Cho mảng A[] gồm n số nguyên không âm và số k. Hãy tìm số lớn nhất của mỗi dãy con liên tục gồm k phần tử của mảng. Ví dụ với mảng A[] = {1, 2, 3, 1, 4, 5, 2, 3, 6}, K = 3, ta có kết quả *3 3 4 5 5 5 6*.

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào n là số phần tử của mảng A[] và số k; dòng kế tiếp đưa vào n số A[i] của mảng; các số được viết cách nhau một vài khoảng trống.
* T, n, A[i] thỏa mãn ràng buộc: 1≤ T ≤100; 1≤ k < n ≤107; 0≤ A[i] ≤107;

Output:

* Đưa ra kết quả mỗi test theo từng dòng.

#include<iostream>

#include<string>

#include<vector>

#include<map>

#include<queue>

#include<cmath>

#include<set>

#include<algorithm>

#define ll long long

using namespace std;

const int oo = 1e7+5;

int A[oo];

int main(){

cin.tie(0); ios\_base::sync\_with\_stdio(false);

int test; cin >> test;

while(test--){

int n, k; cin >> n >> k;

for(int i=0; i<n; i++) cin >> A[i];

int res = 0;

for(int i=0; i<k; i++) res = max(res, A[i]);

cout << res << " ";

for(int i=1; i<=n-k; i++){

if(A[i-1] == res){

res = 0;

for(int j=i; j<=i+k; j++) res = max(res, A[j]);

}

else if(A[k+i-1] > res) res = A[k+i-1];

cout << res << " ";

}

cout << endl;

}

}

Cho mảng X[] gồm n phần tử và mảng Y[] gồm m phần tử. Hãy đếm số các cặp xy>yx, trong đó x€X[] và y€Y[]. Ví dụ X[] = {2, 1, 6 }, Y[] = {1, 5} ta có kết quả là 3 cặp (2, 1), (2, 5), (6, 1).

Input:

* Dòng đầu tiên đưa vào số lượng bộ test T.
* Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm ba dòng: dòng đầu tiên đưa vào n, m tương ứng với số phần tử của mảng X[] và Y[]; dòng tiếp theo là n số X[i] của mảng X[]; dòng cuối cùng là m số của mảng Y[]; các số được viết cách nhau một vài khoảng trống.
* T, n, m, X[i], Y[j] thỏa mãn ràng buộc: 1≤ T ≤100; 1≤ n, m ≤105; 1≤ X[i], Y[j] ≤103.

Output:

* Đưa ra kết quả mỗi test theo từng dòng.

#include<bits/stdc++.h>

using namespace std;

int tam[5];

int n,m,a[100000],b[100000];

int dem(int x){

if(x==0){

return 0;

}

if(x==1){

return tam[0];

}

int \*pos=upper\_bound(b,b+m,x);

int ans=(b+m)-pos;

ans+=tam[0]+tam[1];

if(x==2){

ans-=tam[3]+tam[4];

}

if(x==3){

ans+=tam[2];

}

return ans;

}

int main(){

int t;

cin>>t;

while(t--){

memset(tam,0,sizeof(tam));

cin>>n>>m;

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

cin>>a[i];

}

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

cin>>b[i];

if(b[i]<=4){

tam[b[i]]++;

}

}

sort(b,b+m);

int ans=0;

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

ans+=dem(a[i]);

}

cout<<ans<<endl;

}

}

Cho hai số rất lớn X và Y được biểu diễn như hai xâu ký tự. Nhiệm vụ của bạn là tìm |X-Y|?

#include<bits/stdc++.h>

#include<string>

using namespace std;

string subtraction(string a,string b){

string res="";

while(b.size()<a.size()) b.insert(0,"0");

if(a<b) swap(a,b);

// cout<<a<<endl<<b<<endl;

int borrow=0;

for(int i=a.size()-1;i>=0;i--){

int temp=a[i]-b[i]-borrow;

if(temp<0){

temp+=10;

borrow=1;

}

else borrow=0;

res=(char)(temp%10+48)+res;

}

while(res.length()>1&&res[0]=='0'&&res[1]=='0') res.erase(0,1);

return res;

}

int main(){

int n;

cin>>n;

while(n--){

string a,b;

cin>>a>>b;

if(a.size()<b.size()) swap(a,b);

cout<<subtraction(a,b)<<endl;

}

}

Cho hai số rất lớn X và Y được biểu diễn như hai xâu ký tự. Nhiệm vụ của bạn là tìm X+Y?

