Algorithms

Algorithms by Yu Dongfeng
First version on April 12, 2013
Latest version on October 10, 2016

Contents

CHAPTER 1

Computational Geometry

1.1 Convex Hull 2D

Description

Calculate the convex hull of a given 2D point set.

Methods

template<class T>ConvexHull2D();	
Description	construct an object of ConvexHull2D
Parameters	**Description**
T	type of coordinates
Time complexity	$\Theta(1)$
Space complexity	$\Theta(1)$
Return value	an object of ConvexHull2D

void add(T x,T y);	
Description	add a point
Parameters	**Description**
x	x-coordinate of the point
y	y-coordinate of the point
Time complexity	$\Theta(1)$ (amortized)
Space complexity	$\Theta(1)$ (amortized)
Return value	none

vector<pair<T,T> >run(T d);	
Description	calculate the convex hull
Parameters	**Description**
d	$d = 1$ for upper hull and $d = -1$ for lower hull
Time complexity	$\Theta(n \log n)$ (n is the number of points)
Space complexity	$\Theta(n)$
Return value	result in a vector<pair<T,T> >

Code

Convex Hull 2D.hpp (988 bytes, 38 lines)

```
#include<vector>
using namespace std;
template<class T>struct ConvexHull2D{
    struct point{
        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;
        }
        T x,y;
    };
    T chk(point a,point b,point c){
        return (a-c)*(b-c);
    }
    void add(T x,T y){
        a.push_back(point(x,y));
    }
    struct cmp{
        cmp(T _d):d(_d){}
        bool operator()(point a,point b){
            return a.x!=b.x?a.x<b.x:a.y*d<b.y*d;
        }
        T d;
    };
    vector<pair<T,T> >run(T d){
        sort(a.begin(),a.end(),cmp(d));
        vector<pair<T,T> >r;
        for(int i=0;i<a.size();++i){
            while(r.size()>1&&chk(a[i],r.back(),r[r.size()-2])*d<=0)
                r.pop_back();
            r.push_back(make_pair(a[i].x,a[i].y)),
        }
        return r;
    }
    vector<point>a;
};
```

1.2 Convex Hull 3D

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Convex Hull 3D.hpp (0 bytes, 0 lines)

1.3 Delaunay Triangulation

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
```

1.4 Dynamic Convex Hull (Set)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
```

1.5 Dynamic Convex Hull (Square Root Decomposition)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Dynamic Convex Hull (Square Root Decomposition).hpp (0 bytes, 0 lines)

1.6 Dynamic Convex Hull (Treap)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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{
```

```
20              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;
30          }
            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;
            }
40      };
        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;
50          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]);
            }
```

```
60          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);
                }
70              *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;}
80              *(sp++)=x;
            }
        }me;
        struct array{
            hull**ps,**pp;
            int pm;
            array():
                ps((hull**)malloc(sizeof(hull*))),pp(ps),pm(1){
            }
            ~array(){
90              free(ps);
            }
            int size(){
                return pp-ps;
            }
            hull*operator[](int i){
                return ps[i];
            }
            void push(hull*x){
                if(pp==ps+pm){
100                 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;
```

1.7 Dynamic Farthest Pair

动态插点的最远点对 kd 树需要补替罪羊否则只能插随机的点 ans 存的是距离平方静态内存加速，请用 static 定义注意如果点集形成一个圆，查询效率是很糟糕的

Dynamic Farthest Pair.hpp (1545 bytes, 54 lines)

```
#include<bits/stdc++.h>
using namespace std;
const int N=1000000;
struct DynamicFarthestPair{
    struct node{
        int x,y,x0,y0,x1,y1;
```

```
    node*c[2];
}*root,pool[N],*ptr;
long long ans;
node*make(int x,int y){
    ptr->c[0]=ptr->c[1]=0;
    ptr->x=ptr->x0=ptr->x1=x;
    ptr->y=ptr->y0=ptr->y1=y;
    return ptr++;
}
DynamicFarthestPair():
    ans(0),root(0),ptr(pool){}
void insert(node*&u,int x,int y,int d){
    if(u){
        u->x0=min(u->x0,x);
        u->x1=max(u->x1,x);
        u->y0=min(u->y0,y);
        u->y1=max(u->y1,y);
        insert(u->c[d&&y>u->y||!d&&x>u->x],x,y,1-d);
    }else
        u=make(x,y);
}
long long dist(long long x1,long long y1,int x2,int y2){
    return (x1-x2)*(x1-x2)+(y1-y2)*(y1-y2);
}
long long estim(node*u,int x,int y){
    if(u){
        long long p=max(dist(u->x0,u->y0,x,y),dist(u->x1,u->y0,x,y)),
            q=max(dist(u->x0,u->y1,x,y),dist(u->x1,u->y1,x,y));
        return max(p,q);
    } else
        return 0;
}
void query(node*u,int x,int y){
    ans=max(ans,dist(u->x,u->y,x,y));
    long long e[2];
    for(int i=0;i<2;++i)
        e[i]=estim(u->c[i],x,y);
    int d=e[0]<e[1];
    if(e[d]>ans)
        query(u->c[d],x,y);
    if(e[!d]>ans)
```

```
            query(u->c[!d],x,y);
        }
        void insert(int x,int y){
            insert(root,x,y,0);
            query(root,x,y);
        }
};
```

1.8 Geometry 2D

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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){
```

```
27              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);
        }
37      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){
47          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);
57          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);
67      }
```

```
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){
117     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){
127     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)
137         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{
147     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){
157     return !cmp(det(a-c,b-c),T(0));
    }
}
```

1.9 Geometry 3D

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Geometry 3D.hpp (0 bytes, 0 lines)

1.10 Half-Plane Intersection

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Half-Plane Intersection.hpp (1953 bytes, 71 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;
10  };
    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.back(),ans[ans.size()-2],ans.front())
    )
            ans.pop_back();
        if(ans.size()<3)
            ans.clear();
        return ans;
    }
}
```

1.11 Half-Space Intersection

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Half-Space Intersection.hpp (0 bytes, 0 lines)

1.12 Point Location (Trapezoidal Decomposition)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Point Location (Trapezoidal Decomposition).hpp (0 bytes, 0 lines)

1.13 Point Location (Treap)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Point Location (Treap).hpp (0 bytes, 0 lines)

1.14 Voronoi Diagram

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Voronoi Diagram.hpp (873 bytes, 8 lines)

年第届
200529ACM 世界总决赛的试题解析ICPC
wuyingying 2006-04-02 3 查看评论0 公开原文添加收藏 去年在上海举办时我已经退役，所以没有参加。不过我当时看了比赛的直播，并也看了一下题目。我刚刚又重新看了一下题目，写了一篇对题目算法的简要分析，在此与大家探讨。
Final 本题是一个典型的综合题。模型本身是一个最短路，但加入了计算几何的背景。最短路的计算对于参加

World 的选手自然是小菜一碟，但此题中每条边的是什么？由题目的描述可知，每条边的是穿过的的个数。那么由什么来划分？每个对应的就是一个离这个比离其它都要近的区域。这样的区域是什么？学过计算几何的可能知道，所有的构成的一个平面划分是一个图，图可以在FinalcostcostcellcelltowercelltowertowercellVoronoiVoronoiO(nlogn)的时间里求出。所以本题划分为两个步骤：一是求出图并计算每条边的费用，二是计算最短路。Voronoi 难点在于，图的计算非常繁琐，我相信没有一支队愿意在比赛中写一个求图的程序，所以我们要换一种方法。关键在于，怎样判断一个线段是否穿过了一个？每个都是一些由中垂线围成的凸多边形，如果穿过了这个，就必然会有交点，而这个交点一定是该线段和某条中垂线的交点。所以问题立刻变得简单：要计算线段是否穿过
VoronoiVoronoicellcellcellABtower 的，只需要枚举的中垂线和的交点，再判断这个交点是否离比所有其它都近，如果存在这样的交点，则穿过的。
PicellPiPjABPiPjABPicell 这样子我们只需要一个求线段交点的即可，比起求图，编程复杂度大大下降。而算法的时间复杂度也是可以接受的。
routineVoronoi 说是一个模式识别，但图像可以放大，非常不好处理。可以根据中平行线段间的距离来确定放大的倍数，然后再到图里面进行枚举匹配。但不仅要分情况讨论，还要注意精度。是一道算法和编程都十分繁琐的题目。
pattern

CHAPTER 2

Data Structures

2.1 Discretization

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Discretization.hpp (511 bytes, 16 lines)

```
#ifndef DISCRETIZATION
#define DISCRETIZATION
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    template<class T>struct Discretization{
        vector<T>a;
        void add(T v){a.push_back(v);}
        void build(){
            sort(begin(a),end(a));
            a.erase(unique(begin(a),end(a)),end(a));}
        int order(T v){
            return lower_bound(begin(a),end(a),v)-begin(a)+1;}
        int size(){return a.size();}
        T value(int v){return a[v-1];}};}
#endif
```

2.2 Dynamic Cactus

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Dynamic Cactus.hpp (14822 bytes, 608 lines)

```
#include<iostream>//2015-5-7 版本？仙人掌差评+1 = =
#include<cstdio>
#include<cmath>
#include<algorithm>
#include<queue>
#include<cstring>
#define PAU putchar(' ')
#define ENT putchar('\n')
#define MAXN 50005
#define MAXM 250005
#define is_NULL_tag(x) ((x)==0)
```

```
#define is_NULL_info(x) (x.size==0)
using namespace std;
inline int read(){
    int x=0,sig=1;char ch=getchar();
    while(!isdigit(ch)){if(ch=='-')sig=-1;ch=getchar();}
    while(isdigit(ch))x=10*x+ch-'0',ch=getchar();
    return x*=sig;
}
inline void write(int x){
    if(x==0){putchar('0');return;}if(x<0)putchar('-'),x=-x;
    int len=0,buf[15];while(x)buf[len++]=x%10,x/=10;
    for(int i=len-1;i>=0;i--)putchar(buf[i]+'0');return;
}
char ch;
inline void Pass_Pau(int x){while(x--) getchar();return;}
int n,Q;
struct Info{
    int mi,size;
    long long sum;
};
const int NULL_TAG=0;
const Info NULL_INFO=(Info){2147483647,0,0};
inline Info operator + (const Info &a,const Info &b){return (Info){std::min
    (a.mi,b.mi),a.size+b.size,a.sum+b.sum};}
inline Info operator * (const Info &a,const int &b){return a.size ? (Info){
    a.mi+b,a.size,a.sum+1LL*a.size*b}: a;}
struct splay_node{
    splay_node *ch[2],*fa;
    Info x,sum;
    int tag,tag_sum;
    inline void add_tag(int t){
        x=x*t;sum=sum*t;
        tag=tag+t;tag_sum=tag_sum+t;
        return;
    }
    inline void down(){
        if(is_NULL_tag(tag)) return;
        if(ch[0]) ch[0]->add_tag(tag);
        if(ch[1]) ch[1]->add_tag(tag);
        tag=NULL_TAG;
        return;
```

```
    }
    inline void update(){
        sum=x;
        if(ch[0]) sum=sum+ch[0]->sum;
        if(ch[1]) sum=sum+ch[1]->sum;
        return;
    }
};
splay_node _splay[MAXN+MAXM];
inline int get_parent(splay_node *x,splay_node *&fa){return (fa=x->fa) ? fa
    ->ch[1]==x : -1;}//把父亲扔到里同时返回
    值fad
inline void rotate(splay_node *x){
    splay_node *fa,*gfa;
    int t1,t2;
    t1=get_parent(x,fa);
    t2=get_parent(fa,gfa);
    if((fa->ch[t1]=x->ch[t1^1])) fa->ch[t1]->fa=fa;
    fa->fa=x;x->fa=gfa;x->ch[t1^1]=fa;
    if(t2!=-1) gfa->ch[t2]=x;
    fa->update();
    return;
}
inline void pushdown(splay_node *x){
    static splay_node *stack[MAXN+MAXM];
    int cnt=0;
    while(x) stack[cnt++]=x,x=x->fa;
    while(cnt--) stack[cnt]->down();
    return;
}
inline splay_node * splay(splay_node *x){
    pushdown(x);
    while(1){
        splay_node *fa,*gfa;
        int t1,t2;
        t1=get_parent(x,fa);
        if(t1==-1) break;
        t2=get_parent(fa,gfa);
        if(t2==-1){
            rotate(x);break;
        }else if(t1==t2){
```

```
            rotate(fa);rotate(x);
        }else{
            rotate(x);rotate(x);
        };
    }
    x->update();
    return x;
}
inline splay_node * join(splay_node *a,splay_node *b){
    if(!a) return b;
    if(!b) return a;
    while(a->ch[1]) a->down(),a=a->ch[1];
    splay(a)->ch[1]=b;b->fa=a;
    a->update();
    return a;
}
struct lcc_node;
struct cycle{
    int A,B;
    lcc_node *ex;
};
struct lcc_node{
    lcc_node *ch[2],*fa;
    lcc_node *first,*last;
    bool rev;
    bool isedge;
    bool mpath;
    bool hasmpath;
    bool mpathtag;
    bool hasmpathtag;
    bool hascyctag;
    bool hascyc;
    cycle *cyc;
    cycle *cyctag;
    int totlen;
    int len;
    int size;
    Info x,sum,sub,ex,all;
    int chain_tag,sub_tag,ex_tag_sum;
    inline void add_rev_tag(){
        std::swap(ch[0],ch[1]);
```

```
        std::swap(first,last);
        rev^=1;
        return;
    }
    inline void add_cyc_tag(cycle *c){
        if(isedge) cyc=c;
        cyctag=c;
        hascyctag=1;
        hascyc=c;
        return;
    }
    inline void add_mpath_tag(bool t){
        mpathtag=t;
        hasmpathtag=1;
        mpath=t&isedge;
        hasmpath=t&(isedge|(size>1));
        return;
    }
    inline void add_chain_tag(int t)
    {
        if(is_NULL_tag(t)) return;
        x=x*t;sum=sum*t;
        chain_tag=chain_tag+t;
        all=sum+sub;
        return;
    };
    inline void add_sub_tag(int t);
    inline void down(){
        if(rev){
            if(ch[0]) ch[0]->add_rev_tag();
            if(ch[1]) ch[1]->add_rev_tag();
            rev=0;
        }
        if(hascyctag){
            if(ch[0]) ch[0]->add_cyc_tag(cyctag);
            if(ch[1]) ch[1]->add_cyc_tag(cyctag);
            hascyctag=0;
        }
        if(hasmpathtag){
            if(ch[0]) ch[0]->add_mpath_tag(mpathtag);
            if(ch[1]) ch[1]->add_mpath_tag(mpathtag);
```

```
            hasmpathtag=0;
        }
        if(!is_NULL_tag(chain_tag)){
            if(ch[0]) ch[0]->add_chain_tag(chain_tag);
            if(ch[1]) ch[1]->add_chain_tag(chain_tag);
            chain_tag=NULL_TAG;
        }
        if(!is_NULL_tag(sub_tag)){
            if(ch[0]) ch[0]->add_sub_tag(sub_tag);
            if(ch[1]) ch[1]->add_sub_tag(sub_tag);
            sub_tag=NULL_TAG;
        }
        return;
    }
    inline void update();
};
lcc_node lcc[MAXN+MAXM];
lcc_node *_node_tot;
splay_node *splay_root[MAXN+MAXM];
inline void lcc_node::add_sub_tag(int t){
    if(is_NULL_tag(t)) return;
    sub=sub*t;ex=ex*t;
    sub_tag=sub_tag+t;
    ex_tag_sum=ex_tag_sum+t;
    all=sum+sub;
    // add tag to splay_root
    int id=this-lcc;
    if(splay_root[id]){
        splay_root[id]->add_tag(t);
    }
}
inline void lcc_node::update(){
    totlen=len;
    hascyc=cyc;
    size=1;
    hasmpath=mpath;
    if(ch[0]) totlen+=ch[0]->totlen,hascyc|=ch[0]->hascyc,size+=ch[0]->
    size,hasmpath|=ch[0]->hasmpath;
    if(ch[1]) totlen+=ch[1]->totlen,hascyc|=ch[1]->hascyc,size+=ch[1]->
    size,hasmpath|=ch[1]->hasmpath;
    first=ch[0]?ch[0]->first:this;
```

```
        last=ch[1]?ch[1]->last:this;
        bool s0=ch[0],s1=ch[1];
        if(isedge){
            if(is_NULL_info(ex)){
215             if(s0 && s1){
                    sum=ch[0]->sum+ch[1]->sum;
                    sub=ch[0]->sub+ch[1]->sub;
                }else if(s0){
                    sum=ch[0]->sum;
                    sub=ch[0]->sub;
                }else if(ch[1]){
                    sum=ch[1]->sum;
                    sub=ch[1]->sub;
                }else{
225                 sub=sum=NULL_INFO;
                }
            }else{
                if(s0 && s1){
                    sum=ch[0]->sum+ch[1]->sum;
                    sub=ch[0]->sub+ch[1]->sub+ex;
                }else if(s0){
                    sum=ch[0]->sum;
                    sub=ch[0]->sub+ex;
                }else if(ch[1]){
235                 sum=ch[1]->sum;
                    sub=ch[1]->sub+ex;
                }else{
                    sum=NULL_INFO;
                    sub=ex;
                }
            }
        }else{
            splay_node *root=splay_root[this->lcc];
            if(root){
245             if(s0 && s1){
                    sum=ch[0]->sum+ch[1]->sum+x;
                    sub=ch[0]->sub+ch[1]->sub+root->sum;
                }else if(s0){
                    sum=ch[0]->sum+x;
                    sub=ch[0]->sub+root->sum;
                }else if(ch[1]){
```

```
                sum=ch[1]->sum+x;
                sub=ch[1]->sub+root->sum;
            }else{
                sub=root->sum;
                sum=x;
            }
        }else{
            if(s0 && s1){
                sum=ch[0]->sum+ch[1]->sum+x;
                sub=ch[0]->sub+ch[1]->sub;
            }else if(s0){
                sum=ch[0]->sum+x;
                sub=ch[0]->sub;
            }else if(ch[1]){
                sum=ch[1]->sum+x;
                sub=ch[1]->sub;
            }else{
                sum=x;
                sub=NULL_INFO;
            }
        }
    }
    all=sum+sub;
    return;
};
inline lcc_node * new_edge_node(int u,int v,int len){
    lcc_node *ret=++_node_tot;
    ret->ch[0]=ret->ch[1]=ret->fa=NULL;
    ret->first=ret->last=ret;
    ret->rev=0;
    ret->isedge=1;
    ret->hascyctag=ret->hascyc=0;
    ret->cyc=ret->cyctag=NULL;
    ret->totlen=ret->len=len;
    ret->size=1;
    ret->x=ret->sum=ret->sub=ret->ex=ret->all=NULL_INFO;
    ret->chain_tag=ret->sub_tag=ret->ex_tag_sum=NULL_TAG;
    return ret;
}
inline int get_parent(lcc_node *x,lcc_node *&fa){return (fa=x->fa) ? fa->
    ch[0]==x?0:fa->ch[1]==x?1:-1 : -1;}
```

