Contents

1 Basic

1.1 .vimrc

```
syntax on
se ru nu ai
se ts=4 sts=4 sw=4 st=4 expandtab smarttab
inoremap {<ENTER> {}<LEFT><ENTER><ENTER><UP><TAB>
```

1.2 default code

```cpp
#pragma GCC optimize("Ofast", "no-stack-protector", "
    unroll-loops")
#pragma GCC optimize("no-stack-protector")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,sse4.2,
    popcnt,abm,mmx,avx,tune=native")
#pragma GCC diagnostic ignored "-W"

#include <bits/stdc++.h>
mt19937 rng(0x5EED);
int randint(int lb, int ub)
{ return uniform_int_distribution<int>(lb, ub)(rng); }

#define SECs (clock() / CLOCKS_PER_SEC)

struct KeyHasher {
    size_t operator()(const Key& k) const {
        return k.first + k.second * 100000;
    }
};
typedef unordered_map<Key,int,KeyHasher> map_t;

/*
int __builtin_clz (unsigned int x):
Returns the number of leading 0-bits in x, starting at
    the most significant bit position. If x is 0, the
    result is undefined.

Built-in Function: int __builtin_popcount (unsigned int
    x):
Returns the number of 1-bits in x.
*/

/*increase stack*/

const int size = 256 << 20;
register long rsp asm("rsp");
char *p = (char*)malloc(size) + size, *bak = (char*)rsp
    ;
__asm__("movq %0, %%rsp\n"::"r"(p));
// main
__asm__("movq %0, %%rsp\n"::"r"(bak));

/*
# Pick's theorem
$A=i+\frac{b}{2}-1$

# Laplacian matrix
$L = D - A$

# Extended Catalan number
$\frac{1}{(k-1)n+1} {kn\choose n}$
*/
```

1.3 fasterIO

```cpp
static inline char getRawChar() {
    static char buf[1 << 16], *p = buf, *end = buf;
    if (p == end) {
        if ((end = buf + fread_unlocked(buf, 1, 1 << 16,
            stdin)) == buf) return '\0';
        p = buf;
    }
    return *p++;
}
```

```
while (c = getRawChar() && (unsigned)(c - '0') > 10U) n
    = n * 10 + (c - '0');
```

1.4 rope

```
#include <ext/rope>
using namespace __gnu_cxx;

rope<int> *p[N],*sz[N];   //use merge by size

int pp[N],szz[N];

int ret = p[ver]->at(x);
p[ver]->replace(x,ret);
p[0] = new rope<int>(pp,pp+n+1);
```

1.5 black magic

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/priority_queue.hpp>
#include <ext/rope>
using namespace __gnu_pbds;
using namespace __gnu_cxx;
using namespace std;

__gnu_pbds::priority_queue<int> pq;
__gnu_pbds::priority_queue<int>::point_iterator idx
    [10];
idx[0] = pq.push(1);

typedef tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update> TREE;
TREE name;
*name.find_by_order(0);
name.order_of_key(1);
name.insert(2);
name.delete(3);
name.split(v, b); /// value < v of a split to b
name.join(another TREE);
```

1.6 Lawfung

- Pick's theorem
  $A = i + \frac{b}{2} - 1$

- Laplacian matrix
  $L = D - A$

- Extended Catalan number
  $\frac{1}{(k-1)n+1} \binom{kn}{n}$

2 Data Structure

2.1 disjoint set

```
struct DJS{
  int p[N], sz, rk[N];
  vector<pair<int*,int>> memo;
  vector<size_t> stk;
  void save(){
    stk.push_back(memo.size());
  }
  void undo(){
    while(memo.size() > stk.back()){
      *memo.back().first = memo.back().second;
      memo.pop_back();
    }
    stk.pop_back();
  }
  void assign(int *x, int v){
    memo.push_back({x, *x});
    *x=v;
```

```
  }
  void init(int n){
    for(int i=1; i<=n; i++){
      p[i]=i; rk[i]=0;
    }
    sz=n; memo.clear(); stk.clear();
  }
  int f(int x){
    return x == p[x] ? x : f(p[x]);
  }
  void uni(int a, int b){
    int aa=f(a); int bb=f(b);
    if(aa == bb) return;
    assign(&sz, sz-1);
    if(rk[aa] > rk[bb]) swap(aa, bb);
    assign(&p[aa], bb);
    assign(&rk[bb], max(rk[bb], rk[aa]+1));
  }
} djs;
```

2.2 Persistent treap

```
#include <bits/stdc++.h>
using namespace std;

const int MAX_N = 1e5 + 6;
const int MAX_M = 1e6 + 6;
const int MAX_P = 3e7;

int myRnd() {
    return 10000*(rand()%10000) + (rand()%10000);
}

struct Treap {
    static Treap mem[MAX_P];
    Treap *lc,*rc;
    char c; int sz;
    Treap(){}
    Treap(char _c) : lc(NULL),rc(NULL),sz(1),c(_c){}
} Treap::mem[MAX_P], *ptr=Treap::mem ;

int Sz(Treap* t) {
    return t?t->sz:0;
}
void pull(Treap* t) {
    if (!t) return;
    t->sz = Sz(t->lc) + Sz(t->rc) + 1;
}

Treap* merge(Treap* a,Treap* b) {
    if (!a || !b) return a?a:b;
    Treap* ret;
    if (myRnd() % (Sz(a) + Sz(b)) < Sz(a)) {
        ret = new (ptr++) Treap(*a);
        ret->rc = merge(a->rc,b);
    }
    else {
        ret = new(ptr++) Treap(*b);
        ret->lc=merge(a,b->lc);
    }
    pull(ret);
    return ret;
}

void split(Treap* t,int k,Treap* &a,Treap* &b) {
    if (!t) a=b=NULL;
    else if (Sz(t->lc) + 1 <= k) {
        a = new(ptr++) Treap(*t);
        split(t->rc,k-Sz(t->lc)-1,a->rc,b);
        pull(a);
    }
    else {
        b=new(ptr++) Treap(*t);
        split(t->lc,k,a,b->lc);
        pull(b);
    }
}

int d;
char buf[MAX_M];
```

```cpp
Treap* ver[MAX_N];

ptr = Treap::mem;
v_cnt++;
ver[v_cnt] = ver[v_cnt-1];
split(ver[v_cnt],p,tl,tr);
tl = merge(tl,new(ptr++)Treap(buf[j]));
```

2.3 Link Cut Tree

```cpp
struct SplayNode {
  static SplayNode HOLE;
  SplayNode *ch[2], *par;
  bool rev;
  SplayNode(): par(&HOLE), rev(false) { ch[0] = ch[1] =
      &HOLE; }
  bool isRoot() {
    return (par->ch[0] != this && par->ch[1] != this);
  }
  void push() {
    if (rev) {
      if (ch[0]) ch[0]->rev ^= 1;
      if (ch[1]) ch[1]->rev ^= 1;
      swap(ch[0], ch[1]);
      rev ^= 1;
    }
  }
  void pushFromRoot() {
    if (!isRoot()) par->pushFromRoot();
    push();
  }
  void pull() {
    if (ch[0]) ch[0]->d = d + ch[0]->parLen;
    if (ch[1]) ch[1]->d = d + ch[1]->parLen;
  }
  void rotate() {
    SplayNode *p = par, *gp = p->par;
    bool dir = (p->ch[1] == this);
    par = gp;
    if (!p->isRoot()) gp->ch[gp->ch[1] == p] = this;
    p->ch[dir] = ch[dir ^ 1];
    p->ch[dir]->par = p;
    p->par = this;
    ch[dir ^ 1] = p;
    p->pull(), pull();
  }
  void splay() {
    pushFromRoot();
    while (!isRoot()) {
      if (!par->isRoot()) {
        SplayNode *gp = par->par;
        if ((gp->ch[0] == par) == (par->ch[0] == this))
          rotate();
        else par->rotate();
      }
      rotate();
    }
  }
} SplayNode::HOLE;
namespace LCT {
  SplayNode *access(SplayNode *x) {
    SplayNode *last = &SplayNode::HOLE;
    while (x != &SplayNode::HOLE) {
      x->splay();
      x->ch[1] = last;
      x->pull();
      last = x;
      x = x->par;
    }
    return last;
  }
  void makeRoot(SplayNode *x) {
    access(x);
    x->splay();
    x->rev ^= 1;
  }
  void link(SplayNode *x, SplayNode *y) {
    makeRoot(x);
    x->par = y;
  }
```

```cpp
  void cut(SplayNode *x, SplayNode *y) {
    makeRoot(x);
    access(y);
    y->splay();
    y->ch[0] = &SplayNode::HOLE;
    x->par = &SplayNode::HOLE;
  }
  void cutParent(SplayNode *x) {
    access(x);
    x->splay();
    x->ch[0]->par = &SplayNode::HOLE;
    x->ch[0] = &SplayNode::HOLE;
  }
  SplayNode *findRoot(SplayNode *x) {
    x = access(x);
    while (x->ch[0] != &SplayNode::HOLE) x = x->ch[0];
    x->splay();
    return x;
  }
  SplayNode *query(SplayNode *x, SplayNode *y) {
    makeRoot(x);
    return access(y);
  }
  SplayNode *queryLca(SplayNode *x, SplayNode *y) {
    access(x);
    auto lca = access(y);
    x->splay();
    return lca->data + lca->ch[1]->sum + (x == lca ? 0
        : x->sum);
  }
  void modify(SplayNode *x, int data) {
    x->splay();
    x->data = data;
    x->pull();
  }
}
```

3 Flow

3.1 Flow with lower bound

```cpp
#include <bits/stdc++.h>
using namespace std;

#define SZ(x) ((int)(x).size())

struct Flow {
    static const int N = 8006;
    struct Edge {
        int to,cap,rev;
        Edge(int _to,int _cap,int _rev):to(_to),cap(
            _cap),rev(_rev){}
    };
    vector<Edge> G[N];
    int d[N];
    int S,T,s,t;
    int n;
    int nows,nowt;
    void init(int _n,int _s,int _t) {
        //vertex are numbered from 0 to n, and s and t
            the source/sink in the original graph
        S = _n+1,T= _n+2;
        s = _s,t = _t;
        n = _n;
        for (int i=0;n+3>=i;i++){
            G[i].clear();
            d[i] = 0;
        }
    }
    void add_edge(int from,int to,int low,int upp) {
        G[from].push_back(Edge(to,upp-low,SZ(G[to])));
        G[to].push_back(Edge(from,0,SZ(G[from])-1));
        d[from] -= low;
        d[to] += low;
    }
    void add_edge(int from,int to,int cap) {
        G[from].push_back(Edge(to,cap,SZ(G[to])));
        G[to].push_back(Edge(from,0,SZ(G[from])-1));
```

```
    }
    int iter[N],level[N];
    void BFS() {
        memset(level,-1,sizeof(level)); level[nows] =
            1;
        queue<int> que; que.push(nows);
        while (!que.empty()) {
            int t=que.front(); que.pop();
            for (Edge e:G[t]) {
                if (e.cap > 0 && level[e.to] == -1) {
                    level[e.to] = level[t]+1;
                    que.push(e.to);
                }
            }
        }
    }
    int dfs(int now,int flow) {
        if (now == nowt) return flow;
        for (int &i=iter[now];SZ(G[now])>i;i++) {
            Edge &e = G[now][i];
            if (e.cap > 0 && level[e.to] == level[now
                ]+1) {
                int ret = dfs(e.to,min(flow,e.cap));
                if (ret > 0) {
                    e.cap -= ret; G[e.to][e.rev].cap +=
                        ret;
                    return ret;
                }
            }
        }
        return 0;
    }
    int flow() {
        int ret = 0;
        while (true) {
            BFS();
            if (level[nowt] == -1) break;
            memset(iter,0,sizeof(iter));
            int tmp;
            while ((tmp = dfs(nows,1000000007)) > 0) {
                ret += tmp;
            }
        }
        return ret;
    }
    int get_ans() {
        nows = S,nowt = T;
        int base=0;
        for (int i=0;n>=i;i++) {
            if (d[i] > 0) base += d[i];
            if (d[i] > 0) add_edge(S,i,d[i]);
            if (d[i] < 0) add_edge(i,T,-d[i]);
        }
        add_edge(t,s,0,1000000007);
        if (flow() != base) return -1; //invalid flow
        nows = s,nowt = t;
        return flow();
    }
} flow;
```

3.2 Global Min Cut

```
struct SW {
    //find global min cut in O(V^3)
    //points are ZERO-BASE!!!
    static const int N = 506;
    int adj[N][N],wei[N],n;
    bool vis[N],del[N];
    void init(int _n) {
        n = _n;
        memset(adj,0,sizeof(adj));
        memset(del,0,sizeof(del));
    }
    void add_edge(int x,int y,int w) {
        adj[x][y] += w;
        adj[y][x] += w;
    }
    void search(int &s,int &t) {
        memset(wei,0,sizeof(wei));
        memset(vis,0,sizeof(vis));
```

```
        s = t = -1;
        while (true) {
            int mx=-1, mx_id=0;
            for (int i=0;i<n;++i) {
                if (!del[i] && !vis[i] && mx<wei[i]) {
                    mx_id = i;
                    mx = wei[i];
                }
            }
            if (mx == -1) break;
            vis[mx_id] = true;
            s = t;
            t = mx_id;
            for (int i=0;i<n;++i) {
                if (!vis[i] && !del[i]) {
                    wei[i] += adj[mx_id][i];
                }
            }
        }
    }
    int solve() {
        int ret = 2147483647; //INF
        for (int i=0;i<n-1;++i) {
            int x,y;
            search(x,y);
            ret = min(ret,wei[y]);
            del[y] = true;
            for (int i=0;i<n;++i) {
                adj[x][i] += adj[y][i];
                adj[i][x] += adj[y][i];
            }
        }
        return ret;
    }
} SW;
```

3.3 Gomory Hu Tree

```
def cut(G,s,t) :
  return minimum s-t cut in G

def gomory_hu(G):
  T = {}
  P = [1] * |V(G)|
  for s in [2,n] :
    t = p[s]
    C = cut(G,s,t)
    add(s,t,w(C)) to c
    for i in [s+1,n] :
      if p[i] == t and s-i path exists in G\C :
        p[i] = s
  return T;
```

4 Geometry

4.1 Circle

```
#include <bits/stdc++.h>
using namespace std;

typedef double D;  //maybe long double
typedef pair<D,D> pdd;
const D eps = 1e-9;

struct Circle {
    D x,y,r; pdd cen;
    Circle(){}
    Circle(D _x,D _y,D _r) : x(_x),y(_y),r(_r),cen(
        make_pair(_x,_y)){}
};

struct Cir_inter_type {
    int type;
    vector<pdd> pts;
    Cir_inter_type(){}
    Cir_inter_type(int _type,vector<pdd> _pts) : type(
        _type),pts(_pts){}
```

