ACM部分常用算法模板

作者：沈逸凡

时间：2020/10/17

组合数学

尼姆积

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

/* TEMPLATE BEGINS HERE*/

int n,sg[2][2]={0,0,0,1};
//definition

int mulp(int x,int y){
//calc nim production of x and y as power of 2
    if(x<2)return sg[x][y];

    int m;
    for(m=2;m*m<=x;m*=m);
    int p=x/m,s=y/m,t=y%m;
    int d1=mulp(p,s);
    int d2=mulp(p,t);
    return (m*(d1^d2))^mulp(m/2,d1);
}

int mul(int x,int y){
//calc nim production of x and y
    if(x<y)swap(x,y);
    if(x<2)return sg[x][y];

    int m;
    for(m=2;m*m<=x;m*=m);
    int p=x/m,q=x%m,s=y/m,t=y%m;
    int c1=mul(p,s);
    int c2=mul(p,t)^mul(q,s);
    int c3=mul(q,t);
    return (m*(c1^c2))^c3^mulp(m/2,c1);
}

/* Call this function DIRECTLY for usage */
/* May exploit DP(Memory search) for acceleration*/

int solve(int x,int y){
    return mul(x,y);
}

/* TEMPLATE ENDS HERE*/
```

```
int main(){

}
```

数据结构

CDQ分治

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

const int maxn=1e5+5;

struct node{
    int x,y,z,w,ans;
}a[maxn],b[maxn];

int ans[maxn];
int n,k,nn;
map<node,int> mp;

bool cmpx(node a,node b){
    if(a.x==b.x&&a.y==b.y)return a.z<b.z;
    if(a.x==b.x)return a.y<b.y;
    return a.x<b.x;
}

struct FWT{
    const static int N=maxn<<1;
    int a[N];
    void add(int x,int d){
        while(x<N){
            a[x]+=d;
            x+=x&(-x);
        }
    }
    int ask(int x){
        int res=0;
        while(x){
            res+=a[x];
            x-=x&(-x);
        }
        return res;
    }
}fwt;

void cdq(int l,int r){
    if(l==r)return;
    int m=(l+r)>>1;
    cdq(l,m);
```

```
    cdq(m+1,r);
    sort(a+l,a+m+1,cmpy);
    sort(a+m+1,a+r+1,cmpy);
    int i,j;
    for(i=l,j=m+1;j<=r;++j){
        while(a[i].y<=a[j].y&&i<=m){
            fwt.add(a[i].z,a[i].w);
            ++i;
        }
        a[j].ans+=fwt.ask(a[j].z);
    }
    for(j=l;j<i;++j)fwt.add(a[j].z,-a[j].w);
}

int main(){
    ios::sync_with_stdio(0);
    cin>>n>>k;
    for(int i=0;i<n;++i){
        cin>>b[i].x>>b[i].y>>b[i].z;
    }
    sort(b,b+n,cmpx);
    for(int i=0,c=0;i<n;++i){
        ++c;
        if(b[i].x!=b[i+1].x||b[i].y!=b[i+1].y||b[i].z!=b[i+1].z){
            a[nn]=b[i];
            a[nn].w=c;
            ++nn;
            c=0;
        }
    }
    cdq(0,nn-1);
    for(int i=0;i<nn;++i){
        ans[a[i].ans+a[i].w-1]+=a[i].w;
    }
    for(int i=0;i<n;++i){
        cout<<ans[i]<<endl;
    }
}
```

线性基

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

const int M=30;

struct linear{
    int p[M];
    //空构造函数
    linear(){
        memset(p,0,sizeof(p));
    }
    //复制构造函数
```

```
    linear(const linear &x){
        for(int i=M-1;i>=0;--i)p[i]=x.p[i];
    }
    //清空
    void clear(){
        memset(p,0,sizeof(p));
    }
    //插入
    void insert(int x){
        for(int i=M-1;i>=0;--i){
            if(x&(1<<i)){
                if(p[i])x^=p[i];
                else{
                    p[i]=x;
                    return;
                }
            }
        }
    }
    //查询最大值
    int query(){
        int ans=0;
        for(int i=M-1;i>=0;--i){
            ans=max(ans,ans^p[i]);
        }
        return ans;
    }
    //合并两个线性基
    linear merge(linear x){
        linear ans(x);
        for(int i=M-1;i>=0;--i){
            if(p[i]){
                ans.insert(p[i]);
            }
        }
        return ans;
    }
};

int main(){

}
```

前缀线性基

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

const int N=100005;
const int M=32;
struct PrefixLinearBasis{
    int d[N][M];//前缀线性基
    int pos[N][M];//最后一个修改i这个位置的数
```

```
int num;//线性基中元素个数
PrefixLinearBasis(){
    memset(d,0,sizeof(d));
    memset(pos,0,sizeof(pos));
    num=0;
}
void clear(){
    memset(d,0,sizeof(d));
    memset(pos,0,sizeof(pos));
    num=0;
}
void add(int x){//向线性基中添加x
    num++;
    for(int i=M-1; i>=0; i--){//复制前num-1个线性基
        d[num][i]=d[num-1][i];
        pos[num][i]=pos[num-1][i];
    }

    int P=num;
    for(int i=M-1; i>=0; i--){
        if((x>>i)&1){
            if(d[num][i]){//插入失败
                if(pos[num][i]<P){//交换位置
                    swap(pos[num][i], P);
                    swap(d[num][i],x);
                }
                x^=d[num][i];//异或
            }
            else{//插入成功
                d[num][i]=x;
                pos[num][i]=P;
                break;
            }
        }
    }
}
int queryMax(int l,int r){//查询[l,r]中的最大值
    int res=0;
    for (int i=M-1; i>=0; i--){
        if(pos[r][i]<l)
            continue;
        if ((res^d[r][i])>res)
            res^=d[r][i];
    }
    return res;
}
int queryMin(int l,int r) {//查询[l,r]中的最小值
    for(int i=0; i<M; i++){
        if(pos[r][i]<l)
            continue;
        if(d[r][i])
            return d[r][i];
    }
    return 0;
}
```

```
}PLB;
```

标记永久化线段树

```
/*
例题为luogu3372 [https://www.luogu.com.cn/problem/P3372]
标记永久化线段树
无需pushdown操作的区间改
本模板中只实现区间加操作，区间求和查询
*/
#include<bits/stdc++.h>
using namespace std;
#define int long long

const int maxn=4e5+5;
struct PTST{
    #define ls o<<1,l,m
    #define rs o<<1|1,m+1,r
    int sum[maxn],tag[maxn];
    void build(int o,int l,int r,int a[]){
        if(l==r){
            sum[o]=a[l];
            tag[o]=0;
        }else{
            int m=(l+r)>>1;
            build(ls,a);
            build(rs,a);
            sum[o]=sum[o<<1]+sum[o<<1|1];
        }
    }
    void modify(int o,int l,int r,int L,int R,int d){
        sum[o]+=(min(R,r)-max(L,l)+1)*d;
        if(L<=l&&r<=R){
            tag[o]+=d;
            return;
        }
        int m=(l+r)>>1;
        if(L<=m)modify(ls,L,R,d);
        if(R>m)modify(rs,L,R,d);
    }
    int query(int o,int l,int r,int L,int R,int t){
        if(L<=l&&r<=R)return sum[o]+(min(R,r)-max(L,l)+1)*t;
        int ret=0;
        int m=(l+r)>>1;
        if(L<=m)ret+=query(ls,L,R,t+tag[o]);
        if(R>m)ret+=query(rs,L,R,t+tag[o]);
        return ret;
    }
}ptst;

/*usage as follows*/

int n,m;
```

```
int a[maxn];
void solve(){
    cin>>n>>m;
    for(int i=1;i<=n;++i)cin>>a[i];
    ptst.build(1,1,n,a);
    for(int i=1,op,t1,t2,t3;i<=m;++i){
        // for(int j=1;j<=n;++j){
        //     cout<<ptst.query(1,1,n,j,j,0)<<" ";
        // }cout<<"-----------"<<endl;
        cin>>op;
        if(op==1){
            cin>>t1>>t2>>t3;
            ptst.modify(1,1,n,t1,t2,t3);
        }else{
            cin>>t1>>t2;
            cout<<ptst.query(1,1,n,t1,t2,0)<<"\n";
        }
    }

}

int32_t main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    solve();
}
```

