Algorithms

Algorithms by Yu Dongfeng
First version on November 28, 2015
Latest version on February 13, 2016

Contents

1 **Computational Geometry** **3**
- 1.1 Convex Hull . . . 4
- 1.2 Delaunay Triangulation . . . 5
- 1.3 Dynamic Convex Hull (Set) . . . 9
- 1.4 Dynamic Convex Hull (Treap) . . . 11
- 1.5 Geometry 2D . . . 19
- 1.6 Half-Plane Intersection . . . 23

2 **Data Structures** **27**
- 2.1 Binary Heap . . . 28
- 2.2 Dynamic Sequence . . . 30
- 2.3 Fenwick Tree . . . 34
- 2.4 K-D Tree . . . 35
- 2.5 Link-Cut Tree . . . 37
- 2.6 Pairing Heap . . . 42
- 2.7 Red-Black Tree . . . 45
- 2.8 Self-Adjusting Top Tree . . . 53
- 2.9 Skew Heap . . . 64

3 **Graph Algorithms** **67**
- 3.1 Chordality Test . . . 68
- 3.2 Dominator Tree . . . 69
- 3.3 K Shortest Path . . . 71
- 3.4 Maximal Clique Count . . . 77

3.5 Maximal Planarity Test 78
3.6 Maximum Flow 82
3.7 Maximum Matching 84
3.8 Minimum Cost Maximum Flow 87
3.9 Minimum Spanning Arborescence 89
3.10 Minimum Spanning Tree 91
3.11 Shortest Path 92
3.12 Steiner Tree 94
3.13 Virtual Tree 95

4 Number Theory **99**
4.1 Discrete Logarithm 100
4.2 Integer Factorization (Pollard's Rho Algorithm) 102
4.3 Integer Factorization (Shanks' Square Forms Factorization) 104
4.4 Modular Integer 108
4.5 Mõbius Function 111
4.6 Primality Test 111
4.7 Prime Number 113
4.8 Primitive Root 113
4.9 Sequence 116

5 Numerical Algorithms **119**
5.1 Convolution (Fast Fourier Transform) 120
5.2 Convolution (Karatsuba Algorithm) 121
5.3 Convolution (Number Theoretic Transform) 122
5.4 Fraction 123
5.5 Integer 126
5.6 Linear Programming 133
5.7 Linear System 135
5.8 Matrix 137
5.9 Polynomial Interpolation 138

6 String Algorithms **139**
6.1 Aho-Corasick Automaton 140
6.2 Factor Oracle 141
6.3 Longest Common Substring 142
6.4 Palindromic Tree 143
6.5 String Searching 144
6.6 Suffix Array (DC3 Algorithm) 145
6.7 Suffix Array (Factor Oracle) 148
6.8 Suffix Array (Prefix-Doubling Algorithm) 151
6.9 Suffix Array (Suffix Tree) 152

6.10 Suffix Array (Treap) 155
6.11 Suffix Automaton 159
6.12 Suffix Tree 161

7 Utility Tools **165**
7.1 Checker 166
7.2 Date 166
7.3 Fast Reader 170
7.4 Fast Writer 171
7.5 Number Speller 172

CHAPTER 1

Computational Geometry

1.1 Convex Hull

Convex Hull.hpp (1063 bytes, 36 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct ConvexHull{
    struct point{
        T x,y;
        point(T _x,T _y):
            x(_x),y(_y){
        }
        point operator-(point a){
            return point(x-a.x,y-a.y);
        }
        T operator*(point a){
            return x*a.y-y*a.x;
        }
        int operator<(point a){
            return x==a.x?y<a.y:x<a.x;
        }
    };
    static int check(point a,point b,point c){
        return (a-c)*(b-c)<=0;
    }
    static vector<vector<point> >run(vector<point>a){
        sort(a.begin(),a.end());
        vector<point>u,d;
        for(int i=0;i<a.size();u.push_back(a[i++]))
            while(u.size()>1&&check(a[i],u.back(),u[u.size()-2]))
                u.pop_back();
        for(int i=int(a.size()-1);i>=0;d.push_back(a[i--]))
            while(d.size()>1&&check(a[i],d.back(),d[d.size()-2]))
                d.pop_back();
        vector<vector<point> >r;
        r.push_back(u);
        r.push_back(d);
        return r;
    }
};
```

1.2 Delaunay Triangulation

Delaunay Triangulation.hpp (4889 bytes, 159 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct DelaunayTriangulation{
    const static double E;
    struct poi{
        T x,y;
        poi(T _x=0,T _y=0):
            x(_x),y(_y){
        }
        poi operator-(poi b){
            return poi(x-b.x,y-b.y);
        }
        int operator<(poi b)const{
            if(fabs(x-b.x)<E)
                return y<b.y;
            return x<b.x;
        }
    };
    int n;
    vector<pair<poi,int> >pts;
    vector<vector<int> >egs;
    T det(poi a,poi b){
        return a.x*b.y-a.y*b.x;
    }
    T dot(poi a,poi b){
        return a.x*b.x+a.y*b.y;
    }
    int dir(poi a,poi b,poi c){
        T r=det(c-a,b-a);
        if(r<-E)
            return -1;
        return r>E?1:0;
    }
    int inc(poi a,poi b,poi c,poi d){
        a=a-d;
        b=b-d;
        c=c-d;
        T az=a.x*a.x+a.y*a.y,bz=b.x*b.x+b.y*b.y,cz=c.x*c.x+c.y*c.y;
        return a.x*b.y*cz+b.x*c.y*az+c.x*a.y*bz-a.x*bz*c.y-b.x*a.y*cz-c.x*
    b.y*az>E;
    }
    int crs(poi a,poi b,poi c,poi d){
        return dir(a,b,c)*dir(a,b,d)==-1&&dir(c,d,a)*dir(c,d,b)==-1;
    }
    DelaunayTriangulation():
        n(0),pts(1){
    }
    void add(T x,T y){
        poi a;
        a.x=x;
        a.y=y;
        pts.push_back(make_pair(a,++n));
    }
    poi&pot(int a){
        return pts[a].first;
    }
    void con(int a,int b){
        egs[a].push_back(b);
        egs[b].push_back(a);
    }
    void dco(int a,int b){
        egs[a].erase(find(egs[a].begin(),egs[a].end(),b));
        egs[b].erase(find(egs[b].begin(),egs[b].end(),a));
    }
    void dnc(int l,int r){
        if(r==l)
            return;
        if(r==l+1){
            con(l,r);
            return;
        }
        if(r==l+2){
            if(dir(pot(l),pot(l+1),pot(r)))
                con(l,l+1),con(l+1,r),con(l,r);
            else{
                if(dot(pot(l+1)-pot(l),pot(r)-pot(l))<0)
                    con(l,l+1),con(l,r);
            else if(dot(pot(l)−pot(l+1),pot(r)−pot(l+1))<0)
                con(l,l+1),con(l+1,r);
            else
                con(l,r),con(l+1,r);}
        return;
    }
    int m=(l+r)/2,pl=l,pr=r;
    dnc(l,m);
    dnc(m+1,r);
    for(int f=0;;f=0){
        for(int i=0;i<egs[pl].size();++i){
            int a=egs[pl][i],d=dir(pot(pl),pot(pr),pot(a));
            if(d>0||(d==0&&dot(pot(pl)−pot(a),pot(pr)−pot(a))<0)){
                pl=a;
                f=1;
                break;
            }
        }
        for(int i=0;i<egs[pr].size();++i){
            int a=egs[pr][i],d=dir(pot(pl),pot(pr),pot(a));
            if(d>0||(d==0&&dot(pot(pl)−pot(a),pot(pr)−pot(a))<0)){
                pr=a;
                f=1;
                break;
            }
        }
        if(!f)
            break;
    }
    con(pl,pr);
    for(int pn=−1,wh=0;;pn=−1,wh=0){
        for(int i=0;i<egs[pl].size();++i){
            int a=egs[pl][i],d=dir(pot(pl),pot(pr),pot(a));
            if(d<0&&(pn==−1||inc(pot(pl),pot(pr),pot(pn),pot(a))))
                pn=a;
        }
        for(int i=0;i<egs[pr].size();++i){
            int a=egs[pr][i],d=dir(pot(pl),pot(pr),pot(a));
            if(d<0&&(pn==−1||inc(pot(pl),pot(pr),pot(pn),pot(a))))
                pn=a,wh=1;
        }
```

```
            if(pn==-1)
                break;
            vector<int>ne;
            if(!wh){
                for(int i=0;i<egs[pl].size();++i){
                    int a=egs[pl][i];
                    if(!crs(pot(pn),pot(pr),pot(pl),pot(a)))
                        ne.push_back(a);
                    else
                        egs[a].erase(find(egs[a].begin(),egs[a].end(),pl));
                }
                egs[pl]=ne;
                con(pr,pn);
                pl=pn;
            }else{
                for(int i=0;i<egs[pr].size();++i){
                    int a=egs[pr][i];
                    if(!crs(pot(pn),pot(pl),pot(pr),pot(a)))
                        ne.push_back(a);
                    else
                        egs[a].erase(find(egs[a].begin(),egs[a].end(),pr));
                }
                egs[pr]=ne;
                con(pl,pn);
                pr=pn;
            }
        }
    }
    vector<vector<int> >run(){
        egs.resize(n+1);
        sort(pts.begin()+1,pts.end());
        dnc(1,n);
        vector<vector<int> >res(n+1);
        for(int u=1;u<=n;++u)
            for(int i=0;i<egs[u].size();++i){
                int v=egs[u][i];
                res[pts[u].second].push_back(pts[v].second);
            }
        return res;
    }
};
```

```
template<class T>const double DelaunayTriangulation<T>::E=1e-8;
```

1.3 Dynamic Convex Hull (Set)

Dynamic Convex Hull (Set).hpp (2239 bytes, 77 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct DynamicConvexHull{
    struct point{
        T x,y;
        point(T _x=0,T _y=0):
            x(_x),y(_y){
        }
        point operator-(const point&a)const{
            point p(x-a.x,y-a.y);
            return p;
        }
        T operator*(const point&a)const{
            return x*a.y-y*a.x;
        }
    };
    struct node{
        node**nxt;point p;
        node(node**_n,point _p):
            nxt(_n),p(_p){
        }
        node(const node&a):
            nxt(new node*(*a.nxt)),p(a.p){
        }
        ~node(){
            delete nxt;
        }
        int operator<(const node&a)const{
            if(ctp)
                return p.x==a.p.x?p.y<a.p.y:p.x<a.p.x;
            point p1,p2;
            int f=1;
            if(nxt)
```

```
                p1=*nxt?(*nxt)->p-p:point(0,-1),p2=a.p;
            else
                f=0,p1=*a.nxt?(*a.nxt)->p-a.p:point(0,-1),p2=p;
            T x=p1*p2;
            return f?x<0:x>0;
        }
    };
    static int ctp;
    set<node>nds;
    typedef typename set<node>::iterator P;
    int check(P a,P b,P c){
        return (b->p-a->p)*(c->p-b->p)>=0;
    }
    void next(P a,P b){
        *(a->nxt)=(node*)&*b;
    }
    void insert(T x,T y){
        ctp=1;
        node t(new node*(0),point(x,y));
        P it=nds.insert(t).first,itl1=it,itl2,itr1=it,itr2=it;
        if(it!=nds.begin())
            for(next(--itl1,it);itl1!=nds.begin()&&check(--(itl2=itl1),
    itl1,it);)
                next(itl2,it),nds.erase(itl1),itl1=itl2;
        if(++(itr1=it)!=nds.end())
            next(it,itr1);
        if(itl1!=it&&itr1!=nds.end()&&check(itl1,it,itr1)){
            next(itl1,itr1);
            nds.erase(it);
            return;
        }
        if(itr1!=nds.end())
            for(;++(itr2=itr1)!=nds.end()&&check(it,itr1,itr2);)
                next(it,itr2),nds.erase(itr1),itr1=itr2;
    }
    int size(){
        return nds.size();
    }
    pair<T,T>query(T x,T y){
        ctp=0;
        node t=*nds.lower_bound(node(0,point(x,y)));
        return make_pair(t.p.x,t.p.y);
    }
};
template<class T>int DynamicConvexHull<T>::ctp=0;
```

1.4 Dynamic Convex Hull (Treap)

Dynamic Convex Hull (Treap).hpp (9485 bytes, 327 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct DynamicConvexHull{
    struct point{
        T x,y;
        point(T _x,T _y):
            x(_x),y(_y){
        }
        point operator-(const point&a)const{
            point p(x-a.x,y-a.y);
            return p;
        }
        T operator*(const point&a)const{
            return x*a.y-y*a.x;
        }
        int operator<(const point&a)const{
            return x==a.x?y<a.y:x<a.x;
        }
        int operator==(const point&a)const{
            return x==a.x&&y==a.y;
        }
    };
    struct hull{
        point*pt;
        hull*ch[2],*nb[2];
        int sz,fx;
        hull(point*_pt):
            pt(_pt),sz(1),fx(rand()*1.0/RAND_MAX*1e9){
            ch[0]=ch[1]=nb[0]=nb[1]=0;
        }
        T check(point p){
            return (nb[1]?*nb[1]−>pt−*pt:point(0,−1))*p;
        }
        void update(){
            sz=1;
            for(int i=0;i<2;++i)
                if(ch[i])
                    sz+=ch[i]−>sz;
        }
    };
    static int sz(hull*x){
        return x?x−>sz:0;
    }
    static point&pt(hull*x){
        return*x−>pt;
    }
    static struct memory{
        hull*ps,*pp,**ss,**sp;
        int pm,sm;
        vector<hull*>ns;
        memory():
            ps((hull*)malloc(sizeof(hull))),pp(ps),pm(1),ss((hull**)malloc(
    sizeof(hull*))),sp(ss),sm(1){
            ns.push_back(ps);
        }
        ~memory(){
            free(ss);
            for(int i=0;i<ns.size();++i)
                free(ns[i]);
        }
        hull*create(const hull&x){
            if(sp!=ss){
                −−sp;
                **sp=x;
                return*sp;
            }
            if(pp==ps+pm){
                pp=ps=(hull*)malloc(sizeof(hull)*(pm<<=1));
                ns.push_back(ps);
            }
            *pp=x;
            return pp++;
        }
        void destroy(hull*x){
            if(sp==ss+sm){
                hull**t=(hull**)malloc(sizeof(hull*)*sm<<1);
                memcpy(t,ss,sm*sizeof(hull*));
                free(ss);
                sp=(ss=t)+sm;
                sm<<=1;}
            *(sp++)=x;
        }
    }me;
    struct array{
        hull**ps,**pp;
        int pm;
        array():
            ps((hull**)malloc(sizeof(hull*))),pp(ps),pm(1){
        }
        ~array(){
            free(ps);
        }
        int size(){
            return pp-ps;
        }
        hull*operator[](int i){
            return ps[i];
        }
        void push(hull*x){
            if(pp==ps+pm){
                hull**t=(hull**)malloc(sizeof(hull*)*pm<<1);
                memcpy(t,ps,pm*sizeof(hull*));
                free(ps);
                pp=(ps=t)+pm;
                pm<<=1;
            }
            *(pp++)=x;
        }
    };
    static hull*link(hull*x,hull*y,hull*lb,hull*rb,int d,array&ns){
        hull*r=me.create(*x);
        if(x==lb||x==rb){
            r->nb[d]=y;
            if(y)
                y->nb[!d]=r;
        }else
            r->ch[d]=link(r->ch[d],y,lb,rb,d,ns);
        r->update();
        ns.push(r);
        return r;
    }
    static hull*merge(hull*x,hull*y,hull*lb,hull*rb,array&ns){
        if(!x)
            return y;
        if(!y)
            return x;
        int d=x->fx>y->fx;
        hull*r=me.create(d?*x:*y);
        r->ch[d]=d?merge(r->ch[1],y,lb,rb,ns):merge(x,y->ch[0],lb,rb,ns);
        if(d&&x==lb||!d&&y==rb)
            r->ch[d]=link(r->ch[d],r,lb,rb,!d,ns);
        r->update();
        ns.push(r);
        return r;
    }
    static pair<hull*,hull*>split(hull*x,int k,array&ns){
        if(!x)
            return make_pair((hull*)0,(hull*)0);
        int t=sz(x->ch[0])+1;
        hull*r=me.create(*x);
        ns.push(r);
        pair<hull*,hull*>s=split(x->ch[k>=t],k-t*(k>=t),ns);
        if(k>=t){
            r->ch[1]=s.first;r->update();
            return make_pair(r,s.second);
        }else{
            r->ch[0]=s.second;r->update();
            return make_pair(s.first,r);
        }
    }
    static void turn(hull*&x,int d,int&k){
        k+=(sz((x=x->ch[d])->ch[!d])+1)*(2*d-1);
    }
    static pair<T,T>range(hull*x){
        hull*l=x,*r=x;
        while(l->ch[0])
            l=l->ch[0];
        while(r->ch[1])
            r=r->ch[1];
        return make_pair(pt(l).x,pt(r).x);
    }
    static hull*merge(hull*x,hull*y,array&ns){
        int kp=sz(x->ch[0])+1,kq=sz(y->ch[0])+1,pd[2],qd[2];
        pair<T,T>pr=range(x),qr=range(y);
        int pf=1;
        hull*p=x,*q=y;
        if(pr.second==qr.first&&pr.first==pr.second&&p->ch[pf=0])
            turn(p,0,kp);
        for(point pq=pt(q)-pt(p);;pq=pt(q)-pt(p)){
            pd[0]=(p->nb[0]&&(pt(p->nb[0])-pt(p))*pq<=0)*pf;
            qd[1]=(q->nb[1]&&(pt(q->nb[1])-pt(q))*pq<=0);
            pd[1]=(p->nb[1]&&(pt(p->nb[1])-pt(p))*pq<0)*pf;
            qd[0]=(q->nb[0]&&(pt(q->nb[0])-pt(q))*pq<0);
            if(!(pd[0]+pd[1]+qd[0]+qd[1])){
                hull*l=split(x,kp,ns).first,*r=split(y,kq-1,ns).second,*lb=
    l,*rb=r;
                while(lb->ch[1])
                    lb=lb->ch[1];
                while(rb->ch[0])
                    rb=rb->ch[0];
                return merge(l,r,lb,rb,ns);
            }
            if(!(pd[0]+pd[1]))
                turn(q,qd[1],kq);
            if(!(qd[0]+qd[1]))
                turn(p,pd[1],kp);
            if(pd[0]&&qd[1])
                turn(p,0,kp),turn(q,1,kq);
            if(pd[1]&&qd[1])
                turn(q,1,kq);
            if(pd[0]&&qd[0])turn(p,0,kp);
            if(pd[1]&&qd[0]){
                point vp=pt(p->nb[1])-pt(p),vq=pt(q->nb[0])-pt(q);
                if(vp.x==0&&vq.x==0)
                turn(p,1,kp),turn(q,0,kq);
            else if(vp.x==0)
                turn(p,1,kp);
            else if(vq.x==0)
                turn(q,0,kq);
            else{
                long double m=pr.second,pb=vp.y*(m-pt(p).x),qb=vq.y*(m-
pt(q).x);
                pb=pb/vp.x+pt(p).y;
                qb=qb/vq.x+pt(q).y;
                if(qb>pb+1e-8)
                    turn(q,0,kq);
                else if(pb>qb+1e-8)
                    turn(p,1,kp);
                else if(pt(q->nb[0]).x+pt(p->nb[1]).x<2*m)
                    turn(q,0,kq);
                else
                    turn(p,1,kp);
            }
        }
    }
}
hull*query(hull*x,point p){
    for(hull*y=0;;){
        T d=x->check(p);
        if(d>0)
            y=x,x=x->ch[0];
        else if(d<0)
            x=x->ch[1];
        else
            y=x;
        if(!d||!x)
            return y;
    }
}
struct treap{
    int fx,ct,sz;
    point pt;
    treap*ch[2];
    struct hull*ip,*hu;
    array ns;
```

```
        treap(point _pt):
            fx(rand()*1.0/RAND_MAX*1e9),ct(1),sz(1),pt(_pt),ip(me.create(
    hull(&pt))),hu(ip){
            ch[0]=ch[1]=0;
        }
        ~treap(){
            for(hull**i=ns.ps;i!=ns.pp;++i)
                me.destroy(*i);
            me.destroy(ip);
        }
        void update(){
            for(hull**i=ns.ps;i!=ns.pp;++i)
                me.destroy(*i);
            ns.pp=ns.ps;
            sz=1;
            hu=ip;
            if(ch[0])
                hu=merge(ch[0]->hu,hu,ns),sz+=ch[0]->sz;
            if(ch[1])
                hu=merge(hu,ch[1]->hu,ns),sz+=ch[1]->sz;
        }
    }*root;
    void rotate(treap*&x,int d){
        treap*y=x->ch[d];
        x->ch[d]=y->ch[!d];
        y->ch[!d]=x;
        x=y;
    }
    int insert(treap*&x,point p){
        if(!x)
            x=new treap(p);
        else if(p==x->pt){
            ++x->ct;
            return 0;
        }else{
            int d=x->pt<p;
            if(!insert(x->ch[d],p))
                return 0;
            if(x->ch[d]->fx>x->fx)
                rotate(x,d),x->ch[!d]->update();
            x->update();
        }
        return 1;
    }
    int erase(treap*&x,point p){
        if(p==x->pt){
            if(x->ct>1){
                --x->ct;
                return 0;
            }
            treap*y=x;
            if(!x->ch[0])
                x=x->ch[1],delete y;
            else if(!x->ch[1])
                x=x->ch[0],delete y;
            else{
                int d=x->ch[0]->fx<x->ch[1]->fx;
                rotate(x,d);
                erase(x->ch[!d],p);
                x->update();
            }
            return 1;
        }
        if(erase(x->ch[x->pt<p],p)){
            x->update();
            return 1;
        }else{
            --x->sz;
            return 0;
        }
    }
    void clear(treap*x){
        if(x)
            clear(x->ch[0]),clear(x->ch[1]),delete x;
    }
    DynamicConvexHull():
        root(0){
    }
    ~DynamicConvexHull(){
        clear(root);
    }
    int size(){
        return root?root->sz:0;
    }
    void insert(T x,T y){
        insert(root,point(x,y));
    }
    void erase(T x,T y){
        erase(root,point(x,y));
    }
    pair<T,T>query(T x,T y){
        point r=pt(query(root->hu,point(x,y)));
        return make_pair(r.x,r.y);
    }
};
template<class T>typename DynamicConvexHull<T>::memory DynamicConvexHull<T
    >::me;
```

1.5 Geometry 2D

Geometry 2D.hpp (5031 bytes, 159 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace Geometry2D{
    double eps=1e-8;
    long double pi=acos((long double)-1);
    template<class T>T sqr(T a){
        return a*a;
    }
    template<class T>int cmp(T a,T b){
        if(typeid(T)==typeid(int)||typeid(T)==typeid(long long)){
            if(a==b)
                return 0;
            return a<b?-1:1;
        }
        if(a<b-eps)
            return -1;
        if(a>b+eps)
            return 1;
        return 0;
}
template<class T>struct Point{
    T x,y;
    Point(T _x=0,T _y=0):
        x(_x),y(_y){
    }
    Point<T>&operator+=(const Point<T>&a){
        return*this=*this+a;
    }
    Point<T>&operator-=(const Point<T>&a){
        return*this=*this-a;
    }
};
#define Vector Point
template<class T>Point<T>operator+(const Point<T>&a,const Point<T>&b){
    return Point<T>(a.x+b.x,a.y+b.y);
}
template<class T>Point<T>operator-(const Point<T>&a,const Point<T>&b){
    return Point<T>(a.x-b.x,a.y-b.y);
}
template<class T>Point<T>operator*(T a,const Point<T>&b){
    return Point<T>(b.x*a,b.y*a);
}
template<class T>Point<T>operator*(const Point<T>&a,T b){
    return b*a;
}
template<class T>Point<T>operator/(const Point<T>&a,T b){
    return Point<T>(a.x/b,a.y/b);
}
template<class T>bool operator==(const Point<T>&a,const Point<T>&b){
    return !cmp(a.x,b.x)&&!cmp(a.y,b.y);
}
template<class T>bool operator!=(const Point<T>&a,const Point<T>&b){
    return !(a==b);
}
template<class T>bool operator<(const Point<T>&a,const Point<T>&b){
    int t=cmp(a.x,b.x);
    if(t)
        return t<0;
    return cmp(a.y,b.y)<0;
}
```

