# a1x1+a2x2+…+anxn = d

scanf**(**"%d%d"**,&**n**,&**d**);**

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

scanf**(**"%d"**,&**a**[**i**]);**

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

**if(**d**<**a**[**i**])**

**{**

printf**(**"NO\n"**);**

exit**(**0**);**

**}**

dd**[**0**]=true;**

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

**for(**int j**=**a**[**i**];**j**<=**d**;++**j**)**

**if(!**dd**[**j**]&&**dd**[**j**-**a**[**i**]])**

**{**

dd**[**j**]=true;**

sc**[**j**]=**i**;**

**}**

**if(!**dd**[**d**])**

**{**

printf**(**"NO\n"**);**

exit**(**0**);**

**}**

**for(**int i**=**d**;**i**>**0**;**i**-=**a**[**sc**[**i**]])**

**++**cnt**[**sc**[**i**]];**

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

**++**cnt**[**i**];**

printf**(**"YES\n"**);**

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

**{**

**if(**i**>**1**)**

printf**(**" "**);**

printf**(**"%d"**,**cnt**[**i**]);**

**}**

# BFS

vector **<**int**>** a**[**2309**];**

int d**[**2309**];**

int n**,**m**;**

void bfs**()**

**{**

int u**,**v**;**

queue **<**int**>** qu**;**

qu**.**push**(**1**);**

d**[**1**]=**1**;**

**while(**qu**.**size**())**

**{**

u**=**qu**.**front**();**

qu**.**pop**();**

**for(**int i**=**0**;**v**=**a**[**u**][**i**];++**i**)**

**if(**d**[**v**]==**0**)**

**{**

d**[**v**]=**d**[**u**]+**1**;**

qu**.**push**(**v**);**

**}**

**}**

**}**

main**()**

**{**

int i**,**p**,**q**;**

scanf**(**"%d%d"**,&**n**,&**m**);**

**while(**m**--)**

**{**

scanf**(**"%d%d"**,&**p**,&**q**);**

a**[**p**].**push\_back**(**q**);**

a**[**q**].**push\_back**(**p**);**

**}**

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

a**[**i**].**push\_back**(**0**);**

bfs**();**

**}**

# BIT tang 1 phan tu

**import** java.util.Scanner;  
  
**class** BIT {  
 **int**[] **tree**;  
 **int maxValue**;  
  
 **public** BIT(**int** n) {  
 **maxValue** = n;  
 **tree** = **new int**[n + 1];  
 }  
  
 **public void** update(**int** idx, **int** val) {  
 **while** (idx <= **maxValue**) {  
 **tree**[idx] += val;  
 idx += (idx & (-idx));  
 }  
 }  
  
 **public int** Read(**int** idx) {  
 **int** sum = 0;  
 **while** (idx > 0) {  
 sum += **tree**[idx];  
 idx -= (idx & (-idx));  
 }  
 **return** sum;  
 }  
}  
  
**public class** BinaryIndexTree\_Tang1phantu {  
  
 **public static void** main(String[] args) {  
 **int** n, m, u, v, val;  
 Scanner sc = **new** Scanner(System.***in***);  
 n = sc.nextInt();  
 m = sc.nextInt();  
 BIT b = **new** BIT(n);  
 **for** (**int** i = 1; i <= m; ++i) {  
 **char** c = sc.next().charAt(0);  
 **if** (c == **'u'**) {  
 u = sc.nextInt();  
 val = sc.nextInt();  
 b.update(u, val);  
 } **else** {  
 u = sc.nextInt();  
 v = sc.nextInt();  
 System.***out***.println(b.Read(v) - b.Read(u - 1));  
 }  
 }  
 }  
}

# BIT tang mot doan

**import** java.util.Scanner;  
  
**class** bit\_tree {  
 **int n**;  
 **int**[] **T** = **new int**[123897];  
  
 **void** clear(**int** N) {  
 **int** i;  
 **n** = N;  
 **for** (i = 1; i <= **n**; i++) **T**[i] = 0;  
 }  
  
 **void** update(**int** p, **int** k) {  
 **int** i;  
 **for** (i = p; i <= **n**; i += i & -i) **T**[i] += k;  
 }  
  
 **int** get(**int** p) {  
 **int** i;  
 **int** r = 0;  
 **for** (i = p; i >= 1; i -= i & -i) r += **T**[i];  
 **return** r;  
 }  
}  
  
**class** super\_bit {  
 bit\_tree **tr**, **lazy**;  
  
 **void** clear(**int** N) {  
 **tr**.clear(N);  
 **lazy**.clear(N);  
 }  
  
 **void** update(**int** u, **int** x) {  
 **tr**.update(u, u \* x);  
 **lazy**.update(u, x);  
 }  
  
 **int** get(**int** u) {  
 **return tr**.get(u) + u \* (**lazy**.get(**tr**.**n**) - **lazy**.get(u));  
 }  
  
 **void** update(**int** ll, **int** rr, **int** x) {  
 update(rr, x);  
 update(ll - 1, x);  
 }  
  
 **int** get(**int** ll, **int** rr) {  
 **return** get(rr) - get(ll - 1);  
 }  
}  
  
**public class** BIT\_Tang\_1\_doan {  
 **public static void** main(String[] args) {  
 **int** i, u, x, n, m;  
 **char** option;  
 super\_bit tr = **new** super\_bit();  
 Scanner sc = **new** Scanner(System.***in***);  
 n = sc.nextInt();  
 m = sc.nextInt();  
 tr.clear(n);  
  
 **for** (i = 1; i <= m; i++) {  
 option = sc.next().charAt(0);  
 u = sc.nextInt();  
 x = sc.nextInt();  
 **if** (option == **'+'**) tr.update(u, x);  
 **else** System.***out***.printf(**"%d\n"**, tr.get(u, x));  
 }  
 }  
}

# GCD

public int GCD**(**int a**,** int b**)** **{**

**if** **(**b**==**0**)** **return** a**;**

**return** GCD**(**b**,**a**%**b**);**

**}**

# DFS

*//An Iterative Java program to do DFS traversal from   
//a given source vertex. DFS(int s) traverses vertices   
//reachable from s.***import** java.util.\*;  
  
**public class** GFG  
{  
 *// This class represents a directed graph using adjacency   
 // list representation* **static class** Graph  
 {  
 **int V**; *//Number of Vertices* LinkedList<Integer>[] **adj**; *// adjacency lists   
  
 //Constructor* Graph(**int** V)  
 {  
 **this**.**V** = V;  
 **adj** = **new** LinkedList[V];  
  
 **for** (**int** i = 0; i < **adj**.**length**; i++)  
 **adj**[i] = **new** LinkedList<Integer>();  
  
 }  
  
 *//To add an edge to graph* **void** addEdge(**int** v, **int** w)  
 {  
 **adj**[v].add(w); *// Add w to v’s list.* }  
  
 *// prints all not yet visited vertices reachable from s* **void** DFS(**int** s)  
 {  
 *// Initially mark all vertices as not visited* Vector<Boolean> visited = **new** Vector<Boolean>(**V**);  
 **for** (**int** i = 0; i < **V**; i++)  
 visited.add(**false**);  
  
 *// Create a stack for DFS* Stack<Integer> stack = **new** Stack<>();  
  
 *// Push the current source node* stack.push(s);  
  
 **while**(stack.empty() == **false**)  
 {  
 *// Pop a vertex from stack and print it* s = stack.peek();  
 stack.pop();  
  
 *// Stack may contain same vertex twice. So   
 // we need to print the popped item only   
 // if it is not visited.* **if**(visited.get(s) == **false**)  
 {  
 System.***out***.print(s + **" "**);  
 visited.set(s, **true**);  
 }  
  
 *// Get all adjacent vertices of the popped vertex s   
 // If a adjacent has not been visited, then puah it   
 // to the stack.* Iterator<Integer> itr = **adj**[s].iterator();  
  
 **while** (itr.hasNext())  
 {  
 **int** v = itr.next();  
 **if**(!visited.get(v))  
 stack.push(v);  
 }  
  
 }  
 }  
 }  
  
 *// Driver program to test methods of graph class* **public static void** main(String[] args)  
 {  
 *// Total 5 vertices in graph* Graph g = **new** Graph(5);  
  
 g.addEdge(1, 0);  
 g.addEdge(0, 2);  
 g.addEdge(2, 1);  
 g.addEdge(0, 3);  
 g.addEdge(1, 4);  
  
 System.***out***.println(**"Following is the Depth First Traversal"**);  
 g.DFS(0);  
 }  
}

# To hop modulo

**typedef** long long ll**;**

int test**;**

ll n**,**m**,**res**;**

ll p**[]={**2**,**3**,**5**,**3607**,**3803**};**

ll e**[]={**1**,**2**,**1**,**1**,**1**};**

void input**()**

**{**

scanf**(**"%lld%lld"**,&**m**,&**n**);**

**}**

void analysis**(**ll k**)**

**{**

ll p**=**2**,**e**;**

**while(**k**!=**1**)**

**{**

e**=**0**;**

**while(**k**%**p**==**0**)**

**{**

**++**e**;**

k**/=**p**;**

**}**

**if(**e**!=**0**)**

cout**<<**p**<<**'^'**<<**e**<<**"\n"**;**

**++**p**;**

**if(**k**>**1**&&**p**\***p**>**k**)**

p**=**k**;**

**}**

**}**

void init**()**

**{**

res**=**0**;**

analysis**(**1234567890**);**

exit**(**0**);**

**}**

ll inverse**(**ll a**,**ll m**)**

**{**

ll xa**=**1**,**xm**=**0**,**xr**,**r**,**q**;**

**while(**m**!=**0**)**

**{**

q**=**a**/**m**;**

r**=**a**-**q**\***m**;**a**=**m**;**m**=**r**;**

xr**=**xa**-**q**\***xm**;**xa**=**xm**,**xm**=**xr**;**

**}**

**return** xa**;**

**}**

ll intpower**(**ll p**,**ll k**)**

**{**

**if(**k**==**0**)**

**return** 1**;**

ll result**=**intpower**(**p**,**k**/**2**);**

result**=**result**\***result**;**

**if(**k**%**2**)**

result**=**result**\***p**;**

**return** result**;**

**}**

ll intpowermod**(**ll p**,**ll k**,**ll m**)**

**{**

**if(**k**==**0**)**

**return** 1**%**m**;**

ll result**=**intpowermod**(**p**,**k**/**2**,**m**);**

result**=**result**\***result**%**m**;**

**if(**k**%**2**)**

result**=**result**\***p**%**m**;**

**return** result**;**

**}**

void factorialanalysis**(**ll n**,**ll p**,**ll e**,**ll **&**power**,**ll **&**remainder**)**

