***Week – 2 (05.04.2021 – 13.04.2021)***

***CODES BASED ON WEELKY TASK***

1. ***Sort Array By Parity II:***

class Solution {

public:

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

vector<int> even, odd, res;

if(nums.size() == 1) return nums;

int i;

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

{

if(nums[i]%2 == 0) even.push\_back(nums[i]);

else odd.push\_back(nums[i]);

}

i=0;

while(i<even.size())

{

res.push\_back(even[i]);

res.push\_back(odd[i]);

i++;

}

return res;

}

};

1. ***Intersection of Two Arrays:***

class Solution {

public:

vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {

sort(nums1.begin(), nums1.end());

sort(nums2.begin(), nums2.end());

vector<int> res;

int i=0,j=0;

while(i<nums1.size() && j<nums2.size())

{

if(nums1[i] < nums2[j])

{

i++;

}

else if(nums1[i] > nums2[j])

{

j++;

}

else

{

res.push\_back(nums1[i]);

i++;

j++;

}

}

res.erase(unique(res.begin(), res.end()), res.end());

return res;

}

};

1. ***Intersection of Two Arrays II:***

class Solution {

public:

vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {

sort(nums1.begin(), nums1.end());

sort(nums2.begin(), nums2.end());

vector<int> res;

int i=0,j=0;

while(i<nums1.size() && j<nums2.size())

{

if(nums1[i] < nums2[j])

{

i++;

}

else if(nums1[i] > nums2[j])

{

j++;

}

else

{

res.push\_back(nums1[i]);

i++;

j++;

}

}

res.erase(unique(res.begin(), res.end()), res.end());

return res;

}

};

1. ***Relative Sort Array: (***Can also be solved using map)

class Solution {

public:

vector<int> relativeSortArray(vector<int>& arr1, vector<int>& arr2) {

vector<int> res, block;

int i,j;

bool c;

for(i=0; i<arr2.size(); i++)

for(j=0; j<arr1.size(); j++)

if(arr2[i] == arr1[j])

res.push\_back(arr1[j]);

for(i=0; i<arr1.size(); i++)

{

c = false;

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

{

if(arr1[i] == res[j]) c=true;

}

if(c == false) block.push\_back(arr1[i]);

}

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

for(i=0; i<block.size(); i++) res.push\_back(block[i]);

return res;

}

};

1. ***Maximum Product of Two Elements in an Array:***

class Solution {

public:

int maxProduct(vector<int>& nums) {

vector<int> prod;

int i,j;

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

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

prod.push\_back((nums[i]-1) \* (nums[j]-1));

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

}

};

1. ***Two Sum II - Input array is sorted:***

class Solution {

public:

vector<int> twoSum(vector<int>& numbers, int target) {

vector<int> res;

int i, j, sum;

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

{

if(numbers[i] == 0 && numbers[i+1] == 0 && target == 0)

{

res.push\_back(i+1);

res.push\_back(i+2);

return res;

}

if(numbers[i] == 0 && numbers[i+1] == 0) continue;

for(j=i+1; j<numbers.size(); j++)

{

if(numbers[i] + numbers[j] == target)

{

res.push\_back(i+1);

res.push\_back(j+1);

return res;

}

}

}

return res;

}

};

1. ***Find the Duplicate Number:***

class Solution {

public:

int findDuplicate(vector<int>& nums) {

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

int i;

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

if(nums[i] == nums[i+1]) return nums[i];

return -1;

}

};

1. ***Count Odd Numbers in an Interval Range:***

class Solution {

public:

int countOdds(int low, int high) {

int i, count = 0;

for(i=low; i<=high; i++)

if(i%2!=0) count++;

return count;

}

};

1. ***Three Consecutive Odd:***

class Solution {

public:

bool threeConsecutiveOdds(vector<int>& arr) {

if(arr.size() < 3) return false;

int i;

for(i=0; i<arr.size()-2; i++)

if(arr[i]%2!=0 && arr[i+1]%2!=0 && arr[i+2]%2!=0) return true;

return false;

}

};

1. ***Find Numbers with Even Number of Digits:***

class Solution {

public:

int findNumbers(vector<int>& nums) {

vector<int> dig;

int i, count, d\_count;