```cpp
};
D get_dis(pdd a,pdd b) {
    return sqrt(pow(a.F-b.F,2) + pow(a.S-b.S,2));
}
bool eq(D a,D b) {
    return fabs(a-b) <= eps;
}
bool les(D a,D b) {
    return !eq(a,b) && a<b;
}
bool leq(D a,D b) {
    return les(a,b) || eq(a,b);
}
Cir_inter_type circle_inter(Circle a, Circle b) {
    Cir_inter_type ret;
    D dis=get_dis(a.cen,b.cen);
    if (eq(a.r+b.r,dis)) {
        //outside cut --> type 1
        D x=a.x + (b.x-a.x)*(a.r)/(a.r + b.r);
        D y=a.y + (b.y-a.y)*(a.r)/(a.r + b.r);
        vector<pdd> pts;
        pts.push_back(make_pair(x,y));
        ret = Cir_inter_type(1,pts);
    }
    else if (eq( max(a.r,b.r),min(a.r,b.r) + dis )) {
        //inside cut --> type 2
        if (a.r < b.r) swap(a,b);
        D x=a.x + (b.x - a.x)*a.r/(a.r - b.r);
        D y=a.y + (b.y - a.y)*a.r/(a.r - b.r);
        vector<pdd> pts;
        pts.push_back(make_pair(x,y));
        ret = Cir_inter_type(2,pts);
    }
    else if (les(a.r+b.r,dis)) {
        //no intersection --> type 3
        vector<pdd> pts;
        ret = Cir_inter_type(3,pts);
    }
    else if (les(min(a.r,b.r)+dis,max(a.r,b.r))) {
        //fully inside
        //if a is fully contain b, return type 4
        //if b is fully contain a, return type 5
        vector<pdd> pts;
        if (les(b.r,a.r)) {
            ret = Cir_inter_type(4,pts);
        }
        else if (les(a.r,b.r)) {
            ret = Cir_inter_type(5,pts);
        }
    }
    else if (les(dis,a.r + b.r)) {
        //two intersections --> type 6
        D c1=2*(b.x-a.x)*b.r;
        D c2=2*(b.y-a.y)*b.r;
        D c3=a.r*a.r-b.r*b.r-(b.x-a.x)*(b.x-a.x)-(b.y-a
            .y)*(b.y-a.y);
        D aa = c1*c1 + c2*c2;
        D bb = -2*c2*c3;
        D cc = c3*c3 - c1*c1;
        D sin1 = (-bb + sqrtl(bb*bb-4*aa*cc))/(2*aa);
        D sin2 = (-bb - sqrtl(bb*bb-4*aa*cc))/(2*aa);
        D aaa = c1*c1 + c2*c2;
        D bbb = -2*c1*c3;
        D ccc = c3*c3 - c2*c2;
        D cos1=(-bbb+sqrtl(bbb*bbb-4*aaa*ccc))/(aaa*2);
        D cos2=(-bbb-sqrtl(bbb*bbb-4*aaa*ccc))/(aaa*2);
        vector<pdd> pts;
        if (eq(sin1*sin1 + cos1*cos1,1.)) {
            pts.push_back(make_pair(b.x + b.r*cos1,b.y
                + b.r*(sin1)));
            pts.push_back(make_pair(b.x + b.r*cos2,b.y
                + b.r*(sin2)));
        }
        else {
            pts.push_back(make_pair(b.x + b.r*cos1,b.y
                + b.r*(sin2)));
            pts.push_back(make_pair(b.x + b.r*cos2,b.y
                + b.r*(sin1)));
        }
        ret = Cir_inter_type(6,pts);
    }
    return ret;
}
```

4.2 Half Plane Intersection

```cpp
Pt interPnt( Line l1, Line l2, bool &res ){
    Pt p1, p2, q1, q2;
    tie(p1, p2) = l1; tie(q1, q2) = l2;
    double f1 = (p2 - p1) ^ (q1 - p1);
    double f2 = (p2 - p1) ^ (p1 - q2);
    double f = (f1 + f2);
    if( fabs(f) < eps){ res=0; return {0, 0}; }
    res = true;
    return q1 * (f2 / f) + q2 * (f1 / f);
}
bool isin( Line l0, Line l1, Line l2 ){
    // Check inter(l1, l2) in l0
    bool res; Pt p = interPnt(l1, l2, res);
    return ( (l0.SE - l0.FI) ^ (p - l0.FI) ) > eps;
}
/* If no solution, check: 1. ret.size() < 3
 * Or more precisely, 2. interPnt(ret[0], ret[1])
 * in all the lines. (use (l.S - l.F) ^ (p - l.F) > 0
 */
/* --^-- Line.FI --^-- Line.SE --^-- */
vector<Line> halfPlaneInter( vector<Line> lines ){
    int sz = lines.size();
    vector<double> ata(sz), ord(sz);
    for( int i=0; i<sz; i++) {
        ord[i] = i;
        Pt d = lines[i].SE - lines[i].FI;
        ata[i] = atan2(d.Y, d.X);
    }
    sort( ord.begin(), ord.end(), [&](int i, int j) {
        if( fabs(ata[i] - ata[j]) < eps )
            return ( (lines[i].SE - lines[i].FI) ^
                     (lines[j].SE - lines[i].FI) ) < 0;
        return ata[i] < ata[j];
    });
    vector<Line> fin;
    for (int i=0; i<sz; i++)
        if (!i or fabs(ata[ord[i]] - ata[ord[i-1]]) > eps)
            fin.PB(lines[ord[i]]);
    deque<Line> dq;
    for (int i=0; i<(int)(fin.size()); i++) {
        while((int)(dq.size()) >= 2 and
            not isin(fin[i], dq[(int)(dq.size())-2],
                              dq[(int)(dq.size())-1]))
            dq.pop_back();
        while((int)(dq.size()) >= 2 and
            not isin(fin[i], dq[0], dq[1]))
            dq.pop_front();
        dq.push_back(fin[i]);
    }
    while( (int)(dq.size()) >= 3 and
        not isin(dq[0], dq[(int)(dq.size())-2],
                        dq[(int)(dq.size())-1]))
        dq.pop_back();
    while( (int)(dq.size()) >= 3 and
        not isin(dq[(int)(dq.size())-1], dq[0], dq[1]))
        dq.pop_front();
    vector<Line> res(dq.begin(),dq.end());
    return res;
}
```

4.3 Poly Union

```cpp
#define eps 1e-8
class PY{ public:
    int n;
    Pt pt[5];
    Pt& operator[](const int x){ return pt[x]; }
    void input(){
        int i; n=4;
        for(i=0;i<n;i++) scanf("%lf%lf",&pt[i].x,&pt[i].y);
    }
    double getArea(){
        int i; double s=pt[n-1]^pt[0];
        for(i=0;i<n-1;i++) s+=pt[i]^pt[i+1];
```

```cpp
        return s/2;
    }
};
PY py[500];
pair<double,int> c[5000];
inline double segP(Pt &p,Pt &p1,Pt &p2){
    if(SG(p1.x-p2.x)==0) return (p.y-p1.y)/(p2.y-p1.y);
    return (p.x-p1.x)/(p2.x-p1.x);
}
double polyUnion(int n){
    int i,j,ii,jj,ta,tb,r,d;
    double z,w,s,sum,tc,td;
    for(i=0;i<n;i++) py[i][py[i].n]=py[i][0];
    sum=0;
    for(i=0;i<n;i++){
        for(ii=0;ii<py[i].n;ii++){
            r=0;
            c[r++]=make_pair(0.0,0);
            c[r++]=make_pair(1.0,0);
            for(j=0;j<n;j++){
                if(i==j) continue;
                for(jj=0;jj<py[j].n;jj++){
                    ta=SG(tri(py[i][ii],py[i][ii+1],py[j][jj]));
                    tb=SG(tri(py[i][ii],py[i][ii+1],py[j][jj+1]))
                        ;
                    if(ta==0 && tb==0){
                        if((py[j][jj+1]-py[j][jj])*(py[i][ii+1]-py[
                            i][ii])>0 && j<i){
                            c[r++]=make_pair(segP(py[j][jj],py[i][ii
                                ],py[i][ii+1]),1);
                            c[r++]=make_pair(segP(py[j][jj+1],py[i][
                                ii],py[i][ii+1]),-1);
                        }
                    }else if(ta>=0 && tb<0){
                        tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
                        td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
                        c[r++]=make_pair(tc/(tc-td),1);
                    }else if(ta<0 && tb>=0){
                        tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
                        td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
                        c[r++]=make_pair(tc/(tc-td),-1);
                    }
                }
            }
            sort(c,c+r);
            z=min(max(c[0].first,0.0),1.0);
            d=c[0].second; s=0;
            for(j=1;j<r;j++){
                w=min(max(c[j].first,0.0),1.0);
                if(!d) s+=w-z;
                d+=c[j].second; z=w;
            }
            sum+=(py[i][ii]^py[i][ii+1])*s;
        }
    }
    return sum/2;
}
int main(){
    int n,i,j,k;
    double sum,ds;
    scanf("%d",&n); sum=0;
    for(i=0;i<n;i++){
        py[i].input();
        ds=py[i].getArea();
        if(ds<0){
            for(j=0,k=py[i].n-1;j<k;j++,k--) swap(py[i][j],
                py[i][k]);
            ds=-ds;
        } sum+=ds;
    } printf("%.9f\n",sum/polyUnion(n));
}
```

4.4 Convex Hull 3D

```cpp
#define SIZE(X) (int(X.size()))
#define PI 3.14159265358979323846264338327950288
struct Pt{
    Pt cross(const Pt &p) const
    { return Pt(y * p.z - z * p.y, z * p.x - x * p.z, x *
        p.y - y * p.x); }
```

```cpp
} info[N];
int mark[N][N],n, cnt;;
double mix(const Pt &a, const Pt &b, const Pt &c)
{ return a * (b ^ c); }
double area(int a, int b, int c)
{ return norm((info[b] - info[a]) ^ (info[c] - info[a])
    ); }
double volume(int a, int b, int c, int d)
{ return mix(info[b] - info[a], info[c] - info[a], info
    [d] - info[a]); }
struct Face{
    int a, b, c; Face(){}
    Face(int a, int b, int c): a(a), b(b), c(c) {}
    int &operator [](int k)
    { if (k == 0) return a; if (k == 1) return b; return
        c; }
};
vector<Face> face;
void insert(int a, int b, int c)
{ face.push_back(Face(a, b, c)); }
void add(int v) {
    vector <Face> tmp; int a, b, c; cnt++;
    for (int i = 0; i < SIZE(face); i++) {
        a = face[i][0]; b = face[i][1]; c = face[i][2];
        if(Sign(volume(v, a, b, c)) < 0)
        mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] =
            mark[c][a] = mark[a][c] = cnt;
        else tmp.push_back(face[i]);
    } face = tmp;
    for (int i = 0; i < SIZE(tmp); i++) {
        a = face[i][0]; b = face[i][1]; c = face[i][2];
        if (mark[a][b] == cnt) insert(b, a, v);
        if (mark[b][c] == cnt) insert(c, b, v);
        if (mark[c][a] == cnt) insert(a, c, v);
}}
int Find(){
    for (int i = 2; i < n; i++) {
        Pt ndir = (info[0] - info[i]) ^ (info[1] - info[i])
            ;
        if (ndir == Pt()) continue; swap(info[i], info[2]);
        for (int j = i + 1; j < n; j++) if (Sign(volume(0,
            1, 2, j)) != 0) {
            swap(info[j], info[3]); insert(0, 1, 2); insert
                (0, 2, 1); return 1;
} } return 0; }
int main() {
    for (; scanf("%d", &n) == 1; ) {
        for (int i = 0; i < n; i++) info[i].Input();
        sort(info, info + n); n = unique(info, info + n) -
            info;
        face.clear(); random_shuffle(info, info + n);
        if (Find()) { memset(mark, 0, sizeof(mark)); cnt =
            0;
            for (int i = 3; i < n; i++) add(i); vector<Pt>
                Ndir;
            for (int i = 0; i < SIZE(face); ++i) {
                Pt p = (info[face[i][0]] - info[face[i][1]]) ^
                    (info[face[i][2]] - info[face[i][1]]);
                p = p / norm( p ); Ndir.push_back(p);
            } sort(Ndir.begin(), Ndir.end());
            int ans = unique(Ndir.begin(), Ndir.end()) - Ndir
                .begin();
            printf("%d\n", ans);
        } else printf("1\n");
} }
double calcDist(const Pt &p, int a, int b, int c)
{ return fabs(mix(info[a] - p, info[b] - p, info[c] - p
    ) / area(a, b, c)); }
//compute the minimal distance of center of any faces
double findDist() { //compute center of mass
    double totalWeight = 0; Pt center(.0, .0, .0);
    Pt first = info[face[0][0]];
    for (int i = 0; i < SIZE(face); ++i) {
        Pt p = (info[face[i][0]]+info[face[i][1]]+info[face
            [i][2]]+first)*.25;
        double weight = mix(info[face[i][0]] - first, info[
            face[i][1]]
            - first, info[face[i][2]] - first);
        totalWeight += weight; center = center + p * weight
            ;
    } center = center / totalWeight;
    double res = 1e100; //compute distance
```

```
  for (int i = 0; i < SIZE(face); ++i)
    res = min(res, calcDist(center, face[i][0], face[i
        ][1], face[i][2]));
return res; }
```

4.5 Convex Hull

```
/* Given a convexhull, answer querys in O(\lg N)
CH should not contain identical points, the area should
be > 0, min pair(x, y) should be listed first */
double det( const Pt& p1 , const Pt& p2 )
{ return p1.X * p2.Y - p1.Y * p2.X; }
struct Conv{
  int n;
  vector<Pt> a;
  vector<Pt> upper, lower;
  Conv(vector<Pt> _a) : a(_a){
    n = a.size();
    int ptr = 0;
    for(int i=1; i<n; ++i) if (a[ptr] < a[i]) ptr = i;
    for(int i=0; i<=ptr; ++i) lower.push_back(a[i]);
    for(int i=ptr; i<n; ++i) upper.push_back(a[i]);
    upper.push_back(a[0]);
  }
  int sign( LL x ){ // fixed when changed to double
    return x < 0 ? -1 : x > 0; }
  pair<LL,int> get_tang(vector<Pt> &conv, Pt vec){
    int l = 0, r = (int)conv.size() - 2;
    for( ; l + 1 < r; ){
      int mid = (l + r) / 2;
      if(sign(det(conv[mid+1]-conv[mid],vec))>0)r=mid;
      else l = mid;
    }
    return max(make_pair(det(vec, conv[r]), r),
            make_pair(det(vec, conv[0]), 0));
  }
  void upd_tang(const Pt &p, int id, int &i0, int &i1){
    if(det(a[i0] - p, a[id] - p) > 0) i0 = id;
    if(det(a[i1] - p, a[id] - p) < 0) i1 = id;
  }
  void bi_search(int l, int r, Pt p, int &i0, int &i1){
    if(l == r) return;
    upd_tang(p, l % n, i0, i1);
    int sl=sign(det(a[l % n] - p, a[(l + 1) % n] - p));
    for( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      int smid=sign(det(a[mid%n]-p, a[(mid+1)%n]-p));
      if (smid == sl) l = mid;
      else r = mid;
    }
    upd_tang(p, r % n, i0, i1);
  }
  int bi_search(Pt u, Pt v, int l, int r) {
    int sl = sign(det(v - u, a[l % n] - u));
    for( ; l + 1 < r; ) {
      int mid = (l + r) / 2;
      int smid = sign(det(v - u, a[mid % n] - u));
      if (smid == sl) l = mid;
      else r = mid;
    }
    return l % n;
  }
  // 1. whether a given point is inside the CH
  bool contain(Pt p) {
    if (p.X < lower[0].X || p.X > lower.back().X)
      return 0;
    int id = lower_bound(lower.begin(), lower.end(), Pt
        (p.X, -INF)) - lower.begin();
    if (lower[id].X == p.X) {
      if (lower[id].Y > p.Y) return 0;
    }else if(det(lower[id-1]-p,lower[id]-p)<0)return 0;
    id = lower_bound(upper.begin(), upper.end(), Pt(p.X
        , INF), greater<Pt>()) - upper.begin();
    if (upper[id].X == p.X) {
      if (upper[id].Y < p.Y) return 0;
    }else if(det(upper[id-1]-p,upper[id]-p)<0)return 0;
    return 1;
  }
  // 2. Find 2 tang pts on CH of a given outside point
```

