帅的板子

计算几何

二维凸包

```
#include <bits/stdc++.h>
using namespace std;
struct Point
{
    double x,y,ang;
    Point operator-(const Point& p)const {return {x-p.x,y-p.y,0};}
    double dis(Point p2)
    {
        return sqrt((x-p2.x)*(x-p2.x)+(y-p2.y)*(y-p2.y));
    }
    double cross(Point p2)
    {
        return x*p2.y-y*p2.x;
    }
};
struct baozi
{
    int n;
    vector<Point> v;
    vector<int> ans;
    baozi(int n):n(n),v(n){}
    void getans()
    {
        for(int i=1;i<n;++i)
        {
            if(v[i].y<v[0].y||(fabs(v[i].y-v[0].y)<1e-12&&v[i].x<v[0].x))
            {
                swap(v[0],v[i]);
            }
        }
        Point p=v[0];
        for(int i=1;i<n;++i) v[i].ang=atan2(v[i].y-p.y,v[i].x-p.x);
        sort(v.begin()+1,v.end(),[&](Point a,Point b)//以p为源点的极角排序
        {return fabs(a.ang-b.ang)<1e-12?p.dis(a)<p.dis(b):a.ang<b.ang;});
        ans.push_back(0);
        for(int i=1;i<n;++i)
        {
            while(1<(int)ans.size())
            {
                p=v[ans[(int)ans.size()-1]]-v[ans[(int)ans.size()-2]];
                if(p.cross(v[i]-v[ans[(int)ans.size()-1]])<0) ans.pop_back();
                else break;
            }
            ans.push_back(i);
        }
    }
};
```

```
int main()
{
    int n;cin>>n;baozi bao(n);
    for(int i=0;i<n;++i) cin>>bao.v[i].x>>bao.v[i].y;
    bao.getans();
    for(auto &it:bao.ans)//凸包点集
        cout<<bao.v[it].x<<' '<<bao.v[it].y<<'\n';
}
```

narea

```
#include <bits/stdc++.h>
using namespace std;
using ld=long double;
struct square
{
    vector<ld> x,y;
    square(int sz)
    {
        x.resize(sz+1,0);
        y.resize(sz+1,0);
    }
    ld getss(int sz)
    {
        ld res=0;
        for(int i=0;i<sz-1;++i) res+=x[i]*y[i+1]-x[i+1]*y[i];
        res+=x[sz-1]*y[0]-x[0]*y[sz-1];
        return fabs(res*0.5);
    }
};
int main()
{
    int n;cin>>n;square sq(n);
    for(int i=0;i<n;++i) cin>>sq.x[i]>>sq.y[i];
    cout<<fixed<<setprecision(2)<<sq.getss(n);
}
```

数据结构

并查集

```
#include <bits/stdc++.h>
using namespace std;
struct Dsu
{
    vector<int> fa,rk;
    Dsu(int siz)
    {
        fa.resize(siz+1);
        rk.resize(siz+1,1);
        iota(fa.begin(),fa.end(),0);
    }
    int find(int x){return x==fa[x]?x:(fa[x]=find(fa[x]));}
```

```
    void merge(int i,int j)
    {
        int x=find(i),y=find(j);
        if(x==y) return;
        if(rk[x]<=rk[y]) fa[x]=y;
        else fa[y]=x;
        if(rk[x]==rk[y]) rk[y]++;
    }
};
int main()
{
    int n,m,p;cin>>n>>m>>p;
    Dsu dsu(n);
    for(int i=0,x,y;i<m;++i) cin>>x>>y,dsu.merge(x,y);
    for(int i=0,x,y;i<p;++i)
    {
        cin>>x>>y;
        if(dsu.find(x)==dsu.find(y)) cout<<"One Set"<<'\n';
        else cout<<"NO"<<'\n';
    }
}
```

线段树

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
#define ls tr<<1
#define rs tr<<1|1
struct tree
{
    vector<ll> sum,add,mul;
    tree(int siz)
    {
        sum.resize(4*siz+5,0);mul.resize(4*siz+5,1);
        add.resize(4*siz+5,0);
    }
    void pushup(int tr) {sum[tr]=sum[ls]+sum[rs];}
    void build(int tr,int l,int r)
    {
        int mid=(l+r)>>1,x;
        if(l==r) return cin>>x,sum[tr]=x,void();
        build(ls,l,mid);
        build(rs,mid+1,r);
        pushup(tr);
    }
    void pushdown(int tr,int llen,int rlen)
    {
        if(mul[tr]!=1)
        {
            add[ls]=add[ls]*mul[tr];
            add[rs]=add[rs]*mul[tr];
            sum[ls]=sum[ls]*mul[tr];
            sum[rs]=sum[rs]*mul[tr];
            mul[ls]=mul[ls]*mul[tr];
```

```
            mul[rs]=mul[rs]*mul[tr];
            mul[tr]=1;
        }
        if(add[tr])
        {
            add[ls]=add[ls]+add[tr];
            add[rs]=add[rs]+add[tr];
            sum[ls]=sum[ls]+add[tr]*llen;
            sum[rs]=sum[rs]+add[tr]*rlen;
            add[tr]=0;
        }
    }
    void update(int flag,int tr,int x,int L,int R,int l,int r)
    {
        if(L>r||R<l) return;
        if(L<=l&&R>=r)
        {
            if(flag==2) add[tr]=add[tr]+x,sum[tr]=sum[tr]+x*(r-l+1);
            else mul[tr]=mul[tr]*x,sum[tr]=sum[tr]*x,add[tr]=add[tr]*x;
            return;
        }
        int mid=(l+r)>>1;
        pushdown(tr,mid-l+1,r-mid);
        if(R<=mid) update(flag,ls,x,L,R,l,mid);
        else if(L>mid) update(flag,rs,x,L,R,mid+1,r);
        else update(flag,ls,x,L,mid,l,mid),update(flag,rs,x,mid+1,R,mid+1,r);
        pushup(tr);
    }
    int query(int tr,int L,int R,int l,int r)
    {
        if(L>r||R<l) return 0;
        if(L<=l&&R>=r) return sum[tr];
        int mid=(l+r)>>1;
        pushdown(tr,mid-l+1,r-mid);
        return query(ls,L,R,l,mid)+query(rs,L,R,mid+1,r);
    }
};
int main()
{
    int n,q;cin>>n>>q;
    tree seg(n);seg.build(1,1,n);
    for(int i=0,op,x,y,k;i<q;++i)
    {
        cin>>op>>x>>y;
        if(op<3) cin>>k;
        if(op==3) cout<<seg.query(1,x,y,1,n)<<'\n';//区间和
        else seg.update(op,1,k,x,y,1,n);//1为乘k,2为加k
    }
}
```

树状数组

```
#include <bits/stdc++.h>
using namespace std;
struct node
{
    int s,id;
    node(int aa=0,int bb=0):s(aa),id(bb){}
};
struct Treearr
{
    int n;
    vector<int> a,tree;
    vector<node> b;
    Treearr(int siz)
    {
        a.resize(siz+1);tree.resize(siz+1,0);
        b.resize(siz+1);n=siz;
    }
    int lowbit(int x){return x&(-x);}
    void add(int x,int d=1){while(x<n+1) tree[x]+=d,x+=lowbit(x);}
    int query(int x,int res=0)
    {
        while(x) res+=tree[x],x-=lowbit(x);
        return res;
    }
    void qsort()
    {
        sort(b.begin(),b.end(),[](node x,node y){return x.s==y.s?
x.id<y.id:x.s<y.s;});
        int i=0;while(b[i].id==0) ++i;
        for(int j=i;j-i<n;++j) a[b[j].id]=j-i+1;
    }
};
int main()
{
    int n;cin>>n;Treearr tre(n);
    for(int i=1;i<=n;++i) cin>>tre.a[i],tre.b[i]=node(tre.a[i],i);
    tre.qsort();int ans=0;
    for(int i=1,t;i<=n;++i)
    {
        t=tre.query(n)-tre.query(tre.a[i]-1);
        ans+=t;tre.add(tre.a[i]);
    }
    cout<<ans<<'\n';//逆序对数量
}
```

权值线段树

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

```
    int tot;
    vector<int> val,cnt,lp,rp,sum;
    valtree(int n)
    {
        val.resize(4*n+5,0);cnt.resize(4*n+5,0);
        lp.resize(4*n+5,0);rp.resize(4*n+5,0);sum.resize(4*n+5,0);
        tot=1;
    }
    void push(int tr)
    {
        cnt[tr]=cnt[lp[tr]]+cnt[rp[tr]];
        sum[tr]=sum[lp[tr]]+sum[rp[tr]];
    }
    void update(int tr,int k,int x,int l=1,int r=2e9)
    {
        if(l>=r) return cnt[tr]+=k,val[tr]=x,sum[tr]+=k*x,void();
        int mid=(l+r)>>1;
        if(x<=mid) update(lp[tr]=lp[tr]?lp[tr]:++tot,k,x,l,mid);
        else update(rp[tr]=rp[tr]?rp[tr]:++tot,k,x,mid+1,r);
        push(tr);
    }
    int kth(int tr,int k,int l=1,int r=2e9)
    {
        if(l>=r) return val[tr];
        int mid=(l+r)>>1;
        if(cnt[lp[tr]=lp[tr]?lp[tr]:++tot]>=k) return kth(lp[tr],k,l,mid);
        else return kth(rp[tr]=rp[tr]?rp[tr]:++tot,k-cnt[lp[tr]],mid+1,r);
    }
    int ksum(int tr,int k,int l=1,int r=2e9)
    {
        if(l>=r) return k*val[tr];
        int mid=(l+r)>>1;
        if(cnt[lp[tr]=lp[tr]?lp[tr]:++tot]>=k) return ksum(lp[tr],k,l,mid);
        else return sum[lp[tr]]+ksum(rp[tr]=rp[tr]?rp[tr]:++tot,k-
cnt[lp[tr]],mid+1,r);
    }
};
int main()
{
    valtree tex(100000);
    tex.update(1,100,114);//插入114个100
    tex.update(1,350,200);
    tex.update(1,-50,114);
    cout<<tex.kth(1,50)<<'\n';//第50小数
    cout<<tex.ksum(1,50)<<'\n';//前50小数之和
    cout<<tex.kth(1,51)<<'\n';
    cout<<tex.ksum(1,51)<<'\n';
}
```

ST表

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

