代码库

Spear of Longinus

2016 年 11 月 10 日

目录

1 数论

1.1 快速求逆元

返回结果：

$$x^{-1}(mod)$$

使用条件：$x \in [0, mod)$ 并且 x 与 mod 互质

```
LL inv(LL a,LL p){
        LL d,x,y;
        d=exgcd(a,p,x,y);
        return d==1?(x+p)%p:-1;
}
```

1.2 扩展欧几里德算法

返回结果：

$$ax + by = gcd(a, b)$$

时间复杂度：$\mathcal{O}(nlogn)$

```
LL exgcd(LL a,LL b,LL &x,LL &y){
        if(!b){
                x=1;y=0;return a;
        }else{
                LL d=exgcd(b,a%b,x,y);
                LL t=x;x=y;y=t-a/b*y;
                return d;
        }
}
```

1.3 中国剩余定理

返回结果：

$$x \equiv r_i (mod\ \ p_i)\ \ (0 \leq i < n)$$

使用条件：p_i 需两两互质

```
LL china(int n,int *a,int *m){
        LL M=1,d,x=0,y;
        for(int i=0;i<n;i++)
                M*=m[i];
```

```
        for(int i=0;i<n;i++){
                LL w=M/m[i];
                d=exgcd(m[i],w,d,y);
                y=(y%M+M)%M;
                x=(x+y*w%M*a[i])%M;
        }
        while(x<0)x+=M;
        return x;
}
```

1.4 中国剩余定理 2

```
//merge Ax=B and ax=b to A'x=B'
void merge(LL &A,LL &B,LL a,LL b){
    LL x,y;
    sol(A,-a,b-B,x,y);
    A=lcm(A,a);
    B=(a*y+b)%A;
    B=(B+A)%A;
}
```

1.5 组合数取模

```
LL prod=1,P;
pair<LL,LL> comput(LL n,LL p,LL k){
    if(n<=1)return make_pair(0,1);
    LL ans=1,cnt=0;
    ans=pow(prod,n/P,P);
    cnt=n/p;
    pair<LL,LL>res=comput(n/p,p,k);
    cnt+=res.first;
    ans=ans*res.second%P;
    for(int i=n-n%P+1;i<=n;i++)if(i%p){

        ans=ans*i%P;
    }
    return make_pair(cnt,ans);
}
pair<LL,LL> calc(LL n,LL p,LL k){
    prod=1;P=pow(p,k,1e18);
    for(int i=1;i<P;i++)if(i%p)prod=prod*i%P;
```

```
    pair<LL,LL> res=comput(n,p,k);
//  res.second=res.second*pow(p,res.first%k,P)%P;
//  res.first-=res.first%k;
    return res;
}
LL calc(LL n,LL m,LL p,LL k){
    pair<LL,LL>A,B,C;
    LL P=pow(p,k,1e18);
    A=calc(n,p,k);
    B=calc(m,p,k);
    C=calc(n-m,p,k);
    LL ans=1;
    ans=pow(p,A.first-B.first-C.first,P);
    ans=ans*A.second%P*inv(B.second,P)%P*inv(C.second,P)%P;
    return ans;
}
```

1.6 扩展小步大步

```
LL solve2(LL a,LL b,LL p){
    //a^x=b (mod p)
    b%=p;
    LL e=1%p;
    for(int i=0;i<100;i++){
        if(e==b)return i;
        e=e*a%p;
    }
    int r=0;
    while(gcd(a,p)!=1){
        LL d=gcd(a,p);
        if(b%d)return -1;
        p/=d;b/=d;b=b*inv(a/d,p);
        r++;
    }LL res=BSGS(a,b,p);
    if(res==-1)return -1;
    return res+r;
}
```

1.7 卢卡斯定理

```
LL Lucas(LL n,LL m,LL p){
        LL ans=1;
        while(n&&m){
                LL a=n%p,b=m%p;
                if(a<b)return 0;
                ans=(ans*C(a,b,p))%p;
                n/=p;m/=p;
        }return ans%p;
}
```

1.8 小步大步

返回结果：

$$a^x = b \ \ (mod \ \ p)$$

使用条件：p 为质数
时间复杂度：$\mathcal{O}(\sqrt{n})$

```
LL BSGS(LL a,LL b,LL p){
        LL m=sqrt(p)+.5,v=inv(pw(a,m,p),p),e=1;
        map<LL,LL>hash;hash[1]=0;
        for(int i=1;i<m;i++)
                e=e*a%p,hash[e]=i;
        for(int i=0;i<=m;i++){
                if(hash.count(b))return i*m+hash[b];
                b=b*v%p;
        }return -1;
}
```