#include<bits/stdc++.h>

#include<algorithm>

using namespace std;

string solve(string s1,string s2){

if(s1.length()<s2.length()){

string s=s1;

s1=s2;

s2=s2;

s2=s;

}

s2.insert(0,s1.length()-s2.length(), '0');

int res=0;

string an;

for(int i=s1.length()-1;i>=0;i--){

int tam=s1.at(i)-'0'+s2.at(i)-'0'+res;

an.insert(0,1,tam%10+'0');

res=tam/10;

}

if(res!=0){

an.insert(0,1,res+'0');

}

return an;

}

string sub(string s1,string s2){

if(s1.length()<s2.length()){

string s=s1;

s2=s2;

s2=s;

}else if(s1.length()==s2.length()){

for(int i=0;i<s1.length();i++){

if(s1.at(i)<s2.at(i)){

string s=s1;

s1=s2;

s2=s;

break;

}

}

}

s2.insert(0,s1.length()-s2.length(),'0');

int res=0;

string an;

for(int i=s1.length()-1;i>=0;i--){

if(s1.at(i)-'0'<s2.at(i)-'0'+res){

an.push\_back((s1.at(i)-'0'+10)-(s2.at(i)-'0'+res)+'0');

res=1;

}else{

an.push\_back((s1.at(i)-'0')-(s2.at(i)-'0'+res)+'0');

res=0;

}

}

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

return an;

}

string contro(string s1, string s2){

if(s1.at(0) == '-' && s2.at(0) != '-'){

return sub(s1.substr(1), s2);

}

else if(s1.at(0) != '-' && s2.at(0) == '-'){

return sub(s1, s2.substr(1));

}

else if(s1.at(0) == '-' && s2.at(0) == '-'){

string tl = solve(s1.substr(1), s2.substr(1));

return tl.insert(0, 1, '-');

}

else return solve(s1, s2);

}

int main(){

int t;

cin >> t;

while(t--){

string s1, s2, tam;

cin >> s1 >> s2;

if(s1.length() == 0) cout << s2 << endl;

else if(s2.length() == 0) cout << s1 << endl;

else cout << solve(s1, s2) << endl << endl;

}

}

Tên người dùng sẽ được chuẩn hóa theo nguyên tắc tên được viết sau cùng, phân tách với phần tên đệm và tên bởi dấu phẩy. Các chữ cái nằm trong tên đều được viết hoa; chữ cái đầu tiên của tên đệm và họ được viết hoa, các chữ cái còn lại viết thường.

#include<iostream>

#include<cstring>

using namespace std;

int main(){

string s;getline(cin,s);

while(s[0]==' ') s.erase(s.begin());

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

if(s[i]==' '&&s[i+1]==' '){

s.erase(s.begin()+i+1);

i--;

}

}

int mark=0;

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

if(s[i]==' '&&((s[i+1]>=65&&s[i+1]<=90)||(s[i+1]>=97&&s[i+1]<=122))) mark=1;

}

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

if(s[i]>=65&&s[i]<=90) s[i]+=32;

}

int j=0;int kt=1;

for(int i=s.size()-1;i>=0;i--){

if(s[i]!=' '&&s[i-1]==' '){

j=i;

s.insert(s.begin()+j-1,',');

kt=0;

break;

}

}

if(j==0) s.insert(s.begin(),',');

if(s[0]!=',') s[0]-=32;

for(int i=j+1;i<s.size();i++) s[i]-=32;

for(int i=1;i<j;i++){

if(s[i]==' ') s[i+1]-=32;

}

cout<<s;

}

Dãy số Fibonacci được định nghĩa Fn = Fn-1 + Fn-2, n>1 và F0 = 0, F1 = 1. Dưới đây là một số số Fibonacci : 0, 1, 1, 2, 3, 5, 8, 13, 21…

Nhiệm vụ của bạn là tìm số Fibonacci thứ n.