```
inline void rotate(lcc_node *x){
    int t1,t2;
    lcc_node *fa,*gfa;
    t1=get_parent(x,fa);
    t2=get_parent(fa,gfa);
    if((fa->ch[t1]=x->ch[t1^1])) fa->ch[t1]->fa=fa;
    fa->fa=x;x->fa=gfa;x->ch[t1^1]=fa;
    if(t2!=-1) gfa->ch[t2]=x;
    fa->update();
    return;
}
inline void pushdown(lcc_node *x){
    static lcc_node *stack[MAXN+MAXM];
    int cnt=0;
    while(1){
        stack[cnt++]=x;
        lcc_node *fa=x->fa;
        if(!fa || (fa->ch[0]!=x && fa->ch[1]!=x)) break;
        x=fa;
    }
    while(cnt--) stack[cnt]->down();
    return;
}
inline lcc_node * splay(lcc_node *x){
    pushdown(x);
    while(1){
        int t1,t2;
        lcc_node *fa,*gfa;
        t1=get_parent(x,fa);
        if(t1==-1) break;
        t2=get_parent(fa,gfa);
        if(t2==-1){
            rotate(x);break;
        }else if(t1==t2){
            rotate(fa);rotate(x);
        }else{
            rotate(x);rotate(x);
        }
    }
    x->update();
    return x;
```

```
    }
    inline int getrank(lcc_node *x){
335     splay(x);
        return 1+(x->ch[0]?x->ch[0]->size:0);
    }
    bool _attached[MAXN+MAXM];
    inline void detach_rch(lcc_node *x){
        if(!x->ch[1]) return;
        int X=x->lcc;
        int id=x->ch[1]->first->lcc;
        _attached[id]=1;
        splay_node *p=_splay+id;
345     p->ch[0]=splay_root[X];
        if(splay_root[X]) splay_root[X]->fa=p;
        p->ch[1]=p->fa=NULL;
        p->x=x->ch[1]->all;
        p->tag=p->tag_sum=NULL_TAG;
        p->update();
        splay_root[X]=p;
        x->ch[1]=NULL;
        return;
    }
355 inline void attach_rch(lcc_node *x,lcc_node *y,int id){
        int X=x->lcc;
        _attached[id]=0;
        splay_node *p=_splay+id;
        splay(p);
        if(p->ch[0]) p->ch[0]->fa=NULL;
        if(p->ch[1]) p->ch[1]->fa=NULL;
        splay_root[X]=join(p->ch[0],p->ch[1]);
        y->add_chain_tag(p->tag_sum);
        y->add_sub_tag(p->tag_sum);
365     x->ch[1]=y;
        return;
    }
    inline void attach_rch(lcc_node *x,lcc_node *y,int id,int id2){
        if(_attached[id]) attach_rch(x,y,id);
        else attach_rch(x,y,id2);
        return;
    }
    inline void attach_rch(lcc_node *x,lcc_node *y){
```

```
    if(!y) return;
375     attach_rch(x,y,y->first->lcc);
    return;
}
inline lcc_node * access(lcc_node *x){
    lcc_node *ret=NULL;
    int last_ex_last_id;
    while(x){
        lcc_node *t=splay(x)->ch[0];
        if(!t){
            detach_rch(x);
385             if(ret) attach_rch(x,ret,ret->first->lcc,last_ex_last_id);
            ret=x;x->update();
            x=x->fa;
            continue;
        }
        while(t->ch[1]) t->down(),t=t->ch[1];
        if(!splay(t)->cyc){
            splay(x);
            detach_rch(x);
            if(ret) attach_rch(x,ret,ret->first->lcc,last_ex_last_id);
395             ret=x;x->update();
            x=x->fa;
            continue;
        }
        cycle *c=t->cyc;
        lcc_node *A=lcc+c->A,*B=lcc+c->B,*ex=splay(c->ex);
        bool need_tag_down=false;
        lcc_node *B_ex;
        if(splay(B)->fa==A){
            detach_rch(B);
405             B->ch[1]=ex;ex->fa=B;B->update();
            need_tag_down=true;
            B_ex=B->ch[0]->first;
        }else if(splay(A)->fa==B){
            std::swap(c->A,c->B);std::swap(A,B);ex->add_rev_tag();
            detach_rch(B);
            B->ch[1]=ex;ex->fa=B;B->update();
            need_tag_down=true;
            B_ex=B->ch[0]->last;
        }else{
```

```
            bool f=0;
            if(getrank(A)>getrank(B)){
                std::swap(c->A,c->B);std::swap(A,B);ex->add_rev_tag();
                f=1;
            }
            splay(A)->ch[1]->fa=NULL;A->ch[1]=NULL;A->update();
            splay(B);detach_rch(B);
            B->ch[1]=ex;ex->fa=B;B->update();
            B_ex=f ? B->ch[0]->last : B->ch[0]->first;
        }
        // add tag to ex
        int tag_ex=splay(B_ex)->ex_tag_sum;
        B_ex->ex=NULL_INFO;
        B_ex->update();
        ex=splay(B)->ch[1];
        ex->add_chain_tag(tag_ex);
        ex->add_sub_tag(tag_ex);
        B->update();
        splay(x);c->B=x->lcc;
        if(x->ch[1]->totlen<x->ch[0]->totlen) x->add_rev_tag();
        x->add_mpath_tag(x->ch[1]->totlen==x->ch[0]->totlen);
        x->down();
        c->ex=x->ch[1];x->ch[1]->fa=NULL;
        x->ch[1]=NULL;
        x->update();
        lcc_node *tmp=splay(x->first);
        tmp->ex=c->ex->all;
        tmp->ex_tag_sum=NULL_TAG;
        tmp->update();
        splay(x);
        if(ret) attach_rch(x,ret,ret->first->lcc,last_ex_last_id);
        x->update();
        last_ex_last_id=c->ex->last->lcc;
        if(splay(A)->ch[1]) ret=x,x=x->fa;
        else{
            if(need_tag_down) attach_rch(A,x,c->ex->last->lcc,x->first->lcc)
    ;
            A->ch[1]=x;x->fa=A;A->update();
            ret=A;x=A->fa;
        }
    }
```

```
455     return ret;
    }
    inline void setroot(int x){access(lcc+x)->add_rev_tag();};
    inline bool link(int u,int v,int len){
        if(u==v) return false;
        setroot(u);
        lcc_node *t=access(lcc+v);
        while(t->ch[0]) t->down(),t=t->ch[0];
        if(splay(t)!=lcc+u){
            lcc_node *p=new_edge_node(u,v,len);
465         p->fa=splay(lcc+u);
            lcc[u].ch[0]=p;
            lcc[u].fa=lcc+v;
            lcc[u].update();
            splay(lcc+v)->ch[1]=lcc+u;
            lcc[v].update();
            return true;
        }
        if(t->hascyc) return false;
        lcc_node *ex=new_edge_node(u,v,len);
475     cycle *c=new cycle((cycle){u,v,ex});
        ex->add_cyc_tag(c);
        t->add_cyc_tag(c);
        access(lcc+v);
        return true;
    }
    inline bool cut(int u,int v,int len){
        if(u==v) return false;
        setroot(u);
        lcc_node *t=access(lcc+v);
485     while(t->ch[0]) t->down(),t=t->ch[0];
        if(splay(t)!=lcc+u) return false;
        if(!t->hascyc){
            if(t->size!=3) return false;
            if(t->totlen!=len) return false;
            t=t->ch[1];
            if(t->ch[0]) t->down(),t=t->ch[0];
            splay(t);
            t->ch[0]->fa=NULL;t->ch[1]->fa=NULL;
            return true;
495     }
```

```
    t=splay(lcc+v)->ch[0];
    while(t->ch[1]) t->down(),t=t->ch[1];
    cycle *c=splay(t)->cyc;
    if(!c) return false;
    t=splay(lcc+u)->ch[1];
    while(t->ch[0]) t->down(),t=t->ch[0];
    if(splay(t)->cyc!=c) return false;
    lcc_node *ex=c->ex;
    if(ex->size==1 && ex->len==len){
505     t->add_cyc_tag(NULL);
        t->add_mpath_tag(0);
        delete c;
        return true;
    }
    if(t->size!=3 || t->len!=len) return false;
    // lcc[u].mpath == 0 !
    ex->add_cyc_tag(NULL);
    ex->add_mpath_tag(0);
    ex->add_rev_tag();
515 ex->add_sub_tag(t->ex_tag_sum);
    ex->add_chain_tag(t->ex_tag_sum);
    lcc[u].fa=lcc[v].fa=NULL;
    while(ex->ch[0]) ex->down(),ex=ex->ch[0];
    splay(ex)->ch[0]=lcc+u;lcc[u].fa=ex;ex->update();
    while(ex->ch[1]) ex->down(),ex=ex->ch[1];
    splay(ex)->ch[1]=lcc+v;lcc[v].fa=ex;ex->update();
    delete c;
    return true;
}
525 inline Info query_path(int u,int v){
    setroot(u);
    lcc_node *t=access(lcc+v);
    while(t->ch[0]) t->down(),t=t->ch[0];
    if(splay(t)!=lcc+u) return (Info){-1,-1,-1};
    if(t->hasmpath) return (Info){-2,-2,-2};
    return t->sum;
}
inline Info query_subcactus(int u,int v){
    setroot(u);
535 lcc_node *t=access(lcc+v);
    while(t->ch[0]) t->down(),t=t->ch[0];
```

```
    if(splay(t)!=lcc+u) return (Info){-1,-1,-1};
    Info ret=splay(lcc+v)->x;
    if(splay_root[v]) ret=ret+splay_root[v]->sum;
    return ret;
}
inline bool modify_path(int u,int v,int tag){
    setroot(u);
    lcc_node *t=access(lcc+v);
    while(t->ch[0]) t->down(),t=t->ch[0];
    if(splay(t)!=lcc+u) return false;
    if(t->hasmpath) return false;
    t->add_chain_tag(tag);
    return true;
}
inline bool modify_subcactus(int u,int v,int tag){
    setroot(u);
    lcc_node *t=access(lcc+v);
    while(t->ch[0]) t->down(),t=t->ch[0];
    if(splay(t)!=lcc+u) return false;
    splay(lcc+v);
    lcc[v].x=lcc[v].x*tag;
    if(splay_root[v]) splay_root[v]->add_tag(tag);
    lcc[v].update();
    return true;
}
void init(){
    n=read();Q=read();
    int i;
    static int w[MAXN];
    for(i=1;i<=n;i++){
        w[i]=read();
        lcc[i].first=lcc[i].last=lcc+i;
        lcc[i].size=1;
        lcc[i].x=lcc[i].sum=lcc[i].all=(Info){w[i],1,w[i]};
        lcc[i].sub=lcc[i].ex=NULL_INFO;
        lcc[i].chain_tag=lcc[i].sub_tag=lcc[i].ex_tag_sum=NULL_TAG;
    }
    _node_tot=lcc+n;
    return;
}
void work(){
```

```
    for(int i=1;i<=Q;i++){
        char ch=getchar();
        while(ch<=32) ch=getchar();
        if(ch=='l'){
            Pass_Pau(3);
            int u=read(),v=read(),len=read();
            puts(link(u,v,len) ? "ok" : "failed");
        }else if(ch=='c'){
            Pass_Pau(2);
            int u=read(),v=read(),len=read();
            puts(cut(u,v,len) ? "ok" : "failed");
        }else if(ch=='q'){
            Pass_Pau(4);
            ch=getchar();
            int u=read(),v=read();
            Info ret;
            ret=ch=='1' ? query_path(u,v) : query_subcactus(u,v);
            printf("%d %lld\n",ret.mi,ret.sum);
        }else if(ch=='a'){
            Pass_Pau(2);
            ch=getchar();
            int u=read(),v=read(),val=read();
            puts((ch=='1'?modify_path(u,v,val):modify_subcactus(u,v,val)) ?
    "ok" : "failed");
        }else puts("error");
    }
    return;
}
void print(){
    return;
}
int main(){init();work();print();return 0;}
```

2.3 Dynamic Sequence (Segment Tree)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Dynamic Sequence (Segment Tree).hpp (0 bytes, 0 lines)

2.4 Dynamic Sequence (Treap)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Dynamic Sequence (Treap).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;
    }
};
```

2.5 Dynamic Tree (Link-Cut Tree)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Dynamic 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):
```

```
13          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;
23          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])
33                      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);
43      }
        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);
53          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));
103     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);
113     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;
123 }
    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));
133     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));
183 }
    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));
193     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);
203 }
    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);
213     up(c);
    }
};
```

2.6 Dynamic Tree (Self-Adjusting Top Tree)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Dynamic 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]);
        }
    }
};
```

2.7 Fenwick Tree 1D

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Fenwick Tree 1D.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;
};
```

2.8 Fenwick Tree 2D

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Fenwick Tree 2D.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;
};
```

2.9 Fenwick Tree 3D

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Fenwick Tree 3D.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;
};
```

2.10 K-D Tree 2D

warning: old style will be replaced ... see Suffix Array (DC3) for new style

K-D Tree 2D.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;
```

2.11 K-D Tree 3D

warning: old style will be replaced ... see Suffix Array (DC3) for new style

K-D Tree 3D.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;
```

2.12 Mergeable Set

Description

Maintain sets of elements whose values are in a given range. Two sets can be merged efficently. Range query is also supported.

Methods

template<class T,class U>MergeableSet(U l,U r);	
Description	construct an object of MergeableSet, it is not a set, it maintains sets
Parameters	**Description**
T	type of range information, should support $+$, $+$ is applied when two range do not intersect or they represent the same leaf
U	type of values of elements
l	minimum value of elements
r	maximum value of elements
Time complexity	$\Theta(1)$
Space complexity	$\Theta(1)$
Return value	an object of MergeableSet

node*insert(node*x,T f,U v);	
Description	insert a element into a set
Parameters	**Description**
x	root of the set, use 0 to represent empty set
f	information of the element
v	value of the element
Time complexity	$\Theta(\log\|r-l\|)$
Space complexity	$\Theta(\log\|r-l\|)$
Return value	root of the new set

node*erase(node*x,U v);	
Description	erase the element with certain value
Parameters	**Description**
x	root of the set
v	value of the element
Time complexity	$\Theta(1)$ (amortized)
Space complexity	$\Theta(1)$ (amortized)
Return value	root of the new set

node*merge(node*x,node*y);	
Description	merge two sets
Parameters	**Description**
x	root of one set, use 0 to represent empty set
y	root of another set, use 0 to represent empty set
Time complexity	$\Theta(1)$ (amortized)
Space complexity	$\Theta(1)$ (amortized)
Return value	root of the new set

vector<T>query(node*x,U ql,U qr);	
Description	do range query
Parameters	**Description**
x	root of the set, use 0 to represent empty set
ql	start of the range, itself is included
qr	end of the range, itself is included
Time complexity	$O(\log\|r-l\|)$
Space complexity	$O(\log\|r-l\|)$
Return value	vector of information, that it is empty means no information in that range other wise the result is its first element

void destroy(node*x);	
Description	delete whole set
Parameters	**Description**
x	root of the set, use 0 to represent empty set
Time complexity	$\Theta(1)$ (amortized)
Space complexity	$\Theta(1)$ (amortized)
Return value	none

References

Title	**Author**
线段树的合并——不为人知的实用技巧	黄嘉泰

Code

Mergeable Set.hpp (2254 bytes, 91 lines)

```
#include<vector>
using namespace std;
template<class T,class U>struct MergeableSet{
    struct node{
        node(T _f):f(_f){
            c[0]=c[1]=0;
        }
        T f;
        node*c[2];
    };
    MergeableSet(U l,U r):vl(l),vr(r){
    }
```

```
void update(node*x){
    if(x->c[0]&&x->c[1])
        x->f=x->c[0]->f+x->c[1]->f;
    else
        x->f=x->c[0]?x->c[0]->f:x->c[1]->f;
}
node*insert(node*x,T f,U v,U l=0,U r=0){
    if(!l&&!r)
        l=vl,r=vr;
    if(l==r){
        if(x)
            x->f=x->f+f;
        else
            x=new node(f);
    }else{
        U m=l+(r-l)/2;
        int d=v>m;
        node*y=insert(x?x->c[d]:0,f,v,d?m+1:l,d?r:m);
        if(!x)
            x=new node(y->f);
        x->c[d]=y,update(x);
    }
    return x;
}
node*erase(node*x,U v,U l=0,U r=0){
    if(!l&&!r)
        l=vl,r=vr;
    if(l==r){
        delete x;
        return 0;
    }
    U m=l+(r-l)/2;
    int d=v>m;
    x->c[d]=erase(x?x->c[d]:0,v,d?m+1:l,d?r:m);
    if(!x->c[0]&&!x->c[1]){
        delete x;
        return 0;
    }
    update(x);
    return x;
}
```

```
    node*merge(node*x,node*y,U l=0,U r=0){
        if(!l&&!r)
            l=vl,r=vr;
        if(!x||!y)
            return x?x:y;
        if(l==r)
            x->f=x->f+y->f;
        else{
            U m=l+(r-l)/2;
            x->c[0]=merge(x->c[0],y->c[0],l,m);
            x->c[1]=merge(x->c[1],y->c[1],m+1,r);
            update(x);
        }
        return x;
    }
    vector<T>query(node*x,U ql,U qr,U l=0,U r=0){
        if(!l&&!r)
            l=vl,r=vr;
        if(!x||ql>r||qr<l)
            return vector<T>();
        if(ql<=l&&qr>=r)
            return vector<T>(1,x->f);
        U m=l+(r-l)/2;
        vector<T>u=query(x->c[0],ql,qr,l,m),
            v=query(x->c[1],ql,qr,m+1,r);
        if(v.size()&&u.size())
            u[0]=u[0]+v[0];
        return u.size()?u:v;
    }
    void destroy(node*x){
        if(x)
            destroy(x->c[0]),
            destroy(x->c[1]),
            delete x;
    }
};
    U vl,vr;
};
```

2.13 Persistent Priority Queue

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Persistent Priority Queue.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;
    }
};
```

