***Week – 6 (24.05.2021 – 29.05.2021)***

***LEETCODE CONTEST – 29.05.2021***

Rank - 5198/ 12291

Mark – 7/18

Link - <https://leetcode.com/contest/biweekly-contest-53>

1. ***Substrings of Size Three with Distinct Characters:***

class Solution {

public:

int countGoodSubstrings(string s) {

if(s.size() < 3) return 0;

int i, count = 0;

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

{

vector<bool> v(26, false);

if(!v[s[i] - 'a']) v[s[i] - 'a'] = true;

else continue;

if(!v[s[i+1] - 'a']) v[s[i+1] - 'a'] = true;

else continue;

if(!v[s[i+2] - 'a']) v[s[i+2] - 'a'] = true;

else continue;

count++;

}

return count;

}

};

1. ***Minimize Maximum Pair Sum in Array:***

class Solution {

public:

int minPairSum(vector<int>& nums) {

if(nums.size()%2 != 0 || nums.size() == 0) return 0;

int i, max\_sum = INT\_MIN;

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

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

max\_sum = max(max\_sum, nums[i]+nums[nums.size()-1-i]);

return max\_sum;

}

};

1. ***Get Biggest Three Rhombus Sums in a Grid:***

class Solution {

public:

vector<int> getBiggestThree(vector<vector<int>>& grid) {

int row = grid.size(), col = grid[0].size();

priority\_queue<int> pq;

for (auto &r: grid) {

for(auto &c: r) pq.push(c);

}

for (int k = 1; k < min(row, col); ++k)

{

for (int i = 0; i + 2 \* k < row; ++i)

{

for (int j = 0; j + 2 \* k < col; ++j)

{

int sum = 0;

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

{

sum += grid[i + k - m][j + m];

sum += grid[i + m][j + k + m];

sum += grid[i + k + m][j + 2 \* k - m];

sum += grid[i + 2 \* k - m][j + k - m];

}

pq.push(sum);

}

}

}

vector<int> ret;

while (pq.empty() == false)

{

if (ret.empty() == true || ret.back() != pq.top())

ret.push\_back(pq.top());

if (ret.size() == 3) return ret;

pq.pop();

}

return ret;

}

};

1. ***Minimum XOR Sum of Two Arrays:***

class Solution {

public:

int dp[14][1 << 14];

int dfs(vector<int>& nums1, vector<int>& nums2, int i, int mask) {

if(i >= nums1.size()) return 0;

int res = 1e9;

if(dp[i][mask] < 0x3f3f3f3f) return dp[i][mask];

for(int j = 0; j < nums2.size(); j++) {

if((1 << j) & mask) continue;

int sum = (nums1[i] ^ nums2[j]) + dfs(nums1, nums2, i + 1, mask | (1 << j));

res = min(res, sum);

}

return dp[i][mask] = res;

}

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

memset(dp, 0x3f, sizeof(dp));

return dfs(nums1, nums2, 0, 0);

}

};