#include<bits/stdc++.h>

#define modulo 1000000007

using namespace std;

long long so(long long n){

if(n<2){

return 1;

}

long long a=0,b=1,c;

while(n-->1){

c=(a%modulo+b%modulo)%modulo;

a=b;

b=c;

}

return c;

}

int main(){

int t;cin>>t;

while(t--){

int n;

cin>>n;

cout<<so(n)<<endl;

}

}

Cho mảng các số nhị phân A1[] và A2[] gồm n 0, 1. Hãy tìm khoảng chung dài nhất thỏa mãn: j ≥i và span(i, j) = A1[i] + A1[i+1] + …+A1[j] = A2[i] + A2[i+1] + …+A2[j]. Ví dụ với A1[] = {0, 1, 0, 0, 0, 0}, A2[] = {1, 0, 1, 0, 0, 1} ta có kết quả là 4 tương ứng với A1[1]+ A1[2]+ A1[3]+ A1[4] = A2[1]+ A2[2]+ A2[3]+ A2[4] = 1.

#include<bits/stdc++.h>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

int a[n],b[n];

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

cin>>a[i];

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

cin>>b[i];

int max=1,k=0;

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

int x=0,y=0;

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

x+= a[j];

y+= b[j];

if(x==y)

k=j-i+1; }

if(k>max)

max=k; }

cout<<max;

cout<<endl; }

}

Cho mảng A[] gồm n số nguyên dương. Nhiệm vụ của bạn là hãy sắp xếp lại các phần tử của mảng sao cho A[i] ≥A[i-1] nếu i chẵn,  A[i] ≤A[i-1] nếu i lẻ. Ví dụ với mảng A[] = {1, 2, 2, 1} ta được mảng được sắp A[] = { 1, 2, 1, 2}.

#include<iostream>

#include<algorithm>

using namespace std;

int a[1001];

int main(){

int t,n; cin >> t;

while(t--){

cin >> n;

for (int i=1;i<=n;i++) cin >> a[i];

sort(a,a+n+1);

int x=1,y=(n%2==0)?(n/2+1):(n/2+2);

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

if (i%2==0) cout << a[y++] << " ";

else cout << a[x++] << " ";

}

cout << endl;

}

}

Cho số tự nhiên N. Bạn chỉ được phép sử dụng nhiều nhất một phép đổi chỗ giữa 2 chữ số để nhận được số lớn nhất nhỏ hơn N. Ví dụ với số N=12435, sử dụng một phép đổi chỗ ta nhận được số lớn nhất nhỏ hơn N là 12345. Mặc dù 12354 > 12345 nhưng ta không thể tạo ra số 12345 với chỉ một phép hoán vị. Với số N=12345 ta không có phép đổi chỗ.

#include<iostream>

#include<string>

#include<vector>

#include<queue>

#include<map>

#include<algorithm>

#define ll long long

using namespace std;

const int oo = 1e6+5;

int A[oo];

int main(){

cin.tie(0); ios\_base::sync\_with\_stdio(false);

int test; cin >> test;

while(test--){

string s; cin >> s;

priority\_queue<char> q;

int j = s.length()-1;

while(s.at(j) >= s.at(j-1)){

q.push(s.at(j--));

if(j == 0) break;

}

if(j == 0){

cout << -1 << endl;

continue;

}

q.push(s.at(j));

while(q.top() >= s.at(j-1)) q.pop();

char res = q.top();

int local = s.find(res, j);

char c = s[local]; s[local] = s[j-1]; s[j-1] = c;

cout << s << endl;

}

}

Cho mảng A[] gồm n số nguyên dương. Hãy tìm tổng lớn nhất của dãy con thỏa mãn ràng buộc không có hai phần tử kề nhau thuộc một dãy con. Ví dụ với mảng A[] = {3, 2, 7, 10} ta nhận được kết quả là 13 = 10 + 3. Với mảng A[] =

#include<iostream>

#include<string>

#include<vector>

#include<queue>

#include<map>

#include<algorithm>

#define ll long long

using namespace std;

const int oo = 1e5+5;

ll A[oo], F[oo] = {0};

//F[i] tong max khi xet den so thu i

int main(){

cin.tie(0); ios\_base::sync\_with\_stdio(false);

int test; cin >> test;

while(test--){

int n; cin >> n;

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

cin >> A[i]; F[i] = 0;

}

F[0] = A[0]; F[1] = (A[0] > A[1])? A[0] : A[1];

ll ans = (F[0] > F[1])? F[0] : F[1];

for(int i=2; i<n; i++) F[i] = max(F[i-1], F[i-2] + A[i]);

cout << F[n-1] << endl;

}

}

Cho mảng A[] gồm n số được sinh ra ngẫu nhiên. Hãy tìm độ dài dãy tăng dài nhất các phần tử của mảng. Chú ý, dãy con của mảng không nhất thiết là liên tục. Hai phần tử giống nhau của mảng ta chỉ xem là 1 trong độ dài dãy tăng. Ví dụ với mảng A[] = {5, 8, 3, 7, 9, 1}, ta có kết quả là 3.