```
template<class T>bool operator>(const Point<T>&a,const Point<T>&b){
    return b<a;
}
template<class T>Point<T>NaP(){
    T t=numeric_limits<T>::max();
    return Point<T>(t,t);
}
template<class T>T det(const Point<T>&a,const Point<T>&b){
    return a.x*b.y-a.y*b.x;
}
template<class T>T dot(const Point<T>&a,const Point<T>&b){
    return a.x*b.x+a.y*b.y;
}
template<class T>T abs(const Point<T>&a){
    return sqrt(sqr(a.x)+sqr(a.y));
}
template<class T>T dis(const Point<T>&a,const Point<T>&b){
    return abs(a-b);
}
template<class T>istream&operator>>(istream&s,Point<T>&a){
    return s>>a.x>>a.y;
}
template<class T>ostream&operator<<(ostream&s,const Point<T>&a){
    return s<<a.x<<" "<<a.y;
}
template<class T>struct Segment;
template<class T>struct Line{
    Point<T>u,v;
    Line(const Point<T>&_u=Point<T>(),const Point<T>&_v=Point<T>()):
        u(_u),v(_v){
    }
    Line(const Segment<T>&a):
        u(a.u),v(a.v){
    }
};
template<class T>Point<T>nor(const Line<T>&a){
    Point<T>t=a.v-a.u;
    return Point<T>(t.y,-t.x);
}
template<class T>Point<T>dir(const Line<T>&a){
    return a.v-a.u;
}
template<class T>int dir(const Line<T>a,const Point<T>b){
    return cmp(det(b-a.u,a.v-a.u),T(0));
}
template<class T>Point<T>operator&(const Line<T>&a,const Line<T>&b){
    T p=det(b.u-a.v,b.v-b.u),q=det(a.u-b.v,b.v-b.u);
    return (a.u*p+a.v*q)/(p+q);
}
template<class T>struct Segment{
    Point<T>u,v;
    Segment(const Point<T>&_u=Point<T>(),const Point<T>&_v=Point<T>()):
        u(_u),v(_v){
    }
};
template<class T>Point<T>nor(const Segment<T>&a){
    Point<T>t=a.v-a.u;
    return Point<T>(t.y,-t.x);
}
template<class T>Point<T>dir(const Segment<T>&a){
    return a.v-a.u;
}
template<class T>int dir(const Segment<T>a,const Point<T>b){
    return cmp(b-a.u,a.v-a.u);
}
template<class T>Point<T>operator&(const Line<T>&a,const Segment<T>&b){
    if(dir(a,b.u)*dir(a,b.v)<=0)
        return a&Line<T>(b);
    return NaP<T>();
}
template<class T>Point<T>operator&(const Segment<T>&a,const Line<T>&b){
    return b&a;
}
template<class T>pair<T,T>dis(const Segment<T>&a,const Point<T>&b){
    pair<T,T>d(dis(a.u,b),dis(a.v,b));
    if(d.first>d.second)
        swap(d.first,d.second);
    Point<T>t=Line<T>(b,b+nor(a))&a;
    if(t!=NaP<T>())
        d.first=dis(t,b);
    return d;
}
```

```
    template<class T>pair<T,T>dis(const Point<T>&a,const Segment<T>&b){
        return dis(b,a);
    }
    template<class T>struct Circle{
        Point<T>c;
        T r;
        Circle(const Point<T>&_c=Point<T>(),T _r=0):
            c(_c),r(_r){
        }
    };
    template<class T>T abs(const Circle<T>&a){
        return pi*sqr(a.r);
    }
    template<class T>bool col(const Point<T>&a,const Point<T>&b,const Point
    <T>&c){
        return !cmp(det(a-c,b-c),T(0));
    }
}
```

1.6 Half-Plane Intersection

Half-Plane Intersection.hpp (1950 bytes, 70 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace HalfPlaneIntersection{
    const double E=1e-8;
    struct pot{
        pot(double a=0,double b=0):
            x(a),y(b){
        }
        double x,y;
    };
    double ag(pot p){
        return atan2(double(p.x),double(p.y));
    }
    pot operator+(pot p,pot q){
        return pot(p.x+q.x,p.y+q.y);
    }
```

```
pot operator-(pot p,pot q){
    return pot(p.x-q.x,p.y-q.y);
}
pot operator*(pot p,double q){
    return pot(p.x*q,p.y*q);
}
pot operator/(pot p,double q){
    return pot(p.x/q,p.y/q);
}
double det(pot p,pot q){
    return p.x*q.y-q.x*p.y;
}
double dot(pot p,pot q){
    return p.x*q.x+p.y*q.y;
}
struct lin{
    pot p,q;
    double a;
    lin(pot a,pot b):
        p(a),q(b),a(ag(b-a)){
    }
};
pot operator*(lin a,lin b){
    double a1=det(b.p-a.q,b.q-b.p);
    double a2=det(a.p-b.q,b.q-b.p);
    return (a.p*a1+a.q*a2)/(a1+a2);
}
bool cmp(lin a,lin b){
    if(fabs(a.a-b.a)>E)
        return a.a<b.a;
    else
        return det(a.q-b.p,b.q-b.p)<-E;
}
bool left(lin a,lin b,lin c){
    pot t=a*b;
    return det(t-c.p,c.q-c.p)<-E;
}
deque<lin>run(vector<lin>lns){
    deque<lin>ans;
    sort(lns.begin(),lns.end(),cmp);
    for(int i=0;i<lns.size();++i){
            while(ans.size()>1&&!left(ans.back(),ans[ans.size()-2],lns[i]))
                ans.pop_back();
            while(ans.size()>1&&!left(ans[0],ans[1],lns[i]))
                ans.pop_front();
            if(ans.empty()||fabs(ans.back().a-lns[i].a)>E)
                ans.push_back(lns[i]);}
        while(ans.size()>1&&!left(ans[ans.size()-1],ans[ans.size()-2],ans.
    front()))
            ans.pop_back();
        if(ans.size()<3)
            ans.clear();
        return ans;
    }
}
```

CHAPTER 2

Data Structures

2.1 Binary Heap

Binary Heap.hpp (1629 bytes, 73 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T,class C>struct BinaryHeap{
    struct node{
        node(int _p,T _v):
            p(_p),v(_v){
        }
        int p;
        T v;
    };
    vector<node*>a;
    BinaryHeap():
        a(1){
    }
    ~BinaryHeap(){
        clear();
    }
    void move(int i,int j){
        swap(a[i]->p,a[j]->p);
        swap(a[i],a[j]);
    }
    int check(int i,int j){
        if(!j||j>=a.size()||a[i]->v==a[j]->v)
            return 0;
        return a[i]->v<a[j]->v?-1:1;
    }
    int up(int i){
        if(check(i,i>>1)<0){
            move(i,i>>1);
            return i>>1;
        }else
            return 0;
    }
    int down(int i){
        if(check(i,i<<1)<=0&&check(i,i<<1^1)<=0)
            return a.size();
        if(check(i<<1,i<<1^1)<=0){
            move(i,i<<1);
            return i<<1;
        }else{
            move(i,i<<1^1);
            return i<<1^1;
        }
    }
    void maintain(int i){
        for(int j=up(i);j;i=j,j=up(i));
        for(int j=down(i);j<a.size();i=j,j=down(i));
    }
    void clear(){
        for(int i=1;i<a.size();++i)
            delete a[i];
        a.resize(1);
    }
    node*push(T v){
        a.push_back(new node(a.size(),v));
        node*r=a.back();
        maintain(a.size()-1);
        return r;
    }
    T top(){
        return a[1]->v;
    }
    void pop(){
        move(1,a.size()-1);
        delete a.back();
        a.pop_back();
        maintain(1);
    }
    void modify(node*x,T v){
        x->v=v;
        maintain(x->p);
    }
};
```

2.2 Dynamic Sequence

Dynamic Sequence.hpp (4119 bytes, 177 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct DynamicSequence{
    struct node{
        node(T _i):
            i(_i),v(_i),s(1),r(0){
                c[0]=c[1]=0;
                static int g;
                w=g=(214013*g+2531011);
        }
        T i,v;
        int s,r,w;
        node*c[2];
    }*rt,*sl,*sr;
    struct pool{
        node*ps,*pp,**ss,**sp;
        int pm,sm;
        vector<node*>ns;
        pool():
            ps((node*)malloc(sizeof(node))),pp(ps),pm(1),ss((node**)malloc(
    sizeof(node*))),sp(ss),sm(1){
                ns.push_back(ps);
        }
        ~pool(){
            free(ss);
            for(int i=0;i<ns.size();++i)
                free(ns[i]);
        }
        node*crt(T a){
            if(sp!=ss){
                --sp;
                **sp=node(a);
                return*sp;
            }
            if(pp==ps+pm){
                pp=ps=(node*)malloc(sizeof(node)*(pm<<=1));
                ns.push_back(ps);
            }
            *pp=node(a);
            return pp++;
        }
        void des(node*x){
            if(sp==ss+sm){
                node**t=(node**)malloc(sizeof(node*)*sm<<1);
                memcpy(t,ss,sm*sizeof(node*));
                free(ss);
                sp=(ss=t)+sm;
                sm<<=1;
            }
            *(sp++)=x;
        }
    }me;
    node*bud(T*a,int l,int r){
        if(l>r)
            return 0;
        int m=l+r>>1;
        node*t=me.crt(a[m]);
        t->c[0]=bud(a,l,m-1);
        t->c[1]=bud(a,m+1,r);
        pup(t);
        return t;
    }
    void pdw(node*x){
        for(int d=0;d<2&&(x->i>x->v,1);++d)
            if(x->c[d])
                x->i>x->c[d]->i;
        *x->i;
        *x->v;
        if(x->r){
            -x->i;
            for(int d=0;d<2;++d)
                if(x->c[d])
                    x->c[d]->r^=1;
            swap(x->c[0],x->c[1]);
            x->r=0;
        }
    }
```

```
	void pup(node*x){
		x->i=x->v;
		x->s=1;
		for(int d=0;d<2;++d)
			if(x->c[d])
				pdw(x->c[d]),x->s+=x->c[d]->s,x->i=d?x->i+x->c[d]->i:x->
	c[d]->i+x->i;
	}
	void jon(node*x){
		rt=jon(jon(sl,x),sr);
	}
	node*jon(node*x,node*y){
		if(!x)
			return y;
		if(!y)
			return x;
		pdw(x);
		pdw(y);
		if(x->w<y->w){
			x->c[1]=jon(x->c[1],y);
			pup(x);
			return x;
		}else{
			y->c[0]=jon(x,y->c[0]);
			pup(y);
			return y;
		}
	}
	node*spt(int l,int r){
		spt(rt,l-1);
		node*t=sl;
		spt(sr,r-l+1);
		swap(sl,t);
		return t;
	}
	void spt(node*x,int p){
		if(!x){
			sl=sr=0;
			return;
		}
		pdw(x);
```

```
        int t=x->c[0]?x->c[0]->s:0;
        if(t<p)
            spt(x->c[1],p-t-1),x->c[1]=sl,sl=x;
        else
            spt(x->c[0],p),x->c[0]=sr,sr=x;
        pup(x);
    }
    void clr(node*x){
        if(x)
            clr(x->c[0]),clr(x->c[1]),me.des(x);
    }
    DynamicSequence(T*a=0,int n=0){
        rt=bud(a,1,n);
    }
    ~DynamicSequence(){
        clr(rt);
    }
    void clear(){
        clr(rt);
        rt=0;
    }
    void insert(T a,int p){
        insert(&a-1,1,p);
    }
    void insert(T*a,int n,int p){
        spt(p+1,p);
        jon(bud(a,1,n));
    }
    void erase(int p){
        erase(p,p);
    }
    void erase(int l,int r){
        clr(spt(l,r));
        jon(0);
    }
    T query(int p){
        return query(p,p);
    }
    T query(int l,int r){
        node*t=spt(l,r);
        T i=t->i;
        jon(t);
        return i;
    }
    void modify(T a,int l){
        modify(a,l,l);
    }
    void modify(T a,int l,int r){
        node*t=spt(l,r);
        a>t->i;
        jon(t);
    }
    void reverse(int l,int r){
        node*t=spt(l,r);
        t->r=1;
        jon(t);
    }
    int length(){
        return rt?rt->s:0;
    }
};
```

2.3 Fenwick Tree

Fenwick Tree.hpp (529 bytes, 25 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct FenwickTree{
    FenwickTree(int _n):
        n(_n),l(log2(n)),a(n+1){
    }
    void add(int v,T d){
        for(;v<=n;v+=v&-v)
            a[v]+=d;
    }
    T sum(int v){
        T r=0;
        for(;v;v-=v&-v)
            r+=a[v];
        return r;
    }
    int kth(T k,int r=0){
        for(int i=1<<l;i;i>>=1)
            if(r+i<=n&&a[r+i]<k)
                k-=a[r+=i];
        return r+1;
    }
    int n,l;
    vector<T>a;
};
```

2.4 K-D Tree

K-D Tree.hpp (2467 bytes, 80 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct KDTree{
    struct node{
        node(int x0,int x1,int d):
            color(1),cover(0),dir(d){
                ch[0]=ch[1]=0;
                x[0]=mi[0]=mx[0]=x0;
                x[1]=mi[1]=mx[1]=x1;
        }
        node*ch[2];
        int x[2],mi[2],mx[2],color,cover,dir;
    }*root;
    KDTree(pair<int,int>*a,int n){
        root=build(a,1,n,0);
    }
    static int direct;
    static int cmp(pair<int,int>a,pair<int,int>b){
        if(direct)
            return make_pair(a.second,a.first)<make_pair(b.second,b.first);
        return a<b;
    }
    node*build(pair<int,int>*a,int l,int r,int d){
        int m=(r+l)/2;
        direct=d;
        nth_element(a+l,a+m,a+r+1,cmp);
        node*p=new node((a+m)->first,(a+m)->second,d);
        if(l!=m)
            p->ch[0]=build(a,l,m-1,!d);
        if(r!=m)
            p->ch[1]=build(a,m+1,r,!d);
        for(int i=0;i<2;++i)
            for(int j=0;j<2;++j)
                if(p->ch[j]){
                    p->mi[i]=min(p->mi[i],p->ch[j]->mi[i]);
                    p->mx[i]=max(p->mx[i],p->ch[j]->mx[i]);
                }
        return p;
    }
    void down(node*a){
        if(a->cover){
            for(int i=0;i<2;++i)
                if(a->ch[i])
                    a->ch[i]->cover=a->cover;
            a->color=a->cover;
            a->cover=0;
        }
    }
    void modify(node*a,int mi0,int mx0,int mi1,int mx1,int c){
        if(mi0>a->mx[0]||mx0<a->mi[0]||mi1>a->mx[1]||mx1<a->mi[1])
            return;
        if(mi0<=a->mi[0]&&mx0>=a->mx[0]&&mi1<=a->mi[1]&&mx1>=a->mx[1]){
            a->cover=c;
            return;
        }
        down(a);
        if(mi0<=a->x[0]&&mx0>=a->x[0]&&mi1<=a->x[1]&&mx1>=a->x[1])
            a->color=c;
        for(int i=0;i<2;++i)
            if(a->ch[i])
                modify(a->ch[i],mi0,mx0,mi1,mx1,c);
    }
    void modify(int mi0,int mx0,int mi1,int mx1,int c){
        modify(root,mi0,mx0,mi1,mx1,c);
    }
    int query(node*a,int x0,int x1){
        down(a);
        if(x0==a->x[0]&&x1==a->x[1])
            return a->color;
        direct=a->dir;
        if(cmp(make_pair(x0,x1),make_pair(a->x[0],a->x[1])))
            return query(a->ch[0],x0,x1);
        else
            return query(a->ch[1],x0,x1);
    }
    int query(int x0,int x1){
        return query(root,x0,x1);
    }
};
int KDTree::direct=0;
```

2.5 Link-Cut Tree

Link-Cut Tree.hpp (5518 bytes, 215 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct LinkCutTree{
    struct node{
        node():
            ch({0,0}),pr(0),rev(0){
        }
        node*ch[2],*pr;
        T ifo;
        int rev;
    }*ptrs;
    LinkCutTree(int n):
        ptrs(new node[n]-1){
    }
    ~LinkCutTree(){
        delete ptrs;
    }
    int direct(node*x){
        if(!x->pr)
            return 2;
        if(x==x->pr->ch[0])
            return 0;
        if(x==x->pr->ch[1])
            return 1;
        return 2;
    }
    void down(node*x){
        if(x->rev){
            x->ifo.reverse();
            swap(x->ch[0],x->ch[1]);
            for(int i=0;i<2;++i)
                if(x->ch[i])
                    x->ch[i]->rev^=1;
            x->rev=0;
        }
        x->ifo.down(x->ch[0]?&x->ch[0]->ifo:0,x->ch[1]?&x->ch[1]->ifo:0);
    }
    void up(node*x){
        for(int i=0;i<2;++i)
            if(x->ch[i])
                down(x->ch[i]);
        x->ifo.up(x->ch[0]?&x->ch[0]->ifo:0,x->ch[1]?&x->ch[1]->ifo:0);
    }
    void setchild(node*x,node*y,int d){
        x->ch[d]=y;
        if(y)
            y->pr=x;
        up(x);
    }
    void rotate(node*x){
        node*y=x->pr,*z=y->pr;
        int d1=direct(x),d2=direct(y);
        setchild(y,x->ch[!d1],d1);
        setchild(x,y,!d1);
        if(d2<2)
            setchild(z,x,d2);
        else
            x->pr=z;
    }
    void release(node*x){
        if(direct(x)<2)
            release(x->pr);
        down(x);
    }
    void splay(node*x){
        for(release(x);direct(x)<2;){
            node*y=x->pr;
            if(direct(y)==2)
                rotate(x);
            else if(direct(x)==direct(y))
                rotate(y),rotate(x);
            else
                rotate(x),rotate(x);
        }
    }
    node*access(node*x){
        node*y=0;
        for(;x;y=x,x=x->pr){
            splay(x);
            setchild(x,y,1);
        }
        return y;
    }
    void evert(node*x){
        access(x);
        splay(x);
        x->rev=1;
    }
    void set(int x,T v){
        ptrs[x].ifo=v;
    }
    int linked(int a,int b){
        access((ptrs+a));
        node*z=access((ptrs+b));
        return z==access((ptrs+a));
    }
    void link(int a,int b){
        evert((ptrs+b));
        (ptrs+b)->pr=(ptrs+a);
    }
```

```
void cut(int a,int b){
    access((ptrs+b));
    node*z=access((ptrs+a));
    if(z==(ptrs+a))
        splay((ptrs+b)),(ptrs+b)->pr=0;
    else
        access((ptrs+b)),splay((ptrs+a)),(ptrs+a)->pr=0;
}
int root(int a){
    access((ptrs+a));
    splay((ptrs+a));
    node*r=(ptrs+a);
    while(r->ch[1])
        r=r->ch[1];
    return r-ptrs;
}
void evert(int a){
    evert((ptrs+a));
}
int lca(int a,int b){
    access((ptrs+a));
    return access((ptrs+b))-ptrs;
}
T query(int a){
    splay((ptrs+a));
    T p=(ptrs+a)->ifo;
    p.up(0,0);
    return p;
}
T query(int a,int b){
    if((ptrs+a)==(ptrs+b))
        return query((ptrs+a));
    access((ptrs+a));
    node*c=access((ptrs+b));
    T p=c.ifo;
    if(c==(ptrs+b)){
        splay((ptrs+a));
        T q=(ptrs+a)->ifo;
        q.reverse();
        p.up(&q,0);
        return p;
        }else if(c==(ptrs+a))
            p.up(0,&(ptrs+a)->ch[1]->ifo);
        else{
            splay((ptrs+a));
            T q=(ptrs+a)->ifo;
            q.reverse();
            p.up(&q,&c->ch[1]->ifo);
        }
        return p;
    }
    T equery(int a){
        return query(a);
    }
    T equery(int a,int b){
        access((ptrs+a));
        node*c=access((ptrs+b));
        if(c==(ptrs+b)){
            splay((ptrs+a));
            T q=(ptrs+a)->ifo;
            q.reverse();
            return q;
        }else if(c==(ptrs+a))
            return (ptrs+a)->ch[1]->ifo;
        else{
            splay((ptrs+a));
            node*t=c->ch[1];
            while(t->ch[0])
                t=t->ch[0];
            splay(t);
            if(t->ch[1])
                down(t->ch[1]);
            T p=t->ifo,q=(ptrs+a)->ifo;
            q.reverse();
            p.up(&q,t->ch[1]?&t->ch[1]->ifo:0);
            return p;
        }
    }
    template<class F>void modify(int a,F f){
        splay((ptrs+a));
        f(&(ptrs+a)->ifo);
        up((ptrs+a));
    }
    template<class F>void modify(int a,int b,F f){
        if((ptrs+a)==(ptrs+b)){
            splay((ptrs+a));
            f(0,&(ptrs+a)->ifo,0);
            up((ptrs+a));
            return;
        }
        access((ptrs+a));
        node*c=access((ptrs+b));
        if(c==(ptrs+b))
            splay((ptrs+a)),f(&(ptrs+a)->ifo,&(ptrs+b)->ifo,0);
        else if(c==a)
            f(0,&(ptrs+a)->ifo,&(ptrs+a)->ch[1]->ifo);
        else
            splay(a),f(&(ptrs+a)->ifo,&c->ifo,&c->ch[1]->ifo);
        up(c);
    }
    template<class F>void emodify(int a,F f){
        modify(a,f);
    }
    template<class F>void emodify(int a,int b,F f){
        access((ptrs+a));
        node*c=access((ptrs+b));
        if(c==(ptrs+b))
            splay((ptrs+a)),f(&(ptrs+a)->ifo,0);
        else if(c==a)
            f(0,&(ptrs+a)->ch[1]->ifo);
        else
            splay(a),f(&(ptrs+a)->ifo,&c->ch[1]->ifo);
        up(c);
    }
};
```

2.6 Pairing Heap

Pairing Heap.hpp (2226 bytes, 102 lines)

