***Week – 7 (31.05.2021 – 05.06.2021)***

***RANDOM CODES***

1. ***Maximum Gap:***

class Solution {

public:

int maximumGap(vector<int>& nums) {

if(nums.size()<2) return 0;

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

int i, max\_diff=0;

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

max\_diff = max(max\_diff,nums[i]-nums[i-1]);

return max\_diff;

}

};

1. ***Top K Frequent Elements:***

class Solution {

public:

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

map<int,int> m;

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

m[nums[i]]++;

priority\_queue<pair<int, int>> pq;

int i=0;

vector<int> res;

for(auto e : m)

pq.push({e.second,e.first});

while(!pq.empty() && i<k)

{

res.push\_back(pq.top().second);

pq.pop();

i++;

}

return res;

}

};

1. ***Sort Characters By Frequency:***

class Solution {

public:

string frequencySort(string s) {

map<char,int> m;

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

m[s[i]]++;

priority\_queue<pair<int, char>> pq;

string res="";

int i;

for(auto e : m)

pq.push({e.second,e.first});

while(pq.size())

{

pair<int,char> p= pq.top();

pq.pop();

for(int i=0;i<p.first;i++)

res += p.second;

}

return res;

}

};

1. ***Sort Array by Increasing Frequency:***

class Solution {

public:

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

map<char,int> m;

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

{

m[nums[i]]++;

}

vector<int> ans;

priority\_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>> pq;

for(auto val : m)

{

pq.push({val.second,-val.first});

}

while(pq.size())

{

pair<int,char> p= pq.top();

pq.pop();

for(int i=0;i<p.first;i++)

{

ans.push\_back(-p.second);

}

}

return ans;

}

};

1. ***Top K Frequent Words:***

class Solution {

public:

vector<string> topKFrequent(vector<string>& words, int k) {

map<string,int> m;

vector<string> res;

int i=1;

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

{

m[words[i]]++;

}

priority\_queue<pair<int,string>, vector<pair<int,string>>, greater<pair<int,string>>> pq;

for(auto val : m)

{

pq.push({-val.second,val.first});

}

while(i<=k && pq.size())

{

pair<int,string> p= pq.top();

pq.pop();

res.push\_back(p.second);

i++;

}

return res;

}

};

1. ***K-th Smallest Prime Fraction:***

class Solution {

public:

vector<int> kthSmallestPrimeFraction(vector<int>& arr, int k) {

priority\_queue<pair<double, pair<int,int>>, vector<pair<double, pair<int,int>>>, greater<pair<double, pair<int,int>>>> v;

int i, j, t = 1;

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

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

v.push({(double)arr[i]/arr[j],{arr[i],arr[j]}});

while(t < k)

{

v.pop();

t++;

}

return vector<int>{v.top().second.first,v.top().second.second};

}

};

1. ***Wiggle Sort II:***

class Solution {

public:

void wiggleSort(vector<int>& nums) {

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

vector<int>ans(nums.size());

int i, j=nums.size()-1;

for(i=1; i<nums.size(); i=i+2)

{

ans[i]=nums[j];

j--;

}

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

{

ans[i]=nums[j];

j--;

}

nums=ans;

}

};

1. ***Filter Restaurants by Vegan-Friendly, Price and Distance:***

class Solution {

public:

vector<int> filterRestaurants(vector<vector<int>>& restaurants, int veganFriendly, int maxPrice, int maxDistance) {

priority\_queue<pair<int,int>> pq;

vector<int> v;

int i;

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

{

if(veganFriendly == 1)

{

if(restaurants[i][2] == 1 && restaurants[i][3] <= maxPrice && restaurants[i][4] <= maxDistance) pq.push(make\_pair(restaurants[i][1],restaurants[i][0]));

}

else

{

if(restaurants[i][3] <= maxPrice && restaurants[i][4] <= maxDistance) pq.push(make\_pair(restaurants[i][1],restaurants[i][0]));

}

}

while(!pq.empty())

{

v.push\_back(pq.top().second);

pq.pop();

}

return v;

}

};

1. ***Rank Teams by Votes:***

class Solution {

public:

string rankTeams(vector<string>& s)

{

int m=s.size();

int n=s[0].size();

vector<vector<int>>rank(26,vector<int>(n,0));

for(int j=0;j<n;j++)

{

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

{

rank[s[i][j]-'A'][j]++;

}

}

vector<int>teams;

for(int i=0;i<26;i++)

teams.push\_back(i);

sort(teams.begin(),teams.end(),[&](int &a,int &b){

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

{

if(rank[a][i]!=rank[b][i])

return rank[a][i]>rank[b][i];

}

return a<b;

});

string res="";

for(auto x:teams)

{

res+=(char)(x+'A');

if(res.size()==n)

break;

}

return res;

}

};