1.9 Miller Rabin 素数测试

```
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool check(long long n,int base) {
    long long n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=pw(base,n2,n);
    if((res==1)||(res==n-1)) return 1;
    while(s--) {
        res=mul(res,res,n);
        if(res==n-1) return 1;
    }
```

```
    return 0; // n is not a strong pseudo prime
}
bool isprime(const long long &n) {
    if(n==2)
        return true;
    if(n<2 || n%2==0)
        return false;
    for(int i=0;i<12&&BASE[i]<n;i++){
        if(!check(n,BASE[i]))
            return false;
    }
    return true;
}
```

1.10 Pollard Rho 大数分解

时间复杂度：$\mathcal{O}(n^{1/4})$

```
LL prho(LL n,LL c){
        LL i=1,k=2,x=rand()%(n-1)+1,y=x;
        while(1){
                i++;x=(x*x%n+c)%n;
                LL d=__gcd((y-x+n)%n,n);
                if(d>1&&d<n)return d;
                if(y==x)return n;
                if(i==k)y=x,k<<=1;
        }
}
void factor(LL n,vector<LL>&fat){
        if(n==1)return;
        if(isprime(n)){
                fat.push_back(n);
                return;
        }LL p=n;
        while(p>=n)p=prho(p,rand()%(n-1)+1);
        factor(p,fat);
        factor(n/p,fat);
}
```

1.11 快速数论变换 (zky)

返回结果：

$$c_i = \sum_{0 \le j \le i} a_j \cdot b_{i-j} (mod) \ \ (0 \le i < n)$$

使用说明：$magic$ 是 mod 的原根

时间复杂度：$\mathcal{O}(nlogn)$

```
/*
{(mod,G)}={(81788929,7),(101711873,3),(167772161,3)
                        ,(377487361,7),(998244353,3),(1224736769,3)
                        ,(1300234241,3),(1484783617,5)}
*/
int mo=998244353,G=3;
void NTT(int a[],int n,int f){
        for(register int i=0;i<n;i++)
                if(i<rev[i])
                        swap(a[i],a[rev[i]]);
        for (register int i=2;i<=n;i<<=1){
                static int exp[maxn];
                exp[0]=1;exp[1]=pw(G,(mo-1)/i);
                if(f==-1)exp[1]=pw(exp[1],mo-2);
                for(register int k=2;k<(i>>1);k++)
                        exp[k]=1LL*exp[k-1]*exp[1]%mo;
                for(register int j=0;j<n;j+=i){
                        for(register int k=0;k<(i>>1);k++){
                                register int &pA=a[j+k],&pB=a[j+k+(i>>1)];
                                register int A=pA,B=1LL*pB*exp[k]%mo;
                                pA=(A+B)%mo;
                                pB=(A-B+mo)%mo;
                        }
                }
        }
        if(f==-1){
                int rv=pw(n,mo-2)%mo;
                for(int i=0;i<n;i++)
                        a[i]=1LL*a[i]*rv%mo;
        }
}
void mul(int m,int a[],int b[],int c[]){
        int n=1,len=0;
        while(n<m)n<<=1,len++;
        for (int i=1;i<n;i++)
```

```
                rev[i]=(rev[i>>1]>>1)|((i&1)<<(len-1));
        NTT(a,n,1);
        NTT(b,n,1);
        for(int i=0;i<n;i++)
                c[i]=1LL*a[i]*b[i]%mo;
        NTT(c,n,-1);
}
```

1.12 快速数论变换 (lyx)

```
int Pow(int x,int y,int z){
        if (y==0) return 1;
        LL ret=Pow(x,y>>1,z); (ret*=ret)%=z;
        if (y & 1) (ret*=x)%=z;
        return ret;
}


void Prep(){
        for (len=1, ci=0; len<=N+N; len<<=1, ci++);
        Wi[0]=1, Wi[1]=Pow(G,(Mo-1)/len,Mo);
        for (int i=2; i<=len; i++) Wi[i]=(Wi[i-1]*Wi[1])% Mo;
        for (int i=0; i<len; i++){
                int tmp=0;
                for (int j=i, c=0; c<ci; c++, j>>=1 ) tmp=(tmp <<= 1)|= (j & 1);
                Bel[i]=tmp;
        }
}

void Dft(Arr &a,int sig)
{
        for (int i=0; i<len; i++) tp[Bel[i]]=a[i];
        for (int m=1; m<=len; m<<=1){
                int half=m>>1, bei=len/m;
                for (int i=0; i<half; i++){
                        LL wi=(sig>0)?Wi[i*bei]:Wi[len-i*bei];
                        for (int j=i; j<len; j+=m){
                                int u=tp[j],v=wi*LL(tp[j+half]) % Mo;
                                tp[j]=(u+v) % Mo; tp[j+half]=(u-v+Mo)% Mo;
                        }
                }
        }
```

```
        for (int i=0; i<len; i++) a[i]=tp[i];
}

void Mul(Arr &x,Arr &y,Arr &c,bool same)
{
        if (!same){
                for(int i=0; i<len; i++) a[i]=x[i], b[i]=y[i];
                Dft(a,1),Dft(b,1);
                for(int i=0; i<len; i++) a[i]=a[i]*1ll*b[i] % Mo;
                Dft(a,-1);
                for(int i=0; i<=M; i++) c[i]=a[i]*1ll*Rev % Mo;
        } else
        {
                for(int i=0; i<len; i++) a[i]=x[i];
                Dft(a,1);
                for(int i=0; i<len; i++) a[i]=a[i]*1ll*a[i] % Mo;
                Dft(a,-1);
                for(int i=0; i<=M; i++) c[i]=a[i]*1ll*Rev % Mo;
        }
}

Prep();
Ans[0]=1; Rev=Pow(len,Mo-2,Mo);
for(; K; K>>=1){
        if (K & 1) Mul(Ans,F,Ans,0);
        if (K > 1) Mul(F,F,F,1);
}

printf("%d\n",Ans[M]);
```