```
  // return true with i0, i1 as index of tangent points
  // return false if inside CH
  bool get_tang(Pt p, int &i0, int &i1) {
    if (contain(p)) return false;
    i0 = i1 = 0;
    int id = lower_bound(lower.begin(), lower.end(), p)
        - lower.begin();
    bi_search(0, id, p, i0, i1);
    bi_search(id, (int)lower.size(), p, i0, i1);
    id = lower_bound(upper.begin(), upper.end(), p,
        greater<Pt>()) - upper.begin();
    bi_search((int)lower.size() - 1, (int)lower.size()
        - 1 + id, p, i0, i1);
    bi_search((int)lower.size() - 1 + id, (int)lower.
        size() - 1 + (int)upper.size(), p, i0, i1);
    return true;
  }
  // 3. Find tangent points of a given vector
  // ret the idx of vertex has max cross value with vec
  int get_tang(Pt vec){
    pair<LL, int> ret = get_tang(upper, vec);
    ret.second = (ret.second+(int)lower.size()-1)%n;
    ret = max(ret, get_tang(lower, vec));
    return ret.second;
  }
  // 4. Find intersection point of a given line
  // return 1 and intersection is on edge (i, next(i))
  // return 0 if no strictly intersection
  bool get_intersection(Pt u, Pt v, int &i0, int &i1){
    int p0 = get_tang(u - v), p1 = get_tang(v - u);
    if(sign(det(v-u,a[p0]-u))*sign(det(v-u,a[p1]-u))<0){
      if (p0 > p1) swap(p0, p1);
      i0 = bi_search(u, v, p0, p1);
      i1 = bi_search(u, v, p1, p0 + n);
      return 1;
    }
    return 0;
  }
};
```

5 Graph

5.1 Biconnected Component

```
#include <bits/stdc++.h>
using namespace std;
const int N = 800006;

int low[N],dfn[N];
bool vis[N];
int cnt[N];
int e[N],x[N],y[N];
int stamp;

vector<int> G[N];
vector<int> bcc[N];
int bcc_no = 0;
stack<int> sta;

void dfs(int now,int par) {
    vis[now] = true;
    dfn[now] = low[now] = (++stamp);
    for (int i:G[now]) {
        int to=(e[i]^now);
        if (to == par) continue;
        if (!vis[to]) {
            sta.push(i); dfs(to,now);
            low[now] = min(low[now],low[to]);
            if (low[to] >= dfn[now]) {
                ++bcc_no; int p;
                do {
                    p = sta.top(); sta.pop();
                    bcc[bcc_no].push_back(p);
                } while (p != i);
            }
        }
        else if (dfn[to] < dfn[now]) {
            sta.push(i);
```

```
                low[now] = min(low[now],dfn[to]);
            }
        }
    }
}
```

5.2 general graph macthing

```
const int N = 514, E = (2e5) * 2;
struct Graph{
  int to[E],bro[E],head[N],e;
  int lnk[N],vis[N],stp,n;
  void init( int _n ){
    stp = 0; e = 1; n = _n;
    for( int i = 1 ; i <= n ; i ++ )
      lnk[i] = vis[i] = 0;
  }
  void add_edge(int u,int v){
    to[e]=v,bro[e]=head[u],head[u]=e++;
    to[e]=u,bro[e]=head[v],head[v]=e++;
  }
  bool dfs(int x){
    vis[x]=stp;
    for(int i=head[x];i;i=bro[i]){
      int v=to[i];
      if(!lnk[v]){
        lnk[x]=v,lnk[v]=x;
        return true;
      }else if(vis[lnk[v]]<stp){
        int w=lnk[v];
        lnk[x]=v,lnk[v]=x,lnk[w]=0;
        if(dfs(w)){
          return true;
        }
        lnk[w]=v,lnk[v]=w,lnk[x]=0;
      }
    }
    return false;
  }
  int solve(){
    int ans = 0;
    for(int i=1;i<=n;i++)
      if(!lnk[i]){
        stp++; ans += dfs(i);
      }
    return ans;
  }
} graph;
```

5.3 Tutte matrix

```
## Graph Matching (tutte) ##
#define MAX 400
#define P 1000000007
typedef long long i64;
int mat[MAX][MAX];
i64 tutte[MAX][MAX];
inline int randInt(int n) {
  return ((rand() << 15) ^ rand()) % n;
}
int matRank(i64 a[MAX][MAX], int n, i64 p) {
  int i, j, k, cnt = 0, cur;
  i64 t;
  for (i = 0; i < n; ++i) {
    for (j = i + 1; j < n; ++j) {
      while (a[j][i]) {
        for (t = a[i][i] / a[j][i], k = 0; k < n; ++k)
          {
          a[i][k] = (a[i][k] - a[j][k] * t) % p;
          swap(a[i][k], a[j][k]);
        }
      }
    }
    for (cur = 0, j = i; j < n; ++j) {
      if (a[i][j]) { cur = 1; }
    }
    cnt += cur;
  }
  return cnt;
```

```
}
int maxMatch(const int mat[MAX][MAX], int n) {
  int i, j;
  memset(tutte, 0, sizeof(tutte));
  for (i = 0; i < n; ++i) {
    for (j = i + 1; j < n; ++j) {
      if (mat[i][j]) { tutte[j][i] = -(tutte[i][j] =
          randInt(P)); }
    }
  }
  return matRank(tutte, n, P) >> 1;
}
```

5.4 KM

```
int n , w[MAX][MAX] , lx[MAX] , ly[MAX] , slk[MAX];
int s[MAX] , t[MAX] , good[MAX];
int match(int now){
    s[now] = 1;
    REP(to , 1 , n + 1){
        if(t[to]) continue;
        if(lx[now] + ly[to] == w[now][to]){
            t[to] = 1;
            if(good[to] == 0 || match(good[to]))
                return good[to] = now , 1;
        }
        else slk[to] = min(slk[to] , lx[now] + ly[to] -
            w[now][to]);
    }
    return 0;
}
int update(){
    int val = INF;
    REP(i , 1 , n + 1) if(t[i] == 0) val = min(val ,
        slk[i]);
    REP(i , 1 , n + 1){
        if(s[i]) lx[i] -= val;
        if(t[i]) ly[i] += val;
    }
}
void solve(){
    REP(i , 1 , n + 1) REP(j , 1 , n + 1)
        lx[i] = max(lx[i] , w[i][j]);
    REP(i , 1 , n + 1){
        MEM(slk , INF);
        while(1){
            MEM(s , 0) , MEM(t , 0);
            if(match(i)) break;
            else update();
        }
    }
}
```

5.5 Maximum Weighted Matching(General Graph)

```
struct WeightGraph {
  static const int INF = INT_MAX;
  static const int N = 514;
  struct edge{
    int u,v,w; edge(){}
    edge(int ui,int vi,int wi)
      :u(ui),v(vi),w(wi){}
  };
  int n,n_x;
  edge g[N*2][N*2];
  int lab[N*2];
  int match[N*2],slack[N*2],st[N*2],pa[N*2];
  int flo_from[N*2][N+1],S[N*2],vis[N*2];
  vector<int> flo[N*2];
  queue<int> q;
  int e_delta(const edge &e){
    return lab[e.u]+lab[e.v]-g[e.u][e.v].w*2;
  }
  void update_slack(int u,int x){
    if(!slack[x]||e_delta(g[u][x])<e_delta(g[slack[x]][
        x]))slack[x]=u;
```

```cpp
  }
  void set_slack(int x){
    slack[x]=0;
    for(int u=1;u<=n;++u)
      if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)
        update_slack(u,x);
  }
  void q_push(int x){
    if(x<=n)q.push(x);
    else for(size_t i=0;i<flo[x].size();i++)
      q_push(flo[x][i]);
  }
  void set_st(int x,int b){
    st[x]=b;
    if(x>n)for(size_t i=0;i<flo[x].size();++i)
      set_st(flo[x][i],b);
  }
  int get_pr(int b,int xr){
    int pr=find(flo[b].begin(),flo[b].end(),xr)-flo[b].
        begin();
    if(pr%2==1){
      reverse(flo[b].begin()+1,flo[b].end());
      return (int)flo[b].size()-pr;
    }else return pr;
  }
  void set_match(int u,int v){
    match[u]=g[u][v].v;
    if(u<=n) return;
    edge e=g[u][v];
    int xr=flo_from[u][e.u],pr=get_pr(u,xr);
    for(int i=0;i<pr;++i)set_match(flo[u][i],flo[u][i
        ^1]);
    set_match(xr,v);
    rotate(flo[u].begin(),flo[u].begin()+pr,flo[u].end
        ());
  }
  void augment(int u,int v){
    for(;;){
      int xnv=st[match[u]];
      set_match(u,v);
      if(!xnv)return;
      set_match(xnv,st[pa[xnv]]);
      u=st[pa[xnv]],v=xnv;
    }
  }
  int get_lca(int u,int v){
    static int t=0;
    for(++t;u||v;swap(u,v)){
      if(u==0)continue;
      if(vis[u]==t)return u;
      vis[u]=t;
      u=st[match[u]];
      if(u)u=st[pa[u]];
    }
    return 0;
  }
  void add_blossom(int u,int lca,int v){
    int b=n+1;
    while(b<=n_x&&st[b])++b;
    if(b>n_x)++n_x;
    lab[b]=0,S[b]=0;
    match[b]=match[lca];
    flo[b].clear();
    flo[b].push_back(lca);
    for(int x=u,y;x!=lca;x=st[pa[y]])
      flo[b].push_back(x),flo[b].push_back(y=st[match[x
          ]]),q_push(y);
    reverse(flo[b].begin()+1,flo[b].end());
    for(int x=v,y;x!=lca;x=st[pa[y]])
      flo[b].push_back(x),flo[b].push_back(y=st[match[x
          ]]),q_push(y);
    set_st(b,b);
    for(int x=1;x<=n_x;++x)g[b][x].w=g[x][b].w=0;
    for(int x=1;x<=n;++x)flo_from[b][x]=0;
    for(size_t i=0;i<flo[b].size();++i){
      int xs=flo[b][i];
      for(int x=1;x<=n_x;++x)
        if(g[b][x].w==0||e_delta(g[xs][x])<e_delta(g[b
            ][x]))
          g[b][x]=g[xs][x],g[x][b]=g[x][xs];
      for(int x=1;x<=n;++x)
        if(flo_from[xs][x])flo_from[b][x]=xs;
```

```cpp
    }
    set_slack(b);
  }
  void expand_blossom(int b){
    for(size_t i=0;i<flo[b].size();++i)
      set_st(flo[b][i],flo[b][i]);
    int xr=flo_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
    for(int i=0;i<pr;i+=2){
      int xs=flo[b][i],xns=flo[b][i+1];
      pa[xs]=g[xns][xs].u;
      S[xs]=1,S[xns]=0;
      slack[xs]=0,set_slack(xns);
      q_push(xns);
    }
    S[xr]=1,pa[xr]=pa[b];
    for(size_t i=pr+1;i<flo[b].size();++i){
      int xs=flo[b][i];
      S[xs]=-1,set_slack(xs);
    }
    st[b]=0;
  }
  bool on_found_edge(const edge &e){
    int u=st[e.u],v=st[e.v];
    if(S[v]==-1){
      pa[v]=e.u,S[v]=1;
      int nu=st[match[v]];
      slack[v]=slack[nu]=0;
      S[nu]=0,q_push(nu);
    }else if(S[v]==0){
      int lca=get_lca(u,v);
      if(!lca)return augment(u,v),augment(v,u),true;
      else add_blossom(u,lca,v);
    }
    return false;
  }
  bool matching(){
    memset(S+1,-1,sizeof(int)*n_x);
    memset(slack+1,0,sizeof(int)*n_x);
    q=queue<int>();
    for(int x=1;x<=n_x;++x)
      if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
    if(q.empty())return false;
    for(;;){
      while(q.size()){
        int u=q.front();q.pop();
        if(S[st[u]]==1)continue;
        for(int v=1;v<=n;++v)
          if(g[u][v].w>0&&st[u]!=st[v]){
            if(e_delta(g[u][v])==0){
              if(on_found_edge(g[u][v]))return true;
            }else update_slack(u,st[v]);
          }
      }
      int d=INF;
      for(int b=n+1;b<=n_x;++b)
        if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
      for(int x=1;x<=n_x;++x)
        if(st[x]==x&&slack[x]){
          if(S[x]==-1)d=min(d,e_delta(g[slack[x]][x]));
          else if(S[x]==0)d=min(d,e_delta(g[slack[x]][x
              ])/2);
        }
      for(int u=1;u<=n;++u){
        if(S[st[u]]==0){
          if(lab[u]<=d)return 0;
          lab[u]-=d;
        }else if(S[st[u]]==1)lab[u]+=d;
      }
      for(int b=n+1;b<=n_x;++b)
        if(st[b]==b){
          if(S[st[b]]==0)lab[b]+=d*2;
          else if(S[st[b]]==1)lab[b]-=d*2;
        }
      q=queue<int>();
      for(int x=1;x<=n_x;++x)
        if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&e_delta
            (g[slack[x]][x])==0)
          if(on_found_edge(g[slack[x]][x]))return true;
      for(int b=n+1;b<=n_x;++b)
        if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(
            b);
    }
```

```cpp
      return false;
    }
  pair<long long,int> solve(){
    memset(match+1,0,sizeof(int)*n);
    n_x=n;
    int n_matches=0;
    long long tot_weight=0;
    for(int u=0;u<=n;++u)st[u]=u,flo[u].clear();
    int w_max=0;
    for(int u=1;u<=n;++u)
      for(int v=1;v<=n;++v){
        flo_from[u][v]=(u==v?u:0);
        w_max=max(w_max,g[u][v].w);
      }
    for(int u=1;u<=n;++u)lab[u]=w_max;
    while(matching())++n_matches;
    for(int u=1;u<=n;++u)
      if(match[u]&&match[u]<u)
        tot_weight+=g[u][match[u]].w;
    return make_pair(tot_weight,n_matches);
  }
  void add_edge( int ui , int vi , int wi ){
    g[ui][vi].w = g[vi][ui].w = wi;
  }
  void init( int _n ){
    n = _n;
    for(int u=1;u<=n;++u)
      for(int v=1;v<=n;++v)
        g[u][v]=edge(u,v,0);
  }
} graph;
```

5.6 Minimum Weighted Matching

```cpp
struct Graph {
  // Minimum General Weighted Matching (Perfect Match)
  static const int MXN = 105;
  int n, edge[MXN][MXN];
  int match[MXN],dis[MXN],onstk[MXN];
  vector<int> stk;
  void init(int _n) {
    n = _n;
    for( int i = 0 ; i < n ; i ++ )
      for( int j = 0 ; j < n ; j ++ )
        edge[ i ][ j ] = 0;
  }
  void add_edge(int u, int v, int w)
  { edge[u][v] = edge[v][u] = w; }
  bool SPFA(int u){
    if (onstk[u]) return true;
    stk.PB(u);
    onstk[u] = 1;
    for (int v=0; v<n; v++){
      if (u != v && match[u] != v && !onstk[v]){
        int m = match[v];
        if (dis[m] > dis[u] - edge[v][m] + edge[u][v]){
          dis[m] = dis[u] - edge[v][m] + edge[u][v];
          onstk[v] = 1;
          stk.PB(v);
          if (SPFA(m)) return true;
          stk.pop_back();
          onstk[v] = 0;
        }
      }
    }
    onstk[u] = 0;
    stk.pop_back();
    return false;
  }
  int solve() {
    // find a match
    for (int i=0; i<n; i+=2){
      match[i] = i+1;
      match[i+1] = i;
    }
    while (true){
      int found = 0;
      for( int i = 0 ; i < n ; i ++ )
        onstk[ i ] = dis[ i ] = 0;
      for (int i=0; i<n; i++){
```