#include<iostream>

#include<string>

#include<vector>

#include<queue>

#include<map>

#include<algorithm>

#define ll long long

using namespace std;

const int oo = 1e3+5;

int A[oo], F[oo] = {0};

int main(){

cin.tie(0); ios\_base::sync\_with\_stdio(false);

int test; cin >> test;

while(test--){

int n; cin >> n;

for(int i=0; i<n; i++) cin >> A[i];

F[0] = 1;

int ans = 1;

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

int res = 0;

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

if(A[i] > A[j]){

if(F[j]>res){

res=F[j];

}else{

res=res;

}

}

}

F[i] = res + 1;

if(ans>F[i]){

ans=ans;

}else{

ans=F[i];

}

}

cout << ans << endl;

}

}

Cho mảng A[] gồm n phần tử. Nhiệm vụ của bạn là tìm tổng lớn nhất các dãy con liên tục của mảng A[]. Ví dụ với A[] = { -2, -3, 4, -1, -2, 1, 5, -3} ta có câu trả lời là 7 tương ứng với tổng lớn nhất của dãy con liên tục {4, -1, -2, 1, 5}.

#include<iostream>

#include<string>

#include<vector>

#include<queue>

#include<map>

#include<algorithm>

#define ll long long

using namespace std;

const int oo = 1e6+5;

int A[oo];

ll F[oo];

int main(){

cin.tie(0); ios\_base::sync\_with\_stdio(false);

int test; cin >> test;

while(test--){

int n; cin >> n;

for(int i=0; i<n; i++) cin >> A[i];

F[0] = A[0];

ll ans = A[0];

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

F[i] = (F[i-1] < 0)? A[i] : A[i] + F[i-1];

ans = (ans > F[i])? ans : F[i];

}

cout << ans << endl;

}

}

Cho mảng A[] gồm N phần tử đã được sắp xếp. Hãy tìm K phần tử gần nhất của X. Ví dụ với mảng A[] = {1, 3, 5, 7, 9, 11}, X = 7, K=2 thì ta có 2 phần tử gần nhất của 7 là 5 và 9. Chú ý: X có thể có mặt hoặc không có mặt trong mảng A[].

#include <bits/stdc++.h>

using namespace std;

int main()

{

int t;

cin >> t;

while(t--)

{

int n;

cin >> n;

int a[n + 5];

priority\_queue <int, vector<int>, greater <int> > q;

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

{

cin >> a[i];

}

int k, x, l, r;

cin >> k >> x;

l = lower\_bound(a, a + n, x) - a;

r = upper\_bound(a, a + n, x) - a;

while(a[l] == x && l >= 0) l--;

while(a[r] == x && r < n) r++;

while(k > 0)

{

if(l >= 0)

{

q.push(a[l--]);

k--;

}

if(r < n)

{

q.push(a[r++]);

k--;

}

if(l < 0 && r >= n) break;

}

while(!q.empty())

{

cout << q.top() << " ";

q.pop();

}

cout << endl;

}

}

Cho hai số tự nhiên n, m. Nhiệm vụ của bạn là xác định xem có thể chia các số từ 1 đến n thành hai tập sao cho giá trị tuyệt đối của tổng hai tập là m và tổng các phần tử của cả hai tập là các số đồng nguyên tố (co-prime : nguyên tố cùng nhau) hay không? Ví dụ n =5, m = 7 ta có kết quả là Yes vì ta chia thành 2 tập {1, 2, 3, 5} và 4 có giá trị tuyệt đối của tổng hai tập là 7 và là các số nguyên tố cùng nhau. Với n=6, m=3 ta có câu trả lời là No vì ta có thể tìm ra hai tập {1, 2, 4, 5} và {3, 6} có trị tuyệt đối của tổng là 3 tuy nhiên cặp 12=1+2+4+5 và 9=3 + 6 không là đồng nguyên tố.