```
    vector<vector<int>> f;
    vector<int> Logn;
    int logn;
    st(int siz)
    {
        logn=1;while(logn<=siz) logn<<=1;
        f.resize(siz+1);Logn.resize(siz+1);
        Logn[1]=0;Logn[2]=1;Logn[0]=0;
        for(int i=3;i<=siz;++i) Logn[i]=Logn[i/2]+1;
        for(int i=0;i<=siz;++i) f[i].resize(logn+1,0);
        for(int i=1;i<=siz;++i) cin>>f[i][0];
        for(int j=1;j<=logn;++j)
        {
            for(int i=1;i+(1<<j)-1<=siz;++i)
            {
                f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
            }
        }
    }
    int query(int l,int r)
    {
        int s=Logn[r-l+1];
        return max(f[l][s],f[r-(1<<s)+1][s]);
    }
};
int main()
{
    int n,m;cin>>n>>m;st st(n);
    for(int i=1,x,y;i<=m;++i) cin>>x>>y,cout<<st.query(x,y)<<'\n';//[x,y]最大值
}
```

主席树

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=2e5+5,maxm=5e5+5;
int n,m,cnt,i,a[maxn],op[maxn][4],b[maxn];
int lower(int x)
{
    int l=1,r=cnt,t,mid;
    while(l<=r)
    {
        if(b[mid=(l+r)>>1]<=x) l=(t=mid)+1;
        else r=mid-1;
    }
    return t;
}
struct node
{
    int val,cnt,sum,p;
    node *l,*r;
    node(){val=sum=cnt=p=0;l=r=nullptr;}
    void up(){sum=l->sum+r->sum+cnt;}
}*blank=new(node),*T[maxm],pool[maxn*10],*cur;
void Rotatel(node *&x){node *y=x->r;x->r=y->l;x->up();y->l=x;y->up();x=y;}
```

```
void Rotater(node *&x){node *y=x->l;x->l=y->r;x->up();y->r=x;y->up();x=y;}
void Ins(node *&x,int y,int p)
{
    if(x==blank){x=cur++;x->val=y;x->l=x->r=blank;x->sum=x->cnt=1;x-
>p=rand();return;}
    x->sum+=p;
    if(y==x->val){x->cnt+=p;return;}
    if(y<x->val)
    {
        Ins(x->l,y,p);
        if(x->l->p>x->p) Rotater(x);
    }
    else
    {
        Ins(x->r,y,p);
        if(x->r->p>x->p) Rotatel(x);
    }
}
int Ask(node *x,int y)
{
    int t=0;
    while(x!=blank)
    {
        if(y<x->val) x=x->l;
        else t+=x->l->sum+x->cnt,x=x->r;
    }
    return t;
}
void add(int v,int i,int p)
{
    int a=1,b=cnt,mid,f=1,x=1;
    while(a<b)
    {
        if(f) Ins(T[x],i,p);
        mid=(a+b)>>1;x<<=1;
        if(v<=mid) f=1,b=mid;
        else f=0,a=mid+1,x|=1;
    }
    Ins(T[x],i,p);
}
int kth(int l,int r,int k)
{
    int x=1,a=1,b=cnt,mid;
    while(a<b)
    {
        mid=(a+b)>>1;x<<=1;
        int t=Ask(T[x],r)-Ask(T[x],l-1);
        if(k<=t) b=mid;
        else k-=t,a=mid+1,x|=1;
    }
    return a;
}
void build(int x,int a,int b)
{
    T[x]=blank;
    if(a==b) return;
```

```
    int mid=(a+b)>>1;
    build(x<<1,a,mid),build(x<<1|1,mid+1,b);
}
int main()
{
    blank->l=blank->r=blank;
    cin>>n;cur=pool;
    for(i=1;i<=n;++i) cin>>a[i],b[i]=a[i];
    cnt=n;cin>>m;
    for(i=1;i<=m;++i)
    {
        cin>>op[i][0]>>op[i][1]>>op[i][2];
        if(op[i][0]==1) b[++cnt]=op[i][2];
        else cin>>op[i][3];
    }
    sort(b+1,b+cnt+1);
    for(i=1;i<=n;++i) a[i]=lower(a[i]);
    for(i=1;i<=m;++i) if(op[i][0]==1) op[i][2]=lower(op[i][2]);
    build(1,1,cnt);
    for(i=1;i<=n;++i) add(a[i],i,1);
    for(i=1;i<=m;++i)
    {
        if(op[i][0]==1) //a[x]=y
        add(a[op[i][1]],op[i][1],-1),add(a[op[i][1]]=op[i][2],op[i][1],1);
        else //[x,y]第k小
        cout<<b[kth(op[i][1],op[i][2],op[i][3])]<<'\n';
    }
}
```

主席树第K小

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
struct tree
{
    vector<ll> sum;
    vector<int> lt,rt,root;
    int tot;
    tree(int sz)
    {
        sum.resize(32*sz+5);tot=0;
        lt.resize(32*sz+5);rt.resize(32*sz+5);
        root.resize(32*sz+5);
    }
    void pushup(int p){sum[p]=sum[lt[p]]+sum[rt[p]];}
    int build(int pl,int pr)
    {
        int tr=++tot;
        if(pl==pr) return cin>>sum[tr],tr;
        int mid=(pl+pr)>>1;
        lt[tr]=build(pl,mid);
        rt[tr]=build(mid+1,pr);
        pushup(tr);
        return tr;
```

```
    }
    int update(int pre,int pl,int pr,int pos,int k)
    {
        int tr=++tot;
        lt[tr]=lt[pre];rt[tr]=rt[pre];
        if(pl==pr) return sum[tr]=k,tr;
        int mid=(pl+pr)>>1;
        if(pos<=mid) lt[tr]=update(lt[tr],pl,mid,pos,k);
        else rt[tr]=update(rt[tr],mid+1,pr,pos,k);
        pushup(tr);
        return tr;
    }
    ll query(int p,int pl,int pr,int l,int r)
    {
        if(l<=pl&&r>=pr) return sum[p];
        int mid=(pl+pr)>>1;ll res=0;
        if(l<=mid) res+=query(lt[p],pl,mid,l,r);
        if(r>mid) res+=query(rt[p],mid+1,pr,l,r);
        return res;
    }
};
int main()
{
    int n,m;cin>>n>>m;
    tree tre(n);
    tre.root[0]=tre.build(1,n);
    for(int i=1,op,x,y;i<=m;++i)
    {
        cin>>op>>x;
        if(op) cin>>y;
        if(!op) tre.root[i]=tre.root[x];//回溯到第x次操作后
        else if(op==1) tre.root[i]=tre.update(tre.root[i-1],1,n,x,y);//a[x]=y
        else cout<<tre.query(tre.root[i]=tre.root[i-1],1,n,x,y)<<'\n';//a[x]-a[y]
和
    }
}
```

莫队

```
#include <bits/stdc++.h>
using namespace std;
struct query
{
    int l,r,id;
    query(int a=0,int b=0,int c=0):l(a),r(b),id(c){}
};
struct Muline
{
    vector<int> aa,cnt,belong;
    int now,siz,bnum;
    Muline(int sz)
    {
        siz=sqrt(sz+10);bnum=ceil((double)(sz+10)/siz);now=0;
        aa.resize(sz+10);cnt.resize(sz+10);belong.resize(sz+10);
        for(int i=1;i<=bnum;++i)
```

```
        {
            for(int j=(i-1)*siz+1;j<=i*siz;++j)
            {
                belong[j]=i;
            }
        }
    }
    void getans(int m,vector<query>& q)
    {
        vector<int> ans(m);
        sort(q.begin(),q.end(),[&](query a,query b)
        {return (belong[a.l]^belong[b.l])?belong[a.l]<belong[b.l]:
((belong[a.l]&1)?a.r<b.r:a.r>b.r);});
        int l=1,r=0;
        for(int i=0;i<m;++i)
        {
            int ql=q[i].l,qr=q[i].r;
            while(l<ql) now-=!--cnt[aa[l++]];
            while(l>ql) now+=!cnt[aa[--l]]++;
            while(r<qr) now+=!cnt[aa[++r]]++;
            while(r>qr) now-=!--cnt[aa[r--]];
            ans[q[i].id]=now;
        }
        for(int i=0;i<m;++i) cout<<ans[i]<<' ';//[L,R]有几个不同元素
    }
};
int main()
{
    int n;cin>>n;Muline mule(n);
    for(int i=1;i<=n;++i) cin>>mule.aa[i];
    int m;cin>>m;vector<query> q(m);
    for(int i=0;i<m;++i) cin>>q[i].l>>q[i].r,q[i].id=i;
    mule.getans(m,q);
}
```

珂朵莉树

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
int ksm(ll a,ll n,ll p)
{
    ll res=1;a%=p;
    while(n)
    {
        if(n&1) res=res*a%p;
        n>>=1;
        a=a*a%p;
    }
    return res;
}
struct node
{
    ll l,r;
    mutable ll v;
```

```
    node(ll a,ll b,ll c):l(a),r(b),v(c){;}
    bool operator<(const node &ope) const{return l<ope.l;}
};
struct odt
{
    set<node> tree;
    set<node>::iterator split(ll pos)
    {
        auto it=tree.lower_bound(node(pos,0,0));
        if(it!=tree.end()&&it->l==pos) return it;
        --it;
        ll l=it->l,r=it->r,v=it->v;
        tree.erase(it);
        tree.insert(node(l,pos-1,v));
        return tree.insert(node(pos,r,v)).first;
    }
    void assign(ll l,ll r,ll v)
    {
        auto ed=split(r+1),bn=split(l);
        tree.erase(bn,ed);
        tree.insert(node(l,r,v));
    }
    void add(ll l,ll r,ll v)
    {
        auto ed=split(r+1);
        for(auto it=split(l);it!=ed;++it) it->v+=v;
    }
    ll kth(ll l,ll r,ll k)
    {
        auto ed=split(r+1);
        vector<pair<ll,ll>> ve;
        for(auto it=split(l);it!=ed;++it) ve.push_back({it->v,it->r-it->l+1});
        sort(ve.begin(),ve.end());
        for(auto &p:ve)
        {
            k-=p.second;
            if(k<=0) return p.first;
        }
        return -1;
    }
    ll sop(ll l,ll r,ll x,ll y)
    {
        ll res=0;
        auto ed=split(r+1);
        for(auto it=split(l);it!=ed;++it)
        res=(res+ksm(it->v,x,y)*(it->r-it->l+1))%y;
        return res;
    }
};
int main()
{
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    auto solve=[&]()->void
    {
        const static int mod=1e9+7;
        int n,m,vmax;ll seed;
```

```
        cin>>n>>m>>seed>>vmax;
        auto rnd=[&]()
        {
            int res=seed;
            seed=(seed*7+13)%mod;
            return res;
        };
        odt test;
        for(int i=1,op;i<=n;++i)
        {
            op=rnd();
            test.tree.insert(node(i,i,op%vmax+1));
        }
        for(int i=1,op,ll,rr,x,y;i<=m;++i)
        {
            op=rnd()%4+1;ll=rnd()%n+1;rr=rnd()%n+1;
            if(ll>rr) swap(ll,rr);
            if(op==3) x=rnd()%(rr-ll+1)+1;
            else x=rnd()%vmax+1;
            if(op==4) y=rnd()%vmax+1;
            if(op==1) test.add(ll,rr,x);//区间加x
            else if(op==2) test.assign(ll,rr,x);//区间赋值x
            else if(op==3) cout<<test.kth(ll,rr,x)<<'\n';//区间第x小
            else cout<<test.sop(ll,rr,x,y)<<'\n';//区间x次幂和模y

        }
    };
    solve();
}
```