**{**

ll temp**=**n**,**pe**;**

power**=**0**;**

**while(**temp**)**

**{**

temp**/=**p**;**

power**+=**temp**;**

**}**

remainder**=**1**;**

pe**=**intpower**(**p**,**e**);**

**while(**n**)**

**{**

temp**=**1**;**

**for(**int i**=**1**;**i**<=**pe**;++**i**)**

**if(**i**%**p**!=**0**)**

temp**=(**temp**\***i**)%**pe**;**

remainder**=**remainder**\***intpowermod**(**temp**,**n**/**pe**,**pe**)%**pe**;**

**for(**int i**=**1**;**i**<=**n**%**pe**;++**i**)**

**if(**i**%**p**!=**0**)**

remainder**=(**remainder**\***i**)%**pe**;**

n**/=**p**;**

**}**

**}**

ll getremainder**(**ll n**,**ll k**,**ll p**,**ll e**)**

**{**

ll r1**,**p1**,**r2**,**p2**,**r3**,**p3**,**modulo**,**result**;**

factorialanalysis**(**n**,**p**,**e**,**p1**,**r1**);**

factorialanalysis**(**k**,**p**,**e**,**p2**,**r2**);**

factorialanalysis**(**n**-**k**,**p**,**e**,**p3**,**r3**);**

modulo**=**intpower**(**p**,**e**);**

result**=**r1**\***intpowermod**(**p**,**p1**-**p2**-**p3**,**modulo**)%**modulo**;**

result**=**result**\***inverse**(**r2**,**modulo**)%**modulo**;**

result**=**result**\***inverse**(**r3**,**modulo**)%**modulo**;**

result**=(**result**+**modulo**)%**modulo**;**

**return** result**;**

**}**

ll solve**(**ll n**,**ll k**)**

**{**

ll a**,**m**,**c**,**y**,**result**=**0**;**

**for(**int i**=**0**;**i**<**5**;++**i**)**

**{**

a**=**getremainder**(**n**,**k**,**p**[**i**],**e**[**i**]);**

m**=**intpower**(**p**[**i**],**e**[**i**]);**

c**=**base**/**m**;**

result**=(**result**+**a**\***c**%**base**\***inverse**(**c**,**m**)%**base**)%**base**;**

**}**

result**=(**result**+**base**)%**base**;**

**return** result**;**

**}**

void solve**()**

**{**

**}**

void output**()**

**{**

printf**(**"%lld\n"**,**solve**(**m**+**n**-**1**,**n**-**1**));**

**}**

int main**()**

**{**

freopen**(**"123.inp"**,**"r"**,**stdin**);**

freopen**(**"123.out"**,**"w"**,**stdout**);**

scanf**(**"%d"**,&**test**);**

**while(**test**--)**

**{**

input**();**

init**();**

solve**();**

output**();**

**}**

**return** 0**;**

**}**

///Tim tat ca cac uoc cua 1 so

void timuoc**()**

**{**

md**=**0**;**

int t**;**

c**.**push\_back**(**1**);**

**for(**int i**=**0**;**i**<=**cnt**;++**i**)**

**{**

t**=**1**;**

**for(**int j**=**1**;**j**<=**a**[**i**].**second**;++**j**)** /// a[i].second = e[i];

**{**

t**\*=**a**[**i**].**first**;** ///a[i].first = p[i]

**for(**int z**=**0**;**z**<=**md**;++**z**)**

c**.**push\_back**(**c**[**z**]\***t**);**

**}**

md**=**c**.**size**()-**1**;**

sort**(**c**.**begin**(),**c**.**end**());**

**}**

**}**

///so catalan

#include <bits/stdc++.h>

**using** **namespace** std**;**

int n**,**hmx**;**

long long ans**=**2**;**

int h**[**100001**],**rmin**[**100001**],**a**[**200001**];**

const long long mod**=**1e9**;**

int main**(){**

cin**>>**n**>>**hmx**;**

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

scanf**(**"%d"**,&**h**[**i**]);**

rmin**[**n**]=**h**[**n**];**

**for(**int i**=**n**-**1**;**i**>**0**;**i**--){**

**if(**h**[**i**]>**rmin**[**i**+**1**])break;**

ans**++;**

rmin**[**i**]=**min**(**rmin**[**i**+**1**],**h**[**i**]);**

**}**

cout**<<**ans**<<**'\n'**;**

ans**=**1**;**

n**++;**

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

int m**=**n**+**i**;**

**for(**int p**=**2**;**p**\***p**<=**m**;**p**++)**

**while(**m**%**p**==**0**){**

a**[**p**]++;**

m**/=**p**;**

**}**

**if(**m**>**1**)**a**[**m**]++;**

m**=**i**;**

**for(**int p**=**2**;**p**\***p**<=**m**;**p**++)**

**while(**m**%**p**==**0**){**

a**[**p**]--;**

m**/=**p**;**

**}**

**if(**m**>**1**)**a**[**m**]--;**

**}**

**for(**int i**=**1**;**i**<=**2**\***n**;**i**++)**

**for(**int j**=**1**;**j**<=**a**[**i**];**j**++)**

ans**=(**ans**\***i**)%**mod**;**

cout**<<**ans**<<**'\n'**;**

**return** 0**;**

**}**

# Segment Tree – phan tu lon thu k

#include <bits/stdc++.h>

**using** **namespace** std**;**

**typedef** long long ll**;**

**typedef** int64\_t ll**;**

**typedef** pair**<**int**,**int**>** ii**;**

#define EL printf("\n")

#define pb push\_back

#define mp make\_pair

#define X first

#define Y second

**typedef** vector**<**int**>** data**;**

const int N **=** 100100**;**

int n**,** q**,** a**[**N**],** L**,** R**,** k**,** res**,** cnt**,** f**;**

data t**[**4**\***N**],** nil**;**

data combine**(**data u**,** data v**)**

**{**

data ans **=** nil**;**

int i **=** 0**,** j **=** 0**;**

**while** **(**i **<** u**.**size**()** **and** j **<** v**.**size**())** **{**

**if** **(**u**[**i**]** **<** v**[**j**])** ans**.**pb**(**u**[**i**++]);**

**else** ans**.**pb**(**v**[**j**++]);**

**}**

**while** **(**i **<** u**.**size**())** ans**.**pb**(**u**[**i**++]);**

**while** **(**j **<** v**.**size**())** ans**.**pb**(**v**[**j**++]);**

**return** ans**;**

**}**

void build**(**int k**,** int l**,** int r**)**

**{**

**if** **(**l **==** r**)** **{**

t**[**k**].**pb**(**a**[**l**]);**

**return** **;**

**}**

int mid **=** **(**l**+**r**)/**2**;**

build**(**k**\***2**,** l**,** mid**);**

build**(**k**\***2**+**1**,** mid**+**1**,** r**);**

t**[**k**]** **=** combine**(**t**[**k**\***2**],** t**[**k**\***2**+**1**]);**

**}**

void get**(**int node**,** int l**,** int r**)**

**{**

**if** **(**r **<** L **or** R **<** l**)** **return** **;**

**if** **(**L **<=** l **and** r **<=** R**)** **{**

int i **=** 0**,** j **=** t**[**node**].**size**()-**1**,** pos **=** **-**1**;**

**while** **(**i **<=** j**)** **{**

int mid **=** **(**i**+**j**)/**2**;**

**if** **(**t**[**node**][**mid**]** **<=** res**)** **{**

pos **=** mid**;**

i **=** mid**+**1**;**

**}**

**else** j **=** mid**-**1**;**

**}**

**if** **(**pos **==** **-**1**)** **return** **;**

**if** **(**t**[**node**][**pos**]** **==** res**)** f **=** **true;**

cnt **+=** pos **+** 1**;**

**if** **(**t**[**node**][**pos**]** **==** res**)** cnt**--;**

**return** **;**

**}**

int mid **=** **(**l**+**r**)/**2**;**

get**(**node**\***2**,**l**,**mid**);**

get**(**node**\***2**+**1**,**mid**+**1**,**r**);**

**}**

int main**()**

**{**

//freopen("INP.INP","r",stdin);

//freopen("OUT.OUT","w",stdout);

scanf**(**"%d"**,** **&**n**);**

**for** **(**int i**=**1**;**i**<=**n**;**i**++)** scanf**(**"%d"**,** **&**a**[**i**]);**

build**(**1**,**1**,**n**);**

scanf**(**"%d"**,** **&**q**);**

**while** **(**q**--)** **{**

scanf**(**"%d%d%d"**,** **&**L**,&**R**,&**k**);**

int l **=** 0**,** r **=** t**[**1**].**size**()-**1**;**

**while** **(**l **<=** r**)** **{**

int mid **=** **(**l**+**r**)/**2**;**

res **=** t**[**1**][**mid**];**

cnt **=** 0**;**

f **=** 0**;**

get**(**1**,**1**,**n**);**

**if** **(**cnt **==** k**-**1 **and** f**)** **{**

printf**(**"%d\n"**,** res**);**

**break;**

**}**

**if** **(**cnt **<** k**)** l **=** mid**+**1**;** **else** r **=** mid**-**1**;**

**}**

**}**

**return** 0**;**

**}**

# Segment tree – phan tu, lay max cua mot doan

int n**,**m**;**

int a**[**maxn**];**

struct st

**{**

ll sum**,**suml**,**sumr**,**res**;**

**};**

st tree**[**131073**];**

void update**(**int node**,**int l**,**int r**)**

**{**

**if(**l**==**r**)**

**{**

tree**[**node**].**sum**=**tree**[**node**].**suml**=**tree**[**node**].**sumr**=**tree**[**node**].**res**=**a**[**l**];**

**return;**

**}**

int mid**=(**l**+**r**)>>**1**;**

update**(**2**\***node**,**l**,**mid**);**

update**(**2**\***node**+**1**,**mid**+**1**,**r**);**

tree**[**node**].**sum**=**tree**[**2**\***node**].**sum**+**tree**[**2**\***node**+**1**].**sum**;**

tree**[**node**].**suml**=**maxi**(**tree**[**2**\***node**].**suml**,**tree**[**2**\***node**].**sum**+**tree**[**2**\***node**+**1**].**suml**);**

tree**[**node**].**sumr**=**maxi**(**tree**[**2**\***node**+**1**].**sumr**,**tree**[**2**\***node**].**sumr**+**tree**[**2**\***node**+**1**].**sum**);**

tree**[**node**].**res**=**maxi**(**maxi**(**tree**[**2**\***node**].**res**,**tree**[**2**\***node**+**1**].**res**),**tree**[**2**\***node**].**sumr**+**tree**[**2**\***node**+**1**].**suml**);**

**}**

st mst**(**ll a**,**ll b**,**ll c**,**ll d**)**

**{**

st ans**;**

ans**.**sum**=**a**;**ans**.**suml**=**b**;**ans**.**sumr**=**c**;**ans**.**res**=**d**;**

**return** ans**;**

**}**

bool bang**(**st a**,**st b**)**

**{**

**return** **(**a**.**sum**==**b**.**sum**&&**a**.**suml**==**b**.**suml**&&**a**.**sumr**==**b**.**sumr**&&**a**.**res**==**b**.**res**);**