#include<bits/stdc++.h>

using namespace std;

long long bcnn(long long a,long long b){

if(!a){

return b;

}

return bcnn(b%a,a);

}

int prime(int n,int m){

int x;int y;

if((n-m)%2!=0){

return false;

}

x=(n-m)/2;

y=m+x;

if(bcnn(x,y)==1){

return true;

}else{

return false;

}

}

int main(){

int t;

cin>>t;

while(t--){

int n,m;

cin>>n>>m;

n=(n\*(n+1))/2;

if(prime(n,m)){

cout<<"Yes"<<endl;

}else{

cout<<"No"<<endl;

}

}

}

ho mảng các số nguyên dương A[] gồm n số. Hãy tìm tất cả các số nguyên dương K sao cho tất cả các phần tử của mảng A[] lấy phần dư với K đều bằng nhau. Ví dụ với mảng A[] = {6, 38, 34} ta tìm được các số K = {1, 2, 4} vì:

6%1 = 38%1 = 34%1 =0; 6%2 = 38%2 = 34%2 =0; 6%4 = 38%4 = 34%4 =2;

#include<bits/stdc++.h>

using namespace std;

vector<int> divisor(int n){

vector<int> res;

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

if(n%i==0){

res.push\_back(i);

if(i!=n/i)

res.push\_back(n/i);

}

}

return res;

}

int n,a[100000];

void solve(){

cin>>n;

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

cin>>a[i];

}

int m=\*min\_element(a,a+n);

int M=\*max\_element(a,a+n);

int x=M-m;

vector<int> d=divisor(x);

int dem=0;

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

int r=a[0]%d[i];

bool flag=true;

for(int j=1;j<n;j++){

if(a[j]%d[i]!=r){

flag=false;

break;

}

}

if(flag){

dem++;

}

}

cout<<dem<<endl;

}

int main(){

int t;

cin>>t;

while(t--){

solve();

}

}

Tìm số nghiệm của phương trình đồng dư x2 = 1(mod) p trong khoảng [1,b].  Ví dụ với b=5, p=7 ta tìm được x = 1 €[1,5] để x2 = 1 %7=1. Với b = 8, p=6 ta tìm được x = {1, 5, 7} để x2 = 1(mod 7).

6%1 = 38%1 = 34%1 =0; 6%2 = 38%2 = 34%2 =0; 6%4 = 38%4 = 34%4 =2;

#include<bits/stdc++.h>

using namespace std;

int main(){

long long b,dem,l,i;

long long n,t,k;

cin>>k;

while(k--){

cin>>b>>n;

l=b/n;

b=b%n;

t=0;

for(i=1;i<=b;i++)

if(i\*i%n==1)

t++;

dem=t;

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

if(i\*i%n==1)

t++;

dem=dem+l\*t;

cout<<dem<<endl;

}

}

Cho xâu ký tự S. Nhiệm vụ của bạn là hãy tìm độ dài xâu con nhỏ nhất của S chứa đầy đủ các ký tự của S mỗi ký tự ít nhất một lần. Ví dụ với xâu S=”aabcbcdbca” ta có xâu con nhỏ nhất là “dbca”.

#include<bits/stdc++.h>

using namespace std;

inline int tonum(char c){

return c-'0';

}

int solve(string s){

set<char> se;

for(char x:s){

se.insert(x);

}

int cnt=se.size();

int fre[256]={0};

int dem=0;

int start=0,ans=INT\_MAX;

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

fre[s[i]]++;

if(fre[s[i]]==1)

dem++;

if(dem==cnt){

while(fre[s[start]]>1){

fre[s[start]]--;

start++;

}

ans=min(ans,i-start+1);

}

}

return ans;

}

int main(){

int t;

cin>>t;

while(t--){

string s;

cin>>s;

cout<<solve(s)<<endl;

}

}

Cho số tự nhiên N. Nhiệm vụ của bạn là hãy kiểm tra N có phải là số hoàn hảo hay không. Một số N được gọi là số hoàn hảo nếu tổng các ước số của nó bằng chính nó. Ví dụ N = 6=1 + 2 + 3 là số hoàn hảo.

#include<bits/stdc++.h>

using namespace std;

int ngto(int n){

int i;

if(n==0||n==1){

return 0;

}else{

for(int i=2;i<=sqrt(n);i++){

if(n%i==0){

return 0;

}

}

return 1;

}

}

vector<long long> res;

void init(){

for(int i=1;i<=32;i++){

if(ngto(i)){

long long x=pow(2,i);

if(ngto(x-1)){

long long h=x/2\*(x-1);

res.push\_back(h);

}

}

}

}

bool check(long long n){

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

if(res[i]==n)

return true;

}

return false;

}

int main(){

init();

int t;

cin>>t;

while(t--){

long long n;

cin>>n;

if(check(n)){

cout<<"1"<<endl;

}else{

cout<<"0"<<endl;

}

}

}

Tính toán giá trị đa thức P(n, x) = an-1xn-1 + an-2xn-2 +..+ a0. Cho hai số n và r, hãy tìm C(n, r)% P. Trong đó, P = 109+7.