区间最小值

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
#define ls tr<<1
#define rs tr<<1|1
struct tree
{
    vector<ll> sum,add;
    tree(int siz)
    {
        sum.resize(4*siz+5,-9e18);add.resize(4*siz+5,0);
    }
    void pushup(int tr){sum[tr]=min(sum[ls],sum[rs]);}
    void build(int tr,int l,int r)
    {
        int mid=(l+r)>>1;ll x;
        if(l==r) return cin>>x,sum[tr]=x,void();
        build(ls,l,mid);
        build(rs,mid+1,r);
        pushup(tr);
    }
    void pushdown(int tr)
    {
```

```
        if(add[tr])
        {
            add[ls]=add[ls]+add[tr];
            add[rs]=add[rs]+add[tr];
            sum[ls]=sum[ls]+add[tr];
            sum[rs]=sum[rs]+add[tr];
            add[tr]=0;
        }
    }
    void update(int tr,ll x,int L,int R,int l,int r)
    {
        if(L>r||R<l) return;
        if(L<=l&&R>=r) return add[tr]=(add[tr]+x),sum[tr]=(sum[tr]+x),void();
        int mid=(l+r)>>1;
        pushdown(tr);
        if(R<=mid) update(ls,x,L,R,l,mid);
        else if(L>mid) update(rs,x,L,R,mid+1,r);
        else update(ls,x,L,mid,l,mid),update(rs,x,mid+1,R,mid+1,r);
        pushup(tr);
    }
    ll minm(int tr,int L,int R,int l,int r)
    {
        ll res=9e18;
        if(L>r||R<l) return res;
        if(L<=l&&R>=r) return sum[tr];
        int mid=(l+r)>>1;
        pushdown(tr);
        if(R<=mid) res=min(res,minm(ls,L,R,l,mid));
        else if(L>mid) res=min(res,minm(rs,L,R,mid+1,r));
        else res=min(res,min(minm(ls,L,R,l,mid),minm(rs,L,R,mid+1,r)));
        return res;
    }
};
int main()
{
    int n,q;cin>>n>>q;
    tree tre(n);tre.build(1,1,n);
    for(int i=0,l,r,k,x;i<q;++i)
    {
        cin>>x;
        if(x==1) cin>>l>>r>>k,tre.update(1,k,l,r,1,n);//区间+k
        else cin>>l>>r,cout<<tre.minm(1,l,r,1,n)<<'\n';//区间最值
    }
}
```

数学

中国剩余定理

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
ll exgcd(ll a,ll b,ll &x,ll &y)
{
```

```
    if(!b){x=1;y=0;return a;}
    ll d=exgcd(b,a%b,y,x);
    y-=(a/b)*x;
    return d;
}
ll excrt(vector<ll>&r,vector<ll>&m)
{
    //m是模数
    ll x=r[0],y1,y2,LCM=m[0];
    for(int i=1;i<(int)r.size();++i)
    {
        ll b=m[i],c=((r[i]-x)%b+b)%b;
        ll GCD=exgcd(LCM,b,y1,y2);
        if(c%GCD) return -1;
        y1=y1*(c/GCD)%(b/GCD);
        x+=LCM*y1;
        LCM*=b/GCD;
        x=(x%LCM+LCM)%LCM;
    }
    return x;
}
int main()
{
    int n;cin>>n;
    vector<ll> a(n),b(n);
    for(int i=0;i<n;++i) cin>>a[i]>>b[i];
    cout<<excrt(b,a)<<'\n';
}
```

扩展欧几里得

```
#include <bits/stdc++.h>
using namespace std;
int exgcd(int a,int b,int &x,int &y)
{
    if(!b) return x=1,y=0,a;
    int r=exgcd(b,a%b,y,x);
    y-=(a/b)*x;
    return r;
}
int main()
{
    int a,b;cin>>a>>b;
    int x0,y0;
    int gd=exgcd(a,b,x0,y0);
    //ax+by=gd(c)
    cout<<x0<<' '<<y0<<' '<<gd<<'\n';
    //x=x0*c/gcd(a,b)+k*b/gcd(a,b)
    //y=y0*c/gcd(a,b)-k*a/gcd(a,b)
}
```

FFT

```
#include <bits/stdc++.h>
using namespace std;
const double PI=acos(-1.0);
struct Complex
{
    double x,y;
    Complex(double _x=0.0,double _y=0.0):x(_x),y(_y){}
    Complex operator-(const Complex &b) const {return Complex(x-b.x,y-b.y);}
    Complex operator+(const Complex &b) const {return Complex(x+b.x,y+b.y);}
    Complex operator*(const Complex &b) const {return Complex(x*b.x-
y*b.y,x*b.y+y*b.x);}
};
const int maxn=200020;
char str1[maxn/2],str2[maxn/2];
int sum[maxn],len1,len2;
struct FFT
{
    vector<Complex> x1,x2;
    int len;
    FFT(int siz)
    {
        x1.resize(siz);x2.resize(siz);
        len=siz;
    }
    void init()
    {
        for(int i=0;i<len1;++i) x1[i]=Complex(str1[len1-1-i]-'0',0);
        for(int i=len1;i<len;++i) x1[i]=Complex(0,0);
        for(int i=0;i<len2;++i) x2[i]=Complex(str2[len2-1-i]-'0',0);
        for(int i=len2;i<len;++i) x2[i]=Complex(0,0);
    }
    void change(vector<Complex> &y)
    {
        int i,j,k;
        for(i=1,j=len/2;i<len-1;++i)
        {
            if(i<j) swap(y[i],y[j]);
            k=len/2;
            while(j>=k){j-=k;k/=2;}
            if(j<k) j+=k;
        }
    }
    void fft(vector<Complex> &y,int on)//on==1时是DFT,on==-1时是IDFT
    {
        change(y);
        for(int h=2;h<=len;h<<=1)
        {
            Complex wn(cos(2*PI/h),sin(on*2*PI/h));
            for(int j=0;j<len;j+=h)
            {
                Complex w(1,0);
                for(int k=j;k<j+h/2;++k)
                {
```

```
                    Complex u=y[k],t=w*y[k+h/2];
                    y[k]=u+t;y[k+h/2]=u-t;w=w*wn;
                }
            }
        }
        if(on==-1) for(int i=0;i<len;++i) y[i].x/=len;
    }
    void getans()
    {
        fft(x1,1);fft(x2,1);
        for(int i=0;i<len;++i) x1[i]=x1[i]*x2[i];
        fft(x1,-1);
        for(int i=0;i<len;++i) sum[i]=int(x1[i].x+0.5);
        for(int i=0;i<len;++i)
        {
            sum[i+1]+=sum[i]/10;
            sum[i]%=10;
        }
        len=len1+len2-1;
        while(sum[len]==0&&len>0) --len;
        for(int i=len;i>=0;--i)
        {
            char t=sum[i]+'0';
            cout<<t;
        }
    }
};
int main()
{
    cin>>str1>>str2;len1=strlen(str1),len2=strlen(str2);int len=1;
    while(len<len1*2||len<len2*2) len<<=1;
    FFT fg(len);fg.init();fg.getans();
    cout<<'\n';
}
```

逆元

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
int main()
{
    int n,p;cin>>n>>p;
    vector<ll> inv(n+1);inv[1]=1;
    for(int i=2;i<=n;i++) inv[i]=p-(p/i*inv[p%i]%p)%p;//1~n的模p逆元
    //如果inv[i]表示i!的逆元
    //for(int i=n-1;i>=0;--i) inv[i]=inv[i+1]*(i+1)%p;
}
```

线性基