**}**

st get**(**int node**,**int L**,**int R**,**int l**,**int r**)**

**{**

st ct**=**mst**(-**oo**,-**oo**,-**oo**,-**oo**);**

**if(**l**>**R**||**r**<**L**)**

**return** ct**;**

**if(**l**<=**L**&&**R**<=**r**)**

**return** tree**[**node**];**

st id1**,**id2**;**

int mid**=(**L**+**R**)>>**1**;**

id1**=**get**(**2**\***node**,**L**,**mid**,**l**,**r**);**id2**=**get**(**2**\***node**+**1**,**mid**+**1**,**R**,**l**,**r**);**

**if(**bang**(**id1**,**ct**))**

**return** id2**;**

**if(**bang**(**id2**,**ct**))**

**return** id1**;**

**return** mst**(**id1**.**sum**+**id2**.**sum**,**maxi**(**id1**.**suml**,**id1**.**sum**+**id2**.**suml**),**maxi**(**id2**.**sumr**,**id1**.**sumr**+**id2**.**sum**),**maxi**(**maxi**(**id1**.**res**,**id2**.**res**),**id1**.**sumr**+**id2**.**suml**));**

**}**

void input**()**

**{**

scanf**(**"%d"**,&**n**);**

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

scanf**(**"%d"**,&**a**[**i**]);**

update**(**1**,**1**,**n**);**

scanf**(**"%d"**,&**m**);**

int u**,**v**;**

**for(**int i**=**1**;**i**<=**m**;++**i**)**

**{**

scanf**(**"%d%d"**,&**u**,&**v**);**

printf**(**"%lld\n"**,**get**(**1**,**1**,**n**,**u**,**v**).**res**);**

**}**

**}**

# Hoan vi thu k cua n!

#include

#include

#define fi "HOANVI.INP"

**using** **namespace** std**;**

**typedef** unsigned long long ull**;**

ull dem **=** 0**;**

int n**;**

ull k**;**

ull **\*** gt**;**

int **\*** ThuTu**;**

void Nhap**()** **{**

fstream f**(**fi**,** ios **::** in**);**

f **>>** n**;**

f **>>** k**;**

f**.**close**();**

**}**

void Init**()** **{**

ThuTu **=** **new** int**[**n**+**1**];**

int i**;**

**for** **(**i**=**1**;** i**<=**n**;** i**++)** ThuTu**[**i**]** **=** i**;**

**}**

void XoaViTri**(**int k**)** **{**

int i**;**

**for** **(**i**=**k**;** i**<**n**;** i**++)** ThuTu**[**i**]** **=** ThuTu**[**i**+**1**];**

n**--;**

**}**

void TinhGiaiThua**(**int n**)** **{**

gt **=** **new** ull**[**n**+**1**];**

gt**[**0**]** **=** 1**;**

int i**;**

**for** **(**i**=**1**;** i**<=**n**;** i**++)** gt**[**i**]** **=** i**\***gt**[**i**-**1**];** **}** void ThucHien**()** **{** int x**;** **while** **(**n **>** 0**)** **{**

//x = TimKiemNhiPhan(k, 1, n, n);

x **=** gt**[**n**]/**gt**[**n**-**1**];**

**if** **(**gt**[**n**]%**gt**[**n**-**1**]** **!=** 0**)** x**++;**

cout **<<** ThuTu**[**x**]** **<<** " "**;**

k **=** k **-** **(**x**-**1**)\***gt**[**n**-**1**];**

XoaViTri**(**x**);**

**}**

**}**

int main**()** **{**

Nhap**();**

Init**();**

TinhGiaiThua**(**n**);**

ThucHien**();**

//cout << endl << "k = " << k;

**}**

# Trie

*// Java implementation of search and insert operations   
// on Trie***public class** Trie {  
  
 *// Alphabet size (# of symbols)* **static final int *ALPHABET\_SIZE*** = 26;  
  
 *// trie node* **static class** TrieNode  
 {  
 TrieNode[] **children** = **new** TrieNode[***ALPHABET\_SIZE***];  
  
 *// isEndOfWord is true if the node represents   
 // end of a word* **boolean isEndOfWord**;  
  
 TrieNode(){  
 **isEndOfWord** = **false**;  
 **for** (**int** i = 0; i < ***ALPHABET\_SIZE***; i++)  
 **children**[i] = **null**;  
 }  
 };  
  
 **static** TrieNode *root*;  
  
 *// If not present, inserts key into trie   
 // If the key is prefix of trie node,   
 // just marks leaf node* **static void** insert(String key)  
 {  
 **int** level;  
 **int** length = key.length();  
 **int** index;  
  
 TrieNode pCrawl = *root*;  
  
 **for** (level = 0; level < length; level++)  
 {  
 index = key.charAt(level) - **'a'**;  
 **if** (pCrawl.**children**[index] == **null**)  
 pCrawl.**children**[index] = **new** TrieNode();  
  
 pCrawl = pCrawl.**children**[index];  
 }  
  
 *// mark last node as leaf* pCrawl.**isEndOfWord** = **true**;  
 }  
  
 *// Returns true if key presents in trie, else false* **static boolean** search(String key)  
 {  
 **int** level;  
 **int** length = key.length();  
 **int** index;  
 TrieNode pCrawl = *root*;  
  
 **for** (level = 0; level < length; level++)  
 {  
 index = key.charAt(level) - **'a'**;  
  
 **if** (pCrawl.**children**[index] == **null**)  
 **return false**;  
  
 pCrawl = pCrawl.**children**[index];  
 }  
  
 **return** (pCrawl != **null** && pCrawl.**isEndOfWord**);  
 }  
  
 *// Driver* **public static void** main(String args[])  
 {  
 *// Input keys (use only 'a' through 'z' and lower case)* String keys[] = {**"the"**, **"a"**, **"there"**, **"answer"**, **"any"**,  
 **"by"**, **"bye"**, **"their"**};  
  
 String output[] = {**"Not present in trie"**, **"Present in trie"**};  
  
  
 *root* = **new** TrieNode();  
  
 *// Construct trie* **int** i;  
 **for** (i = 0; i < keys.**length** ; i++)  
 *insert*(keys[i]);  
  
 *// Search for different keys* **if**(*search*(**"the"**) == **true**)  
 System.***out***.println(**"the --- "** + output[1]);  
 **else** System.***out***.println(**"the --- "** + output[0]);  
  
 **if**(*search*(**"these"**) == **true**)  
 System.***out***.println(**"these --- "** + output[1]);  
 **else** System.***out***.println(**"these --- "** + output[0]);  
  
 **if**(*search*(**"their"**) == **true**)  
 System.***out***.println(**"their --- "** + output[1]);  
 **else** System.***out***.println(**"their --- "** + output[0]);  
  
 **if**(*search*(**"thaw"**) == **true**)  
 System.***out***.println(**"thaw --- "** + output[1]);  
 **else** System.***out***.println(**"thaw --- "** + output[0]);  
  
 }  
}  
*// This code is contributed by Sumit Ghosh*

# Ham Z: Cho một chuỗi S có độ dài n, thuật toán Z Function tạo ra một mảng Z mà tại mỗi vị trí i, ta có Zi là độ dài chuỗi con dài nhất là tiền tố của S bắt đầu tại vị trí i,

**import** java.util.Scanner;  
  
**public class** HamZ {  
 **public static void** main(String[] args) {  
 Scanner sc = **new** Scanner(System.***in***);  
 **int** l = 0, r = 0;  
 String S = sc.nextLine();  
*// s = "0" + s;* **int** n = S.length();  
 **int**[] Z = **new int**[100001];  
 **int** L = 0, R = 0;  
 Z[0] = n;  
 **for** (**int** i = 1; i < n; i++)  
 **if** (i > R) {  
 L = R = i;  
 **while** (R < n && S.charAt(R) == S.charAt(R - L)) R++;  
 Z[i] = R - L;  
 R--;  
 } **else** {  
 **int** k = i - L;  
 **if** (Z[k] < R - i + 1) Z[i] = Z[k];  
 **else** {  
 L = i;  
 **while** (R < n && S.charAt(R) == S.charAt(R - L)) R++;  
 Z[i] = R - L;  
 R--;  
 }  
 }  
 **for**(**int** i=0;i<n;++i)  
 System.***out***.print(Z[i]+**" "**);  
 }  
}

# Cho mảng a. tìm ra đoạn l -> r có min(a[l..r])\*(r-l+1) max nhất có thể trong các đoạn l -> r.

int test**,**n**,**res1**,**res2**;**

int a**[**maxn**],**l**[**maxn**],**r**[**maxn**];**

ll res**;**

void input**()**

**{**

scanf**(**"%d"**,&**n**);**

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

scanf**(**"%d"**,&**a**[**i**]);**

**}**

void init**()**

**{**

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

**{**

l**[**i**]=**i**;**

**while(**a**[**i**]<=**a**[**l**[**i**]-**1**])**

l**[**i**]=**l**[**l**[**i**]-**1**];**

**}**

**for(**int i**=**n**;**i**>=**1**;--**i**)**

**{**

r**[**i**]=**i**;**

**while(**a**[**i**]<=**a**[**r**[**i**]+**1**])**

r**[**i**]=**r**[**r**[**i**]+**1**];**

**}**

res**=**0**;**

**}**

void solve**()**

**{**

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

**if(**res**<**1ll**\***a**[**i**]\*(**r**[**i**]-**l**[**i**]+**1**))**

**{**

res**=**1ll**\***a**[**i**]\*(**r**[**i**]-**l**[**i**]+**1**);**

res1**=**l**[**i**];**res2**=**r**[**i**];**

**}**

**}**

void output**()**

**{**

printf**(**"%lld %d %d\n"**,**res**,**res1**,**res2**);**

**}**

# Hash

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

**using** **namespace** std**;**

#define long long long

const int N**=**1000006**,** BASE**=**1000000007**;**

int m**,** n**;**

char a**[**N**],** b**[**N**];**

long A**[**N**],** B**[**N**],** M**[**N**];**

void hash\_build**(**char a**[],** int n**,** long H**[])** **{**

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

H**[**i**]** **=** **(**H**[**i**-**1**]** **\*** M**[**1**]** **+** a**[**i**])** **%** BASE**;**

**}**

long hash\_range**(**long H**[],** int L**,** int R**)** **{**

**return** **(**H**[**R**]** **-** H**[**L**-**1**]\***M**[**R**-**L**+**1**]** **+** 1LL**\***BASE**\***BASE**)** **%** BASE**;**

**}**

main**()** **{**

M**[**0**]=**1**;** M**[**1**]=**2309**;**

**for** **(**int i**=**2**;** i**<**N**;** i**++)**

M**[**i**]** **=** M**[**i**-**1**]** **\*** M**[**1**]** **%** BASE**;**

scanf**(**"%s"**,** a**+**1**);** m**=**strlen**(**a**+**1**);**

scanf**(**"%s"**,** b**+**1**);** n**=**strlen**(**b**+**1**);**

hash\_build**(**a**,** m**,** A**);**

hash\_build**(**b**,** n**,** B**);**0

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

**if** **(**hash\_range**(**A**,** i**,** i**+**n**-**1**)** **==** B**[**n**])**

printf**(**"%d "**,** i**);**

**}**

printf**(**"\n"**);**

**}**

////////////////////////////////////////////////////

/// Hash không quan trọng đúng thứ tự

/// A[] = {2,3,4};