```cpp
        stk.clear();
        if (!onstk[i] && SPFA(i)){
          found = 1;
          while (SZ(stk)>=2){
            int u = stk.back(); stk.pop_back();
            int v = stk.back(); stk.pop_back();
            match[u] = v;
            match[v] = u;
          }
        }
      }
      if (!found) break;
    }
    int ret = 0;
    for (int i=0; i<n; i++)
      ret += edge[i][match[i]];
    ret /= 2;
    return ret;
  }
}graph;
```

5.7 Minimum mean cycle

```cpp
/* minimum mean cycle O(VE) */
struct MMC{
#define E 101010
#define V 1021
#define inf 1e9
#define eps 1e-6
  struct Edge { int v,u; double c; };
  int n, m, prv[V][V], prve[V][V], vst[V];
  Edge e[E];
  vector<int> edgeID, cycle, rho;
  double d[V][V];
  void init( int _n )
  { n = _n; m = 0; }
  // WARNING: TYPE matters
  void addEdge( int vi , int ui , double ci )
  { e[ m ++ ] = { vi , ui , ci }; }
  void bellman_ford() {
    for(int i=0; i<n; i++) d[0][i]=0;
    for(int i=0; i<n; i++) {
      fill(d[i+1], d[i+1]+n, inf);
      for(int j=0; j<m; j++) {
        int v = e[j].v, u = e[j].u;
        if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
          d[i+1][u] = d[i][v]+e[j].c;
          prv[i+1][u] = v;
          prve[i+1][u] = j;
        }
      }
    }
  }
  double solve(){
    // returns inf if no cycle, mmc otherwise
    double mmc=inf;
    int st = -1;
    bellman_ford();
    for(int i=0; i<n; i++) {
      double avg=-inf;
      for(int k=0; k<n; k++) {
        if(d[n][i]<inf-eps) avg=max(avg,(d[n][i]-d[k][i
          ])/(n-k));
        else avg=max(avg,inf);
      }
      if (avg < mmc) tie(mmc, st) = tie(avg, i);
    }
    FZ(vst); edgeID.clear(); cycle.clear(); rho.clear()
      ;
    for (int i=n; !vst[st]; st=prv[i--][st]) {
      vst[st]++;
      edgeID.PB(prve[i][st]);
      rho.PB(st);
    }
    while (vst[st] != 2) {
      int v = rho.back(); rho.pop_back();
      cycle.PB(v);
      vst[v]++;
    }
```

```
      reverse(ALL(edgeID));
      edgeID.resize(SZ(cycle));
      return mmc;
   }
} mmc;
```

5.8 Heavy-Light decomposition

```
#define MAX 100900
#define ls (now << 1)
#define rs (now << 1 | 1)
#define mid (l + r >> 1)

int siz[MAX] , son[MAX] , dep[MAX] , ffa[MAX];
int top[MAX] , idx[MAX] , idpo = 0;
int n , m;
int e[MAX][3];
vector<int> v[MAX];
struct node{ int big , sml; } st[MAX * 4];
void init(){
    REP(i , 0 , MAX) v[i].clear();
    MEM(siz , 0) , MEM(son , 0) , MEM(dep , 0) , MEM(
        ffa , 0);
    MEM(top , 0) , MEM(idx , 0) , idpo = 0;
}
void DFS1(int now , int fa , int deep){
    siz[now] = 1;
    dep[now] = deep;
    ffa[now] = fa;
    int big = 0;
    REP(i , 0 , v[now].size()){
        int to = v[now][i];
        if(to != fa){
            DFS1(to , now , deep + 1);
            siz[now] += siz[to];
            if(siz[to] > big) big = siz[to] , son[now]
                = to;
        }
    }
}
void DFS2(int now , int fa , int root){
    top[now] = root;
    idx[now] = ++idpo;
    if(son[now] != 0) DFS2(son[now] , now , root);
    REP(i , 0 , v[now].size()){
        int to = v[now][i];
        if(to != fa && to != son[now]) DFS2(to , now ,
            to);
    }
}
void solveinit(){
    DFS1(1 , 0 , 0);
    DFS2(1 , 0 , 1);
    REP(i , 2 , n + 1){
        int a = e[i][0] , b = e[i][1] , c = e[i][2];
        if(dep[a] < dep[b]) swap(a , b);
        update(1 , 1 , n , idx[a] , c);
    }
}
void query(int a , int b){
    node ans;
    ans.big = -INF , ans.sml = INF;
    int t1 = top[a] , t2 = top[b];
    while(t1 != t2){
        if(dep[t1] < dep[t2]) swap(t1 , t2) , swap(a ,
            b);
        ans = pull(ans , query(1 , 1 , n , idx[t1] ,
            idx[a]));
        a = ffa[t1] , t1 = top[a];
    }
    if(dep[a] > dep[b]) swap(a , b);
    if(a != b) ans = pull(ans , query(1 , 1 , n , idx[
        son[a]] , idx[b]));
    return cout << ans.sml << " " << ans.big << endl ,
        void();
}
init();
REP(i , 2 , n + 1){
    int a , b , c; cin >> a >> b >> c;
    e[i][0] = a , e[i][1] = b , e[i][2] = c;
```

```
      v[a].pb(b); v[b].pb(a);
}
solveinit();
query(a , b);
```

5.9 Dynamic MST

```
/* Dynamic MST O( Q lg^2 Q )
 (qx[i], qy[i])->chg weight of edge No.qx[i] to qy[i]
 delete an edge: (i, \infty)
 add an edge: change from \infty to specific value
 */
const int SZ=M+3*MXQ;
int a[N],*tz;
int find(int xx){
   int root=xx; while(a[root]) root=a[root];
   int next; while((next=a[xx])){a[xx]=root; xx=next; }
   return root;
}
bool cmp(int aa,int bb){ return tz[aa]<tz[bb]; }
int kx[N],ky[N],kt, vd[N],id[M], app[M];
bool extra[M];
void solve(int *qx,int *qy,int Q,int n,int *x,int *y,
    int *z,int m1,long long ans){
   if(Q==1){
      for(int i=1;i<=n;i++) a[i]=0;
      z[ qx[0] ]=qy[0]; tz = z;
      for(int i=0;i<m1;i++) id[i]=i;
      sort(id,id+m1,cmp); int ri,rj;
      for(int i=0;i<m1;i++){
         ri=find(x[id[i]]); rj=find(y[id[i]]);
         if(ri!=rj){ ans+=z[id[i]]; a[ri]=rj; }
      }
      printf("%lld\n",ans);
      return;
   }
   int ri,rj;
   //contract
   kt=0;
   for(int i=1;i<=n;i++) a[i]=0;
   for(int i=0;i<Q;i++){
      ri=find(x[qx[i]]); rj=find(y[qx[i]]); if(ri!=rj) a[
          ri]=rj;
   }
   int tm=0;
   for(int i=0;i<m1;i++) extra[i]=true;
   for(int i=0;i<Q;i++) extra[ qx[i] ]=false;
   for(int i=0;i<m1;i++) if(extra[i]) id[tm++]=i;
   tz=z; sort(id,id+tm,cmp);
   for(int i=0;i<tm;i++){
      ri=find(x[id[i]]); rj=find(y[id[i]]);
      if(ri!=rj){
         a[ri]=rj; ans += z[id[i]];
         kx[kt]=x[id[i]]; ky[kt]=y[id[i]]; kt++;
      }
   }
   for(int i=1;i<=n;i++) a[i]=0;
   for(int i=0;i<kt;i++) a[ find(kx[i]) ]=find(ky[i]);
   int n2=0;
   for(int i=1;i<=n;i++) if(a[i]==0)
   vd[i]=++n2;
   for(int i=1;i<=n;i++) if(a[i])
   vd[i]=vd[find(i)];
   int m2=0, *Nx=x+m1, *Ny=y+m1, *Nz=z+m1;
   for(int i=0;i<m1;i++) app[i]=-1;
   for(int i=0;i<Q;i++) if(app[qx[i]]==-1){
      Nx[m2]=vd[ x[ qx[i] ] ]; Ny[m2]=vd[ y[ qx[i] ] ];
          Nz[m2]=z[ qx[i] ];
      app[qx[i]]=m2; m2++;
   }
   for(int i=0;i<Q;i++){ z[ qx[i] ]=qy[i]; qx[i]=app[qx[
       i]]; }
   for(int i=1;i<=n2;i++) a[i]=0;
   for(int i=0;i<tm;i++){
      ri=find(vd[ x[id[i]] ]);  rj=find(vd[ y[id[i]] ]);
      if(ri!=rj){
         a[ri]=rj; Nx[m2]=vd[ x[id[i]] ];
         Ny[m2]=vd[ y[id[i]] ]; Nz[m2]=z[id[i]]; m2++;
      }
   }
```

```
  int mid=Q/2;
  solve(qx,qy,mid,n2,Nx,Ny,Nz,m2,ans);
  solve(qx+mid,qy+mid,Q-mid,n2,Nx,Ny,Nz,m2,ans);
}
int x[SZ],y[SZ],z[SZ],qx[MXQ],qy[MXQ],n,m,Q;
void init(){
  scanf("%d%d",&n,&m);
  for(int i=0;i<m;i++) scanf("%d%d%d",x+i,y+i,z+i);
  scanf("%d",&Q);
  for(int i=0;i<Q;i++){ scanf("%d%d",qx+i,qy+i); qx[i
    ]--; }
}
void work(){ if(Q) solve(qx,qy,Q,n,x,y,z,m,0); }
int main(){init(); work(); }
```

5.10 Minimum Steiner Tree

```
// Minimum Steiner Tree
// O(V 3^T + V^2 2^T)
struct SteinerTree{
#define V 33
#define T 8
#define INF 1023456789
  int n , dst[V][V] , dp[1 << T][V] , tdst[V];
  void init( int _n ){
    n = _n;
    for( int i = 0 ; i < n ; i ++ ){
      for( int j = 0 ; j < n ; j ++ )
        dst[ i ][ j ] = INF;
      dst[ i ][ i ] = 0;
    }
  }
  void add_edge( int ui , int vi , int wi ){
    dst[ ui ][ vi ] = min( dst[ ui ][ vi ] , wi );
    dst[ vi ][ ui ] = min( dst[ vi ][ ui ] , wi );
  }
  void shortest_path(){
    for( int k = 0 ; k < n ; k ++ )
      for( int i = 0 ; i < n ; i ++ )
        for( int j = 0 ; j < n ; j ++ )
          dst[ i ][ j ] = min( dst[ i ][ j ],
                 dst[ i ][ k ] + dst[ k ][ j ] );
  }
  int solve( const vector<int>& ter ){
    int t = (int)ter.size();
    for( int i = 0 ; i < ( 1 << t ) ; i ++ )
      for( int j = 0 ; j < n ; j ++ )
        dp[ i ][ j ] = INF;
    for( int i = 0 ; i < n ; i ++ )
      dp[ 0 ][ i ] = 0;
    for( int msk = 1 ; msk < ( 1 << t ) ; msk ++ ){
      if( msk == ( msk & (-msk) ) ){
        int who = __lg( msk );
        for( int i = 0 ; i < n ; i ++ )
          dp[ msk ][ i ] = dst[ ter[ who ] ][ i ];
        continue;
      }
      for( int i = 0 ; i < n ; i ++ )
        for( int submsk = ( msk - 1 ) & msk ; submsk ;
               submsk = ( submsk - 1 ) & msk )
          dp[ msk ][ i ] = min( dp[ msk ][ i ],
                     dp[ submsk ][ i ] +
                     dp[ msk ^ submsk ][ i ] );
      for( int i = 0 ; i < n ; i ++ ){
        tdst[ i ] = INF;
        for( int j = 0 ; j < n ; j ++ )
          tdst[ i ] = min( tdst[ i ],
                  dp[ msk ][ j ] + dst[ j ][ i ] );
      }
      for( int i = 0 ; i < n ; i ++ )
        dp[ msk ][ i ] = tdst[ i ];
    }
    int ans = INF;
    for( int i = 0 ; i < n ; i ++ )
      ans = min( ans , dp[ ( 1 << t ) - 1 ][ i ] );
    return ans;
  }
} solver;
```

5.11 Maximum Clique

```
struct maximum_clique {
  static const int MAX_N = 81;
  typedef bitset<MAX_N> bst;
  bst N[MAX_N],empty;
  int n,ans;
  void init(int _n) {
    //point from 0 ~ n-1
    n=_n;
    for (int i=0;MAX_N>i;i++) {
      N[i] = empty;
    }
  }
  void add_edge(int a,int b) {
    N[a][b] = N[b][a] = 1;
  }
  void sagiri(bst R,bst P,bst X) {
    if (P==empty && X==empty) {
      ans = max(ans,(int)R.count());
      return;
    }
    bst tmp=P|X;
    int u;
    if ((R|P|X).count() <= ans) return;
    for (u=0;n>u;u++) {
      if (tmp[u]) break;
    }
    bst now = P&~N[u];  //P-N[u]
    for (int v=0;n>v;v++) {
      if (now[v]) {
        R[v] = true;
        sagiri(R,P&N[v],X&N[v]);
        R[v] = false; P[v] = false; X[v] = true;
      }
    }
  }
  int solve() {
    ans=0;
    bst R=empty,P,X=empty;
    P.flip();
    sagiri(R,P,X);
    return ans;
  }
} solver;
```

6 Math

6.1 Big Integer

```
struct Bigint{
  static const int LEN = 60;
  static const int BIGMOD = 10000;
  int s;
  int vl, v[LEN];
  //  vector<int> v;
  Bigint() : s(1) { vl = 0; }
  Bigint(long long a) {
    s = 1; vl = 0;
    if (a < 0) { s = -1; a = -a; }
    while (a) {
      push_back(a % BIGMOD);
      a /= BIGMOD;
    }
  }
  Bigint(string str) {
    s = 1; vl = 0;
    int stPos = 0, num = 0;
    if (!str.empty() && str[0] == '-') {
      stPos = 1;
      s = -1;
    }
    for (int i=SZ(str)-1, q=1; i>=stPos; i--) {
      num += (str[i] - '0') * q;
      if ((q *= 10) >= BIGMOD) {
        push_back(num);
        num = 0; q = 1;
      }
    }
```

