution {

public:

int smallestRangeI(vector<int>& A, int K) {

return max(0, \*max\_element(A.begin(),A.end()) - \*min\_element(A.begin(),A.end()) - 2\*K);

}

};

1. ***Calculate Money in Leetcode Bank:***

class Solution {

public:

int totalMoney(int n) {

int i=1, j=0, sum=0, day=1;

while(day<=n)

{

i = 1;

while(i<=7 && day<=n)

{

sum = sum + i + j;

i++;

day++;

}

j++;

}

return sum;

}

};

1. ***Subtract the Product and Sum of Digits of an Integer:***

class Solution {

public:

int subtractProductAndSum(int n) {

int sum=0, prod=1, rem;

while(n>0)

{

rem = n%10;

sum = sum + rem;

prod = prod \* rem;

n= n/10;

}

return (prod-sum);

}

};

1. ***Search in Rotated Sorted Array II:***

class Solution {

public:

bool search(vector<int>& nums, int target) {

int i;

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

if(nums[i] == target) return true;

return false;

}

};

1. ***Height Checker:***

class Solution {

public:

int heightChecker(vector<int>& heights) {

vector<int> h = heights;

sort(h.begin(), h.end());

int i, count = 0;

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

if(h[i] != heights[i]) count++;

return count;

}

};

1. ***Partition Array Into Three Parts With Equal Sum:***

class Solution {

public:

bool canThreePartsEqualSum(vector<int>& arr) {

int i=0, sum = 0, s=0, j=0;

sum = accumulate(arr.begin(), arr.end(), sum);

if(sum%3 != 0) return false;

sum = sum/3;

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

{

s = s + arr[i];

if(s == sum)

{

s = 0;

j++;

if(j == 3) break;

}

}

if(j==3) return true;

else return false;

}

};

1. ***Sum of Even Numbers After Queries:***

class Solution {

public:

vector<int> sumEvenAfterQueries(vector<int>& A, vector<vector<int>>& queries) {

vector<int> res;

int i, j, sum=0;

for(j=0; j<A.size(); j++)

if(A[j]%2 == 0) sum+=A[j];

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

{

if(A[queries[i][1]]%2 == 0) sum = sum - A[queries[i][1]];

A[queries[i][1]] = A[queries[i][1]] + queries[i][0];

if(A[queries[i][1]]%2 == 0) sum = sum + A[queries[i][1]];

res.push\_back(sum);

}

return res;

}

};

1. ***Flip String to Monotone Increasing:***

class Solution {

public:

int minFlipsMonoIncr(string s) {

int i, one=0, zero=0;

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

{

if(s[i] == '1') one++;

else zero++;

zero = min(zero, one);

}

return zero;

}

};

1. ***Richest Customer Wealth:***

class Solution {

public:

int maximumWealth(vector<vector<int>>& accounts) {

int i, j, sum, maxwel = INT\_MIN;

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

{

sum = accumulate(accounts[i].begin(), accounts[i].end(), 0);

maxwel = max(maxwel, sum);

}

return maxwel;

}

};

1. ***Shift 2D Grid:***

class Solution {

public:

vector<vector<int>> shiftGrid(vector<vector<int>>& grid, int k) {

if(k==0) return grid;

vector<vector<int>> res(grid.size(), vector<int>(grid[0].size(), 0));

int i,j,l;

for(l=0; l<k; l++)

{

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

for(j=0; j<grid[0].size()-1; j++)

res[i][j+1] = grid[i][j];

for(i=0; i<grid.size()-1; i++)

res[i+1][0] = grid[i][grid[0].size()-1];

res[0][0] = grid[grid.size()-1][grid[0].size()-1];

grid = res;

}

return res;

}

};

1. ***Check If It Is a Straight Line:***

class Solution {

public:

bool checkStraightLine(vector<vector<int>>& coordinates) {

int i, x0, y0, dx, dy, x, y;

x0 = coordinates[0][0];

y0 = coordinates[0][1];

dx = coordinates[1][0] - x0;

dy = coordinates[1][1] - y0;

for(i=1; i<coordinates.size(); i++)

{

x = coordinates[i][0];

y = coordinates[i][1];

if(dx \* (y - y0) != dy \* (x - x0)) return false;

}

return true;

}

};

1. ***Can Place Flowers:***

class Solution {

public:

bool canPlaceFlowers(vector<int>& flowerbed, int n) {

int i, count=0;

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

{

if((flowerbed[i]==0 && (i==0 || flowerbed[i-1]==0)) && (i==flowerbed.size()-1 || flowerbed[i+1] == 0))

{

flowerbed[i] = 1;

count++;

}

}

if(count >= n) return true;

else return false;

}

};