可持久化线段树

```
/*
例题为luogu3834 [https://www.luogu.com.cn/problem/P3834]
静态区间第k小，主席树
其他需求请修改modify/query逻辑
该模板不带修
*/
#include<bits/stdc++.h>
using namespace std;

const int maxn=2e5+5,LOG=20;
struct PST{
    int rt[maxn],sum[maxn*LOG],lson[maxn*LOG],rson[maxn*LOG],n=0;
    void build(int &o,int l,int r){
        o = ++n;
        if(l==r)return;
        int m=(l+r)>>1;
        build(lson[o],l,m);
        build(rson[o],m+1,r);
    }
    int modify(int o,int l,int r,int x){
        int p=++n;
        lson[p]=lson[o],rson[p]=rson[o],sum[p]=sum[o]+1;
        if(l<r){
            int m=(l+r)>>1;
```

```
            if(x<=m)lson[p]=modify(lson[p],l,m,x);
            else rson[p]=modify(rson[p],m+1,r,x);
        }
        return p;
    }
    int query(int lo,int ro,int l,int r,int k){
        if(l==r)return l;
        int m=(l+r)>>1;
        int x=sum[lson[ro]]-sum[lson[lo]];
        if(x>=k)return query(lson[lo],lson[ro],l,m,k);
        else return query(rson[lo],rson[ro],m+1,r,k-x);
    }
}pst;

/*usage as follows*/

int n,m;
int a[maxn],ind[maxn];
vector<int> v;
void solve(){
    cin>>n>>m;
    for(int i=0;i<n;++i){
        cin>>a[i];
        v.push_back(a[i]);
    }
    sort(v.begin(),v.end());
    v.erase(unique(v.begin(),v.end()),v.end());
    int N=v.size();
    for(int i=0;i<n;++i){
        int t=lower_bound(v.begin(),v.end(),a[i])-v.begin()+1;
        ind[t]=a[i];
        a[i]=t;
    }

    pst.build(pst.rt[0],1,N);
    for(int i=0;i<n;++i)pst.rt[i+1]=pst.modify(pst.rt[i],1,N,a[i]);
    for(int i=0,ql,qr,qk;i<m;++i){
        cin>>ql>>qr>>qk;;
        cout<<ind[pst.query(pst.rt[ql-1],pst.rt[qr],1,N,qk)]<<'\n';
    }

}

int main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    solve();
}
```

计算几何

凸包

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

const int maxn=1e5+6;

struct node{
    double x,y;
    node operator-(node rhs){
        return {x-rhs.x,y-rhs.y};
    }
    double operator&(node rhs){
        return x*rhs.y-y*rhs.x;
    }
    double dis(node rhs){
        return sqrt((x-rhs.x)*(x-rhs.x)+(y-rhs.y)*(y-rhs.y));
    }
}a[maxn];
bool cmp(node lhs,node rhs){
    return ((lhs-a[1])&(rhs-a[1]))>0;
}

struct CH{
    int n,s[maxn],top;
    void graham(int n){
        this->n=n;
        top=0;
        for(int i=1;i<=n;++i){
            cin>>a[i].x>>a[i].y;
            if(a[i].y<a[1].y||(a[i].y==a[1].y&&a[i].x<a[1].x)){
                swap(a[1],a[i]);
            }
        }
        sort(a+2,a+n+1,cmp);
        for(int i=1;i<=n;++i){
            while(top>2&&((a[s[top]]-a[s[top-1]])&(a[i]-a[s[top]]))<0)--top;
            s[++top]=i;
        }
    }
    double getans(){
        double ans=0;
        for(int i=1;i<top;++i)ans+=a[s[i]].dis(a[s[i+1]]);
        ans+=a[s[top]].dis(a[s[1]]);
        return ans;
    }
}ch;

int main(){
    ios::sync_with_stdio(0);
```

```
    int n;
    cin>>n;
    ch.graham(n);
    cout<<fixed<<setprecision(2)<<ch.getans()<<endl;
}
```

三角形外心

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

struct node{
    double x,y;
    double dis(const node &o){
        return sqrt((x-o.x)*(x-o.x)+(y-o.y)*(y-o.y));
    }
};

node findO(const node &p,const node &q,const node &r){
    //input should be nonlinear
    double a = 2 * (p.x - q.x);
    double b = 2 * (p.y - q.y);
    double c = p.x * p.x + p.y * p.y - q.x * q.x - q.y * q.y;
    double d = 2 * (p.x - r.x);
    double e = 2 * (p.y - r.y);
    double f = p.x * p.x + p.y * p.y - r.x * r.x - r.y * r.y;
    double g = a*e-b*d;
    return {(c*e-f*b)/g,(a*f-d*c)/g};
}

int main(){
    return 0;
}
```

线段交判定

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

struct point{
    double x,y;
    point operator-(const point &p)const{
        return {x-p.x,y-p.y};
    }
};

double mul(const point &a,const point &b){
    return a.x*b.y-a.y*b.x;
}

double cross(const point &a,const point &b,const point &c,const point &d){
    if(max(a.x,b.x)<min(c.x,d.x))return 0;
```

```
    if(max(a.y,b.y)<min(c.y,d.y))return 0;
    if(min(a.x,b.x)>max(c.x,d.x))return 0;
    if(min(a.y,b.y)>max(c.y,d.y))return 0;
    if(mul(c-a,b-a)*mul(b-a,d-a)<0)return 0;
    if(mul(a-c,d-c)*mul(d-c,b-c)<0)return 0;
    return 1;
}

int main(){
    return 0;
}
```

点和向量定义

```
#define LL long long
struct point {
    LL x,y;
    point operator+(const point &obj)const{
        return {x+obj.x,y+obj.y};
    }
    point operator-(const point &obj)const{
        return {x-obj.x,y-obj.y};
    }
    double norm(){
        return sqrt(x*x+y*y+0.0);
    }
    LL norm2(){
        return x*x+y*y;
    }
};

double dis(const point &a,const point &b){
    return (a-b).norm();
}

double det(const point &a,const point &b){
    return a.x*b.y-a.y*b.x;
}

double det(const point &o,const point &a,const point &b){
    return det(a-o,b-o);
}

double dot(const point &a,const point &b){
    return a.x*b.x+a.y*b.y;
}

double dot(const point &o,const point &a,const point &b){
    return dot(a-o,b-o);
}

double areaOfTriangle(const point &a,const point &b,const point &c){
    return fabs(det(a,b,c)/2);
```

```
}
```

线段

```
struct segment{
    point s,t;
};
```

点在线段上

```
bool isPointOnSegment(const point &o,const segment &l){
    double mix=min(l.s.x,l.t.x);
    double mxx=max(l.s.x,l.t.x);
    double miy=min(l.s.y,l.t.y);
    double mxy=max(l.s.y,l.t.y);
    return mix<=o.x&&o.x<=mxx&&miy<=o.y&&o.y<=mxy;
}
```

点线距

```
double dis(const point &o,const segment &l){
    return fabs(det(o,l.s,l.t)/dis(l.s,l.t));
}
```

反三角函数求角度

```
double angle(const point &o,const point &a,const point &b){
    return acos(1.0*dot(o,a,b)/dis(a,o)/dis(b,o));
}

double angle(const point &o,const point &a,const point &b){
    point da=a-o;
    point db=b-o;
    return fabs(atan2(1.0*da.y,1.0*da.x)-atan2(1.0*db.y,1.0*db.x));
}
```

扇形面积

```
double areaOfSector(const circle &c,const point &a,const point &b){
    return angle(c.cn,a,b)*c.r*c.r/2;
}
```

圆和线段交

```
polygon circleIntersectSegment(const circle &c,const segment &l){
    polygon ret;
    point a=l.s,b=l.t;
    if(a.x==b.x){
        double d=fabs(c.cn.x-a.x);
        if(d<c.r){
            double dy=sqrt(c.r*c.r-d*d);
            point p1={a.x,c.cn.y+dy};
            point p2={a.x,c.cn.y-dy};
            if(isPointOnSegment(p1,l))ret.push_back(p1);
            if(isPointOnSegment(p2,l))ret.push_back(p2);
        }
    }else{
        double k=(b.y-a.y)/(b.x-a.x);
        double bb=a.y-k*a.x;
        double x0=c.cn.x;
        double y0=c.cn.y;
        double A=k*k+1;
        double B=2*(k*(bb-y0)-x0);
        double C=x0*x0+(bb-y0)*(bb-y0)-c.r*c.r;
        double delta=B*B-4*A*C;
        if(delta>0){
            double t1=(-B+sqrt(delta))/2/A;
            double t2=(-B-sqrt(delta))/2/A;
            point p1={t1,k*t1+bb};
            point p2={t2,k*t2+bb};
            if(isPointOnSegment(p1,l))ret.push_back(p1);
            if(isPointOnSegment(p2,l))ret.push_back(p2);
        }
    }
    return ret;
}
```

简单多边形