```cpp
    }
    if (num) push_back(num);
    n();
  }
  int len() const {
    return vl;
    //     return SZ(v);
  }
  bool empty() const { return len() == 0; }
  void push_back(int x) {
    v[vl++] = x;
    //     v.PB(x);
  }
  void pop_back() {
    vl--;
    //     v.pop_back();
  }
  int back() const {
    return v[vl-1];
    //     return v.back();
  }
  void n() {
    while (!empty() && !back()) pop_back();
  }
  void resize(int nl) {
    vl = nl;
    fill(v, v+vl, 0);
    //     v.resize(nl);
    //     fill(ALL(v), 0);
  }
  void print() const {
    if (empty()) { putchar('0'); return; }
    if (s == -1) putchar('-');
    printf("%d", back());
    for (int i=len()-2; i>=0; i--) printf("%.4d",v[i]);
  }
  friend std::ostream& operator << (std::ostream& out,
      const Bigint &a) {
    if (a.empty()) { out << "0"; return out; }
    if (a.s == -1) out << "-";
    out << a.back();
    for (int i=a.len()-2; i>=0; i--) {
      char str[10];
      snprintf(str, 5, "%.4d", a.v[i]);
      out << str;
    }
    return out;
  }
  int cp3(const Bigint &b)const {
    if (s != b.s) return s - b.s;
    if (s == -1) return -(-*this).cp3(-b);
    if (len() != b.len()) return len()-b.len();//int
    for (int i=len()-1; i>=0; i--)
      if (v[i]!=b.v[i]) return v[i]-b.v[i];
    return 0;
  }
  bool operator<(const Bigint &b)const
    { return cp3(b)<0; }
  bool operator<=(const Bigint &b)const
    { return cp3(b)<=0; }
  bool operator==(const Bigint &b)const
    { return cp3(b)==0; }
  bool operator!=(const Bigint &b)const
    { return cp3(b)!=0; }
  bool operator>(const Bigint &b)const
    { return cp3(b)>0; }
  bool operator>=(const Bigint &b)const
    { return cp3(b)>=0; }
  Bigint operator - () const {
    Bigint r = (*this);
    r.s = -r.s;
    return r;
  }
  Bigint operator + (const Bigint &b) const {
    if (s == -1) return -(-(*this)+(-b));
    if (b.s == -1) return (*this)-(-b);
    Bigint r;
    int nl = max(len(), b.len());
    r.resize(nl + 1);
    for (int i=0; i<nl; i++) {
      if (i < len()) r.v[i] += v[i];
      if (i < b.len()) r.v[i] += b.v[i];
```

```cpp
      if(r.v[i] >= BIGMOD) {
        r.v[i+1] += r.v[i] / BIGMOD;
        r.v[i] %= BIGMOD;
      }
    }
    r.n();
    return r;
  }
  Bigint operator - (const Bigint &b) const {
    if (s == -1) return -(-(*this)-(-b));
    if (b.s == -1) return (*this)+(-b);
    if ((*this) < b) return -(b-(*this));
    Bigint r;
    r.resize(len());
    for (int i=0; i<len(); i++) {
      r.v[i] += v[i];
      if (i < b.len()) r.v[i] -= b.v[i];
      if (r.v[i] < 0) {
        r.v[i] += BIGMOD;
        r.v[i+1]--;
      }
    }
    r.n();
    return r;
  }
  Bigint operator * (const Bigint &b) {
    Bigint r;
    r.resize(len() + b.len() + 1);
    r.s = s * b.s;
    for (int i=0; i<len(); i++) {
      for (int j=0; j<b.len(); j++) {
        r.v[i+j] += v[i] * b.v[j];
        if(r.v[i+j] >= BIGMOD) {
          r.v[i+j+1] += r.v[i+j] / BIGMOD;
          r.v[i+j] %= BIGMOD;
        }
      }
    }
    r.n();
    return r;
  }
  Bigint operator / (const Bigint &b) {
    Bigint r;
    r.resize(max(1, len()-b.len()+1));
    int oriS = s;
    Bigint b2 = b; // b2 = abs(b)
    s = b2.s = r.s = 1;
    for (int i=r.len()-1; i>=0; i--) {
      int d=0, u=BIGMOD-1;
      while(d<u) {
        int m = (d+u+1)>>1;
        r.v[i] = m;
        if((r*b2) > (*this)) u = m-1;
        else d = m;
      }
      r.v[i] = d;
    }
    s = oriS;
    r.s = s * b.s;
    r.n();
    return r;
  }
  Bigint operator % (const Bigint &b) {
    return (*this)-(*this)/b*b;
  }
};
```

6.2 FFT

```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 2*262144;
typedef long double ld;
#define ld double
typedef complex<ld> cplx;
const ld PI = acos(-1);
const cplx I(0,1);
cplx omega[MAXN+1];
void pre_fft() {
```

```
  for (int i=0;i<=MAXN;i++) {
    omega[i] = exp(i*2*PI/MAXN*I);
  }
}
void fft(int n,cplx a[],bool inv=false) {
  int basic=MAXN/n;
  int theta=basic;
  for (int m=n;m>=2;m>>=1) {
    int mh=m>>1;
    for (int i=0;i<mh;i++) {
      cplx w=omega[inv?MAXN-(i*theta%MAXN):i*theta%MAXN
        ];
      for (int j=i;j<n;j+=m) {
        int k=j+mh;
        cplx x=a[j]-a[k];
        a[j] += a[k];
        a[k] = w*x;
      }
    }
    theta = (theta*2)%MAXN;
  }
  int i=0;
  for (int j=1;j<n-1;j++) {
    for (int k=n>>1;k>(i^=k);k>>=1) ;
    if (j<i) swap(a[i],a[j]);
  }
  if (inv) {
    for (int i=0;i<n;i++) a[i]/=n;
  }
}

cplx a[MAXN],b[MAXN],c[MAXN];
//how to use :
/*
pre_fft();
fft(n,a);
fft(n,b);
for (int i=0;n>i;i++) {
  c[i] = a[i]*b[i];
}
fft(n,c,1);
*/
```

6.3 NTT

```
// Remember coefficient are mod P
/*
(mod,root)
(65537,3)
(23068673,3)
(998244353,3)
(1107296257,10)
(2013265921,31)
(2885681153,3)
*/
typedef long long ll;
const int maxn = 65536;

struct NTT{
    ll mod = 2013265921, root = 31;
    ll omega[maxn+1];
    void prentt() {
        ll x=fpow(root,(mod-1)/maxn);
        omega[0] = 1;
        for (int i=1;i<=maxn;++i) {
            omega[i] = omega[i-1] * x % mod;
        }
    }
    void real_init(ll _mod,ll _root) {
        mod = _mod;
        root = _root;
        prentt();
    }
    ll fpow(ll a,ll n) {
        (n += mod-1) %= mod - 1;
        ll r = 1;
        for (; n; n>>=1) {
            if (n&1) (r*=a)%=mod;
            (a*=a)%=mod;
        }
```

```
        return r;
    }
    void bitrev(vector<ll> &v,int n) {
        int z = __builtin_ctz(n)-1;
        for (int i=0;i<n;++i) {
            int x=0;
            for (int j=0;j<=z;++j) x ^= ((i>>j&1) << (z
                -j));
            if (x>i) swap(v[x],v[i]);
        }
    }
    void ntt(vector<ll> &v,int n) {
        bitrev(v,n);
        for (int s=2;s<=n;s<<=1) {
            int z = s>>1;
            for (int i=0;i<n;i+=s) {
                for (int k=0;k<z;++k) {
                    ll x = v[i+k+z] * omega[maxn/s * k]
                        % mod;
                    v[i+k+z] = (v[i+k] + mod - x)%mod;
                    (v[i+k] += x) %= mod;
                }
            }
        }
    }
    void intt(vector<ll> &v,int n) {
        ntt(v,n);
        reverse(v.begin()+1,v.end());
        ll inv = fpow(n,mod-2);
        for (int i=0;i<n;++i) {
            (v[i] *= inv) %= mod;
        }
    }
    vector<ll> conv(vector<ll> a,vector<ll> b) {
        int sz=1;
        while (sz < a.size() + b.size() - 1) sz <<= 1;
        vector<ll> c(sz);
        while (a.size() < sz) a.push_back(0);
        while (b.size() < sz) b.push_back(0);
        ntt(a,sz), ntt(b,sz);
        for (int i=0;i<sz;++i) c[i] = (a[i] * b[i]) %
            mod;
        intt(c,sz);
        while (c.size() && c.back() == 0) c.pop_back();
        return c;
    }
};
```

6.4 FWT

```
void FWT(ll a[],int n){
    for(int d = 1 ;d < n; d <<= 1)  // d = half of
        block size
        for(int  i = 0; i < n; i += d + d ) // every
            block
            for(int j = i; j < i + d; j++){   //
                processing
                ll x = a[j], y = a[j + d];
                //FWT
                //XOR
                a[j] = x + y;   a[j + d] = x - y;
                //AND
                a[j] = x + y;
                //OR
                a[j + d] = y + x;
                //IFWT
                //XOR
                a[j] = (x + y) / 2;   a[j + d] = (x - y
                    ) / 2;
                //AND
                a[j] = x - y;
                //OR
                a[j + d] = y - x;
            }
}
```

6.5 Gauss

```cpp
const int GAUSS_MOD = 100000007LL;
struct GAUSS{
    int n;
    vector<vector<int>> v;
    int ppow(int a , int k){
        if(k == 0) return 1;
        if(k % 2 == 0) return ppow(a * a % GAUSS_MOD ,
            k >> 1);
        if(k % 2 == 1) return ppow(a * a % GAUSS_MOD ,
            k >> 1) * a % GAUSS_MOD;
    }
    vector<int> solve(){
        vector<int> ans(n);
        REP(now , 0 , n){
            REP(i , now , n) if(v[now][now] == 0 && v[i
                ][now] != 0)
                swap(v[i] , v[now]); // det = -det;
            if(v[now][now] == 0) return ans;
            int inv = ppow(v[now][now] , GAUSS_MOD - 2)
                ;
            REP(i , 0 , n) if(i != now){
                int tmp = v[i][now] * inv % GAUSS_MOD;
                REP(j , now , n + 1) (v[i][j] +=
                    GAUSS_MOD - tmp * v[now][j] %
                    GAUSS_MOD) %= GAUSS_MOD;
            }
        }
        REP(i , 0 , n) ans[i] = v[i][n + 1] * ppow(v[i
            ][i] , GAUSS_MOD - 2) % GAUSS_MOD;
        return ans;
    }
    // gs.v.clear() , gs.v.resize(n , vector<int>(n + 1
        , 0));
} gs;
```

6.6 Miller Rabin

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long LL;

LL mul(LL a,LL b,LL mod) {
    return a*b%mod;
    //calculate a*b % mod
    LL r=0;
    a%=mod; b%=mod;
    while (b) {
        if (b&1) r=(a+r>=mod?a+r-mod:a+r);
        a=(a+a>=mod?a+a-mod:a+a);
        b>>=1;
    }
    return r;
}

LL pow(LL a,LL n,LL mod) {
    if (n==0) return 1LL;
    else if (n==1) return a%mod;
    return mul( pow(mul(a,a,mod),n/2,mod),n%2?a:1,mod )
        ;
}

const bool PRIME = 1, COMPOSITE = 0;
bool miller_robin(LL n,LL a) {
    if (__gcd(a,n) == n) return PRIME;
    if (__gcd(a,n) != 1) return COMPOSITE;
    LL d=n-1,r=0,ret;
    while (d%2==0) {
        r++;
        d/=2;
    }
    ret = pow(a,d,n);
    if (ret==1 ||ret==n-1) return PRIME;
    while (r--) {
        ret = mul(ret,ret,n);
        if (ret==n-1) return PRIME;
    }
    return COMPOSITE;
}
```

```cpp
bool isPrime(LL n) {
    //for int: 2,7,61
    LL as[7] =
        {2,325,9375,28178,450775,9780504,1795265022};
    for (int i=0;7>i;i++) {
        if (miller_robin(n,as[i]) == COMPOSITE) return
            COMPOSITE;
    }
    return PRIME;
}
```

6.7 Pollard Rho

```cpp
//const int G = (1LL<<31)-1;
LL mull(LL a,LL b,LL mod) {
    //if (a<G && b<G) return a*b%mod;
    LL ret = 0;
    LL now = a;
    while (b) {
        if (b&1) ret = addd(ret, now, mod);
        now = addd(now, now, mod);
        b >>= 1;
    }
    return ret;
}
LL ppow(LL a,LL n,LL mod) {
    LL ret = 1;
    LL now = a;
    while (n) {
        if (n&1) ret = mull(ret, now, mod);
        now = mull(now, now, mod);
        n >>= 1;
    }
    return ret;
}
LL gcd(LL a, LL b) {
    if (b==0) return a;
    else return gcd(b, a%b);
}
const bool PRIME = 1, COMPOSITE = 0;
bool miller_rabin(LL n, LL a) {
    if (gcd(n, a) == n) return PRIME;
    else if (gcd(n, a) != 1) return COMPOSITE;
    LL d = n - 1, r = 0;
    while (d % 2 == 0) {
        d >>= 1;
        ++r;
    }
    LL ret = ppow(a, d, n);
    if (ret == 1 || ret == n - 1) return PRIME;
    while (r--) {
        ret = mull(ret, ret, n);
        if (ret == n - 1) return PRIME;
    }
    return COMPOSITE;
}
bool isPrime(LL n) {
    LL as[7] = {2, 325, 9375, 28178, 450775, 9780504,
        1795265022};
    for (int i = 0; 7 > i; ++i) {
        if (miller_rabin(n, as[i]) == COMPOSITE) return
            COMPOSITE;
    }
    return PRIME;
}

const LL C = 2934852462451LL;
const LL D = 126871905557494LL;
LL rnd = 98134513458734897LL;
LL myRnd() {
    return rnd = (rnd + C) ^ D;
}

LL a, c;

LL doo(LL x, LL n) {
    return addd( mull( a, mull(x, x, n), n ), c, n);
}

#define aabs(x) (x) >= 0 ? (x):-(x)
```

```cpp
LL solve(LL n) {
    if (isPrime(n)) return n;
    if (!(n & 1)) return 2;
    a = myRnd() % n;
    if (!a) a=1;
    c = myRnd() % n;
    while (c == 0 || c == 2) c = myRnd()%n;
    LL start = myRnd()%n;
    LL s1 = doo(start, n);
    LL s2 = doo(s1, n);
    while (true) {
        if (s1 == s2) {
            start = myRnd()%n;
            //a=myRnd()+1;
            a = myRnd() % n;
            if (!a) a = 1;
            c = myRnd() % n;
            while (c == 0 || c == 2) c = myRnd() % n;
            s1 = doo(start, n);
            s2 = doo(s1, n);
            continue;
        }
        LL _ = gcd(aabs(s1 - s2), n);
        if (_ != 1) {
            return min(solve(_), solve(n / _));
        }
        s1 = doo(s1, n);
        s2 = doo(s2, n);
        s2 = doo(s2, n);
    }
}
```

6.8 Pi

```cpp
## Meissel-Lehmer ##
```cpp
#define MEM1(a) memset( (a) , 0 , sizeof( (a) ) );
const int N = 320000 + 6;
const int C = 10005;
const int D = 306;
LL pi_form[N];
LL phi_form[C][D];
LL p2_form[C][D];
LL p[N];
bool prime[N];
void init() {
  MEM1(phi_form);
  MEM1(p2_form);
  prime[0] = prime[1] = 1;
  int id=1;
  for (int i=2;N>i;i++) {
    if (!prime[i]) {
      for (LL j=i*1LL*i;N>j;j+=i) prime[j] = 1;
      p[id++] = i;
    }
    pi_form[i] = pi_form[i-1] + (!prime[i]);
  }
}
LL pi(LL m);
LL p2(LL m,LL n) {
  //cout<<"p2 = "<<p2_form[m][n]<<endl;
  if (m<C && n<D && p2_form[m][n] != -1) return p2_form
    [m][n];
  if (p[n] == 0) return 0;
  LL ret = 0, tmp=sqrt(m);
  for (LL i=n+1;p[i] <= tmp;i++) ret += pi(m/p[i]) - pi
    (p[i]) + 1;
  if (m < C && n < D) p2_form[m][n] = ret;
  return ret;
}
LL phi2(LL m,LL n) {
  if (m < C && n < D && phi_form[m][n] != -1) return
    phi_form[m][n];
  if (!n) return m;
  if (p[n] >= m) return 1;
  if (m<C && n<D) return phi_form[m][n] =  phi2(m,n-1)
    - phi2(m/p[n],n-1);
  return phi2(m,n-1) - phi2(m/p[n],n-1);
}
```