```
#include<bits/stdc++.h>
```

```
using namespace std;
template<class T,class C>struct PairingHeap{
    PairingHeap():
        root(0),siz(0){
    }
    ~PairingHeap(){
        clear(root);
    }
    struct node{
        node(const T&_val):
            val(_val),ch(0),br(0),pr(0){
        }
        T val;
        node*ch,*br,*pr;
    }*root;
    int siz;
    void merge(node*&x,node*y){
        if(!x)
            x=y;
        else if(y){
            if(C()(y->val,x->val))
                swap(x,y);
            y->br=x->ch;
            if(x->ch)
                x->ch->pr=y;
            y->pr=x;
            x->ch=y;
        }
    }
    void cut(node*&x,node*y){
        if(x==y)
            x=0;
        else{
            if(y==y->pr->ch)
                y->pr->ch=y->br;
            else
                y->pr->br=y->br;
            if(y->br)
                y->br->pr=y->pr;
            y->pr=y->br=0;
        }
}
node*split(node*x){
    vector<node*>t;
    for(node*i=x->ch;i;i=i->br)
        t.push_back(i);
    x->ch=0;
    node*r=0;
    for(int i=0;i<t.size();++i)
        t[i]->pr=t[i]->br=0;
    for(int i=0;i+1<t.size();i+=2)
        merge(t[i],t[i+1]);
    for(int i=0;i<t.size();i+=2)
        merge(r,t[i]);
    return r;
}
void clear(node*x){
    if(x){
        clear(x->ch);
        clear(x->br);
        delete x;
    }
}
void clear(){
    clear(root);
    root=0;
    siz=0;
}
node*push(T a){
    node*r=new node(a);
    merge(root,r);
    ++siz;
    return r;
}
void erase(node*x){
    cut(root,x);
    merge(root,split(x));
    --siz;
}
T top(){
    return root->val;
}
```

```
    void pop(){
        erase(root);
    }
    void merge(PairingHeap<T,C>&a){
        merge(root,a.root);
        a.root=0;
        siz+=a.siz;
        a.siz=0;
    }
    void modify(node*x,T v){
        if(C()(x->val,v))
            x->val=v,merge(root,split(x));
        else
            x->val=v,cut(root,x),merge(root,x);
    }
    int size(){
        return siz;
    }
};
```

2.7 Red-Black Tree

Red-Black Tree.hpp (7432 bytes, 307 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T,class C>struct RedBlackTree{
    struct node{
        node(T _v,node*l,node*r,node*_p,int _b,int _s):
            v(_v),p(_p),b(_b),s(_s){
            c[0]=l;
            c[1]=r;
        }
        T v;
        node*c[2],*p;
        int b,s;
    }*root,*nil;
    void clear(node*x){
        if(x!=nil){
            clear(x->c[0]);
            clear(x->c[1]);
            delete x;
        }
    }
    void rotate(node*x,int d){
        node*y=x->c[!d];
        x->c[!d]=y->c[d];
        if(y->c[d]!=nil)
            y->c[d]->p=x;
        y->p=x->p;
        if(x->p==nil)
            root=y;
        else
            x->p->c[x!=x->p->c[0]]=y;
        y->c[d]=x;
        x->p=y;
        y->s=x->s;
        x->s=x->c[0]->s+x->c[1]->s+1;
    }
    void insert_fixup(node*z){
        while(!z->p->b){
            int d=z->p==z->p->p->c[0];
            node*y=z->p->p->c[d];
            if(!y->b)
                z->p->b=1,y->b=1,(z=z->p->p)->b=0;
            else{
                if(z==z->p->c[d])
                    rotate(z=z->p,!d);
                z->p->b=1;
                z->p->p->b=0;
                rotate(z->p->p,d);
            }
        }
        root->b=1;
    }
    void erase(node*z){
        node*y;
        for(y=z;y!=nil;y=y->p)
            --y->s;
        if(z->c[0]==nil||z->c[1]==nil)
```

```
            y=z;
        else{
            for(y=z->c[1];y->c[0]!=nil;)
                y=y->c[0];
            z->v=y->v;
            y=z->c[1];
            while(y->c[0]!=nil)
                --y->s,y=y->c[0];
        }
        node*x=y->c[y->c[0]==nil];
        x->p=y->p;
        if(y->p==nil)
            root=x;
        else
            y->p->c[y!=y->p->c[0]]=x;
        if(y->b)
            erase_fixup(x);
        delete y;
    }
    void erase_fixup(node*x){
        while(x!=root&&x->b){
            int d=x==x->p->c[0];
            node*w=x->p->c[d];
            if(!w->b){
                w->b=1;
                x->p->b=0;
                rotate(x->p,!d);
                w=x->p->c[d];
            }
            if(w->c[0]->b&&w->c[1]->b)
                w->b=0,x=x->p;
            else{
                if(w->c[d]->b)
                    w->c[!d]->b=1,w->b=0,rotate(w,d),w=x->p->c[d];
                w->b=x->p->b;
                x->p->b=1;
                w->c[d]->b=1;
                rotate(x->p,!d);
                x=root;
            }
        }
        x->b=1;
    }
    node*clone(node*x,node*y){
        if(x.size==0)
            return nil;
        node*z=new node(*x);
        z->c[0]=clone(x->c[0],z);
        z->c[1]=clone(x->c[1],z);
        z->p=y;
        return z;
    }
    node*precursor(node*x){
        if(x->c[0]->count){
            for(x=x->c[0];x->c[1]->count;)
                x=x->c[1];
            return x;
        }else{
            node*y=x->p;
            while(y->count&&x==y->c[0])
                x=y,y=y->p;
            return y;
        }
    }
    node*successor(node*x){
        if(x->c[1]->count){
            for(x=x->c[1];x->c[0]->count;)
                x=x->c[0];
            return x;
        }else{
            node*y=x->p;
            while(y->count&&x==y->c[1])
                x=y,y=y->p;
            return y;
        }
    }
    RedBlackTree(){
        root=nil=(node*)malloc(sizeof(node));
        nil->b=1;
        nil->s=0;
    }
    RedBlackTree(const RedBlackTree&a){
        nil=new node(*a.nil);
        root=clone(a.root,nil);
    }
    ~RedBlackTree(){
        clear(root);
        free(nil);
    }
    RedBlackTree&operator=(const RedBlackTree&a){
        clear(root);
        root=clone(a.root,nil);
        return*this;
    }
    node*begin(){
        node*z=root;
        while(z!=nil&&z->c[0]!=nil)
            z=z->c[0];
        return z;
    }
    node*reverse_begin(){
        node*z=root;
        while(z!=nil&&z->c[1]!=nil)
            z=z->c[1];
        return z;
    }
    node*end(){
        return nil;
    }
    node*reverse_end(){
        return nil;
    }
    void clear(){
        clear(root);
        root=nil;
    }
    void insert(T a){
        node*y=nil,*x=root;
        while(x!=nil)
            y=x,++x->s,x=x->c[C()(x->v,a)];
        node*z=new node(a,nil,nil,y,0,1);
        if(y==nil)
            root=z;
        else
            y->c[C()(y->v,z->v)]=z;
        insert_fixup(z);
    }
    void erase(T a){
        node*z=root;
        for(;;)
            if(C()(a,z->v))
                z=z->c[0];
            else if(C()(z->v,a))
                z=z->c[1];
            else
                break;
        erase(z);
    }
    int count(T a){
        return count_less_equal(a)-count_less(a);
    }
    int count_less(T a){
        int r=0;
        node*z=root;
        while(z!=nil)
            if(C()(z->v,a))
                r+=z->c[0]->s+1,z=z->c[1];
            else
                z=z->c[0];
        return r;
    }
    int count_less_equal(T a){
        int r=0;
        node*z=root;
        while(z!=nil){
            if(!C()(a,z->v))
                r+=z->c[0]->s+1,z=z->c[1];
            else
                z=z->c[0];
        }
        return r;
    }
    int count_greater(T a){
        int r=0;
        node*z=root;
        while(z!=nil)
            if(C()(a,z->v))
                r+=z->c[1]->s+1,z=z->c[0];
            else
                z=z->c[1];
        return r;
    }
    int count_greater_equal(T a){
        int r=0;
        node*z=root;
        while(z!=nil)
            if(!C()(z->v,a))
                r+=z->c[1]->s+1,z=z->c[0];
            else
                z=z->c[1];
        return r;
    }
    node*nth_element(int a){
        node*z=root;
        for(;;)
            if(z->c[0]->s>=a)
                z=z->c[0];
            else if((z->c[0]->s+1)<a)
                a-=z->c[0]->s+1,z=z->c[1];
            else
                return z;
    }
    node*precursor(T a){
        node*z=root,*r=nil;
        while(z!=nil)
            if(C()(z->v,a))
                r=z,z=z->c[1];
            else
                z=z->c[0];
        return r;
    }
    node*successor(T a){
        node*z=root,*r=nil;
        while(z!=nil)
            if(C()(a,z->v))
```

```
                r=z,z=z->c[0];
            else
                z=z->c[1];
        return r;
    }
    node*find(T a){
        node*z=root,*r=nil;
        while(z!=nil)
            if(C()(a,z->v))
                z=z->c[0];
            else if(C()(z->v,a))
                z=z->c[1];
            else
                break;
        return r;
    }
    node*lower_bound(T a){
        node*z=root,*r=nil;
        while(z!=nil)
            if(C()(z->v,a))
                r=z,z=z->c[1];
            else if(C()(a,z->v))
                z=z->c[0];
            else
                r=z,z=z->c[0];
        return r;
    }
    node*upper_bound(T a){
        return successor(a);
    }
    pair<node*,node*> equal_range(T a){
        return make_pair(lower_bound(a),upper_bound(a));
    }
    int size(){
        return root->s;
    }
    int empty(){
        return !root->s;
    }
    T front(){
        return*begin();
    }
    T back(){
        return*reverse_begin();
    }
};
```

2.8 Self-Adjusting Top Tree

Self-Adjusting Top Tree.hpp (12629 bytes, 443 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct SelfAdjustingTopTree{
    const static int inf=~0u>>1;
    static void gmin(int&a,int b){
        a=min(a,b);
    }
    static void gmax(int&a,int b){
        a=max(a,b);
    }
    struct treap{
        SelfAdjustingTopTree*tr;
        treap(struct SelfAdjustingTopTree*a,int n):
            tr(a),ns(n){
        }
        struct node{
            node(){
            }
            node(int a,int b,int c,int d,int e){
                ch[0]=ch[1]=0;
                val=a;
                fix=rand();
                add=0;
                mi=vmi=b;
                mx=vmx=c;
                sum=vsum=d;
                siz=vsiz=e;
                sam=inf;
            }
        node*ch[2];
        int val,fix,vmi,vmx,vsum,vsiz,mi,mx,sum,siz,add,sam;
    };
    vector<node>ns;
    void down(node*a){
        if(a->sam!=inf){
            a->mi=a->mx=a->vmi=a->vmx=a->sam;
            a->vsum=a->sam*a->vsiz;
            a->sum=a->sam*a->siz;
            (&tr->ns[0]+(a-&ns[0]))->viradd=0;
            (&tr->ns[0]+(a-&ns[0]))->virsam=a->sam;
            (&tr->ns[0]+(a-&ns[0]))->add=0;
            (&tr->ns[0]+(a-&ns[0]))->sam=a->sam;
            for(int i=0;i<=1;++i)
                if(a->ch[i])
                    a->ch[i]->add=0,a->ch[i]->sam=a->sam;
            a->sam=inf;
        }
        if(a->add){
            a->mi+=a->add;
            a->mx+=a->add;
            a->vmi+=a->add;
            a->vmx+=a->add;
            a->vsum+=a->add*a->vsiz;
            a->sum+=a->add*a->siz;
            (&tr->ns[0]+(a-&ns[0]))->viradd+=a->add;
            (&tr->ns[0]+(a-&ns[0]))->add+=a->add;
            for(int i=0;i<=1;++i)
                if(a->ch[i])
                    a->ch[i]->add+=a->add;
            a->add=0;
        }
    }
    void update(node*a){
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                down(a->ch[i]);
        a->mi=a->vmi;
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                gmin(a->mi,a->ch[i]->mi);
            a->mx=a->vmx;
            for(int i=0;i<=1;++i)
                if(a->ch[i])
                    gmax(a->mx,a->ch[i]->mx);
            a->sum=a->vsum;
            for(int i=0;i<=1;++i)
                if(a->ch[i])
                    a->sum+=a->ch[i]->sum;
            a->siz=a->vsiz;
            for(int i=0;i<=1;++i)
                if(a->ch[i])
                    a->siz+=a->ch[i]->siz;
        }
        void rotate(node*&a,int d){
            node*b=a->ch[d];
            a->ch[d]=b->ch[!d];
            b->ch[!d]=a;
            update(a);
            update(b);
            a=b;
        }
        void insert(node*&a,node*b){
            if(!a)
                a=b;
            else{
                down(a);
                int d=b->val>a->val;
                insert(a->ch[d],b);
                update(a);
                if(a->ch[d]->fix<a->fix)
                    rotate(a,d);
            }
        }
        void erase(node*&a,int b){
            down(a);
            if(a->val==b){
                if(!a->ch[0])
                    a=a->ch[1];
                else if(!a->ch[1])
                    a=a->ch[0];
                else{
```

```
                int d=a->ch[1]->fix<a->ch[0]->fix;
                down(a->ch[d]);
                rotate(a,d);
                erase(a->ch[!d],b);
                update(a);
            }
        }else{
            int d=b>a->val;
            erase(a->ch[d],b);
            update(a);
        }
    }
};
int n;
SelfAdjustingTopTree(int _n,vector<int>*to,int*we,int rt):
    trp(this,_n+1),ns(_n+1),n(_n){
    build(to,we,rt);
}
struct node{
    node(){}
    node(int a,node*b){
        ch[0]=ch[1]=0;
        pr=b;
        vir=0;
        val=a;
        mi=mx=a;
        siz=1;
        rev=virsum=add=0;
        virmi=inf;
        virmx=-inf;
        sam=inf;
        virsam=inf;
        virsiz=0;
        viradd=0;
    }
    node*ch[2],*pr;
    int val,mi,mx,sum,virmi,virmx,virsum,virsam,viradd,virsiz,rev,sam,
siz,add;
    treap::node*vir;
};
vector<node>ns;
```

```
    treap trp;
    int direct(node*a){
        if(!a->pr)
            return 3;
        else if(a==a->pr->ch[0])
            return 0;
        else if(a==a->pr->ch[1])
            return 1;
        else
            return 2;
    }
    void down(node*a){
        if(a->rev){
            swap(a->ch[0],a->ch[1]);
            for(int i=0;i<=1;++i)
                if(a->ch[i])
                    a->ch[i]->rev^=1;
            a->rev=0;
        }
        if(a->sam!=inf){
            a->val=a->mi=a->mx=a->sam;
            a->sum=a->sam*a->siz;
            for(int i=0;i<=1;++i)
                if(a->ch[i])a->ch[i]->sam=a->sam,a->ch[i]->add=0;
            a->sam=inf;
        }
        if(a->add){
            a->val+=a->add;
            a->mi+=a->add;
            a->mx+=a->add;
            a->sum+=a->add*a->siz;
            for(int i=0;i<=1;++i)
                if(a->ch[i])a->ch[i]->add+=a->add;
            a->add=0;
        }
        if(a->virsam!=inf){
            if(a->virsiz){
                a->virmi=a->virmx=a->virsam;
                a->virsum=a->virsam*a->virsiz;
                if(a->vir)
                    a->vir->add=0,a->vir->sam=a->virsam;
                for(int i=0;i<=1;++i)
                    if(a->ch[i])
                        a->ch[i]->viradd=0,a->ch[i]->virsam=a->virsam;
            }
            a->virsam=inf;
        }
        if(a->viradd){
            if(a->virsiz){
                a->virmi+=a->viradd;
                a->virmx+=a->viradd;
                a->virsum+=a->viradd*a->virsiz;
                if(a->vir)a->vir->add+=a->viradd;
                for(int i=0;i<=1;++i)
                    if(a->ch[i])
                        a->ch[i]->viradd+=a->viradd;
            }
            a->viradd=0;
        }
    }
    void update(node*a){
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                down(a->ch[i]);
        if(a->vir)
            trp.down(a->vir);
        a->mi=a->val;
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                gmin(a->mi,a->ch[i]->mi);
        a->virmi=inf;
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                gmin(a->virmi,a->ch[i]->virmi);
        if(a->vir)
            gmin(a->virmi,a->vir->mi);
        a->mx=a->val;
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                gmax(a->mx,a->ch[i]->mx);
        a->virmx=-inf;
        for(int i=0;i<=1;++i)
```

```
            if(a->ch[i])
                gmax(a->virmx,a->ch[i]->virmx);
        if(a->vir)
            gmax(a->virmx,a->vir->mx);
        a->sum=a->val;
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                a->sum+=a->ch[i]->sum;
        a->virsum=0;
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                a->virsum+=a->ch[i]->virsum;
        if(a->vir)
            a->virsum+=a->vir->sum;
        a->siz=1;
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                a->siz+=a->ch[i]->siz;
        a->virsiz=0;
        for(int i=0;i<=1;++i)
            if(a->ch[i])
                a->virsiz+=a->ch[i]->virsiz;
        if(a->vir)
            a->virsiz+=a->vir->siz;
    }
    void setchd(node*a,node*b,int d){
        a->ch[d]=b;
        if(b)
            b->pr=a;
        update(a);
    }
    void connect(node*a,node*b){
        down(a);
        *(&trp.ns[0]+(a-&ns[0]))=treap::node(a-&ns[0],min(a->virmi,a->mi),
    max(a->virmx,a->mx),a->virsum+a->sum,a->virsiz+a->siz);
        trp.insert(b->vir,&trp.ns[0]+(a-&ns[0]));
    }
    void disconnect(node*a,node*b){
        trp.erase(b->vir,a-&ns[0]);
    }
    void rotate(node*a){
        node*b=a->pr,*c=a->pr->pr;
        int d1=direct(a),d2=direct(b);
        setchd(b,a->ch[!d1],d1);
        setchd(a,b,!d1);
        if(d2<2)
            setchd(c,a,d2);
        else if(d2==2){
            disconnect(b,c);
            connect(a,c);
            a->pr=c;
        }else
            a->pr=0;
    }
    void release(node*a){
        if(direct(a)<2)
            release(a->pr);
        else if(a->pr)
            disconnect(a,a->pr),connect(a,a->pr);
        down(a);
    }
    void splay(node*a){
        release(a);
        while(direct(a)<2){
            node*b=a->pr;
            if(!b->pr||direct(b)>1)
                rotate(a);
            else if(direct(a)==direct(b))
                rotate(b),rotate(a);
            else
                rotate(a),rotate(a);
        }
    }
    node*access(node*a){
        node*b=0;
        while(a){
            splay(a);
            if(a->ch[1])
                connect(a->ch[1],a);
            if(b)
                disconnect(b,a);
            setchd(a,b,1);
            b=a;
            a=a->pr;
        }
        return b;
    }
    void evert(node*a){
        access(a);
        splay(a);
        a->rev=1;
    }
    int qchain(node*a,node*b,int d){
        access(a);
        node*c=access(b);
        splay(c);
        splay(a);
        int ret=c->val;
        if(d==1){
            if(a!=c)
                gmin(ret,a->mi);
            if(c->ch[1])
                down(c->ch[1]),gmin(ret,c->ch[1]->mi);
        }else if(d==2){
            if(a!=c)
                gmax(ret,a->mx);
            if(c->ch[1])
                down(c->ch[1]),gmax(ret,c->ch[1]->mx);
        }else if(d==3){
            if(a!=c)
                ret+=a->sum;
            if(c->ch[1])
                down(c->ch[1]),ret+=c->ch[1]->sum;
        }
        return ret;
    }
    void mchain(node*a,node*b,int u,int d){
        access(a);
        node*c=access(b);
        splay(c);
        splay(a);
        if(d==1){
            c->val+=u;
            if(a!=c)
                a->add=u,disconnect(a,c),connect(a,c);
            if(c->ch[1])
                down(c->ch[1]),c->ch[1]->add=u;
        }else if(d==2){
            c->val=u;
            if(a!=c)
                a->sam=u,disconnect(a,c),connect(a,c);
            if(c->ch[1])
                down(c->ch[1]),c->ch[1]->sam=u;
        }
        update(c);
    }
    int qtree(node*a,int d){
        access(a);
        splay(a);
        int ret=a->val;
        if(d==1){
            if(a->vir)
                trp.down(a->vir),gmin(ret,a->vir->mi);
        }else if(d==2){
            if(a->vir)
                trp.down(a->vir),gmax(ret,a->vir->mx);
        }else if(d==3){
            if(a->vir)
                trp.down(a->vir),ret+=a->vir->sum;
        }
        return ret;
    }
    void mtree(node*a,int u,int d){
        access(a);
        splay(a);
        if(d==1){
            a->val+=u;
            if(a->vir)
                trp.down(a->vir),a->vir->add=u;
        }else if(d==2){
            a->val=u;
            if(a->vir)
                trp.down(a->vir),a->vir->sam=u;
        }
        update(a);
    }
    void stparent(node*a,node*b){
        access(b);
        if(access(a)!=a){
            splay(a);
            node*c=a->ch[0];
            down(c);
            while(c->ch[1])
                c=c->ch[1],down(c);
            splay(c);
            c->ch[1]=0;
            update(c);
            access(b);
            splay(b);
            connect(a,b);
            a->pr=b;
            update(b);
        }
    }
    void build(vector<int>*to,int*we,int rt){
        vector<int>pr(n);
        vector<int>vec;
        queue<int>qu;
        qu.push(rt);
        while(!qu.empty()){
            int u=qu.front();
            qu.pop();
            vec.push_back(u);
            for(int i=0;i<to[u].size();++i){
                int v=to[u][i];
                if(v!=pr[u])
                    qu.push(v),pr[v]=u;
            }
        }
        for(int i=0;i<n;++i){
            int u=vec[i];
            ns[u]=node(we[u],pr[u]?&ns[0]+pr[u]:0);
        }
        for(int i=n-1;i>=0;--i){
            int u=vec[i];
            update(&ns[0]+u);
            if(pr[u])
                connect(&ns[0]+u,&ns[0]+pr[u]);
        }
    }
};
```

2.9 Skew Heap

Skew Heap.hpp (1220 bytes, 61 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T,class C>struct SkewHeap{
    SkewHeap():
        root(0),siz(0){
    }
    ~SkewHeap(){
        clear(root);
    }
    struct node{
        node(T _val):
            val(_val){
            ch[0]=ch[1]=0;
        }
        T val;
        node*ch[2];
    }*root;
    int siz;
    node*merge(node*x,node*y){
        if(!x)
            return y;
        if(!y)
            return x;
        if(C()(y->val,x->val))
            swap(x,y);
        swap(x->ch[0],x->ch[1]=merge(x->ch[1],y));
        return x;
    }
    void clear(node*x){
        if(x){
            clear(x->ch[0]);
            clear(x->ch[1]);
            delete x;
        }
    }
    void clear(){
        clear(root);
        root=0;
        siz=0;
    }
    void push(T a){
        root=merge(root,new node(a));
        ++siz;
    }
    T top(){
        return root->val;
    }
    void pop(){
        root=merge(root->ch[0],root->ch[1]);
        --siz;
    }
    void merge(SkewHeap<T,C>&a){
        root=merge(root,a.root);
        a.root=0;
        siz+=a.siz;
        a.siz=0;
    }
    int size(){
        return siz;
    }
};
```