```
#include <bits/stdc++.h>
using ull =unsigned long long;
using namespace std;
struct Linechick
{
    vector<ull> p;
    int rank;
    Linechick(){p.resize(64,0);rank=0;}
    void insert(ull x)
    {
        for(int i=63;~i;--i)
        {
            if(!(x>>i)) continue;
            if(!p[i]) {p[i]=x;++rank;break;}
            x^=p[i];
        }
    }
    ull getmax(ull ans=0)
    {
        for(int i=63;~i;--i) ans=max(ans,ans^p[i]);
        return ans;
    }
    int check(ull x,int tot=0)
    {
        for(int i=63;~i;--i)
        {
            if(x>>i&1ll) x^=p[i];
            if(p[i]) ++tot;
        }
        return x?-1:tot;
    }
};
int main()
{
    int n,m;cin>>n;ull a;Linechick lck;m=n;
    while(n--) cin>>a,lck.insert(a);
    cout<<lck.getmax()<<'\n';
    cin>>n;
    while(n--)
    {
        cin>>a;int t=lck.check(a);
        if(t==-1) cout<<-1<<'\n';
        else cout<<((ull)1<<(m-t))<<'\n';//构造方案数
    }
}
```

卢卡斯定理

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
template<const int T>
```

```
struct ModInt
{
    const static int mod=T;
    int x;
    ModInt(int x=0):x(x%mod){}
    ModInt(long long x):x(int(x%mod)){}
    int val() {return x;}
    ModInt operator + (const ModInt &a) const {int x0=x+a.x;return ModInt(x0<mod?
x0:x0-mod);}
    ModInt operator - (const ModInt &a) const {int x0=x-a.x;return ModInt(x0<0?
x0+mod:x0);}
    ModInt operator * (const ModInt &a) const {return ModInt(1ll*x*a.x%mod);}
    ModInt operator / (const ModInt &a) const {return *this*a.inv();}
    bool operator == (const ModInt &a) const {return x==a.x;};
    bool operator != (const ModInt &a) const {return x!=a.x;};
    void operator += (const ModInt &a) {x+=a.x;if(x>=mod) x-=mod;}
    void operator -= (const ModInt &a) {x-=a.x;if(x<0) x+=mod;}
    void operator *= (const ModInt &a) {x=1ll*x*a.x%mod;}
    void operator /= (const ModInt &a) {*this = *this / a; }
    friend ModInt operator + (int y,const ModInt &a){int x0=y+a.x; return
ModInt(x0<mod?x0:x0-mod);}
    friend ModInt operator - (int y,const ModInt &a){int x0=y-a.x; return
ModInt(x0<0?x0+mod:x0);}
    friend ModInt operator * (int y,const ModInt &a){return
ModInt(1ll*y*a.x%mod);}
    friend ModInt operator / (int y,const ModInt &a){return ModInt(y)/a;}
    friend ostream &operator<<(ostream &os,const ModInt &a) {return os<<a.x;}
    friend istream &operator>>(istream &is,ModInt &t){return is>>t.x;}
    ModInt pow(int64_t n) const
    {
        ModInt res(1),mul(x);
        while(n)
        {
            if(n&1) res*=mul;
            mul*=mul;
            n>>=1;
        }
        return res;
    }
    ModInt inv() const
    {
        int a=x,b=mod,u=1,v=0;
        while(b)
        {
            int t=a/b;
            a-=t*b;swap(a,b);
            u-=t*v;swap(u,v);
        }
        if(u<0) u+=mod;
        return u;
    }
};
using mint=ModInt<998244353>;
struct Lucas
{
    vector<mint> fac;
```

```
    Lucas(int siz)
    {
        fac.resize(siz+1);
        fac[0]=fac[1]=1;
        for(int i=2;i<=siz;++i) fac[i]=fac[i-1]*i;
    }
    mint qpow(mint b,ll n)
    {
        mint res=1;
        while(n)
        {
            if(n&1) res=res*b;
            b=b*b;n>>=1;
        }
        return res;
    }
    mint C(ll n,ll m,int mod=998244353)
    {
        return m>n?0:fac[n]*qpow(fac[m]*fac[n-m],mod-2);
    }
    mint lucas(ll n,ll m,int mod=998244353,mint ans=1)
    {
        while(n&&m&&(ans!=0))
        {
            ans=ans*C(n%mod,m%mod);
            n/=mod;m/=mod;
        }
        return ans;
    }
};
int main()
{
    ll n;cin>>n;Lucas luc(n);
    cin>>n;
    while(n--)
    {
        ll x,y;cin>>x>>y;
        cout<<luc.lucas(x,y)<<'\n';
    }
}
```

矩阵快速幂

```
#include <bits/stdc++.h>
using namespace std;
template<const int T>
struct ModInt
{
    const static int mod=T;
    int x;
    ModInt(int x = 0):x(x%mod){}
    ModInt(long long x):x(int(x%mod)){}
    int val() {return x;}
    ModInt operator + (const ModInt &a) const {int x0=x+a.x;return ModInt(x0<mod?
x0:x0-mod);}
```

```
    ModInt operator - (const ModInt &a) const {int x0=x-a.x;return ModInt(x0<0?
x0+mod:x0);}
    ModInt operator * (const ModInt &a) const {return ModInt(1ll * x * a.x %
mod); }
    ModInt operator / (const ModInt &a) const {return *this*a.inv();}
    bool operator == (const ModInt &a) const {return x==a.x;};
    bool operator != (const ModInt &a) const {return x!=a.x;};
    void operator += (const ModInt &a) {x+=a.x;if(x>=mod) x-=mod;}
    void operator -= (const ModInt &a) {x-=a.x;if(x<0) x+=mod;}
    void operator *= (const ModInt &a) {x=1ll*x*a.x%mod;}
    void operator /= (const ModInt &a) {*this = *this / a; }
    friend ModInt operator + (int y,const ModInt &a){int x0=y+a.x; return
ModInt(x0<mod?x0:x0-mod);}
    friend ModInt operator - (int y,const ModInt &a){int x0=y-a.x; return
ModInt(x0<0?x0+mod:x0);}
    friend ModInt operator * (int y,const ModInt &a){return
ModInt(1ll*y*a.x%mod);}
    friend ModInt operator / (int y,const ModInt &a){return ModInt(y)/a;}
    friend ostream &operator<<(ostream &os,const ModInt &a) {return os<<a.x;}
    friend istream &operator>>(istream &is,ModInt &t){return is>>t.x;}
    ModInt pow(int64_t n) const
    {
        ModInt res(1),mul(x);
        while(n)
        {
            if(n&1) res*=mul;
            mul*=mul;
            n>>=1;
        }
        return res;
    }
    ModInt inv() const
    {
        int a=x,b=mod,u=1,v=0;
        while(b)
        {
            int t=a/b;
            a-=t*b;swap(a,b);
            u-=t*v;swap(u,v);
        }
        if(u<0) u+=mod;
        return u;
    }
};
using mint=ModInt<998244353>;
struct Mat
{
    vector<vector<mint>> mat;
    int n;
    Mat(int nt)
    {
        n=nt;mat.resize(nt);
        for(int i=0;i<nt;++i) mat[i].resize(nt,0);
    }
    void init(){for(int i=0;i<n;++i) mat[i][i]=1;}
    Mat operator*(Mat b)
```

```
    {
        Mat c(n);
        for(int i=0;i<n;++i)
        {
            for(int j=0;j<n;++j)
            {
                c.mat[i][j]=0;
                for(int k=0;k<n;++k)
                {
                    c.mat[i][j]=c.mat[i][j]+mat[i][k]*b.mat[k][j];
                }
            }
        }
        return c;
    }
    Mat operator+(Mat b)
    {
        Mat c(n);
        for(int i=0;i<n;++i)
        {
            for(int j=0;j<n;++j)
            {
                c.mat[i][j]=mat[i][j]+b.mat[i][j];
            }
        }
        return c;
    }
    Mat quickpow(int m)
    {
        Mat res1(n),a(n);
        res1.init();a.mat=mat;
        while(m)
        {
            if(m&1) res1=res1*a;
            a=a*a;m>>=1;
        }
        return res1;
    }
};
int main()
{
    int n;cin>>n;Mat p(2);
    p.mat[0][0]=p.mat[1][0]=p.mat[0][1]=1;p.mat[1][1]=0;
    for(int i=0,x;i<n;++i)
    {
        cin>>x;Mat q=p.quickpow(x);
        cout<<q.mat[0][0]<<'\n';//乘上x个(1,1\n1,0)的结果,也是斐波那契第x+1项的值
    }
}
```

莫反

```
#include <bits/stdc++.h>
using namespace std;
struct Mofan
{
    vector<int> mu,prime;
    vector<bool> vis;
    Mofan(int sz)
    {
        mu.resize(sz+1,0);vis.resize(sz+1,0);
        vis[1]=1;mu[1]=1;
        for(int i=2;i<=sz;++i)
        {
            if(!vis[i]) mu[i]=-1,prime.push_back(i);
            for(int j=0;j<(int)prime.size()&&i*prime[j]<=sz;++j)
            {
                mu[i*prime[j]]=-mu[i];
                vis[i*prime[j]]=1;
                if(i%prime[j]==0)
                {
                    mu[i*prime[j]]=0;
                    break;
                }
            }
        }
    }
};
int main()
{
    int n;cin>>n;Mofan mof(n);
}
```

Miller_Rabin

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
struct Rhpr
{
    ll max_factor,n;
    Rhpr(ll m){n=m;max_factor=0;}
    ll quick_pow(ll x,ll p,ll mod,ll ans=1)
    {
        while(p)
        {
            if(p&1) ans=(__int128)ans*x%mod;
            x=(__int128)x*x%mod;
            p>>=1;
        }
        return ans;
    }
    bool Miller_Rabin(ll p)
```

```
    {
        if(p<2) return 0;
        if(p==2) return 1;
        if(p==3) return 1;
        ll d=p-1,r=0;
        while(!(d&1)) ++r,d>>=1;
        for(ll k=0;k<10;++k)
        {
            ll a=rand()%(p-2)+2;
            ll x=quick_pow(a,d,p);
            if(x==1||x==p-1) continue;
            for(int i=0;i<r-1;++i)
            {
                x=(__int128) x*x%p;
                if(x==p-1) break;
            }
            if(x!=p-1) return 0;
        }
        return 1;
    }
    ll Pollard_Rho(ll x)
    {
        ll s=0,t=0,val=1;
        ll c=(ll)rand()%(x-1)+1;
        int step=0,goal=1;
        for(goal=1;;goal*=2,s=t,val=1)
        {
            for(step=1;step<=goal;++step)
            {
                t=((__int128) t*t+c)%x;
                val=(__int128) val*abs(t-s)%x;
                if((step%127)==0)
                {
                    ll d=__gcd(val,x);
                    if(d>1) return d;
                }
            }
            ll d=__gcd(val,x);
            if(d>1) return d;
        }
    }
    void fac(ll x)
    {
        if(x<=max_factor||x<2) return;
        if(Miller_Rabin(x)) return max_factor=max(max_factor,x),void();
        ll p=x;
        while(p>=x) p=Pollard_Rho(x);
        while((x%p)==0) x/=p;
        fac(x),fac(p);
    }
};
int main()
{
    int t;cin>>t;
    while(t--)
    {
```