```
typedef vector<point> polygon;
```

点在凸包内

```
bool isToLeft(const point &o,const segment &l){
    return det(l.s,l.t,o)>0;
}

bool isPointInConvexHull(const point &o,const polygon &p){
    bool tmp=1;
    for(int i=0;i<p.size()-1;++i){
        if(!isToLeft(o,(segment)(p[i],p[i+1])))tmp=0;
    }
    if(tmp)return 1;
```

```
    bool tmp=1;
    for(int i=0;i<p.size()-1;++i){
        if(!isToLeft(o,(segment)(p[i+1],p[i])))tmp=0;
    }
    if(tmp)return 1;
    return 0;
}
```

水平序凸包

```
bool cmp(point a,point b){
    return a.x<b.x||(a.x==b.x&&a.y<b.y);
}

polygon convexHull(polygon p){
    sort(p.begin(),p.end(),cmp);
    polygon ret;
    for(int i=0;i<p.size();++i){
        while(ret.size()>1&&det(*ret.rbegin(),*++ret.rbegin(),p[i])
<=0)ret.pop_back();
        ret.push_back(p[i]);
    }
    int m=ret.size();
    for(int i=p.size()-2;~i;--i){
        while(ret.size()>m&&det(*ret.rbegin(),*++ret.rbegin(),p[i])
<=0)ret.pop_back();
        ret.push_back(p[i]);
    }
    //此段代码中求出凸包，初始点会在末尾位置出现
    // ret.pop_back();
    return ret;
}
```

旋转卡壳

```
double rotateCalipers(polygon P){
    double ret=1e15+7;
    int p=1;
    for(int i=0;i<P.size()-1;++i){
        while(abs(det(P[i],P[i+1],P[p]))<abs(det(P[i],P[i+1],P[p+1]))){
            ++p;
            p%=P.size()-1;
        }
        double tmp=abs(det(P[i],P[i+1],P[p]));
        tmp/=dis(P[i],P[i+1]);
        ret=min(tmp,ret);
    }
    return ret;
}
```

三角剖分求面积

```
const point o={0.0,0.0};
double areaOfPolygon(const polygon &P){
    double ret=0;
    for(int i=0;i<P.size()-1;++i){
        ret+=det(o,P[i],P[i+1]);
    }
    return fabs(ret/2);
}
```

三角剖分求重心

```
const point o={0.0,0.0};
double areaOfPolygon(const polygon &P){
    double ret=0;
    for(int i=0;i<P.size()-1;++i){
        ret+=det(o,P[i],P[i+1]);
    }
    return fabs(ret/2);
}
```

三角剖分和圆面积交

```
double areaOfCircleIntersectSegment(const circle &c,point a,point b){
    double ret=0.0;
    if(dis(c.cn,a)>dis(c.cn,b))swap(a,b);
    if(dis(c.cn,b)<=c.r){
        ret=areaOfTriangle(c.cn,a,b);
    }else{
        polygon p=circleIntersectSegment(c,a,b);
        switch (p.size())
        {
        case 0:
            ret=areaOfSector(c,a,b);
            break;
        case 1:
            ret=areaOfTriangle(c.cn,a,p[0])+areaOfSector(c,p[0],b);
            break;
        case 2:
            if(dis(p[0],a)>dis(p[1],a))swap(p[0],p[1]);

 ret=areaOfTriangle(c.cn,p[0],p[1])+areaOfSector(c,a,p[0])+areaOfSector(c,b,p[
1]);
            break;
        }
    }
    return ret;
}
```

辛普森积分

```
double f(double x){
    return 1-x+x*x-x*x*x*x+x*x*x*x*x;//any arbitrary function
}

double F(double a,double b){
    return (f(a)+4*f((a+b)/2)+f(b))/6*(b-a);
}

double simpson(double a,double b,double eps){
    double m=(a+b)/2;
    double s=F(a,b),l=F(a,m),r=F(m,b);
    if(fabs(s-a-l)<eps)return s;
    return simpson(a,m,eps)+simpson(m,b,eps);
}
```

图论

KM 带权二分图匹配

```
/*
例题为luogu6577 [https://www.luogu.com.cn/problem/P6577]
二分图带权最大匹配（最大费用最大流）
复杂度点数n相关 O(n^3)

使用时调用
1. 输出km.n
2. 初始化边权e：要求完全匹配边权初始化为-inf，
   允许不完美匹配但是边权最大，初始化为0，下标从1开始
3. 调用KM.solve()

ps:
mb表示完美匹配下与右部第 i 个点相匹配的左部点的编号


*/

#include<bits/stdc++.h>
using namespace std;
#define int long long

const int maxn=505,inf=1e9+7;
struct KM{
    int n,m,e[maxn][maxn];
    int mb[maxn],vb[maxn],ka[maxn],kb[maxn],p[maxn],c[maxn];
    int qf,qb,q[maxn];
    void bfs(int u){
        int a,v=0,vl=0,d;
        for(int i=1;i<=n;i++) p[i]=0,c[i]=inf;
```

```
        mb[v]=u;
        do {
            a=mb[v],d=inf,vb[v]=1;
            for(int b=1;b<=n;b++)if(!vb[b]){
                if(c[b]>ka[a]+kb[b]-e[a][b])
                    c[b]=ka[a]+kb[b]-e[a][b],p[b]=v;
                if(c[b]<d) d=c[b],vl=b;
            }
            for(int b=0;b<=n;b++)
                if(vb[b]) ka[mb[b]]-=d,kb[b]+=d;
                else c[b]-=d;
            v=vl;
        } while(mb[v]);
        while(v) mb[v]=mb[p[v]],v=p[v];
    }
    int solve(){
        for(int i=1;i<=n;i++) mb[i]=ka[i]=kb[i]=0;
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n;j++) vb[j]=0;
            bfs(i);
        }
        int res=0;
        for(int i=1;i<=n;i++) res+=e[mb[i]][i];
        return res;
    }

}km;

/*usage as follows*/

int32_t main(){
    cin>>km.n>>km.m;
    for(int i=1;i<=km.n;++i){
        for(int j=1;j<=km.n;++j){
            km.e[i][j]=-inf;
        }
    }
    for(int i=1,x,y,z;i<=km.m;++i){
        cin>>x>>y>>z;
        km.e[x][y]=max(km.e[x][y],z);;
    }
    cout<<km.solve()<<endl;
    for(int i=1;i<=km.n;++i)cout<<km.mb[i]<<" ";
    cout<<endl;
}
```

最大流Dinic

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

struct flow{
```

```cpp
static const int maxn=5e5+5;
struct edge{int to,cap,flow,rev;};
vector<edge> g[maxn];
int s,t,dis[maxn],cur[maxn],n;
queue<int> q;

void init(int ss,int tt,int nn){
    s=ss,t=tt,n=nn;
    for(int i=1;i<=n;++i)g[i].clear();
}

void addedge(int u,int v,int w){
    g[u].push_back({v,w,0,g[v].size()});
    g[v].push_back({u,0,0,g[u].size()-1});
}

bool bfs(){
    for(int i=1;i<=n;++i){
        dis[i]=-1;
        cur[i]=0;
    }
    dis[s]=0;
    q.push(s);
    while(!q.empty()){
        int p=q.front();
        q.pop();
        for(auto e:g[p]){
            if(e.cap>e.flow&&dis[e.to]==-1){
                dis[e.to]=dis[p]+1;
                q.push(e.to);
            }
        }
    }
    return dis[t]!=-1;
}

int dfs(int p,int a){
    if(p==t)return a;
    int ans=0,now;
    for(int &i = cur[p];i<g[p].size();++i){
        edge &e = g[p][i];
        if(e.cap>e.flow&&dis[p]+1==dis[e.to]){
            now = dfs(e.to,min(a,e.cap-e.flow));
            e.flow += now;
            g[e.to][e.rev].flow -= now;
            ans += now;
            a -= now;
            if(!a) break;
        }
    }
    return ans;
}

int solve(){
    int ans=0;
```

```
        while(bfs())ans+=dfs(s,(int)1e9);
        return ans;
    }

}flow;


/*Usage Code below*/

int main(){
    int s=1,t=2,n=100;  //s,t must be >= 1
    flow.init(s,t,n);   //n must be the total number of nodes
    flow.addedge(s,t,1);//add edges (from,to,capacity)
    flow.solve()        //get answer at last
}
```