```cpp
LL pi(LL m) {
  //cout<<"pi = "<<m<<endl;
  if (m < N) return pi_form[m];
  else {
    LL n=ceil(cbrt(m));
    return phi2(m,n) + n - 1 - p2(m,n);
  }
}
//init(); cin >> n; cout << pi(n); (n <= 10^11)
```
```

6.9 Debrujin

```cpp
int res[maxn], aux[maxn], a[maxn], sz;

void db(int t, int p, int n, int k) {
    if (sz >= tg) return;
    if (t > n) {
        if (n % p == 0) {
            for (int i = 1; i <= p && sz < tg; ++i) res
                [sz++] = aux[i];
        }
    } else {
        aux[t] = aux[t - p];
        db(t + 1, p, n, k);
        for (int i = aux[t - p] + 1; i < k; ++i) {
            aux[t] = i;
            db(t + 1, t, n, k);
        }
    }
}

int de_bruijn(int k, int n) {
    // return cyclic string of length k^n such that
        every string of length n using k character
        appears as a substring.
    if (k == 1) {
        res[0] = 0;
        return 1;
    }
    for (int i = 0; i < k * n; i++) aux[i] = 0;
    sz = 0;
    db(1, 1, n, k);
    return sz;
}
```

7 String

7.1 string tools

```cpp
const KMP_SIZE = ;
struct KMP{
    string s;
    int f[KMP_SIZE] , pos;
    void solve(){
        f[0] = pos = -1;
        REP(i , 1 , s.size()){
            while(pos != -1 && s[pos + 1] != s[i]) pos
                = f[pos];
            if(s[pos + 1] == s[i]) pos ++;
            f[i] = pos;
        }
    }
};
const int ZVALUE_SIZE = ;
struct Z_VALUE{
    string s;
    int l = 0 , r = 0 , z[ZVALUE_SIZE];
    void solve(){
        REP(i , 0 , s.size()){
            z[i] = max(min(z[i - l] , r - i) , 0LL);
            while(i + z[i] < s.size() && s[z[i]] == s[i
                + z[i]]){
                l = i , r = i + z[i];
                z[i] ++;
            }
        }
```

```
            }
        }
};
const int PALINDROME_MAX = 2 * ;
struct Palindrome{
    string s , ss; // ss = input
    int z[PALINDROME_MAX];
    void solve(){
        s.resize(ss.size() + ss.size() + 1 , '.');
        REP(i , 0 , ss.size()) s[i + i + 1] = ss[i];
        int l = 0 , r = 0;
        REP(i , 0 , s.size()){
            z[i] = max(min(z[l + l - i] , r - i) , 1);
            while(i - z[i] >= 0 && i + z[i] < s.size()
                && s[i - z[i]] == s[i + z[i]]){
                l = i , r = i + z[i];
                z[i] ++;
            }
        }
    }
};
```

7.2 Aho–Corasick algorithm

```
#include <bits/stdc++.h>
using namespace std;

struct AC_Automata {
    static const int N = 2e4 + 6;
    static const int SIGMA = 26;
    int ch[N][SIGMA];
    int val[N];
    int sz;
    int last[N],fail[N];
    int que[N],qs,qe;
    int cnt[N];
    void init() {
        sz = 1;
        memset(ch[0],0,sizeof(ch[0]));
        qs = qe  = 0;
        memset(cnt,0,sizeof(cnt)); memset(val,0,sizeof(
            val)); memset(last,0,sizeof(last));
    }
    int idx(char c) {
        return c-'a';
    }
    int insert(string s,int v) {
        int now=0;
        int n=s.size();
        for (int i=0;n>i;i++) {
            int c=idx(s[i]);
            if (!ch[now][c]) {
                memset(ch[sz],0,sizeof(ch[sz]));
                val[sz] = 0;
                ch[now][c] = sz++;
            }
            now = ch[now][c];
        }
        val[now] = v;
        return now;
    }
    void print(int j) {
        if (j) {
            //now we match string v[j]
            print(last[j]);  //may match multiple
                strings
        }
    }
    void getFail() {
        qs=0,qe=0;
        fail[0]=0;
        for (int c=0;SIGMA >c;c++) {
            int now=ch[0][c];
            if (now) {
                fail[now] = 0;
                que[qe++] = now;
                last[now] = 0;
            }
        }
        while (qs != qe) {
```

```
            int t=que[qs++];
            for (int c=0;SIGMA > c;c++) {
                int now=ch[t][c];
                if (!now) continue;
                que[qe++] = now;
                int v=fail[t];
                while (v && !ch[v][c]) v=fail[v];
                fail[now] = ch[v][c];
                last[now] = val[ fail[now] ]? fail[now
                    ]:last[ fail[now] ];
            }
        }
    }
    void Find(string s) {
        getFail();
        int n=s.size();
        int now=0;
        for (int i=0;n>i;i++) {
            int c=idx(s[i]);
            while (now && !ch[now][c]) now = fail[now];
            now = ch[now][c];
            cnt[now]++;
        }
        for (int i=qe-1;i>=0;i--) {
            cnt[ fail[que[i]] ] += cnt[ que[i] ];
        }
    }
    void AC_evolution() {
        for (qs=1;qs!=qe;) {
            int now=que[qs++];
            for (int i=0;SIGMA>i;i++) {
                if (ch[now][i] == 0) ch[now][i] = ch[
                    fail[now]][i];
            }
        }
    }
} ac;

const int N = 156;
string s[N];
int ed[N];

ac.init();
ac.insert(s[i],i);
ac.Find();
ac.cnt[ ac.insert(s[i],i) ];
```

7.3 Suffix array

```
const int SA_SIZE = ;
const int logn = 1 + ;
string s;
int sa[SA_SIZE] , rk[SA_SIZE] , lcp[SA_SIZE];
int tma[2][SA_SIZE] , c[SA_SIZE] , sp[SA_SIZE][logn];

int getsa(){
    -> update m = ? // how many char
    int *x = tma[0] , *y = tma[1] , n = s.size() , m =
        200;
    REP(i , 0 , m) c[i] = 0;
    REP(i , 0 , n) c[x[i] = s[i]] ++;
    REP(i , 1 , m) c[i] += c[i - 1];
    RREP(i , n - 1 , 0) sa[--c[x[i]]] = i;
    for(int k = 1 ; k <= n ; k <<= 1){
        REP(i , 0 , m) c[i] = 0;
        REP(i , 0 , n) c[x[i]] ++;
        REP(i , 1 , m) c[i] += c[i - 1];
        int p = 0;
        REP(i , n - k , n) y[p ++] = i;
        REP(i , 0 , n) if(sa[i] >= k) y[p ++] = sa[i] -
            k;
        RREP(i , n - 1 , 0) sa[--c[x[y[i]]]] = y[i];
        y[sa[0]] = p = 0;
        REP(i , 1 , n) {
            if( x[sa[i]] == x[sa[i - 1]] && sa[i] + k <
                n && sa[i - 1] + k < n &&
                x[sa[i] + k] == x[sa[i - 1] + k] );
            else p ++;
            y[sa[i]] = p;
        }
    }
```

```
            swap(x , y);
            if(p + 1 == n) break;
            m = p + 1;
        }
    }
}
void getlcp(){
    int tmp = 0 , n = s.size();
    REP(i , 0 , n) rk[sa[i]] = i;
    REP(i , 0 , n){
        if(rk[i] == 0) lcp[0] = 0;
        else {
            if(tmp) tmp --;
            int po = sa[rk[i] - 1];
            while(tmp + po < n && tmp + i < n && s[tmp
                + i] == s[tmp + po]) tmp ++;
            lcp[rk[i]] = tmp;
        }
    }
}
void getsp(){
    int n = s.size();
    REP(i , 0 , n) sp[rk[i]][0] = s.size() - i;
    REP(i , 1 , n) sp[i - 1][1] = lcp[i];
    REP(i , 2 , logn){
        REP(j , 0 , n){
            if(j + (1 << (i - 2)) >= s.size()) continue
                ;
            sp[j][i] = min(sp[j][i - 1] , sp[j + (1 <<
                (i - 2))][i - 1]);
        }
    }
}
int Query(int L , int R){
    int tmp = (L == R) ? 0 : 32 - __builtin_clz(R - L);
    if(tmp == 0) return sp[L][0];
    else return min(sp[L][tmp] , sp[R - (1 << (tmp - 1)
        )][tmp]);
}
int Find(string ss){
    int L = 0 , R = s.size() , now;
    while(R - L > 1){
        now = (L + R) / 2;
        if(s[sa[now]] == ss[0]) break;
        else if(s[sa[now]] > ss[0]) R = now;
        else if(s[sa[now]] < ss[0]) L = now;
    }
    if(s[sa[now]] != ss[0]) return 0;
    REP(i , 1 , ss.size()){
        int pre = now , ty = 0;
        if(sa[now] + i >= s.size()) L = now , ty = 0;
        else if(s[sa[now] + i] == ss[i]) continue;
        else if(s[sa[now] + i] > ss[i]) R = now , ty =
            1;
        else if(s[sa[now] + i] < ss[i]) L = now , ty =
            0;

        while(R - L > 1){
            now = (L + R) / 2;
            if(sa[now] + i >= s.size()){
                if(ty == 0) R = now;
                if(ty == 1) L = now;
            }
            else if(ty == 0 && Query(pre , now) < i) R
                = now;
            else if(ty == 1 && Query(now , pre) < i) L
                = now;
            else if(s[sa[now] + i] == ss[i]) break;
            else if(s[sa[now] + i] > ss[i]) R = now;
            else if(s[sa[now] + i] < ss[i]) L = now;
        }
        if(sa[now] + i >= s.size()) return 0;
        if(s[sa[now] + i] != ss[i]) return 0;
    }
    L = now , R = now;
    RREP(i , 19 , 0){
        if(R + (1 << i) >= s.size()) continue;
        else if(Query(L , R + (1 << i)) >= ss.size()) R
            += (1 << i);
    }
    RREP(i , 19 , 0){
        if(L - (1 << i) < 0) continue;
```

```
        else if(Query(L - (1 << i) , R) >= ss.size()) L
            -= (1 << i);
    }
    return R - L + 1;
}
/*
how to use :
1. cin >> s;
2. getsa() , getlcp() , getsp();
3. string ss;
4. cin >> ss;
5. cout << Find(ss) << endl;
*/
```

7.4 Lexicographically Smallest Rotation

```
string s;
const int N = 4000006;
int f[N];
void solve() {
    s = s + s;
    int n = (int)s.size();
    for (int i=0;i<n;++i) f[i] = -1;
    int k=0;
    for (int j=1;j<n;++j) {
        char sj = s[j];
        int i = f[j-k-1];
        while (i != -1 && sj != s[k+i+1]) {
            if (sj < s[k+i+1]) {
                k = j-i-1;
            }
            i = f[i];
        }
        if (sj != s[k+i+1]) {
            if (sj < s[k]) {
                k = j;
            }
            f[j-k] = -1;
        }
        else f[j-k] = i+1;
    }
    n>>=1;
    if (k >= n) k-= n;
    for (int i=k;i<k+n;++i) {
        cout << s[i];
    }
    cout << endl;
}
```

8 Boook

8.1 Block Tree

```
//Query on Tree 1, SPOJ

#define MAX 10900
#define INF 0x3f3f3f3f

int t , n , m , N = 100;
vector<int> v[MAX] , g[MAX];
int pa[MAX] , dep[MAX] , val[MAX];
int siz[MAX] , id[MAX] , mm[MAX];
void init(){
    REP(i , 0 , n + 1) id[i] = 0;
    REP(i , 0 , n + 1) v[i].clear();
    REP(i , 0 , n + 1) g[i].clear();
}
void DFS(int now , int fa , int deep){
    pa[now] = fa , dep[now] = deep;
    if(id[now] == 0) siz[id[now] = now] = 1;
    for(auto to : v[now]){
        if(to == fa) continue;
        if(siz[id[now]] + 1 < N){
            g[now].pb(to);
            siz[id[to] = id[now]] ++;
```

```
        }
            DFS(to , now , deep + 1);
        }
}
void build(int now , int v){
    mm[now] = max(v , val[now]);
    for(auto to : g[now]){
        build(to , mm[now]);
    }
}
int query(int a , int b){
    int res = 0;
    while(a != b){
        if(id[a] == id[b]){
            if(dep[a] < dep[b]) swap(a , b);
            res = max(res , val[a]);
            a = pa[a];
        }
        else {
            if(dep[id[a]] < dep[id[b]]) swap(a , b);
            res = max(res , mm[a]);
            a = pa[id[a]];
        }
    }
    return res;
}
int x[MAX][3];
char c[MAX];
int32_t main(){
    scanf("%d" , &t);
    REP(times , 0 , t){
        scanf("%d" , &n);
        init();
        REP(i , 1 , n){
            REP(j , 0 , 3) scanf("%d" , &x[i][j]);
            v[x[i][0]].pb(x[i][1]);
            v[x[i][1]].pb(x[i][0]);
        }
        DFS(1 , 0 , 0);
        REP(i , 1 , n){
            if(dep[x[i][0]] > dep[x[i][1]]) val[x[i
                ][0]] = x[i][2];
            else val[x[i][1]] = x[i][2];
        }
        REP(i , 1 , n + 1){
            if(id[i] == i) build(i , -INF);
        }
        int q , w , tmp;
        while(scanf("%s",c) == 1){
            if(c[0] == 'D') break;
            scanf("%d%d" , &q , &w);
            if(c[0] == 'C'){
                if(dep[x[q][0]] > dep[x[q][1]]) val[x[q
                    ][0]] = w , tmp = x[q][0];
                else val[x[q][1]] = w , tmp = x[q][1];
                if(tmp == id[tmp]) build(tmp , -INF);
                else build(tmp , mm[pa[tmp]]);
            }
            else if(c[0] == 'Q'){
                printf("%d\n", query(q , w));
            }
        }
    }
    return 0;
}
```