CHAPTER 3

Graph Algorithms

3.1 Chordality Test

Chordality Test.hpp (1343 bytes, 42 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct ChordalityTest{
    int n,ns;
    vector<vector<int> >to;
    ChordalityTest(int _n):
        n(_n),ns(n),to(n+1){
    }
    void add(int u,int v){
        to[u].push_back(v),to[v].push_back(u);
    }
    bool run(){
        vector<int>pos(n+1),idx(n+2),lab(n+1),tab(n+1);
        vector<list<int>>qu(n);
        for(int i=1;i<=n;++i)
            qu[0].push_back(i);
        for(int b=0,i=1,u=0;i<=n;++i,u=0){
            for(;u?++b,0:1;--b)
                for(auto j=qu[b].begin();j!=qu[b].end()&&!u;qu[b].erase(j++)
    )
                    if(!pos[*j]&&lab[*j]==b)
                        u=*j;
            pos[u]=ns,idx[ns--]=u;
            for(int v:to[u])
                if(!pos[v])
                    b=max(b,++lab[v]),qu[lab[v]].push_back(v);}
        for(int i=1,u=idx[1],v=-1;i<=n;++i,u=idx[i],v=-1){
            for(int w:to[u])
                if(pos[w]>pos[u]&&(v==-1||pos[w]<pos[v]))
                    v=w;
            if(v!=-1){
                for(int w:to[v])
                    tab[w]=1;
                for(int w:to[u])
                    if(pos[w]>pos[u]&&w!=v&&!tab[w])
                        return false;
                for(int w:to[v])
                    tab[w]=0;
            }
        }
        return true;
    }
};
```

3.2 Dominator Tree

Dominator Tree.hpp (2916 bytes, 94 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct DominatorTree{
    int n,r;
    vector<vector<int> >to,rto,chd,rsemi;
    vector<int>dfn,res,prt,rdfn,semi,misemi;
    DominatorTree(int _n,int _r):n(_n),r(_r),to(n+1),rto(n+1),dfn(n+1),res(
    n+1),prt(n+1),rdfn(1),semi(n+1),misemi(n+1),chd(n+1),rsemi(n+1){
    }
    int fd(int a){
        stack<int>stk;
        for(int b=a;prt[b]!=prt[prt[b]];b=prt[b])
            stk.push(b);
        for(int b;stk.empty()?0:(b=stk.top(),stk.pop(),1);){
            if(dfn[semi[misemi[prt[b]]]]<dfn[semi[misemi[b]]])
                misemi[b]=misemi[prt[b]];
            prt[b]=prt[prt[b]];
        }
        return prt[a];
    }
    void add(int a,int b){
        to[a].push_back(b);
        rto[b].push_back(a);
    }
    void dfs(){
        stack<pair<int,int> >stk;
        semi[r]=r;
        for(stk.push(make_pair(r,0));!stk.empty();){
            int a=stk.top().first,i=stk.top().second;
            stk.pop();
            if(!i)
                dfn[a]=rdfn.size(),rdfn.push_back(a);
            if(i<to[a].size()){
                stk.push(make_pair(a,i+1));
                int b=to[a][i];
                if(!semi[b])
                    semi[b]=a,chd[a].push_back(b),
                    stk.push(make_pair(b,0));
            }
        }
        semi[r]=0;
    }
    void calcsemi(){
        for(int i=1;i<=n;++i)
            prt[i]=i,misemi[i]=i;
        for(int i=rdfn.size()-1;i>=1;--i){
            int a=rdfn[i];
            for(int b:rto[a]){
                if(!dfn[b])
                    continue;
                if(dfn[b]<dfn[a]){
                    if(dfn[b]<dfn[semi[a]])
                        semi[a]=b;
                }else{
                    int c=fd(b);
                    if(dfn[semi[c]]<dfn[semi[a]])
                        semi[a]=semi[c];
                    if(dfn[semi[misemi[b]]]<dfn[semi[a]])
                        semi[a]=semi[misemi[b]];
                }
            }
            for(int b:chd[a])
                prt[b]=a;
        }
    }
    void calcres(){
        for(int i=1;i<=n;++i)
            prt[i]=i,misemi[i]=i,rsemi[semi[i]].push_back(i);
        for(int i=rdfn.size()-1;i>=1;--i){
            int a=rdfn[i];
            for(int b:rsemi[a]){
                fd(b);
                int c=misemi[b];
                if(dfn[semi[c]]>dfn[semi[prt[b]]])
                    c=prt[b];
                if(semi[c]==semi[b])
                    res[b]=semi[b];
                else
                    res[b]=-c;}
            for(int b:chd[a])
                prt[b]=a;
        }
        for(int i=1;i<rdfn.size();++i){
            int a=rdfn[i];
            if(res[a]<0)
                res[a]=res[-res[a]];
        }
    }
    vector<int>run(){
        dfs();
        calcsemi();
        calcres();
        return res;
    }
};
```

3.3 K Shortest Path

Description

Find the length of k shortest path between two vertices in a given weighted directed graph. The path does not need to be loopless. But the edge weights must be non-negative.

Methods

template<class T>KShortestPath<T>::KShortestPath(int n);	
Description	construct an object of KShortestPath
Parameters	**Description**
T	type of edge weights, be careful since the result can be $\Theta(nkC)$
n	number of vertices
Time complexity	$\Theta(n)$
Space complexity	$\Theta(11n)$
Return value	an object of KShortestPath

template<class T>void KShortestPath<T>::add(int a,int b,T c);	
Description	add a directed weighted edge to the graph
Parameters	**Description**
a	start vertex of the edge, indexed from one
b	end vertex of the edge, indexed from one
c	weight of the edge, should be non-negative
Time complexity	$\Theta(1)$ (amortized)
Space complexity	$\Theta(1)$ (amortized)
Return value	none

template<class T>T KShortestPath<T>::run(int s,int t,int k);	
Description	find the length of k shortest path
Parameters	**Description**
s	start vertex of the path, indexed from one
t	end vertex of the path, indexed from one
k	k in 'k shortest path'
Time complexity	$O((n+m)\log n + k\log(nmk))$
Space complexity	$O(n\log n + m + k\log(nm))$
Return value	length of k shortest path from s to t or -1 if it doesn't exist

Fields

Name	**Description**

Performance

Problem	**Constraints**	**Time**	**Memory**	**Date**
JDFZ P2978	$N = 10^4, M = 10^5, K = 10^4$	324 ms	14968 kB	2016-02-13

References

Title	Author
堆的可持久化和 k 短路	俞鼎力

Code

K Shortest Path.hpp (5105 bytes, 170 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct KShortestPath{
    KShortestPath(int _n):
        n(_n),m(1<<(int)ceil(log2(n)+1e-8)),from(n+1,-1),
        tov(n+1),wev(n+1),to(n+1),we(n+1),inf(numeric_limits<T>::max()),
        sg(2*m,make_pair(inf,0)),di(n+1,inf),nxt(n+1),chd(n+1),torev(n+1){
    }
    ~KShortestPath(){
        for(int i=0;i<all.size();++i)
            free(all[i]);
    }
    void add(int u,int v,T w){
        tov[v].push_back(u);
        wev[v].push_back(w);
        to[u].push_back(v);
        we[u].push_back(w);
        torev[v].push_back(to[u].size()-1);
    }
    int upd(T&a,T b,T c){
        if(b!=inf&&c!=inf&&b+c<a){
            a=b+c;
            return 1;
        }
        return 0;
    }
    void mod(int u,T d){
        for(sg[u+m-1]=make_pair(d,u),u=u+m-1>>1;u;u>>=1)
            sg[u]=min(sg[u<<1],sg[u<<1^1]);
    }
    template<class T2>struct node{
        node(T2 _v):
            v(_v),s(0),l(0),r(0){
        }
        T2 v;
        int s;
        node*l,*r;
    };
    template<class T2>node<T2>*merge(node<T2>*a,node<T2>*b){
        if(!a||!b)
            return a?a:b;
        if(a->v>b->v)
            swap(a,b);
        a->r=merge(a->r,b);
        if(!a->l||a->l->s<a->r->s)
            swap(a->l,a->r);
        a->s=(a->r?a->r->s:-1)+1;
        return a;
    }
    template<class T2>node<T2>*mak(T2 v){
        node<T2>*t=(node<T2>*)malloc(sizeof(node<T2>));
        *t=node<T2>(v);
        all.push_back(t);
        return t;
    }
    template<class T2>node<T2>*pmerge(node<T2>*a,node<T2>*b){
        if(!a||!b)
            return a?a:b;
        if(a->v>b->v)
            swap(a,b);
        node<T2>*r=mak(a->v);
        r->l=a->l;
        r->r=pmerge(a->r,b);
        if(!r->l||r->l->s<r->r->s)
            swap(r->l,r->r);
        r->s=(r->r?r->r->s:-1)+1;
        return r;
    }
    struct edge{
        edge(T _l,int _v):
            l(_l),v(_v){
        }
        bool operator>(const edge&a){
            return l>a.l;
        }
        T l;
        int v;
    };
    struct edgeheap{
        edgeheap(node<edge>*r):
            root(r){
        }
        bool operator>(const edgeheap&a){
            return root->v.l>a.root->v.l;
        }
        node<edge>*root;
    };
    edgeheap merge(edgeheap a,edgeheap b){
        return edgeheap(pmerge(a->root,b->root));
    }
    edgeheap popmin(edgeheap a){
        return edgeheap(pmerge(a.root->l,a.root->r));
    }
    node<edgeheap>*popmin(node<edgeheap>*a){
          node<edgeheap>*x=pmerge(a->l,a->r);
          a=mak(popmin(a->v));
          if(a->v.root)
              x=pmerge(x,a);
          return x;
    }
    struct path{
        path(int _vp,int _v,T _l,T _d,node<edgeheap>*_c):
            vp(_vp),v(_v),l(_l),d(_d),can(_c){
        }
        bool operator<(const path&a)const{
            return l>a.l;
        }
        int vp,v;
        T l,d;
        node<edgeheap>*can;
    };
    T run(int s,int t,int k){
        di[t]=0;
        for(int i=1;i<=n;++i)
```

```
            sg[i+m−1]=make_pair(di[i],i);
        for(int i=m−1;i>=1;−−i)
            sg[i]=min(sg[i<<1],sg[i<<1^1]);
        for(int u=sg[1].second;sg[1].first!=inf;u=sg[1].second){
            mod(u,inf),tre.push_back(u);
            for(int i=0;i<tov[u].size();++i){
                int v=tov[u][i];
                T w=wev[u][i];
                if(upd(di[v],di[u],w))
                    mod(v,di[v]),nxt[v]=u,
                    from[v]=torev[u][i];
            }
        }
        for(int i=0;i<tre.size();++i){
            queue<node<edge>*>qu;
            for(int j=0;j<to[tre[i]].size();++j)
                if(di[to[tre[i]][j]]!=inf&&j!=from[tre[i]])
                    qu.push(mak(edge(we[tre[i]][j]−di[tre[i]]+di[to[tre[i]][
    j]],to[tre[i]][j])));
            for(node<edge>*x,*y;qu.size()>1;)
                x=qu.front(),qu.pop(),y=qu.front(),qu.pop(),
                qu.push(merge(x,y));
            if(qu.size())
                chd[tre[i]]=pmerge(mak(edgeheap(qu.front())),chd[nxt[tre[i
    ]]]);
            else
                chd[tre[i]]=chd[nxt[tre[i]]];
        }
        priority_queue<path>pth;
        if(di[s]==inf)
            return −1;
        pth.push(path(0,s,di[s],0,0));
        for(int i=1;i<k;++i){
            if(pth.empty())
                return −1;
            path p=pth.top();
            pth.pop();
            if(p.can){
                edge t=p.can−>v.root−>v;
                pth.push(path(p.vp,t.v,p.l−p.d+t.l,t.l,popmin(p.can)));
            }
            if(chd[p.v]){
                edge t=chd[p.v]->v.root->v;
                pth.push(path(p.v,t.v,p.l+t.l,t.l,popmin(chd[p.v])));
            }
        }
        return pth.size()?pth.top().l:-1;
    }
    T inf;
    int n,m;
    vector<T>di;
    vector<int>nxt,tre,from;
    vector<void*>all;
    vector<node<edgeheap>*>chd;
    vector<pair<T,int> >sg;
    vector<vector<T> >wev,we;
    vector<vector<int> >tov,to,torev;
};
```

3.4 Maximal Clique Count

Maximal Clique Count.hpp (927 bytes, 34 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<int N>struct MaximalCliqueCount{
    int n,r;
    vector<bitset<N> >e,rht,msk;
    MaximalCliqueCount(int _n):
        n(_n),e(n),rht(n),msk(n),r(0){
    }
    void add(int u,int v){
        e[u-1][v-1]=e[v-1][u-1]=1;
    }
    void dfs(int u,bitset<N>cur,bitset<N>can){
        if(cur==can){
            ++r;
            return;
        }
        for(int v=0;v<u;++v)
```

```
            if(can[v]&&!cur[v]&&(e[v]&rht[u]&can)==(rht[u]&can))
                return;
        for(int v=u+1;v<n;++v)
            if(can[v])
                dfs(v,cur|msk[v],can&e[v]);
    }
    int run(){
        for(int i=1;i<=n;++i){
            rht[i-1]=bitset<N>(string(n-i,'1')+string(i,'0'));
            msk[i-1]=bitset<N>(1)<<i-1;
            e[i-1]|=msk[i-1];
        }
        for(int i=0;i<n;++i)
            dfs(i,msk[i],e[i]);
        return r;
    }
};
```

3.5 Maximal Planarity Test

Maximal Planarity Test.hpp (5195 bytes, 165 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct MaximalPlanarityTesting{
    int n,m;
    vector<set<int> >to2;
    vector<vector<int> >to;
    vector<int>dec,rmd,mrk,invc,rt;
    vector<list<int>::iterator>dpos,pos;
    bool order(int v1,int v2,int vn){
        rt[0]=v1;
        rt[1]=v2;
        rt[n-1]=vn;
        fill(invc.begin(),invc.end(),0);
        invc[v1]=1;
        invc[v2]=1;
        invc[vn]=1;
        list<int>deg;
    dpos[vn]=deg.insert(deg.begin(),vn);
    fill(dec.begin(),dec.end(),0);
    dec[v1]=2;
    dec[v2]=2;
    dec[vn]=2;
    for(int i=n-1;i>=2;--i){
        if(deg.empty())
            return false;
        int v=*deg.begin();
        deg.erase(deg.begin());
        invc[v]=-1;
        rt[i]=v;
        for(int u:to[v]){
            if(invc[u]==1){
                if(u!=v1&&u!=v2&&dec[u]==2)
                    deg.erase(dpos[u]);
                --dec[u];
                if(u!=v1&&u!=v2&&dec[u]==2)
                    dpos[u]=deg.insert(deg.begin(),u);
            }else if(invc[u]==0)
                invc[u]=2;
        }
        for(int u:to[v])
            if(invc[u]==2)
                for(int w:to[u])
                    if(invc[w]==1){
                        if(w!=v1&&w!=v2&&dec[w]==2)
                            deg.erase(dpos[w]);
                        ++dec[w];
                        if(w!=v1&&w!=v2&&dec[w]==2)
                            dpos[w]=deg.insert(deg.begin(),w);
                        ++dec[u];
                    }else if(invc[w]==2)
                        ++dec[u];
        for(int u:to[v]){
            if(invc[u]==2){
                invc[u]=1;
                if(dec[u]==2)
                    dpos[u]=deg.insert(deg.begin(),u);
            }
        }
        }
        return true;
    }
    bool embed(){
        list<int>ext;
        int mker=0;
        fill(mrk.begin(),mrk.end(),0);
        pos[rt[1]]=ext.insert(ext.begin(),rt[1]);
        pos[rt[2]]=ext.insert(ext.begin(),rt[2]);
        pos[rt[0]]=ext.insert(ext.begin(),rt[0]);
        fill(rmd.begin(),rmd.end(),0);
        rmd[rt[1]]=1;
        rmd[rt[2]]=1;
        rmd[rt[0]]=1;
        for(int i=3;i<n;++i){
            int v=rt[i];
            rmd[v]=1;
            vector<int>can;
            ++mker;
            for(int u:to[v])
                if(rmd[u])
                    mrk[u]=mker,can.push_back(u);
            int start=-1,end=-1;
            for(int u:can){
                list<int>::iterator it=pos[u];
                if(it==list<int>::iterator())
                    return false;
                if(it==ext.begin()){
                    if(start!=-1)
                        return false;
                    start=u;
                }else{
                    list<int>::iterator tmp=it;
                    if(mrk[*(--tmp)]!=mker){
                        if(start!=-1)
                            return false;
                        start=u;
                    }
                }
                list<int>::iterator tmp=it;++tmp;
                if(tmp==ext.end()){
                    if(end!=-1)
                        return false;
                    end=u;
                }else{
                    if(mrk[*tmp]!=mker){
                        if(end!=-1)
                            return false;
                        end=u;
                    }
                }
            }
            if(start==-1||end==-1)
                return false;
            for(int u:can)
                if(u!=start&&u!=end)
                    ext.erase(pos[u]),pos[u]=list<int>::iterator();
            pos[v]=ext.insert(pos[end],v);
        }
        return true;
    }
    bool istri(int u,int v,int w){
        return to2[u].count(v)&&to2[v].count(w)&&to2[w].count(u);
    }
    MaximalPlanarityTesting(int _n):
        n(_n),to(n),to2(n),m(0),rt(n),invc(n),dec(n),dpos(n),pos(n),rmd(n),
    mrk(n){
    }
    void add(int u,int v){
        to[u-1].push_back(v-1);
        to[v-1].push_back(u-1);
        to2[u-1].insert(v-1);
        to2[v-1].insert(u-1);++m;
    }
    bool run(){
        if(n==1&&m==0)
            return true;
        if(n==2&&m==1)
            return true;
        if(n==3&&m==3)
            return true;
        if(n<=3)
```

```
            return false;
        if(m!=3*n-6)
            return false;
        int v1;
        for(v1=0;v1<n;++v1)
            if(to[v1].size()<3)
                return false;
        for(v1=0;v1<n;++v1)
            if(to[v1].size()<=5)
                break;
        if(v1>=n)
            return false;
        int v2=to[v1].back();
        for(int i=0;i+1<to[v1].size();++i){
            int vn=to[v1][i];
            if(istri(v1,v2,vn)){
                if(!order(v1,v2,vn))
                    continue;
                if(!embed())
                    continue;
                return true;
            }
        }
        return false;
    }
};
```

3.6 Maximum Flow

Maximum Flow.hpp (2311 bytes, 79 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct MaximumFlow{
    struct edge{
        int v;
        T c,l;
        edge(int _v,T _c):
            v(_v),c(_c),l(_c){
		}
	};
	int n,src,snk;
	vector<edge>egs;
	vector<vector<int> >bge;
	vector<int>hei,gap,cur,frm;
	MaximumFlow(int _n,int _src,int _snk):
		bge(_n),hei(_n,_n),gap(_n+1),n(_n),cur(_n),frm(_n),src(_src-1),snk(
	_snk-1){
	}
	void lab(){
		hei[snk]=0;
		queue<int>qu;
		qu.push(snk);
		for(int u;qu.empty()?0:(u=qu.front(),qu.pop(),1);)
			for(int i=0;i<bge[u].size();++i){
				edge&e=egs[bge[u][i]],&ev=egs[bge[u][i]^1];
				if(ev.c>0&&hei[e.v]==n)
					hei[e.v]=hei[u]+1,qu.push(e.v);
			}
		for(int i=0;i<n;++i)
			++gap[hei[i]];
	}
	T aug(){
		T f=0;
		for(int u=snk;u!=src;u=egs[frm[u]^1].v)
			if(f<=0||f>egs[frm[u]].c)
				f=egs[frm[u]].c;
		for(int u=snk;u!=src;u=egs[frm[u]^1].v)
			egs[frm[u]].c-=f,egs[frm[u]^1].c+=f;
		return f;
	}
	void add(int u,int v,T c){
		bge[u-1].push_back(egs.size());
		egs.push_back(edge(v-1,c));
		bge[v-1].push_back(egs.size());
		egs.push_back(edge(u-1,0));
	}
	T run(){
		lab();
		T r=0;
```

```
        for(int u=src;hei[src]!=n;){
            if(u==snk)
                r+=aug(),u=src;
            int f=0;
            for(int i=cur[u];i<bge[u].size();++i){
                edge&e=egs[bge[u][i]];
                if(e.c>0&&hei[u]==hei[e.v]+1){
                    f=1;
                    frm[e.v]=bge[u][i];
                    u=e.v;
                    break;
                }
            }
            if(!f){
                int mh=n-1;
                for(int i=0;i<bge[u].size();++i){
                    edge&e=egs[bge[u][i]];
                    if(e.c>0&&mh>hei[e.v])
                        mh=hei[e.v];
                }
                if(!--gap[hei[u]])
                    break;
                ++gap[hei[u]=mh+1];
                cur[u]=0;
                if(u!=src)
                    u=egs[frm[u]^1].v;
            }
        }
        return r;
    }
};
```

3.7 Maximum Matching

Maximum Matching.hpp (3123 bytes, 112 lines)

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

```
int n;
vector<int>res,nxt,mrk,vis,top,prt,rnk;
vector<vector<int> >to;
queue<int>qu;
MaximumMatching(int _n):
    n(_n),res(n+1),nxt(n+1),mrk(n+1),vis(n+1),top(n+1),to(n+1),prt(n+1)
,rnk(n+1){
}
int fd(int x){
    return x==prt[x]?x:prt[x]=fd(prt[x]);
}
void lk(int x,int y){
    if(rnk[x=fd(x)]>rnk[y=fd(y)])
        prt[y]=x;
    else if(rnk[x]<rnk[y])
        prt[x]=y;
    else
        prt[x]=y,++rnk[y];
}
int lca(int x,int y){
    static int t;
    ++t;
    for(;;swap(x,y))
        if(x){
            x=top[fd(x)];
            if(vis[x]==t)
                return x;
            vis[x]=t;
        if(res[x])
            x=nxt[res[x]];
        else
            x=0;
        }
}
void uni(int x,int p){
    for(;fd(x)!=fd(p);){
        int y=res[x],z=nxt[y];
        if(fd(z)!=fd(p))
            nxt[z]=y;
        if(mrk[y]==2)
            mrk[y]=1,qu.push(y);
            if(mrk[z]==2)
                mrk[z]=1,qu.push(z);
            int t=top[fd(z)];
            lk(x,y);
            lk(y,z);
            top[fd(z)]=t;
            x=z;
        }
    }
    void aug(int s){
        for(int i=1;i<=n;++i)
            nxt[i]=0,mrk[i]=0,top[i]=i,prt[i]=i,rnk[i]=0;
        mrk[s]=1;
        qu=queue<int>();
        for(qu.push(s);!qu.empty();){
            int x=qu.front();
            qu.pop();
            for(int i=0;i<to[x].size();++i){
                int y=to[x][i];
                if(res[x]==y||fd(x)==fd(y)||mrk[y]==2)
                    continue;
                if(mrk[y]==1){
                    int z=lca(x,y);
                    if(fd(x)!=fd(z))
                        nxt[x]=y;
                    if(fd(y)!=fd(z))
                        nxt[y]=x;
                    uni(x,z);
                    uni(y,z);
                }else if(!res[y]){
                    for(nxt[y]=x;y;){
                        int z=nxt[y],mz=res[z];
                        res[z]=y;
                        res[y]=z;
                        y=mz;
                    }
                    return;
                }else{
                    nxt[y]=x;
                    mrk[res[y]]=1;
                    qu.push(res[y]);
                    mrk[y]=2;
                }
            }
        }
    }
    void add(int x,int y){
        to[x].push_back(y);
        to[y].push_back(x);
    }
    int run(){
        for(int i=1;i<=n;++i)
            if(!res[i])
                for(int j=0;j<to[i].size();++j)
                    if(!res[to[i][j]]){
                        res[to[i][j]]=i;
                        res[i]=to[i][j];
                        break;
                    }
        for(int i=1;i<=n;++i)
            if(!res[i])
                aug(i);
        int r=0;
        for(int i=1;i<=n;++i)
            if(res[i])
                ++r;
        return r/2;
    }
};
```