#include<bits/stdc++.h>

using namespace std;

long long mod=1000000007;

long long n,x;

void solve(){

long long a[n+1];

long long res=0;

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

cin>>a[i];

}

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

res=x\*(res+a[i])%mod;

}

res+=a[n-1];

cout<<res<<endl;

}

int main(){

int t;

cin>>t;

while(t--){

cin>>n>>x;

solve();

}

}

Cho mảng A[] gồm n số nguyên dương và số k. Nhiệm vụ của bạn là đếm số các cặp phần tử có hiệu nhỏ hơn k. Ví dụ A[] = {1, 10, 4, 2 }, k=3 ta nhận được kết quả là 2 tương ứng với hiệu các cặp (1, 2), (4, 2).

#include<bits/stdc++.h>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n,x;

cin>>n>>x;

int a[n];

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

cin>>a[i];

}

long long dem=0;

sort(a,a+n);

int hie=0;

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

int j=lower\_bound(a,a+n,a[i]+x)-a;

hie=j-i-1;

dem+=hie;

}

cout<<dem<<endl;

}

}

Matrix 6

[Bài làm tốt nhất](http://code.ptit.edu.vn/student/question/CPPMAT06/top)

Cho ma trận vuông A[N][N]. Hãy in các phần tử thuộc vùng biên.

#include<bits/stdc++.h>

using namespace std;

int main(){

int t;

cin>>t;

while(t--){

int n;

cin>>n;

int a[n][n];

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

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

cin>>a[i][j];

}

}

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

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

if(i==0||i==n-1||j==0||j==n-1){

cout<<a[i][j]<<" ";

}else{

cout<<" ";

}

}cout<<endl;

}

cout<<endl;

}

}

Cho ma trận vuông A[N][N]. Hãy in các phần tử thuộc theo hình con rắn.

#include<bits/stdc++.h>

using namespace std;

const int oo = 105;

int A[oo][oo];

int main(){

int test; cin >> test;

while(test--){

int n; cin >> n;

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

for(int j=0; j<n; j++) cin >> A[i][j];

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

// neu la hang chan thi ta in tu 0->n

if(i % 2 == 0){

for(int j=0; j<n; j++) cout << A[i][j] << " ";

}//le thi in tu n-1 -> 0

else{

for(int j=n-1; j>=0; j--) cout << A[i][j] << " ";

}

}cout << endl;

}

}

Cho ma trận A[N][M] chỉ bao gồm các số 0 và 1. Hãy sửa đổi các phần tử của ma trận A[][] theo nguyên tắc: nếu phần tử A[i][j] = 1 ta thay tất cả các phần tử của hàng i, cột j bởi 1. Ví dụ với ma trận dưới đây sẽ minh họa cho phép biến đổi:

#include<bits/stdc++.h>

#define ll long long

using namespace std;

const int oo = 105;

int A[oo][oo], B[oo][oo];

// dung 1 mang B de mang A ko thay doi

int main(){

int test; cin >> test;

while(test--){

int n, m; cin >> n >> m;

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

for(int j=0; j<m; j++){cin >> A[i][j];

B[i][j] = A[i][j];

}

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

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

if(A[i][j] == 1){

// cho cot j = 1

for(int k=0; k<n; k++) B[k][j] = 1;

// cho hang i bang 1

for(int k=0; k<m; k++) B[i][k] = 1;

}

}

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

for(int j=0; j<m; j++) cout << B[i][j] << " ";

cout << endl;

}cout << endl;

}

}

Cho xâu ký tự S bao gồm các số từ 0 đến 9. Nhiệm vụ của bạn là đếm số các xâu con của S là số chia hết cho 8 và không chia hết cho 3.

#include<bits/stdc++.h>

#include<string.h>

#include<ctype.h>

using namespace std;

string s;

long long solve(int k){

int dem=0;

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

int sum=0;

for(int j=i;j<s.length();j++){

sum=sum\*10+s[j]-'0';

sum=sum%k;

if(sum==0){

dem++;

}

}

}

return dem;

}

int main(){

int t;

cin>>t;

while(t--){

cin>>s;

cout<<solve(8)-solve(24)<<endl;

}

}