2.14 Persistent Set

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Persistent Set.hpp (0 bytes, 0 lines)

2.15 Priority Queue (Binary Heap)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Priority Queue (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);
    }
};
```

2.16 Priority Queue (Pairing Heap)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Priority Queue (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);
    }
87  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
97          x->val=v,cut(root,x),merge(root,x);
    }
    int size(){
        return siz;
    }
};
```

2.17 Priority Queue (Skew Heap)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Priority Queue (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(){
 8      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;
    }
};
```

2.18 Range Minimum Query

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Range Minimum Query.hpp (7403 bytes, 228 lines)

```
template<typename VALUE_TYPE,int MEMORY_SIZE>struct Memory{
        VALUE_TYPE memory_buffer[MEMORY_SIZE],*memory_pointer;
        Array<VALUE_TYPE*,MEMORY_SIZE>stack;
        inline void Construct(){
            memory_pointer=memory_buffer;
            stack.Construct();
        }
        inline void Destruct(){
            stack.Destruct();
        }
        inline VALUE_TYPE*New(){
            VALUE_TYPE*t;
            if(!stack.Empty()){
                t=stack.Back();
                stack.PopBack();
            }else
                t=memory_pointer++;
            return t;
        }
        inline void Delete(VALUE_TYPE*a){
            stack.PushBack(a);
        }
    };
    template<typename VALUE_TYPE,int VERTEX_SIZE,int EDGE_SIZE>struct
    AdjacencyList{
        struct node{
```

```
        node*next;
        VALUE_TYPE value;
    };
    Memory<node,EDGE_SIZE>memory;
    struct Iterator{
        node*pointer;
        inline Iterator(node*_pointer=0):
        pointer(_pointer){
        }
        inline Iterator&operator++(){
            pointer=pointer->next;
            return*this;
        }
        inline Iterator operator++(int){
            Iterator t=*this;
            pointer=pointer->next;
            return t;
        }
        inline VALUE_TYPE&operator*(){
            return pointer->value;
        }
        inline VALUE_TYPE*operator->(){
            return&pointer->value;
        }
        inline bool operator==(const Iterator&a){
            return pointer==a.pointer;
        }
        inline bool operator!=(const Iterator&a){
            return pointer!=a.pointer;
        }
    };
    node*begin[VERTEX_SIZE],*edge_pointer;
    inline void Construct(const int&vertex_count){
        std::fill(begin,begin+vertex_count,(node*)0);
        memory.Construct();
    }
    inline void Destruct(){
        memory.Destruct();
    }
    inline void AddEdge(const int&a,const VALUE_TYPE&b){
        node*t=memory.New();
```

```
            t->next=begin[a];
            t->value=b;
            begin[a]=t;
        }
        inline Iterator Begin(const int&a){
            return Iterator(begin[a]);
        }
        inline Iterator End(const int&a){
            return Iterator(0);
        }
    };
    template<int VERTEX_SIZE,int BLOCK_SIZE,int POW_BLOCK_SIZE,int
    BLOCK_COUNT,int LG_BLOCK_COUNT>struct LowestCommonAncestor{
        int vertex_size,root,block_size,block_count,dfs_begin[VERTEX_SIZE],
    lg[BLOCK_COUNT],block[POW_BLOCK_SIZE][BLOCK_SIZE][BLOCK_SIZE],
    block_index[BLOCK_COUNT];
        AdjacencyList<int,VERTEX_SIZE,VERTEX_SIZE*2>adjacency_list;
        std::pair<int,int>dfs_sequence[VERTEX_SIZE*2],sparse_table[
    LG_BLOCK_COUNT][BLOCK_COUNT];
        inline void Construct(const int&_vertex_size,const int&_root){
            vertex_size=_vertex_size;
            root=_root;
            adjacency_list.Construct(vertex_size);
        }
        inline void Destruct(){
            adjacency_list.Destruct();
        }
        inline void AddEdge(const int&a,const int&b){
            adjacency_list.AddEdge(a,b);
            adjacency_list.AddEdge(b,a);
        }
        inline void Build(){
            block_size=std::max(2,int(std::log(double(2*vertex_size-1))/(
    log(2.0)*2)));
            block_count=(2*vertex_size-1)%block_size?(2*vertex_size-1)/
    block_size+1:(2*vertex_size-1)/block_size;
            build_dfs_sequence();
            build_sparse_table();
            build_block();
        }
        inline void build_dfs_sequence(){
```

```
        static int prt[VERTEX_SIZE],dpt[VERTEX_SIZE];
        prt[root]=-1;
        dpt[root]=0;
        std::pair<int,int>*dfs=dfs_sequence;
        static Array<std::pair<int,typename AdjacencyList<int,
    VERTEX_SIZE,VERTEX_SIZE*2>::Iterator>,VERTEX_SIZE>stk;
        stk.Construct();
        stk.PushBack(std::make_pair(root,adjacency_list.Begin(root)));
109     while(!stk.Empty()){
            int u=stk.Back().first;
            typename AdjacencyList<int,VERTEX_SIZE,VERTEX_SIZE*2>::
    Iterator i=stk.Back().second;
            stk.PopBack();
            if(i==adjacency_list.Begin(u))
                dfs_begin[u]=dfs-dfs_sequence;
            *dfs++=std::make_pair(dpt[u],u);
            if(i!=adjacency_list.End(u)&&*i==prt[u])
                ++i;
            if(i!=adjacency_list.End(u)){
119             int v=*i;
                stk.PushBack(std::make_pair(u,++i));
                prt[v]=u;
                dpt[v]=dpt[u]+1;
                stk.PushBack(std::make_pair(v,adjacency_list.Begin(v)));
            }
        }
        stk.Destruct();
    }
    inline void build_sparse_table(){
129     for(int i=0;(1<<i)<=block_count;++i)
            for(int j=0;j+(1<<i)-1<block_count;++j)
                if(i==0)
                    sparse_table[i][j]=*std::min_element(dfs_sequence+j*
    block_size,dfs_sequence+std::min((j+1)*block_size,2*vertex_size-1));
                else
                    sparse_table[i][j]=std::min(sparse_table[i-1][j],
    sparse_table[i-1][j+(1<<(i-1))]);
        lg[1]=0;
        for(int i=2;i<=block_count;++i)
            lg[i]=lg[i-1]+((1<<(lg[i-1]+1))<=i?1:0);
    }
```

```
139         inline std::pair<int,int>query_sparse_table(const int&a,const int&b
    ){
            int t=lg[b - a + 1];
            return std::min(sparse_table[t][a],sparse_table[t][b-(1<<t)+1])
    ;
        }
        inline void build_block(){
            for(int i=0;i<(1<<(block_size-1));++i){
                static std::pair<int,int>t[BLOCK_SIZE];
                for(int j=1;j<block_size;++j)
                    if((i>>(block_size-j-1))&1)
                        t[j]=std::make_pair(t[j-1].first+1,j);
149                 else
                        t[j]=std::make_pair(t[j-1].first-1,j);
                for(int j=0;j<block_size;++j){
                    std::pair<int,int>tmp=t[j];
                    for(int k=j;k<block_size;++k){
                        CoreLibrary::MakeMin(tmp,t[k]);
                        block[i][j][k]=tmp.second;
                    }
                }
            }
159         for(int i=0;i<block_count;++i){
                int t=0;
                for(int j=i*block_size+1;j<(i+1)*block_size;++j)
                    if(j>=2*vertex_size-1||dfs_sequence[j].first-
    dfs_sequence[j-1].first==1)
                        t=(t<<1)+1;
                    else
                        t<<=1;
                block_index[i]=t;
            }
        }
169     inline std::pair<int,int>query_block(const int&a,const int&b){
            int t=a/block_size;
            return dfs_sequence[block[block_index[t]][a-t*block_size][b-t*
    block_size]+t*block_size];
        }
        inline int Query(int a,int b){
            a=dfs_begin[a];
            b=dfs_begin[b];
```

```
            if(a>b)
                std::swap(a,b);
            int ia=a/block_size,ib=b/block_size;
179         if(ia==ib)
                return query_block(a,b).second;
            if(ia+1==ib)
                return std::min(query_block(a,(ia+1)*block_size-1),
    query_block(ib*block_size,b)).second;
            return std::min(std::min(query_block(a,(ia+1)*block_size-1),
    query_block(ib*block_size,b)),query_sparse_table(ia+1,ib-1)).second;
        }
    };
    template<typename VALUE_TYPE,typename COMPARER_TYPE,int SEQUENCE_SIZE,
    int BLOCK_SIZE,int POW_BLOCK_SIZE,int BLOCK_COUNT,int LG_BLOCK_COUNT>
    struct RangeMinimumQuery{
        LowestCommonAncestor<SEQUENCE_SIZE,BLOCK_SIZE,POW_BLOCK_SIZE,
    BLOCK_COUNT,LG_BLOCK_COUNT>lowest_common_ancestor;
        VALUE_TYPE*sequence_begin,*sequence_end;
189     inline void Construct(VALUE_TYPE*_sequence_begin,VALUE_TYPE*
    _sequence_end){
            sequence_begin=_sequence_begin;
            sequence_end=_sequence_end;
            static int prt[SEQUENCE_SIZE];
            Array<int,SEQUENCE_SIZE>stk;
            stk.Construct();
            for(int i=0;i<sequence_end-sequence_begin;++i){
                typename Array<int,SEQUENCE_SIZE>::Iterator j=stk.End()-1;
                while(j>=stk.Begin()&&COMPARER_TYPE()(*(sequence_begin+i),*(
    sequence_begin+*j)))
                    --j;
199             if(j<stk.Begin())
                    prt[i]=-1;
                else
                    prt[i]=*j;
                if(j+1!=stk.End())
                    prt[*(j+1)]=i;
                while(j+1!=stk.End())
                    stk.PopBack();
                stk.PushBack(i);
            }
209         int root;
```

```
            for(int i=0;i<sequence_end-sequence_begin;++i)
                if(prt[i]==-1){
                    root=i;
                    break;
                }
                lowest_common_ancestor.Construct(sequence_end-
    sequence_begin,root);
                for(int i=0;i<sequence_end-sequence_begin;++i)
                    if(prt[i]!=-1)
                        lowest_common_ancestor.AddEdge(i,prt[i]);
219             lowest_common_ancestor.Build();
                stk.Destruct();
        }
        inline void Destruct(){
            lowest_common_ancestor.Destruct();
        }
        inline int Query(const int&a,const int&b){
            return *(sequence_begin+lowest_common_ancestor.Query(a,b));
        }
    };
```

2.19 Set (Red-Black Tree)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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

```
    #include<bits/stdc++.h>
 2  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;
12          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){
22      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;
32      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;
42          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;
    }
52  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;
182     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
192             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)
202         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){
212         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;
222     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;
232     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(;;)
242         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)
252         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(){
302     return*begin();
    }
    T back(){
        return*reverse_begin();
    }
};
```

2.20 Set (Treap)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Set (Treap).hpp (2216 bytes, 91 lines)

```
#include<bits/stdc++.h>
using namespace std;
 3 template<class T,class C>struct Set{
    struct node{
        node(T u){
            c[0]=c[1]=0,v=u,s=1;
            f=rand()*1.0/RAND_MAX*2e9;
        }
        T v;
        node*c[2];
        int s,f;
    }*j,*k;
13  int size(node*x){
        return x?x->s:0;
    }
    void update(node*x){
        x->s=1;
        for(int i=0;i<2;++i)
            x->s+=size(x->c[i]);
    }
    node*merge(node*x,node*y){
        if(!x||!y)
23          return x?x:y;
        if(x->f<y->f)
            x->c[1]=merge(x->c[1],y),y=x;
        else
```

```
            y->c[0]=merge(x,y->c[0]);
        update(y);
        return y;
    }
    void split(node*x,int t){
        if(x){
            int s=size(x->c[0]);
            if(s>=t)
                split(x->c[0],t),
                x->c[0]=k,k=x;
            else
                split(x->c[1],t-s-1),
                x->c[1]=j,j=x;
            update(x);
        }else
            j=k=0;
    }
    void clear(node*x){
        if(x){
            clear(x->c[0]);
            clear(x->c[1]);
            delete x;
        }
    }
    node*find(node*z,T a){
        node*r=0;
        while(z)
            if(C()(a,z->v))
                z=z->c[0];
            else if(C()(z->v,a))
                z=z->c[1];
            else
                break;
        return r;
    }
    node*select(node*z,int a){
        for(;;)
            if(size(z->c[0])>=a)
                z=z->c[0];
            else if(size(z->c[0])+1<a)
                a-=size(z->c[0])+1,z=z->c[1];
```

```
            else
                return z;
        }
        pair<node*,int>count(node*z,T a,int d){
            int c=0;
73          node*r=0;
            while(z)
                if(C()(d?a:z->v,d?z->v:a))
                    r=z,c+=size(z->c[d])+1,
                    z=z->c[!d];
                else
                    z=z->c[d];
            return make_pair(r,c);
        }
        node*erase(node*x,T v){
83          split(x,count(x,v,0).second);
            node*y=j;split(k,1);delete j;
            return merge(y,k);
        }
        node*insert(node*x,T v){
            split(x,count(x,v,0).second);
            return merge(merge(j,new node(v)),k);
        }
    };
```

2.21 Union Find Set

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Union Find Set.hpp (278 bytes, 15 lines)

```
const int N=100000;
struct UFS{
    int p[N+1],n;
    UFS(int _n):
        n(_n){
        for(int i=1;i<=n;++i)
            p[i]=i;
    }
    int find(int x){
```

```
        return p[x]==x?x:p[x]=find(p[x]);
    }
    int link(int x,int y){
        p[find(x)]=y;
    }
};
```

2.22 Virtual Tree

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Virtual Tree.hpp (2352 bytes, 55 lines)

```
#ifndef VIRTUAL_TREE
#define VIRTUAL_TREE
#include<bits/stdc++.h>
namespace CTL{
    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();}};}
#endif
```

CHAPTER 3

Dynamic Programming

3.1 Knapsack Problem

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Knapsack Problem.hpp (1100 bytes, 27 lines)

```
#ifndef KNAPSACK_PROBLEM
#define KNAPSACK_PROBLEM
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    template<class T>struct KnapsackProblem{
        int n,v;vector<int>vol,vsum;vector<T>val;
        vector<vector<T> >dp;vector<vector<int> >to;
        KnapsackProblem(int _n,int _v):
            n(_n),v(_v),vol(n+1),vsum(n+1),val(n+1),to(n+1){}
        void set(int a,int p,int v,T w){
            to[p].push_back(a);vol[a]=v;val[a]=w;}
        void work(int x){
            for(int i=0;i<to[x].size();++i){
                int y=to[x][i];work(y);vector<T>tdp=dp[x];
                for(int j=0;j<=vsum[x]&&j<=v;++j){
                    for(int k=0;k<=vsum[y]&&j+k<=v;++k)
                        dp[x][j+k]=max(dp[x][j+k],tdp[j]+dp[y][k]);}
                vsum[x]+=vsum[y];}
            vsum[x]+=vol[x];
            for(int i=v;i>=0;--i)
                if(i<vol[x])dp[x][i]=0;
                else dp[x][i]=dp[x][i-vol[x]]+val[x];}
        T run(){
            dp=vector<vector<T> >(n+1,vector<T>(v+1));
            work(0);return dp[0][v];}};}
#endif
```

CHAPTER 4

Miscellaneous Topics

4.1 Checker (Linux)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

4.2 Checker (Windows)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Checker (Windows).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
```

4.3 Date

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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);
}
```

4.4 Expression Evaluation

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Expression Evaluation.hpp (3275 bytes, 69 lines)

```
#ifndef EXPRESSION_EVALUATION
#define EXPRESSION_EVALUATION
#include<bits/stdc++.h>
namespace CTL{
    namespace ExpressionEvaluation{
        typedef long long T;
        T run(string exp){
            map<string,int>oid;map<int,int>pro,pri,dir;
            oid["("]=0;pro[0]=6;pri[0]=0;dir[0]=1;
            oid[")"]=1;pro[1]=1;pri[1]=0;dir[1]=0;
            oid["+'"]=2;pro[2]=5;pri[2]=5;dir[2]=1;
            oid["-'"]=3;pro[3]=5;pri[3]=5;dir[3]=1;
            oid["+"]=4;pro[4]=2;pri[4]=2;dir[4]=0;
            oid["-"]=5;pro[5]=2;pri[5]=2;dir[5]=0;
            oid["*"]=6;pro[6]=3;pri[6]=3;dir[6]=0;
            oid["/"]=7;pro[7]=3;pri[7]=3;dir[7]=0;
            oid["^"]=8;pro[8]=4;pri[8]=4;dir[8]=1;
            exp="("+exp+")";stack<T>vas;
            stack<int>ops;int lstnum=0;
            for(int i=0;i<exp.size();){
                while(i<exp.size()&&isspace(exp[i]))++i;
                if(i==exp.size())break;
                if(isdigit(exp[i])){
                    int j=i;
```

```
25                  while(j+1<exp.size()&&
                        (isdigit(exp[j+1])||exp[j+1]=='.'))++j;
                    stringstream ss;T v;ss<<exp.substr(i,j-i+1);ss>>v;
                    vas.push(v);lstnum=1;i=j+1;
                }else{
                    string o(1,exp[i++]);
                    if((o[0]=='+'||o[0]=='-')&&!lstnum)
                        o+="'";int id=oid[o];
                    for(;ops.size()&&(pri[ops.top()]>pro[id]||
                        (pri[ops.top()]==pro[id]&&!dir[id]));){
35                      int dop=ops.top();ops.pop();
                        if(dop==3){
                            T x=vas.top();vas.pop();
                            vas.push(-x);lstnum=1;
                        }else if(dop==4){
                            T y=vas.top();vas.pop();
                            T x=vas.top();vas.pop();
                            vas.push(x+y);lstnum=1;
                        }else if(dop==5){
                            T y=vas.top();vas.pop();
45                          T x=vas.top();vas.pop();
                            vas.push(x-y);lstnum=1;
                        }else if(dop==6){
                            T y=vas.top();vas.pop();
                            T x=vas.top();vas.pop();
                            vas.push(x*y);lstnum=1;
                        }else if(dop==7){
                            T y=vas.top();vas.pop();
                            T x=vas.top();vas.pop();
                            if(!y)
55                              return numeric_limits<T>::max();
                            vas.push(x/y);lstnum=1;
                        }else if(dop==8){
                            T y=vas.top();vas.pop();
                            T x=vas.top(),r=1;vas.pop();
                            if(x==0||x==1)y=1;if(x==-1)y%=2;
                            for(T t=1;t<=y;++t)
                                r*=x;vas.push(r);lstnum=1;}}
                    if(id!=1)ops.push(id),lstnum=0;
                    else if(ops.empty())
65                      return numeric_limits<T>::max();
```