MCMF-SPFA

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

#define int long long

//DS
const int N=5005,M=10005,inf=1e9+7;
struct edge{int to,next,weight,cost;}e[M*2];
int tail[N],cnt=0;
void add_edge(int from,int to,int w,int c){
    e[cnt].to=to;
    e[cnt].weight=w;    //weight is capacity
    e[cnt].cost=c;      //cost is the cost per unit of flow
    e[cnt].next=tail[from];
    tail[from]=cnt;
    ++cnt;
}

//GLOBAL param.
int n,m,s,t;
int ansflow,anscost;

//SPFA
int pre[N],flow[N],track[N],dis[N],vis[N];
int spfa(){
    for(int i=1;i<=n;++i)pre[i]=0,vis[i]=0,dis[i]=inf;
    flow[s]=inf,pre[s]=s,vis[s]=1,dis[s]=0;
    queue<int> q;q.push(s);
    while(!q.empty()){
        int cur=q.front();q.pop();vis[cur]=0;
        for(int i=tail[cur];i;i=e[i].next){
            int nx=e[i].to;
            int w=e[i].weight;
            int c=e[i].cost;
            if(w>0&&dis[nx]>dis[cur]+c){
```

```
                dis[nx]=dis[cur]+c;
                pre[nx]=cur;
                flow[nx]=min(w,flow[cur]);
                track[nx]=i;
                if(!vis[nx]){
                    vis[nx]=1;
                    q.push(nx);
                }
            }
        }
    }
    return dis[t]!=inf;
}

//MCMF
void mcmf(){
    ansflow=0,anscost=0;
    while(spfa()){
        ansflow+=flow[t];
        anscost+=flow[t]*dis[t];
        int x=t;
        while(x!=s){
            e[track[x]].weight-=flow[t];
            e[track[x]^1].weight+=flow[t];
            x=pre[x];
        }
    }
    cout<<ansflow<<" "<<anscost<<endl;
}

void solve(){
    cin>>n>>m>>s>>t;
    cnt=0;
    for(int i=0;i<=n;++i)tail[i]=-1;
    for(int i=0,t1,t2,t3,t4;i<m;++i){
        cin>>t1>>t2>>t3>>t4;
        add_edge(t1,t2,t3,t4);
        add_edge(t2,t1,0,-t4);
    }
    mcmf();
}

int32_t main(){
    ios::sync_with_stdio(0);
    solve();
    solve();
}
```

三元环计数

```
/*
例题为luogu1989 [https://www.luogu.com.cn/problem/P1989]
无向图上三元环计数
复杂度为m*sqrt(m) 其中m为边数
*/

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

const int maxn=1e5+5;
struct TRIPLE{

    int n;//number of nodes
    vector<int> g[maxn];//graph
    vector<pair<int,int>> edge;//edge cache
    int deg[maxn];//num of degree
    int tag[maxn];//status cache for counting

    void init(int n){
        this->n=n;
        for(int i=1;i<=n;++i){
            g[i].clear();
            deg[i]=0;
            tag[i]=0;
        }
        edge.clear();
    }

    void addedge(int u,int v){
        edge.push_back({u,v});
        deg[u]++;
        deg[v]++;
    }

    void build(){
        for(auto e:edge){
            int u=e.first;
            int v=e.second;
            if(deg[u]>deg[v]||(deg[u]==deg[v]&&u>v)){
                g[u].push_back(v);
            }else{
                g[v].push_back(u);
            }
        }
    }

    int solve(){
        int ret=0;
        for(int u=1;u<=n;++u){
            for(auto v:g[u]){
                tag[v]=u;
            }
```

```
            for(auto v:g[u]){
                for(auto w:g[v]){
                    if(tag[w]==u){
                        ++ret;
                    }
                }
            }
        }
        return ret;
    }

}tri;

/*usage as follows*/

int main(){
    int n,m;
    cin>>n>>m;
    tri.init(n);//initiate with node number
    for(int i=1,u,v;i<=m;++i){
        cin>>u>>v;
        tri.addedge(u,v);//add edges
    }
    tri.build();//build the graph
    cout<<tri.solve()<<endl;//calculate the answer
}
```

线性代数

高斯消元求模数意义下行列式

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

const int maxn=505;
const int mod=1e9+7;

int n;//矩阵为n*n,范围[1,n]
int d[maxn][maxn];

int guass(){
    int res=0;//行列式的值
    for(int i=1;i<=n;i++){//枚举主对角线上第i个元素
        for(int j=i+1;j<=n;j++){//枚举剩下的行
            while(d[j][i]){//辗转相除
                int t=d[i][i]/d[j][i];
                for(int k=i;k<=n-1;k++)//转为倒三角
                    d[i][k]=((d[i][k]-t*d[j][k])%mod+mod)%mod;
                swap(d[i],d[j]);//交换i、j两行
                res=-res;//取负
            }
```

```
        }
        res=(res*d[i][i])%mod;
    }
    return res;
}
```

数论

min25求素数前缀和精简版

```
/*
计算1~n区间质数个数
速度和正常写法接近，但是大大降低了代码量
该版本N=sqrt(MAXN=1e11)，使用时根据上界调整，N取上界的sqrt即可
测试题目：Libre 6235 [https://loj.ac/problem/6235]
*/

#include<bits/stdc++.h>
using namespace std;
#define int long long

struct NumOfPrime{
    static const int N = 316300;
    int g[N<<1], a[N<<1];
    int id, cnt, n, sn, prime[N];
    inline int get(int x){ return x<=sn?x:id-n/x+1;}
    int solve(int x){
        n=x;sn=sqrt(n);
        cnt=id=0;
        for(int i=1; i<=n; i=a[id]+1) a[++id]=n/(n/i), g[id]=a[id]-1;
        for(int i=2; i<=sn; ++i){
            if(g[i]!=g[i-1]){
                prime[++cnt]=i;
                int sq=i*i;
                for(int j=id; a[j]>=sq; --j) g[j]-=g[get(a[j]/i)]-(cnt-1);
            }
        }
        return g[id];
    }
}np;


int n;
int32_t main() {
    while(cin>>n)cout<<np.solve(n)<<endl;
}
```

min25求素数前缀和

```
/*
计算1~n区间质数个数
使用经典min25筛计算公式
g(n,j) = g(n,j-1)-F(p_{j}) [g(n/p_{j},j-1)-g(p_{j-1},j-1)]
                    ......if p_{j}^2>n
       = g(n,j-1)
                    ......if p_{j}^2<=n
g(n,j)的物理意义是小于等于n的数里，在进行前j个质数的埃氏筛之后，剩下的所有数的F(*)和，
F要为完全积性函数
g(n,0)即将所有数带进F里求和
实际上我们要求的就是F(*)=1时，g(n,|P|)的值，其中|P|为n及以下的质数的个数
最后一项预处理所有小于等于sqrt(n)的pj的前缀和，可以递推求解
测试题目: Libre 6235 [https://loj.ac/problem/6235]
*/
#include<bits/stdc++.h>
using namespace std;
#define int long long

struct Sieve{
    static const int maxn=1e6+5;
    bool notp[maxn];
    vector<int> prime;
    void build(){
        for(int i=2;i<maxn;++i){
            if(!notp[i])prime.push_back(i);
            for(auto p:prime){
                if(i*p>=maxn)break;
                notp[i*p]=1;
                if(i%p==0)break;
            }
        }
    }
    int id1[maxn],id2[maxn];
    int v[maxn],f[maxn];
    #define getid(x) ((x) <= lim ? id1[(x)] : id2[n / (x)])
    int solve(int n){
        if(n<=1)return 0;
        int lim=sqrt(n),cnt=0;
        for(int i=1;i<=n;i=n/(n/i)+1)v[cnt++]=n/i;
        for(int i=0;i<cnt;++i)getid(v[i])=i;
        for(int i=0;i<cnt;++i)f[i]=v[i]-1;
        for(int i=0;prime[i]<=lim;++i){
            for(int j=0;j<cnt&&prime[i]*prime[i]<=v[j];++j){
                f[j]-=f[getid(v[j]/prime[i])]-i;
            }
        }
        return f[0];
    }
}sieve;

int32_t main(){
    ios::sync_with_stdio(0);
```

```
    cin.tie(0);cout.tie(0);

    sieve.build();
    int n;cin>>n;
    cout<<sieve.solve(n)<<endl;
}
```