```
        srand((unsigned)time(NULL));
        ll n;cin>>n;Rhpr rhp(n);rhp.fac(n);
        if(rhp.max_factor==n) printf("Prime\n");
        else cout<<rhp.max_factor<<'\n';
    }
}
```

斯特林数

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
using ld=long double;
template<const int T>
struct ModInt
{
    const static int mod=T;
    int x;
    ModInt(int x=0):x(x%mod){}
    ModInt(long long x):x(int(x%mod)){}
    int val() {return x;}
    ModInt operator + (const ModInt &a) const {int x0=x+a.x;return ModInt(x0<mod?
x0:x0-mod);}
    ModInt operator - (const ModInt &a) const {int x0=x-a.x;return ModInt(x0<0?
x0+mod:x0);}
    ModInt operator * (const ModInt &a) const {return ModInt(1ll*x*a.x%mod);}
    ModInt operator / (const ModInt &a) const {return *this*a.inv();}
    bool operator == (const ModInt &a) const {return x==a.x;};
    bool operator != (const ModInt &a) const {return x!=a.x;};
    void operator += (const ModInt &a) {x+=a.x;if(x>=mod) x-=mod;}
    void operator -= (const ModInt &a) {x-=a.x;if(x<0) x+=mod;}
    void operator *= (const ModInt &a) {x=1ll*x*a.x%mod;}
    void operator /= (const ModInt &a) {*this = *this / a; }
    friend ModInt operator + (int y,const ModInt &a){int x0=y+a.x; return
ModInt(x0<mod?x0:x0-mod);}
    friend ModInt operator - (int y,const ModInt &a){int x0=y-a.x; return
ModInt(x0<0?x0+mod:x0);}
    friend ModInt operator * (int y,const ModInt &a){return
ModInt(1ll*y*a.x%mod);}
    friend ModInt operator / (int y,const ModInt &a){return ModInt(y)/a;}
    friend ostream &operator<<(ostream &os,const ModInt &a) {return os<<a.x;}
    friend istream &operator>>(istream &is,ModInt &t){return is>>t.x;}
    ModInt pow(int64_t n) const
    {
        ModInt res(1),mul(x);
        while(n)
        {
            if(n&1) res*=mul;
            mul*=mul;
            n>>=1;
        }
        return res;
    }
    ModInt inv() const
    {
```

```
        int a=x,b=mod,u=1,v=0;
        while(b)
        {
            int t=a/b;
            a-=t*b;swap(a,b);
            u-=t*v;swap(u,v);
        }
        if(u<0) u+=mod;
        return u;
    }
};
using mint=ModInt<998244353>;
struct Strling
{
    vector<vector<mint>> f;
    Strling(int siz)
    {
        f.resize(siz+1);
        for(int i=0;i<=siz;++i) f[i].resize(siz+1);
        f[0][0]=1;
        for(int i=1;i<=siz;++i)
        {
            for(int j=1;j<=siz;++j)
            {
                f[i][j]=j*f[i-1][j]+f[i-1][j-1];
            }
        }
    }
};
int main()
{
    Strling strl(5000);
    cout<<strl.f[56][52]<<'\n';
}
```

杨辉三角

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
template<const int T>
struct ModInt
{
    const static int mod=T;
    int x;
    ModInt(int x = 0):x(x%mod){}
    ModInt(long long x):x(int(x%mod)){}
    int val() {return x;}
    ModInt operator + (const ModInt &a) const {int x0=x+a.x;return ModInt(x0<mod?
x0:x0-mod);}
    ModInt operator - (const ModInt &a) const {int x0=x-a.x;return ModInt(x0<0?
x0+mod:x0);}
    ModInt operator * (const ModInt &a) const {return ModInt(1ll * x * a.x %
mod); }
    ModInt operator / (const ModInt &a) const {return *this*a.inv();}
```

```
    bool operator == (const ModInt &a) const {return x==a.x;};
    bool operator != (const ModInt &a) const {return x!=a.x;};
    void operator += (const ModInt &a) {x+=a.x;if(x>=mod) x-=mod;}
    void operator -= (const ModInt &a) {x-=a.x;if(x<0) x+=mod;}
    void operator *= (const ModInt &a) {x=1ll*x*a.x%mod;}
    void operator /= (const ModInt &a) {*this = *this / a; }
    friend ModInt operator + (int y,const ModInt &a){int x0=y+a.x; return
ModInt(x0<mod?x0:x0-mod);}
    friend ModInt operator - (int y,const ModInt &a){int x0=y-a.x; return
ModInt(x0<0?x0+mod:x0);}
    friend ModInt operator * (int y,const ModInt &a){return
ModInt(1ll*y*a.x%mod);}
    friend ModInt operator / (int y,const ModInt &a){return ModInt(y)/a;}
    friend ostream &operator<<(ostream &os,const ModInt &a) {return os<<a.x;}
    friend istream &operator>>(istream &is,ModInt &t){return is>>t.x;}
    ModInt pow(int64_t n) const
    {
        ModInt res(1),mul(x);
        while(n)
        {
            if(n&1) res*=mul;
            mul*=mul;
            n>>=1;
        }
        return res;
    }
    ModInt inv() const
    {
        int a=x,b=mod,u=1,v=0;
        while(b)
        {
            int t=a/b;
            a-=t*b;swap(a,b);
            u-=t*v;swap(u,v);
        }
        if(u<0) u+=mod;
        return u;
    }
};
using mint=ModInt<998244353>;
struct Yanghui
{
    vector<vector<mint>> C;
    Yanghui(int sz)
    {
        C.resize(sz+1);
        for(int i=0;i<=sz;++i) C[i].resize(sz+1);
        C[0][0]=1;
        for(int i=1;i<=sz;++i)
        {
            C[i][0]=C[i][i]=1;
            for(int j=1;j<=i-1;++j)
            {
                C[i][j]=C[i-1][j-1]+C[i-1][j];
            }
        }
```

```
    }
};
int main()
{
    Yanghui yan(500);
    cout<<yan.C[4][2]<<'\n';
}
```

图论

树剖

```
#include <bits/stdc++.h>
using namespace std;
struct edge
{
    int to,ne;
    edge(int x=0,int y=0):to(x),ne(y){}
};
struct Cuttree
{
    int tot;
    vector<edge> e;
    vector<int> head,dep,siz,son,top,fa;
    Cuttree(int sz)
    {
        head.resize(sz+1,0);fa.resize(sz+1,0);
        dep.resize(sz+1,0);siz.resize(sz+1,0);
        son.resize(sz+1,0);top.resize(sz+1,0);
        tot=0;e.resize(2*sz+1,edge());
    }
    void add(int x,int y)
    {
        e[++tot].to=y;
        e[tot].ne=head[x];
        head[x]=tot;
    }
    void dfs1(int x)
    {
        siz[x]=1;
        dep[x]=dep[fa[x]]+1;
        for(int i=head[x];i;i=e[i].ne)
        {
            int dd=e[i].to;
            if(dd==fa[x])continue;
            fa[dd]=x;
            dfs1(dd);
            siz[x]+=siz[dd];
            if(!son[x]||siz[son[x]]<siz[dd])son[x]=dd;
        }
    }
    void dfs2(int x,int tv)
    {
        top[x]=tv;
```

```
        if(son[x])dfs2(son[x],tv);
        for(int i=head[x];i;i=e[i].ne)
        {
            int dd=e[i].to;
            if(dd==fa[x]||dd==son[x]) continue;
            dfs2(dd,dd);
        }
    }
    int lca(int x,int y)
    {
        while(top[x]!=top[y])
        {
            if(dep[top[x]]>=dep[top[y]])x=fa[top[x]];
            else y=fa[top[y]];
        }
        return dep[x]<dep[y]?x:y;
    }
};
int main()
{
    int n,m,s;cin>>n>>m>>s;//s为根
    Cuttree cte(n);
    for(int i=1,x,y;i<n;++i)cin>>x>>y,cte.add(x,y),cte.add(y,x);
    cte.dfs1(s);cte.dfs2(s,s);
    for(int i=1,x,y;i<=m;++i) cin>>x>>y,cout<<cte.lca(x,y)<<'\n';
}
```

dijkstra

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=1e4+4;
struct edge
{
    int v,w;
    edge(int x=0,int y=0):v(x),w(y){}
};
struct node
{
    int dis,u;
    node(int x=0,int y=0):dis(x),u(y){}
    bool operator>(const node& a)const{ return dis>a.dis;}
};
struct Dist
{
    vector<vector<edge>> e;
    vector<int> dis;
    vector<bool> vis;
    priority_queue<node,vector<node>,greater<node>> qu;
    int s;
    Dist(int ss,int sz)
    {
        s=ss;e.resize(sz+1);
        dis.resize(sz+1,2e9);vis.resize(sz+1,0);
    }
```

```
    void dijkstra()
    {
        dis[s]=0;qu.push({0,s});
        while(!qu.empty())
        {
            int u=qu.top().u;qu.pop();
            if(vis[u]) continue;
            vis[u]=1;
            for(auto ed:e[u])
            {
                int v=ed.v,w=ed.w;
                if(dis[v]>dis[u]+w)
                {
                    dis[v]=dis[u]+w;
                    qu.push({dis[v],v});
                }
            }
        }
    }
};
int main()
{
    int n,m;cin>>n>>m;Dist dist(1,n);
    for(int i=0,x,y,z;i<m;++i)
    {
        cin>>x>>y>>z;
        dist.e[x].push_back({y,z});
        dist.e[y].push_back({x,z});
    }
    dist.dijkstra();
    for(int i=1;i<=n;++i) cout<<dist.dis[i]<<' ';//源点到i的最短距离
    return 0;
}
```