```
                else ops.pop();}}
            return ops.size()?numeric_limits<T>::max():
                vas.top();}}}
#endif
```

4.5 Fast Reader

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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")){
32      ipt();
    }
    ~FastReader(){
        fclose(f);
    }
    template<class T>FastReader&operator>>(T&a){
        int n=1;
        ign();
        if(*p=='-')
            n=-1,++p;
42      for(a=0;isd();)
            a=a*10+*p++-'0';
        a*=n;
        return*this;
    }
    FastReader&operator>>(char&a){
        ign();
        a=*p++;
        return*this;
    }
52  FastReader&operator>>(char*a){
        for(ign();isc();)
            *a++=*p++;
        *a=0;
        return*this;
    }
    char get(){
        return*p++;
    }
};
```

4.6 Fast Writer

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
};
```

4.7 Large Stack

Description

Make system stack larger. Simply put this code before main function, and the system stack will be enlarged.

Fields

#define STACK_SIZE 64	
Description	the size of system stack in MB

Code

Large Stack.hpp (845 bytes, 32 lines)

```
#include<cstdlib>
using namespace std;
#define STACK_SIZE 64
#if __GNUC__
    #if __x86_64__||__ppc64__
        extern int _main(void)__asm__("_main");
    #else
        extern int _main(void)__asm__("__main");
    #endif
    int __main();
    int _main() {
        __main();
        exit(0);
    }
    int main(){
        __asm__ __volatile__(
            #if __x86_64__||__ppc64__
                "movq %0,%%rsp\n"
                "pushq $exit\n"
                "jmp _main\n"
            #else
                "movl %0,%%esp\n"
                "pushl $_exit\n"
                "jmp __main\n"
```

```
            #endif
                ::"r"((char*)malloc(STACK_SIZE<<20)+(STACK_SIZE<<20))
        );
    }
    #define main __main
#elif defined(_MSC_VER)
    #pragma comment(linker,"/STACK:1024000000,1024000000")
#endif
```

4.8 Main (CPP)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Main (CPP).hpp (338 bytes, 18 lines)

```
#include<bits/stdc++.h>
#define lp(i,l,r)for(auto i=l;i<=r;++i)
#define rp(i,r,l)for(auto i=r;i>=l;--i)
using namespace std;
typedef long long ll;
typedef long double ld;
void _main();
int main(){
    ios::sync_with_stdio(0);
    _main();
    #ifndef ONLINE_JUDGE
    for(;;);
    #endif
    return 0;
}
void _main(){

}
```

4.9 Number Speller

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
}
```

CHAPTER 5

Graph Algorithms

5.1 Bipartite Graph Maximum Matching

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Bipartite Graph Maximum Matching.hpp (3121 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){
38          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);
48              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>();
58          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))
68                          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;
    }
};
```

5.2 Bipartite Graph Maximum Weight Matching

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Bipartite Graph Maximum Weight Matching.hpp (4522 bytes, 259 lines)

```
int n,nx,ny,m;
int link[MaxN],lx[MaxN],ly[MaxN],slack[MaxN];
int visx[MaxN],visy[MaxN],w[MaxN][MaxN];

bool DFS(int x) {
    visx[x] = 1;
    for (int y = 1;y <= ny;y++) {
        if (visy[y]) continue;
        int t = lx[x] + ly[y] - w[x][y];
        if (t == 0) {
            visy[y] = 1;
            if (link[y] == -1 || DFS(link[y])) {
                link[y] = x;
                return true;
            }
        }
        else if (slack[y] > t) slack[y] = t;
    }
    return false;
}
void KM() {
    int i,j;
    memset (link,-1,sizeof(link));
    memset (ly,0,sizeof(ly));
    for (i = 1;i <= nx;i ++) for (j = 1,lx[i] = -INF;j <= ny;j ++)
        if (w[i][j] > lx[i]) lx[i] = w[i][j];

    for (int x = 1;x <= nx;x ++) {
        for (i = 1;i <= ny;i++) slack[i] = INF;
        while (1) {
            memset (visx,0,sizeof(visx));
            memset (visy,0,sizeof(visy));
            if (DFS(x)) break;
            int d = INF;
            for (i = 1;i <= ny;i ++) if (!visy[i]&&d > slack[i]) d = slack[i
```

```
    ];
            for (i = 1;i <= nx;i ++) if (visx[i]) lx[i] -= d;
            for (i = 1;i <= ny;i ++)
                if (visy[i]) ly[i] += d;
                else slack[i] -= d;
        }
    }
}


#include <cstdio>
#include <algorithm>

using namespace std;

typedef long long s64;

const int BufferSize = 1 << 16;

char buffer[BufferSize];
char *head, *tail;

inline char nextChar()
{
    if (head == tail)
    {
        int l = fread(buffer, 1, BufferSize, stdin);
        tail = (head = buffer) + l;
    }
    return *head++;
}

inline int getint()
{
    char c;
    while ((c = nextChar()) < '0' || c > '9');

    int res = c - '0';
    while ((c = nextChar()) >= '0' && c <= '9')
        res = res * 10 + c - '0';
    return res;
```

```
}

namespace Writer
{
    const int BufferSize = 2000;

    char buffer[BufferSize];
    char *tail = buffer;

    inline void putint(int x)
    {
        if (x == 0)
            *tail++ = '0';
        else
        {
            char s[10], *t = s;
            while (x != 0)
                *t++ = x % 10 + '0', x /= 10;
            while (t-- != s)
                *tail++ = *t;
        }
        *tail++ = '\n';
    }

    inline void final()
    {
        fwrite(buffer, 1, tail - buffer, stdout);
    }
};

inline void relax(int &a, const int &b)
{
    if (b > a)
        a = b;
}
inline void tense(int &a, const int &b)
{
    if (b < a)
        a = b;
}
```

```
const int MaxNL = 405;
const int MaxNR = 405;
const int INF = 0x3f3f3f3f;

int m, nL, nR, nVer;
int mat[MaxNL][MaxNR];

s64 tot_weight;
int mateL[MaxNL];
int mateR[MaxNR];
int labL[MaxNL];
int labR[MaxNR];
int faR[MaxNR];

int slackR[MaxNR];
int slackRV[MaxNR];

bool bookL[MaxNL];
bool bookR[MaxNR];

int q_n, q[MaxNL];

inline void augment(int v)
{
    while (v)
    {
        int nv = mateL[faR[v]];
        mateL[faR[v]] = v;
        mateR[v] = faR[v];
        v = nv;
    }
}

inline bool on_found_edge(const int &u, const int &v)
{
    bookR[v] = true;
    faR[v] = u;

    int nv = mateR[v];
    if (!nv)
    {
```

```
            augment(v);
159         return true;
        }
        else
        {
            bookL[nv] = true;
            q[++q_n] = nv;
        }
        return false;
    }

169 inline void match(const int &sv)
    {
        for (int u = 1; u <= nVer; ++u)
            bookL[u] = false;
        for (int v = 1; v <= nVer; ++v)
        {
            bookR[v] = false;
            slackRV[v] = faR[v] = 0;
            slackR[v] = INF;
        }
179
        bookL[q[q_n = 1] = sv] = true;
        while (true)
        {
            for (int i = 1; i <= q_n; ++i)
            {
                int u = q[i];
                for (int v = 1; v <= nVer; ++v)
                    if (!bookR[v])
                    {
189                     int d = labL[u] + labR[v] - mat[u][v];
                        if (!d)
                        {
                            if (on_found_edge(u, v))
                                return;
                        }
                        else if (d < slackR[v])
                        {
                            slackR[v] = d;
                            slackRV[v] = u;
```

```
                    }
                }
            }

            int delta = INF;
            for (int v = 1; v <= nVer; ++v)
                if (!bookR[v])
                    tense(delta, slackR[v]);
            for (int u = 1; u <= nVer; ++u)
                if (bookL[u])
                    labL[u] -= delta;
            for (int v = 1; v <= nVer; ++v)
            {
                if (bookR[v])
                    labR[v] += delta;
                else if (slackR[v] != INF)
                    slackR[v] -= delta;
            }

            q_n = 0;
            for (int v = 1; v <= nVer; ++v)
                if (!bookR[v] && !slackR[v])
                {
                    if (on_found_edge(slackRV[v], v))
                        return;
                }
        }
    }

    inline void calc_max_weight_match()
    {
        for (int u = 1; u <= nL; ++u)
            match(u);

        tot_weight = 0ll;
        for (int u = 1; u <= nL; ++u)
            tot_weight += labL[u];
        for (int v = 1; v <= nR; ++v)
            tot_weight += labR[v];
    }
```

```
int main()
{
    nL = getint(), nR = getint(), m = getint();
    nVer = max(nL, nR);

    while (m--)
    {
        int u = getint(), v = getint();
        relax(labL[u], mat[u][v] = getint());
    }

    calc_max_weight_match();

    printf("%lld\n", tot_weight);
    for (int u = 1; u <= nL; ++u)
        Writer::putint(mat[u][mateL[u]] ? mateL[u] : 0);

    Writer::final();
    return 0;
}
```

5.3 Chordality Test

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
22          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])
32                  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;
    }
42  };
```

5.4 Dominator Tree

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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){
8   }
    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]];
        }
18      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();){
28          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));
38          }
        }
        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]){
48              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]])
58                      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);
68      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
78                  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;
    }
};
```

5.5 General Graph Maximum Matching

warning: old style will be replaced ... see Suffix Array (DC3) for new style

General Graph 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);
106         int r=0;
            for(int i=1;i<=n;++i)
                if(res[i])
                    ++r;
            return r/2;
        }
};
```

5.6 General Graph Maximum Weight Matching

warning: old style will be replaced ... see Suffix Array (DC3) for new style

General Graph Maximum Weight Matching.hpp (8898 bytes, 438 lines)

```
// From http://uoj.ac/submission/16359 By vfleaking
// Adapted by lch1475369, 2015.10.26
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <vector>
using namespace std;
 8
inline int getint()
{
    char c;
    while (c = getchar(), '0' > c || c > '9');

    int res = c - '0';
    while (c = getchar(), '0' <= c && c <= '9')
        res = res * 10 + c - '0';
    return res;
18 }

class Match
{
static int INF;
typedef long long s64;
struct edge
{
```

```
    int v, u, w;
    edge(){}
28  edge(const int &_v, const int &_u, const int &_w)
        : v(_v), u(_u), w(_w){}
};

const int MaxN;
const int MaxM;
const int MaxNX;

int n, m;
edge **mat;
38
int n_matches;
s64 tot_weight;
int *mate;
int *lab;

int q_n, *q;
int *fa, *col;
int *slackv;

48 int n_x;
int *bel, **blofrom;
vector<int> *bloch;

bool *book;

public:
Match(int _N) : MaxN(_N), MaxM(_N*(_N-1)/2), MaxNX(_N+_N)
{
    mat = new edge*[MaxNX + 1];
58  for (int i = 0; i <= MaxNX; ++i)
        mat[i] = new edge[MaxNX + 1];
    blofrom = new int*[MaxNX + 1];
    for (int i = 0; i <= MaxNX; ++i)
        blofrom[i] = new int[MaxN + 1];
    bloch = new vector<int>[MaxNX + 1];
    mate = new int[MaxNX + 1];
    lab = new int[MaxNX + 1];
    q = new int[MaxN];
```

```
        fa = new int[MaxNX + 1];
68      col = new int[MaxNX + 1];
        slackv = new int[MaxNX + 1];
        book = new bool[MaxNX + 1];
        bel = new int[MaxNX + 1];
    }

    private:
    template <class T>
    inline void tension(T &a, const T &b)
    {
78      if (b < a)
            a = b;
    }
    template <class T>
    inline void relax(T &a, const T &b)
    {
        if (b > a)
            a = b;
    }
    template <class T>
88  inline int size(const T &a)
    {
        return (int)a.size();
    }

    inline int e_delta(const edge &e) // does not work inside blossoms
    {
        return lab[e.v] + lab[e.u] - mat[e.v][e.u].w * 2;
    }
    inline void update_slackv(int v, int x)
98  {
        if (!slackv[x] || e_delta(mat[v][x]) < e_delta(mat[slackv[x]][x]))
            slackv[x] = v;
    }
    inline void calc_slackv(int x)
    {
        slackv[x] = 0;
        for (int v = 1; v <= n; v++)
            if (mat[v][x].w > 0 && bel[v] != x && col[bel[v]] == 0)
                update_slackv(v, x);
```

```
108 }

    inline void q_push(int x)
    {
        if (x <= n)
            q[q_n++] = x;
        else
        {
            for (int i = 0; i < size(bloch[x]); i++)
                q_push(bloch[x][i]);
118     }
    }
    inline void set_mate(int xv, int xu)
    {
        mate[xv] = mat[xv][xu].u;
        if (xv > n)
        {
            edge e = mat[xv][xu];
            int xr = blofrom[xv][e.v];
            int pr = find(bloch[xv].begin(), bloch[xv].end(), xr) - bloch[xv].
        begin();
128         if (pr % 2 == 1)
            {
                reverse(bloch[xv].begin() + 1, bloch[xv].end());
                pr = size(bloch[xv]) - pr;
            }

            for (int i = 0; i < pr; i++)
                set_mate(bloch[xv][i], bloch[xv][i ^ 1]);
            set_mate(xr, xu);

138         rotate(bloch[xv].begin(), bloch[xv].begin() + pr, bloch[xv].end());
        }
    }
    inline void set_bel(int x, int b)
    {
        bel[x] = b;
        if (x > n)
        {
            for (int i = 0; i < size(bloch[x]); i++)
                set_bel(bloch[x][i], b);
```

```
148     }
    }

    inline void augment(int xv, int xu)
    {
        while (true)
        {
            int xnu = bel[mate[xv]];
            set_mate(xv, xu);
            if (!xnu)
158             return;
            set_mate(xnu, bel[fa[xnu]]);
            xv = bel[fa[xnu]], xu = xnu;
        }
    }
    inline int get_lca(int xv, int xu)
    {
        for (int x = 1; x <= n_x; x++)
            book[x] = false;
        while (xv || xu)
168     {
            if (xv)
            {
                if (book[xv])
                    return xv;
                book[xv] = true;
                xv = bel[mate[xv]];
                if (xv)
                    xv = bel[fa[xv]];
            }
178         swap(xv, xu);
        }
        return 0;
    }

    inline void add_blossom(int xv, int xa, int xu)
    {
        int b = n + 1;
        while (b <= n_x && bel[b])
            b++;
188     if (b > n_x)
```

```
            n_x++;

        lab[b] = 0;
        col[b] = 0;

        mate[b] = mate[xa];

        bloch[b].clear();
        bloch[b].push_back(xa);
198     for (int x = xv; x != xa; x = bel[fa[bel[mate[x]]]])
            bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push(bel
        [mate[x]]);
        reverse(bloch[b].begin() + 1, bloch[b].end());
        for (int x = xu; x != xa; x = bel[fa[bel[mate[x]]]])
            bloch[b].push_back(x), bloch[b].push_back(bel[mate[x]]), q_push(bel
        [mate[x]]);

        set_bel(b, b);

        for (int x = 1; x <= n_x; x++)
        {
208         mat[b][x].w = mat[x][b].w = 0;
            blofrom[b][x] = 0;
        }
        for (int i = 0; i < size(bloch[b]); i++)
        {
            int xs = bloch[b][i];
            for (int x = 1; x <= n_x; x++)
                if (mat[b][x].w == 0 || e_delta(mat[xs][x]) < e_delta(mat[b][x])
        )
                    mat[b][x] = mat[xs][x], mat[x][b] = mat[x][xs];
            for (int x = 1; x <= n_x; x++)
218             if (blofrom[xs][x])
                    blofrom[b][x] = xs;
        }
        calc_slackv(b);
    }
    inline void expand_blossom1(int b) // lab[b] == 1
    {
        for (int i = 0; i < size(bloch[b]); i++)
            set_bel(bloch[b][i], bloch[b][i]);
```

```
228     int xr = blofrom[b][mat[b][fa[b]].v];
        int pr = find(bloch[b].begin(), bloch[b].end(), xr) - bloch[b].begin();
        if (pr % 2 == 1)
        {
            reverse(bloch[b].begin() + 1, bloch[b].end());
            pr = size(bloch[b]) - pr;
        }

        for (int i = 0; i < pr; i += 2)
        {
238         int xs = bloch[b][i], xns = bloch[b][i + 1];
            fa[xs] = mat[xns][xs].v;
            col[xs] = 1, col[xns] = 0;
            slackv[xs] = 0, calc_slackv(xns);
            q_push(xns);
        }
        col[xr] = 1;
        fa[xr] = fa[b];
        for (int i = pr + 1; i < size(bloch[b]); i++)
        {
248         int xs = bloch[b][i];
            col[xs] = -1;
            calc_slackv(xs);
        }

        bel[b] = 0;
    }
    inline void expand_blossom_final(int b) // at the final stage
    {
        for (int i = 0; i < size(bloch[b]); i++)
258     {
            if (bloch[b][i] > n && lab[bloch[b][i]] == 0)
                expand_blossom_final(bloch[b][i]);
            else
                set_bel(bloch[b][i], bloch[b][i]);
        }
        bel[b] = -1;
    }

    inline bool on_found_edge(const edge &e)
```

```
268 {
        int xv = bel[e.v], xu = bel[e.u];
        if (col[xu] == -1)
        {
            int nv = bel[mate[xu]];
            fa[xu] = e.v;
            col[xu] = 1, col[nv] = 0;
            slackv[xu] = slackv[nv] = 0;
            q_push(nv);
        }
278     else if (col[xu] == 0)
        {
            int xa = get_lca(xv, xu);
            if (!xa)
            {
                augment(xv, xu), augment(xu, xv);
                for (int b = n + 1; b <= n_x; b++)
                    if (bel[b] == b && lab[b] == 0)
                        expand_blossom_final(b);
                return true;
288         }
            else
                add_blossom(xv, xa, xu);
        }
        return false;
    }

    bool match()
    {
        for (int x = 1; x <= n_x; x++)
298         col[x] = -1, slackv[x] = 0;

        q_n = 0;
        for (int x = 1; x <= n_x; x++)
            if (bel[x] == x && !mate[x])
                fa[x] = 0, col[x] = 0, slackv[x] = 0, q_push(x);
        if (q_n == 0)
            return false;

        while (true)
308     {
```

```
        for (int i = 0; i < q_n; i++)
        {
            int v = q[i];
            for (int u = 1; u <= n; u++)
                if (mat[v][u].w > 0 && bel[v] != bel[u])
                {
                    int d = e_delta(mat[v][u]);
                    if (d == 0)
                    {
318                     if (on_found_edge(mat[v][u]))
                            return true;
                    }
                    else if (col[bel[u]] == -1 || col[bel[u]] == 0)
                        update_slackv(v, bel[u]);
                }
        }

        int d = INF;
        for (int v = 1; v <= n; v++)
328         if (col[bel[v]] == 0)
                tension(d, lab[v]);
        for (int b = n + 1; b <= n_x; b++)
            if (bel[b] == b && col[b] == 1)
                tension(d, lab[b] / 2);
        for (int x = 1; x <= n_x; x++)
            if (bel[x] == x && slackv[x])
            {
                if (col[x] == -1)
                    tension(d, e_delta(mat[slackv[x]][x]));
338             else if (col[x] == 0)
                    tension(d, e_delta(mat[slackv[x]][x]) / 2);
            }

        for (int v = 1; v <= n; v++)
        {
            if (col[bel[v]] == 0)
                lab[v] -= d;
            else if (col[bel[v]] == 1)
                lab[v] += d;
348     }
        for (int b = n + 1; b <= n_x; b++)
```

```
            if (bel[b] == b)
            {
                if (col[bel[b]] == 0)
                    lab[b] += d * 2;
                else if (col[bel[b]] == 1)
                    lab[b] -= d * 2;
            }

        q_n = 0;
        for (int v = 1; v <= n; v++)
            if (lab[v] == 0) // all unmatched vertices' labels are zero!
    cheers!
                return false;
        for (int x = 1; x <= n_x; x++)
            if (bel[x] == x && slackv[x] && bel[slackv[x]] != x && e_delta(
    mat[slackv[x]][x]) == 0)
            {
                if (on_found_edge(mat[slackv[x]][x]))
                    return true;
            }
        for (int b = n + 1; b <= n_x; b++)
            if (bel[b] == b && col[b] == 1 && lab[b] == 0)
                expand_blossom1(b);
    }
    return false;
}

void calc_max_weight_match()
{
    for (int v = 1; v <= n; v++)
        mate[v] = 0;

    n_x = n;
    n_matches = 0;
    tot_weight = 0;

    bel[0] = 0;
    for (int v = 1; v <= n; v++)
        bel[v] = v, bloch[v].clear();
    for (int v = 1; v <= n; v++)
        for (int u = 1; u <= n; u++)
```

```
            blofrom[v][u] = v == u ? v : 0;

        int w_max = 0;
        for (int v = 1; v <= n; v++)
            for (int u = 1; u <= n; u++)
                relax(w_max, mat[v][u].w);
        for (int v = 1; v <= n; v++)
            lab[v] = w_max;

398     while (match())
            n_matches++;

        for (int v = 1; v <= n; v++)
            if (mate[v] && mate[v] < v)
                tot_weight += mat[v][mate[v]].w;
    }

    public:
    int Main()
408 {
        n = getint(), m = getint();

        for (int v = 1; v <= n; v++)
            for (int u = 1; u <= n; u++)
                mat[v][u] = edge(v, u, 0);

        for (int i = 0; i < m; i++)
        {
            int v = getint(), u = getint(), w = getint();
418         mat[v][u].w = mat[u][v].w = w;
        }

        calc_max_weight_match();

        printf("%lld\n", tot_weight);
        for (int v = 1; v <= n; v++)
            printf("%d ", mate[v]);
        printf("\n");
    }
428
    };
```

```
int Match::INF = 2147483647;

int main()
{
    Match test(400);
    test.Main();
    return 0;
}
```

