**Unique Questions According to Me**

1. Smallest Positive Integer that can not be represented as Sum

ANS. 1st approach (Find all the possible subsets and compare with the numbers from 1):

2nd approach ():

long long smallestpositive(vector<long long> array, int n)

{

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

long long macx=1;

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

{

if (array[i]<=macx)

macx+=array[i];

else

break;

}

return macx;

}

2. Print matrix in Anti-Spiral form

ANS. 1st approach (find the spiral order and then reverse the vector):

#include <bits/stdc++.h>

using namespace std;

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

int m=matrix.size();

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

int t=0,b=m-1,l=0,r=n-1;

vector<int> res;

while (l<=r && t<=b)

{

for (int i=l;i<=r;i++)

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

t++;

if (t>b)

break;

for (int i=t;i<=b;i++)

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

r--;

if (r<l)

break;

for (int i=r;i>=l;i--)

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

b--;

if (b<t)

break;

for (int i=b;i>=t;i--)

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

l++;

}

return res;

}

int main()

{

vector<vector<int>> mat

{

{1, 2, 3, 4, 5},

{6, 7, 8, 9, 10},

{11, 12, 13, 14, 15},

{16, 17, 18, 19, 20}

};

vector<int> res= spiralOrder(mat);

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

for (int i:res)

cout<<i<<" ";

return 0;

}

3. My Calender-I

ANS. 1st approach (Brute force):

vector<pair<int,int>> v;

MyCalendar() {

}

bool book(int start, int end) {

if (v.size()==0)

v.push\_back({start,end});

else

{

for (auto it:v)

{

if (start<it.second && end>it.first)

return false;

}

v.push\_back({start,end});

}

return true;

}

2nd approach ():

multiset<pair<int,int>> m;

MyCalendar() {

}

bool book(int start, int end) {

m.insert({start,1});

m.insert({end,-1});

int s=0;

for (auto it:m)

{

s+=it.second;

if (s>1)

{

m.erase(m.find({start,1}));

m.erase(m.find({end,-1}));

return false;

}

}

return true;

}

4. Product of Array Except Self

ANS. 1st approach (multiply all numbers and divide the multiplication by each number):

2nd approach (calculate left and right products in two vectors TC-O(N),SC-O(N)):

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

int n=nums.size();

vector<int> left(n,1);

vector<int> right(n,1);

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

left[i]=left[i-1]\*nums[i-1];

for (int i=n-2;i>=0;i--)

right[i]=right[i+1]\*nums[i+1];

vector<int> ans(n,0);

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

ans[i]=left[i]\*right[i];

return ans;

}

3rd approach (Use ans vector for storing the left multiplication TC-O(N),SC-O(1)):

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

int n=nums.size();

vector<int> ans(n,1);

int prod=nums[n-1];

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

ans[i]=ans[i-1]\*nums[i-1];

for (int i=n-2;i>=0;i--)

{

ans[i]=ans[i]\*prod;

prod\*=nums[i];

}

return ans;

}

5. Sort with Difficulty

ANS. 1st approach ():

#include <bits/stdc++.h>

using namespace std;

int main() {

int p, s; cin>>p>>s;

vector<pair<int, int>> ans;

for(int i =0; i<p; i++){

vector<pair<int, int>> problem(s);

for(int i = 0;i<s;i++){

cin>>problem[i].first;

}

for(int i = 0;i<s;i++){

cin>>problem[i].second;

}

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

int c = 0;

for(int i = 0; i<s-1; i++){

if(problem[i].second>problem[i+1].second){

c++;

}

}

ans.push\_back({c,i+1});

}

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

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

cout<<ans[i].second<<endl;

return 0;

}

6. Greedy Candidates

Ans. 1st approach ():

#include <bits/stdc++.h>

using namespace std;

int main() {

int t;

cin>>t;

while (t--)

{

int N,M;

cin>>N>>M;

vector<int> minSalary(N,0);

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

cin>>minSalary[i];

vector<int> offeredSalary(M,0);

vector<int> maxJobOffers(M,0);

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

{

cin>>offeredSalary[i];

cin>>maxJobOffers[i];

}

vector<vector<int>> arr(N,vector<int> (M,0));

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

{

string s;

cin>>s;

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

arr[i][j] = s[j] - '0';

}

int jobs=0;

long long salary=0;

set<int> s;

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

{

int selected=-1;

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

{

if (arr[i][j] && maxJobOffers[j]>0 && offeredSalary[j]>minSalary[i])

