Contents

1 Basic

1.1 .vimrc

```
se nu
syntax on
se t_Co=256
se autoindent
se ts=4 sts=0 expandtab smarttab
se ruler

inoremap {<ENTER> {}<LEFT><ENTER><ENTER><UP><TAB>
```

1.2 defalut code

```
#include <bits/stdc++.h>
using namespace std;
//debug
#ifdef grief
#define debug(...) do{\
  fprintf(stderr , "%s - %d : (%s) = " ,
      __PRETTY_FUNCTION__ , __LINE__ , #__VA_ARGS__ );\
  _DO(__VA_ARGS__);\
}while(0)
template<typename I>        void _DO(I&&x){ cerr<<x<<
    endl;  }
template<typename I,typename...T> void _DO(I&&x,T&&...
    tail){  cerr<<x<<" , "; _DO(tail...); }
template<typename _a,typename _b>   ostream& operator
    << (ostream &_s,const pair<_a,_b> &_p)
{ return _s<<'('<<_p.first<<','<<_p.second<<')';  }
#define IOS
#else
#define debug(...)
#define IOS ios_base::sync_with_stdio(0); cin.tie(0);
#endif
//type
typedef long long LL;
typedef pair<int,int> pii;
typedef pair<LL,LL> pll;
typedef priority_queue<pll,vector<pll>,greater<pll> >
    heap;
//macro
#define SZ(x) ((LL)(x).size())
#define ALL(x) (x).begin(),(x).end()
#define F first
#define S second
#define pb push_back
#define eb emplace_back
#define stp setprecision(18)<<fixed
const LL INF=0x3f3f3f3f3f3f3f3f;
const int NF=0x3f3f3f3f;
const LL MAX=1e5+5,Mlg=__lg(MAX)+2;
const LL MOD=1e9+7;
// ready~ go!
// let's coding and have fun!
// I can't solve this problem!
int main(){
  IOS
  return 0;
}
```

1.3 fasterIO

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

const int N = 100006;

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); // □0 + 1□ □ □ □
name.order_of_key(1);  //if(□ □ □ □ □i) return □ □
name.insert(2);
name.delete(3);
name.split(v, b); /// a □□ < v □□□□□ b
name.join(another TREE); // □□□□□□
```

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 Mo's algo with modification

```
#include <iostream>
#include <stdio.h>
#include <algorithm>
#include <cmath>
#include <cstring>
using namespace std;

const int MAX_N = 5e4 + 6;
const int MAX_M = 1e6 + 6;

int cnt[MAX_M];
int s[MAX_N];

struct Query {
    int L,R,Lid,Rid,T,id;
    void give_val(int _L,int _R,int _Lid,int _Rid,int
        _T,int _id) {
        L = _L; R = _R; Lid = _Lid;
        Rid = _Rid; T = _T; id = _id;
    }
    bool operator<(const Query &q2) {
        if (Lid != q2.Lid) return Lid < q2.Lid;
        if (Rid != q2.Rid) return Rid < q2.Rid;
        return T < q2.T;
    }
} query[MAX_N];

struct Modify {
    int pos,ori_val,after_val;
    void give_val(int _pos,int _ori_val,int _after_val)
         {
        pos = _pos;
        ori_val = _ori_val;
        after_val = _after_val;
    }
} modify[MAX_N];
int ans[MAX_N];
int cur_ans;
void add(int x) {}
void sub(int x) {}
void addTime(int T,int L,int R) {
    if (L <= modify[T].pos && modify[T].pos <= R) {
        sub(s[modify[T].pos]);
        add(modify[T].after_val);
    }
    s[modify[T].pos] = modify[T].after_val;
}
void subTime(int T,int L,int R) {
    if (L <= modify[T].pos && modify[T].pos <= R) {
        sub(s[modify[T].pos]);
        add(modify[T].ori_val);
    }
    s[modify[T].pos] = modify[T].ori_val;
}

int T=-1;
int qid=0;
int B = pow(max(n,m),2.0/3.0);
if (B<=0) B=1;
sort(query+1,query+qid+1);
int L=1,R=0;
for (int i=1;qid >= i;i++) {
    if (query[i].L > query[i].R) {
        ans[query[i].id] = 0;
        continue;
    }
```