3.8 Minimum Cost Maximum Flow

Minimum Cost Maximum Flow.hpp (2278 bytes, 82 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class F=int,class C=int>struct MinimumCostMaximumFlow{
    struct edge{
        edge(int _v,F _c,C _w):
            v(_v),c(_c),w(_w){
        }
        int v;
        F c;
        C w;
    };
    MinimumCostMaximumFlow(int _n,int _src,int _snk,F _all):
        n(_n),src(_src-1),snk(_snk-1),bg(_n),vis(n),dis(n),all(_all),flow
    (0),cost(0){}
    void add(int u,int v,F c,C w){
        bg[u-1].push_back(eg.size());
        eg.push_back(edge(v-1,c,w));
        bg[v-1].push_back(eg.size());
        eg.push_back(edge(u-1,0,-w));
    }
    int spfa(){
        vector<int>in(n,0);
        queue<int>qu;
        fill(vis.begin(),vis.end(),0);
        dis[src]=0;
        vis[src]=in[src]=1;
        qu.push(src);
        while(!qu.empty()){
            int u=qu.front();
            qu.pop();
            in[u]=0;
            for(int i=0;i<bg[u].size();++i){
                edge&e=eg[bg[u][i]];
                if(e.c!=0&&(!vis[e.v]||dis[u]+e.w<dis[e.v])){
                    dis[e.v]=dis[u]+e.w;
                    vis[e.v]=1;
                    if(!in[e.v]){
                        in[e.v]=1;
                        qu.push(e.v);
                    }
                }
            }
        }
        return vis[snk]&&dis[snk]<0;
    }
    F dfs(int u,F f){
        if(u==snk)
```

```
            return f;
        F g=f;
        vis[u]=1;
        for(int i=0;i<bg[u].size();++i){
            edge&e=eg[bg[u][i]],&ev=eg[bg[u][i]^1];
            if(e.c!=0&&dis[e.v]==dis[u]+e.w&&!vis[e.v]){
                F t=dfs(e.v,min(g,e.c));
                g-=t;
                e.c-=t;
                ev.c+=t;
                cost+=t*e.w;
                if(g==0)
                    return f;
            }
        }
        return f-g;
    }
    pair<F,C>run(){
        while(all!=0&&spfa()){
            F t;
            do{
                fill(vis.begin(),vis.end(),0);
                flow+=(t=dfs(src,all));
                all-=t;
            }while(t!=0);
        }
        return make_pair(flow,cost);
    }
    int n,src,snk;
    vector<vector<int> >bg;
    vector<edge>eg;
    vector<int>vis;
    vector<C>dis;
    F all,flow;
    C cost;
};
```

3.9 Minimum Spanning Arborescence

Minimum Spanning Arborescence.hpp (1933 bytes, 64 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct MinimumSpanningArborescence{
    struct eg{
        int u,v;
        T w;
    };
    int n,rt;
    vector<eg>egs;
    vector<int>vi,in,id;
    vector<T>inw;
    MinimumSpanningArborescence(int _n,int _rt):
        n(_n),rt(_rt),vi(n+1),in(n+1),inw(n+1),id(n+1){
    }
    void add(int u,int v,T w){
        eg e;
        e.u=u;
        e.v=v;
        e.w=w;
        egs.push_back(e);
    }
    T run(){
        int nv=0;
        for(T r=0;;n=nv,nv=0,rt=id[rt]){
            for(int i=1;i<=n;++i)
                in[i]=-1;
            for(int i=0;i<egs.size();++i)
                if(egs[i].u!=egs[i].v&&(in[egs[i].v]==-1||egs[i].w<inw[egs[
    i].v]))
                    in[egs[i].v]=egs[i].u,inw[egs[i].v]=egs[i].w;
            for(int i=1;i<=n;++i)
                if(i!=rt&&in[i]==-1)
                    return numeric_limits<T>::max();
            for(int i=1;i<=n;++i){
                if(i!=rt)
                    r+=inw[i];
                id[i]=-1,vi[i]=0;
            }
            for(int i=1;i<=n;++i)
                if(i!=rt&&!vi[i]){
                    int u=i;
                    do{
                        vi[u]=i;
                        u=in[u];
                    }while(!vi[u]&&u!=rt);
                    if(u!=rt&&vi[u]==i){
                        int v=u;
                        ++nv;
                        do{
                            id[v]=nv;
                            v=in[v];
                        }while(v!=u);
                    }
                }
            if(nv==0)
                return r;
            for(int i=1;i<=n;++i)
                if(id[i]==-1)
                    id[i]=++nv;
            for(int i=0;i<egs.size();++i)
                egs[i].w-=inw[egs[i].v],egs[i].u=id[egs[i].u],
                egs[i].v=id[egs[i].v];
        }
    }
};
```

3.10 Minimum Spanning Tree

Minimum Spanning Tree.hpp (1049 bytes, 44 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T,class C=less<T> >struct MinimumSpanningTree{
    struct edge{
        T w;
        int u,v;
        int operator<(const edge&b)const{
            return C()(w,b.w);
        }
    };
    int n;
    vector<edge>egs;
    vector<int>pr;
    MinimumSpanningTree(int _n):
        n(_n),pr(n+1){
    }
    void add(int u,int v,T w){
        edge e;
        e.u=u;
        e.v=v;
        e.w=w;
        egs.push_back(e);
    }
    int fd(int x){
        return x==pr[x]?x:pr[x]=fd(pr[x]);
    }
    void lk(int x,int y){
        pr[fd(x)]=y;
    }
    pair<T,vector<edge> >run(){
        vector<edge>ret;
        T sum=0;
        sort(egs.begin(),egs.end());
        for(int i=1;i<=n;++i)
            pr[i]=i;
        for(int i=0;i<egs.size();++i){
            int u=egs[i].u,v=egs[i].v;
            T w=egs[i].w;
            if(fd(u)!=fd(v))
                lk(u,v),ret.push_back(egs[i]),sum+=w;
        }
        return make_pair(sum,ret);
    }
};
```

3.11 Shortest Path

Shortest Path.hpp (1293 bytes, 45 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct ShortestPath{
    int n,m;
    vector<vector<int> >to;
    vector<vector<T> >we;
    T inf;
    vector<pair<T,int> >sg;
    vector<T>di;
    ShortestPath(int _n):
        n(_n),m(1<<(int)ceil(log2(n)+1e-8)),to(n+1),we(n+1),inf(
    numeric_limits<T>::max()),sg(2*m,make_pair(inf,0)),di(n+1,inf){
    }
    void set(int u,T d){
        di[u]=d;
    }
    void add(int u,int v,T w){
        to[u].push_back(v);
        we[u].push_back(w);
    }
    int upd(T&a,T b,T c){
        if(b!=inf&&c!=inf&&b+c<a){
            a=b+c;
            return 1;
        }
        return 0;
    }
    void mod(int u,T d){
        for(sg[u+m-1]=make_pair(d,u),u=(u+m-1)>>1;u;u>>=1)
            sg[u]=min(sg[u<<1],sg[u<<1^1]);
    }
    vector<T>run(){
        for(int i=1;i<=n;++i)
            sg[i+m-1]=make_pair(di[i],i);
        for(int i=m-1;i>=1;--i)
            sg[i]=min(sg[i<<1],sg[i<<1^1]);
        for(int u=sg[1].second;sg[1].first!=inf?(mod(u,inf),1):0;u=sg[1].
    second)
            for(int i=0;i<to[u].size();++i){
                int v=to[u][i];
                T w=we[u][i];
                if(upd(di[v],di[u],w))
                    mod(v,di[v]);
            }
        return di;
    }
};
```

3.12 Steiner Tree

Steiner Tree.hpp (1745 bytes, 56 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct SteinerTree{
    int n,k,z;
    T inf=numeric_limits<T>::max();
    vector<vector<T> >wei,dp;
    vector<int>im;
    SteinerTree(int _n):
        n(_n),k(0),wei(n+1,vector<T>(n+1,inf)),im(n+1){
    }
    void set(int u){
        if(!im[u])
            im[z=u]=++k;
    }
    void add(int u,int v,T w){
        wei[u][v]=wei[v][u]=min(w,wei[u][v]);
    }
    int upd(T&a,T b,T c){
        if(b!=inf&&c!=inf&&b+c<a){
            a=b+c;
            return 1;
        }
        return 0;
    }
    int ins(int s,int u){
        return im[u]&&((s>>im[u]-1)&1);
    }
    T run(){
        for(int l=1;l<=n;++l)
            for(int i=1;i<=n;++i)
                for(int j=1;j<=n;++j)
                    upd(wei[i][j],wei[i][l],wei[l][j]);
        dp=vector<vector<T> >(1<<k-1,vector<T>(n+1,inf));
        fill(begin(dp[0]),end(dp[0]),0);
        for(int s=1;s<(1<<k-1);++s){
            queue<int>qu;
            vector<int>in(n+1);
            for(int u=1;u<=n;++u){
                if(ins(s,u))
                    continue;
                qu.push((u));
                in[u]=1;
                for(int t=(s-1)&s;t;t=(t-1)&s)
                    upd(dp[s][u],dp[t][u],dp[s^t][u]);
                for(int v=1;v<=n;++v)
                    if(ins(s,v))
                        upd(dp[s][u],dp[s^(1<<im[v]-1)][v],wei[u][v]);
            }
            for(int u;qu.empty()?0:(u=qu.front(),qu.pop(),in[u]=0,1);)
                for(int v=1;v<=n;++v)
                    if(!ins(s,v)&&upd(dp[s][v],dp[s][u],wei[u][v])&&!in[v])
                        in[v]=1,qu.push(v);
        }
        return k?dp[(1<<k-1)-1][z]:0;
    }
};
```

3.13 Virtual Tree

Virtual Tree.hpp (2375 bytes, 77 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct VirtualTree{
    int n,r,l;
    vector<vector<int> >to,vto,up;
```

```
vector<int>lst,dp,dfn,edf,imp;
VirtualTree(int _n,int _r):
    n(_n),r(_r),l(ceil(log2(n)+1e-8)),to(n+1),vto(n+1),up(n+1,vector<
int>(l+1)),dp(n+1),dfn(n+1),edf(n+1),imp(n+1){
}
void add(int u,int v){
    to[u].push_back(v);
    to[v].push_back(u);
}
void vadd(int u,int v){
    vto[u].push_back(v);
}
int lca(int u,int v){
    if(dp[u]<dp[v])
        swap(u,v);
    for(int i=0;i<=l;++i)
        if(((dp[u]-dp[v])>>i)&1)
            u=up[u][i];
    if(u==v)
        return u;
    for(int i=l;i>=0;--i)
        if(up[u][i]!=up[v][i])
            u=up[u][i],v=up[v][i];
    return up[u][0];
}
void dfs(int u){
    dfn[u]=++dfn[0];
    for(int i=1;i<=l;++i)
        up[u][i]=up[up[u][i-1]][i-1];
    for(int i=0;i<to[u].size();++i){
        int v=to[u][i];
        if(v!=up[u][0])
            up[v][0]=u,dp[v]=dp[u]+1,dfs(v);
    }
    edf[u]=dfn[0];
}
void build(){
    dfs(r);
}
void run(int*a,int m){
    for(int i=0;i<lst.size();++i)
```

```
            imp[lst[i]]=0,vto[lst[i]].clear();
        vector<pair<int,int> >b(m+1);
        for(int i=1;i<=m;++i)
            imp[a[i]]=1,b[i]=make_pair(dfn[a[i]],a[i]);
        sort(b.begin()+1,b.end());
        vector<int>st(1,r);
        lst=st;
        for(int i=1;i<=m;++i){
            int u=b[i].second,v=st.back();
            if(u==r)
                continue;
            if(dfn[u]<=edf[v])
                st.push_back(u);
            else{
                int w=lca(u,v);
                while(st.size()>=2&&dp[st[st.size()-2]]>=dp[w]){
                    vadd(st[st.size()-2],*st.rbegin());
                    lst.push_back(*st.rbegin()),st.pop_back();
                }
                if(st.size()>=2&&w!=st[st.size()-1]){
                    vadd(w,*st.rbegin()),lst.push_back(*st.rbegin());
                    st.pop_back(),st.push_back(w);
                }
                st.push_back(u);
            }
        }
        while(st.size()>=2){
            vadd(st[st.size()-2],*st.rbegin());
            lst.push_back(*st.rbegin()),st.pop_back();
        }
    }
};
```

CHAPTER 4

Number Theory

4.1 Discrete Logarithm

Discrete Logarithm.hpp (1819 bytes, 74 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace DiscreteLogarithm{
    typedef long long T;
    int ti[1<<16],va[1<<16],mp[1<<16],nx[1<<16],hd[1<<16],tm,nw;
    void ins(int x,int v){
        int y=x&65535;
        if(ti[y]!=tm)
            ti[y]=tm,hd[y]=0;
        for(int i=hd[y];i;i=nx[i])
            if(va[i]==x){
                mp[i]=v;
                return;
            }
        va[++nw]=x;
        mp[nw]=v;
        nx[nw]=hd[y];
        hd[y]=nw;
    }
    int get(int x){
        int y=x&65535;
        if(ti[y]!=tm)
            ti[y]=tm,hd[y]=0;
        for(int i=hd[y];i;i=nx[i])
            if(va[i]==x){
                return mp[i];
            }
        return -1;
    }
    T pow(T a,T b,T c){
        T r=1;
        for(;b;b&1?r=r*a%c:0,b>>=1,a=a*a%c);
        return r;
    }
    T gcd(T a,T b){
        return b?gcd(b,a%b):a;
    }
    void exg(T a,T b,T&x,T&y){
        if(!b)
            x=1,y=0;
        else
            exg(b,a%b,y,x),y-=a/b*x;
    }
    T inv(T a,T b){
        T x,y;
        exg(a,b,x,y);
        return x+b;
    }
    T bgs(T a,T b,T c){
        ++tm;
        nw=0;
        T m=sqrt(c);
        for(T i=m-1,u=pow(a,i,c),v=inv(a,c);i>=0;--i,u=u*v%c)
            ins(u,i);
        for(T i=0,u=1,v=inv(pow(a,m,c),c);i*m<=c;++i,u=u*v%c){
            T t=u*b%c,j;
            if((j=get(t))!=-1)
                return i*m+j;
        }
        return -1;
    }
    T run(T a,T b,T c){
        T u=1,t=0;
        a=(a%c+c)%c;
        b=(b%c+c)%c;
        for(int i=0;i<32;++i)
            if(pow(a,i,c)==b)
                return i;
        for(T d;(d=gcd(a,c))!=1;++t,u=a/d*u%c,b/=d,c/=d)
            if(b%d)
                return -1;
        return (u=bgs(a,b*inv(u,c)%c,c))<0?-1:u+t;
    }
}
```

4.2 Integer Factorization (Pollard's Rho Algorithm)

Integer Factorization (Pollard's Rho Algorithm).hpp (2848 bytes, 93 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace IntegerFactorization{
    template<class T>T mul(T x,T y,T z){
        if(typeid(T)==typeid(int))
            return (long long)x*y%z;
        else if(typeid(T)==typeid(long long))
            return (x*y-(T)(((long double)x*y+0.5)/z)*z+z)%z;
        else
            return x*y%z;
    }
    template<class T>T pow(T a,T b,T c){
        T r=1;
        for(;b;b&1?r=mul(r,a,c):0,b>>=1,a=mul(a,a,c));
        return r;
    }
    template<class T>int chk(T a,int c=10){
        if(a==2)
            return 1;
        if(a%2==0||a<2)
            return 0;
        static int pi[]={2,7,61},pl
    []={2,325,9375,28178,450775,9780504,1795265022};
        if(typeid(T)==typeid(int))
            c=3;
        else if(typeid(T)==typeid(long long))
            c=7;
        T u=a-1,t=0,p=1;
        for(;u%2==0;u/=2,++t);
        for(int i=0;i<c;++i){
            if(typeid(T)==typeid(int))
                p=pi[i]%a;
            else if(typeid(T)==typeid(long long))
                p=pl[i]%a;
            else
                p=(p*29+7)%a;
        if(!p||p==1||p==a-1)
            continue;
        T x=pow(p,u,a);
        if(x==1)
            continue;
        for(int j=0;x!=a-1&&j<t;++j){
            x=mul(x,x,a);
            if(x==1)
                return 0;
        }
        if(x==a-1)
            continue;
        return 0;
    }
    return 1;
}
template<class T>T gcd(T a,T b){
    if(a<0)
        a=-a;
    if(b<0)
        b=-b;
    return b?gcd(b,a%b):a;
}
template<class T>T rho(T a,T c){
    T x=double(rand())/RAND_MAX*(a-1),y=x;
    for(int i=1,k=2;;){
        x=(mul(x,x,a)+c)%a;
        T d=gcd(y-x,a);
        if(d!=1&&d!=a)
            return d;
        if(y==x)
            return a;
        if(++i==k)
            y=x,k=2*k;
    }
}
template<class T>vector<pair<T,int> >run(T a){
    if(a==1)
        return vector<pair<T,int> >();
    if(chk(a))
        return vector<pair<T,int> >(1,make_pair(a,1));
        T b=a;
        while((b=rho(b,T(double(rand())/RAND_MAX*(a-1))))==a);
        vector<pair<T,int> >u=run(b),v=run(a/b),r;
        for(int pu=0,pv=0;pu<u.size()||pv<v.size();){
            if(pu==u.size())
                r.push_back(v[pv++]);
            else if(pv==v.size())
                r.push_back(u[pu++]);
            else if(u[pu].first==v[pv].first)
                r.push_back(make_pair(u[pu].first,(u[pu].second+v[pv].second
    ))),++pu,++pv;
            else if(u[pu].first>v[pv].first)
                r.push_back(v[pv++]);
            else
                r.push_back(u[pu++]);}
        return r;
    }
}
```

4.3 Integer Factorization (Shanks' Square Forms Factorization)

Integer Factorization (Shanks' Square Forms Factorization).hpp (4675 bytes, 147 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace IntegerFactorization{
    typedef long long ll;
    typedef unsigned long long ull;
    ll lim=368934881469425832611;
    ull srt(const ull&a){
        ull b=sqrt(a);
        b-=b*b>a;
        return b+=(b+1)*(b+1)<=a;
    }
    int sqr(const ull&a,ll&b){
        b=srt(a);
        return b*b==a;
```

```
}
ull gcd(const ull&a,const ull&b){
    return b?gcd(b,a%b):a;
}
ll amb(ll a,const ll&B,const ll&dd,const ll&D){
    for(ll q=(dd+B/2)/a,b=q*a*2−B,c=(D−b*b)/4/a,qc,qcb,a0=a,b0=a,b1=b,
c0=c;;b1=b,c0=c){
        if(c0>dd)
            qcb=c0−b,b=c0+qcb,c=a−qcb;
        else{
            q=(dd+b/2)/c0;
            if(q==1)
                qcb=c0−b,b=c0+qcb,c=a−qcb;
            else
                qc=q*c0,qcb=qc−b,b=qc+qcb,c=a−q*qcb;
        }
        if(a=c0,b==b1)
            break;
        if(b==b0&&a==a0)
            return 0;
    }
    return a&1?a:a>>1;
}
ull fac(const ull&n){
    if(n&1^1)
        return 2;
    if(n%3==0)
        return 3;
    if(n%5==0)
        return 5;
    if(srt(n)*srt(n)==n)
        return srt(n);
    static ll d1,d2,a1,b1,c1,dd1,L1,a2,b2,c2,dd2,L2,a,q,c,qc,qcb,D1,D2,
bl1[1<<19],bl2[1<<19];
    int p1=0,p2=0,ac1=1,ac2=1,j,nm4=n&3;
    if(nm4==1)
        D1=n,D2=5*n,d2=srt(D2),dd2=d2/2+d2%2,b2=(d2−1)|1;
    else
        D1=3*n,D2=4*n,dd2=srt(D2),d2=dd2*2,b2=d2;
    d1=srt(D1),b1=(d1−1)|1,c1=(D1−b1*b1)/4,c2=(D2−b2*b2)/4,L1=srt(d1),
L2=srt(d2),dd1=d1/2+d1%2;
```