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

{

count = 0;

while(nums[i] > 0)

{

count++;

nums[i] = nums[i]/10;

}

if(count%2 == 0) dig.push\_back(count);

}

return dig.size();

}

};

1. ***Rotate Array:***

class Solution {

public:

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

k = k%nums.size();

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

reverse(nums.begin(), nums.begin()+k);

reverse(nums.begin()+k, nums.end());

}

};

1. ***Transpose Matrix:***

class Solution {

public:

vector<vector<int>> transpose(vector<vector<int>>& matrix) {

vector<vector<int>> res;

int i,j;

for(i=0; i<matrix[0].size(); i++)

{

vector<int> v(matrix.size());;

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

v[j] = matrix[j][i];

res.push\_back(v);

}

return res;

}

};

1. ***Shuffle the Array:***

class Solution {

public:

vector<int> shuffle(vector<int>& nums, int n) {

vector<int> res;

int i;

for(i=0; i<nums.size()/2; i++)

{

res.push\_back(nums[i]);

res.push\_back(nums[i+(nums.size()/2)]);

}

return res;

}

};

1. ***Valid Anagram:***

class Solution {

public:

bool isAnagram(string s, string t) {

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

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

if(s == t) return true;

else return false;

}

};

1. ***Trapping Rain Water:***

class Solution {

public:

int trap(vector<int>& height) {

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

int i,j,l,r,sum=0;

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

{

l = height[i];

for(j=0; j<i; j++) l = max(l,height[j]);

r = height[i];

for(j=i+1; j<height.size(); j++) r = max(r,height[j]);

sum = sum + (min(l,r) - height[i]);

}

return sum;

}

};

1. ***Remove All Adjacent Duplicates In String:***

class Solution {

public:

string removeDuplicates(string S) {

int i;

bool isdup = false;

for(i=0; i<S.size()-1 && S.size()!=0; i++)

{

if(S[i] == S[i+1])

{

S.erase(i,2);

i = (i) ? i-2 : i-1;

}

}

return S;

}

};

1. ***Permutations:***

class Solution {

public:

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

if(nums.size() <= 1) return {nums};

vector<vector<int>> res;

int i,j;

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

{

vector<int> v1 = nums;

v1.erase(v1.begin()+i);

auto r = permute(v1);

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

{

vector<int> v2 = r[j];

v2.insert(v2.begin(), nums[i]);

res.push\_back(v2);

}

}

return res;

}

};

1. ***Permutations II:***

class Solution {

public:

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

if(nums.size() <= 1) return {nums};

vector<vector<int>> res;

int i,j;

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

{

vector<int> v1 = nums;

v1.erase(v1.begin()+i);

auto r = permuteUnique(v1);

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

{

vector<int> v2 = r[j];

v2.insert(v2.begin(), nums[i]);

res.push\_back(v2);

}

}

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

res.erase(unique(res.begin(), res.end()), res.end());

return res;

}

};

1. ***Global and Local Inversions:***

class Solution {

public:

bool isIdealPermutation(vector<int>& A) {

int i;

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

if(abs(A[i] - i) > 1) return false;

return true;

}

};

1. ***Pascal's Triangle:***

class Solution {

public:

vector<vector<int>> generate(int numRows) {

vector<vector<int>> res(numRows);

int i,j;

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

{

res[i].resize(i+1);

res[i][0] = res[i][i] = 1;

for(j=1; j<i; j++)

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

}

return res;

}

};

1. ***Pascal's Triangle II:***

class Solution {

public:

vector<int> getRow(int rowIndex) {

vector<vector<int>> res(rowIndex+1);

int i,j;

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

{

res[i].resize(i+1);

res[i][0] = res[i][i] = 1;

for(j=1; j<i; j++)

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

}

return res[rowIndex];

}

};

1. ***Set Matrix Zeroes:***

class Solution {

public:

void setZeroes(vector<vector<int>>& matrix) {

vector<vector<int>> check = matrix;

int i,j,k;

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

for(j=0; j<matrix[i].size(); j++)

if(matrix[i][j] == 0)