min25注释版

```
/*
洛谷P3525,对积性函数求前缀和模板 [https://www.luogu.com.cn/problem/P5325]
min25筛
*/

#include<bits/stdc++.h>
using namespace std;
#define int long long
const int mod=1e9+7,inv2=(mod+1)/2,inv6=(mod+1)/6;
const int maxn=1e6+5;

struct SieveMin25{
    #define id(x) ((x)<=s?id1[(x)]:id2[n/(x)])
    int prime[maxn],num,sp1[maxn],sp2[maxn],vis[maxn];
    void build(){
        num=0;
        for(int i=2;i<maxn;++i){
            if(!vis[i]){
                prime[++num]=i;
                sp1[num]=(sp1[num-1]+i)%mod;
                sp2[num]=(sp2[num-1]+i*i)%mod;
            }
            for(int j=1;j<=num&&prime[j]*i<maxn;++j){
                vis[i*prime[j]]=1;
                if(i%prime[j]==0)break;
            }
        }
    }
    int n,s,tot,g1[maxn],g2[maxn],w[maxn],id1[maxn],id2[maxn];
    int S(int x,int y){
        if(prime[y]>=x)return 0;
        int k=id(x);
        int ans=((g2[k]-g1[k]-(sp2[y]-sp1[y]))%mod+mod)%mod;
        for(int i=y+1;i<=num&&prime[i]*prime[i]<=x;++i){
            int pe=prime[i];
            for (int e=1;pe<=x;++e,pe=pe*prime[i]){
                int xx=pe%mod;
                ans=(ans+xx*(xx-1)%mod*(S(x/pe,i)+(e!=1)))%mod;
            }
        }
        return ans%mod;
    }
    int solve(int x){
        tot=0;
```

```
        n=x,s=sqrt(n);
        for(int l=1,r;l<=n;l=r+1){//整除分块用于枚举所有因子
            r=n/(n/l);
            w[++tot]=n/l;
            int x=(n/l)%mod;
            g1[tot]=(x*(x+1)%mod*inv2%mod+mod-1)%mod;
            //分别计算在给g1[l]=g1(n/l,0),g2[l]=g2(n/l,0),
            //即将所有数视为质数求和,去掉1的位置
            g2[tot]=(x*(x+1)%mod*(2*x%mod+1)%mod*inv6%mod+mod-1)%mod;
            id(n/l)=tot;
        }
        for(int i=1;i<=num;++i){//枚举每一个素数
            for(int j=1;j<=tot&&prime[i]*prime[i]<=w[j];++j){
                int k=id(w[j]/prime[i]);
                g1[j]-=prime[i]*(g1[k]-sp1[i-1])%mod;//按照dp式迭代
                g2[j]-=prime[i]*prime[i]%mod*(g2[k]-sp2[i-1])%mod;
                g1[j]=(g1[j]%mod+mod)%mod;
                g2[j]=(g2[j]%mod+mod)%mod;
            }
        }
        return (S(n,0)+1)%mod;
    }
}sieve;

int32_t main(){
    int n;cin>>n;
    sieve.build();
    cout<<sieve.solve(n)<<endl;
}
```

杜教筛求mu/phi

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

#define int long long
const int maxn=1<<22;
int mu[maxn],phi[maxn];
unordered_map<int,int> mus,phis;

bool notp[maxn];
vector<int> primes;
void init(){
    mu[1]=phi[1]=1;
    for(int i=2;i<maxn;++i){
        if(!notp[i]){
            mu[i]=-1;
            phi[i]=i-1;
            primes.push_back(i);
        }
        for(auto p:primes){
            if(p*i>=maxn)break;
            notp[p*i]=1;
```

```
            if(i%p==0){
                mu[p*i]=0;
                phi[p*i]=phi[i]*p;
                break;
            }else{
                mu[p*i]=-mu[i];
                phi[p*i]=phi[i]*(p-1);
            }
        }
    }
    for(int i=1;i<maxn;++i){
        mu[i]+=mu[i-1];
        phi[i]+=phi[i-1];
    }
}

int getmu(int n){
    if(n<maxn)return mu[n];
    if(mus.find(n)!=mus.end())return mus[n];
    int res=1;
    for(int l=2,r,u;l<=n;l=r+1){
        r=n/(u=n/l);
        res-=(r-l+1)*getmu(u);
    }
    return mus[n]=res;
}

int getphi(int n){
    if(n<maxn)return phi[n];
    if(phis.find(n)!=phis.end())return phis[n];
    int res=n*(n+1)/2;
    for(int l=2,r,u;l<=n;l=r+1){
        r=n/(u=n/l);
        res-=(r-l+1)*getphi(u);
    }
    return phis[n]=res;
}

int32_t main(){
    init();
    int T;cin>>T;
    while(T--){
        int n;cin>>n;
        phis.clear();
        mus.clear();
        cout<<getphi(n)<<" "<<getmu(n)<<"\n";
    }
}
```

TeslaDeng Min25 老版 $O(\frac{n^{\frac{3}{4}}}{\ln n})$

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int maxn = 2000;
const int N = 710000;
const int mod = 1e9+7;

int b[maxn],c[maxn][maxn],Inv[maxn];
ll sqr,n; /// sqr为sqrt(n)
ll w[N],id1[N],id2[N];

int tot; ///记录对于要筛的n,sqrt(n)以内质数的个数
int isp[N],p[N];
ll zh[N][3]; ///zh[i][k]记录(p[1])^k + (p[2])^k + ... + (p[i])^k
ll g[N][3];


ll poww(ll a,int b)
{
    ll ans = 1,base = a%mod;
    while(b)
    {
        if(b&1)
        {
            ans*=base;
            ans%=mod;
        }
        base*=base;
        base%=mod;
        b>>=1;
    }
    return ans;
}


ll sigma_f(ll n,int k) ///得到∑i^k, i:1~n
{
    if(k==0) return n;
    n++;
    n%=mod;
    ll tmp = n;
    ll ans=0;
    for (int i=1;i<=k+1;i++)
    {
        ans += 1LL*c[k+1][i]*b[k+1-i]%mod*n%mod;
        ans %= mod;
        n *= tmp%mod;
        n %= mod;
    }
    ans *= Inv[k+1];
    ans %= mod;
```

```
    ans += mod;
    ans %= mod;
    return ans;
}



void get_p(int n,int w)
{
    tot = 0;
    memset(isp,1,sizeof(isp));
    isp[0] = 0;
    isp[1] = 0;
    for(int i=2;i<=n;i++)
    {
        if(isp[i])
        {
            p[++tot] = i;
            ll wait = 1;
            for(int j=0;j<=w;j++)
            {
                zh[tot][j] = zh[tot-1][j] + wait;
                zh[tot][j] %= mod;
                wait *= i;
            }
        }

        for(int j=1;p[j]*i<=n&&j<=i;j++)
        {
            isp[i*p[j]] = 0;
            if(i%p[j]==0) break;
        }
    }
}


void get_g(ll n,int t)
//对每个x=n/i,求出∑[i是质数](i^t) (i from 1 to x)。每个对应的值存储在g[x][t]中
{
    int m = 0;
    ll i=1,r;
    while(i<=n)
    {
        ll len = n/i;
        r = n/len;
        if(len<=sqr) id1[len] = ++m;
        else id2[r] = ++m;
        for(int ww=0;ww<=t;ww++)
        {
            g[m][ww] = sigma_f(len,ww)-1;
            g[m][ww] %= mod;
            g[m][ww] += mod;
            g[m][ww] %= mod;
        }
        w[m] = len; ///w[i]记录了形如n/k的第i大的取值是多少
```

```
        i = r+1;
    }


    for(int i=1;i<=tot;i++)
    {
        for(int j=1;j<=m;j++)
        {
            if(1LL*p[i]*p[i]>w[j]) break;
            else
            {
                int op;
                if(w[j]/p[i]<=sqr) op = id1[w[j]/p[i]];
                else op = id2[n/(w[j]/p[i])];
                for(int ww=0;ww<=t;ww++)
                {
                    g[j][ww] = g[j][ww] - poww(p[i],ww)*((g[op][ww]-zh[i-1]
[ww])%mod);
                    g[j][ww] %= mod;
                    g[j][ww] += mod;
                    g[j][ww] %= mod;
                }
            }
        }
    }
}

inline ll get_value(int wz,int k)
{
    ll w = (g[wz][2] + 2*g[wz][1] - g[wz][0]) -
           (zh[k-1][2] + 2*zh[k-1][1] - zh[k-1][0]);
    w %= mod;
    w += mod;
    w %= mod;
    //ll w = (g[wz][1]-g[wz][0])-(zh[k-1][1]-zh[k-1][0]);
    return w;
    //自己填写f(x)的表达式（在质数时）
    //比如f(x) = x^2 + 2*x - 1,
    //就写(g[wz][2] + 2*g[wz][1] - g[wz][0]) -
    //(zh[k-1][2] + 2*zh[k-1][1] - zh[k-1][0])
}

ll f(ll p,ll k) ///计算f(p^k)处的值
{
    if(k==1) return (p*p+2*p-1)%mod;
    return -3; ///自己填写
}




ll get_s(ll x,int k)
{
    if(x<=1||p[k]>x) return 0;
    int wz;
```

```
    if(x<=sqr) wz = id1[x];
    else wz = id2[n/x];
    ll ans = get_value(wz,k);
    //if(k==1) ans += 2;
    for(int i=k;i<=tot&&1LL*p[i]*p[i]<=x;++i)
    {
        for(ll l=p[i],e=1;l*p[i]<=x;l=l*p[i],++e)
        {
            ans = ans + (get_s(x/l,i+1)*f(p[i],e)%mod)%mod+f(p[i],e+1);
            ans %= mod;
        }
    }
    ans += mod;
    ans %= mod;
    return ans;

}

void init()
{
    c[0][0]=1;
    for (int i=1;i<maxn;i++)
    {
        for (int j=1;j<=i;j++) c[i][j]=(c[i-1][j-1]+c[i-1][j]) % mod;
        c[i][0]=1;
    }
    Inv[1]=1;
    for (int i=2;i<maxn;i++) Inv[i]=1LL*Inv[mod % i] * (mod-mod/i) % mod;
    b[0]=1;
    for (int i=1;i<maxn;i++)
    {
        b[i]=0;
        for (int k=0;k<i;k++) b[i]=(b[i]+1LL*c[i+1][k]*b[k] % mod) % mod;
        b[i]=(1LL*b[i]*(-Inv[i+1]) % mod+mod)%mod;
    }
}

void solve(ll n)
{
    init();
    sqr = sqrt(n);
    get_p(sqr,2);
    get_g(n,2);
    ll ans = get_s(n,1)+1;
    cout << ans << endl;
}

int main()
{
    while(cin >> n)
    {solve(n);}
}
```