```
        for(int i=a1=a2=1;ac1||ac2;++i){
            #define m(t)\
            if(ac##t){\
                c=c##t;\
                q=c>dd##t?1:(dd##t+b##t/2)/c;\
                if(q==1)\
                    qcb=c-b##t,b##t=c+qcb,c##t=a##t-qcb;\
                else\
                    qc=q*c,qcb=qc-b##t,b##t=qc+qcb,c##t=a##t-q*qcb;\
                if((a##t=c)<=L##t)\
                    bl##t[p##t++]=a##t;\
            }
            m(1)m(2)
            if(i&1)
                continue;
            #define m(t)\
            if((ac##t=ac##t&a##t!=1)&&sqr(a##t,a)){\
                if(a<=L##t)\
                    for(j=0;j<p##t;j++)\
                        if(a==bl##t[j]){\
                            a=0;\
                            break;\
                        }\
                if(a>0){\
                    if((q=gcd(a,b##t))>1)\
                        return q*q;\
                    q=amb(a,b##t,dd##t,D##t);\
                    if(nm4==5-2*t&&(q=amb(a,b##t,dd##t,D##t))%(2*t+1)==0)\
                        q/=2*t+1;\
                    if(q>1)\
                        return q;\
                }\
            }
            m(1)m(2)
            #undef m
        }
        for(int i=3;;i+=2)
            if(n%i==0)
                return i;
    }
    ll mul(const ll&x,const ll&y,const ll&z){
        return(x*y-(ll)(((long double)x*y+0.5)/z)*z+z)%z;
    }
    ll pow(ll a,ll b,const ll&c){
        ll r=1;
        for(;b;b&1?r=mul(r,a,c):0,b>>=1,a=mul(a,a,c));
        return r;
    }
    int chk(const ll&a){
        if(a==2)
            return 1;
        if(a%2==0||a<2)
            return 0;
        static int pf[]={2,325,9375,28178,450775,9780504,1795265022};
        ll u=a-1,t=0,p;
        for(;u%2==0;u/=2,++t);
        for(int i=0;i<7;++i){
            p=pf[i]%a;
            if(!p||p==a-1)
                continue;
            ll x=pow(p,u,a);
            if(x==1)
                continue;
            for(int j=0;x!=a-1&&j<t;++j){
                x=mul(x,x,a);
                if(x==1)
                    return 0;
            }
            if(x==a-1)
                continue;
            return 0;
        }
        return 1;
    }
    vector<pair<ll,int> >run(const ll&a){
        if(a==1)
            return vector<pair<ll,int> >();
        if(chk(a))
            return vector<pair<ll,int> >(1,make_pair(a,1));
        ll b=fac(a);
        vector<pair<ll,int> >u=run(b),v=run(a/b),r;
        for(int pu=0,pv=0;pu<u.size()||pv<v.size();){
            if(pu==u.size())
                r.push_back(v[pv++]);
            else if(pv==v.size())
                r.push_back(u[pu++]);
            else if(u[pu].first==v[pv].first)
                r.push_back(make_pair(u[pu].first,(u[pu].second+v[pv].second
    ))),++pu,++pv;
            else if(u[pu].first>v[pv].first)
                r.push_back(v[pv++]);
            else
                r.push_back(u[pu++]);}
        return r;
    }
}
```

4.4 Modular Integer

Modular Integer.hpp (2886 bytes, 98 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct ModularInteger{
    ModularInteger(T t=0):
        v(t){
        if(v<0||v>=p)
            v=(v%p+p)%p;
    }
    ModularInteger<T>&operator=(T a){
        v=a;
        if(v<0||v>=p)
            v%=p;
        return*this;
    }
    ModularInteger<T>operator-(){
        return v?p-v:0;
    }
    ModularInteger<T>&operator+=(ModularInteger<T>a){
        return*this=*this+a;
    }
    ModularInteger<T>&operator-=(ModularInteger<T>a){
        return*this=*this-a;
    }
    ModularInteger<T>&operator*=(ModularInteger<T>a){
        return*this=*this*a;
    }
    ModularInteger<T>&operator/=(ModularInteger<T>a){
        return*this=*this/a;
    }
    T v;
    static T p;
};
template<class T>ModularInteger<T>pow(ModularInteger<T>a,long long b){
    ModularInteger<T>r(1);
    for(;b;b>>=1,a=a*a)
        if(b&1)
            r=r*a;
    return r;
}
template<class T>ModularInteger<T>inv(ModularInteger<T>a){
    return pow(a,a.p-2);
}
template<class T>vector<ModularInteger<T> >sqrt(ModularInteger<T>a){
    vector<ModularInteger<T> >r;
    if(!a.v)
        r.push_back(ModularInteger<T>(0));
    else if(pow(a,a.p-1>>1).v==1){
        int s=a.p-1,t=0;
        ModularInteger<T>b=1;
        for(;pow(b,a.p-1>>1).v!=a.p-1;b=rand()*1.0/RAND_MAX*(a.p-1));
        for(;s%2==0;++t,s/=2);
        ModularInteger<T>x=pow(a,(s+1)/2),e=pow(a,s);
        for(int i=1;i<t;++i,e=x*x/a)
            if(pow(e,1<<t-i-1).v!=1)
                x=x*pow(b,(1<<i-1)*s);
        r.push_back(x);
        r.push_back(-x);
    }
    return r;
}
```

```
template<class T>ModularInteger<T>operator+(ModularInteger<T>a,
    ModularInteger<T>b){
    ModularInteger<T>c(a.v+b.v);
    if(c.v>=a.p)
        c.v-=a.p;
    return c;
}
template<class T>ModularInteger<T>operator-(ModularInteger<T>a,
    ModularInteger<T>b){
    ModularInteger<T>c(a.v-b.v);
    if(c.v<0)
        c.v+=a.p;
    return c;
}
template<class T>ModularInteger<T>operator*(ModularInteger<T>a,
    ModularInteger<T>b){
    if(typeid(T)!=typeid(int))
        return ModularInteger<T>((a.v*b.v-(long long)(((long double)a.v*b.v
    +0.5)/a.p)*a.p+a.p)%a.p);
    else
        return ModularInteger<T>((long long)a.v*b.v%a.p);
}
template<class T>ModularInteger<T>operator/(ModularInteger<T>a,
    ModularInteger<T>b){
    return a*inv(b);
}
template<class T>bool operator==(ModularInteger<T>a,ModularInteger<T>b){
    return a.v==b.v;
}
template<class T>bool operator!=(ModularInteger<T>a,ModularInteger<T>b){
    return a.v!=b.v;
}
template<class T>istream&operator>>(istream&s,ModularInteger<T>&a){
    s>>a.v;
    return s;
}
template<class T>ostream&operator<<(ostream&s,ModularInteger<T>a){
    s<<a.v;
    if(a.v<0||a.v>=a.p)
        a.v%=a.p;
    return s;
}
template<class T>T ModularInteger<T>::p=1e9+7;
```

4.5 Möbius Function

Möbius Function.hpp (534 bytes, 21 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace MobiusFunction{
    vector<int>run(int n){
        vector<int>p,ntp(n+1),u(n+1);
        ntp[1]=1;
        u[1]=1;
        for(int i=2;i<=n;++i){
            if(!ntp[i])
                p.push_back(i),u[i]=-1;
            for(int j=0;j<p.size()&&p[j]*i<=n;++j){
                ntp[p[j]*i]=1;
                if(i%p[j]==0)
                    break;
                else
                    u[p[j]*i]=-u[i];
            }
        }
        return u;
    }
}
```

4.6 Primality Test

Primality Test.hpp (1509 bytes, 52 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace PrimalityTest{
    template<class T>T mul(T x,T y,T z){
        if(typeid(T)==typeid(int))
            return (long long)x*y%z;
        else if(typeid(T)==typeid(long long))
            return (x*y-(T)(((long double)x*y+0.5)/z)*z+z)%z;
        else
            return x*y%z;
    }
    template<class T>T pow(T a,T b,T c){
        T r=1;
        for(;b;b&1?r=mul(r,a,c):0,b>>=1,a=mul(a,a,c));
        return r;
    }
    template<class T>int run(T a,int c=10){
        if(a==2)
            return 1;
        if(a%2==0||a<2)
            return 0;
        static int pi[]={2,7,61},pl
    []={2,325,9375,28178,450775,9780504,1795265022};
        if(typeid(T)==typeid(int))
            c=3;
        else if(typeid(T)==typeid(long long))
            c=7;
        T u=a-1,t=0,p=1;
        for(;u%2==0;u/=2,++t);
        for(int i=0;i<c;++i){
            if(typeid(T)==typeid(int))
                p=pi[i]%a;
            else if(typeid(T)==typeid(long long))
                p=pl[i]%a;
            else
                p=(p*29+7)%a;
            if(!p||p==1||p==a-1)
                continue;
            T x=pow(p,u,a);
            if(x==1)
                continue;
            for(int j=0;x!=a-1&&j<t;++j){
                x=mul(x,x,a);
                if(x==1)
                    return 0;
            }
            if(x==a-1)
                continue;
            return 0;
        }
        return 1;
    }
}
```

4.7 Prime Number

Prime Number.hpp (473 bytes, 18 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace PrimeNumber{
    pair<vector<int>,vector<int> >run(int n){
        vector<int>p,ntp(n+1);
        ntp[1]=1;
        for(int i=2;i<=n;++i){
            if(!ntp[i])
                p.push_back(i);
            for(int j=0;j<p.size()&&p[j]*i<=n;++j){
                ntp[p[j]*i]=1;
                if(i%p[j]==0)
                    break;
            }
        }
        return make_pair(p,ntp);
    }
}
```

4.8 Primitive Root

Primitive Root.hpp (3256 bytes, 106 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace PrimitiveRoot{
    template<class T>T mul(T x,T y,T z){
        if(typeid(T)==typeid(int))
            return (long long)x*y%z;
        else
            return (x*y-(T)(((long double)x*y+0.5)/z)*z+z)%z;
    }
    template<class T>T pow(T a,T b,T c){
        T r=1;
        for(;b;b&1?r=mul(r,a,c):0,b>>=1,a=mul(a,a,c));
        return r;
    }
    template<class T>bool chk(T a,int c=10){
        if(a==1)
            return false;
        T u=a-1,t=0;
        for(;u%2==0;u/=2,++t);
        for(int i=0;i<c;++i){
            T x=pow(T(rand()*1.0/RAND_MAX*(a-2)+1),u,a),y;
            for(int j=0;j<t;++j){
                y=x;
                x=mul(x,x,a);
                if(x==1&&y!=1&&y!=a-1)
                    return false;
            }
            if(x!=1)
                return false;
        }
        return true;
    }
    template<class T>T gcd(T a,T b){
        if(a<0)
            a=-a;
        if(b<0)
            b=-b;
        return b?gcd(b,a%b):a;
    }
    template<class T>T rho(T a,T c){
        T x=double(rand())/RAND_MAX*(a-1),y=x;
        for(int i=1,k=2;;){
            x=(mul(x,x,a)+c)%a;
            T d=gcd(y-x,a);
            if(d!=1&&d!=a)
                return d;
            if(y==x)
                return a;
            if(++i==k)
                y=x,k=2*k;
        }
    }
    template<class T>vector<pair<T,int> >fac(T a){
        if(a==1)
            return vector<pair<T,int> >();
        if(chk(a))
            return vector<pair<T,int> >(1,make_pair(a,1));
        T b=a;
        while((b=rho(b,T(double(rand())/RAND_MAX*(a-1))))==a);
        vector<pair<T,int> >u=fac(b),v=fac(a/b),r;
        for(int pu=0,pv=0;pu<u.size()||pv<v.size();){
            if(pu==u.size())
                r.push_back(v[pv++]);
            else if(pv==v.size())
                r.push_back(u[pu++]);
            else if(u[pu].first==v[pv].first)
                r.push_back(make_pair(u[pu].first,(u[pu].second+v[pv].second
    ))),++pu,++pv;
            else if(u[pu].first>v[pv].first)
                r.push_back(v[pv++]);
            else
                r.push_back(u[pu++]);}
        return r;
    }
    template<class T>void dfs(vector<pair<T,int> >&f,int i,T now,vector<T>&
    r){
        if(i==f.size()){
            r.push_back(now);
            return;
        }
        for(int j=0;j<=f[i].second;++j,now*=f[i].first)
            dfs(f,i+1,now,r);
```

```
	}
	template<class T>T run(T a){
		vector<pair<T,int> >fa=fac(a),fpa;
		if(fa.size()==0||fa.size()>2)
			return -1;
		if(fa.size()==1&&fa[0].first==2&&fa[0].second>2)
			return -1;
		if(fa.size()==2&&fa[0]!=make_pair(T(2),1))
			return -1;
		T pa=a;
		for(int i=0;i<fa.size();++i)
			pa=pa/fa[i].first*(fa[i].first-1);
		fpa=fac(pa);
		vector<T>fs;
		dfs(fpa,0,1,fs);
		for(T g=1,f=0;;++g,f=0){
			for(int i=0;i<fs.size();++i)
				if(fs[i]!=pa&&pow(g,fs[i],a)==1){
					f=1;
					break;
				}
			if(!f)
				return g;
		}
	}
}
```

4.9 Sequence

Numbers n such that a Hadamard matrix of order n exists.
1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 144, 148, 152, 156, 160, 164, 168, 172, 176, 180, 184, 188, 192, 196, 200, 204, 208, 212, 216, 220, 224, 228, 232, 236, 240, ...

Catalan numbers: $C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$. Also called Segner numbers.
1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, 18367353072152, 69533550916004, 263747951750360, 1002242216651368, 3814986502092304, ...

Bell or exponential numbers: number of ways to partition a set of n labeled elements.
1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, 1382958545, 10480142147, 82864869804, 682076806159, 5832742205057, 51724158235372, 474869816156751, 4506715738447323, 44152005855084346, 445958869294805289, 4638590332229999353, 49631246523618756274, ...

CHAPTER 5

Numerical Algorithms

5.1 Convolution (Fast Fourier Transform)

Convolution (Fast Fourier Transform).hpp (1300 bytes, 39 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace Convolution{
    typedef complex<double>T;
    void fft(vector<T>&a,int n,double s,vector<int>&rev){
        T im(0,1);
        double pi=acos(-1);
        for(int i=0;i<n;++i)
            if(i<rev[i])
                swap(a[i],a[rev[i]]);
        for(int i=1,m=2;(1<<i)<=n;++i,m<<=1){
            T wm=exp(s*im*2.0*pi/double(m)),w;
            for(int j=(w=1,0);j<n;j+=m,w=1)
                for(int k=0;k<(m>>1);++k,w*=wm){
                    T u=a[j+k],v=w*a[j+k+(m>>1)];
                    a[j+k]=u+v;
                    a[j+k+(m>>1)]=u-v;
                }
        }
    }
    vector<double>run(const vector<double>&a,const vector<double>&b){
        int l=ceil(log2(a.size()+b.size()-1)),n=1<<l;
        vector<int>rv;
        for(int i=(rv.resize(n),0);i<n;++i)
            rv[i]=(rv[i>>1]>>1)|((i&1)<<(l-1));
        vector<T>ta(n),tb(n);
        copy(a.begin(),a.end(),ta.begin());
        copy(b.begin(),b.end(),tb.begin());
        fft(ta,n,1,rv);
        fft(tb,n,1,rv);
        for(int i=0;i<n;++i)
            ta[i]*=tb[i];
        fft(ta,n,-1,rv);
        vector<double>c(a.size()+b.size()-1);
        for(int i=0;i<c.size();++i)
            c[i]=real(ta[i])/n;
        return c;
    }
}
```

5.2 Convolution (Karatsuba Algorithm)

Convolution (Karatsuba Algorithm).hpp (1416 bytes, 43 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace Convolution{
    template<class T>void kar(T*a,T*b,int n,int l,T**r){
        T*rl=r[l],*rll=r[l-1];
        for(int i=0;i<2*n;++i)
            *(rl+i)=0;
        if(n<=30){
            for(int i=0;i<n;++i)
                for(int j=0;j<n;++j)
                    *(rl+i+j)+=*(a+i)**(b+j);
            return;
        }
        kar(a,b,n>>1,l-1,r);
        for(int i=0;i<n;++i)
            *(rl+i)+=*(rll+i),*(rl+i+(n>>1))+=*(rll+i);
        kar(a+(n>>1),b+(n>>1),n>>1,l-1,r);
        for(int i=0;i<n;++i)
            *(rl+i+n)+=*(rll+i),*(rl+i+(n>>1))+=*(rll+i);
        for(int i=0;i<(n>>1);++i){
            *(rl+(n<<1)+i)=*(a+(n>>1)+i)-*(a+i);
            *(rl+i+(n>>1)*5)=*(b+i)-*(b+(n>>1)+i);
        }
        kar(rl+(n<<1),rl+(n>>1)*5,n>>1,l-1,r);
        for(int i=0;i<n;++i)
            *(rl+i+(n>>1))+=*(rll+i);}
    template<class T>vector<T>run(vector<T>a,vector<T>b){
        int l=ceil(log2(max(a.size(),b.size()))+1e-8);
        vector<T>rt(a.size()+b.size()-1);
        a.resize(1<<l);
        b.resize(1<<l);
        T**r=new T*[l+1];
        for(int i=0;i<=l;++i)
            r[i]=new T[(1<<i)*3];
        kar(&a[0],&b[0],1<<l,l,r);
        for(int i=0;i<rt.size();++i)
            rt[i]=*(r[l]+i);
        for(int i=0;i<=l;++i)
            delete r[i];
        delete r;
        return rt;
    }
}
```

5.3 Convolution (Number Theoretic Transform)

Convolution (Number Theoretic Transform).hpp (1620 bytes, 51 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace Convolution{
    typedef long long T;
    T pow(T a,T b,T c){
        T r=1;
        for(;b;b&1?r=r*a%c:0,b>>=1,a=a*a%c);
        return r;
    }
    void ntt(vector<T>&a,int n,int s,vector<int>&rev,T p,T g){
        g=s==1?g:pow(g,p-2,p);
        vector<T>wm;
        for(int i=0;1<<i<=n;++i)
            wm.push_back(pow(g,(p-1)>>i,p));
        for(int i=0;i<n;++i)
            if(i<rev[i])
                swap(a[i],a[rev[i]]);
        for(int i=1,m=2;1<<i<=n;++i,m<<=1){
            vector<T>wmk(1,1);
            for(int k=1;k<(m>>1);++k)
                wmk.push_back(wmk.back()*wm[i]%p);
            for(int j=0;j<n;j+=m)
```

```
                for(int k=0;k<(m>>1);++k){
                    T u=a[j+k],v=wmk[k]*a[j+k+(m>>1)]%p;
                    a[j+k]=u+v;
                    a[j+k+(m>>1)]=u-v+p;
                    if(a[j+k]>=p)
                        a[j+k]-=p;
                    if(a[j+k+(m>>1)]>=p)
                        a[j+k+(m>>1)]-=p;
                }
            }
    }
    vector<T>run(vector<T>a,vector<T>b,T p=15*(1<<27)+1,T g=31){
        int tn,l=ceil(log2(tn=a.size()+b.size()-1)),n=1<<l;
        vector<int>rv;
        for(int i=(rv.resize(n),0);i<n;++i)
            rv[i]=(rv[i>>1]>>1)|((i&1)<<(l-1));
        a.resize(n);
        b.resize(n);
        ntt(a,n,1,rv,p,g);
        ntt(b,n,1,rv,p,g);
        for(int i=0;i<n;++i)
            a[i]=a[i]*b[i]%p;
        ntt(a,n,-1,rv,p,g);
        n=pow(n,p-2,p);
        for(T&v:a)
            v=v*n%p;
        return a.resize(tn),a;
    }
}
```

5.4 Fraction

Fraction.hpp (2217 bytes, 100 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct Fraction{
    T p,q;
    int s;
    T gcd(T a,T b){
        return b?gcd(b,a%b):a;
    }
    void reduce(){
        T d=gcd(p,q);
        p/=d;
        q/=d;
        if(p==0)
            s=0;
    }
    Fraction(int _s=0,T _p=0,T _q=1):
        s(_s),p(_p),q(_q){
        reduce();
    }
    Fraction(string a){
        if(a[0]=='-'){
            s=-1;
            a=a.substr(1,a.size()-1);
        }else if(a[0]=='+'){
            s=1;
            a=a.substr(1,a.size()-1);
        }else
            s=1;
        stringstream ss;
        char tc;
        ss<<a;
        ss>>p>>tc>>q;
        reduce();
    }
    Fraction(const char*a){
        *this=Fraction(string(a));
    }
    Fraction<T>&operator=(string a){
        return*this=Fraction<T>(a);
    }
    Fraction<T>&operator=(const char*a){
        return*this=Fraction<T>(a);
    }
};
template<class T>ostream&operator<<(ostream&s,const Fraction<T>&a){
    if(a.s==-1)
```

```
        s<<'-';
    return s<<a.p<<'/'<<a.q;
}
template<class T>istream&operator>>(istream&s,Fraction<T>&a){
    string t;
    s>>t;
    a=t;
    return s;
}
template<class T>vector<string>real(const Fraction<T>&a){
    vector<string>r;
    stringstream ss;
    string st;
    if(a.s<0)
        r.push_back("-");
    else
        r.push_back("+");
    T p=a.p,q=a.q;
    ss<<p/q;
    ss>>st;
    r.push_back(st);
    p%=q;
    st.clear();
    map<T,int>mp;
    while(true){
        if(p==0){
            r.push_back(st);
            r.push_back("");
            return r;
        }
        if(mp.count(p)){
            r.push_back(st.substr(0,mp[p]));
            r.push_back(st.substr(mp[p],st.size()-mp[p]));
            return r;
        }
        p*=10;
        mp[p/10]=st.size();
        st.push_back('0'+p/q);
        p%=q;
    }
    return r;
```

```
}
template<class T>string decimal(const Fraction<T>&a){
    string r;
    vector<string>t=real(a);
    if(t[0]=="-")
        r.push_back('-');
    r+=t[1];
    if(t[2].size()||t[3].size())
        r+="."+t[2];
    if(t[3].size())
        r+="("+t[3]+")";
    return r;
}
```

5.5 Integer

Integer.hpp (6378 bytes, 269 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct Integer operator+(Integer a,Integer b);
Integer operator+(Integer a,int b);
Integer operator-(Integer a,Integer b);
Integer operator*(Integer a,Integer b);
Integer operator*(Integer a,Integer b);
Integer operator/(Integer a,Integer b);
Integer operator%(Integer a,Integer b);
Integer operator%(Integer a,int b);
Integer operator%(Integer a,long long b);
bool operator!=(Integer a,int b);
bool operator<=(Integer a,int b);
struct Integer{
    operator bool(){
        return *this!=0;
    }
    Integer(long long a=0){
        if(a<0){
            s=-1;
            a=-a;
        }else
            s=a!=0;
        do{
            d.push_back(a%B);
            a/=B;
        }while(a);
    }
    Integer(string a){
        s=(a[0]=='-')?-1:(a!="0");
        for(int i=a.size()-1;i>=(a[0]=='-');i-=L){
            int t=0,j=max(i-L+1,int(a[0]=='-'));
            for(int k=j;k<=i;++k)
                t=t*10+a[k]-'0';
            d.push_back(t);
        }
    }
    Integer(const Integer&a){
        d=a.d;
        s=a.s;
    }
    Integer&operator=(long long a){
        return*this=Integer(a);
    }
    Integer&operator+=(Integer a){
        return*this=*this+a;
    }
    Integer&operator-=(Integer a){
        return*this=*this-a;
    }
    Integer&operator*=(Integer a){
        return*this=*this*a;
    }
    Integer&operator/=(Integer a){
        return*this=*this/a;
    }
    Integer&operator%=(Integer a){
        return*this=*this%a;
    }
    Integer&operator++(){
        return*this=*this+1;
    }
    operator string()const{
        string r;
        for(int i=0;i<d.size();++i){
            stringstream ts;
            ts<<d[i];
            string tt;
            ts>>tt;
            reverse(tt.begin(),tt.end());
            while(i+1!=d.size()&&tt.size()<L)
                tt.push_back('0');
            r+=tt;
        }
        reverse(r.begin(),r.end());
        return r;
    }
    int s;
    vector<int>d;
    static const int B=1e8,L=8;
};
string str(const Integer&a){
    return string(a);
}
bool operator<(Integer a,Integer b){
    if(a.s!=b.s)
        return a.s<b.s;
    if(a.d.size()!=b.d.size())
        return (a.s!=1)^(a.d.size()<b.d.size());
    for(int i=a.d.size()-1;i>=0;--i)
        if(a.d[i]!=b.d[i])
            return (a.s!=1)^(a.d[i]<b.d[i]);
    return false;
}
bool operator>(Integer a,Integer b){
    return b<a;
}
bool operator<=(Integer a,Integer b){
    return !(a>b);
}
bool operator>=(Integer a,Integer b){
    return !(a<b);
}
```

```
bool operator==(Integer a,Integer b){
    return !(a<b)&&!(a>b);
}
bool operator!=(Integer a,Integer b){
    return !(a==b);
}
istream&operator>>(istream&s,Integer&a){
    string t;
    s>>t;
    a=Integer(t);
    return s;
}
ostream&operator<<(ostream&s,Integer a){
    if(a.s==-1)
        s<<'-';
    for(int i=a.d.size()-1;i>=0;--i){
        if(i!=a.d.size()-1)
            s<<setw(Integer::L)<<setfill('0');
        s<<a.d[i];
    }
    s<<setw(0)<<setfill(' ');
    return s;
}
void dzero(Integer&a){
    while(a.d.size()>1&&a.d.back()==0)
        a.d.pop_back();
}
Integer operator-(Integer a){
    a.s*=-1;
    if(a.d.size()==1&&a.d[0]==0)
        a.s=1;
    return a;
}
Integer operator+(Integer a,int b){
    return a+Integer(b);
}
Integer operator*(Integer a,int b){
    return a*Integer(b);
}
Integer operator%(Integer a,int b){
    return a%Integer(b);
```