5.7 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

Performance

Problem	Constraints	Time	Memory	Date
JDFZ 2978	$N = 10^4, M = 10^5, K = 10^4$	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;
63      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){
        }
73      bool operator>(const edge&a){
            return l>a.l;
        }
        T l;
        int v;
    };
    struct edgeheap{
        edgeheap(node<edge>*r):
            root(r){
        }
83      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));
93  }
    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):
103         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){
113     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];
123             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])));
133         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)
143         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)));
153         }
            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;
163 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;
};
```

5.8 Least Common Ancestor

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Least Common Ancestor.hpp (1451 bytes, 51 lines)

```
#include<queue>
#include<vector>
namespace lca{
    using namespace std;
    const int N=10000,LGN=20;
    struct lca{
        vector<int>to[N+1];
        int n,up[N+1][LGN+1],dp[N+1],vis[N+10];
        lca(int _n):
            n(_n){
        }
        void add(int u,int v){
            to[u].push_back(v);
            to[v].push_back(u);
        }
        void build(){
            fill(vis+1,vis+n,0);
            queue<int>qu;
            qu.push(1);
            vis[1]=1;
            for(int i=0;i<=LGN;++i)
                up[1][i]=1;
            while(!qu.empty()){
                int u=qu.front();
                qu.pop();
                for(int i=1;i<=LGN;++i)
                    up[u][i]=up[up[u][i-1]][i-1];
                for(int v:to[u])
                    if(!vis[v]){
                        vis[v]=1;
                        up[v][0]=u;
                        dp[v]=dp[u]+1;
                        qu.push(v);
                    }
            }
        }
        int query(int u,int v){
            if(dp[u]<dp[v])
                swap(u,v);
            for(int i=0;i<=LGN;++i)
```

```
                if(((dp[u]-dp[v])>>i)&1)
                    u=up[u][i];
            if(u==v)
                return u;//注意不要漏掉这句代码
            for(int i=LGN;i>=0;--i)
                if(up[u][i]!=up[v][i])
                    u=up[u][i],v=up[v][i];
            return up[u][0];
        }
    };
}
```

5.9 Maximal Clique Count

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
};
```

5.10 Maximal Planarity Test

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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);
66      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;
76          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;
86              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;
96                  }
                }
                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;
    }
};
```

5.11 Minimum Product Spanning Tree

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Minimum Product Spanning Tree.hpp (0 bytes, 0 lines)

5.12 Minimum Spanning Arborescence

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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];
        }
    }
};
```

5.13 Minimum Spanning Tree

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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);
    }
};
```

5.14 Optimum Branching

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Optimum Branching.hpp (1556 bytes, 34 lines)

```
#ifndef OPTIMUM_BRANCHING
```

```
#define OPTIMUM_BRANCHING
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    template<class T>struct OptimumBranching{
        struct eg{int u,v;T w;};int n,rt;
        vector<eg>egs;vector<int>vi,in,id;vector<T>inw;
        OptimumBranching(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];}}};}
#endif
```

5.15 Shortest Path (Dijkstra's Algorithm)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Shortest Path (Dijkstra's Algorithm).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;
    }
};
```

5.16 Shortest Path (SPFA)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Shortest Path (SPFA).hpp (1078 bytes, 43 lines)

```
#include<algorithm>
#include<queue>
#include<vector>
using namespace std;
const int N=100000;
template<class COST>struct SPFA{
    int n,src,vis[N],in[N];
    COST di[N];
    vector<int>to[N];
    vector<COST>we[N];
    SPFA(int _n,int _src):
        n(_n),src(_src-1){}
    void add(int u,int v,COST w){
        to[u-1].push_back(v-1);
        we[u-1].push_back(w);
    }
    void run(){
        di[src]=0;
        fill(vis,vis+n,0);
        vis[src]=1;
        fill(in,in+n,0);
        in[src]=1;
        queue<int>qu;
        qu.push(src);
        while(!qu.empty()){
            int u=qu.front();
            qu.pop();
```

```
            in[u]=0;
            for(int i=0;i<to[u].size();++i){
                int v=to[u][i];
                COST w=we[u][i];
                if(!vis[v]||di[u]+w<di[v]){
                    vis[u]=1;
                    di[v]=di[u]+w;
                    if(!in[v]){
                        in[v]=1;
                        qu.push(v);
                    }
                }
            }
        }
    }
};
```

5.17 Steiner Tree

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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]);
    }
```

```
18      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);
        }
28      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);
38              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]);
48              }
                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;
        }
    };
```

5.18 Theorems

考虑一棵树 **T**，**T1** 是 **T** 接上一个节点 **u**，**T2** 是 **T** 接上另外一个节点 **v**。则无根树 **T1** 和 **T2** 的同构等价于以 **u** 为根的有根树 **T1** 和以 **v** 为根的有根树 **T2** 同构。

证明方法考虑以树的规模来进行数学归纳法，同时配合反证。

5.19 Tree Hashing

Description

给定一棵树，对于每一个节点计算出以其为根的 hash 值。同构的有根树的 hash 值保证相同。这个 hash 值是一个 long long，因为用了两个质数。冲突率非常低。时间复杂度是 O(nlgn) 的。

Tree Hashing.hpp (1689 bytes, 71 lines)

```
#include<bits/stdc++.h>
using namespace std;
struct TreeHashing{
    int n,p,d2;
    vector<int>hu,hd,sz;
    vector<vector<int>>adj;
    vector<long long>pw,hash;
    TreeHashing(int _n):
        n(_n),pw(2*n),hu(n+1),hd(n+1),sz(n+1,1),
        d2(0),adj(n+1),hash(n+1),p(1e9+7){
    }
    void add(int u,int v){
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    void down(int u,int d1=0,int f=0){
        vector<pair<int,int>>t;
        for(int v:adj[u])
            if(v!=f){
                down(v,d1,u);
                if(!d2)
                    sz[u]+=sz[v]*(1-d1);
                t.push_back(make_pair(sz[v]*2,hd[v]));
```

```
            }
        if(d1*f)
            t.push_back(make_pair((n-sz[u])*2,hu[u]));
        sort(t.begin(),t.end());
        int l=1,&s=d1?*((int*)&hash[u]+d2):(hd[u]=0);
        for(auto i:t){
            s=(s+pw[l]*i.second)%p;
            l+=i.first;
        }
        s=(s+pw[l])%p;
    }
    void up(int u,int f=0){
        int l=0,sl=0,sr=1;
        vector<pair<pair<int,int>,int>>t;
        if(f)
            t.push_back(make_pair(make_pair((n-sz[u])*2,hu[u]),0));
        for(int i=0;i<adj[u].size();++i)
            if(adj[u][i]!=f){
                int v=adj[u][i];
                t.push_back(make_pair(make_pair(sz[v]*2,hd[v]),v));
            }
        sort(t.begin(),t.end());
        for(auto i:t){
            sl=(sl+i.first.second*pw[l])%p;
            l+=i.first.first;
        }
        for(int i=int(t.size()-1);i>=0;--i){
            sl=(sl-t[i].first.second*pw[l-=t[i].first.first]%p+p)%p;
            if(i+1<t.size())
                sr=(sr*pw[t[i+1].first.first]+t[i+1].first.second)%p;
            if(t[i].second)
                hu[t[i].second]=(sl+sr*pw[l])*pw[1]%p;
        }
        for(int v:adj[u])
            if(v!=f)
                up(v,u);
    }
    void run(){
        pw[0]=1;
        for(int i=1;i<2*n;++i)
            pw[i]=pw[i-1]*2%p;
```

```
        down(1),up(1),down(1,1);
        d2=1,p=1e9+9;
        for(int i=1;i<2*n;++i)
            pw[i]=pw[i-1]*7%p;
        down(1),up(1),down(1,1);
    }
};
```

5.20 Virtual Tree

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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 6

Linear Programming

6.1 Linear Programming

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Linear Programming.hpp (2225 bytes, 52 lines)

```
#ifndef LINEAR_PROGRAMMING
#define LINEAR_PROGRAMMING
#include<bits/stdc++.h>
namespace CTL{
    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=-l;}
        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);
43          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;}};}
#endif
```

6.2 Maximum Flow

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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):
8           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];
58                  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];
68              }
                if(!--gap[hei[u]])
                    break;
                ++gap[hei[u]=mh+1];
                cur[u]=0;
                if(u!=src)
                    u=egs[frm[u]^1].v;
            }
        }
        return r;
78  }
};
```

6.3 Minimum Cost Maximum Flow

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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

```
1   #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;
};
```

CHAPTER 7

Game Theory

7.1 K-Based Dynamic Subtraction Game

warning: old style will be replaced ... see Suffix Array (DC3) for new style

K-Based Dynamic Subtraction Game.hpp (565 bytes, 14 lines)

```
#ifndef K_BASED_DYNAMIC_SUBTRACTION_GAME
#define K_BASED_DYNAMIC_SUBTRACTION_GAME
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    namespace KBasedDynamicSubtractionGame{
        int run(int n,int k){
8           vector<int>a=vector<int>(1,1),b=a;
            for(int i=-1;b.back()<n;b.push_back(a.back()+b[i]*(i>=0)))
                for(a.push_back(b.back()+1);a[i+1]*k<a.back();++i);
            if(a.back()==n)return 0;
            for(int i=a.size()-1;i>=0;n-=a[i]*(n>a[i]),--i)
                if(n==a[i])return a[i];}}}
#endif
```

7.2 Symmetric Game Of No Return

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Symmetric Game Of No Return.hpp (2665 bytes, 64 lines)

```
#ifndef SYMMETRIC_GAME_OF_NO_RETURN
#define SYMMETRIC_GAME_OF_NO_RETURN
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    struct SymmetricGameOfNoReturn{
7       int n;vector<int>mh,nx,mk,vs,tp,pr,rk;
        vector<bool>wi;queue<int>qu;
        vector<vector<int> >to;
        SymmetricGameOfNoReturn(int _n):
            n(_n),mh(n+1),nx(n+1),mk(n+1),wi(n+1,true),
            vs(n+1),tp(n+1),to(n+1),pr(n+1),rk(n+1){}
        int fd(int x){return x==pr[x]?x:pr[x]=fd(pr[x]);}
```

```
        void lk(int x,int y){
            if(rk[x=fd(x)]>rk[y=fd(y)])pr[y]=x;
            else if(rk[x]<rk[y])pr[x]=y;
17          else pr[x]=y,++rk[y];}
        int lca(int x,int y){
            static int t;++t;
            for(;;swap(x,y))if(x){
                x=tp[fd(x)];if(vs[x]==t)return x;vs[x]=t;
                if(mh[x])x=nx[mh[x]];else x=0;}}
        void uni(int x,int p){
            for(;fd(x)!=fd(p);){
                int y=mh[x],z=nx[y];
                if(fd(z)!=fd(p))nx[z]=y;
27              if(mk[y]==2)mk[y]=1,qu.push(y);
                if(mk[z]==2)mk[z]=1,qu.push(z);
                int t=tp[fd(z)];lk(x,y);lk(y,z);
                tp[fd(z)]=t;x=z;}}
        void aug(int s,int t){
            for(int i=1;i<=n;++i)
                nx[i]=0,mk[i]=0,tp[i]=i,pr[i]=i,rk[i]=0;
            mk[s]=1;qu=queue<int>();
            for(qu.push(s);!qu.empty();){
                int x=qu.front();qu.pop();if(t)wi[x]=0;
37              for(int i=0;i<to[x].size();++i){
                    int y=to[x][i];
                    if(mh[x]==y||fd(x)==fd(y)||mk[y]==2)
                        continue;
                    if(mk[y]==1){
                        int z=lca(x,y);
                        if(fd(x)!=fd(z))nx[x]=y;
                        if(fd(y)!=fd(z))nx[y]=x;
                        uni(x,z);uni(y,z);
                    }else if(!mh[y]){
47                      nx[y]=x;while(y){
                            int z=nx[y],mz=mh[z];
                            mh[z]=y;mh[y]=z;y=mz;}
                        return;
                    }else{
                        nx[y]=x;mk[mh[y]]=1;
                        qu.push(mh[y]);mk[y]=2;}}}}
        void add(int x,int y){
```

```
            to[x].push_back(y);to[y].push_back(x);}
        vector<bool>run(){
57          for(int i=1;i<=n;++i)if(!mh[i])
                for(int j=0;j<to[i].size();++j)
                    if(!mh[to[i][j]]){
                        mh[to[i][j]]=i;mh[i]=to[i][j];break;}
            for(int i=1;i<=n;++i)if(!mh[i])aug(i,0);
            for(int i=1;i<=n;++i)if(!mh[i])aug(i,1);
            return wi;}};}
#endif
```

CHAPTER 8

Number Theory

8.1 Discrete Logarithm

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
37      }
        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);
47          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;
57              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)
67              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;
        }
    }
```

8.2 Discrete Square Root

Description

Find the solutions to $x^2 \equiv a \pmod{n}$.

Methods

vector<int>run(int a,int n);	
Description	find all solutions to the equation that are less than n
Parameters	**Description**
a	a in the equation, should be less than n
n	n in the equation
Time complexity	$O(\sqrt{n}\log n)$ (expected)
Space complexity	$O(\sqrt{n}\log n)$
Return value	all solutions in a vector, not sorted

Performance

Problem	Constraints	Time	Memory	Date
UVaOJ 1426	$N = 10^9$	23 ms		2016-02-19

Code

Discrete Square Root.hpp (3692 bytes, 122 lines)

```
#include<cmath>
#include<vector>
using namespace std;
namespace DiscreteSquareRoot{
    typedef long long ll;
    int ti[1<<16],va[1<<16],mp[1<<16],nx[1<<16],hd[1<<16],tm,nw;
    #define clr\
        int y=x&65535;\
        if(ti[y]!=tm)ti[y]=tm,hd[y]=0;
    int*get(int x){
        clr
        for(int i=hd[y];i;i=nx[i])
            if(va[i]==x)return&mp[i];
```

```
        return 0;
    }
    void ins(int x,int v){
        clr
        va[++nw]=x,mp[nw]=v;
        nx[nw]=hd[y],hd[y]=nw;
    }
    int pow(int a,int b,int n){
        int r=1;
        for(;b;b&1?r=(ll)r*a%n:0,b>>=1,a=(ll)a*a%n);
        return r;
    }
    int gcd(int a,int b){
        return b?gcd(b,a%b):a;
    }
    void exg(int a,int b,int&x,int&y){
        if(!b)x=1,y=0;
        else exg(b,a%b,y,x),y-=a/b*x;
    }
    int inv(int a,int b){
        int x,y;
        exg(a,b,x,y);
        return x+b;
    }
    int bgs(int a,int b,int n){
        ++tm,nw=0;
        int m=sqrt(n);
        for(int i=0,u=1;i<m;++i)
            ins(u,i),u=(ll)u*a%n;
        for(int i=0,u=1,v=inv(pow(a,m,n),n);i*m<=n;++i){
            int t=(ll)u*b%n,*j=get(t);
            if(j)return i*m+*j;
            u=(ll)u*v%n;
        }
        return -1;
    }
    int prt(int p,int pk){
        if(p==2)return 5;
        int pi=pk/p*(p-1);
        vector<int>t;
        for(int i=2;i*i<=pi;++i)
```