/// B[] = {1,4,2,3};

/// day lien tiep xuat hien : 2,3,4

#include <cstdio>

#include <algorithm>

#include <cmath>

#include <cstdlib>

#include <vector>

#include <queue>

#include <deque>

#include <stack>

#include <climits>

#include <cstring>

#include <ctime>

#include <fstream>

#include <iomanip>

#include <iostream>

#include <map>

#include <set>

#include <sstream>

**using** **namespace** std**;**

#define maxi(a,b) ((a)>(b)?(a):(b))

#define mini(a,b) ((a)<(b)?(a):(b))

#define ttd(a) ((a)>0?(a):-(a))

#define maxn 100001

#define base 1000000007ll

**typedef** long long ll**;**

ll n**,**m**,**res**;**

ll a**[**maxn**],**b**[**maxn**];**

ll M**[**maxn**],**A**[**maxn**],**B**[**maxn**];**

void hash\_build**(**ll a**[],** ll n**,** ll H**[])**

**{**

**for** **(**ll i**=**1**;** i**<=**n**;** **++**i**)**

H**[**i**]** **=** **(**H**[**i**-**1**]** **+** M**[**a**[**i**]]** **)** **%** base**;**

**}**

ll hash\_range**(**ll H**[],** ll L**,** ll R**)**

**{**

**return** **(**H**[**R**]** **-** H**[**L**-**1**]** **+** 1ll**\***base**\***base**)** **%** base**;**

**}**

void input**()**

**{**

M**[**0**]=**1**;** M**[**1**]=**2309**;**

**for** **(**ll i**=**2**;** i**<**maxn**;** **++**i**)**

M**[**i**]** **=** M**[**i**-**1**]** **\*** M**[**1**]** **%** base**;**

scanf**(**"%lld"**,&**n**);**

**for(**ll i**=**1**;**i**<=**n**;++**i**)**

scanf**(**"%lld"**,&**a**[**i**]);**

hash\_build**(**a**,**n**,**A**);**

scanf**(**"%lld"**,&**m**);**

**for(**ll i**=**1**;**i**<=**m**;++**i**)**

scanf**(**"%lld"**,&**b**[**i**]);**

hash\_build**(**b**,**m**,**B**);**

**}**

void init**()**

**{**

**if(**n**>**m**)**

**{**

printf**(**"NO\n"**);**

exit**(**0**);**

**}**

res**=**m**+**1**;**

**}**

void solve**()**

**{**

**for(**ll i**=**n**;**i**<=**m**;++**i**)**

**if(**A**[**n**]==**hash\_range**(**B**,**i**-**n**+**1**,**i**))**

**{**

res**=**i**-**n**+**1**;**

**break;**

**}**

**}**

void output**()**

**{**

**if(**res**==**m**+**1**)**

printf**(**"NO\n"**);**

**else**

printf**(**"YES\n%lld\n"**,**res**);**

**}**

int main**()**

**{**

freopen**(**"esign.inp"**,**"r"**,**stdin**);**

freopen**(**"esign.out"**,**"w"**,**stdout**);**

input**();**

init**();**

solve**();**

output**();**

**return** 0**;**

**}**

# Nhan an do

long long X**(**long long a**,**long long b**)**

**{**

long long res**=**0**;**

**while(**b**>**0**){**

**if((**b**%**2**)==**1**){**

res**+=**a**;**

res**%=**md**;}**

a**\*=**2**;**

a**%=**md**;**

b**/=**2**;}**

**return** res**%**md**;**

**}**

# Phuong trinh ax + by = c Nghiem nguyen

ll a**,**b**,**c**;**

int res**;**

void input**()**

**{**

scanf**(**"%lld%lld"**,&**a**,&**b**);**

**}**

void init**()**

**{**

**}**

ii nghiem**(**ll a**,**ll b**)**

**{**

ll m**=**a**,**n**=**b**;**

ll xm**=**1**,**xn**=**0**,**q**,**xr**,**r**;**

**while(**n**)**

**{**

q**=**m**/**n**;**

r**=**m**-**q**\***n**;**

xr**=**xm**-**q**\***xn**;**

m**=**n**;**xm**=**xn**;**

n**=**r**;**xn**=**xr**;**

**}**

**return** ii**(**m**,**xm**);**

**}**

void solve**()**

**{**

cout**<<**nghiem**(**a**,**b**).**second**<<**" "**<<(**nghiem**(**a**,**b**).**first**-**a**\***nghiem**(**a**,**b**).**second**)/**b**;**

**}**

# Segmenttree – tang 1 doan, tinh tong

#include <bits/stdc++.h>

#define N 50010

**using** **namespace** std**;**

int n**,** q**,** L**,** R**;**

char t**;**

int64\_t add**[**8**\***N**],** tree**[**8**\***N**];**

int64\_t v**,** res**;**

void update**(**int node**,** int l**,** int r**)**

**{**

tree**[**node**]** **+=** **(**r**-**l**+**1**)\***add**[**node**];**

add**[**node**\***2**]** **+=** add**[**node**];**

add**[**node**\***2**+**1**]** **+=** add**[**node**];**

add**[**node**]** **=** 0**;**

**if(**r**<**L **or** R**<**l**)** **return;**

**if(**L**<=**l **and** r**<=**R**)**

**{**

tree**[**node**]** **+=** **(**r**-**l**+**1**)\***v**;**

add**[**node**\***2**]** **+=** v**;**

add**[**node**\***2**+**1**]** **+=** v**;**

**return;**

**}**

int m**=(**l**+**r**)/**2**;**

update**(**node**\***2**,** l**,** m**);**

update**(**node**\***2**+**1**,** m**+**1**,** r**);**

tree**[**node**]** **=** tree**[**node**\***2**]** **+** tree**[**node**\***2**+**1**];**

**}**

void get**(**int node**,** int l**,** int r**)**

**{**

tree**[**node**]** **+=** **(**r**-**l**+**1**)\***add**[**node**];**

add**[**node**\***2**]** **+=** add**[**node**];**

add**[**node**\***2**+**1**]** **+=** add**[**node**];**

add**[**node**]** **=** 0**;**

**if(**r**<**L **or** R**<**l**)** **return;**

**if(**L**<=**l **and** r**<=**R**)**

**{**

res **+=** tree**[**node**];**

**return;**

**}**

int m**=(**l**+**r**)/**2**;**

get**(**node**\***2**,** l**,** m**);**

get**(**node**\***2**+**1**,** m**+**1**,** r**);**

**}**

int main**()**

**{**

freopen**(**"123.inp"**,**"r"**,**stdin**);**

scanf**(**"%d%d"**,&**n**,&**q**);**

**while(**q**--)**

**{**

scanf**(**" %c"**,&**t**);**

**if(**t**==**'+'**)**

**{**

scanf**(**"%d%d%lld"**,** **&**L**,** **&**R**,** **&**v**);**

update**(**1**,** 1**,** n**);**

**}**

**if(**t**==**'?'**)**

**{**

scanf**(**"%d%d"**,** **&**L**,** **&**R**);**

res**=**0**;**

get**(**1**,** 1**,** n**);**

printf**(**"%lld\n"**,**res**);**

**}**

**}**

**return** 0**;**

**}**

# Giao diem

#include <bits/stdc++.h>

**using** **namespace** std**;**

struct point

**{**

double x**,**y**;**

**};**

point make\_point**(**double a**,**double b**)**

**{**

point result**;**

result**.**x**=**a**;**result**.**y**=**b**;**

**return** result**;**

**}**

double cp**(**point u**,**point v**)**

**{**

**return** u**.**x**\***v**.**y**-**u**.**y**\***v**.**x**;**

**}**

point solvesle**(**point a**,**point b**,**point c**)**

**{**

double delta**=**cp**(**a**,**b**);**

**return** make\_point**(**cp**(**c**,**b**)/**delta**,**cp**(**a**,**c**)/**delta**);**

**}**

point **operator** **-** **(**point a**,**point b**)**

**{**

**return** make\_point**(**a**.**x**-**b**.**x**,**a**.**y**-**b**.**y**);**

**}**

int main**()**

**{**

point a**,**b**,**c**,**d**;**

cin**>>**a**.**x**>>**a**.**y**>>**b**.**x**>>**b**.**y**>>**c**.**x**>>**c**.**y**>>**d**.**x**>>**d**.**y**;**

point r**=**solvesle**(**b**-**a**,**c**-**d**,**c**-**a**);**

///giao 2 duong thang

**if(**isnan**(**r**.**x**))**

cout**<<**"2 duong thang trung nhau"**;**

**else**

**if(**isinf**(**r**.**x**))**

cout**<<**"2 duong thang song song"**;**

**else**

**{**

p**=**a**+(**b**-**a**)\***r**.**x**;**

cout**<<**fixed**<<**setprecision**(**4**)<<**p**.**x**<<**" "**<<**p**.**y**;**

**}**

**}**

# Prim

cây bao trùm nhỏ nhất của một đồ thị vô hướng có trọng số liên thông

**typedef** pair**<**int**,** int**>** ii**;**

const int N**=**100005**,** oo**=**0x3c3c3c3c**;**

int n**,** m**,** d**[**N**];**

vector**<**int**>** a**[**N**],** b**[**N**];**

int prim**(**int u**)** **{**

int Sum **=** 0**;**

priority\_queue**<**ii**>** qu**;**

**for** **(**int i**=**1**;** i**<=**n**;** i**++)** d**[**i**]=**oo**;**

qu**.**push**(**ii**(**0**,** u**));** d**[**u**]=**0**;**

**while** **(**qu**.**size**())** **{**

ii Pop**=**qu**.**top**();** qu**.**pop**();**

int u**=**Pop**.**second**,** du**=-**Pop**.**first**;**

**if** **(**du**!=**d**[**u**])** **continue;**

Sum**+=**d**[**u**];** d**[**u**]=**0**;**

**for** **(**int i**=**0**;** int v**=**a**[**u**][**i**];** i**++)**

**if** **(**d**[**v**]** **>** b**[**u**][**i**])** **{**

d**[**v**]=**b**[**u**][**i**];**

qu**.**push**(**ii**(-**d**[**v**],** v**));**

**}**

**}**

**return** Sum**;**

**}**

main**()** **{**

scanf**(**"%d%d"**,** **&**n**,** **&**m**);**

**for** **(**int i**=**1**;** i**<=**m**;** i**++)** **{**

int x**,** y**,** z**;**

scanf**(**"%d%d%d"**,** **&**x**,** **&**y**,** **&**z**);**

a**[**x**].**push\_back**(**y**);**

b**[**x**].**push\_back**(**z**);**

a**[**y**].**push\_back**(**x**);**

b**[**y**].**push\_back**(**z**);**

**}**

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

a**[**i**].**push\_back**(**0**);**

cout **<<** prim**(**1**)** **<<** endl**;**

**}**

# RMQ: phần tử nhỏ nhất trong dãy từ L đến R

**for** **(**i**=**1**;** i**<=**n**;** i**++)** M**[**i**][**0**]=**a**[**i**];**

**for** **(**k**=**1**;** **(**1**<<**k**)<=**n**;** k**++)**

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

M**[**i**][**k**]** **=** max**(**M**[**i**][**k**-**1**],** M**[**i**+(**1**<<(**k**-**1**))][**k**-**1**]);**

int u**,** v**;**

**while(**m**--){**

scanf**(**"%d%d"**,** **&**u**,** **&**v**);**

k**=**log2**(**v**-**u**+**1**);**

printf**(**"%d\n"**,** max**(**M**[**u**][**k**],**M**[**v**-(**1**<<**k**)+**1**][**k**]));**