启发式合并

```
#include <bits/stdc++.h>
using namespace std;
struct Dstree
{
    vector<vector<int>> g;
    vector<int> sz,big,col,L,R,Node,cnt,ans,dep;
    int totcol,totdfn;
    Dstree(int siz,int s=1)
    {
        g.resize(siz+1);sz.resize(siz+1,0);
        big.resize(siz+1,0);col.resize(siz+1,0);
        L.resize(siz+1,0);R.resize(siz+1,0);
        Node.resize(siz+1,0);cnt.resize(siz+1,0);
        ans.resize(siz+1,0);totcol=totdfn=0;
        dep.resize(siz+1,0);dep[s]=1;
    }
    void add(int u)
    {
        if(cnt[col[u]]==0) ++totcol;
```

```
        ++cnt[col[u]];
    }
    void del(int u)
    {
        --cnt[col[u]];
        if(cnt[col[u]]==0) --totcol;
    }
    int getAns() {return totcol;}
    void dfs0(int u,int p)
    {
        L[u]=++totdfn;
        Node[totdfn]=u;
        sz[u]=1;
        for(int v:g[u])
        {
            if(v!=p)
            {
                dfs0(v,u);
                sz[u]+=sz[v];
                if(!big[u]||sz[big[u]]<sz[v]) big[u]=v;
            }
        }
        R[u]=totdfn;
    }
    void dfs1(int u,int p,bool keep)
    {
        for(int v:g[u])if(v!=p&&v!=big[u]) dfs1(v,u,false);
        if(big[u]) dfs1(big[u],u,true);
        for(int v:g[u])
        {
            if(v!=p&&v!=big[u])
            {
                for(int i=L[v];i<=R[v];++i) add(Node[i]);
            }
        }
        add(u);ans[u]=getAns();
        if(keep==false)for(int i=L[u];i<=R[u];++i) del(Node[i]);
    }
};
int main()
{
    int n;cin>>n;Dstree dst(n);
    for(int i=1;i<=n;++i) cin>>dst.col[i];
    for(int i=1,u,v;i<n;++i)
    {
        cin>>u>>v;
        dst.g[u].push_back(v);
        dst.g[v].push_back(u);
    }
    dst.dfs0(1,0);dst.dfs1(1,0,false);
    cin>>n;
    for(int i=1,x;i<=n;++i) cin>>x,cout<<dst.ans[x]<<'\n';//x子树颜色数量
}
```

欧拉回路

```
#include <bits/stdc++.h>
using namespace std;
struct Hier
{
    vector<vector<int>> edge;
    vector<vector<bool>> vis;
    int n,m;stack<int> ans;
    Hier(int nn,int mm):n(nn),m(mm)
    {
        edge.resize(n+1);
        vis.resize(n+1);
    }
    void dfs(int x)
    {
        for(int i=0;i<(int)edge[x].size();++i)
        {
            if(vis[x][i]) continue;
            vis[x][i]=true;
            int j=lower_bound(edge[edge[x][i]].begin(),edge[edge[x][i]].end(),x)-
edge[edge[x][i]].begin();
            vis[edge[x][i]][j]=true;
            dfs(edge[x][i]);
        }
        ans.push(x);
    }
};
int main()
{
    int n,m;cin>>n>>m;
    Hier he(n,m);
    for(int i=0,x,y;i<m;++i)
    {
        cin>>x>>y;
        he.edge[x].push_back(y);
        he.vis[x].push_back(0);
        if(x!=y)
        {
            he.edge[y].push_back(x);
            he.vis[y].push_back(0);
        }
    }
    he.dfs(1);
    if(m+1!=(int)he.ans.size()) cout<<-1<<'\n';
    else
    {
        set<pair<int,int>> vis;
        vector<int> ans;
        while(!he.ans.empty()) ans.push_back(he.ans.top()),he.ans.pop();
        for(int i=1;i<=m;++i)
        {
            if(vis.count({min(ans[i-1],ans[i]),max(ans[i-1],ans[i])}))
            {
                return cout<<-1<<'\n',0;
```

```
            }
            vis.insert({min(ans[i-1],ans[i]),max(ans[i-1],ans[i])});
        }
        for(auto &it:ans) cout<<it<<' ';//输出欧拉通路
    }
}
```

Johnson全源最短路

```
#include <bits/stdc++.h>
using namespace std;
struct Edge
{
    int u,v,w;
    Edge(int x=0,int y=0,int z=0):u(x),v(y),w(z){}
};
struct Johnson
{
    vector<Edge> edges;
    vector<vector<Edge>> adj;
    vector<int> dist;
    Johnson(){}
    bool BellmanFord(vector<Edge> &ed,int n,int s)
    {
        dist.resize(n+1,INT_MAX);dist[s]=0;
        bool negtiveCycle;
        for(int i=1;i<=n;++i)
        {
            negtiveCycle=false;
            for(auto e:ed)
            {
                if(dist[e.u]<INT_MAX&&dist[e.v]>dist[e.u]+e.w)
                {
                    dist[e.v]=dist[e.u]+e.w;
                    negtiveCycle=true;
                }
            }
            if(!negtiveCycle) break;
        }
        return negtiveCycle;
    }
    vector<int> Dijkstra(int n,int s)
    {

 priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>> pq;
        vector<int> dist(n+1,INT_MAX);
        vector<bool> vis(n+1,false);
        dist[s]=0;pq.push({0,s});
        while(!pq.empty())
        {
            auto node=pq.top();pq.pop();
            int u=node.second;
            if(vis[u]) continue;
```

```
            vis[u]=true;
            for(auto e:adj[u])
            {
                int v=e.v,w=e.w;
                if(dist[u]<INT_MAX &&dist[v]>dist[u]+w)
                {
                    dist[v]=dist[u]+w;
                    pq.push({dist[v],v});
                }
            }
        }
        return dist;
    }
    void getans(int n)
    {
        vector<Edge> edges_new=edges;
        for(int v=1;v<=n;++v) edges_new.push_back({0,v,0});
        bool hasCycle=BellmanFord(edges_new,n,0);
        if(hasCycle)
        {
            cout<<"-1"<<'\n';
            return ;
        }
        for(auto &e:edges_new) e.w=e.w+dist[e.u]-dist[e.v];
        adj.resize(n+1);
        for(auto const e:edges_new) adj[e.u].push_back(e);
        vector<vector<int>> dist_all(n+1,vector<int>(n+1,0));
        for(int i=1;i<=n;++i)
        {
            auto d=Dijkstra(n,i);
            for(int j=1;j<=n;++j)
            {
                if(d[j]==INT_MAX) continue;
                dist_all[i][j]=d[j]+dist[j]-dist[i];
            }
        }
        for(int i=1;i<=n;++i)
        {
            for(int j=1;j<=n;++j)
            {
                cout<<i<<' '<<j<<' '<<dist_all[i][j]<<'\n';//i到j的最短距离
            }
        }
    }
};
int main()
{
    int n,m;cin>>n>>m;Johnson john;
    for(int i=0,x,y,z;i<m;++i)
    {
        cin>>x>>y>>z;
        john.edges.push_back({x,y,z});
        john.edges.push_back({y,x,z});
    }
    john.getans(n);
}
```

最小生成树

```
#include <bits/stdc++.h>
using namespace std;
struct node
{
    int x,y,z;
    node(int xx=0,int yy=0,int zz=0):x(xx),y(yy),z(zz){}
};
struct Mintree
{
    vector<node> edge;
    vector<int> fa,cnt;
    int n;long long sum;
    Mintree(int sz)
    {
        fa.resize(sz+1);edge.clear();
        iota(fa.begin(),fa.end(),0);
        cnt.resize(sz+1,1);n=sz;sum=0;
    }
    int get(int x) {return x==fa[x]?x:fa[x]=get(fa[x]);}
    void build()
    {
        sort(edge.begin(),edge.end(),[](node a,node b){return a.z<b.z;});
        for(auto &it:edge)
        {
            int x=get(it.x);
            int y=get(it.y);
            if(x==y) continue;
            cnt[x]+=cnt[y];fa[y]=x;
            sum+=it.z;
        }
        int ans=0;
        for(int i=1;i<=n;++i)if(i==fa[i]) ++ans;
        if(ans>1) cout<<"impossible"<<'\n';
        else cout<<sum<<'\n';
    }
};
int main()
{
    int n,m;cin>>n>>m;Mintree mte(n);
    for(int i=1,u,v,w;i<=m;++i)
    {
        cin>>u>>v>>w;
        mte.edge.push_back(node(u,v,w));
    }
    mte.build();
}
```

倍增求LCA

```
#include <bits/stdc++.h>
using namespace std;
pair<vector<int>,vector<int>>
    remaketre(int n,vector<array<int,3>>edges,
        function<bool(array<int,3>,array<int,3>)>comp=less<>())
{
    vector<int> p(2*n);
    for(int i=1;i<2*n;i++) p[i]=i;
    function<int(int)> find=[&](int x){if(x!=p[x])p[x]=find(p[x]);return p[x];};
    int cnt=n;
    sort(edges.begin(),edges.end(),comp);
    vector<int> value(n+1,0),parent(2*n,0);
    for(auto [weight,from,to]:edges)
    {
        from=find(from);
        to=find(to);
        if(from==to) continue;
        p[from]=++cnt;
        p[to]=cnt;
        value.push_back(weight);
        parent[from]=cnt;
        parent[to]=cnt;
    }
    parent.resize(value.size());
    return {value,parent};
}
struct LCA
{
    LCA(int n,vector<int>& parent):parent(parent),depth(n+1),pa(n+1)
    {
        for(int i=1;i<=n;++i)
        {
            if(this->parent[i]<=0) this->parent[i]=i;
            pa[i][0]=this->parent[i];
        }
        for(int i=1;i<20;++i)
        {
            for(int j=1;j<=n;j++)
            {
                pa[j][i]=pa[pa[j][i-1]][i-1];
            }
        }
        function<int(int)> get_depth=[&](int now)
        {
            if(now<=0||now>n) return 0;
            if(depth[now]) return depth[now];
            if(parent[now]<=0||parent[now]==now) return depth[now]=1;
            return depth[now]=get_depth(parent[now])+1;
        };
        for(int i=1;i<=n;++i) depth[i]=get_depth(i);
    }
    int lca(int u,int v)
    {
```

```
        if(pa[u][19]!=pa[v][19]) return -1;
        if(depth[u]<depth[v]) swap(u,v);
        for(int i=19;depth[u]>depth[v];--i)
        {
            if(depth[pa[u][i]]>=depth[v])u=pa[u][i];
        }
        if(u==v) return u;
        for(int i=19;i>=0;--i)
        {
            if(pa[u][i]!=pa[v][i])u=pa[u][i],v=pa[v][i];
        }
        return pa[u][0];
    }
private:
    vector<int> parent,depth;
    vector<array<int,20>> pa;
};
int main()
{
    int n,m;
    cin>>n>>m;
    vector<array<int,3>> edges;
    for(int i=1,x,y,w;i<=m;++i)
    {
        cin>>x>>y>>w;
        edges.push_back({w,x,y});
    }
    auto [value,parent]=remaketre(n,edges);
    vector<vector<int>> nex;
    int s=value.size()-1;nex.resize(s+1);
    for(int i=1;i<=s;++i)
    {
        if(parent[i]) nex[parent[i]].push_back(i);//新树,s为根
    }
    LCA t((int)value.size()-1,parent);
    int q;cin>>q;
    for(int i=0,x,y,idx;i<q;++i)
    {
        cin>>x>>y;idx=t.lca(x,y);
        if(idx==-1) cout<<-1<<'\n';//不连通
        else cout<<value[idx]<<'\n';//路径最大边权
    }
}
```