{

for(k=0; k<matrix.size(); k++)

check[k][j] = 0;

for(k=0; k<matrix[i].size(); k++)

check[i][k] = 0;

}

matrix = check;

}

};

1. ***Diagonal Traverse II:***

class Solution {

public:

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

vector<int> res;

int i=0 ,j=0;

map<int, vector<int>> m;

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

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

m[i+j].push\_back(nums[i][j]);

for(auto i : m)

{

reverse(i.second.begin(), i.second.end());

for(auto e : i.second)

res.push\_back(e);

}

return res;

}

};

1. ***Sum of All Odd Length Subarrays:***

class Solution {

public:

int sumOddLengthSubarrays(vector<int>& arr) {

int sum = 0, i, j, k;

for(i=0; i<arr.size(); i=i+2)

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

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

sum = sum + arr[k+j];

return sum;

}

};

1. ***Rotate Image:***

class Solution {

public:

void rotate(vector<vector<int>>& matrix) {

vector<vector<int>> m =matrix;

int i,j;

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

{

for(j=0; j<matrix[0].size(); j++)

m[i][j] = matrix[j][i];

reverse(m[i].begin(), m[i].end());

}

matrix = m;

}

};

1. ***Diagonal Traverse:***

class Solution {

public:

vector<int> findDiagonalOrder(vector<vector<int>>& matrix) {

vector<int> res;

int i=0 ,j=0;

map<int, vector<int>> m;

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

for(j=0; j<matrix[i].size();j++)

m[i+j].push\_back(matrix[i][j]);

for(auto i : m)

{

if(i.first%2 == 0) reverse(i.second.begin(), i.second.end());

for(auto e : i.second)

res.push\_back(e);

}

return res;

}

};

1. ***Unique Number of Occurrences:***

class Solution {

public:

bool uniqueOccurrences(vector<int>& arr) {

int i,p=1;

map<int, int> m;

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

m[arr[i]]++;

vector<int> v;

for(auto e : m)

v.push\_back(e.second);

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

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

if(v[i] == v[i+1]) return false;

return true;

}

};

1. ***Move Zeroes:***

class Solution {

public:

void moveZeroes(vector<int>& nums) {

if(nums.size() < 2) return;

vector<int> num, zeros, res;

int i;

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

{

if(nums[i] == 0) zeros.push\_back(nums[i]);

else num.push\_back(nums[i]);

}

res = num;

for(i=0; i<zeros.size(); i++) res.push\_back(0);

nums = res;

}

};

1. ***Max Consecutive Ones:***

class Solution {

public:

int findMaxConsecutiveOnes(vector<int>& nums) {

int i, m=0, mx=0;

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

{

if(nums[i] == 1) m++;

else

{

mx = max(mx, m);

m = 0;

}

}

mx = max(mx, m);

return mx;

}

};

1. ***Find All Duplicates in an Array:***

class Solution {

public:

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

if(nums.size() < 2) return {};

vector<int> res;

int i;

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

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

if(nums[i] == nums[i+1]) res.push\_back(nums[i]);

return res;

}

};

1. ***Reshape the Matrix:***

class Solution {

public:

vector<vector<int>> matrixReshape(vector<vector<int>>& nums, int r, int c) {

if(r\*c != nums.size()\*nums[0].size()) return nums;

int i, j, k=0;

vector<vector<int>> res;

vector<int> v;

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

for(j=0; j<nums[0].size(); j++)

v.push\_back(nums[i][j]);

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

{

vector<int> v1(c);

for(j=0; j<c; j++)

{

v1[j] = v[k];

k++;

}

res.push\_back(v1);

}

return res;

}

};

1. ***Strings:***

#include <iostream>

#include <string>

using namespace std;

int main() {

    // Complete the program

    string str1, str2;

    int i;

    cin>>str1>>str2;

    cout<<str1.length()<<" "<<str2.length()<<"\n"<<str1+str2<<"\n";

    char c1 = str1[0], c2 = str2[0];

    cout<<c2;

    for(i=1; i<str1.length(); i++) cout<<str1[i];

    cout<<" ";

    cout<<c1;

    for(i=1; i<str2.length(); i++) cout<<str2[i];

    return 0;