8.2 Dancing Link

```
#define MAX 1050
#define INF 0x3f3f3f3f
struct DLX{
    int n , sz , s[MAX];
    int row[MAX * 100] , col[MAX * 100];
    int l[MAX * 100] , r[MAX * 100] , u[MAX * 100] , d[
        MAX * 100];
    int ans;
    void init(int n){
        this -> n = n;
        ans = INF;
        REP(i , 0 , n + 1){
```

```
            u[i] = d[i] = i;
            l[i] = i - 1;
            r[i] = i + 1;
        }
        r[n] = 0 , l[0] = n;
        sz = n + 1;
        MEM(s , 0);
    }
    void AddRow(int rr , vector<int> sol){
        int tmp = sz;
        for(auto to : sol){
            l[sz] = sz - 1;
            r[sz] = sz + 1;
            d[sz] = to;
            u[sz] = u[to];
            d[u[to]] = sz , u[to] = sz;
            row[sz] = rr , col[sz] = to;
            s[to] ++ , sz ++;
        }
        r[sz - 1] = tmp , l[tmp] = sz - 1;
    }
#define FOR(i , way , to) for(int i = way[to] ; i != to
    ; i = way[i])
    void remove(int c){
        l[r[c]] = l[c];
        r[l[c]] = r[c];
        FOR(i , d , c) FOR(j , r , i){
            u[d[j]] = u[j];
            d[u[j]] = d[j];
            --s[col[j]];
        }
    }
    int restore(int c){
        FOR(i , u , c) FOR(j , l , i){
            ++s[col[j]];
            u[d[j]] = j;
            d[u[j]] = j;
        }
        l[r[c]] = c;
        r[l[c]] = c;
    }
    void DFS(int floor){
        if(r[0] == 0){
            ans = min(ans , floor);
            return;
        }
        if(floor >= ans) return;
        int c = r[0];
        FOR(i , r , 0) if(s[i] < s[c]) c = i;
        remove(c);
        FOR(i , d , c){
            FOR(j , r , i) remove(col[j]);
            DFS(floor + 1);
            FOR(j , l , i) restore(col[j]);
        }
        restore(c);
    }
} solver;
int n , m;
int32_t main(){
    IOS;
    while(cin >> n >> m){
        solver.init(m);
        REP(i , 0 , n){
            int nn , in;
            cin >> nn;
            vector<int> sol;
            REP(j , 0 , nn) cin >> in , sol.pb(in);
            solver.AddRow(i , sol);
        }
        solver.DFS(0);
        if(solver.ans == INF) cout << "No" << endl;
        else cout << solver.ans << endl;
    }
    return 0;
}
```

8.3 Joseph Problem

```
int main() {
```

```
    long long  n, k, i, x = 0, y;
    scanf( "%I64d%I64d", &n, &k );
    for( i = 2; i <= k && i <= n; ++i ) x = ( x + k ) % i
        ;
    for( ; i <= n; ++i ) {
        y = ( i - x - 1 ) / k;
        if( i + y > n ) y = n - i;
        i += y;
        x = ( x + ( y + 1 ) % i * k ) % i;
    }
    printf( "%I64d\n", x + 1 );
    return 0;
}
```

8.4 High Speed Linear Recursion

```
#define MAX 100000
#define INF 0x3f3f3f3f
#define mod 10000
int n , k , x[MAX] , c[MAX];
vector<int> mul(vector<int> a , vector<int> b){
    vector<int> ans(n + n + 1);
    REP(i , 1 , n + 1) REP(j , 1 , n + 1)
        ans[i + j] = (ans[i + j] + (a[i] * b[j])) % mod
            ;
    RREP(i , n + n , n + 1){
        REP(j , 1 , n + 1) ans[i - j] = (ans[i - j] +
            ans[i] * c[j]) % mod;
        ans[i] = 0;
    }
    return ans;
}
vector<int> ppow(vector<int> a , int k){
    if(k == 1) return a;
    if(k % 2 == 0) return     ppow(mul(a , a) , k >> 1)
        ;
    if(k % 2 == 1) return mul(ppow(mul(a , a) , k >> 1)
        , a);
}
int main(){
    IOS;
    while(cin >> n && n){
        REP(i , 1 , n + 1) cin >> x[i];
        REP(i , 1 , n + 1) cin >> c[i];
        vector<int> v(n + n + 1);
        v[1] = 1;
        cin >> k , k ++;
        v = ppow(v , k);
        int ans = 0;
        REP(i , 1 , n + 1) ans = (ans + x[i] * v[i]) %
            mod;
        cout << ans << endl;
    }
    return 0;
}
```

8.5 Segment Max segment sum

```
#define int long long
#define MAX 300900
#define INF 10000000000090LL
#define ls (now << 1)
#define rs (now << 1 | 1)
#define mid ((l + r) >> 1)
int n , m , x[MAX];
class N{
public: int tag , sml , sum , none;
} b[MAX * 4];
void Pull(int now , int l , int r){
    if(l == r){
        if(b[now].tag){
            b[now].sum = b[now].tag;
            b[now].none = 0;
            b[now].sml = b[now].tag;
        }
        else{
            b[now].sum = 0;
            b[now].none = 1;
```

```
            b[now].sml = INF;
        }
    }
    else {
        b[now].sml = min(b[ls].sml , b[rs].sml);
        if(b[now].tag) b[now].sml = min(b[now].sml , b[
            now].tag);

        b[now].sum = b[ls].sum + b[rs].sum;
        b[now].none = b[ls].none + b[rs].none;
        if(b[now].tag) b[now].sum += b[now].tag * b[now
            ].none , b[now].none = 0;
    }
}
void take_tag(int now , int l , int r , int val){
    if(b[now].tag && b[now].tag < val) b[now].tag = 0;
    if(l != r && b[ls].sml < val) take_tag(ls , l , mid
        , val);
    if(l != r && b[rs].sml < val) take_tag(rs , mid + 1
        , r , val);
    Pull(now , l , r);
}
void Build(int now , int l , int r){
    b[now].none = 0;
    if(l == r) b[now].tag = b[now].sml = b[now].sum = x
        [l];
    else {
        Build(ls , l , mid) , Build(rs , mid + 1 , r);
        Pull(now , l , r);
    }
}
void update(int now , int l , int r , int ql , int qr ,
    int val){
    if(b[now].tag >= val) return ;
    if(ql <= l && r <= qr){
        take_tag(now , l , r , val);
        b[now].tag = val;
        Pull(now , l , r);
    }
    else{
        if(qr <= mid) update(ls , l , mid , ql , qr ,
            val);
        else if(mid + 1 <= ql) update(rs , mid + 1 , r
            , ql , qr , val);
        else update(ls , l , mid , ql , qr , val) ,
            update(rs , mid + 1 , r , ql , qr , val);
        Pull(now , l , r);
    }
}
PII query(int now , int l , int r , int ql , int qr){
    if(ql <= l && r <= qr) return mp(b[now].sum , b[now
        ].none);
    else {
        PII ans = mp(0 , 0);
        if(qr <= mid) ans = query(ls , l , mid , ql ,
            qr);
        else if(mid + 1 <= ql) ans = query(rs , mid + 1
            , r , ql , qr);
        else {
            PII a = query(ls , l , mid , ql , qr);
            PII b = query(rs , mid + 1 , r , ql , qr);
            ans = mp(a.A + b.A , a.B + b.B);
        }
        if(b[now].tag != 0) ans.A += ans.B * b[now].tag
            , ans.B = 0;
        return ans;
    }
}
int32_t main(){
    IOS;
    cin >> n >> m;
    REP(i , 1 , n + 1) cin >> x[i];
    Build(1 , 1 , n);
    REP(i , 1 , m + 1){
        int ty , l , r , v;
        cin >> ty;
        if(ty == 1){
            cin >> l >> r >> v;
            update(1 , 1 , n , l , r , v);
        }
        if(ty == 2){
            cin >> l >> r;
```

```
            cout << query(1 , 1 , n , l , r).A << endl;
        }
    }
    return 0;
}
```

8.6 Primitive root

```
#define int int_fast64_t
int n;
int ppow(int a , int k , int mod){
    if(k == 0) return 1;
    if(k % 2 == 0) return ppow(a * a % mod , k >> 1 ,
        mod);
    if(k % 2 == 1) return ppow(a * a % mod , k >> 1 ,
        mod) * a % mod;
}
int32_t main(){
    IOS;
    while(cin >> n){
        if(n == 2){
            cout << 1 << endl;
            continue;
        }
        vector<int> sol;
        int val = n - 1;
        REP(i , 2 , INF){
            if(i * i > val) break;
            else if(val % i == 0){
                sol.pb(i);
                while(val % i == 0) val /= i;
            }
        }
        if(val != 1) sol.pb(val);
        int ans;
        REP(i , 2 , INF){
            int ok = 1;
            for(auto to : sol){
                if(ppow(i , (n - 1) / to , n) == 1){
                    ok = 0;
                    break;
                }
            }
            if(ok){
                ans = i;
                break;
            }
        }
        cout << ans << endl;
    }
    return 0;
}
```

8.7 Same remainder Equation

```
#define INF 0x3f3f3f3f
void extgcd(long long a , long long b , long long &d ,
    long long &x , long long &y){
    if(b == 0) d = a , x = 1 , y = 0;
    else extgcd(b , a % b , d , y , x) , y -= (a / b) *
        x;
}
long long n;
vector<long long> v , m;
int main(){
    while(cin >> n){
        v.clear() , m.clear();
        long long ans , mod , d , x , y;
        REP(i , 0 , n) cin >> mod >> ans , m.pb(mod) ,
            v.pb(ans);
        mod = m[0] , ans = v[0];
        REP(i , 1 , n){
            long long res = ((v[i] - ans) % m[i] + m[i
                ]) % m[i];
            extgcd(mod , m[i] , d , x , y);
            if(res % d != 0){ ans = -1; break; }

            res = (res / d * x % m[i] + m[i]) % m[i];
```

```
            ans = ans + res * mod;
            mod = mod * m[i] / d;
        }
        if(ans == -1) cout << ans << endl;
        else cout << ans % mod << endl;
    }
    return 0;
}
```

8.8 Stone merge

```
#define int long long
#define MAX 50900

int n , x[MAX] , ans = 0;
vector<int> v;
int DFS(int now){
    int val = v[now] + v[now + 1];
    ans += val;
    v.erase(v.begin() + now);
    v.erase(v.begin() + now);
    int id = 0;
    RREP(i , now - 1 , 0) if(v[i] >= val) { id = i + 1;
        break; }
    v.insert(v.begin() + id , val);
    while(id >= 2 && v[id - 2] <= v[id]){
        int dis = v.size() - id;
        DFS(id - 2);
        id = v.size() - dis;
    }
}
int32_t main(){
    IOS;
    cin >> n;
    REP(i , 0 , n) cin >> x[i];
    REP(i , 0 , n){
        v.pb(x[i]);
        while(v.size() >= 3 && v[v.size() - 3] <= v[v.
            size() - 1])
            DFS(v.size() - 3);
    }
    while(v.size() > 1) DFS(v.size() - 2);
    cout << ans << endl;
    return 0;
}
```

8.9 Range modify and query BIT

```
#define int long long
#define MAX 250
#define INF 0x3f3f3f3f
int n , m , k;
int bit[4][MAX][MAX];
void update(int c[MAX][MAX] , int a , int b , int val){
    for(int i = a + 10 ; i < MAX ; i += i & -i)
        for(int j = b + 10 ; j < MAX ; j += j & -j)
            c[i][j] += val;
}
int update(int x , int y , int val){
    update(bit[0] , x , y , val);
    update(bit[1] , x , y , -val * x);
    update(bit[2] , x , y , -val * y);
    update(bit[3] , x , y , val * x * y);
}
void update(int a , int b , int x , int y , int val){
    update(a , b , val);
    update(a , y + 1 , -val);
    update(x + 1 , b , -val);
    update(x + 1 , y + 1 , val);
}
int query(int c[MAX][MAX] , int a , int b){
    int cnt = 0;
    for(int i = a + 10 ; i > 0 ; i -= i & -i)
        for(int j = b + 10 ; j > 0 ; j -= j & -j)
            cnt += c[i][j];
    return cnt;
}
int query(int x , int y){
```

```
    int cnt = 0;
    cnt += query(bit[0] , x , y) * (x + 1) * (y + 1);
    cnt += query(bit[1] , x , y) * (y + 1);
    cnt += query(bit[2] , x , y) * (x + 1);
    cnt += query(bit[3] , x , y);
    return cnt;
}
int query(int a , int b , int x , int y){
    int cnt = 0;
    cnt += query(a - 1 , b - 1);
    cnt -= query(a - 1 , y);
    cnt -= query(x , b - 1);
    cnt += query(x , y);
    return cnt;
}
int32_t main(){
    IOS;
    cin >> n >> m >> k;
    int tmp;
    REP(i , 1 , n + 1) REP(j , 1 , m + 1){
        cin >> tmp;
        update(i , j , i , j , tmp);
    }
    REP(i , 1 , k + 1){
        int a , b , x , y , val , add;
        cin >> a >> b >> x >> y >> val >> add;
        int sum = query(b , a , y , x);
        if(sum < val * (x - a + 1) * (y - b + 1)){
            update(b , a , y , x , add);
        }
    }
    REP(i , 1 , n + 1){
        REP(j , 1 , m + 1) cout << query(i , j , i , j)
            << " ";
        cout << endl;
    }
    return 0;
}
```

8.10 Manhattan Spanning Tree

```
#define edge pair<int , PII>
#define MAX 50090
#define INF 0x3f3f3f3f

int n , sol[MAX];
PII x[MAX];
vector<edge> v;
class djs{
public:
    int x[MAX];
    void init(){ REP(i , 0 , MAX) x[i] = i; }
    int Find(int now){ return x[now] == now ? now : x[
        now] = Find(x[now]); }
    void Union(int a , int b){ x[Find(a)] = Find(b); }
    int operator[](int now){ return Find(now); }
} ds;
PII bit[MAX];
void update(int from , int val , int id){
    for(int i = from ; i < MAX ; i += i & -i)
        bit[i] = max(bit[i] , mp(val , id));
}
int query(int from){
    PII res = bit[from];
    for(int i = from ; i > 0 ; i -= i & -i)
        res = max(res , bit[i]);
    return res.B;
}
int cmp(int a , int b){
    return x[a] < x[b];
}
int DIS(int q , int w){
    return abs(x[q].A - x[w].A) + abs(x[q].B - x[w].B);
}
void BuildEdge(){
    vector<int> uni;
    REP(i , 0 , MAX) bit[i] = mp(-INF , -1);
    REP(i , 0 , n) sol[i] = i;
    REP(i , 0 , n) uni.pb(x[i].B - x[i].A);
    sort(ALL(uni));
```

```
    uni.resize(unique(ALL(uni)) - uni.begin());
    sort(sol , sol + n , cmp);
    REP(i , 0 , n){
        int now = sol[i];
        int tmp = x[sol[i]].B - x[sol[i]].A;
        int po = lower_bound(ALL(uni) , tmp) - uni.
            begin() + 1;
        int id = query(po);
        if(id >= 0) v.pb(mp(DIS(id , now) , mp(id , now
            )));
        update(po , x[now].A + x[now].B , now);
    }
}
void Build(){
    BuildEdge();
    REP(i , 0 , n) swap(x[i].A , x[i].B);
    BuildEdge();
    REP(i , 0 , n) x[i].A *= -1;
    BuildEdge();
    REP(i , 0 , n) swap(x[i].A , x[i].B);
    BuildEdge();
}
int solveKruskal(){
    ds.init();
    sort(ALL(v));
    int res = 0;
    REP(i , 0 , v.size()){
        int dis = v[i].A;
        PII tmp = v[i].B;
        if(ds[tmp.A] != ds[tmp.B]){
            ds.Union(tmp.A , tmp.B);
            res += dis;
        }
    }
    return res;
}
int32_t main(){
    IOS;
    cin >> n;
    REP(i , 0 , n) cin >> x[i].A >> x[i].B;
    Build();
    int ans = solveKruskal();
    cout << ans << endl;
    return 0;
}
```