```
            if(pi%i==0)
                t.push_back(i),t.push_back(pi/i);
        for(int g=2;;++g){
            int f=1;
            for(int i=0;i<t.size();++i)
                if(pow(g,t[i],pk)==1){f=0;break;}
            if(f)return g;
        }
    }
    int phi(int p,int pk){
        return p-2?pk/p*(p-1)/2:pk/8;
    }
    vector<int>apk(int a,int p,int k,int pk){
        vector<int>r;
        if(!a)
            for(int d=pow(p,k+1>>1,pk+1),x=0;x<pk;x+=d)
                r.push_back(x);
        else if(gcd(a,pk)==1){
            if(p==2&&k<=2){
                for(int i=1;i<pk;++i)
                    if(i*i%pk==a)r.push_back(i);
            }else{
                int ia,g=prt(p,pk);
                if((ia=bgs(g,a,pk))!=-1&&ia%2==0){
                    r.push_back(pow(g,ia/2,pk));
                    r.push_back(pow(g,ia/2+phi(p,pk),pk));
                    if(p==2){
                        r.push_back(pk-pow(g,ia/2,pk));
                        r.push_back(pk-pow(g,ia/2+phi(p,pk),pk));
                    }
                }
            }
        }else{
            int l=0,pl2=1;
            for(;a%p==0;++l,a/=p,pl2*=(l%2?1:p));
            if(l%2==0)r=apk(a,p,k-l,pk/pl2/pl2);
            for(int i=r.size()-1;l%2==0&&i>=0;--i)
                for(int j=(r[i]*=pl2,1);j<pl2;++j)
                    r.push_back(r[i]+pk/pl2*j);
        }
        return r;
```

```
    }
    vector<int>mer(vector<int>a,int&n,vector<int>b,int m){
        vector<int>r;
        for(int i=0;i<a.size();++i)
            for(int j=0;j<b.size();++j){
                ll t=(ll)m*inv(m,n)*a[i]+(ll)n*inv(n,m)*b[j];
                r.push_back(t%(n*m));
            }
        return n*=m,r;
    }
    vector<int>run(int a,int n){
        vector<int>r,t;int m;
        if(n==1)return vector<int>(1);
        for(int p=2,k,pk;p*p<=n;++p)
            if(n%p==0){
                for(k=0,pk=1;n%p==0;++k,n/=p,pk*=p);
                if((t=apk(a%pk,p,k,pk)).size())
                    r=r.size()?mer(r,m,t,pk):(m=pk,t);
                else
                    return vector<int>();
            }
        if(n==1)return r;
        if((t=apk(a%n,n,1,n)).size())
            return r.size()?mer(r,m,t,n):t;
        return vector<int>();
    }
}
```

8.3 Divisor

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Divisor.hpp (471 bytes, 13 lines)

```
#ifndef DIVISOR
#define DIVISOR
#include<bits/stdc++.h>
namespace CTL{
    namespace Divisor{
        template<class T>void dfs(vector<pair<T,int> >&a,
```

```
            int i,T now,vector<T>&r){
8           if(i==a.size()){r.push_back(now);return;}
            for(int j=0;j<=a[i].second;++j,now*=a[i].first){
                dfs(a,i+1,now,r);}}
        template<class T>vector<T>run(vector<pair<T,int> >a){
            vector<T> r;dfs(a,0,1,r);return r;}}}
#endif
```

8.4 Eulers Totient Function

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Eulers Totient Function.hpp (592 bytes, 16 lines)

```
#ifndef EULERS_TOTIENT_FUNCTION
#define EULERS_TOTIENT_FUNCTION
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    namespace EulersTotientFunction{
        vector<int>run(int n){
8           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]=i-1;
                for(int j=0;j<p.size()&&p[j]*i<=n;++j){
                    ntp[p[j]*i]=1;
                    if(i%p[j]==0){u[p[j]*i]=u[i]*p[j];break;}
                    else u[p[j]*i]=u[i]*(p[j]-1);}}
            return u;}}}
#endif
```

8.5 Greatest Common Divisor

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Greatest Common Divisor.hpp (254 bytes, 15 lines)

```
typedef long long ll;
ll gcd(ll a,ll b){
```

```
    return b?gcd(b,a%b):a;
}
ll egcd(ll a,ll b,ll&x,ll&y){
    if(!b){
        x=1;
        y=0;
        return a;
    }else{
        ll d=egcd(b,a%b,y,x);
        y-=a/b*x;
        return d;
    }
}
```

8.6 Integer Factorization (Pollard's Rho Algorithm)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
}
```

8.7 Integer Factorization (Shanks' Square Forms Factorization)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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];
47      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;\
57              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)
67              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){\
77                  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;
    }
}
```

8.8 Modular Integer

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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);
53      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);
63  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;
}
73 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;
```

8.9 Möbius Function

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
}
```

8.10 Nth Root Modulo M

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Nth Root Modulo M.hpp (4098 bytes, 97 lines)

```
#ifndef N_TH_ROOT_MODULO_M
#define N_TH_ROOT_MODULO_M
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    namespace NthRootModuloM{
        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;}
        int chk(T a,int c=10){
            if(a==1)return 0;
            T u=a-1,t=0;for(;u%2==0;u/=2,++t);
            for(int i=0;i<c;++i){
                T x=pow(rand()*1.0/RAND_MAX*(a-2)+1,u,a),y;
                for(int j=0;j<t;++j){
                    y=x,x=x*x%a;
                    if(x==1&&y!=1&&y!=a-1)return 0;}
                if(x!=1)return 0;}
            return 1;}
        T gcd(T a,T b){
            if(a<0)a=-a;if(b<0)b=-b;return b?gcd(b,a%b):a;}
        T rho(T a,T c){
            T x=double(rand())/RAND_MAX*(a-1),y=x;
            for(int i=1,k=2;;){
                x=(x*x%a+c)%a;T d=gcd(y-x,a);
                if(d!=1&&d!=a)return d;
```

```
29              if(y==x)return a;
                if(++i==k)y=x,k=2*k;}}
        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,double(rand())/RAND_MAX*(a-1)))
                ==a);
39          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;}
49      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);}}
        T prt(T a){
            T pa=a-1;
            vector<pair<T,int> >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)
59                  if(fs[i]!=pa&&pow(g,fs[i],a)==1){f=1;break;}
                if(!f)return g;}}
        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];}
69          return -1;}
```

```
        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;}
79          return -1;}
        T pow(T a,T b){return b?pow(a,b-1)*a:1;}
        T spk(T a,T b,T p,T k){
            T pk=1;for(int i=1;i<=k;++i)pk*=p;b%=pk;
            if(!b)return pow(p,k-1-(k-1)/a);
            T c0=0,b0=b;while(b0%p==0)b0/=p,++c0,pk/=p;
            if(c0%a)return 0;
            T g=prt(p),ib0=bgs(g,b0,pk),
                ppk=pk/p*(p-1),d=gcd(a,ppk);
            return ib0%d?0:d*pow(p,c0-c0/a);}
89      T run(T a,T b,T c){
            b=(b%c+c)%c;if(c==1)return 1;
            if(a==0)return b==1?c:0;
            T r=1;vector<pair<T,int> >fa=fac(c);
            for(int i=0;i<fa.size();++i)
                if(!(r*=spk(a,b,fa[i].first,fa[i].second)))
                    return 0;
            return r;}}}
#endif
```

8.11 Primality Test

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Primality Test.hpp (1509 bytes, 52 lines)

```
#include<bits/stdc++.h>
using namespace std;
namespace PrimalityTest{
4   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;
    }
}
```

8.12 Prime Number

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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);
    }
}
```

8.13 Primitive Root Modulo M

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Primitive Root Modulo M.hpp (2949 bytes, 71 lines)

```
#ifndef PRIMITIVE_ROOT_MODULO_M
#define PRIMITIVE_ROOT_MODULO_M
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    namespace PrimitiveRootModuloM{
        typedef long long T;
        T mul(T x,T y,T z){
            return (x*y-(T)(((long double)x*y+0.5)/
                (long double)z)*z+z)%z;}
        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;}
        int chk(T a,int c=10){
            if(a==1)return 0;
            T u=a-1,t=0;for(;u%2==0;u/=2,++t);
            for(int i=0;i<c;++i){
                T x=pow(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 0;}
                if(x!=1)return 0;}
            return 1;}
        T gcd(T a,T b){
            if(a<0)a=-a;if(b<0)b=-b;return b?gcd(b,a%b):a;}
        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;}}
        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,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;}
        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);}}
        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(2ll,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;}}}}
#endif
```

8.14 Primitive Root

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
        }
    }
}
```

8.15 Sequences

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 9

Numerical Algorithms

9.1 Convolution (Fast Fourier Transform)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
}
```

9.2 Convolution (Karatsuba Algorithm)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
}
```

9.3 Convolution (Number Theoretic Transform)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
27                  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;
37      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);
47      for(T&v:a)
            v=v*n%p;
        return a.resize(tn),a;
    }
}
```

9.4 Fraction

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
}
```

9.5 Integer

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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);
}
```

9.6 Integral Table

含有 $ax+b$ 的积分 **($a \neq 0$)**

1. $\int \frac{x}{ax+b} = \frac{1}{a}\ln|ax+b| + C$
2. $\int (ax+b)^{\mu}x = \frac{1}{a(\mu+1)}(ax+b)^{\mu+1} + C(\mu \neq 1)$
3. $\int \frac{x}{ax+b}x = \frac{1}{a^2}(ax+b-b\ln|ax+b|) + C$
4. $\int \frac{x^2}{ax+b}x = \frac{1}{a^3}\left(\frac{1}{2}(ax+b)^2 - 2b(ax+b) + b^2\ln|ax+b|\right) + C$
5. $\int \frac{x}{x(ax+b)} = -\frac{1}{b}\ln\left|\frac{ax+b}{x}\right| + C$
6. $\int \frac{x}{x^2(ax+b)} = -\frac{1}{bx} + \frac{a}{b^2}\ln\left|\frac{ax+b}{x}\right| + C$
7. $\int \frac{x}{(ax+b)^2}x = \frac{1}{a^2}\left(\ln|ax+b| + \frac{b}{ax+b}\right) + C$
8. $\int \frac{x^2}{(ax+b)^2}x = \frac{1}{a^3}\left(ax+b-2b\ln|ax+b| - \frac{b^2}{ax+b}\right) + C$
9. $\int \frac{x}{x(ax+b)^2} = \frac{1}{b(ax+b)} - \frac{1}{b^2}\ln\left|\frac{ax+b}{x}\right| + C$

含有 $\sqrt{ax+b}$ 的积分

1. $\int \sqrt{ax+b}\mathrm{d}x = \frac{2}{3a}\sqrt{(ax+b)^3} + C$
2. $\int x\sqrt{ax+b}\mathrm{d}x = \frac{2}{15a^2}(3ax-2b)\sqrt{(ax+b)^3} + C$
3. $\int x^2\sqrt{ax+b}\mathrm{d}x = \frac{2}{105a^3}(15a^2x^2 - 12abx + 8b^2)\sqrt{(ax+b)^3} + C$
4. $\int \frac{x}{\sqrt{ax+b}}\mathrm{d}x = \frac{2}{3a^2}(ax-2b)\sqrt{ax+b} + C$

5. $\int \frac{x^2}{\sqrt{ax+b}}\mathrm{d}x = \frac{2}{15a^3}(3a^2x^2 - 4abx + 8b^2)\sqrt{ax+b} + C$

6. $\int \frac{\mathrm{d}x}{x\sqrt{ax+b}} = \begin{cases} \frac{1}{\sqrt{b}} \ln\left|\frac{\sqrt{ax+b}-\sqrt{b}}{\sqrt{ax+b}+\sqrt{b}}\right| + C & (b > 0) \\ \frac{2}{\sqrt{-b}} \arctan\sqrt{\frac{ax+b}{-b}} + C & (b < 0) \end{cases}$

7. $\int \frac{\mathrm{d}x}{x^2\sqrt{ax+b}} = -\frac{\sqrt{ax+b}}{bx} - \frac{a}{2b}\int \frac{\mathrm{d}x}{x\sqrt{ax+b}}$

8. $\int \frac{\sqrt{ax+b}}{x}\mathrm{d}x = 2\sqrt{ax+b} + b\int \frac{\mathrm{d}x}{x\sqrt{ax+b}}$

9. $\int \frac{\sqrt{ax+b}}{x^2}\mathrm{d}x = -\frac{\sqrt{ax+b}}{x} + \frac{a}{2}\int \frac{\mathrm{d}x}{x\sqrt{ax+b}}$

含有 $x^2 \pm a^2$ 的积分

1. $\int \frac{\mathrm{d}x}{x^2+a^2} = \frac{1}{a} \arctan\frac{x}{a} + C$

2. $\int \frac{\mathrm{d}x}{(x^2+a^2)^n} = \frac{x}{2(n-1)a^2(x^2+a^2)^{n-1}} + \frac{2n-3}{2(n-1)a^2}\int \frac{\mathrm{d}x}{(x^2+a^2)^{n-1}}$

3. $\int \frac{\mathrm{d}x}{x^2-a^2} = \frac{1}{2a} \ln\left|\frac{x-a}{x+a}\right| + C$

含有 $ax^2 + b(a > 0)$ 的积分

1. $\int \frac{\mathrm{d}x}{ax^2+b} = \begin{cases} \frac{1}{\sqrt{ab}} \arctan\sqrt{\frac{a}{b}}x + C & (b > 0) \\ \frac{1}{2\sqrt{-ab}} \ln\left|\frac{\sqrt{a}x-\sqrt{-b}}{\sqrt{a}x+\sqrt{-b}}\right| + C & (b < 0) \end{cases}$

2. $\int \frac{x}{ax^2+b}\mathrm{d}x = \frac{1}{2a} \ln\left|ax^2+b\right| + C$

3. $\int \frac{x^2}{ax^2+b}\mathrm{d}x = \frac{x}{a} - \frac{b}{a}\int \frac{\mathrm{d}x}{ax^2+b}$

4. $\int \frac{\mathrm{d}x}{x(ax^2+b)} = \frac{1}{2b} \ln\frac{x^2}{|ax^2+b|} + C$

5. $\int \frac{\mathrm{d}x}{x^2(ax^2+b)} = -\frac{1}{bx} - \frac{a}{b}\int \frac{\mathrm{d}x}{ax^2+b}$

6. $\int \frac{\mathrm{d}x}{x^3(ax^2+b)} = \frac{a}{2b^2} \ln\frac{|ax^2+b|}{x^2} - \frac{1}{2bx^2} + C$

7. $\int \frac{\mathrm{d}x}{(ax^2+b)^2} = \frac{x}{2b(ax^2+b)} + \frac{1}{2b}\int \frac{\mathrm{d}x}{ax^2+b}$

含有 $ax^2+bx+c(a>0)$ 的积分

1. $\frac{x}{ax^2+bx+c}=\begin{cases}\frac{2}{\sqrt{4ac-b^2}}\arctan\frac{2ax+b}{\sqrt{4ac-b^2}}+C & (b^2<4ac)\\ \frac{1}{\sqrt{b^2-4ac}}\ln\left|\frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|+C & (b^2>4ac)\end{cases}$

2. $\int\frac{x}{ax^2+bx+c}x=\frac{1}{2a}\ln|ax^2+bx+c|-\frac{b}{2a}\int\frac{x}{ax^2+bx+c}$

含有 $\sqrt{x^2+a^2}(a>0)$ 的积分

1. $\int\frac{\mathrm{d}x}{\sqrt{x^2+a^2}}=\operatorname{arsh}\frac{x}{a}+C_1=\ln(x+\sqrt{x^2+a^2})+C$

2. $\int\frac{\mathrm{d}x}{\sqrt{(x^2+a^2)^3}}=\frac{x}{a^2\sqrt{x^2+a^2}}+C$

3. $\int\frac{x}{\sqrt{x^2+a^2}}\mathrm{d}x=\sqrt{x^2+a^2}+C$

4. $\int\frac{x}{\sqrt{(x^2+a^2)^3}}\mathrm{d}x=-\frac{1}{\sqrt{x^2+a^2}}+C$

5. $\int\frac{x^2}{\sqrt{x^2+a^2}}x=\frac{x}{2}\sqrt{x^2+a^2}-\frac{a^2}{2}\ln(x+\sqrt{x^2+a^2})+C$

6. $\int\frac{x^2}{\sqrt{(x^2+a^2)^3}}x=-\frac{x}{\sqrt{x^2+a^2}}+\ln(x+\sqrt{x^2+a^2})+C$

7. $\int\frac{x}{x\sqrt{x^2+a^2}}=\frac{1}{a}\ln\frac{\sqrt{x^2+a^2}-a}{|x|}+C$

8. $\int\frac{x}{x^2\sqrt{x^2+a^2}}=-\frac{\sqrt{x^2+a^2}}{a^2x}+C$

9. $\int\sqrt{x^2+a^2}x=\frac{x}{2}\sqrt{x^2+a^2}+\frac{a^2}{2}\ln(x+\sqrt{x^2+a^2})+C$

10. $\int\sqrt{(x^2+a^2)^3}x=\frac{x}{8}(2x^2+5a^2)\sqrt{x^2+a^2}+\frac{3}{8}a^4\ln(x+\sqrt{x^2+a^2})+C$

11. $\int x\sqrt{x^2+a^2}x=\frac{1}{3}\sqrt{(x^2+a^2)^3}+C$

12. $\int x^2\sqrt{x^2+a^2}x=\frac{x}{8}(2x^2+a^2)\sqrt{x^2+a^2}-\frac{a^4}{8}\ln(x+\sqrt{x^2+a^2})+C$

13. $\int\frac{\sqrt{x^2+a^2}}{x}x=\sqrt{x^2+a^2}+a\ln\frac{\sqrt{x^2+a^2}-a}{|x|}+C$

14. $\int\frac{\sqrt{x^2+a^2}}{x^2}x=-\frac{\sqrt{x^2+a^2}}{x}+\ln(x+\sqrt{x^2+a^2})+C$