```
}
Integer operator%(Integer a,long long b){
    return a%Integer(b);
}
bool operator!=(Integer a,int b){
    return a!=Integer(b);
}
bool operator<=(Integer a,int b){
    return a<=Integer(b);
}
Integer operator+(Integer a,Integer b){
    if(a.s*b.s!=-1){
        Integer c;c.s=a.s?a.s:b.s;
        c.d.resize(max(a.d.size(),b.d.size())+1);
        for(int i=0;i<c.d.size()-1;++i){
            if(i<a.d.size())
                c.d[i]+=a.d[i];
            if(i<b.d.size())
                c.d[i]+=b.d[i];
            if(c.d[i]>=Integer::B){
                c.d[i]-=Integer::B;
                ++c.d[i+1];
            }
        }
        dzero(c);
        return c;
    }
    return a-(-b);
}
Integer operator-(Integer a,Integer b){
    if(a.s*b.s==1){
        if(a.s==-1)
            return (-b)-(-a);
        if(a<b)
            return -(b-a);
        if(a==b)
            return 0;
        for(int i=0;i<b.d.size();++i){
            a.d[i]-=b.d[i];
            if(a.d[i]<0){
                a.d[i]+=Integer::B;
                --a.d[i+1];
            }
        }
        dzero(a);
        return a;
    }
    return a+(-b);
}
Integer operator*(Integer a,Integer b){
    vector<long long>t(a.d.size()+b.d.size());
    for(int i=0;i<a.d.size();++i)
        for(int j=0;j<b.d.size();++j)
            t[i+j]+=(long long)a.d[i]*b.d[j];
    for(int i=0;i<t.size()-1;++i){
        t[i+1]+=t[i]/Integer::B;
        t[i]%=Integer::B;
    }
    Integer c;
    c.s=a.s*b.s;c.d.resize(t.size());
    copy(t.begin(),t.end(),c.d.begin());
    dzero(c);
    return c;
}
Integer div2(Integer a){
    for(int i=a.d.size()-1;i>=0;--i){
        if(i)
            a.d[i-1]+=(a.d[i]&1)*Integer::B;
        a.d[i]>>=1;
    }
    dzero(a);
    if(a.d.size()==1&&a.d[0]==0)
        a.s=0;
    return a;
}
Integer operator/(Integer a,Integer b){
    if(!a.s)
        return 0;
    if(a.s<0)
        return-((-a)/b);
    if(a<b)
        return 0;
    Integer l=1,r=1;
    while(r*b<=a)
        r=r*2;
    while(l+1<r){
        Integer m=div2(l+r);
        if(m*b>a)
            r=m;
        else
            l=m;
    }
    return l;
}
Integer operator%(Integer a,Integer b){
    return a-a/b*b;
}
Integer gcd(Integer a,Integer b){
    Integer r=1;
    while(a!=0&&b!=0){
        if(!(a.d[0]&1)&&!(b.d[0]&1)){
            a=div2(a);
            b=div2(b);
            r=r*2;
        }else if(!(a.d[0]&1))
            a=div2(a);
        else if(!(b.d[0]&1))
            b=div2(b);
        else{
            if(a<b)
                swap(a,b);
            a=div2(a-b);
        }
    }
    if(a!=0)
        return r*a;
    return r*b;
}
int length(Integer a){
    a.s=1;
    return string(a).size();
}
int len(Integer a){
    return length(a);
}
```

5.6 Linear Programming

Linear Programming.hpp (2522 bytes, 89 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct LinearProgramming{
    const double E;
    int n,m,p;
    vector<int>mp,ma,md;
    vector<vector<double> >a;
    vector<double>res;
    LinearProgramming(int _n,int _m):
        n(_n),m(_m),p(0),a(n+2,vector<double>(m+2)),mp(n+1),ma(m+n+2),md(m
    +2),res(m+1),E(1e-8){
    }
    void piv(int l,int e){
        swap(mp[l],md[e]);
        ma[mp[l]]=l;
        ma[md[e]]=-1;
        double t=-a[l][e];
        a[l][e]=-1;
        vector<int>qu;
        for(int i=0;i<=m+1;++i)
            if(fabs(a[l][i]/=t)>E)
                qu.push_back(i);
        for(int i=0;i<=n+1;++i)
            if(i!=l&&fabs(a[i][e])>E){
                t=a[i][e];
                a[i][e]=0;
                for(int j=0;j<qu.size();++j)
                    a[i][qu[j]]+=a[l][qu[j]]*t;
            }
        if(-p==l)
            p=e;
        else if(p==e)
```

```
            p=-1;
    }
    int opt(int d){
        for(int l=-1,e=-1;;piv(l,e),l=-1,e=-1){
            for(int i=1;i<=m+1;++i)
                if(a[d][i]>E){
                    e=i;
                    break;
                }
            if(e==-1)
                return 1;
            double t;
            for(int i=1;i<=n;++i)
                if(a[i][e]<-E&&(l==-1||a[i][0]/-a[i][e]<t))
                    t=a[i][0]/-a[i][e],l=i;
            if(l==-1)
                return 0;
        }
    }
    double&at(int x,int y){
        return a[x][y];
    }
    vector<double>run(){
        for(int i=1;i<=m+1;++i)
            ma[i]=-1,md[i]=i;
        for(int i=m+2;i<=m+n+1;++i)
            ma[i]=i-(m+1),mp[i-(m+1)]=i;
        double t;
        int l=-1;
        for(int i=1;i<=n;++i)
            if(l==-1||a[i][0]<t)
                t=a[i][0],l=i;
        if(t<-E){
            for(int i=1;i<=n;++i)
                a[i][m+1]=1;
            a[n+1][m+1]=-1;
            p=m+1;
            piv(l,m+1);
            if(!opt(n+1)||fabs(a[n+1][0])>E)
                return vector<double>();
            if(p<0)
```

```
                for(int i=1;i<=m;++i)
                    if(fabs(a[-p][i])>E){
                        piv(-p,i);
                        break;
                    }
            for(int i=0;i<=n;++i)
                a[i][p]=0;
        }
        if(!opt(0))
            return vector<double>();
        res[0]=a[0][0];
        for(int i=1;i<=m;++i)
            if(ma[i]!=-1)
                res[i]=a[ma[i]][0];
        return res;
    }
};
```

5.7 Linear System

Linear System.hpp (1477 bytes, 56 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct LinearSystem{
    int n;
    vector<vector<T> >a;
    vector<int>main,pos;
    vector<T>ans;
    int cmp(T a){
        if(typeid(T)==typeid(double)||typeid(T)==typeid(long double)||
    typeid(T)==typeid(float)){
            if(a<-1e-8)
                return -1;
            if(a>1e-8)
                return 1;
            return 0;
        }
        if(a<0)
```

```
            return -1;
        if(a>0)
            return 1;
        return 0;
    }
    T&at(int i,int j){
        return a[i][j];
    }
    vector<T>&at(int i){
        return a[i];
    }
    LinearSystem(int _n):
        n(_n),a(n+1,vector<T>(n+1)),main(n+1),pos(n+1),ans(n){
    }
    vector<T>run(){
        for(int i=1;i<=n;++i){
            int j=1;
            for(;j<=n&&!cmp(a[i][j]);++j);
            if(j<=n){
                main[i]=j;
                pos[j]=i;
                T t=a[i][j];
                for(int k=0;k<=n;++k)
                    a[i][k]/=t;
                for(int k=1;k<=n;++k)
                    if(k!=i&&cmp(a[k][j])){
                        t=a[k][j];
                        for(int l=0;l<=n;++l)
                            a[k][l]-=a[i][l]*t;
                    }
            }
        }
        for(int i=1;i<=n;++i){
            if(!pos[i])
                return vector<T>();
            ans[i-1]=a[pos[i]][0];
        }
        return ans;
    }
};
```

5.8 Matrix

Matrix.hpp (1457 bytes, 51 lines)

```
#include<bits/stdc++.h>
template<class T,int N>struct Matrix{
    Matrix(T t=0){
        for(int i=0;i<N;++i)
            for(int j=0;j<N;++j)
                u[i][j]=i==j?t:0;
    }
    T u[N][N];
};
template<class T,int N>Matrix<T,N>operator+(const Matrix<T,N>&a,const
    Matrix<T,N>&b){
    Matrix<T,N>c;
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
            c.u[i][j]=a.u[i][j]+b.u[i][j];
    return c;
}
template<class T,int N>Matrix<T,N>operator*(const Matrix<T,N>&a,const
    Matrix<T,N>&b){
    Matrix<T,N>c;
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
            for(int k=0;k<N;++k)
                c.u[i][j]+=a.u[i][k]*b.u[k][j];
    return c;
}
template<class T,int N>Matrix<T,N>operator*(const Matrix<T,N>&a,const T&b){
    Matrix<T,N>c=a;
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
            c.u[i][j]*=b;
    return c;
}
template<class T,int N>Matrix<T,N>operator/(const Matrix<T,N>&a,const T&b){
    Matrix<T,N>c=a;
    for(int i=0;i<N;++i)
```

```
        for(int j=0;j<N;++j)
            c.u[i][j]/=b;
    return c;
}
template<class T,int N>Matrix<T,N>pow(Matrix<T,N>a,long long b){
    Matrix<T,N>r(1);
    for(;b;a=a*a,b>>=1)
        if(b&1)
            r=r*a;
    return r;
}
template<class T,int N>ostream&operator<<(ostream&s,const Matrix<T,N>a){
    for(int i=0;i<N;++i)
        for(int j=0;j<N;++j)
            s<<a.u[i][j]<<(j+1==N?'\n':' ');
    return s;
}
```

5.9 Polynomial Interpolation

Polynomial Interpolation.hpp (372 bytes, 15 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>T PolynomialInterpolation(vector<T>x,vector<T>y,T x0){
    T r=0;
    for(int i=0;i<x.size();++i){
        T p=1,q=1;
        for(int j=0;j<x.size();++j)
            if(j!=i){
                p*=(x0-x[j]);
                q*=(x[i]-x[j]);
            }
        r+=p/q*y[i];
    }
    return r;
}
```

CHAPTER 6

String Algorithms

6.1 Aho-Corasick Automaton

Aho-Corasick Automaton.hpp (1369 bytes, 50 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct AhoCorasickAutomaton{
    struct node{
        node(int m):
            tr(m),fail(0),cnt(0){
        }
        vector<node*>tr;
        node*fail;
        int cnt;
    };
    int m;
    node*root;
    vector<node*>all;
    AhoCorasickAutomaton(int _m):
        m(_m),root(new node(m)),all(1,root){
    }
    ~AhoCorasickAutomaton(){
        for(int i=0;i<all.size();++i)
            delete all[i];
    }
    node*insert(int*s){
        node*p;
        for(p=root;*s!=-1;p=p->tr[*(s++)])
            if(!p->tr[*s])
                p->tr[*s]=new node(m);
        return p;
    }
    void build(){
        queue<node*>qu;
        for(int i=0;i<m;++i)
            if(!root->tr[i])
                root->tr[i]=root;
            else
                root->tr[i]->fail=root,qu.push(root->tr[i]);
        for(node*u;qu.size()?(u=qu.front(),qu.pop(),all.push_back(u),1):0;)
```

```
            for(int i=0;i<m;++i)
                if(!u->tr[i])
                    u->tr[i]=u->fail->tr[i];
                else
                    u->tr[i]->fail=u->fail->tr[i],qu.push(u->tr[i]);
    }
    void run(int*s){
        for(node*p=root;*s!=-1;++(p=p->tr[*(s++)])->cnt);
    }
    void count(){
        for(int i=all.size()-1;i>=1;--i)
            all[i]->fail->cnt+=all[i]->cnt;
    }
};
```

6.2 Factor Oracle

Factor Oracle.hpp (569 bytes, 16 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T,int N,int M,T D>struct FactorOracle{
    void insert(T*s,int n){
        memset(tr,(lrs[0]=0,sp[0]=-1),4*M);
        for(int i=0,j,c=s[i]-D,u,v;i<n;c=s[++i]-D){
            memset(tr+i+1,(lrs[i+1]=0)-1,4*M);
            for(j=i;j>-1&&tr[j][c]<0;tr[j][c]=i+1,j=sp[u=j]);
            if(v=sp[i+1]=j<0?0:tr[j][c]){
                for(v=v-1==sp[u]?u:v-1;sp[u]!=sp[v];v=sp[v]);
                lrs[i+1]=min(lrs[u],lrs[v])+1;
            }
        }
    }
    int sp[N+1],lrs[N+1],tr[N+1][M];
};
```

6.3 Longest Common Substring

Longest Common Substring.hpp (1181 bytes, 28 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T,int N,int M,T D>struct LongestCommonSubstring{
    void ins(int c){
        memset(tr+i+1,(lrs[i+1]=0)-1,4*M);
        for(j=i;j>-1&&((v=tr[j][c])>=l1+2&&v<=l1+lb+1||v<0);tr[j][c]=i+1+lb
    ,j=sp[u=j]);
        if(v=sp[i+1]=j<0?0:tr[j][c]-(tr[j][c]>l1+1)*lb){
            for(v=v-1==sp[u]?u:v-1;sp[u]!=sp[v];v=sp[v]);
            lrs[i+1]=min(lrs[u],lrs[v])+1;
        }
        if(sp[i+1]<=l1)
            tm[sp[i+1]]=max(tm[sp[i+1]],lrs[i+1]);
    }
    int run(vector<pair<int,T*> >s){
        swap(s[0],*min_element(s.begin(),s.end()));
        l1=s[k=lb=0].first;
        memset(mi,63,4*N+4);
        memset(tr,(lrs[0]=0,sp[0]=-1),4*M+4);
        for(i=0;i<l1;ins(*(s[0].second+i)-D),++i);
        for(k=1,ins(M);k<s.size();lb+=s[k++].first){
            memset(tm,0,4*N+4);
            for(i=l1+1;i-l1-1<s[k].first;ins(*(s[k].second+i-l1-1)-D),++i)
    ;
            for(i=l1;i;mi[i]=min(mi[i],tm[i]),tm[sp[i]]=max(tm[sp[i]],lrs[i
    ]*!!tm[i]),--i);
        }
        return min(*max_element(mi+1,mi+l1+1),l1);
    }
    int sp[2*N+2],lrs[2*N+2],tr[2*N+2][M+1],mi[N+1],tm[N+1],l1,lb,i,j,k,u,v
    ;
};
```

6.4 Palindromic Tree

Palindromic Tree.hpp (1327 bytes, 50 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct PalindromicTree{
    struct node{
        node(int m,node*f,int l):
            nxt(m),fail(f),len(l){
        }
        vector<node*>nxt;
        node*fail;
        T val;
        int len;
    }*root;
    int m;
    vector<int>str;
    vector<node*>all;
    PalindromicTree(int _m):
        m(_m){
        node*n0=new node(m,0,-2),*n1=new node(m,n0,-1),*n2=new node(m,n1,0)
    ;
        all.push_back(n0);
        all.push_back(n1);
        all.push_back(n2);
        fill(n0->nxt.begin(),n0->nxt.end(),n2);
        root=n1;
    }
    ~PalindromicTree(){
        for(int i=0;i<all.size();++i)
            delete all[i];
    }
    node*find(node*x){
        while(x->fail&&str[str.size()-x->len-2]!=str[str.size()-1])
            x=x->fail;
        return x;
    }
    node*insert(node*p,int c,T v){
        if(p==root)
```

```
            str=vector<int>(1,-1);
        str.push_back(c);
        p=find(p);
        if(!p->nxt[c]){
            node*np=(p->nxt[c]=new node(m,find(p->fail)->nxt[c],p->len+2))
    ;
            all.push_back(np);
        }
        p->nxt[c]->val+=v;
        return p->nxt[c];
    }
    void count(){
        for(int i=all.size()-1;i>=1;--i)
            all[i]->fail->val+=all[i]->val;
    }
};
```

6.5 String Searching

String Searching.hpp (682 bytes, 25 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct StringSearching{
    StringSearching(T*a):
        b(2,a[1]),f(2),l(2){
        for(int i=0;a[l]?1:(--l,0);b.push_back(a[l++])){
            for(;i&&a[i+1]!=a[l];i=f[i]);
            f.push_back(i=i+(a[i+1]==a[l]));
        }
        for(int i=2;i<l;++i)
            if(a[f[i]+1]==a[i+1])
                f[i]=f[f[i]];
    }
    int run(T*a,int p){
        for(int i=p?p+l:1,j=p?f[l]:0;a[i];++i){
            for(;j&&b[j+1]!=a[i];j=f[j]);
            if((j+=b[j+1]==a[i])==l)
                return i-l+1;
        }
        return 0;
    }
    int l;
    vector<T>b;
    vector<int>f;
};
```

6.6 Suffix Array (DC3 Algorithm)

Suffix Array (DC3 Algorithm).hpp (2952 bytes, 107 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct SuffixArray{
    int*sa,*ht,*rk,*ts,*ct,ln;
    SuffixArray(int*s){
        int m=0;
        for(ln=0;s[ln+1];)
            m=max(m,s[++ln]);
        crt(sa,ln);
        crt(ht,ln);
        crt(rk,ln);
        crt(ts,ln);
        crt(ct,max(ln,m));
        dc3(s,ln,m,sa,rk);
        for(int i=1;i<=ln;++i){
            if(rk[i]==1){
                ht[1]=0;
                continue;
            }
            int&d=ht[rk[i]]=max(i==1?0:ht[rk[i-1]]-1,0);
            for(;i+d<=ln&&sa[rk[i]-1]+d<=ln&&s[i+d]==s[sa[rk[i]-1]+d];++d);
        }
    }
    ~SuffixArray(){
        del(sa);
        del(ht);
        del(rk);
        del(ts);
        del(ct);
    }
    void crt(int*&a,int n){
        a=new int[n+1];
    }
    void del(int*a){
        delete a;
    }
    #define fc(i)(p0[i]+d>n||!p0[i]?0:s[p0[i]+d])
    int cmp(int*p0,int i,int*s,int n){
        for(int d=0;d<3;++d)
            if(fc(i)!=fc(i−1))
                return 1;
        return 0;
    }
    void sot(int*p0,int n0,int*s,int n,int m,int d){
        memset(ct,0,(m+1)*4);
        for(int i=1;i<=n0;++i)
            ++ct[fc(i)];
        for(int i=1;i<=m;++i)
            ct[i]+=ct[i−1];
        for(int i=n0;i>=1;−−i)
            ts[ct[fc(i)]−−]=p0[i];
        memcpy(p0+1,ts+1,n0*4);
    }
    #define fc(d)if(s[i+d]!=s[j+d])return s[i+d]<s[j+d];if(i==n−d||j==n−d)
    return i==n−d;
    bool cmp(int*s,int n,int*r,int i,int j){
        fc(0)
        if(j%3==1)
            return r[i+1]<r[j+1];
        fc(1)
        return r[i+2]<r[j+2];
    }
    #undef fc
    void dc3(int*s,int n,int m,int*a,int*r){
        int n0=n−(n/3)+1,*a0,*s0,i,j=0,k=n/3+bool(n%3)+1,l;
        crt(s0,n0);
        s0[k]=1;
        crt(a0,n0+1);
        a0[k]=0;
        for(i=1;i<=n;i+=3)
            a0[++j]=i,a0[j+k]=i+1;
        for(i=2;i>=0;--i)
            sot(a0,n0,s,n,m,i);
        r[a0[1]]=1;
        for(i=2;i<=n0;++i)
            r[a0[i]]=r[a0[i-1]]+cmp(a0,i,s,n);
        for(i=1,j=0;i<=n;i+=3)
            s0[++j]=r[i],s0[j+k]=r[i+1];
        if(r[a0[n0]]==n0){
            memcpy(r+1,s0+1,n0*4);
            for(i=1;i<=n0;++i)
                a0[a[i]=r[i]]=i;
        }else
            dc3(s0,n0,r[a0[n0]],a0,a);
        for(i=1,j=0;i<=n;i+=3)
            r[i]=a[++j],r[i+1]=a[j+k];
        j=0;
        if(n%3==0)
            s0[++j]=n;
        for(i=1;i<=n0;++i)
            if(a0[i]<k){
                a0[i]=3*a0[i]-2;
                if(a0[i]!=1)
                    s0[++j]=a0[i]-1;
            }else
                a0[i]=(a0[i]-k)*3-1;
        sot(s0,j,s,n,m,0);
        for(i=1,k=2,l=0;i<=j||k<=n0;)
            if(k>n0||i<=j&&cmp(s,n,r,s0[i],a0[k]))
                a[++l]=s0[i++];
            else
                a[++l]=a0[k++];
        for(i=1;i<=n;++i)
            r[a[i]]=i;
        del(a0);
        del(s0);
    }
};
```

6.7 Suffix Array (Factor Oracle)

Description

Use a factor oracle to construct a suffix array and it's height array from a given string. It is theoretically slow, but usually fast in practice. Object of it should be static since it has large data members.