**}**

# Segment Tree: Chen x vao vi tri y

int Max**[**N**],** Size**[**N**],** Height**[**N**],** Left**[**N**],** Right**[**N**],** Peak**;**

void update**(**int id**){**

int ll**=**Left**[**id**],** rr**=**Right**[**id**];**

Max**[**id**]=**max**(**Max**[**ll**],** Max**[**rr**]);**

Size**[**id**]=**Size**[**ll**]+**Size**[**rr**];**

Height**[**id**]=**max**(**Height**[**ll**],** Height**[**rr**])+**1**;**

**}**

int create**(**int Value**){**

int id **=** **++**Peak**;**

Max**[**id**]=**Value**;**

Size**[**id**]=**1**;**

**return** id**;**

**}**

int link**(**int ll**,** int u**,** int rr**){**

Left**[**u**]=**ll**,** Right**[**u**]=**rr**;**

**return** update**(**u**),** u**;**

**}**

int right\_rotate**(**int u**){**

int ll **=** Left**[**u**];**

**return** link**(**Left**[**ll**],** ll**,** link**(**Right**[**ll**],** u**,** Right**[**u**]));**

**}**

int left\_rotate**(**int u**){**

int rr **=** Right**[**u**];**

**return** link**(**link**(**Left**[**u**],** u**,** Left**[**rr**]),** rr**,** Right**[**rr**]);**

**}**

int balance**(**int u**){**

**if** **(**abs**(**Height**[**Left**[**u**]]-**Height**[**Right**[**u**]])<=**1**)** **return** u**;**

//Left[u]=balance(Left[u]); // to be careful

//Right[u]=balance(Right[u]);

bool x**=**Height**[**Left**[**u**]]>**Height**[**Right**[**u**]];**

int v**=(**x**?**Left**[**u**]:**Right**[**u**]);**

bool y**=**Height**[**Left**[**v**]]>**Height**[**Right**[**v**]];**

**if** **(**x **&&** y**)** u**=**right\_rotate**(**u**);**

**if** **(!**x **&&** **!**y**)** u**=**left\_rotate**(**u**);**

**if** **(**x **&&** **!**y**)** **{** Left**[**u**]=**v**=**left\_rotate**(**v**);** u**=**right\_rotate**(**u**);** **}**

**if** **(!**x **&&** y**)** **{** Right**[**u**]=**v**=**right\_rotate**(**v**);** u**=**left\_rotate**(**u**);** **}**

**return** u**;**

**}**

struct node **{**

int ll**,** rr**,** id**;**

node **(**int L**,** int X**)** **{** ll**=**L**,** id**=**X**;** rr**=**ll**+**Size**[**id**]-**1**;** **}**

node left**(){** **return** node**(**ll**,** Left**[**id**]);** **}**

node right**(){** **return** node**(**ll**+**Size**[**Left**[**id**]],** Right**[**id**]);** **}**

void insert**(**int u**,** int Value**)** **{**

**if** **(**ll**>**u **||** u**>**rr**)** **return** **;**

**if** **(**ll**==**rr**)** **{**

Left**[**id**]=**create**(**Value**);**

Right**[**id**]=**create**(**Max**[**id**]);**

update**(**id**);** **return** **;**

**}**

left**().**insert**(**u**,** Value**);**

right**().**insert**(**u**,** Value**);**

Left**[**id**]=**balance**(**Left**[**id**]);**

Right**[**id**]=**balance**(**Right**[**id**]);**

update**(**id**);**

**}**

int max\_range**(**int L**,** int R**)** **{**

**if** **(**L**>**rr **||** ll**>**R **||** L**>**R**)** **return** **-**oo**;**

**if** **(**L**<=**ll **&&** rr**<=**R**)** **return** Max**[**id**];**

int Max1 **=** left**().**max\_range**(**L**,** R**);**

int Max2 **=** right**().**max\_range**(**L**,** R**);**

**return** max**(**Max1**,** Max2**);**

**}**

**};**

ostream**&** **operator** **<<** **(**ostream**&** cout**,** node a**){**

**if** **(**a**.**ll**==**a**.**rr**)** **return** cout **<<** Max**[**a**.**id**];**

**return** cout **<<** "(" **<<** a**.**left**()** **<<** " " **<<** a**.**right**()** **<<** ")"**;**

//return cout << a.left() << " " << a.right();

**}**

main**(){**

create**(-**oo**);**

int m**;** scanf**(**"%d"**,** **&**m**);**

**while** **(**m**--){**

char c**;** int x**,** y**;**

scanf**(**" %c%d%d"**,** **&**c**,** **&**x**,** **&**y**);**

// if (rand()&1){ c='A'; x=rand()%10\*100; y=rand()%Size[1]+1; }

// else do { c='Q'; x=rand()%Size[1]+1; y=rand()%Size[1]+1; } while (x>y||y==Size[1]);

// printf("%c %d %d (%d)\n", c, x, y, m);

**if** **(**c**==**'A'**)** node**(**1**,** 1**).**insert**(**y**,** x**);**

**else** printf**(**"%d\n"**,** node**(**1**,** 1**).**max\_range**(**x**,** y**));**

// cout << node(1,1) << endl;

**}**

**}**

# Segment Tree

Bài toán

Cho hai mảng a**,** b**,** thực hiện các thao tác**.**

1. Gán k phần tử của a tính từ vị trí x vào k phần tử của b tại vị trí y **(**gán b**[**y**+**i**]** bằng a**[**x**+**i**]** với mọi 0**<=**i**<**k**)**

2. In ra giá trị phần tử thứ x**.**

Độ phức tạp

O**(**logn**)** với mỗi truy vấn**.**

Code này của Nguyễn Tiến Trung Kiên

#include <stdio.h>

#include <iostream>

#include <algorithm>

**using** **namespace** std**;**

#define long long long

#define f1(i,n) for (int i=1; i<=n; i++)

#define f0(i,n) for (int i=0; i<n; i++)

**typedef** pair**<**int**,** int**>** ii**;**

#define X first

#define Y second

#define N 2000006

int n**,** m**,** a0**[**N**],** b0**[**N**];**

int Lazy**[**N**];**

struct node **{**

int ll**,** rr**,** Index**,** Size**;**

node**(**int L**,** int R**,** int ID**){** ll**=**L**,** rr**=**R**,** Index**=**ID**,** Size**=**rr**-**ll**+**1**;** **}**

node left**(){** **return** node**(**ll**,** **(**ll**+**rr**)/**2**,** Index**\***2**);** **}**

node right**(){** **return** node**((**ll**+**rr**)/**2**+**1**,** rr**,** Index**\***2**+**1**);** **}**

void access**(){**

**if** **(**Lazy**[**Index**]** **&&** ll**!=**rr**){**

Lazy**[**Index**\***2**]=**Lazy**[**Index**];**

Lazy**[**Index**\***2**+**1**]=**Lazy**[**Index**]+**left**().**Size**;**

Lazy**[**Index**]=**0**;**

**}**

**}**

int get**(**int X**){** access**();**

**if** **(**rr**<**X **||** X**<**ll**)** **return** 0**;**

**if** **(**ll**==**rr**)** **return** Lazy**[**Index**];**

**return** left**().**get**(**X**)** **+** right**().**get**(**X**);**

**}**

int update**(**int L**,** int R**,** int X**){** access**();**

**if** **(**ll**>**R **||** L**>**rr **||** L**>**R**)** **return** 0**;**

**if** **(**L**<=**ll **&&** ll**<=**rr **&&** rr**<=**R**)** **{** Lazy**[**Index**]=**X**;** access**();** **return** Size**;** **}**

int Solved **=** left**().**update**(**L**,** R**,** X**);**

**return** Solved **+** right**().**update**(**L**,** R**,** X**+**Solved**);**

**}**

**};**

main**(){**

ios **::** sync\_with\_stdio**(false);**

cin **>>** n **>>** m**;**

f1**(**i**,**n**)** cin **>>** a0**[**i**];**

f1**(**i**,**n**)** cin **>>** b0**[**i**];**

f1**(**i**,**m**){**

int ch**,** x**,** y**,** z**;**

cin **>>** ch**;**

**if** **(**ch**==**1**)** **{** cin **>>** x **>>** y **>>** z**;** node**(**1**,**n**,**1**).**update**(**y**,** y**+**z**-**1**,** x**);** **}**

**else** **{** cin **>>** x**;** y**=**node**(**1**,**n**,**1**).**get**(**x**);** cout **<<** **(**y**==**0**?**b0**[**x**]:**a0**[**y**])** **<<** '\n'**;** **}**

**}**

**}**

# Dijkstra

#include <stdio.h>

#include <vector>

#include <queue>

**using** **namespace** std**;**

const int oo **=** 1000111000**;**

**typedef** pair **<**int**,** int**>** ii**;**

vector **<**ii**>** a**[**2309**];**

int n**,** m**;**

int d**[**2309**];**

void dijkstra**()**

**{**

priority\_queue **<**ii**,**vector**<**ii**>,**greater**<**ii**>** **>** pq**;**

int u**,**v**,**du**,**uv**;**

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

d**[**i**]=**oo**;**

d**[**1**]=**0**;**

pq**.**push**(**ii**(**0**,**1**));**

**while(**pq**.**size**())**

**{**

u**=**pq**.**top**().**second**;**

du**=**pq**.**top**().**first**;**

pq**.**pop**();**

**if(**u**==**t**)**

**return;**

**if(**du**==**d**[**u**])**

**for(**int i**=**0**;**v**=**a**[**u**][**i**].**second**;++**i**)**

**{**

uv**=**a**[**u**][**i**].**first**;**

**if(**d**[**v**]>**du**+**uv**)**

**{**

d**[**v**]=**du**+**uv**;**

pq**.**push**(**ii**(**d**[**v**],**v**));**

**}**

**}**

**}**

**}**

int main**()**

**{**

int p**,**q**,**m**,**w**;**

scanf**(**"%d%d"**,&**n**,&**m**);**

**while(**m**--)**

**{**

scanf**(**"%d%d%d"**,&**p**,&**q**,&**w**);**

a**[**p**].**push\_back**(**ii**(**w**,**q**));**

a**[**q**].**push\_back**(**ii**(**w**,**p**));**

**}**

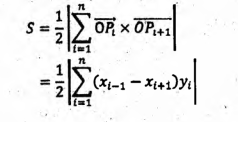
**for(**int i**=**1**;**i**<=**n**;++**i**)**

a**[**i**].**push\_back**(**ii**(**0**,**0**));**

dijkstra**();**

**}**

# Dien tich da giac



# Tim bao loi cua da giac

const int N **=** 20000**;**

struct Point **{**

long long x**,** y**;**

bool **operator** **<** **(**const Point **&**v**)** const **{** **return** x **==** v**.**x **?** y **<** v**.**y **:** x **<** v**.**x**;** **}**

long long cross**(**const Point **&**p**,** const Point **&**q**)** const **{** **return** **(**p**.**x **-** x**)** **\*** **(**q**.**y **-** y**)** **-** **(**p**.**y **-** y**)** **\*** **(**q**.**x **-** x**);** **}**