```
    while (query[i].R > R) add(s[++R]);
    while (query[i].L < L) add(s[--L]);
    while (query[i].R < R) sub(s[R--]);
    while (query[i].L > L) sub(s[L++]);
    while (query[i].T > T) addTime(++T,L,R);
    while (query[i].T < T) subTime(T--,L,R);
    ans[query[i].id] = cur_ans;
}
```

2.3 Persistent treap

```
#include <iostream>
#include <stdio.h>
#include <ctime>
#include <cstdlib>
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];
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]));
```

2.4 Weighted interval domination

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <algorithm>
#include <queue>
using namespace std;

typedef pair<int,int> pii;

int l[613456],r[613456],w[613456];

vector<pii> sagiri[613456];

int dp[812464];

const int INF = (1<<29);

struct Node
{
    Node *lc,*rc;
    int mn;
    Node():lc(NULL),rc(NULL),mn(INF){}
    void pull()
    {
        mn = min(lc->mn,rc->mn);
    }
};

Node* Build(int L,int R)
{
    Node* node = new Node();
    if (L==R) return node;
    int mid=(L+R)>>1;
    node->lc=Build(L,mid);
    node->rc=Build(mid+1,R);
    return node;
}

void modify(Node* node,int L,int R,int pos,int val)
{
    if (L==R)
    {
        node->mn = val;
        return;
    }
    int mid=(L+R)>>1;
    if (pos <= mid) modify(node->lc,L,mid,pos,val);
    else modify(node->rc,mid+1,R,pos,val);
    node->pull();
    return;
}

int query(Node* node,int L,int R,int l,int r)
{
    if (l>R || L>r) return INF;
    else if (l<=L && R<=r) return node->mn;
    int mid=(L+R)>>1;
    return min(query(node->lc,L,mid,l,r),query(node->rc
        ,mid+1,R,l,r));
}

int main ()
{
    int T;
    scanf("%d",&T);
    while (T--)
    {
        int n;
        scanf("%d",&n);
        vector<int> v;
        for (int i=1;n>=i;i++)
        {
            scanf("%d %d %d",&l[i],&r[i],&w[i]);
            //++l[i];
            v.push_back(l[i]);
            v.push_back(r[i]);
```

```
        }
        sort(v.begin(),v.end());
        v.resize(unique(v.begin(),v.end()) - v.begin())
            ;
        for (int i=1;n>=i;i++)
        {
            l[i] = lower_bound(v.begin(),v.end(),l[i])
                - v.begin() + 1;
            r[i] = lower_bound(v.begin(),v.end(),r[i])
                - v.begin() + 1;
        }
        int nn = v.size();
        for (int i=1;n>=i;i++)
        {
            sagiri[ r[i] ].push_back( make_pair(l[i],w[
                i]) );
        }
        priority_queue<pii,vector<pii>,greater<pii> >
            pq;
        Node* root = Build(1,nn);
        for (int i=1;nn>=i;i++)
        {
            if (sagiri[i].size() == 0)
            {
                dp[i] = dp[i-1];
            }
            else
            {
                for (pii p:sagiri[i])
                {
                    int l = p.first;
                    int r = i;
                    int w = p.second;
                    while (pq.size() && pq.top().second
                         <= l) pq.pop();
                    pq.push(make_pair(query(root,1,nn,l
                        ,r-1) + w,r));
                }
                dp[i] = pq.top().first;
            }
            modify(root,1,nn,i,dp[i]);
            //cout << "i = "<<i<<", dp = "<<dp[i]<<endl
                ;
        }
        for (int i=1;n>=i;i++)
        {
            sagiri[ r[i] ].clear();
        }
        printf("%d\n",dp[nn]);
    }
}
```

2.5 Link Cut Tree

```
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;
  }
  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 up down

```
#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;
```

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){}
};

#define F first
#define S second

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;
}
```