TeslaDeng Min25 新版 $O(n^{\frac{2}{3}})$

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int maxn = 2000000+100;

/*****************************
f()函数中(31-37行) 填函数在质数幂次处的表达式
pow_sum()函数中(38-43行) 填幂和函数（如果需要更高次的话可以在这里添加）
202-205行按要求填写
f_p[][0/1/2/3/...]分别代表质数个数/质数和/质数平方和/质数三次方和/...根据自己需要添加
例：如果该函数在质数处表达式为f(p) = p^2+3*p+1，
    则表明需要质数个数/质数和/质数平方和，即f_p[][0],f_p[][1],f_p[][2]
*****************************/

ll poww(ll a,ll b){
    ll res = 1;
    ll base = a;
    while(b){
        if(b&1){
            res *= base;
            //res %= mod;
        }
        base *= base;
        //base %= mod;
        b>>=1;
    }
    return res;
}

inline ll f(ll p,int e){
    if(p==1||e==0) return 1;
    ///return f(p^e)
    ll res = poww(p,e);
    return res*res+3*res+1;

}
ll pow_sum(ll n,int k){
    ///return sum(i^k),i from 1 to n.
    if(k==0) return n;
    if(k==1) return n*(n+1)/2;
    if(k==2) return n*(n+1)*(2*n+1)/6;
}
ll f_p[maxn][3];///F_prime(id(n/i))
ll n;
int n_2; ///(int)sqrt(n)
int n_3; ///(int)pow(n,1.0/3.0)
int n_6; ///(int)pow(n,1.0/6.0)
ll val_id[maxn]; ///give the id, return the id-th number like 'n/i' ,
(val_id[1] = 1)
int val_id_num; ///how many numbers like 'n/i'
int val_id_num_3; ///how many numbers like 'n/i' below n/n_3;
```

```
int p[200000+100];
bool isp[maxn];
int p_sz_2; ///pi(n_2)
int p_sz_3; ///pi(n_3)
int p_sz_6; ///pi(n_6)
void init(){
    n_2 = (int)sqrt(n);
    n_3 = (int)pow(n,1.0/3.0);
    n_6 = (int)pow(n,1.0/6.0);
    val_id_num = 0;
    for(ll i=1;i<=n;){
        val_id[++val_id_num] = i;
        if(i==n) break;
        i = n/(n/(i+1));
    }
    memset(isp,1,sizeof isp);
    isp[1] = 0;
    for(int i=2;i<=n_2;i++){
        if(isp[i]){
            p[++p_sz_2] = i;
            if(i<=n_3) p_sz_3++;
            if(i<=n_6) p_sz_6++;
        }
        for(int j=1;j<=p_sz_2&&p[j]*i<=n_2;j++){
            isp[i*p[j]] = 0;
            if(i%p[j]==0) break;
        }
    }
}
inline int get_id(ll k){ ///give a number like 'n/i', return the id of it
    if(k>n_2) return val_id_num-n/k+1;
    else return k;
}
ll c[maxn];
int lowbit(int n){return n & (-n);}
void add(int x,ll d){
    while(x<maxn){
        c[x]+=d;
        x+=lowbit(x);
    }
}
ll sum(int x){
    ll ans=0;
    while(x){
        ans+=c[x];
        x-=lowbit(x);
    }
    return ans;
}

struct node{
    int k_max;
    ll val;
    ll f_val;
};
```

```
void update_bfs(int k,int type){
    queue<node> q;
    while(!q.empty()) q.pop();
    int e = 1;
    for(ll i=p[k];i<n/n_3;i*=p[k]){
        node st;
        st.k_max = k;
        st.val = i;
        if(type==-1)st.f_val = f(p[k],e);
        else st.f_val = poww(i,type);
        q.push(st);
        e++;
    }
    while(!q.empty()){
        node hd = q.front();
        q.pop();
        if((hd.val!=p[hd.k_max]&&type>=0)||type==-1) {
            //if(type==-1)cout << "****" << hd.val << "****" << hd.f_val <<
endl;
            ll w = n/hd.val;
            w = n/w;
            //cout << hd.val << "[" << w<<" , " << val_id[val_id_num] << "]"
<< endl;
            if(type==-1){
                add(get_id(w),hd.f_val);
                add(val_id_num+1,-1ll*hd.f_val);
            }
            else{
                add(get_id(w),-1ll*hd.f_val);
                add(val_id_num+1,hd.f_val);
            }
        }
        for(int i=hd.k_max+1;hd.val*p[i]<n/n_3&&i<=p_sz_2;i++){
            ll res = p[i];
            for(int e=1;;e++){
                if(hd.val*res<n/n_3){
                    node nxt;
                    nxt.k_max = i;
                    nxt.val = hd.val*res;
                    if(type==-1) nxt.f_val = hd.f_val*f(p[i],e);
                    else nxt.f_val = hd.f_val*poww(res,type);
                    q.push(nxt);
                }
                else break;
                res *= p[i];
            }
        }
    }
}
void get_f_p(ll n,int times){
    for(int i=1;i<=val_id_num;i++){
        for(int j=0;j<=times;j++){
            f_p[i][j] = pow_sum(val_id[i],j)-1;
        }
    }
```

```
    int now;
    //for(now=1;now<=p_sz_2;now++){
    for(now=1;p[now]<=n_6;now++){
        for(int j=val_id_num;j>=1;j--){
            ll w = val_id[j]/p[now];
            if(w<p[now]) break;
            ll val=1;
            for(int k = 0;k<=times;k++){
                f_p[j][k] = f_p[j][k] - val*(f_p[get_id(w)][k]-f_p[p[now-1]]
[k]);
                val *= p[now];
            }
        }
    }
    int nnow = now;
    int val = 1;
    for(int tt = 0;tt<=times;tt++){
        now = nnow;
        memset(c,0,sizeof c);
        add(1,f_p[1][tt]);
        for(int i=2;val_id[i]<n/n_3;i++){
            add(i,f_p[i][tt] - f_p[i-1][tt]);
        }
        for(;p[now]<=n_3;now++){
            for(int j=val_id_num;j>=1;j--){
                ll w = val_id[j]/p[now];
                if(val_id[j]<n/n_3) break;
                if(w<p[now]) break;
                if(w<n/n_3) f_p[j][tt] = f_p[j][tt] -
                    (sum(get_id(w)) - sum(p[now-1]))*poww(p[now],tt);
                else f_p[j][tt] = f_p[j][tt] -
                    (f_p[get_id(w)][tt]-sum(p[now-1]))*poww(p[now],tt);
            }
            update_bfs(now,tt);
        }
        for(int i=1;i<=val_id_num&&val_id[i]<n/n_3;i++)
            f_p[i][tt] = sum(i);
        for(;now<=p_sz_2;now++){
            for(int j=val_id_num;j>=1;j--){
                ll w = val_id[j]/p[now];
                if(val_id[j]<n/n_3) break;
                if(w<p[now]) break;
                f_p[j][tt] -= (f_p[get_id(w)][tt]-f_p[p[now-1]]
[tt])*poww(p[now],tt);
            }
        }
    }

    for(int i=1;i<=val_id_num;i++){
        ///if f(p) = p^2+3p+1,then write:f_p[i][0] =
        //f_p[i][2] + 3*f_p[i][1] + f_p[i][0];
        f_p[i][0] = f_p[i][2] + 3*f_p[i][1] + f_p[i][0];
    }