Tarjan

```
#include <bits/stdc++.h>
using namespace std;
struct Tarjan
{
    vector<vector<int>> e,e1,ans;//旧图,新图,存储强连通块
    vector<int> col,dfn,low,stk,in;
    int cnt,top,_cnt;//_cnt为强连通数量
    Tarjan(int sz)
    {
```

```
        e.resize(sz+1);e1.resize(sz+1);
        ans.resize(sz+1);col.resize(sz+1);
        dfn.resize(sz+1);low.resize(sz+1);
        stk.resize(sz+1);in.resize(sz+1);
        cnt=top=_cnt=0;
    }
    void tarjan(int u)
    {
        dfn[u]=low[u]=++cnt;
        stk[++top]=u;
        for(auto v:e[u])
        {
            if(!dfn[v]) tarjan(v),low[u]=min(low[u],low[v]);
            else if(!col[v]) low[u]=min(low[u],dfn[v]);
        }
        if(low[u]==dfn[u])
        {
            col[u]=++_cnt;
            ans[_cnt].push_back(u);
            while(stk[top]!=u) ans[_cnt].push_back(stk[top]),col[stk[top-
-]]=_cnt;
            --top;
        }
    }
    void topo()
    {
        int res=0;
        queue<int> qu;
        for(int i=1;i<=_cnt;++i) if(!in[i]) qu.push(i);
        while(!qu.empty())
        {
            ++res;
            int t=qu.front();qu.pop();
            for(auto &c:e1[t])
            {
                --in[c];
                if(in[c]==0) qu.push(c);
            }
        }
        cout<<res<<'\n';
    }
    void getans(int m)//所有强连通分量
    {
        for(int i=1;i<=m;++i) if(!dfn[i]) tarjan(i);
        for(int i=1;i<=m;++i)
        {
            if(!ans[col[i]].empty())
            {
                for(auto &t:ans[col[i]]) cout<<t<<' ';
                cout<<'\n';
                ans[col[i]].clear();
            }
        }
    }
    void build(int m)//建立新图
    {
```

```
        for(int i=1;i<=m;++i)
        {
            for(auto &c:e[i])
            {
                if(col[i]!=col[c])
                {
                    e1[col[i]].push_back(col[c]);
                    ++in[col[c]];
                }
            }
        }
        topo();//获取新图所有非环点数量
    }
};
int main()
{
    int n,m;cin>>n>>m;Tarjan tarj(n);
    for(int i=0,x,y;i<m;++i)
    {
        cin>>x>>y;
        tarj.e[x].push_back(y);
    }
    tarj.getans(n);
    tarj.build(n);
}
```

2-SET

```
#include <bits/stdc++.h>
using namespace std;
struct TwoSat
{
    int n;
    vector<vector<int>> e;
    vector<bool> ans;
    TwoSat(int n):n(n),e(2*n),ans(n){}
    //加析取条件(u,v),f,g代表x,y是否为非,f=1代表x为非
    void addClause(int u,bool f,int v,bool g)
    {
        e[2*u+!f].push_back(2*v+g);
        e[2*v+!g].push_back(2*u+f);
    }
    //找是否有合法解
    bool satisfiable()
    {
        vector<int> id(2*n,-1),dfn(2*n,-1),low(2*n,-1);
        vector<int> stk;
        int now=0,cnt=0;
        function<void(int)> tarjan = [&](int u)
        {
            stk.push_back(u);
            dfn[u]=low[u]=now++;
            for(auto v:e[u])
            {
                if(dfn[v]==-1)
```

```
                {
                    tarjan(v);
                    low[u]=min(low[u],low[v]);
                }
                else if(id[v]==-1)
                {
                    low[u]=min(low[u],dfn[v]);
                }
            }
            if(dfn[u]==low[u])
            {
                int v;
                do
                {
                    v=stk.back();
                    stk.pop_back();
                    id[v]=cnt;
                }while(v!=u);
                ++cnt;
            }
        };
        for(int i=0;i<2*n;++i) if(dfn[i]==-1) tarjan(i);
        for (int i=0;i<n;++i)
        {
            if(id[2*i]==id[2*i+1]) return false;
            ans[i]=id[2*i]>id[2*i+1];
        }
        return true;
    }
    //返回一种 (x1,...,xn) 的合法解
    vector<bool> answer() {return ans;}
};
int main()
{
    int n,m;cin>>n>>m;TwoSat ts(n);
    for(int i=0,x,y;i<m;++i)
    {
        cin>>x>>y;
        //加析取条件(u,v),它们的符号分别为f,g,f=1代表u为非
        ts.addClause(x,1,y,1);//a[x]和a[y]不能同时为0
        ts.addClause(x,0,y,0);//a[x]和a[y]不能同时为1
    }
    //找是否有合法解
    if(!ts.satisfiable()) cout<<-1<<'\n';
    else
    {
        //返回合法解
        auto ans=ts.answer();
        for(auto it:ans) cout<<it<<' ';
    }
}
```

二分图

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
struct Edge
{
    int to,id;
    operator int()const{return to;}
};
struct LowLink
{
    int n;
    vector<vector<Edge>> g;
    vector<int> in,out,low;
    int ts;
    LowLink(const vector<vector<Edge>>&g):n(int(g.size())-1),g(g)
    {
        ts=0;low.assign(n+1,0);
        in.assign(n+1,0);out.assign(n+1,0);
        for(int i=1;i<=n;++i) if(!in[i]) tarjan(i,-1);
        id.assign(n+1,0);cnt=0;
        for(int i=1;i<=n;++i) if(!id[i]) dfs(i,-1);
        group.resize(cnt+1);
        for(int i=1;i<=n;++i) group[id[i]].push_back(i);
    }
    void tarjan(int u,int from)
    {
        in[u]=low[u]=++ts;
        for(auto j:g[u])
        {
            if(!in[j]) tarjan(j,j.id),low[u]=min(low[u],low[j]);
            else if(j.id!=from) low[u]=min(low[u],in[j]);
        }
        out[u]=ts;
    }
    int cnt;
    vector<vector<int>> group;
    vector<int> id;
    void dfs(int u,int fa)
    {
        if (fa!=-1&&low[u]<=in[fa]) id[u]=id[fa];
        else id[u]=++cnt;
        for(auto j:g[u]) if(!id[j]) dfs(j,u);
    }
    void getans()//输出边双连通分量
    {
        vector<vector<int>> ng(cnt+1);
        for(int i=1;i<=n;++i)
        {
            for(auto [j,id1]:g[i])
            {
                if(id[i]!=id[j])
                {
                    ng[id[i]].push_back(id[j]);
```

```
                    cout<<i<<' '<<j<<'\n';
                }
            }
        }
    }
};
int main()
{
    int n,m;cin>>n>>m;
    vector<vector<Edge>>g(n+1);
    for(int i=0,a,b;i<m;++i)
    {
        cin>>a>>b;
        g[a].push_back({b,i});
        g[b].push_back({a,i});
    }
    LowLink lk(g);
    lk.getans();
}
```

字符串

AC自动机

```
#include <bits/stdc++.h>
using namespace std;
struct Acrobo
{
    int tot;
    vector<vector<int>> son;
    vector<int> fail,q,len,id;
    Acrobo(int sz)
    {
        son.resize(sz+1);fail.resize(sz+1);
        q.resize(sz+1);len.resize(sz+1);
        id.resize(sz+1);tot=0;
        for(int i=0;i<sz+1;++i)
        {
            son[i].resize(26,0);
        }
    }
    void insert(string s,int p)
    {
        for(int l=s.length(),x=0,i=0,w;i<l;++i)
        {
            if(!son[x][w=s[i]-'a']) son[x][w]=++tot;
            x=son[x][w];
            if(i==l-1) id[x]=p;
        }
    }
    void make()
    {
        int h=1,t=0,i,x;fail[0]=-1;
        for(i=0;i<26;++i) if(son[0][i]) q[++t]=son[0][i];
```

```
        while(h<=t) for(x=q[h++],i=0;i<26;++i)
        {
            if(son[x][i]) fail[son[x][i]]=son[fail[x]][i],q[++t]=son[x][i];
            else son[x][i]=son[fail[x]][i];
        }
    }
    void find(string s)
    {
        for(int l=s.length(),x=0,i=0,w,j;i<l;++i)
        {
            x=son[x][w=s[i]-'a'];
            for(j=x;j;j=fail[j])
            {
                if(id[j]) cout<<i+1<<':'<<id[j]<<'\n';
            }
        }
    }
};
int main()
{
    int n,m;cin>>n>>m;
    Acrobo acr((int)1e6);
    string s;
    for(int i=1;i<=n;++i)
    {
        cin>>s;
        acr.len[i]=s.length();
        acr.insert(s,i);
    }
    acr.make();
    while(m--) cin>>s,acr.find(s);
}
```

字符串哈希

```
#include <bits/stdc++.h>
using namespace std;
struct Hash
{
    const static int mod=1e9+7;
    const static int p=233;
    int gethash(string s,int res=0)
    {
        for(auto &it:s) res=(res*p+it-'a')%mod;
        return res;
    }
};
int main()
{
    Hash ha;
    string s;cin>>s;
    cout<<ha.gethash(s)<<'\n';
}
```

KMP