**}** p**[**N**],** poly**[**N**];**

int n**;**

void enter**()** **{**

scanf**(**"%d"**,** **&**n**);**

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

scanf**(**"%lld%lld"**,** **&**p**[**i**].**x**,** **&**p**[**i**].**y**);**

**}**

long long size**(**Point poly**[],** int k**)** **{**

long long S **=** **(**poly**[**k**-**1**].**x **-** poly**[**0**].**x**)** **\*** **(**poly**[**k**-**1**].**y **+** poly**[**0**].**y**);**

**for(**int i **=** 1**;** i **<** k**;** **++**i**)**

S **+=** **(**poly**[**i**-**1**].**x **-** poly**[**i**].**x**)** **\*** **(**poly**[**i**-**1**].**y **+** poly**[**i**].**y**);**

**return** S**;**

printf**(**"%lld\n"**,** S**);**

**}**

void solve**()** **{**

sort**(**p**,** p**+**n**);** int k **=** 0**;**

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

**while(**k **>=** 2 **&&** poly**[**k**-**2**].**cross**(**poly**[**k**-**1**],** p**[**i**])** **<=** 0**)** **--**k**;**

poly**[**k**++]** **=** p**[**i**];**

**}**

**for(**int i **=** n**-**2**,** t **=** k**+**1**;** i **>=** 0**;** **--**i**)** **{**

**while(**k **>=** t **&&** poly**[**k**-**2**].**cross**(**poly**[**k**-**1**],** p**[**i**])** **<=** 0**)** **--**k**;**

poly**[**k**++]** **=** p**[**i**];**

**}**

printf**(**"%lld\n"**,** size**(**poly**,** k**));**

**}**

# Euler graph

int n**,**m**,**u**,**v**;**

struct st

**{**

int d**,**c**,**cs**;**

st**(**int a**,**int b**,**int c**):** d**(**a**),**c**(**b**),**cs**(**c**)** **{}**

**};**

vector **<**st**>** canh**[**maxm**];**

bool dd**[**maxm**];**

stack **<**int**>** s**;**

void input**()**

**{**

cin**>>**n**>>**m**;**

**for(**int i**=**1**;**i**<=**m**;**i**++)**

**{**

cin**>>**u**>>**v**;**

canh**[**u**].**push\_back**(**st**(**u**,**v**,**i**));**

canh**[**v**].**push\_back**(**st**(**v**,**u**,**i**));**

**}**

**}**

void solve**()**

**{**

s**.**push**(**1**);**

**while(!**s**.**empty**())**

**{**

int i**=**0**;**

int u**=**s**.**top**();**

**while(**i**<**canh**[**u**].**size**()&&**dd**[**canh**[**u**][**i**].**cs**])**

i**++;**

**if(**i**<**canh**[**u**].**size**())**

**{**

s**.**push**(**canh**[**u**][**i**].**c**);**

dd**[**canh**[**u**][**i**].**cs**]=true;**

**}**

**else**

**{**

cout**<<**u**<<**" "**;**

s**.**pop**();**

**}**

**}**

**}**

# KMP – so khop chuoi

const int N **=** 1000006**;**

int m**,** n**,** Prev**[**N**];**

char a**[**N**],** b**[**N**];**

main**()** **{**

scanf**(**"%s%s"**,** a**+**1**,** b**+**1**);**

m**=**strlen**(**a**+**1**),** n**=**strlen**(**b**+**1**);**

Prev**[**0**]** **=** **-**1**;**

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

Prev**[**i**]=**0**;**

**for** **(**int j**=**Prev**[**i**-**1**];** j**!=-**1**;** j**=**Prev**[**j**])**

**if** **(**b**[**j**+**1**]==**b**[**i**])** **{**

Prev**[**i**]=**j**+**1**;**

**break;**

**}**

**}**

int u**=**0**;**

**for** **(**int i**=**1**;** i**<=**m**;** i**++)** **{**

**while** **(**u**!=**0 **&&** b**[**u**+**1**]!=**a**[**i**])** u**=**Prev**[**u**];**

**if** **(**b**[**u**+**1**]==**a**[**i**])** u**++;**

**if** **(**u**==**n**)** printf**(**"%d "**,** i**-**n**+**1**);**

**}**

puts**(**""**);**

**}**

# Matrix

template **<**class \_T**>**

struct matrix **{**

\_T a**[**SIZE**][**SIZE**];**

int m**,** n**;**

\_T**\*** **operator[]** **(**int i**){** **return** a**[**i**];** **}**

const \_T**\*** **operator[]** **(**int i**)** const **{** **return** a**[**i**];** **}**

void clear**(**int M**,** int N**){** m**=**M**,**n**=**N**;** int i**,** j**;** **for** **(**i**=**1**;** i**<=**m**;** i**++)** **for** **(**j**=**1**;** j**<=**n**;** j**++)** a**[**i**][**j**]=**0**;** **}**

void **operator** **=** **(**const matrix **&**A**);**

matrix **(**int M**,** int N**){** clear**(**M**,** N**);** **}**

**};**

template **<**class \_T**>** matrix**<**\_T**>**

basedMatrix**(**int n**){**

matrix**<**\_T**>** R**(**n**,**n**);**

**for** **(**int i**=**1**;** i**<=**n**;** i**++)** R**[**i**][**i**]=**1**;**

**return** R**;**

**}**

template **<**class \_T**>** matrix**<**\_T**>**

**operator** **\*** **(**const matrix**<**\_T**>** **&**A**,** const matrix**<**\_T**>** **&**B**){**

matrix**<**\_T**>** C**(**A**.**m**,** B**.**n**);**

int i**,** j**,** k**;**

**for** **(**i**=**1**;** i**<=**A**.**m**;** i**++)**

**for** **(**j**=**1**;** j**<=**B**.**n**;** j**++)**

**for** **(**k**=**1**;** k**<=**A**.**n**;** k**++)**

C**[**i**][**j**]** **+=** A**[**i**][**k**]** **\*** B**[**k**][**j**];**

**return** C**;**

**}**

template **<**class \_T**>** matrix**<**\_T**>**

**operator** **^** **(**const matrix**<**\_T**>** **&**A**,** int k**){**

**if** **(**k**==**0**)** **return** basedMatrix**<**\_T**>(**A**.**n**);**

**if** **(**k**==**1**)** **return** A**;**

matrix **<**\_T**>** p **=** A**^(**k**/**2**);**

p**=**p**\***p**;**

**if** **(**k**&**1**)** **return** p**\***A**;**

**else** **return** p**;**

**}**

template **<**class \_T**>** void matrix**<**\_T**>** **::**

**operator** **=** **(**const matrix**<**\_T**>** **&**A**){**

int i**,** j**;**

**for** **(**i**=**1**;** i**<=**A**.**m**;** i**++)**

**for** **(**j**=**1**;** j**<=**A**.**n**;** j**++)**

a**[**i**][**j**]=**A**[**i**][**j**];**

m**=**A**.**m**,** n**=**A**.**n**;**

**}**

matrix **<**long long**>** A**(**2**,**2**);**

main**(){**

A**[**1**][**1**]=**1**;**

A**[**2**][**1**]=**1**;**

A**[**1**][**2**]=**1**;**

A**[**2**][**2**]=**0**;**

A**=**A**^**90**;**

printf**(**"%lld\n"**,** A**[**2**][**2**]);**

**}**

# Khoang cach Ngan nhat giua 2 da giac

const double eps**=**0.0000001**;**

const double oo**=**10001.0**;**

int n1**,**n2**;**

**typedef** pair **<**double**,**double**>** ii**;**

ii a**[**maxn**],**b**[**maxn**];**

double res**;**

void input**()**

**{**

scanf**(**"%d"**,&**n1**);**

**for(**int i**=**1**;**i**<=**n1**;++**i**)**

cin**>>**a**[**i**].**x**>>**a**[**i**].**y**;**

scanf**(**"%d"**,&**n2**);**

**for(**int i**=**1**;**i**<=**n2**;++**i**)**

cin**>>**b**[**i**].**x**>>**b**[**i**].**y**;**

**}**

void init**()**

**{**

a**[**n1**+**1**]=**a**[**1**];**b**[**n2**+**1**]=**b**[**1**];**

res**=**oo**;**

**}**

ii **operator** **+(**ii u**,**ii v**)**

**{**

**return** ii**(**u**.**x**+**v**.**x**,**u**.**y**+**v**.**y**);**

**}**

ii **operator** **-(**ii u**,**ii v**)**

**{**

**return** ii**(**u**.**x**-**v**.**x**,**u**.**y**-**v**.**y**);**

**}**

double **operator** **\*(**ii u**,**ii v**)**

**{**

**return** u**.**x**\***v**.**x**+**u**.**y**\***v**.**y**;**

**}**

double cp**(**ii u**,**ii v**)**

**{**

**return** u**.**x**\***v**.**y**-**u**.**y**\***v**.**x**;**

**}**

ii tuyentinh**(**ii a**,**ii b**,**ii c**)**

**{**

**return** ii**(**cp**(**c**,**b**)/**cp**(**a**,**b**),**cp**(**a**,**c**)/**cp**(**a**,**b**));**

**}**

double giai**(**ii C**,**ii A**,**ii B**)**

**{**

ii D**=**ii**(**C**.**x**+**A**.**y**-**B**.**y**,**C**.**y**+**B**.**x**-**A**.**x**);**

ii r**=**tuyentinh**(**B**-**A**,**C**-**D**,**C**-**A**);**

ii M**=**A**+**ii**((**B**-**A**).**x**\***r**.**x**,(**B**-**A**).**y**\***r**.**x**);**

ii vtMA**=**ii**(**A**.**x**-**M**.**x**,**A**.**y**-**M**.**y**),**vtMB**=**ii**(**B**.**x**-**M**.**x**,**B**.**y**-**M**.**y**);**