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];
int 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;
  }
  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;
  }
  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);
  }
  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

```
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++){
        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

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

typedef long long LL;
const int N = 1002;
const LL INF = (1LL<<41);
const LL INF2 = (1LL<<30);

LL adj[N][N];
LL dp[N][N];  //dp[i][j] --> from point 0 to point j
    threw i roads

int main()
{
    int n;
    scanf("%d",&n);
    for (int i=0;n+1>=i;i++)
    {
        for (int j=0;n+1>=j;j++)
        {
            dp[i][j] = INF;
        }
    }
    for (int i=1;n>=i;i++)
    {
        for (int j=1;n>=j;j++)
        {
            scanf("%lld",&adj[i][j]);
            if (!adj[i][j]) adj[i][j] = INF;
        }
    }
    for (int i=0;n>=i;i++)
    {
        adj[0][i] = 0;
        adj[i][0] = INF;
        if (i) dp[1][i] = 0;
    }
    for (int i=2;n+1>=i;i++)
    {
        //dp[i][j] --> i roads from 0 to j
        for (int j=0;n>=j;j++)
        {
            for (int k=0;n>=k;k++)
            {
                dp[i][j] = min(dp[i][j],dp[i-1][k] +
                    adj[k][j]);
            }
        }
    }
    LL ansup = INF2, ansdown = 1;
    for (int i=1;n>=i;i++)
    {
        //go threw all possible i's
        LL tmpup = 0,tmpdown = 1;
        if (dp[n+1][i] == INF) continue;
        for (int j=1;n>=j;j++)
        {
            if (dp[j][i] == INF) continue;
            LL up = dp[n+1][i] - dp[j][i];
            LL down = n - j + 1;
            //up/down > tmpup/tmpdown
            if (up >= 0)
            {
                if (up * tmpdown > down * tmpup)
                {
                    tmpup = up;
                    tmpdown = down;
                }
            }
        }
        if (tmpup == 0) continue;
        //ansup/ansdown > tmpup / tmpdown
        if (ansup * tmpdown > ansdown * tmpup)
```

```
        {
            ansup = tmpup;
            ansdown = tmpdown;
        }
    }
    LL gcd = __gcd(ansup,ansdown);
    ansup /= gcd;
    ansdown /= gcd;
    if (ansup == INF2) puts("-1 -1");
    else printf("%lld %lld\n",ansup,ansdown);
}

//adj[i][j] --> cost from i to j
```

5.8 Heavy-Light decomposition

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
using namespace std;
using namespace __gnu_pbds;
#define REP(i,j,k)      for(int i = j ; i < k ; ++i)
#define RREP(i,j,k)     for(int i = j ; i >=k ; --i)
#define A     first
#define B     second
#define mp    make_pair
#define pb    push_back
#define PII pair<int , int>
#define MEM(i,j)    memset(i , j , sizeof i)
#define ALL(i)      i.begin() , i.end()
#define DBGG(i,j)      cout << i << " " << j << endl
#define DB4(i,j,k,l)   cout << i << " " << j << " " << k
     << " " << l << endl
#define IOS cin.tie(0) , cout.sync_with_stdio(0)
#define endl "\n"
//
    /--------------------------------------------------

#define MAX 100900
#define INF 0x3f3f3f3f
#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);
    }
}
node pull(node a , node b){ return (node){max(a.big , b
    .big) , min(a.sml , b.sml)}; }
int update(int now , int l , int r , int k , int val){
    if(l == r) st[now].big = st[now].sml = val;
    else{
        if(k <= mid + 0) update(ls , l , mid + 0 , k ,
            val);
        if(mid + 1 <= k) update(rs , mid + 1 , r , k ,
            val);
        st[now] = pull(st[ls] , st[rs]);
    }
}
node query(int now , int l , int r , int ql , int qr){
    if(ql <= l && r <= qr) return st[now];
    else if(qr <= mid + 0) return query(ls , l , mid +
        0 , ql , qr);
    else if(mid + 1 <= ql) return query(rs , mid + 1 ,
        r , ql , qr);
    return pull(query(ls , l , mid + 0 , ql , qr) ,
        query(rs , mid + 1 , r , ql , qr));
}
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();
}
int32_t main(){
    IOS;
    while(cin >> n){
        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();
        cin >> m;
        REP(i , 1 , m + 1){
            int a , b;
            cin >> a >> b;
            query(a , b);
        }
    }
    return 0;
}
```

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;
    }
  }
  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];
    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

```
#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);
pre_fft();
fft(n,b);
for (int i=0;n>i;i++) {
  c[i] = a[i]*b[i];
}
pre_fft();
fft(n,c,1);
*/
```

6.3 FWT

