***Week – 5 (26.04.2021 – 02.05.2021)***

***CODES BASED ON WEELKY TASK***

1. ***Search a 2D Matrix II:***

class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

int i=0, j=matrix[0].size()-1;

while(i>=0 && j>=0 && i<matrix.size() && j<matrix[0].size())

{

if(matrix[i][j] == target) return true;

else if(matrix[i][j] < target) i++;

else j--;

}

return false;

}

};

1. ***Beautiful Array:***

class Solution {

public:

int part(vector<int> &arr, int a, int b, int mask)

{

int i, j = a;

for(i=a; i<=b; i++)

{

if((arr[i] & mask) != 0)

{

swap(arr[i], arr[j]);

j++;

}

}

return j;

}

void sort(vector<int> &arr, int a, int b, int mask)

{

if(a>=b) return;

int mid = part(arr, a, b, mask);

sort(arr, a, mid-1, mask<<1);

sort(arr, mid, b, mask<<1);

}

vector<int> beautifulArray(int n) {

vector<int> res;

int i=0;

for(i=0; i<n; i++) res.push\_back(i+1);

sort(res, 0, n-1, 1);

return res;

}

};

1. ***Median of Two Sorted Arrays:***

class Solution {

public:

vector<int> msort(vector<int> nums1, vector<int> nums2)

{

vector<int> nums;

int i=0, j=0, m, n;

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

{

if(nums1[i]<=nums2[j]) nums.push\_back(nums1[i++]);

else nums.push\_back(nums2[j++]);

}

for(m=i; m<nums1.size(); m++) nums.push\_back(nums1[m]);

for(m=j; m<nums2.size(); m++) nums.push\_back(nums2[m]);

return nums;

}

double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2)

{

vector<int> nums = msort(nums1, nums2);

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

if (nums.size() % 2 == 1) return double(nums[nums.size()/2]);

return (double(nums[nums.size()/2 -1]) + double(nums[nums.size()/2]))/2;

}

};

1. ***Merge k Sorted Lists:***

class Solution {

public:

ListNode\* mergeKLists(vector<ListNode\*>& lists) {

if(lists.size() == 0) return NULL;

vector<int> v;

int i;

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

{

ListNode\* temp = lists[i];

while(temp!=NULL)

{

v.push\_back(temp->val);

temp = temp->next;

}

}

if(v.size() == 0) return NULL;

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

ListNode\* start = new ListNode(v[0]);

ListNode\* temp1 = start;

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

{

temp1->next = new ListNode(v[i]);

temp1 = temp1->next;

}

return start;

}

};

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. ***Majority Element:***

class Solution {

public:

int majorityElement(vector<int>& nums) {

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

int i, start, count;;

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

{

if(nums[i] == nums[i+(nums.size()/2)]) return nums[i];

}

return -1;

}

};

1. ***Kth Largest Element in an Array:***

class Solution {

public:

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

int max,i;

if(nums.size() == 1) return nums[0];

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

if(k == 1) return nums[nums.size()-1];

if(k == nums.size()) return nums[0];

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

{

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

max = nums[i];

}

return max;

}

};

1. ***Longest Substring with At Least K Repeating Characters:***

class Solution {

public:

int longsubstr(string s, int start, int end, int k)

{

if (end < k) return 0;

int countMap[26] = {0};

for (int i = start; i < end; i++)

countMap[s[i] - 'a']++;

for (int mid = start; mid < end; mid++) {

if (countMap[s[mid] - 'a'] >= k) continue;

int midNext = mid + 1;

while (midNext < end && countMap[s[midNext] - 'a'] < k) midNext++;

return max(longsubstr(s, start, mid, k),

longsubstr(s, midNext, end, k));

}

return end-start;

}

int longestSubstring(string s, int k) {

return longsubstr(s, 0, s.size(), k);

}

};