8.11 Integer Split

```
#define MAX 50900
#define mod 1000000007LL
int n , dp[MAX];
int32_t main(){
    dp[0] = 1;
    REP(i , 1 , MAX){
        REP(j , 1 , MAX){
            int tmp = j * (j * 3 - 1) / 2;
            if(tmp > i) break;
            else if(j % 2 == 1) dp[i] = (dp[i] + dp[i -
                tmp]) % mod;
            else if(j % 2 == 0) dp[i] = (dp[i] - dp[i -
                tmp] + mod) % mod;
        }
        REP(j , 1 , MAX){
            int tmp = j * (j * 3 + 1) / 2;
            if(tmp > i) break;
            else if(j % 2 == 1) dp[i] = (dp[i] + dp[i -
                tmp]) % mod;
            else if(j % 2 == 0) dp[i] = (dp[i] - dp[i -
                tmp] + mod) % mod;
        }
    }
    cin >> n;
    cout << dp[n] << endl;
    return 0;
}
```

8.12 K Cover Tree

```
#define MAX 100090
#define INF 0x3f3f3f3f
int n , k , dp[MAX] , ans;
vector<int> v[MAX];
void DFS(int now , int fa){
    if(v[now].size() == 1 && v[now][0] == fa)
        return dp[now] = -1 , void();
    int sml = INF , big = -INF;
    for(auto to : v[now]) if(to != fa){
        DFS(to , now);
        sml = min(sml , dp[to]);
        big = max(big , dp[to]);
    }
    if(sml == -k) dp[now] = k , ans ++;
    else if(big - 1 >= abs(sml)) dp[now] = big - 1;
    else dp[now] = sml - 1;
}
int32_t main(){
    IOS;
    cin >> n >> k;
    REP(i , 2 , n + 1){
        int a , b; cin >> a >> b;
        v[a].pb(b); v[b].pb(a);
    }
    if(k == 0) cout << n << endl;
    else {
        DFS(0 , 0) , ans += dp[0] < 0;
        cout << ans << endl;
    }
    return 0;
}
```

8.13 Maximum M Sum

```
--------------------Greedy--------------------
#define int long long
#define MAX 50900
#define INF 0x3f3f3f3f
int n , m , fr[MAX] , ba[MAX];
int v[MAX] , idx = 1;
set<PII> cc;
void erase(int id){
    if(id == 0) return;
    int f = fr[id] , b = ba[id];
    ba[fr[id]] = b , fr[ba[id]] = f;
    cc.erase(mp(abs(v[id]) , id));
}
int32_t main(){
    cin >> n >> m;
    int sum = 0 , pos = 0 , ans = 0;
    REP(i , 0 , n){
        int tmp; cin >> tmp;
        if(tmp == 0) continue;
        if((tmp >= 0 && sum >= 0) || (tmp <= 0 && sum
            <= 0)){
            sum += tmp;
        }
        else {
            if(sum > 0) ans += sum , pos ++;
            v[idx ++] = sum , sum = tmp;
        }
    }
    if(sum) v[idx ++] = sum;
    if(sum > 0) ans += sum , pos ++;
    REP(i , 0 , idx){
        fr[i + 1] = i;
        ba[i] = i + 1;
        if(i) cc.insert(mp(abs(v[i]) , i));
    } ba[idx - 1] = 0;
    while(pos > m){
        auto tmp = cc.begin();
        int val = (*tmp).A , id = (*tmp).B;
        cc.erase(tmp);
        if(v[id] < 0 && (fr[id] == 0 || ba[id] == 0))
            continue;
        if(v[id] == 0) continue;
        ans -= val , pos --;
        v[id] = v[fr[id]] + v[id] + v[ba[id]];
        cc.insert(mp(abs(v[id]) , id));
        erase(fr[id]) , erase(ba[id]);
```

```
    }
    cout << ans << endl;
    return 0;
}
--------------------Aliens--------------------
#define int int_fast64_t
#define MAX 2000090
#define INF 0x3f3f3f3f
int n , k , x[MAX];
PII dp[MAX] , rd[MAX]; // max value , times , can be
    buy , times
int judge(int now){
    dp[1] = mp(0 , 0) , rd[1] = mp(-x[1] , 0);
    REP(i , 2 , n + 1){
        dp[i] = max(dp[i - 1] , mp(rd[i - 1].A + x[i] -
            now , rd[i - 1].B + 1));
        rd[i] = max(rd[i - 1] , mp(dp[i - 1].A - x[i]
                , dp[i - 1].B));
    }
    return dp[n].B;
}
int32_t main(){
    IOS;
    cin >> n >> k;
    n ++;
    REP(i , 2 , n + 2) cin >> x[i];
    REP(i , 1 , n + 1) x[i] += x[i - 1];
    if(judge(0) <= k) cout << dp[n].A << endl;
    else {
        int l = 0 , r = 1000000000000LL;
        while(r - l > 1){
            int mid = l + ((r - l) >> 1) , res = judge(
                mid);
            if(res == k) return cout << dp[n].A + dp[n
                ].B * mid << endl , 0;
            else if(res < k) r = mid;
            else if(res > k) l = mid;
        }
        judge(l);
        cout << dp[n].A + k * l << endl;
    }
    return 0;
}
```

8.14 Sigma Problem

```
//problem->for(int i = 1 ; i <= n ; i ++) ans += pow(i
    , k);
#define int long long
#define MAX 2020
#define INF 0x3f3f3f3f
#define mod 1000000007LL

int b[MAX] , c[MAX][MAX] , ni[MAX];
int ppow(int a , int k){
    if(k == 0) return 1;
    if(k % 2 == 0) return ppow(a * a % mod , k >> 1);
    if(k % 2 == 1) return ppow(a * a % mod , k >> 1) *
        a % mod;
}
void solveinit(){
    REP(i , 0 , MAX){
        REP(j , 0 , i + 1){
            if(j == 0 || j == i) c[i][j] = 1;
            else c[i][j] = (c[i - 1][j] + c[i - 1][j -
                1]) % mod;
        }
    }
    REP(i , 1 , MAX) ni[i] = ppow(i , mod - 2);
    b[0] = 1;
    REP(i , 1 , MAX){
        REP(j , 0 , i) b[i] = (b[i] + c[i + 1][j] * b[j
            ]) % mod;
        b[i] = b[i] * ni[i + 1] % mod;
        b[i] = mod - b[i];
    }
}
int t , n , k;
int32_t main(){
    solveinit();
```

```cpp
        cin >> t;
        REP(times , 0 , t){
            cin >> n >> k;
            n %= mod;
            int ans = 0 , np = 1;
            REP(i , 1 , k + 2){
                np = np * (n + 1) % mod;
                ans = (ans + c[k + 1][i] * np % mod * b[k +
                    1 - i] % mod) % mod;
            }
            ans = (ans * ni[k + 1]) % mod;
            cout << ans << endl;
        }
        return 0;
}
```

8.15 Range Color Online

```cpp
#include <bits/stdc++.h>
using namespace std;
const int MAX_N = 1e5 + 6;
const int MAX_M = 3e5 + 6;
struct Node {
  int lc,rc;
  int val;
  void give_val(int _lc,int _rc,int _val) {
    lc=_lc;rc=_rc;val = _val;
  }
} node[530*MAX_N];
int bit_root[MAX_N],root[MAX_N];
int node_cnt;
int getNode(int id) {
  int ret = ++node_cnt;
  node[ret] = node[id];
  return ret;
}
void pull(int id) {
  node[id].val = node[node[id].lc].val + node[node[id].
      rc].val;
}
void init(int id,int L,int R) {
  if (L==R) {
    node[id].give_val(0,0,0); return;
  }
  node[id].give_val(++node_cnt,++node_cnt,0);
  int mid=(L+R)>>1;
  init(node[id].lc,L,mid);
  init(node[id].rc,mid+1,R);
  return;
}
void modify(int old_id,int new_id,int L,int R,int pos,
    int val) {
  if (L==R) {
    node[new_id].val += val;return;
  }
  int mid=(L+R)>>1;
  if (pos <= mid) {
    node[new_id].lc = getNode(node[old_id].lc);
    modify(node[old_id].lc,node[new_id].lc,L,mid,pos,
        val);
  }
  else {
    node[new_id].rc = getNode(node[old_id].rc);
    modify(node[old_id].rc,node[new_id].rc,mid+1,R,pos,
        val);
  }
  pull(new_id);
  return;
}
int query(int id,int L,int R,int l,int r) {
  if (l<=L && R<=r) return node[id].val;
  int mid=(L+R)>>1;
  if (mid + 1 > r) return query(node[id].lc,L,mid,l,r);
  else if (l > mid) return query(node[id].rc,mid+1,R,l,
      r);
  return query(node[id].lc,L,mid,l,r) + query(node[id].
      rc,mid+1,R,l,r);
}
set<int> st[MAX_M];
int last[MAX_N];
```

```cpp
int s[MAX_N];
int n,q;
typedef long long LL;
void modify_bit(int L,int R,int pos,int val) {
  for (int i=L;n>=i;i+=(i&(-i))) {
    modify(bit_root[i],bit_root[i],1,n,pos,val);
  }
  if (R==n) return;
  for (int i=R+1;n>=i;i+=(i&(-i))) {
    modify(bit_root[i],bit_root[i],1,n,pos,-val);
  }
}
int query_bit(int C,int L,int R) {
  int ret=0;
  for (int i=C;i>0;i-=(i&(-i))){
    ret += query(bit_root[i],1,n,L,R);
  }
  return ret;
}
int main (){
  int k,m;
  scanf("%d %d %d %d",&n,&q,&m,&k);
  node_cnt = 0; root[0] = ++node_cnt; init(root[0],1,n)
      ;
  map<int,int> mp;
  for (int i=1;n>=i;i++) {
    bit_root[i] = getNode(root[0]);
  }
  int id=1;
  for (int i=1;n>=i;i++) {
    int x; scanf("%d",&x);
    int ret=0; auto iter=mp.find(x);
    if (iter == mp.end()) {
      mp.insert(make_pair(x,id));
      ret=id; id++;
    }
    else {
      ret=iter->second;
    }
    root[i] = getNode(root[i-1]);
    if (last[ret] == 0) {
      modify(root[i-1],root[i],1,n,i,1);
    }
    else {
      modify(root[i-1],root[i],1,n,i,1);
      modify(root[i],root[i],1,n,last[ret],-1);
    }
    last[ret] = i; st[ret].insert(i); s[i] = ret;
  }
  int pre_ans=0;
  for (int i=1;q>=i;i++) {
    int a,b,c;
    scanf("%d %d %d",&a,&b,&c);
    if (a==0) {
      //one base !!! query(b,c)
      pre_ans = query(root[c],1,n,b,c);
      pre_ans += query_bit(c,b,c);
      printf("%d\n",pre_ans);
    }
    else {
      //one base!!! a[b] = c
      c = (LL(pre_ans)*c%m)*k;
      if (mp[c] == s[b]) continue;
      int del=s[b]; auto iter=st[del].find(b);
      int ed = n+1; ++iter;
      if (iter != st[del].end()) ed = *(iter);
      //b ~ ed - 1
      modify_bit(b,ed-1,b,-1);
      iter--;
      if (iter != st[del].begin()) {
        int start=*(--iter);
        modify_bit(b,ed-1,start,1);
      }
      st[del].erase(st[del].find(b));
      //finish delete

      //now let's add
      int ret=0;
      auto iter3=mp.find(c);
      if (iter3 == mp.end()) {
        mp.insert(make_pair(c,id));
        ret=id;
```

```
        id++;
      }
      else if (iter3->second == 0) {
        mp[c] = id;
        ret=id;
        id++;
      }
      else {
        ret=iter3->second;
      }
      auto iter4 = st[ret].insert(b).first;
      ed = n+1;
      ++iter4;
      if (iter4 != st[ret].end()) {
        ed = *(iter4);
      }
      --iter4;
      modify_bit(b,ed-1,b,1);
      if (iter4 != st[ret].begin()) {
        int start = *(--iter4);
        modify_bit(b,ed-1,start,-1);
      }
      s[b] = ret;
      st[ret].insert(b);
    }
  }
}
```

8.16 Minimum Enclosing Cycle

```
pdd arr[MAX];
pdd cen;
double r;
inline double dis(pdd a,pdd b){ return hypot(a.X-b.X,a.
    Y-b.Y);  }
int n,m;
inline double sq(double x){return x*x;}
pdd external(pdd p1,pdd p2,pdd p3){
  double a1=p1.X-p2.X,a2=p1.X-p3.X;
  double b1=p1.Y-p2.Y,b2=p1.Y-p3.Y;
  double c1=( sq(p1.X)-sq(p2.X)+sq(p1.Y)-sq(p2.Y) )/2;
  double c2=( sq(p1.X)-sq(p3.X)+sq(p1.Y)-sq(p3.Y) )/2;
  double dd=a1*b2-a2*b1;
  return pdd( (c1*b2-c2*b1)/dd , (a1*c2-a2*c1)/dd );
}
int main(){
  IOS
  srand(time(0));
  while(cin>>n>>m){
    if(n+m==0)  return 0;
    for(int i=0;i<m;i++){
      cin>>arr[i].X>>arr[i].Y;
    }
    random_shuffle(arr,arr+m);
    r=0;
    for(int i=0;i<m;i++){
      if(dis(cen,arr[i])>r){
        cen=arr[i]; r=0;
        for(int j=0;j<i;j++){
          if(dis(cen,arr[j])>r){
            cen=pdd( (arr[i].X+arr[j].X)/2 , (arr[i].Y+
                arr[j].Y)/2 );
            r=dis(cen,arr[j]);
            for(int k=0;k<j;k++){
              if(dis(cen,arr[k])>r){
                cen=external(arr[i],arr[j],arr[k]);
                r=dis(cen,arr[j]);
              }
            }
          }
        }
      }
    }
    cout<<stp<<r<< '\n';
  }
  return 0;
}
```

8.17 Triangle

```
PII p[MAX];
int n , idx[MAX] , pos[MAX];
long long wnt;
vector<PII> v;

inline PII operator + (PII x , PII y){ return mp(x.A +
    y.A , x.B + y.B); }
inline PII operator - (PII x , PII y){ return mp(x.A -
    y.A , x.B - y.B); }
inline long long cross(PII x , PII y){ return 1ll * x.A
     * y.B - 1ll * x.B * y.A; }
inline long long calcArea(PII x , PII y , PII z){
    long long val = abs(cross(y - x , z - x));
    return val;
}
inline int cmp1(PII x , PII y){
    x = p[x.B] - p[x.A];
    y = p[y.B] - p[y.A];
    return cross(x , y) > 0;
}
int32_t main(){
    IOS;
    cin >> n >> wnt , wnt += wnt;
    REP(i , 1 , n + 1) cin >> p[i].A >> p[i].B;
    sort(p + 1 , p + 1 + n);
    REP(i , 1 , n + 1) idx[i] = i , pos[i] = i;
    REP(i , 1 , n + 1) REP(j , i + 1 , n + 1) v.pb(mp(i
        , j));
    sort(ALL(v) , cmp1);

    for(auto line : v){
        int fr = pos[line.A] , ba = pos[line.B] , now;
        if(fr > ba) swap(fr , ba);
        now = fr;
        RREP(i , 10 , 0){
            int to = now - (1 << i);
            if(to >= 1 && calcArea(p[idx[fr]] , p[idx[
                ba]] , p[idx[to]]) <= wnt) now = to;
        }
        now = ba;
        RREP(i , 10 , 0){
            int to = now + (1 << i);
            if(to <= n && calcArea(p[idx[fr]] , p[idx[
                ba]] , p[idx[to]]) <= wnt) now = to;
        }
        swap(idx[fr] , idx[ba]) , swap(pos[line.A] ,
            pos[line.B]);
    }
    cout << "No" << endl;
    return 0;
}
```