```
struct FWT{
    int n , a[FWT_MAX] , b[FWT_MAX];
    void Fwt(int x[FWT_MAX] , int l , int r){
        if(l == r) return;
        int m = ((l + r) >> 1) + 1;
        Fwt(x , l , m - 1) , Fwt(x , m , r);
        REP(i , 0 , m - l){
            int q = x[l + i] , w = x[m + i];
            x[l + i] = (q + w) % FWT_MOD; // and
            x[m + i] = w % FWT_MOD; // and

            x[l + i] = (q + w) % FWT_MOD;//xor
            x[m + i] = (q - w + FWT_MOD) % FWT_MOD;//
                xor

            x[l + i] = q % FWT_MOD;//or
            x[m + i] = (q + w) % FWT_MOD;//or
        }
    }
     void IFwt(int x[FWT_MAX] , int l , int r){
        if(l == r) return;
        int m = ((l + r) >> 1) + 1;
        REP(i , 0 , m - l){
            int q = x[l + i] , w = x[m + i];
            x[l + i] = (q - w + FWT_MOD) % FWT_MOD;//
                and
            x[m + i] = w % FWT_MOD;//and

            x[l + i] = (q + w) * FWT_INV2 % FWT_MOD;//
                xor
            x[m + i] = (q - w + FWT_MOD) * FWT_INV2 %
                FWT_MOD;//xor

            x[l + i] = q % FWT_MOD;//or
            x[m + i] = (w - q + FWT_MOD) % FWT_MOD;//or
        }
        IFwt(x , l , m - 1) , IFwt(x , m , r);
    }
    void solve(){
        Fwt(a , 0 , n - 1);
        Fwt(b , 0 , n - 1);
        REP(i , 0 , n) a[i] = a[i] * b[i] % FWT_MOD;
        IFwt(a , 0 , n - 1);
    }
};
```

6.4 Gauss

```
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;
```

6.5 linear prime

```
int dv[MAX] , p[MAX] , po = 0;
void getprime(){
    REP(i , 2 , MAX){
        if(dv[i] == 0) dv[i] = i , p[po ++] = i;
        REP(j , 0 , po){
            if(i * p[j] >= MAX) break;
            dv[i * p[j]] = p[j];
            if(i % p[j] == 0) break;
        }
    }
}
```

6.6 Miller Robin

```
#include <iostream>
#include <stdio.h>
#include <algorithm>
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;
}

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;
    }*/
    LL as[3]={2,7,61};
    for (int i=0;3>i;i++) {
        if (miller_robin(n,as[i]) == COMPOSITE) return
            COMPOSITE;
    }
    return PRIME;
}

int main () {
    LL n;
    while (scanf("%lld",&n) != EOF) {
        if (isPrime(n) == PRIME) puts("1");  //prime
        else puts("0");
    }
}
```

6.7 Pollard Rho

```
//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)
    {
        //cout<<"i = "<<i<<endl;
        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)

LL solve(LL n)
{
    if (isPrime(n)) return n;
    if (!(n&1)) return 2;
    a=myRnd()%n;
    if (!a) a=1;
    c=myRnd()%n;
    //cout<<"a = "<<a<<" , c = "<<c<<endl;
    while (c == 0 || c==2) c=myRnd()%n;
    LL start = myRnd()%n;
    //cout<<"start = "<<start<<" , ";
    LL s1=doo(start,n);//cout<<"s1 = "<<s1<<endl;
    LL s2=doo(s1,n);
    //cout<<"n = "<<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;
        }
        //cout<<"s1 = "<<s1<<" , s2 = "<<s2<<endl;
        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 Phi

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);
}
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)
```

7 String

7.1 string tools

```
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 AC automota

```
#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();
        //cout<<"qe = "<<qe<<endl;
        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];

int main ()
{
    ios::sync_with_stdio(0);
    cin.tie(0);
    int n;
    while (cin >> n)
    {
        if (!n) break;
        ac.init();
        for (int i=1;n>=i;i++)
        {
            cin >>s[i];
            ed[i] = ac.insert(s[i],i);
        }
        string t;
        cin >> t;
        ac.Find(t);
        int mx=0;
        for (int i=1;n>=i;i++)
        {
            mx = max(mx,ac.cnt[ ed[i] ]);
        }
        cout << mx <<endl;
        for (int i=1;n>=i;i++)
        {
            if(ac.cnt[ ed[i] ] == mx) cout << s[i] <<
                endl;
        }
    }
}
```

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;
}
```