}
```

```
ll F[2000000+100];
void get_f_3(ll n){
    //V(F_{pi(n^(1/3))+1},n)
    ll q = p[p_sz_3+1];
    for(int now=1;now<=val_id_num;now++){
        if(val_id[now]<q){
            F[now] = 1;
        }
        else if(val_id[now]<q*q){
            F[now] = 1+(f_p[now][0]-f_p[q-1][0]);
        }
        else{
            F[now] = 1+(f_p[now][0]-f_p[q-1][0]);
            for(int pp=p_sz_3+1;p[pp]<=(int)
(sqrt(val_id[now]))&&pp<=p_sz_2;pp++){
                F[now] += f(p[pp],2) +
                (f(p[pp],1))*(f_p[get_id(val_id[now]/p[pp])][0]-
f_p[get_id(p[pp])][0]);
            }
        }
    }
}
void get_f_6(ll n){
    ///V(F_{pi(n^(1/6))+1},n)
    memset(c,0,sizeof c);
    add(1,F[1]);
    for(int i=2;val_id[i]<n/n_3;i++){
        add(i,F[i] - F[i-1]);
    }
    for(int k=p_sz_3;k>p_sz_6;k--){
        int now = val_id_num;
        for(;val_id[now]>=n/n_3;now--){
            int e = 1;
            ll _p = p[k];
            while(val_id[now]/_p){
                if(val_id[now]/_p>=n/n_3){
                    F[now] += F[get_id(val_id[now]/_p)]*f(p[k],e);
                }
                else{
                    F[now] += sum(get_id(val_id[now]/_p))*f(p[k],e);
                }
                _p *= p[k];
                e++;
            }
        }
        if(k==1) break;
        //cout << "******" << p[k] << "******" << n/n_3 << endl;
        update_bfs(k,-1);
        //bfs to update [lpf(i)==P{k-1}]f(i)
    }
    for(int i=1;i<=val_id_num&&val_id[i]<n/n_3;i++)
        F[i] = sum(i);
}
void get_f(ll n){
    for(int k=p_sz_6;k>=1;k--){
```

```
        for(int now = val_id_num;now>=1;now--){
            int e = 1;
            ll _p = p[k];
            while(val_id[now]/_p){
                    F[now] += F[get_id(val_id[now]/_p)]*f(p[k],e);
                _p *= p[k];
                e++;
            }
        }
    }
}
int main(){
    //n = 1000000000;
    //1e10:455052511,0.83s/0.58s 1e12:37607912018 9.224s/5.105s
    cin >> n;
    init();
    get_f_p(n,2);
    get_f_3(n);
    get_f_6(n);
    get_f(n);
    for(int i=1;i<=val_id_num;i++){
        cout << val_id[i] << " : " << F[i] << endl;
    }

}
```

BM求线性递推式

任何形如 $F_n = \sum_{i=1}^{n-1} F_i \times a_i$ 给定前几项后可以自动差出第 n 项，初始项数越多，答案越准。复杂度为线性。(带模)

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
#include <vector>
#include <string>
#include <map>
#include <set>
#include <cassert>
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
```

```
const ll mod=1000000007;
ll powmod(ll a,ll b) {
    ll res=1;
    a%=mod;
    assert(b>=0);
    for(;b;b>>=1){
        if(b&1)res=res*a%mod;
        a=a*a%mod;
    }
    return res;
}
// head

ll n;
namespace linear_seq {
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];

    vector<int> Md;
    void mul(ll *a,ll *b,int k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1;i>=k;i--) if (_c[i])
            rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-
_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
        ll ans=0,pnt=0;
        int k=SZ(a);
        assert(SZ(a)==SZ(b));
        rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (int p=pnt;p>=0;p--) {
            mul(res,res,k);
            if ((n>>p)&1) {
                for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
                rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s) {
        VI C(1,1),B(1,1);
        int L=0,m=1,b=1;
        rep(n,0,SZ(s)) {
            ll d=0;
            rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
            if (d==0) ++m;
```

```
            else if (2*L<=n) {
                VI T=C;
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L; B=T; b=d; m=1;
            } else {
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (SZ(C)<SZ(B)+m) C.pb(0);
                rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    int gao(VI a,ll n) {
        VI c=BM(a);
        c.erase(c.begin());
        rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
    }
};

int main() {
    /*push_back 进去前 8~10 项左右、最后调用 gao 得第 n 项*/
    vector<int>v;
    v.push_back(3);
    v.push_back(9);
    v.push_back(20);
    v.push_back(46);
    v.push_back(106);
    v.push_back(244);
    v.push_back(560);
    v.push_back(1286);
    v.push_back(2956);
    v.push_back(6794);
    int nCase;
    scanf("%d", &nCase);
    while(nCase--){
        scanf("%lld", &n);
        printf("%lld\n",1LL * linear_seq::gao(v,n-1) % mod); ///求第n项
    }
}
```

字符串算法

ac自动机

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

const int maxn=1e6+5;

struct Aho{
    int nxt[maxn][26],fail[maxn],val[maxn],tot;
    void init(){
        tot=0;
        memset(val,0,sizeof(val));
        memset(nxt,0,sizeof(nxt));
        memset(fail,0,sizeof(fail));
    }
    void insert(const string &s){
        int u=0;
        for(auto x:s){
            int i=x-'a';
            if(!nxt[u][i])nxt[u][i]=++tot;
            u=nxt[u][i];
        }
        val[u]++;//number of endings++
    }
    void build(){
        queue<int> q;
        q.push(0);
        while(!q.empty()){
            int u=q.front();
            q.pop();
            for(int i=0;i<26;++i){
                if(nxt[u][i]){
                    fail[nxt[u][i]]=u?nxt[fail[u]][i]:0;
                    q.push(nxt[u][i]);
                }else nxt[u][i]=nxt[fail[u]][i];
            }
        }
    }
    int getans(int u){
        int res=0;
        while(u&&~val[u]){
            res+=val[u];
            val[u]=-1;
            u=fail[u];
        }
        return res;
    }
    int match(const string &s){
        int u=0,res=0;
        for(auto x:s){
            int i=x-'a';
            u=nxt[u][i];
            res+=getans(u);
            //本质上是穷举结尾跳跃式寻找后缀，
```

```
            //在模式串高度重复时单次getans复杂度会退化到O(n)
        }
        return res;
    }
}aho;

int T,n;
string s;

int main(){
    ios::sync_with_stdio(0);
    T=1;
    while(T--){
        cin>>n;
        aho.init();
        while(n--){
            cin>>s;
            aho.insert(s);
        }
        aho.build();
        cin>>s;
        cout<<aho.match(s)<<endl;
    }
}
```

KMP

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

string s,t;
int nxt[1000005];
void get_next(string s,int nxt[]){
    int i=0,j=-1;
    nxt[0]=-1;
    while(i<(int)s.length()){
        if(j==-1||s[i]==s[j])nxt[++i]=++j;
        else j=nxt[j];
    }
}

void kmp(string t,string s,int nxt[]){
    int i=0,j=0;
    while(i<(int)t.length()){
        if(j==-1||t[i]==s[j])++i,++j;
        else j=nxt[j];
        if(j==(int)s.length())printf("%d\n",i-j+1),j=nxt[j];
    }
    //匹配首元素下标为: i-s.length()
}
int main(){
    cin>>t>>s;
    get_next(s,nxt);
```

```
    kmp(t,s,nxt);
    for(int i=1;i<=(int)s.length();++i)printf("%d ",nxt[i]);
    cout<<endl;
    return 0;
}
```