}

1. ***Simple Array Sum:***

int simpleArraySum(vector<int> ar) {

     int i=0, sum=0;

     for(i=0; i<ar.size(); i++)

        sum = sum + ar[i];

    return sum;

}

1. ***Compare the Triplets:***

vector<int> compareTriplets(vector<int> a, vector<int> b) {

    vector<int> res;

    int i, Alice=0, Bob=0;

    for(i=0; i<a.size(); i++)

    {

        if(a[i] > b[i]) Alice++;

        else if(a[i] < b[i]) Bob++;

        else continue;

    }

    res.push\_back(Alice);

    res.push\_back(Bob);

    return res;

}

1. ***A Very Big Sum:***

long aVeryBigSum(vector<long> ar) {

    long sum=0;

    int i;

    for(i=0; i<ar.size(); i++) sum = sum + ar[i];

    return sum;

}

1. ***Spiral Matrix:***

class Solution {

public:

vector<int> spiralOrder(vector<vector<int>>& matrix) {

vector<int> res;

int startx = 0, starty = 0, endx = matrix.size()-1, endy = matrix[0].size()-1, i;

while(startx<=endx && starty<=endy)

{

for(i=starty; i<=endy; i++)

res.push\_back(matrix[startx][i]);

for(i=startx+1; i<=endx; i++)

res.push\_back(matrix[i][endy]);

if(startx < endx && starty < endy)

{

for(i=endy-1; i>=starty; i--)

res.push\_back(matrix[endx][i]);

for(i=endx-1; i>=startx+1; i--)

res.push\_back(matrix[i][starty]);

}

startx++;

starty++;

endx--;

endy--;

}

return res;

}

};

1. ***Maximum Subarray:***

class Solution {

public:

int maxSubArray(vector<int>& nums) {

int i, sum = 0, m=INT\_MIN;

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

{

sum = max(nums[i], sum+nums[i]);

m = max(m, sum);

}

return m;

}

};

1. ***Find Pivot Index:***

class Solution {

public:

int pivotIndex(vector<int>& nums) {

int i, sum=0, l, r;

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

l = 0;

r = sum;

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

{

if(l == r-nums[i]) return i;

l = l + nums[i];

r = r - nums[i];

}

return -1;

}

};

1. ***Lexicographical Numbers:***

class Solution {

public:

vector<int> lexicalOrder(int n) {

vector<string> s;

vector<int> v;

int i;

for(i=1; i<=n; i++)

s.push\_back(to\_string(i));

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

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

v.push\_back(stoi(s[i]));

return v;

}

};

1. ***Reverse Words in a String:***

class Solution {

public:

string reverseWords(string s) {

if(s.size() == 1 && s[0]!=' ') return s;

if((s.size() == 1 && s[0]==' ') || s.size() == 0) return NULL;

int i = 0, j = s.size()-1, k = 0, re=0;

string res="", result="";

while(i < s.size())

{

if(s[i] != ' ') break;

i++;

}

while(j >= 0)

{

if(s[j] != ' ') break;

j--;

}

while(i <= j)

{

if(s[i] == ' ')

{

if(i==1 || i!=re+1)

{

if(result.size()>0) result = res + ' ' + result;

else result = res;

}

res ="";

re = i;

}

else res = res+s[i];

i++;

}

if(result.size() == 0) result = res;

else result = res + ' ' + result;

return result;

}

};

1. ***Find All Numbers Disappeared in an Array:***

class Solution {

public:

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

vector<int> res;

int j, k=1;

map<int, int> m;

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

m[j] = 0;

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

m[nums[j]]++;

for(auto i : m)

if(i.second < 1) res.push\_back(i.first);

return res;

}

};

1. ***Longest Consecutive Sequence:***

class Solution {

public:

int longestConsecutive(vector<int>& nums) {

if(nums.size() <2) return nums.size();

int i, j, mx=INT\_MIN, count = 1;

bool check = true;

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

nums.erase(unique(nums.begin(), nums.end()), nums.end());

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

{

check = false;

if(nums[i] + 1 == nums[i+1])

{

count++;

check = true;

}

else

{

mx = max(mx,count);

count=1;

}

}

mx = max(mx,count);

return mx;

}

};