Methods

template<class T,int N,int M,T D>SuffixArray<T,N,M,D>::SuffixArray();	
Description	construct an object of SuffixArray
Parameters	**Description**
T	type of character, usually char
N	maximum length of input string
M	size of alphabet
D	offset of alphabet, use 'a' for lowercase letters
Time complexity	$\Theta(1)$
Space complexity	$\Theta((M+13)N)$
Return value	an object of SuffixArray

template<class T,int N,int M,T D>void SuffixArray<T,N,M,D>::build(T*s,int n);	
Description	build suffix array and height array
Parameters	**Description**
s	string from which to build a suffix array, indexed from zero
n	length of s
Time complexity	$O((M+n)n)$
Space complexity	$\Theta(n)$
Return value	none

Fields

Name	**Description**
sa	suffix array, indexed from one
ht	height array, indexed from one

Performance

Problem	**Constraints**	**Time**	**Memory**	**Date**
Tyvj P1860	$N = 2\times10^5, M = 26$	1247 ms (10 cases)	33012 kB	2016-02-12

References

Title	Author
Factor oracle, Suffix oracle	Cyril Allauzen, Maxime Crochemore, Mathieu Raffinot
Computing repeated factors with a factor oracle	Arnaud Lefebvre, Thierry Lecroq

Code

Suffix Array (Factor Oracle).hpp (2640 bytes, 71 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T,int N,int M,T D>struct SuffixArray{
    int val(int i,int d){
        return d<0?(d>-2?lrs[i]:n-1-lrs[i]):s[n-i+lrs[i]+d]-D;
    }
    void sort(int*a,int*b,int m,int d){
        static int c[N];
        memset(c,0,4*(d>=0?M:n));
        for(i=1;i<=m;++c[val(a[i],d)],++i);
        for(i=1;i<(d>=0?M:n);c[i]+=c[i-1],++i);
        for(i=m;i>=1;b[c[val(a[i],d)]--]=a[i],--i);
    }
    void sort(int a,int b,int d,int l){
        sort(z+a-1,t,b-a+1,d);
        memcpy(z+a,t+1,(b-a+1)*4);
        for(i=a,j;i<=b;i=j+1){
            for(j=i;j+1<=b&&val(z[j],d)==val(z[j+1],d);++j);
            if(j-i)
                sort(i,j,d+1,l);
        }
    }
    void add(int&b,int v){
        cv[++cp]=v,cn[cp]=b,b=cp;
    }
    void dfs(int u){
        #define m(p,q)\
            for(int i=p##b[u],j;i;){\
                for(*z=0,j=i;cn[j]&&lrs[cv[j]]==lrs[cv[cn[j]]];z[++z[0]]=cv[
    j],j=cn[j]);\
            z[++z[0]]=cv[j],sort(1,*z,0,q);\
            for(z[0]=1;i!=cn[j];cv[i]=z[z[0]++],i=cn[i]);\
        }
    m(l,0)
    for(int i=lb[u];i;dfs(cv[i]),i=cn[i]);
    sa[++*sa]=n+1-u,*sa-=!u;
    m(r,1)
    for(int i=rb[u];i;dfs(cv[i]),i=cn[i]);
}
void build(T*_s,int _n){
    n=_n,s=_s,memset(tr,(cp=*sa=*vl=*vr=*lb=*rb=*lrs=0,*z=-1),4*M);
    for(int i=0,c=s[n-1-i]-D,u,v;i<n;c=s[n-1-++i]-D){
        memset(tr+i+1,(lb[i+1]=rb[i+1]=lrs[i+1]=0)-1,4*M);
        for(j=i;j>-1&&tr[j][c]<0;tr[j][c]=i+1,j=z[u=j]);
        if(v=z[i+1]=j<0?0:tr[j][c]){
            for(v=v-1==z[u]?u:v-1;z[u]!=z[v];v=z[v]);
            lrs[i+1]=min(lrs[u],lrs[v])+1;
        }
        for(j=0;n-(z[i+1]-lrs[i+1]-j)<n&&s[n-(z[i+1]-lrs[i+1]-j)]==s[
n-1-i+lrs[i+1]+j];++j);
        if(n-(z[i+1]-lrs[i+1]-j)<n&&s[n-(z[i+1]-lrs[i+1]-j)]>s[n-1-i
+lrs[i+1]+j])
            vl[++*vl]=i+1;
        else
            vr[++*vr]=i+1;
    }
    sort(vl,t,*vl,-1),sort(vr,vl,*vr,-2);
    for(i=*vl;i;add(lb[z[t[i]]],t[i]),--i);
    for(i=*vr;i;add(rb[z[vl[i]]],vl[i]),--i);
    dfs(0);
    for(i=1;i<=n;++i)
        rk[sa[i]]=i;
    for(i=1;i<=n;++i){
        if(rk[i]==1){
            ht[1]=0;
            continue;
        }
        int&d=ht[rk[i]]=max(i==1?0:ht[rk[i-1]]-1,0);
        for(;i+d<=n&&sa[rk[i]-1]+d<=n&&s[i+d-1]==s[sa[rk[i]-1]+d-1];++
d);
    }
    }
    T*s;
    int n,sa[N+1],ht[N+1],rk[N+1],lrs[N+1],tr[N+1][M],i,j,lb[N+1],rb[N+1],
    cv[N+1],cn[N+1],cp,vl[N+1],vr[N+1],t[N+1],z[N+1];
};
```

6.8 Suffix Array (Prefix-Doubling Algorithm)

Suffix Array (Prefix-Doubling Algorithm).hpp (1357 bytes, 55 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct SuffixArray{
    int*a,*h,*r,*t,*c,n,m;
    #define lp(u,v)for(int i=u;i<=v;++i)
    #define rp(u,v)for(int i=u;i>=v;--i)
    void sort(){
        memset(c+1,0,m*4);
        lp(1,n)
            ++c[r[t[i]]];
        lp(2,m)
            c[i]+=c[i-1];
        rp(n,1)
            a[c[r[t[i]]]--]=t[i];
    }
    SuffixArray(int*s){
        for(n=m=0;s[n+1];m=max(m,s[++n]));
        a=new int[4*n+max(n,m)+3];
        h=a+n;
        r=h+n+1;
        t=r+n+1;
        c=t+n;
        lp(1,n)
            t[i]=i,r[i]=s[i];
        sort();
        for(int l=1;l<=n;l<<=1,r[a[n]]==n?l=n+1:m=r[a[n]]){
            t[0]=0;
            lp(n-l+1,n)
                t[++t[0]]=i;
            lp(1,n)
                if(a[i]>l)
                    t[++t[0]]=a[i]-l;
            sort();
            swap(r,t);
            r[a[1]]=1;
            lp(2,n)
                r[a[i]]=r[a[i-1]]+(t[a[i]]!=t[a[i-1]]||a[i]+l>n||a[i-1]+l>n
    ||t[a[i]+l]!=t[a[i-1]+l]);
        }
        int l=0;
        a[0]=n+1;
        lp(1,n){
            if(r[i]==1)
                l=0;
            l-=(l>0);
            int j=a[r[i]-1];
            for(;s[i+l]==s[j+l];++l);
            h[r[i]]=l;
        }
    }
    #undef lp
    #undef rp
    ~SuffixArray(){
        delete a;
    }
};
```

6.9 Suffix Array (Suffix Tree)

Suffix Array (Suffix Tree).hpp (2849 bytes, 115 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T,int N,int M,T D>struct SuffixTree{
    struct node;
    struct edge{
        edge():
            l(0),r(0),t(0){
        }
        int length(){
            return r-l;
        }
        T*l,*r;
        node*t;
    }pe[2*N],*ep=pe;
    edge*newedge(T*l,T*r,node*t){
        ep->l=l;
        ep->r=r;
        ep->t=t;
        return ep++;
    }
    struct node{
        node():
            s(0),c({0}){
        }
        node*s;
        edge*c[M+1];
    }pn[2*N+1],*np=pn;
    SuffixTree():
        root(np++),ct(0){
    }
    void extend(T*s){
        for(;ae&&al>=ae->length();){
            s+=ae->length();
            al-=ae->length();
            an=ae->t;
            ae=al?an->c[*s-D]:0;
        }
    }
    bool extend(int c){
        if(ae){
            if(*(ae->l+al)-D-c)
                return true;
            ++al;
        }else{
            if(!an->c[c])
                return true;
            ae=an->c[c];
            al=1;
            if(pr)
                pr->s=an;
        }
        extend(ae->l);
        return false;
    }
    void dfs(node*u,int d){
        int t=0,s=0;
        for(int i=0;i<M+1;++i)
            if(u->c[i]){
                if(!t)
                    t=1;
                else if(!s){
                    s=1;
                    *sp++=d;
                }
                dfs(u->c[i]->t,d+u->c[i]->length());
            }
        if(s)
            --sp;
        else if(!t&&sp!=sk){
            *hp++=*(sp-1);
            *fp++=ct-d+1;
        }
    }
    void build(T*s,int n){
        s[n++]=M+D;
        ct+=n;
        an=root;
        ae=al=0;
        for(T*p=s;p!=s+n;++p)
            for(pr=0;extend(*p-D);){
                edge*x=newedge(p,s+n,np++);
                if(!ae)
                    an->c[*p-D]=x;
                else{
                    edge*&y=an->c[*ae->l-D];
                    y=newedge(ae->l,ae->l+al,np++);
                    y->t->c[*(ae->l+=al)-D]=ae;
                    y->t->c[*p-D]=x;
                    ae=y;
                }
                if(pr)
                    pr->s=ae?ae->t:an;
                pr=ae?ae->t:an;
                int r=1;
                if(an==root&&!al)
                    break;
                if(an==root)
                    --al;
                else{
                    an=an->s?an->s:root;
                    r=0;
                }
                if(al){
                    T*t=ae->l+(an==root)*r;
                    ae=an->c[*t-D];
                    extend(t);
                }else
                    ae=0;
            }
        dfs(root,0);
    }
    edge*ae;
    node*root,*an,*pr;
    int al,ct,sk[N],*sp=sk,ht[N],*hp=ht,sa[N],*fp=sa;
};
```

6.10 Suffix Array (Treap)

Suffix Array (Treap).hpp (3803 bytes, 147 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct SuffixArray{
    struct node{
        node*c[2],*p;
        T v;
        int f,s,l,h,m;
        double t;
        node(node*_p,T _v,int _l):
            f(rand()*1.0/RAND_MAX*1e9),p(_p),v(_v),s(1),l(_l),h(0),m(0),t(5
    e8){
            c[0]=c[1]=0;
        }
    }*root;
    vector<T>a;
    SuffixArray():
        root(new node(0,0,0)),a(1){
    }
    ~SuffixArray(){
        clear(root);
    }
    void relabel(node*x,double l,double r){
        x->t=(l+r)/2;
        if(x->c[0])
            relabel(x->c[0],l,x->t);
        if(x->c[1])
            relabel(x->c[1],x->t,r);
    }
    void update(node*x){
        x->s=1;
        x->m=x->h;
        for(int i=0;i<2;++i)
            if(x->c[i])
                x->s+=x->c[i]->s,x->m=min(x->m,x->c[i]->m);
    }
    void rotate(node*&x,int d){
        node*y=x->c[d];
        x->c[d]=y->c[!d];
        y->c[!d]=x;
        y->s=x->s;
        y->m=x->m;
        update(x);
        x=y;
    }
    void clear(node*x){
        if(!x)
            return;
        clear(x->c[0]);
        clear(x->c[1]);
        delete x;
    }
    node*insert(node*&x,node*p,T v,node*l,node*r){
        int d=x->v!=v?x->v<v:x->p->t<p->t;
        double tl=l?l->t:0,tr=r?r->t:1e9;
        node*y;
        if(d)
            l=x;
        else
            r=x;
        if(!x->c[d]){
            y=new node(p,v,p->l+1);
            y->t=((l?l->t:0)+(r?r->t:1e9))/2;
            y->m=y->h=l->v==y->v?lcp(l->p,y->p)+1:0;
            if(r)
                r->h=r->v==y->v?lcp(r->p,y->p)+1:0;
            x->c[d]=y;
        }else
            y=insert(x->c[d],p,v,l,r);
        update(x);
        if(x->c[d]->f>x->f)
            rotate(x,d),relabel(x,tl,tr);
        return y;
    }
    node*insert(node*p,T v){
        a.push_back(v);
        return insert(root,p,v,0,0);
    }
    void erase(node*&x,node*y){
        if(x==y){
            if(!x->c[0]){
                x=x->c[1];
                delete y;
            }else if(!x->c[1]){
                x=x->c[0];
                delete y;
            }else{
                int d=x->c[0]->f<x->c[1]->f;
                rotate(x,d);
                erase(x->c[!d],y);
                --x->s;
            }
        }else
            erase(x->c[x->t<y->t],y),update(x);
    }
    void erase(node*y){
        erase(root,y);
        a.pop_back();
    }
    bool check(node*x,T*y,node*&p,int&l){
        if(p){
            int t=x->c[p->t>x->t]?x->c[p->t>x->t]->m:~0u>>1;
            if(p->t>x->t)
                t=min(t,p->h);
            else
                t=min(t,x->h);
            if(t<l)
                return x->t<p->t;
        }
        for(p=x;l+1<=x->l&&y[l+1];++l)
            if(a[x->l-l]!=y[l+1])
                return a[x->l-l]<y[l+1];
        return y[l+1]!=0;
    }
    int count(node*x,T*y){
        int r=0,l=0;
        for(node*p=0;x;)
            if(check(x,y,p,l))
                r+=(x->c[0]?x->c[0]->s:0)+1,x=x->c[1];
            else
                x=x->c[0];
        return r;
    }
    int count(T*y){
        T*t=y;
        while(*(t+1))
            ++t;
        int r=-count(root,y);
        ++*t;
        r+=count(root,y);
        --*t;
        return r;
    }
    int lcp(node*x,double u,double v,double l,double r){
        if(v<l||u>r||!x)
            return ~0u>>1;
        if(u<l&&v>=r)
            return x->m;
        int t=u<x->t&&v>=x->t?x->h:~0u>>1;
        t=min(t,lcp(x->c[0],u,v,l,x->t));
        t=min(t,lcp(x->c[1],u,v,x->t,r));
        return t;
    }
    int lcp(node*x,node*y){
        if(x->t>y->t)
            swap(x,y);
        return lcp(root,x->t,y->t,0,1e9);
    }
};
```

6.11 Suffix Automaton

Suffix Automaton.hpp (1694 bytes, 59 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T>struct SuffixAutomaton{
    struct node{
        node(vector<node*>&all,int m,node*_pr=0,int _ln=0,T _va=T()):
            pr(_pr),tr(m),ln(_ln),va(_va){
            all.push_back(this);
        }
        T va;
        int ln;
        node*pr;
        vector<node*>tr;
    };
    SuffixAutomaton(int _m):
        root(new node(all,m)),m(_m){
    }
    ~SuffixAutomaton(){
        for(int i=0;i<all.size();++i)
            delete all[i];
    }
    node*insert(node*lst,int c,T v){
        node*p=lst,*np=p->tr[c]?0:new node(all,m,0,lst->ln+1,v);
        for(;p&&!p->tr[c];p=p->pr)
            p->tr[c]=np;
        if(!p)np->pr=root;
        else{
            node*q=p->tr[c];
            if(p==lst)
                np=q;
            if(q->ln==p->ln+1)
                p==lst?(q->va+=v):(np->pr=q,0);
            else{
                node*nq=new node(all,m,q->pr,p->ln+1,p==lst?v:T());
                nq->tr=q->tr;
                q->pr=np->pr=nq;
                if(p==lst)
                    np=nq;
                for(;p&&p->tr[c]==q;p=p->pr)
                    p->tr[c]=nq;
            }
        }
        return np;
    }
    void count(){
        vector<int>cnt(all.size());
        vector<node*>tmp=all;
        for(int i=0;i<tmp.size();++i)
            ++cnt[tmp[i]->ln];
        for(int i=1;i<cnt.size();++i)
            cnt[i]+=cnt[i-1];
        for(int i=0;i<tmp.size();++i)
            all[--cnt[tmp[i]->ln]]=tmp[i];
        for(int i=int(all.size())-1;i>0;--i)
            all[i]->pr->va+=all[i]->va;
    }
    int m;
    node*root;
    vector<node*>all;
```

```
};
```

6.12 Suffix Tree

Suffix Tree.hpp (2296 bytes, 94 lines)

```
#include<bits/stdc++.h>
using namespace std;
template<class T,int N,int M,T D>struct SuffixTree{
    struct node;
    struct edge{
        edge():
            l(0),r(0),t(0){
        }
        int length(){
            return r-l;
        }
        T*l,*r;
        node*t;
    }pe[2*N],*ep=pe;
    edge*newedge(T*l,T*r,node*t){
        ep->l=l;
        ep->r=r;
        ep->t=t;
        return ep++;
    }
    struct node{
        node():
            s(0),c({0}){
        }
        node*s;
        edge*c[M];
    }pn[2*N+1],*np=pn;
    SuffixTree():
        root(np++),ct(0){
    }
    void extend(T*s){
        for(;ae&&al>=ae->length();){
            s+=ae->length();
            al-=ae->length();
            an=ae->t;
            ae=al?an->c[*s-D]:0;
        }
    }
    bool extend(int c){
        if(ae){
            if(*(ae->l+al)-D-c)
                return true;
            ++al;
        }else{
            if(!an->c[c])
                return true;
            ae=an->c[c];
            al=1;
            if(pr)
                pr->s=an;
        }
        extend(ae->l);
        return false;
    }
    void insert(T*s,int n){
        ct+=n;
        an=root;
        ae=al=0;
        for(T*p=s;p!=s+n;++p)
            for(pr=0;extend(*p-D);){
                edge*x=newedge(p,s+n,np++);
                if(!ae)
                    an->c[*p-D]=x;
                else{
                    edge*&y=an->c[*ae->l-D];
                    y=newedge(ae->l,ae->l+al,np++);
                    y->t->c[*(ae->l+=al)-D]=ae;
                    y->t->c[*p-D]=x;
                    ae=y;
                }
                if(pr)
                    pr->s=ae?ae->t:an;
                pr=ae?ae->t:an;
                int r=1;
                if(an==root&&!al)
                    break;
                if(an==root)
                    --al;
                else{
                    an=an->s?an->s:root;
                    r=0;
                }
                if(al){
                    T*t=ae->l+(an==root)*r;
                    ae=an->c[*t-D];
                    extend(t);
                }else
                    ae=0;
            }
    }
    edge*ae;
    int al,ct;
    node*root,*an,*pr;
};
```

CHAPTER 7

Utility Tools

7.1 Checker

Checker.bat (166 bytes, 7 lines)

```
:again
generator > input.txt
program1 < input.txt > output1.txt
program2 < input.txt > output2.txt
fc output1.txt output2.txt
if errorlevel 1 pause
goto again
```

7.2 Date

Date.hpp (3596 bytes, 145 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct Date{
    int y,m,d,w;
    Date&operator++(){
        return*this=*this+1;
    }
    bool leap(int a)const{
        return a%400==0||(a%4==0&&a%100!=0);
    }
    int month_sum(int a,int b)const{
        if(b==0)
            return 0;
        if(b==1)
            return 31;
        return 59+leap(a)+30*(b-2)+(b+1)/2-1+(b>=8&&b%2==0);
    }
    string month_name(int a)const{
        if(a==1)
            return"January";
        if(a==2)
            return"February";
        if(a==3)
            return"March";
        if(a==4)
            return"April";
        if(a==5)
            return"May";
        if(a==6)
            return"June";
        if(a==7)
            return"July";
        if(a==8)
            return"August";
        if(a==9)
            return"September";
        if(a==10)
            return"October";
        if(a==11)
            return"November";
        if(a==12)
            return"December";
    }
    string day_name(int a)const{
        if(a==0)
            return"Sunday";
        if(a==1)
            return"Monday";
        if(a==2)
            return"Tuesday";
        if(a==3)
            return"Wednesday";
        if(a==4)
            return"Thursday";
        if(a==5)
            return"Friday";
        if(a==6)
            return"Saturday";
    }
    operator int()const{
        int t=(y-1)*365+(y-1)/4-(y-1)/100+(y-1)/400+month_sum(y,m-1)+d;
        if(y==1752&&m>=9&&d>2||y>1752)
            t-=11;
        t-=min(y-1,1700)/400-min(y-1,1700)/100;
        if(y<=1700&&y%400!=0&&y%100==0&&m>2)
            ++t;
        return t;
    }
    Date(int _y,int _m,int _d):
        y(_y),m(_m),d(_d),w((int(*this)+5)%7){
    }
    Date(int a){
        int yl=0,yr=1e7;
        while(yl+1<yr){
            int ym=(yl+yr)/2;
            if(int(Date(ym,12,31))<a)
                yl=ym;
            else
                yr=ym;
        }
        y=yr;
        int ml=0,mr=12;
        while(ml+1<mr){
            int mm=(ml+mr)/2,mt;
            if(mm==2){
                if(y<=1700)
                    mt=28+(y%4==0);
                else
                    mt=28+(y%4==0&&y%100!=0||y%400==0);
            }else if(mm<=7)
                mt=30+mm%2;
            else
                mt=31-mm%2;
            if(int(Date(y,mm,mt))<a)
                ml=mm;
            else
                mr=mm;
        }
        m=mr;
        for(int i=1;;++i){
            if(y==1752&&m==9&&i>2&&i<14)
                continue;
            if(int(Date(y,m,i))==a){
                d=i;
                break;
            }
        }
        w=(5+a)%7;
    }
    operator string()const{
        stringstream s;
        string t;
        s<<day_name(w)+", "+month_name(m)+" "<<d<<", "<<y;
        getline(s,t);
        return t;
    }
};
ostream&operator<<(ostream&s,const Date&a){
    return s<<string(a);
}
int operator-(const Date&a,const Date&b){
    return int(a)-int(b);
}
Date operator+(const Date&a,int b){
    return Date(int(a)+b);
}
Date operator-(const Date&a,int b){
    return Date(int(a)-b);
}
bool operator<(const Date&a,const Date&b){
    if(a.y==b.y&&a.m==b.m)
        return a.d<b.d;
    if(a.y==b.y)
        return a.m<b.m;
    return a.y<b.y;
}
bool operator>(const Date&a,const Date&b){
    return b<a;
}
bool operator!=(const Date&a,const Date&b){
    return a.y!=b.y||a.m!=b.m||a.d!=b.d;
}
bool operator==(const Date&a,const Date&b){
    return !(a!=b);
}
```

7.3 Fast Reader

Fast Reader.hpp (1251 bytes, 61 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct FastReader{
    FILE*f;
    char*p,*e;
    vector<char>v;
    void ipt(){
        for(int i=1,t;;i<<=1){
            v.resize(v.size()+i);
            if(i!=(t=fread(&v[0]+v.size()-i,1,i,f))){
                p=&v[0],e=p+v.size()-i+t;
                break;
            }
        }
    }
    void ign(){
        while(p!=e&&isspace(*p))
            ++p;
    }
    int isc(){
        return p!=e&&!isspace(*p);
    }
    int isd(){
        return p!=e&&isdigit(*p);
    }
    FastReader(FILE*_f):
        f(_f){
        ipt();
    }
    FastReader(string _f):
        f(fopen(_f.c_str(),"r")){
        ipt();
    }
    ~FastReader(){
        fclose(f);
    }
    template<class T>FastReader&operator>>(T&a){
        int n=1;
        ign();
        if(*p=='-')
            n=-1,++p;
        for(a=0;isd();)
            a=a*10+*p++-'0';
        a*=n;
        return*this;
    }
    FastReader&operator>>(char&a){
        ign();
        a=*p++;
        return*this;
    }
    FastReader&operator>>(char*a){
        for(ign();isc();)
            *a++=*p++;
        *a=0;
        return*this;
    }
    char get(){
        return*p++;
    }
};
```

7.4 Fast Writer

Fast Writer.hpp (866 bytes, 39 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct FastWriter{
    FILE*f;
    vector<char>p;
    FastWriter(FILE*_f):
        f(_f){
    }
    FastWriter(string _f):
        f(fopen(_f.c_str(),"w")){
    }
    ~FastWriter(){
        if(p.size())
            fwrite(&p[0],1,p.size(),f);
        fclose(f);
    }
    FastWriter&operator<<(char a){
        p.push_back(a);
        return*this;
    }
    FastWriter&operator<<(const char*a){
        while(*a)
            p.push_back(*a++);
        return*this;
    }
    template<class T>FastWriter&operator<<(T a){
        if(a<0)
            p.push_back('-'),a=-a;
        static char t[19];
        char*q=t;
        do{
            T b=a/10;
            *q++=a-b*10+'0',a=b;
        }while(a);
        while(q>t)
            p.push_back(*--q);
        return*this;
    }
};
```

7.5 Number Speller

Number Speller.hpp (2143 bytes, 72 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace NumberSpeller{
    template<class T>string run(T a){
        map<T,string>m;
        m[0]="zero";
        m[1]="one";
        m[2]="two";
        m[3]="three";
        m[4]="four";
        m[5]="five";
        m[6]="six";
        m[7]="seven";
        m[8]="eight";
        m[9]="nine";
        m[10]="ten";
        m[11]="eleven";
        m[12]="twelve";
        m[13]="thirteen";
        m[14]="fourteen";
        m[15]="fifteen";
        m[16]="sixteen";
        m[17]="seventeen";
        m[18]="eighteen";
        m[19]="nineteen";
        m[20]="twenty";
        m[30]="thirty";
        m[40]="forty";
        m[50]="fifty";
        m[60]="sixty";
        m[70]="seventy";
        m[80]="eighty";
        m[90]="ninety";
        if(a<0)
            return"minus "+run(-a);
        if(m.count(a))
            return m[a];
        if(a<100)
            return run(a/10*10)+"-"+run(a%10);
        if(a<1000&&a%100==0)
            return run(a/100)+" hundred";
        if(a<1000)
            return run(a/100*100)+" and "+run(a%100);
        vector<string>t;
        t.push_back("thousand");
        t.push_back("million");
        t.push_back("billion");
        t.push_back("trillion");
        t.push_back("quadrillion");
        t.push_back("quintillion");
        t.push_back("sextillion");
        t.push_back("septillion");
        t.push_back("octillion");
        t.push_back("nonillion");
        t.push_back("decillion");
        t.push_back("undecillion");
        t.push_back("duodecillion");
        t.push_back("tredecillion");
        t.push_back("quattuordecillion");
        t.push_back("quindecillion");
        string r=a%1000?run(a%1000):"";
        a/=1000;
        for(int i=0;a;++i,a/=1000)
            if(a%1000){
                if(!i&&r.find("and")==string::npos&&r.find("hundred")==
    string::npos&&r.size())
                    r=run(a%1000)+" "+t[i]+" and "+r;
                else
                    r=run(a%1000)+" "+t[i]+(r.size()?", ":"")+r;
            }
        return r;
    }
}
```