最小表示法

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

const int maxn=1e6+5;
string s;
int nxt[maxn];
int i,j,k,t,n;

int minRP(){
    i=0,j=1;
    while(i<n&&j<n){
        k=0;
        while(s[(i+k)%n]==s[(j+k)%n]&&k<n)++k;
        if(k==n)break;
        if(s[(i+k)%n]>s[(j+k)%n]){
            i=i+k+1;
        }else{
            j=j+k+1;
        }
        if(i==j)++j;
    }
    return min(i,j);
}

int maxRP(){
    i=0,j=1;
    while(i<n&&j<n){
        k=0;
        while(s[(i+k)%n]==s[(j+k)%n]&&k<n)++k;
        if(k==n)break;
        if(s[(i+k)%n]<s[(j+k)%n]){
            i=i+k+1;
        }else{
            j=j+k+1;
        }
        if(i==j)++j;
    }
    return min(i,j);
}

int main(){
    ios::sync_with_stdio(false);
    while(cin>>s){
        n=s.size();
        i=0,j=-1,nxt[0]=-1;
```

```
        while(i<n){
            if(j==-1||s[i]==s[j])nxt[++i]=++j;
            else j=nxt[j];
        }
        int len=(n-nxt[n]);
        if(n%len==0)len=n/len;
        else len=1;
        cout<<minRP()+1<<' '<<len<<' ';
        cout<<maxRP()+1<<' '<<len<<endl;
    }
}
```

回文自动机

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

/*Definition Starts*/

const int N=1e6+5,M=26;
struct PAM{
    int nxt[N][M],fail[N];
    int len[N],s[N];
    int n,tot,last;

    int addnode(int l){
        memset(nxt[tot],0,sizeof(nxt[tot]));
        len[tot]=l;
        return tot++;
    }

    void init(){
        n=tot=last=0;
        addnode(0);
        addnode(-1);
        fail[0]=1;
        s[0]=-1;
    }

    int getfail(int x){
        while(s[n-len[x]-1]!=s[n])x=fail[x];
        return x;
    }

    void insert(int i){
        s[++n]=i;
        int u=getfail(last);
        if(!nxt[u][i]){
            int v=addnode(len[u]+2);
            fail[v]=nxt[getfail(fail[u])][i];
            nxt[u][i]=v;
        }
        last=nxt[u][i];
```

```
    }

    void build(string &s){
        init();
        for(auto x:s)insert(x-'a');
    }

}pam;

/*Definition Ends*/

int main(){
    string s;
    cin>>s;
    pam.build(s);
}
```

后缀数组 桶排

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

const int maxn=1e6+5;
int s[maxn];
char tmp[maxn];

struct SA{
    //KINDLY REMINDS: multiple usage requires MEMSET(oldrk) and MEMSET(rk) !!!
    //KINDLY REMINS: MORE STRINGS REQUIRES BIGGER m!!!!!!!!!!!!!!!!!!!!!!!!!
    int sa[maxn],rk[maxn],h[maxn];
    int oldrk[maxn<<1],cnt[maxn],id[maxn],px[maxn];
    int n,m;
    bool cmp(int x,int y,int w){
        return oldrk[x]==oldrk[y]&&oldrk[x+w]==oldrk[y+w];
    }
    void build(int s[],int n){
        m=500;
        this->n=n;
        //init
        memset(cnt,0,sizeof(cnt));
        for(int i=1;i<=n;++i)++cnt[rk[i]=s[i]];
        for(int i=1;i<=m;++i)cnt[i]+=cnt[i-1];
        for(int i=n;i;--i)sa[cnt[rk[i]]--]=i;
        //calculate rk[]&sa[]
        for(int w=1,p=0;w<n;w<<=1,m=p){
            p=0;
            for(int i=n;i>n-w;--i)id[++p]=i;
            for(int i=1;i<=n;++i)if(sa[i]>w)id[++p]=sa[i]-w;
            memset(cnt,0,sizeof(cnt));
            for(int i=1;i<=n;++i)++cnt[px[i]=rk[id[i]]];
            for(int i=1;i<=m;++i)cnt[i]+=cnt[i-1];
            for(int i=n;i;--i)sa[cnt[px[i]]--]=id[i];
            memcpy(oldrk,rk,sizeof(rk));
```

```
            p=0;
            for(int i=1;i<=n;++i){
                rk[sa[i]]=cmp(sa[i],sa[i-1],w)?p:++p;
            }
        }
        //calculate h[]
        for(int i=1,k=0;i<=n;++i){
            if(k)--k;
            while(s[i+k]==s[sa[rk[i]-1]+k])++k;
            h[rk[i]]=k;
        }
    }
    void solve(){
        for(int i=1;i<=n;++i)cout<<sa[i]<<" ";
        // for(int i=1;i<=n;++i){
        //     for(int j=sa[i];j<=n;++j)cout<<(char)s[j];
        //     cout<<" "<<h[i]<<endl;
        // }
    }
}sa;

int main(){
    ios::sync_with_stdio(0);
    cin>>(tmp+1);
    int len=strlen(tmp+1);
    for(int i=1;i<=len;++i)s[i]=tmp[i];
    sa.build(s,len);
    sa.solve();
}
```

后缀数组 Qsort

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

const int maxn=1e6+5;
/*SuffixArrayWithQSort: INDEX START BY 1*/

int sa[maxn], rk[maxn << 1], oldrk[maxn << 1], height[maxn];
void get_sa(int *s,int n)
{
    for (int i = 1; i <= n; i++) rk[i] = s[i];
    for (int t = 1; t <= n; t *= 2)
    {
        iota(sa+1, sa+1+n, 1);
        memcpy(oldrk, rk, (n+1)*sizeof(int));
        sort(sa + 1, sa + 1 + n, [&t](int x, int y) { return rk[x] == rk[y] ?
rk[x + t] < rk[y + t] : rk[x] < rk[y];});
        for (int p = 0, i = 1; i <= n; i++)
        {
            if (oldrk[sa[i]] == oldrk[sa[i - 1]] and oldrk[sa[i] + t] ==
oldrk[sa[i - 1] + t])
                rk[sa[i]] = p;
```

```
            else
                rk[sa[i]] = ++p;
        }
    }
}

int p[maxn],n;
string s;
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin>>s;
    n=s.size();
    for(int i=0;i<n;++i)p[i+1]=s[i];
    get_sa(p,n);
    for(int i=1;i<=n;++i)cout<<sa[i]<<' ';
}
```

多项式

FFT

```
#include<bits/stdc++.h>
using namespace std;
#define cp complex<double>
const double PI=acos(-1.0);
const int N=1005;
int n=1,res[N];
string s1,s2;

void init(cp *omg,cp *inv){
    for(int i=0;i<N;++i){
        omg[i]=cp(cos(2*PI*i/n),sin(2*PI*i/n));
        inv[i]=conj(omg[i]);
    }
}
void fft(cp *a,cp *omg){
    int lim=0;
    while((1<<lim)<n)++lim;
    for(int i=0;i<n;++i){
        int t=0;
        for(int j=0;j<lim;++j)if((i>>j)&1)t|=(1<<(lim-j-1));
        if(i<t)swap(a[i],a[t]);
    }
    for(int l=2;l<=n;l<<=1){
        int m=l/2;
        for(cp *p=a;p!=a+n;p+=l){
            for(int i=0;i<m;++i){
                cp t=omg[n/l*i]*p[i+m];
                p[i+m]=p[i]-t;
```

```
                p[i]+=t;
            }
        }
    }
}

int main(){
    while(cin>>s1>>s2){
        cp a[N],b[N],omg[N],inv[N];
        memset(res,0,sizeof(res));
        int l1=s1.length(),l2=s2.length();
        n=1;
        while(n<l1+l2)n<<=1;
        for(int i=0;i<l1;++i)a[i].real(s1[l1-i-1]-'0');
        for(int i=0;i<l2;++i)b[i].real(s2[l2-i-1]-'0');
        init(omg,inv);
        fft(a,omg);
        fft(b,omg);
        for(int i=0;i<n;++i)a[i]*=b[i];
        fft(a,inv);
        for(int i=0;i<n;++i){
            res[i]+=floor(a[i].real()/n+0.5);
            res[i+1]+=res[i]/10;
            res[i]%=10;
        }
        int flag=0;
        for(int i=n;~i;--i){
            if(flag||res[i]){
                flag=1;
                cout<<res[i];
            }
        }
        if(flag==0)cout<<0;
        cout<<endl;
    }
}
```