```
#include <bits/stdc++.h>
using namespace std;
struct Kmp
{
    string s,p;
    vector<int> nxt;
    Kmp(string ss,string pp)
    {
        s=ss;p=pp;
        nxt.resize(s.size());
    }
    void kmp()
    {
        int i,j,n=s.size(),m=p.size();
        for(nxt[0]=j=-1,i=1;i<n;nxt[i++]=j)
        {
            while(~j&&s[j+1]!=s[i]) j=nxt[j];
            if(s[j+1]==s[i]) ++j;
        }
        for(i=0;i<n;++i) cout<<nxt[i]+1<<' ';//前i个字符最长公共前后缀
        cout<<'\n';
        for(j=-1,i=0;i<m;++i)
        {
            while(~j&&s[j+1]!=p[i]) j=nxt[j];
            if(s[j+1]==p[i]) ++j;
            if(j==n-1) cout<<i+1<<' ',j=nxt[j];//匹配成功位置(末尾)
        }
    }
};
int main()
{
    string s,p;cin>>s>>p;
    Kmp km(s,p);//p中找s
    km.kmp();
}
```

马拉车

```
#include <bits/stdc++.h>
using namespace std;
struct Mana
{
    string s;
    vector<int> f,g;
    Mana(string ss)
    {
        s=ss;f.resize(2*ss.size()+3);
        g.resize(2*ss.size()+3);f[0]=0;
    }
    void Manacher()
    {
        string ps="$";ps+='#';
```

```
        for(auto &it:s) ps+=it,ps+='#';
        ps+='@';
        int i,r,p,m=ps.size();
        for(r=p=0,f[1]=1,i=2;i<m;++i)
        {
            for(f[i]=r>i?min(r-i,f[2*p-i]):1;ps[i-f[i]]==ps[i+f[i]];f[i]++);
            if(i+f[i]>r) r=i+f[i],p=i;
        }
        for(i=0;i<m;++i) g[i]=0;
        for(i=2;i<m;++i) g[i-f[i]+1]=max(g[i-f[i]+1],i+1);
        for(i=1;i<=m;++i) g[i]=max(g[i],g[i-1]);
        for(i=2;i<m-1;i+=2) cout<<g[i]-i<<' ';//以i开头maxlen
    }
};
int main()
{
    string s;cin>>s;
    Mana man(s);
    man.Manacher();
}
```

回文自动机

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
struct PAM
{
    int size,tot,last;
    vector<int> cnt,len,fail;
    vector<char> s;
    vector<vector<int>> tr;
    PAM(int sz)
    {
        cnt.resize(sz+1);tr.resize(26);
        for(int i=0;i<26;++i) tr[i].resize(sz+1);
        len.resize(sz+1);fail.resize(sz+1);
        s.resize(sz+1);
    }
    int node(int l)// 建立一个新节点，长度为 l
    {
        size++;
        for(int i=0;i<26;++i) tr[i][size]=0;
        len[size]=l;
        fail[size]=cnt[size]=0;
        return size;
    }
    void init()// 初始化
    {
        size=-1;last=0;
        s[tot=0]='$';
        node(0);node(-1);
        fail[0]=1;
    }
    int getfail(int x)// 找后缀回文
```

```
    {
        while(s[tot-len[x]-1]!=s[tot]) x=fail[x];
        return x;
    }
    void add(char c)// 建树
    {
        s[++tot]=c;
        int now=getfail(last);
        if(!tr[c][now])
        {
            int x=node(len[now]+2);
            fail[x]=tr[c][getfail(fail[now])];
            tr[c][now]=x;
        }
        last=tr[c][now];
        cnt[last]++;
    }
};
int main()
{
    string s;cin>>s;
    int n=s.size();s='$'+s;
    PAM pa(n*3);
    pa.init();
    for(int i=1;i<=n;++i)pa.add(s[i]-'a');
    for(int i=pa.size;i>=2;--i)
    {
        pa.cnt[pa.fail[i]]+=pa.cnt[i];
    }
    ll ans=0;
    for(int i=2;i<=pa.size;++i)
    {
        //len为长度,cnt为数量,size为回文串数量
        ans=max(ans,1ll*pa.len[i]*pa.cnt[i]);
    }
    cout<<ans<<'\n';
}
```

杂项

比较

```
#include <bits/stdc++.h>
using namespace std;
const double PI=acos(-1.0);
struct Complex
{
    double x,y;
    Complex(double _x=0.0,double _y=0.0):x(_x),y(_y){}
    Complex operator-(const Complex &b) const {return Complex(x-b.x,y-b.y);}
    Complex operator+(const Complex &b) const {return Complex(x+b.x,y+b.y);}
    Complex operator*(const Complex &b) const {return Complex(x*b.x-
y*b.y,x*b.y+y*b.x);}
};
```

```
int main()
{

}
```

取模类

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;
using ld=long double;
using ull=unsigned long long;
template<const int T>
struct ModInt
{
    const static int mod=T;
    int x;
    ModInt(int x = 0):x(x%mod){}
    ModInt(long long x):x(int(x%mod)){}
    int val() {return x;}
    ModInt operator + (const ModInt &a) const {int x0=x+a.x;return ModInt(x0<mod?
x0:x0-mod);}
    ModInt operator - (const ModInt &a) const {int x0=x-a.x;return ModInt(x0<0?
x0+mod:x0);}
    ModInt operator * (const ModInt &a) const {return ModInt(1ll * x * a.x %
mod); }
    ModInt operator / (const ModInt &a) const {return *this*a.inv();}
    bool operator == (const ModInt &a) const {return x==a.x;};
    bool operator != (const ModInt &a) const {return x!=a.x;};
    void operator += (const ModInt &a) {x+=a.x;if(x>=mod) x-=mod;}
    void operator -= (const ModInt &a) {x-=a.x;if(x<0) x+=mod;}
    void operator *= (const ModInt &a) {x=1ll*x*a.x%mod;}
    void operator /= (const ModInt &a) {*this = *this / a; }
    friend ModInt operator + (int y,const ModInt &a){int x0=y+a.x; return
ModInt(x0<mod?x0:x0-mod);}
    friend ModInt operator - (int y,const ModInt &a){int x0=y-a.x; return
ModInt(x0<0?x0+mod:x0);}
    friend ModInt operator * (int y,const ModInt &a){return
ModInt(1ll*y*a.x%mod);}
    friend ModInt operator / (int y,const ModInt &a){return ModInt(y)/a;}
    friend ostream &operator<<(ostream &os,const ModInt &a) {return os<<a.x;}
    friend istream &operator>>(istream &is,ModInt &t){return is>>t.x;}
    ModInt pow(int64_t n) const
    {
        ModInt res(1),mul(x);
        while(n)
        {
            if(n&1) res*=mul;
            mul*=mul;
            n>>=1;
        }
        return res;
    }
    ModInt inv() const
    {
```

```
        int a=x,b=mod,u=1,v=0;
        while(b)
        {
            int t=a/b;
            a-=t*b;swap(a,b);
            u-=t*v;swap(u,v);
        }
        if(u<0) u+=mod;
        return u;
    }
};
using mint=ModInt<998244353>;
int main()
{

}
```

随机数

```
#include <bits/stdc++.h>
using namespace std;
using ull=unsigned long long;
int main()
{
    int n;cin>>n;
    mt19937_64 rnd(0);
    uniform_int_distribution<ull> make(0,(ull)1<<63);
    vector<ull> hsh(n);
    for(int i=0;i<n;++i) hsh[i]=make(rnd);
    for(int i=0;i<n;++i) cout<<hsh[i]<<'\n';
}
```

pbds

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;
typedef pair<int, int> pii;
#define pb push_back
#define mp make_pair
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
__gnu_pbds::tree<pair<int,int>, __gnu_pbds::null_type,less<pair<int,int>>,
                 __gnu_pbds::rb_tree_tag,
                 __gnu_pbds::tree_order_statistics_node_update>
    trr;
/*
insert(x)：向树中插入一个元素 x，返回 std::pair<point_iterator, bool>。
erase(x)：从树中删除一个元素/迭代器 x，返回一个 bool 表明是否删除成功。
order_of_key(x)：返回 x 以 Cmp_Fn 比较的排名。
find_by_order(x)：返回 Cmp_Fn 比较的排名所对应元素的迭代器。
lower_bound(x)：以 Cmp_Fn 比较做 lower_bound，返回迭代器。
```

```
upper_bound(x)：以 Cmp_Fn 比较做 upper_bound，返回迭代器。
join(x)：将 x 树并入当前树，前提是两棵树的类型一样，不相交，如果相交抛出异。x 树被删除。
split(x,b)：以 Cmp_Fn 比较，小于等于 x 的属于当前树，其余的属于 b 树。
empty()：返回是否为空。
size()：返回大小。
*/
int main()
{
    int cnt=0;
    trr.insert(mp(1,cnt++));
    trr.insert(mp(5,cnt++));
    trr.insert(mp(4,cnt++));
    trr.insert(mp(3,cnt++));
    trr.insert(mp(2,cnt++));
    //树上元素 {{1,0},{2,4},{3,3},{4,2},{5,1}}
    auto it=trr.lower_bound(mp(2,0));
    trr.erase(it);
    //树上元素 {{1,0},{3,3},{4,2},{5,1}}
    auto it2=trr.find_by_order(1);
    cout<<(*it2).first<<endl;
    //输出排名 0 1 2 3 中的排名 1 的元素的 first:1
    int pos=trr.order_of_key(*it2);
    cout<<pos<<endl;
    // 输出排名
    decltype(trr) newtr;
    trr.split(*it2, newtr);
    for(auto i=newtr.begin();i!=newtr.end();++i)
    {
        cout<<(*i).first<<' ';
    }
    cout<<endl;
    //{4,2},{5,1} 被放入新树
    trr.join(newtr);
    for(auto i=trr.begin();i!=trr.end();++i) cout<<(*i).first<<' ';
    cout<<endl;
    cout<<newtr.size()<<endl;
    // 将 newtr 树并入 trr 树，newtr 树被删除。
}
```