含有 $\sqrt{x^2-a^2}(a>0)$ 的积分

1. $\int \frac{x}{\sqrt{x^2-a^2}} = \frac{x}{|x|}\operatorname{arch}\frac{|x|}{a} + C_1 = \ln\left|x+\sqrt{x^2-a^2}\right| + C$
2. $\int \frac{x}{\sqrt{(x^2-a^2)^3}} = -\frac{x}{a^2\sqrt{x^2-a^2}} + C$
3. $\int \frac{x}{\sqrt{x^2-a^2}}\mathrm{d}x = \sqrt{x^2-a^2} + C$
4. $\int \frac{x}{\sqrt{(x^2-a^2)^3}}\mathrm{d}x = -\frac{1}{\sqrt{x^2-a^2}} + C$
5. $\int \frac{x^2}{\sqrt{x^2-a^2}}x = \frac{x}{2}\sqrt{x^2-a^2} + \frac{a^2}{2}\ln|x+\sqrt{x^2-a^2}| + C$
6. $\int \frac{x^2}{\sqrt{(x^2-a^2)^3}}x = -\frac{x}{\sqrt{x^2-a^2}} + \ln|x+\sqrt{x^2-a^2}| + C$
7. $\int \frac{x}{x\sqrt{x^2-a^2}} = \frac{1}{a}\arccos\frac{a}{|x|} + C$
8. $\int \frac{x}{x^2\sqrt{x^2-a^2}} = \frac{\sqrt{x^2-a^2}}{a^2x} + C$
9. $\int \sqrt{x^2-a^2}x = \frac{x}{2}\sqrt{x^2-a^2} - \frac{a^2}{2}\ln|x+\sqrt{x^2-a^2}| + C$
10. $\int \sqrt{(x^2-a^2)^3}x = \frac{x}{8}(2x^2-5a^2)\sqrt{x^2-a^2} + \frac{3}{8}a^4\ln|x+\sqrt{x^2-a^2}| + C$
11. $\int x\sqrt{x^2-a^2}x = \frac{1}{3}\sqrt{(x^2-a^2)^3} + C$
12. $\int x^2\sqrt{x^2-a^2}x = \frac{x}{8}(2x^2-a^2)\sqrt{x^2-a^2} - \frac{a^4}{8}\ln|x+\sqrt{x^2-a^2}| + C$
13. $\int \frac{\sqrt{x^2-a^2}}{x}x = \sqrt{x^2-a^2} - a\arccos\frac{a}{|x|} + C$
14. $\int \frac{\sqrt{x^2-a^2}}{x^2}x = -\frac{\sqrt{x^2-a^2}}{x} + \ln|x+\sqrt{x^2-a^2}| + C$

含有 $\sqrt{a^2-x^2}(a>0)$ 的积分

1. $\int \frac{x}{\sqrt{a^2-x^2}} = \arcsin\frac{x}{a} + C$
2. $\frac{x}{\sqrt{(a^2-x^2)^3}} = \frac{x}{a^2\sqrt{a^2-x^2}} + C$
3. $\int \frac{x}{\sqrt{a^2-x^2}}x = -\sqrt{a^2-x^2} + C$
4. $\int \frac{x}{\sqrt{(a^2-x^2)^3}}x = \frac{1}{\sqrt{a^2-x^2}} + C$
5. $\int \frac{x^2}{\sqrt{a^2-x^2}}x = -\frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a} + C$

6. $\int \frac{x^2}{\sqrt{(a^2-x^2)^3}}x = \frac{x}{\sqrt{a^2-x^2}} - \arcsin\frac{x}{a} + C$
7. $\int \frac{x}{x\sqrt{a^2-x^2}} = \frac{1}{a}\ln\frac{a-\sqrt{a^2-x^2}}{|x|} + C$
8. $\int \frac{x}{x^2\sqrt{a^2-x^2}} = -\frac{\sqrt{a^2-x^2}}{a^2x} + C$
9. $\int \sqrt{a^2-x^2}x = \frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a} + C$
10. $\int \sqrt{(a^2-x^2)^3}x = \frac{x}{8}(5a^2-2x^2)\sqrt{a^2-x^2} + \frac{3}{8}a^4\arcsin\frac{x}{a} + C$
11. $\int x\sqrt{a^2-x^2}x = -\frac{1}{3}\sqrt{(a^2-x^2)^3} + C$
12. $\int x^2\sqrt{a^2-x^2}x = \frac{x}{8}(2x^2-a^2)\sqrt{a^2-x^2} + \frac{a^4}{8}\arcsin\frac{x}{a} + C$
13. $\int \frac{\sqrt{a^2-x^2}}{x}x = \sqrt{a^2-x^2} + a\ln\frac{a-\sqrt{a^2-x^2}}{|x|} + C$
14. $\int \frac{\sqrt{a^2-x^2}}{x^2}x = -\frac{\sqrt{a^2-x^2}}{x} - \arcsin\frac{x}{a} + C$

含有 $\sqrt{\pm ax^2+bx+c}(a>0)$ 的积分

1. $\int \frac{x}{\sqrt{ax^2+bx+c}} = \frac{1}{\sqrt{a}}\ln|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}| + C$
2. $\int \sqrt{ax^2+bx+c}x = \frac{2ax+b}{4a}\sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8\sqrt{a^3}}\ln|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}| + C$
3. $\int \frac{x}{\sqrt{ax^2+bx+c}}x = \frac{1}{a}\sqrt{ax^2+bx+c} - \frac{b}{2\sqrt{a^3}}\ln|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}| + C$
4. $\int \frac{x}{\sqrt{c+bx-ax^2}} = -\frac{1}{\sqrt{a}}\arcsin\frac{2ax-b}{\sqrt{b^2+4ac}} + C$
5. $\int \sqrt{c+bx-ax^2}x = \frac{2ax-b}{4a}\sqrt{c+bx-ax^2} + \frac{b^2+4ac}{8\sqrt{a^3}}\arcsin\frac{2ax-b}{\sqrt{b^2+4ac}} + C$
6. $\int \frac{x}{\sqrt{c+bx-ax^2}}x = -\frac{1}{a}\sqrt{c+bx-ax^2} + \frac{b}{2\sqrt{a^3}}\arcsin\frac{2ax-b}{\sqrt{b^2+4ac}} + C$

含有 $\sqrt{\pm\frac{x-a}{x-b}}$ 或 $\sqrt{(x-a)(x-b)}$ 的积分

1. $\int \sqrt{\frac{x-a}{x-b}}x = (x-b)\sqrt{\frac{x-a}{x-b}} + (b-a)\ln(\sqrt{|x-a|} + \sqrt{|x-b|}) + C$
2. $\int \sqrt{\frac{x-a}{b-x}}x = (x-b)\sqrt{\frac{x-a}{b-x}} + (b-a)\arcsin\sqrt{\frac{x-a}{b-x}} + C$
3. $\int \frac{x}{\sqrt{(x-a)(b-x)}} = 2\arcsin\sqrt{\frac{x-a}{b-x}} + C\,(a<b)$

4.

$$\int \sqrt{(x-a)(b-x)}x = \frac{2x-a-b}{4}\sqrt{(x-a)(b-x)}+ \frac{(b-a)^2}{4}\arcsin\sqrt{\frac{x-a}{b-x}}+C,(a<b) \quad (9.1)$$

含有三角函数的积分

1. $\int \sin xx = -\cos x + C$
2. $\int \cos xx = \sin x + C$
3. $\int \tan xx = -\ln|\cos x| + C$
4. $\int \cot xx = \ln|\sin x| + C$
5. $\int \sec xx = \ln\left|\tan\left(\frac{\pi}{4}+\frac{x}{2}\right)\right| + C = \ln|\sec x + \tan x| + C$
6. $\int \csc xx = \ln\left|\tan\frac{x}{2}\right| + C = \ln|\csc x - \cot x| + C$
7. $\int \sec^2 xx = \tan x + C$
8. $\int \csc^2 xx = -\cot x + C$
9. $\int \sec x \tan xx = \sec x + C$
10. $\int \csc x \cot xx = -\csc x + C$
11. $\int \sin^2 xx = \frac{x}{2} - \frac{1}{4}\sin 2x + C$
12. $\int \cos^2 xx = \frac{x}{2} + \frac{1}{4}\sin 2x + C$
13. $\int \sin^n xx = -\frac{1}{n}\sin^{n-1} x\cos x + \frac{n-1}{n}\int \sin^{n-2} xx$
14. $\int \cos^n xx = \frac{1}{n}\cos^{n-1} x\sin x + \frac{n-1}{n}\int \cos^{n-2} xx$
15. $\frac{x}{\sin^n x} = -\frac{1}{n-1}\frac{\cos x}{\sin^{n-1} x} + \frac{n-2}{n-1}\int \frac{x}{\sin^{n-2} x}$
16. $\frac{x}{\cos^n x} = \frac{1}{n-1}\frac{\sin x}{\cos^{n-1} x} + \frac{n-2}{n-1}\int \frac{x}{\cos^{n-2} x}$
17.

$$\begin{aligned}&\int \cos^m x \sin^n xx \\ =&\frac{1}{m+n}\cos^{m-1} x\sin^{n+1} x + \frac{m-1}{m+n}\int \cos^{m-2} x\sin^n xx \\ =&-\frac{1}{m+n}\cos^{m+1} x\sin^{n-1} x + \frac{n-1}{m+1}\int \cos^m x\sin^{n-2} xx\end{aligned}$$

18. $\int \sin ax \cos bx x = -\frac{1}{2(a+b)} \cos(a+b)x - \frac{1}{2(a-b)} \cos(a-b)x + C$
19. $\int \sin ax \sin bx x = -\frac{1}{2(a+b)} \sin(a+b)x + \frac{1}{2(a-b)} \sin(a-b)x + C$
20. $\int \cos ax \cos bx x = \frac{1}{2(a+b)} \sin(a+b)x + \frac{1}{2(a-b)} \sin(a-b)x + C$
21. $\int \frac{x}{a+b\sin x} = \begin{cases} \frac{2}{\sqrt{a^2-b^2}} \arctan \frac{a\tan\frac{x}{2}+b}{\sqrt{a^2-b^2}} + C & (a^2 > b^2) \\ \frac{1}{\sqrt{b^2-a^2}} \ln \left| \frac{a\tan\frac{x}{2}+b-\sqrt{b^2-a^2}}{a\tan\frac{x}{2}+b+\sqrt{b^2-a^2}} \right| + C & (a^2 < b^2) \end{cases}$
22. $\int \frac{x}{a+b\cos x} = \begin{cases} \frac{2}{a+b}\sqrt{\frac{a+b}{a-b}} \arctan \left(\sqrt{\frac{a-b}{a+b}} \tan\frac{x}{2} \right) + C & (a^2 > b^2) \\ \frac{1}{a+b}\sqrt{\frac{a+b}{a-b}} \ln \left| \frac{\tan\frac{x}{2}+\sqrt{\frac{a+b}{b-a}}}{\tan\frac{x}{2}-\sqrt{\frac{a+b}{b-a}}} \right| + C & (a^2 < b^2) \end{cases}$
23. $\int \frac{x}{a^2\cos^2 x + b^2 \sin^2 x} = \frac{1}{ab} \arctan \left(\frac{b}{a} \tan x \right) + C$
24. $\int \frac{x}{a^2\cos^2 x - b^2 \sin^2 x} = \frac{1}{2ab} \ln \left| \frac{b\tan x + a}{b\tan x - a} \right| + C$
25. $\int x \sin ax x = \frac{1}{a^2} \sin ax - \frac{1}{a} x \cos ax + C$
26. $\int x^2 \sin ax x = -\frac{1}{a} x^2 \cos ax + \frac{2}{a^2} x \sin ax + \frac{2}{a^3} \cos ax + C$
27. $\int x \cos ax x = \frac{1}{a^2} \cos ax + \frac{1}{a} x \sin ax + C$
28. $\int x^2 \cos ax x = \frac{1}{a} x^2 \sin ax + \frac{2}{a^2} x \cos ax - \frac{2}{a^3} \sin ax + C$

含有反三角函数的积分 **(**其中 $a > 0$**)**

1. $\int \arcsin \frac{x}{a} x = x \arcsin \frac{x}{a} + \sqrt{a^2 - x^2} + C$
2. $\int x \arcsin \frac{x}{a} x = (\frac{x^2}{2} - \frac{a^2}{4}) \arcsin \frac{x}{a} + \frac{x}{4}\sqrt{x^2 - x^2} + C$
3. $\int x^2 \arcsin \frac{x}{a} x = \frac{x^3}{3} \arcsin \frac{x}{a} + \frac{1}{9}(x^2 + 2a^2)\sqrt{a^2 - x^2} + C$
4. $\int \arccos \frac{x}{a} x = x\, arccos \frac{x}{a} - \sqrt{a^2 - x^2} + C$
5. $\int x \arccos \frac{x}{a} x = (\frac{x^2}{2} - \frac{a^2}{4}) \arccos \frac{x}{a} - \frac{x}{4}\sqrt{a^2 - x^2} + C$
6. $\int x^2 \arccos \frac{x}{a} x = \frac{x^3}{3} \arccos \frac{x}{a} - \frac{1}{9}(x^2 + 2a^2)\sqrt{a^2 - x^2} + C$
7. $\int \arctan \frac{x}{a} x = x \arctan \frac{x}{a} - \frac{a}{2} \ln(a^2 + x^2) + C$
8. $\int x \arctan \frac{x}{a} x = \frac{1}{2}(a^2 + x^2) \arctan \frac{x}{a} - \frac{a}{2} x + C$
9. $\int x^2 \arctan \frac{x}{a} x = \frac{x^3}{3} \arctan \frac{x}{a} - \frac{a}{6} x^2 + \frac{a^3}{6} \ln(a^2 + x^2) + C$

含有指数函数的积分

1. $\int a^x x = \frac{1}{\ln a} a^x + C$
2. $\int^{ax} x = \frac{1}{a} a^{ax} + C$
3. $\int x^{ax} x = \frac{1}{a^2}(ax - 1)a^{ax} + C$
4. $\int x^{nax} x = \frac{1}{a} x^{nax} - \frac{n}{a} \int x^{n-1ax} x$
5. $\int x a^x x = \frac{x}{\ln a} a^x - \frac{1}{(\ln a)^2} a^x + C$
6. $\int x^n a^x x = \frac{1}{\ln a} x^n a^x - \frac{n}{\ln a} \int x^{n-1} a^x x$
7. $\int^{ax} \sin bx x = \frac{1}{a^2+b^2}^{ax}(a \sin bx - b \cos bx) + C$
8. $\int^{ax} \cos bx x = \frac{1}{a^2+b^2}^{ax}(b \sin bx + a \cos bx) + C$
9. $\int^{ax} \sin^n bx x = \frac{1}{a^2+b^2n^2}^{ax} \sin^{n-1} bx(a \sin bx - nb \cos bx) + \frac{n(n-1)b^2}{a^2+b^2n^2} \int^{ax} \sin^{n-2} bx x$
10. $\int^{ax} \cos^n bx x = \frac{1}{a^2+b^2n^2}^{ax} \cos^{n-1} bx(a \cos bx + nb \sin bx) + \frac{n(n-1)b^2}{a^2+b^2n^2} \int^{ax} \cos^{n-2} bx x$

含有对数函数的积分

1. $\int \ln x x = x \ln x - x + C$
2. $\int \frac{x}{x \ln x} = \ln \left| \ln x \right| + C$
3. $\int x^n \ln x x = \frac{1}{n+1} x^{n+1}(\ln x - \frac{1}{n+1}) + C$
4. $\int (\ln x)^n x = x(\ln x)^n - n \int (\ln x)^{n-1} x$
5. $\int x^m (\ln x)^n x = \frac{1}{m+1} x^{m+1} (\ln x)^n - \frac{n}{m+1} \int x^m (\ln x)^{n-1} x$

9.7 Linear Programming

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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=-l;
    }
    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;
    }
};
```

9.8 Linear System

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
};
```

9.9 Matrix Inverse

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Matrix Inverse.hpp (0 bytes, 0 lines)

9.10 Matrix

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
}
```

9.11 Polynomial Exponential Function (Karatsuba Algorithm)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Polynomial Exponential Function (Karatsuba Algorithm).hpp (2297 bytes, 45 lines)

```
#ifndef EXPONENTIAL_FUNCTION_OF_POLYNOMIAL
#define EXPONENTIAL_FUNCTION_OF_POLYNOMIAL
#include<algorithm>
#include<cmath>
#include<vector>
namespace CTL{
    using namespace std;
    template<class T>struct ExponentialFunctionOfPolynomial{
        static 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),
19              *(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);}
        static void inv(vector<T>&a,int n,vector<T>&b,T**r){
            vector<T>c(n);b[0]=T(1)/a[0];fill(b.begin()+1,b.begin()+n,0);
            for(int i=1,m=2;(1<<i)<=n;++i,m<<=1){
                kar(&a[0],&b[0],m,i,r);
                for(int j=0;j<m;++j)c[j]=-r[i][j];c[0]+=T(2);
                kar(&b[0],&c[0],m,i,r);
                for(int j=0;j<m;++j)b[j]=r[i][j];}}
29      static void log(vector<T>&a,int n,vector<T>&b,T**r){
            fill(b.begin(),b.begin()+n,0);for(int i=1;i<n;++i)b[i-1]=a[i]*T
    (i);
            vector<T>c(n);inv(a,n,c,r);int l=round(log2(n));
            kar(&b[0],&c[0],n,l,r);for(int i=0;i<n;++i)b[i]=r[l][i];
            for(int i=n-2;i>=0;--i)b[i+1]=b[i]/T(i+1);b[0]=0;}
        static vector<T>run(vector<T>a){
            int tn,l=ceil(log2(tn=a.size())+1e-8),n=1<<l;a.resize(n);
            vector<T>b(n),c=b,d=c;b[0]=1;
            T**r=new T*[l+1];for(int i=0;i<=l;++i)r[i]=new T[(1<<i)*3];
            for(int i=1,m=2;i<=l;++i,m<<=1){
39              copy(b.begin(),b.begin()+m,d.begin());log(b,m,c,r);
                for(int j=0;j<m;++j)c[j]-=a[j];
                kar(&d[0],&c[0],m,i,r);
                for(int j=0;j<m;++j)b[j]-=r[i][j];}
            for(int i=0;i<=l;++i)delete r[i];delete r;
            return b.resize(tn),b;}};}
#endif
```

9.12 Polynomial Exponential Function (Number Theoretic Transform)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Polynomial Exponential Function (Number Theoretic Transform).hpp (5136 bytes, 80 lines)