**if(**vtMA**\***vtMB**<=**0**)**

**return(**sqrt**((**M**.**x**-**C**.**x**)\*(**M**.**x**-**C**.**x**)+(**M**.**y**-**C**.**y**)\*(**M**.**y**-**C**.**y**)));**

**return** **-**1**;**

**}**

void solve**()**

**{**

**for(**int i**=**1**;**i**<=**n1**;++**i**)**

**for(**int j**=**1**;**j**<=**n2**;++**j**)**

**{**

double g**=**giai**(**a**[**i**],**b**[**j**],**b**[**j**+**1**]);**

**if(**g**==-**1**)**

**continue;**

res**=**mini**(**res**,**g**);**

**}**

**for(**int i**=**1**;**i**<=**n2**;++**i**)**

**for(**int j**=**1**;**j**<=**n1**;++**j**)**

**{**

double g**=**giai**(**b**[**i**],**a**[**j**],**a**[**j**+**1**]);**

**if(**g**==-**1**)**

**continue;**

res**=**mini**(**res**,**g**);**

**}**

**for(**int i**=**1**;**i**<=**n1**;++**i**)**

**for(**int j**=**1**;**j**<=**n2**;++**j**)**

res**=**mini**(**res**,**sqrt**((**b**[**j**].**x**-**a**[**i**].**x**)\*(**b**[**j**].**x**-**a**[**i**].**x**)+(**b**[**j**].**y**-**a**[**i**].**y**)\*(**b**[**j**].**y**-**a**[**i**].**y**)));**

**}**

void output**()**

**{**

cout**<<**setprecision**(**6**)<<**fixed**<<(**res**+**eps**)/**2**<<**endl**;**

**}**

# Check diem nam trong dagiac

const int maxN**=**100001**;**

const double epsilon**=**1e-6**;**

struct point

**{**

double x**,**y**;**

**};**

**typedef** point vec**;**

point p**[**maxN**];**

point A**,**B**;**

int n**;**

bool inside**;**

vec make\_vec**(**double a**,** double b**)**

**{**

vec u**;**

u**.**x**=**a**;** u**.**y**=**b**;**

**return** u**;**

**}**

vec **operator** **-(**vec u**,**vec v**)**

**{**

**return** make\_vec**(**u**.**x**-**v**.**x**,**u**.**y**-**v**.**y**);**

**}**

double **operator** **\*(**vec u**,** vec v**)**

**{**

**return** **(**u**.**x**\***v**.**x**+**u**.**y**\***v**.**y**);**

**}**

double cp**(**vec u**,** vec v**)**

**{**

**return** **(**u**.**x**\***v**.**y**-**u**.**y**\***v**.**x**);**

**}**

bool iszero**(**double a**)**

**{**

**return** **(**abs**(**a**)<=**epsilon**);**

**}**

void Enter**()**

**{**

cin**>>**n**>>**A**.**x**>>**A**.**y**;**

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

cin**>>**p**[**i**].**x**>>**p**[**i**].**y**;**

p**[**n**+**1**]=**p**[**1**];**

**}**

bool onsegment**(**vec u**,** vec v**)**

**{**

**return** **(**iszero**(**cp**(**u**,**v**))&&(**u**\***v**<=**0**));**

**}**

void make\_ray**()**

**{**

bool ok**;**

**do**

**{**

ok**=false;**

srand**(**time**(**0**));**

B**.**x**=**rand**();** B**.**y**=**rand**();**

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

**if** **(**iszero**(**cp**(**B**-**A**,**p**[**i**]-**A**)))**

**{**

ok**=true;**

**break;**

**}**

**}**

**while** **(**ok**);**

**}**

vec solveSLE**(**vec a**,** vec b**,** vec c**)**

**{**

double D**,**Dx**,**Dy**;**

D**=**cp**(**a**,**b**);**Dx**=**cp**(**c**,**b**);**Dy**=**cp**(**a**,**c**);**

**return** make\_vec**(**Dx**/**D**,**Dy**/**D**);**

**}**

bool cut**(**point C**,** point D**)**

**{**

vec r**;**

r**=**solveSLE**(**B**-**A**,**C**-**D**,**C**-**A**);**

**return** **(**r**.**x**>=**0 **&&** 0.0**<=**r**.**y **&&** r**.**y**<=**1.0**);**

**}**

bool check**()**

**{**

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

**if** **(**onsegment**(**p**[**i**]-**A**,**p**[**i**+**1**]-**A**))**

**return** **true;**

make\_ray**();**

bool res**=false;**

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

**if** **(**cut**(**p**[**i**],**p**[**i**+**1**]))**

res**=!**res**;**

**return** res**;**

**}**

int main**()**

**{**

freopen**(**"123.inp"**,**"r"**,**stdin**);**

Enter**();**

inside**=**check**();**

**if** **(**inside**)**

cout**<<**"Inside the polygon.\n"**;**

**else**

cout**<<**"Outside the polygon.\n"**;**

**return** 0**;**

**}**

# Do dai giao diem giua duong thang va da giac

const double eps**=**1e-6**;**

**typedef** pair **<**double**,**double**>** ii**;**

ii p**[**maxn**],**d1**,**d2**,**dn**,**dt**;**

int n**;**

int dem**,**res**;**

double kc**;**

vector **<**ii**>** hr**;**

ii **operator** **+(**ii u**,**ii v**)**

**{**

**return** ii**(**u**.**x**+**v**.**x**,**u**.**y**+**v**.**y**);**

**}**

ii **operator** **-(**ii u**,**ii v**)**

**{**

**return** ii**(**u**.**x**-**v**.**x**,**u**.**y**-**v**.**y**);**

**}**

double **operator** **\*(**ii u**,**ii v**)**

**{**

**return** **(**u**.**x**\***v**.**x**+**u**.**y**\***v**.**y**);**

**}**

double cp**(**ii u**,**ii v**)**

**{**

**return(**u**.**x**\***v**.**y**-**u**.**y**\***v**.**x**);**

**}**

bool iszero**(**double a**)**

**{**

**return** **(**ttd**(**a**)<=**eps**);**

**}**

void input**()**

**{**

scanf**(**"%d"**,&**n**);**

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

cin**>>**p**[**i**].**x**>>**p**[**i**].**y**;**

p**[**n**+**1**]=**p**[**1**];**

cin**>>**d1**.**x**>>**d1**.**y**>>**d2**.**x**>>**d2**.**y**;**

**}**

void init**()**

**{**

srand**(**time**(**0**));**

**}**

bool onsegment**(**ii u**,**ii v**)**

**{**

**return** **(**iszero**(**cp**(**u**,**v**))&&(**u**\***v**<=**0**));**

**}**

ii tuyentinh**(**ii A**,**ii B**,**ii C**)**

**{**

**return** ii**((**cp**(**C**,**B**))/(**cp**(**A**,**B**)),(**cp**(**A**,**C**))/(**cp**(**A**,**B**)));**

**}**

bool check**(**ii diem**)**

**{**

dt**=**diem**;**

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

**if(**onsegment**(**p**[**i**]-**dt**,**p**[**i**+**1**]-**dt**))**

**return** **true;**

bool result**=**0**;**

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

**if(((**p**[**i**].**y**>**dt**.**y**)!=(**p**[**i**+**1**].**y**>**dt**.**y**))&&(**dt**.**x**<(**p**[**i**+**1**].**x**-**p**[**i**].**x**)\*(**dt**.**y**-**p**[**i**].**y**)/(**p**[**i**+**1**].**y**-**p**[**i**].**y**)+**p**[**i**].**x**))**

result**=!**result**;**

**return** result**;**

**}**

bool cut**(**ii A**,**ii B**,**ii C**,**ii D**)**

**{**

bool result**;**

ii r**=**tuyentinh**(**B**-**A**,**C**-**D**,**C**-**A**);**

result**=(**r**.**x**>=**0**&&**0.000000**<=**r**.**y**&&**r**.**y**<=**1.000000**);**

**if(**result**)**

**{**

ii z**=**A**+**ii**((**B**-**A**).**x**\***r**.**x**,(**B**-**A**).**y**\***r**.**x**);**

kc**=**sqrt**((**A**.**x**-**z**.**x**)\*(**A**.**x**-**z**.**x**)+(**A**.**y**-**z**.**y**)\*(**A**.**y**-**z**.**y**));**

**}**

**return** result**;**

**}**

ii giao**(**ii A**,**ii B**,**ii C**,**ii D**)**

**{**

ii r**=**tuyentinh**(**B**-**A**,**C**-**D**,**C**-**A**);**

**if(**isnan**(**r**.**x**)||**isnan**(**r**.**y**))**

**return** ii**(-**1.2345**,**1.2345**);**

**if(**r**.**x**<**0.0**||**r**.**x**>**1.0**||**r**.**y**<**0.0**||**r**.**y**>**1.0**)**

**return** ii**(**1.2345**,-**1.2345**);**

ii M**=**A**+**ii**((**B**-**A**).**x**\***r**.**x**,(**B**-**A**).**y**\***r**.**x**);**

**return** M**;**

**}**

void solve**()**

**{**

bool c1**=**check**(**d1**),**c2**=**check**(**d2**);**

**if(!**c1**&&!**c2**)**

**{**

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

**{**

ii giai**=**giao**(**d1**,**d2**,**p**[**i**],**p**[**i**+**1**]);**

**if(**giai**==**ii**(-**1.2345**,**1.2345**))**

**{**

kc**=**sqrt**((**p**[**i**].**x**-**p**[**i**+**1**].**x**)\*(**p**[**i**].**x**-**p**[**i**+**1**].**x**)+(**p**[**i**].**y**-**p**[**i**+**1**].**y**)\*(**p**[**i**].**y**-**p**[**i**+**1**].**y**));**

res**=**100**\***kc**;**

printf**(**"%d\n"**,**res**);**

**return;**

**}**

**if(**giai**==**ii**(**1.2345**,-**1.2345**))**

**continue;**

hr**.**push\_back**(**giai**);**

**}**

**if(**hr**.**size**()==**0**)**

printf**(**"0\n"**);**

**else**

**{**

**if(**hr**.**size**()==**3**&&**hr**[**0**]==**hr**[**1**])**

hr**[**1**]=**hr**[**2**];**

kc**=**sqrt**((**hr**[**0**].**x**-**hr**[**1**].**x**)\*(**hr**[**0**].**x**-**hr**[**1**].**x**)+(**hr**[**0**].**y**-**hr**[**1**].**y**)\*(**hr**[**0**].**y**-**hr**[**1**].**y**));**

res**=**100**\***kc**;**

printf**(**"%d\n"**,**res**);**

**}**

**}**

**if(**c1**&&**c2**)**

**{**

kc**=**sqrt**((**d1**.**x**-**d2**.**x**)\*(**d1**.**x**-**d2**.**x**)+(**d1**.**y**-**d2**.**y**)\*(**d1**.**y**-**d2**.**y**));**