{

if (selected==-1 || offeredSalary[selected]<offeredSalary[j])

{

selected=j;

}

}

}

if (selected>=0)

{

jobs++;

salary+=(long long)offeredSalary[selected];

maxJobOffers[selected]--;

s.insert(selected);

}

}

cout<<jobs<<" "<<salary<<" "<<M-s.size()<<endl;

}

return 0;

}

7. Redcue To Zero

ANS. 1st approach ():

#include <bits/stdc++.h>

using namespace std;

int main() {

int t;

cin>>t;

while (t--)

{

long long x,y;

cin>>x>>y;

if (x>y) swap(x,y);

if (x==y) cout<<x<<endl;

else if (x==0) cout<<-1<<endl;

else

{

long long ans=0;

while (x<y)

{

x\*=2;

ans++;

}

cout<<ans+y<<endl;

}

}

return 0;

}

8. Count Pairs in array whose sum is divisible by k

ANS. 1st approach (O(n2)):

2nd approach (O(n)):

int countKdivPairs(int A[], int n, int K)

{

int freq[k]={0};

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

++freq[A[i] % K];

int sum = freq[0] \* (freq[0] - 1) / 2;

for (int i = 1; i <= K / 2 && i != (K - i); i++)

sum += freq[i] \* freq[K – i];

if (K % 2 == 0)

sum += (freq[K / 2] \* (freq[K / 2] - 1) / 2);

return sum;

}

9. Swapping pairs make sum equal

Ans. 1st Approach (Use Hashing to store the second array):

int findSwapValues(int A[], int n, int B[], int m)

{

unordered\_map<int,int> mp;

int suma=accumulate(A,A+n,0);

int sumb=accumulate(B,B+m,0);

if ((suma-sumb)%2==1) return -1;

int target=(suma-sumb)/2;

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

mp[B[i]]=i;

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

{

if (mp.find(A[i]-target)!=mp.end())

return 1;

}

return -1;

}

2nd Approach (Sort the array and check for Target):

int findSwapValues(int A[], int n, int B[], int m)

{

long long suma=accumulate(A,A+n,0);

long long sumb=accumulate(B,B+m,0);

if (suma==sumb) return 1;

if ((suma-sumb)%2==1) return -1;

long long target=(suma-sumb)/2;

sort(A,A+n);

sort(B,B+m);

int i=0,j=0;

while (i<n && j<m)

{

long long currdiff=A[i]-B[j];

if (currdiff==target)

return 1;

else if (currdiff<target)

i++;

else

j++;

}

return -1;

}

10. Phone Number(Leetcode 17)

ANS. 1st approach (BackTracking Solution):

void solve(int i,string digits,vector<string>&ans,string temp,map<char,string> &mp)

{

if (i==digits.size())

{

ans.push\_back(temp);

return;

}

string comb=mp[digits[i]];

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

solve(i+1,digits,ans,temp+comb[j],mp);

}

vector<string> letterCombinations(string digits) {

map<char,string> mp;

mp['2']="abc";

mp['3']="def";

mp['4']="ghi";

mp['5']="jkl";

mp['6']="mno";

mp['7']="pqrs";

mp['8']="tuv";

mp['9']="wxyz";

vector<string> ans;

if (digits=="") return ans;

solve(0,digits,ans,"",mp);

return ans;

}

11. Find Maximum of Two Numbers without using any conditional Operators

Ans. 1st approach ():

12. Find Number of Trailing Zeroes in Factorial of Number

Ans. 1st approach (we have to count the number of 5 and 2 because these two will comtribute towards making zeroes at the end of Number i.e 10 . we will get 2’s easily by any even number. So we will count only 5’s):

int trailingZeroes(int n) {

int count=0;

for (long long i=5;n/i>0;i\*=5)

count+=n/i;

return count;

}

13. Find two Numbers each from one array such that their abs Diff is Minimum

Ans. 1st approach ():

void findPair(int A[], int B[])

{

int m = A.length - 1;

int n = B.length - 1;

int i = 0;

int j = 0;

int min = Integer.MAX\_VALUE;

int a, b;

while (i < m && j < n) {

if (Math.abs(A[i]-B[j]) < min) {

min = Math.abs(A[i]-B[j]);

a = A[i];

b = B[j];

if (min == 0)

break; /\* absolute difference cannot be less than 0 \*/

}

if (A[i] < B[j])

i++;

else

j++;

}

System.out.println("The pair with minimum absolute difference is "+a+","+b);

}