```
#ifndef EXPONENTIAL_FUNCTION_OF_POLYNOMIAL
#define EXPONENTIAL_FUNCTION_OF_POLYNOMIAL
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    namespace ExponentialFunctionOfPolynomial{
        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;}}}
        void dco(vector<T>&a,vector<T>&b,int n,vector<T>&c,vector<T>&u1,
    vector<T>&u2,T p,T g){
            for(int i=0;i<n;++i)u1[i]=a[i];for(int i=n;i<2*n;++i)u1[i]=0;
            for(int i=0;i<n;++i)u2[i]=b[i];for(int i=n;i<2*n;++i)u2[i]=0;
            vector<int>rev(2*n);int l=round(log2(n));
            for(int i=0;i<2*n;++i)rev[i]=(rev[i>>1]>>1)|((i&1)<<l);
            ntt(u1,2*n,1,rev,p,g);ntt(u2,2*n,1,rev,p,g);
            for(int i=0;i<2*n;++i)u1[i]=u1[i]*u2[i]%p;ntt(u1,2*n,-1,rev,p,g
    );
            for(int i=0,t=pow(2*n,p-2,p);i<n;++i)c[i]=u1[i]*t%p;}
        struct big{big(){a[0]=1;for(int i=1;i<5;++i)a[i]=0;}T a[5];};
        void mul(big&b,T c){
            for(int i=0;i<5;++i)b.a[i]*=c;
            for(int i=0;i<5;++i)if(b.a[i]>=(1<<27)){b.a[i+1]+=(b.a[i]>>27);b
    .a[i]&=((1<<27)-1);}}
        void add(big&a,big&b){
            for(int i=0;i<5;++i){a.a[i]+=b.a[i];if(a.a[i]>=(1<<27))++a.a[i
    +1],a.a[i]&=((1<<27)-1);}}
        int cmp(big&a,big&b){
```

```
        for(int i=4;i>=0;--i){if(a.a[i]<b.a[i])return -1;if(a.a[i]>b.a
    [i])return 1;}return 0;}
    void div(big&a){for(int i=4;i>=0;--i){if((a.a[i]&1)&&i)a.a[i
    -1]+=(1<<27);a.a[i]>>=1;}}
    void mml(big&a,big&b,big&t){
        for(int i=0;i<5;++i)t.a[i]=0;
        for(int i=0;i<5;++i)for(int j=0;j<5;++j)t.a[i+j]+=a.a[i]*b.a[j];
        for(int i=0;i<5;++i)if(t.a[i]>=(1<<27)){
            if(i==4){for(int j=0;j<5;++j)t.a[j]=(1<<27)-1;return;}
            t.a[i+1]+=(t.a[i]>>27);t.a[i]&=((1<<27)-1);}}
    void mod(big&a,big&b){
        big l,r=a;int t=cmp(a,b);if(t==-1)return;if(t==0){for(int i=0;i
    <5;++i)a.a[i]=0;return;}
        while(1){
            big m=l;add(m,r);div(m);if(!cmp(m,l)||!cmp(m,r))break;
            big tm;mml(m,b,tm);if(cmp(tm,a)==-1)l=m;else r=m;}
        big tm;mml(l,b,tm);for(int i=0;i<5;++i)if((a.a[i]-=tm.a[i])<0)a
    .a[i]+=(1<<27),--a.a[i+1];}
    T cob(T c1,T c2,T c3,T p1,T p2,T p3,T p){
        big b1;mul(b1,pow(p2*p3%p1,p1-2,p1));mul(b1,c1);mul(b1,p2);mul(
    b1,p3);
        big b2;mul(b2,pow(p1*p3%p2,p2-2,p2));mul(b2,c2);mul(b2,p1);mul(
    b2,p3);
        big b3;mul(b3,pow(p1*p2%p3,p3-2,p3));mul(b3,c3);mul(b3,p1);mul(
    b3,p2);
        big b4;mul(b4,p1);mul(b4,p2);mul(b4,p3);add(b1,b2);add(b1,b3);
    mod(b1,b4);
        T u0=1,u1=(1<<27)%p,u2=(u1<<27)%p,u3=(u2<<27)%p,u4=(u3<<27)%p;
        return (u0*b1.a[0]+u1*b1.a[1]+u2*b1.a[2]+u3*b1.a[3]+u4*b1.a[4])%
    p;}
    void con(vector<T>&a,vector<T>&b,int n,vector<T>&c,vector<T>&u1,
    vector<T>&u2,T p,T g){
        if(g){dco(a,b,n,c,u1,u2,p,g);return;}
        T p1=15*(1<<27)+1,g1=31,p2=63*(1<<25)+1,g2=5,p3=127*(1<<24)+1,g3
    =3;
        vector<T>c1(n),c2(n),c3(n);dco(a,b,n,c1,u1,u2,p1,g1);
        dco(a,b,n,c2,u1,u2,p2,g2);dco(a,b,n,c3,u1,u2,p3,g3);
        for(int i=0;i<n;++i)c[i]=cob(c1[i],c2[i],c3[i],p1,p2,p3,p);}
    void inv(vector<T>&a,int n,vector<T>&b,vector<T>&u1,vector<T>&u2,T
    p,T g){
        vector<T>c(n),d(n);b[0]=pow(a[0],p-2,p);fill(b.begin()+1,b.
```

```
    begin()+n,0);
            for(int i=1,m=2;(1<<i)<=n;++i,m<<=1){
                con(a,b,m,c,u1,u2,p,g);
                for(int j=0;j<m;++j)if(c[j]>0)c[j]=p-c[j];
66              if((c[0]+=2)>=p)c[0]-=p;con(b,c,m,d,u1,u2,p,g);
               for(int i=0;i<n;++i)b[i]=d[i];}}
        void log(vector<T>&a,int n,vector<T>&b,vector<T>&u1,vector<T>&u2,T
    p,T g){
            fill(b.begin(),b.begin()+n,0);for(int i=1;i<n;++i)b[i-1]=a[i]*i
    %p;
            vector<T>c(n),d(n);inv(a,n,c,u1,u2,p,g);
            con(b,c,n,d,u1,u2,p,g);for(int i=0;i<n;++i)b[i]=d[i];
            for(int i=n-2;i>=0;--i)b[i+1]=b[i]*pow(i+1,p-2,p)%p;b[0]=0;}
        vector<T>run(vector<T>a,T p=15*(1<<27)+1,T g=31){
            int tn,l=ceil(log2(tn=a.size())+1e-8),n=1<<l;a.resize(n);
            vector<T>b(n),u1(2*n),u2(2*n),c=b,d=c;b[0]=1;
76          for(int i=1,m=2;i<=l;++i,m<<=1){
                log(b,m,c,u1,u2,p,g);for(int j=0;j<m;++j){c[j]-=a[j];if(c[j
    ]<0)c[j]+=p;}
                con(c,b,m,d,u1,u2,p,g);for(int j=0;j<m;++j){b[j]-=d[j];if(b
    [j]<0)b[j]+=p;}}
            return b.resize(tn),b;}}}
#endif
```

9.13 Polynomial Interpolation

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Polynomial Interpolation.hpp (372 bytes, 15 lines)

```
1   #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]);
```

```
11              }
            r+=p/q*y[i];
        }
        return r;
    }
```

CHAPTER 10

String Algorithms

10.1 Aho-Corasick Automaton

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
};
```

10.2 Factor Oracle

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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];
};
```

10.3 Longest Common Palindromic Substring

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Longest Common Palindromic Substring.hpp (1752 bytes, 41 lines)

```
#ifndef LONGEST_COMMON_PALINDROMIC_SUBSTRING
#define LONGEST_COMMON_PALINDROMIC_SUBSTRING
#include<bits/stdc++.h>
namespace CTL{
    using namespace std;
    struct LongestCommonPalindromicSubstring{
        struct node{
            node(int m,node*f,int l):nx(m),fa(f),ln(l){}
            vector<node*>nx;node*fa;complex<int>va;int ln;}*rt;
        int m;vector<int>st;vector<node*>ns;
        LongestCommonPalindromicSubstring(int _m):m(_m){
            node*n0=new node(m,0,-2),
                *n1=new node(m,n0,-1),*n2=new node(m,n1,0);
            ns.push_back(n0);ns.push_back(n1);ns.push_back(n2);
            fill(n0->nx.begin(),n0->nx.end(),n2);rt=n1;}
        ~LongestCommonPalindromicSubstring(){
            for(int i=0;i<ns.size();++i)delete ns[i];}
        node*find(node*x){
            while(x->fa&&st[st.size()-x->ln-2]!=st[st.size()-1])
                x=x->fa;
            return x;}
        node*insert(node*p,int c,complex<int>v){
            st.push_back(c);p=find(p);
            if(!p->nx[c]){
                node*np=(p->nx[c]=
                    new node(m,find(p->fa)->nx[c],p->ln+2));
                ns.push_back(np);}
            p->nx[c]->va+=v;
            return p->nx[c];}
        int run(int*a,int*b){
            node*p=rt;st=vector<int>(1,-1);
            for(int i=1;a[i]!=-1;++i)p=insert(p,a[i],1);
            p=rt;st=vector<int>(1,-1);
            for(int i=1;b[i]!=-1;++i)
                p=insert(p,b[i],complex<int>(0,1));
```

```
            for(int i=ns.size()-1;i>=1;--i)ns[i]->fa->va+=ns[i]->va;
            int r=0;for(int i=0;i<ns.size();++i)
                if(real(ns[i]->va)&&imag(ns[i]->va))
                    r=max(r,ns[i]->ln);
            return r;}};}
#endif
```

10.4 Longest Common Substring

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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
    ;
};
```

10.5 Palindromic Tree

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
    }
};
```

10.6 String Matching

Description

Find the occurrences of a pattern in a text using KMP algorithm. The prefix array is also provided.

Methods

template<class T>StringMatching<T>::StringMatching(T*p,int t=1);	
Description	construct an object of SuffixMatching for a given pattern
Parameters	**Description**
T	type of character
t	whether to optimize the prefix array, do not turn it on if you want to use the prefix array
p	pattern, indexed from one, ended by zero
Time complexity	$\Theta(\|p\|)$
Space complexity	$\Theta(\|p\|)$
Return value	an object of StringMatching

template<class T>int StringMatching<T>::run(T*t,int k=0);	
Description	given an occurence of the pattern in a text, find the next occurrence
Parameters	**Description**
t	text, indexed from one, ended by zero
k	start index of the last occurence of the pattern, use zero if there is none
Time complexity	$O(\|t\|)$
Space complexity	$\Theta(1)$
Return value	start index of the next occurence of the pattern

Fields

template<class T>vector<int>StringMatching<T>::f;	
Description	prefix array of KMP algorithm, indexed from one

Performance

Problem	Constraints	Time	Memory	Date
POJ 3461	$\|p\| = 10^4, \|t\| = 10^6$	141 ms	1340 kB	2016-02-14

References

Title	Author
Fast Pattern Matching in Strings	Donald E. Knuth, James H. Morris, Vaughan R. Pratt

Code

String Matching.hpp (686 bytes, 25 lines)

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

10.7 Suffix Array (DC3 Algorithm)

Description

Construct a suffix array and it's height array from a given string using DC3 algorithm.

Methods

template<class T,int M,T D>SuffixArray<T,M,D>::SuffixArray(T*s,int n);	
Description	construct an object of SuffixArray and in the mean time construct the suffix array and height array
Parameters	**Description**
T	type of character, usually char
M	size of alphabet
D	offset of alphabet, use 'a' for lowercase letters
s	string from which to build a suffix array, indexed from one
n	length of s
Time complexity	$\Theta(n+M)$
Space complexity	$\Theta(10n+M)$
Return value	an object of SuffixArray

Fields

template<class T,int M,T D>int*SuffixArray<T,M,D>::sa;	
Description	suffix array, indexed from one

template<class T,int M,T D>int*SuffixArray<T,M,D>::ht;	
Description	height array, indexed from one

Performance

Problem	**Constraints**	**Time**	**Memory**	**Date**
UOJ 35	$N = 10^5, M = 26$	416 ms (18+ cases)	4248 kB	2016-02-14

References

Title	**Author**
后缀数组——处理字符串的有力工具	罗穗骞

Code

Suffix Array (DC3 Algorithm).hpp (2656 bytes, 82 lines)

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

```
template<class T,int M,int D>struct SuffixArray{
    int*sa,*ht,*rk,*ts,*ct,*st;
    SuffixArray(T*s,int n){
        crt(st,n),crt(sa,n),crt(ht,n);
        crt(rk,n),crt(ts,n),crt(ct,max(n,M));
        for(int i=1;i<=n;++i)st[i]=s[i]-D+1;
        dc3(st,n,M,sa,rk);
        for(int 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
                &&st[i+d]==st[sa[rk[i]-1]+d];++d);
        }
    }
    ~SuffixArray(){
        del(sa),del(ht),del(rk);
        del(ts),del(ct),del(st);
    }
    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);
        for(r[a0[1]]=1,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];
        if(j=0,n%3==0)
            s0[++j]=n;
        for(i=1;i<=n0;++i)
            if(a0[i]>=k)
                a0[i]=(a0[i]-k)*3-1;
            else
                if((a0[i]=3*a0[i]-2)!=1)s0[++j]=a0[i]-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);
    }
};
```

10.8 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

template<class T,int M,T D>int SuffixArray<T,M,D>::sa[N+1];	
Description	suffix array, indexed from one

template<class T,int M,T D>int SuffixArray<T,M,D>::ht[N+1];	
Description	height array, indexed from one

Performance

Problem	Constraints	Time	Memory	Date
Tyvj 1860	$N = 2 \times 10^5, M = 26$	1154 ms (10 cases)	33012 kB	2016-02-14

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){
8       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){
18          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)\
28          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]);
38  }
    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;
            }
48          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);
58      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);
        }
68  }
    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];
};
```

10.9 Suffix Array (Prefix-Doubling Algorithm)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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);
 9      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];
19      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)
29              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]);
        }
39      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;
        }
49  }
    #undef lp
    #undef rp
    ~SuffixArray(){
        delete a;
    }
};
```

10.10 Suffix Array (SA-IS Algorithm)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Suffix Array (SA-IS Algorithm).hpp (0 bytes, 0 lines)

10.11 Suffix Array (Suffix Tree)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
```

```
};
```

10.12 Suffix Array (Treap)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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);
    }
};
```

10.13 Suffix Automaton

warning: old style will be replaced ... see Suffix Array (DC3) for new style

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;
};
```

10.14 Suffix Tree (Suffix Automaton)

Description

Use a suffix automaton to build a suffix tree. It has large data members, make its object static.

Methods

template<class T,int N,int M,T D>SuffixTree<T,N,M,D>::SuffixTree();	
Description	construct an object of SuffixTree
Parameters	**Description**
T	type of character, usually char
N	maximum length of string
M	size of alphabet
D	offset of alphabet, use 'a' for lowercase letters
Time complexity	$\Theta(1)$
Space complexity	$\Theta(8NM)$
Return value	an object of SuffixTree

template<class T,int N,int M,T D>void SuffixTree<T,N,M,D>::build(const T*s,int n);	
Description	build suffix tree for a given string
Parameters	**Description**
s	string from which to build a suffix tree, indexed from zero
n	length of s
Time complexity	$\Theta(nM)$
Space complexity	$\Theta(1)$
Return value	an object of SuffixTree

Fields

template<class T,int N,int M,T D>int SuffixTree<T,N,M,D>::nc;	
Description	number of nodes in suffix tree, they are labeled from one to nc, note that nc can be almost 2*$\|s\|$

template<class T,int N,int M,T D>int SuffixTree<T,N,M,D>::pr[2*N];	
Description	parent array of the suffix tree

template<class T,int N,int M,T D>int SuffixTree<T,N,M,D>::ch[2*N][M];	
Description	children array of the suffix tree

template<class T,int N,int M,T D>const T*SuffixTree<T,N,M,D>::el[2*N][M];	
Description	the start pointer of the string on children edge

template<class T,int N,int M,T D>const T*SuffixTree<T,N,M,D>::er[2*N][M];	
Description	the end pointer of the string on children edge, itself is not included

template<class T,int N,int M,T D>int SuffixTree<T,N,M,D>::tr[2*N][M];	
Description	$tr[u][i]$ is the node that represents $\{(D+i)+s \mid u \text{ represents } s\}$

template<class T,int N,int M,T D>int SuffixTree<T,N,M,D>::dp[2*N];	
Description	depth array of the suffix tree

template<class T,int N,int M,T D>int SuffixTree<T,N,M,D>::id[2*N];	
Description	$id[u]$ is the start of a postion where the strings u represents occur

template<class T,int N,int M,T D>int SuffixTree<T,N,M,D>::sf[2*N];	
Description	$sf[u]$ means whether u represents a suffix

References

Title	**Author**
后缀自动机	陈立杰

Code

Suffix Tree (Suffix Automaton).hpp (1010 bytes, 29 lines)

```
#include<cstring>
template<class T,int N,int M,T D>struct SuffixTree{
    int node(){
        pr[++nc]=dp[nc]=sf[nc]=0;
        memset(tr[nc],0,4*M);
        return nc;
    }
    void build(const T*s,int n){
        nc=0,node();
        for(int i=n-1,c,p=1,q,np,nq;i>=0;--i,p=np){
            dp[np=node()]=dp[p]+1,id[np]=i+1,sf[np]=1;
            for(c=s[i]-D;p&&!tr[p][c];p=pr[p])
                tr[p][c]=np;
            if(p&&dp[q=tr[p][c]]!=dp[p]+1){
                dp[nq=node()]=dp[p]+1,pr[nq]=pr[q],id[nq]=i+1;
                memcpy(tr[pr[q]=pr[np]=nq],tr[q],4*M);
                for(;p&&tr[p][c]==q;p=pr[p])
                    tr[p][c]=nq;
            }else
                pr[np]=p?q:1;
        }
        for(int i=2,j,c;i<=nc;++i)
            c=s[id[i]+dp[j=pr[i]]-1]-D,
            el[j][c]=s+id[i]+dp[j]-1,
            er[j][c]=s+id[i]+dp[ch[j][c]=i]-1;
    }
    const T*el[2*N][M],*er[2*N][M];
    int nc,pr[2*N],tr[2*N][M],dp[2*N],id[2*N],sf[2*N],ch[2*N][M];
};
```

10.15 Suffix Tree (Ukkonen's Algorithm)

warning: old style will be replaced ... see Suffix Array (DC3) for new style

Suffix Tree (Ukkonen's Algorithm).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;
            }
        }
91      edge*ae;
        int al,ct;
        node*root,*an,*pr;
    };
```