res**=**100**\***kc**;**

printf**(**"%d\n"**,**res**);**

**}**

**if(**c1**&&!**c2**)**

**{**

dt**=**d1**;**dn**=**d2**;**

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

**if(**cut**(**dt**,**dn**,**p**[**i**],**p**[**i**+**1**]))**

**break;**

res**=**100**\***kc**;**

printf**(**"%d\n"**,**res**);**

**}**

**if(!**c1**&&**c2**)**

**{**

dt**=**d2**;**dn**=**d1**;**

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

**if(**cut**(**dt**,**dn**,**p**[**i**],**p**[**i**+**1**]))**

**break;**

res**=**100**\***kc**;**

printf**(**"%d\n"**,**res**);**

**}**

# Kruskal – Tim cay khung nho nhat

int n**,**m**;**

int lab**[**maxn**];**

struct st

**{**

int u**,**v**,**w**;**

bool dd**;**

**};**

st e**[**maxm**];**

int dem**;**

void input**()**

**{**

cin**>>**n**>>**m**;**

**for(**int i**=**1**;**i**<=**m**;**i**++)**

cin**>>**e**[**i**].**u**>>**e**[**i**].**v**>>**e**[**i**].**w**;**

**}**

int findset**(**int i**)**

**{**

int result**;**

**if(**lab**[**i**]<=**0**)**

result**=**i**;**

**else**

**{**

result**=**findset**(**lab**[**i**]);**

lab**[**i**]=**result**;**

**}**

**return** result**;**

**}**

bool cmp**(**st a**,**st b**)**

**{**

**return** a**.**w**<**b**.**w**;**

**}**

void hop**(**int r**,**int s**)**

**{**

**if(**lab**[**r**]<**lab**[**s**])**

lab**[**s**]=**r**;**

**else**

**{**

**if(**lab**[**r**]==**lab**[**s**])**

lab**[**s**]--;**

lab**[**r**]=**s**;**

**}**

**}**

void solve**()**

**{**

sort**(**e**+**1**,**e**+**m**+**1**,**cmp**);**

**for(**int i**=**1**;**i**<=**m**;**i**++)**

**{**

int r**=**findset**(**e**[**i**].**u**);**

int s**=**findset**(**e**[**i**].**v**);**

**if(**r**!=**s**)**

**{**

e**[**i**].**dd**=true;**

dem**++;**

hop**(**r**,**s**);**

**}**

**}**

**}**

void output**()**

**{**

int s**=**0**;**

**for(**int i**=**1**;**i**<=**m**;**i**++)**

**if(**e**[**i**].**dd**)**

cout**<<**e**[**i**].**u**<<**" "**<<**e**[**i**].**v**<<**endl**;**

**}**

# Floyd – duong di ngan nhat (co trong so am)

main**(){**

int i**,**j**,**k**,** p**,**q**,**w**;**

scanf**(**"%d%d"**,** **&**n**,** **&**m**);**

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

**for** **(**j**=**1**;** j**<=**n**;** j**++)**

a**[**i**][**j**]** **=** oo**;**

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

a**[**i**][**i**]** **=** 0**;**

**for** **(**i**=**1**;** i**<=**m**;** i**++)**

**{**

scanf**(**"%d%d%d"**,** **&**p**,** **&**q**,** **&**w**);**

a**[**p**][**q**]** **=** a**[**q**][**p**]** **=** w**;**

**}**

**for** **(**k**=**1**;** k**<=**n**;** k**++)**

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

**for** **(**j**=**1**;** j**<=**n**;** j**++)**

minimize**(**a**[**i**][**j**],** a**[**i**][**k**]** **+** a**[**k**][**j**]);**

printf**(**"%d\n"**,** a**[**1**][**n**]);**

**}**

# Tim khop va cau

int n**,**m**;**

vector **<**int**>** l**[**maxn**];**

int num**[**maxn**],**cha**[**maxn**],**low**[**maxn**],**numcon**[**maxn**];**

int dem**;**

int res2**;**

set **<**int**>** res1**;**

void input**()**

**{**

scanf**(**"%d%d"**,&**n**,&**m**);**

**for(**int i**=**1**;**i**<=**m**;++**i**)**

**{**

int u**,**v**;**scanf**(**"%d%d"**,&**u**,&**v**);**

l**[**u**].**push\_back**(**v**);**l**[**v**].**push\_back**(**u**);**

**}**

**}**

void init**()**

**{**

**}**

void dfs**(**int u**)**

**{**

**++**dem**;**

num**[**u**]=**dem**;**

low**[**u**]=**oo**;**

**for(**int v**=**0**;**v**<**l**[**u**].**size**();++**v**)**

**if(**l**[**u**][**v**]!=**cha**[**u**])**

**if(**num**[**l**[**u**][**v**]]>**0**)**

low**[**u**]=**mini**(**low**[**u**],**num**[**l**[**u**][**v**]]);**

**else**

**{**

cha**[**l**[**u**][**v**]]=**u**;**

dfs**(**l**[**u**][**v**]);**

low**[**u**]=**mini**(**low**[**u**],**low**[**l**[**u**][**v**]]);**

**}**

**}**

void dfs**()**

**{**

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

**if(**num**[**i**]==**0**)**

**{**

cha**[**i**]=-**1**;**

dfs**(**i**);**

**}**

**}**

void solve**()**

**{**

dfs**();**

cout**<<**"cau: \n"**;**

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

**if(**cha**[**i**]!=-**1**)**

**{**

**++**numcon**[**cha**[**i**]];**

**if(**low**[**i**]>=**num**[**i**])**

cout**<<**i**<<**" "**<<**cha**[**i**]<<**"\n"**;**

**}**

cout**<<**"khop: \n"**;**

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

**if(**cha**[**i**]==-**1**)**

**{**

**if(**numcon**[**i**]>=**2**)**

res1**.**insert**(**i**);**

**}**

**else**

**if(**low**[**i**]>=**num**[**cha**[**i**]]&&**cha**[**cha**[**i**]]!=-**1**)**

res1**.**insert**(**cha**[**i**]);**

**}**

void output**()**

**{**

**for(**set **<**int**>** **::** iterator it**=**res1**.**begin**();**it**!=**res1**.**end**();++**it**)**

cout**<<\***it**<<**"\n"**;**

**}**

# Tarjan – Tim thanh phan lien thong manh

int n**,** m**,** Num**[**N**],** Low**[**N**],** cnt**=**0**;**

vector**<**int**>** a**[**N**];**

stack**<**int**>** st**;**

int Count**=**0**;**

void visit**(**int u**)** **{**

Low**[**u**]=**Num**[**u**]=++**cnt**;**

st**.**push**(**u**);**

**for** **(**int i**=**0**;** int v**=**a**[**u**][**i**];** i**++)**

**if** **(**Num**[**v**])**

Low**[**u**]=**min**(**Low**[**u**],** Num**[**v**]);**

**else** **{**

visit**(**v**);**

Low**[**u**]=**min**(**Low**[**u**],** Low**[**v**]);**

**}**

**if** **(**Num**[**u**]==**Low**[**u**])** **{** // found one

Count**++;**

int v**;**

**do** **{**

v**=**st**.**top**();** st**.**pop**();**

Num**[**v**]=**Low**[**v**]=**oo**;** // remove v from graph

**}** **while** **(**v**!=**u**);**

**}**

**}**

main**(){**

scanf**(**"%d%d"**,** **&**n**,** **&**m**);**

**for** **(**int i**=**1**;** i**<=**m**;** i**++)** **{**

int x**,** y**;**

scanf**(**"%d%d"**,** **&**x**,** **&**y**);**

a**[**x**].**push\_back**(**y**);**

**}**

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

a**[**i**].**push\_back**(**0**);**

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

**if** **(!**Num**[**i**])** visit**(**i**);**

cout **<<** Count **<<** endl**;**

**}**

# Tim day con tang dai nhat

int n**;**

int a**[**maxn**],**b**[**maxn**],**res**;**

vector **<**int**>** ds**;**

void input**()**

**{**

cin**>>**n**;**

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

cin**>>**a**[**i**];**

b**[**0**]=**INT\_MAX**;**

**}**

void solve**()**

**{**

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

**{**

int t**=**res**;**

**while(**a**[**i**]>=**b**[**t**])**

t**--;**

b**[**t**+**1**]=**a**[**i**];**

**if(**res**<**t**+**1**)**

**{**

res**=**t**+**1**;**

ds**.**push\_back**(**a**[**i**]);**

**}**

**}**

**}**

void output**()**

**{**

cout**<<**res**<<**endl**;**

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

cout**<<**ds**[**i**]<<**" "**;**

**}**

# Eratosthene – Xay dung bang so nguyen to

void eratosthene**(**int n**)**

**{**

maxn**=**1000001**;**

int j**;**

bool prime**[**maxn**];**

memset**(**prime**,false,sizeof(**prime**));**

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

**if(!**prime**[**i**])**

**{**

j**=**i**\***i**;**

**while(**j**<=**n**)**

**{**

prime**[**j**]=true;**

j**+=**i**;**

**}**

**}**

**}**

# Kiem tra so nguyen to

bool isprime**(**int n**)**

**{**

int k**,**sqrtn**;**

**if(**n**==**2**||**n**==**3**)**

**return** **true;**

**if(**n**==**1**||**n**%**2**==**0**||**n**%**3**==**0**)**

**return** **false;**

sqrtn**=**sqrt**(**n**);**

k**=-**1**;**

**while(**k**<=**sqrtn**)**

**{**

k**+=**6**;**

**if(**n**%**k**==**0**||**n**%(**k**+**2**)==**0**)**

**break;**

**}**

**return(**k**>**sqrtn**);**

**}**

# Phi ham Euler – den so nho hon n, cung nguyen to voi n

int a**[**maxsize**];**

void factor**(**int n**,**int **&**sz**)**

**{**

int j**=**1**;**

**for(**int i**=**2**;**i**\***i**<=**n**;**i**++)**

**{**

**if(**n**%**i**==**0**)**

**{**

a**[**j**]=**i**;**

j**++;**

**while(**n**%**i**==**0**)**

n**/=**i**;**

**}**

**}**

**if(**n**>**1**)**

**{**

a**[**j**]=**n**;**

j**++;**

**}**

sz**=**j**-**1**;**

**}**

int main**()**

**{**

freopen**(**"123.inp"**,**"r"**,**stdin**);**

int t**,**n**,**k**,**tam**;**

scanf**(**"%d"**,&**t**);**

**for(**int i**=**1**;**i**<=**t**;++**i**)**

**{**

scanf**(**"%d"**,&**n**);**

factor**(**n**,**k**);**

tam**=**1**;**

**for(**int j**=**1**;**j**<=**k**;**j**++)**

**{**

tam**\*=**a**[**j**]-**1**;**

n**/=**a**[**j**];**

**}**

printf**(**"%d\n"**,**tam**\***n**);**

**}**

**}**