**Data Structures(week-2)**

1. 1-D Array

1. Non-decreasing Arrays

#include <iostream>

using namespace std;

int main() {

long long testcase;

cin >> testcase;

while(testcase--){

long long n;

cin >> n;

long long a[n],i,j,k,b[n],rem,diff;

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

cin >> a[i];

}

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

if(i==0){

b[i] = a[i];

}

else if(a[i]>b[i-1]){

b[i] = a[i];

}else{

rem = b[i-1]%a[i];

diff = rem == 0 ? 0 : a[i] - rem;

b[i] = b[i-1]+diff;

}

}

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

cout << b[i] << " ";

}

cout << "\n";

}

}

2.Minimum Additions

#include<bits/stdc++.h>

using namespace std;

#define ll long long

int main(){

int testcase;

cin >> testcase;

while(testcase--){

int n,k;

cin >> n >> k;

ll a[n],avg,result,sum=0;

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

cin >> a[i];

sum+=a[i];

}

avg=sum/n;

if(avg<=k){

cout << 0 << '\n';

}else{

result = sum/(k+1) - n +1;

cout << result << '\n';

}

}

return 0;

}

3. Scoreboard queries

#include<bits/stdc++.h>

using namespace std;

#define ll long long

int main(){

int testcase;

cin >> testcase;

while(testcase--){

int n,q;

cin >> n >> q;

int a[n];

unordered\_map<int,int>m;

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

cin >> a[i];

m[a[i]]++;

}

while(q--){

int l,r;

cin >> l >> r;

int ele=a[l-1];

a[l-1]=r;

if(m.count(ele)>0){

m[ele]--;

if(m[ele]==0)

m.erase(ele);

}

m[r]++;

cout << m.size() +1 << '\n';

}

}

return 0;

}

4. Bracket Sequences

#include<bits/stdc++.h>

using namespace std;

#define ll long long

int main(){

string s;

cin >> s;

int n=s.length();

int ans=0;

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

int c=0;

if(s[i]=='('){ // run only when first bracket = (

for(int j=i;j<i+n;j++){

if(s[j%n]=='(')

c++;

else

c--; // for ) bracket

if(c<0) // when more ) bracket than (

break;

}

if(c==0) // if correct seq then only increasing ans

ans++;

}

}

cout << ans;

}

Multi-dimensional Array

1. Grid and phrase

#include <iostream>

using namespace std;

int main() {

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

cin>>n>>m;

char arr[n][m];

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

for(j=0;j<m;j++){

cin >> arr[i][j];

}

}

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

{

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

{

if(arr[i][j]=='s')

{

//checks horizontally

if(arr[i][j+1]=='a'&&arr[i][j+2]=='b'&&arr[i][j+3] =='a'&&j+3<m)

count++;

//checks vertically

if(arr[i+1][j]=='a'&&arr[i+2][j]=='b'&&arr [i+3][j]=='a'&&i+3<n)

count++;

//checks top-bottom diagonal

if(arr[i+1][j+1]=='a'&&arr[i+2][j+2]=='b'&& arr[i+3][j+3]=='a'&&i+3<n&&j+3<m)

count++;

//checks bottom-up diagonal

if(arr[i-1][j+1]=='a'&&arr[i-2][j+2]=='b'&&

arr[i-3][j+3]=='a'&&i-3>=0&&j+3<m)

count++;

}

}

}

cout<<count;

}

1. Left or right

#include<bits/stdc++.h>

using namespace std;

#define ll long long

int main()

{

ll n,q;

cin>>n>>q;

ll arr[n]={0};

unordered\_map<ll,vector<ll>>m;

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

{

cin>>arr[i];

m[arr[i]].push\_back(i);

}

while(q--)

{

ll start,destNo,answer=0;

char d;

cin>>start>>destNo>>d;

if(m.find(destNo) == m.end())

{

cout<<-1<<endl;

}

else if(arr[start]==destNo)

{

cout<<0<<endl;

}

else if(d=='L')

{

ll miniL=INT\_MAX;

ll miniR=INT\_MAX;

for(auto iter:m[destNo])

{

ll index=iter;

if(index<start)

{

miniL=min(start-index,miniL);

}

else if(index>start)

{

miniR=min(start+n-index,miniR);

}

}

cout<<min(miniL,miniR)<<endl;

}

else if(d=='R')

{

ll miniL=INT\_MAX;

ll miniR=INT\_MAX;

for(auto iter:m[destNo])

{

ll index=iter;

if(index>start)

{

miniL=min(index-start,miniL);

}

else if(index<start)

{

miniR=min(index+n-start,miniR);

}

}

cout<<min(miniL,miniR)<<endl;

}

}

}

1. Find the String

#include <iostream>

#include <vector>

#include <unordered\_map>

#include <bits/stdc++.h>

#define ll long long

int main()

{

using namespace std;

int tests;

cin >> tests;

while(tests--)

{

ll row,col;

cin >> row >> col;

vector<vector<char>> arr(row , vector<char>(col));

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

{

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

{

cin >> arr[i][j];

}

}

string str ;

cin >> str;

int i = 0,index = 0;

// Iterate till the entire string is iterated

while(index < str.size())

{

// set a flag as false so as to indicate whether a occurence has been found

bool flag = false;

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

{

// if we have a occurence then remove it increase the row index set flag true

if(str[index] == arr[i][j])

{

//cout << i << " " << j << endl;

arr[i][j] = '.';

index++;

i = (i + 1) % row;

flag = true;

}

}

// if flag is still false means it didnt get a occurence

if(!flag)

{

cout << "No";

return 0;

}

}

cout << "Yes" << endl;

}

}

1. Roy and Symmetric logs

#include <iostream>

using namespace std;

int main() {

int testcase;

cin >> testcase;

while(testcase--){

int n;

cin >> n;

char arr[n][n];

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

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

cin >> arr[i][j];

}

}

bool rowf = true,colf = true;

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

bool flag = true;

for(int j=0;j<n/2;j++){

if(arr[i][j] != arr[i][n-j-1]){

flag = false;

break;

}

}

if(!flag){

rowf = false;

break;

}

}

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

bool flag = true;

for(int j=0;j<n/2;j++){

if(arr[j][i] != arr[n-j-1][i]){

flag = false;

break;

}

}

if(!flag){

colf = false;

break;

}

}

if(rowf && colf){

cout <<"YES" <<endl;

}else{

cout << "NO" <<endl;

}

}

}