1.13 原根

```
vector<LL>fct;
bool check(LL x,LL g){
        for(int i=0;i<fct.size();i++)
                if(pw(g,(x-1)/fct[i],x)==1)
                        return 0;
        return 1;
}
LL findrt(LL x){
        LL tmp=x-1;
        for(int i=2;i*i<=tmp;i++){
```

```
                if(tmp%i==0){
                        fct.push_back(i);
                        while(tmp%i==0)tmp/=i;
                }
        }if(tmp>1)fct.push_back(tmp);
        // x is 1,2,4,p^n,2p^n
        // x has phi(phi(x)) primitive roots
        for(int i=2;i<int(1e9);i++)if(check(x,i))
                return i;
        return -1;
}
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool check(long long n,int base) {
    long long n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=pw(base,n2,n);
    if((res==1)||(res==n-1)) return 1;
    while(s--) {
        res=mul(res,res,n);
        if(res==n-1) return 1;
    }
    return 0; // n is not a strong pseudo prime
}
bool isprime(const long long &n) {
    if(n==2)
        return true;
    if(n<2 || n%2==0)
        return false;
    for(int i=0;i<12&&BASE[i]<n;i++){
        if(!check(n,BASE[i]))
            return false;
    }
    return true;
}
```

1.14 线性递推

```
//已知 a_0,a_1,...,a_{m-1}\\
        a_n = c_0 * a_{n-m} + ... + c_{m-1} * a_{n-1}\\
        求 a_n = v_0 * a_0 + v_1 * a_1 + ... + v_{m-1} * a_{m-1}\\

```

```cpp
void linear_recurrence(long long n, int m, int a[], int c[], int p) {
	long long v[M] = {1 % p}, u[M << 1], msk = !!n;
	for(long long i(n); i > 1; i >>= 1) {
		msk <<= 1;
	}
	for(long long x(0); msk; msk >>= 1, x <<= 1) {
		fill_n(u, m << 1, 0);
		int b(!!(n & msk));
		x |= b;
		if(x < m) {
			u[x] = 1 % p;
		}else {
			for(int i(0); i < m; i++) {
				for(int j(0), t(i + b); j < m; j++, t++) {
					u[t] = (u[t] + v[i] * v[j]) % p;
				}
			}
			for(int i((m << 1) - 1); i >= m; i--) {
				for(int j(0), t(i - m); j < m; j++, t++) {
					u[t] = (u[t] + c[j] * u[i]) % p;
				}
			}
		}
		copy(u, u + m, v);
	}
	//a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
	for(int i(m); i < 2 * m; i++) {
		a[i] = 0;
		for(int j(0); j < m; j++) {
			a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
		}
	}
	for(int j(0); j < m; j++) {
		b[j] = 0;
		for(int i(0); i < m; i++) {
			b[j] = (b[j] + v[i] * a[i + j]) % p;
		}
	}
	for(int j(0); j < m; j++) {
		a[j] = b[j];
	}
}
```

1.15 线性筛

```
void sieve(){
        f[1]=mu[1]=phi[1]=1;
        for(int i=2;i<maxn;i++){
                if(!minp[i]){
                        minp[i]=i;
                        minpw[i]=i;
                        mu[i]=-1;
                        phi[i]=i-1;
                        f[i]=i-1;
                        p[++p[0]]=i;//Case 1 prime
                }
                for(int j=1;j<=p[0]&&(LL)i*p[j]<maxn;j++){
                        minp[i*p[j]]=p[j];
                        if(i%p[j]==0){
                                //Case 2 not coprime
                                minpw[i*p[j]]=minpw[i]*p[j];
                                phi[i*p[j]]=phi[i]*p[j];
                                mu[i*p[j]]=0;
                                if(i==minpw[i]){
                                        f[i*p[j]]=i*p[j]-i;//Special Case for f(p^k)
                                }else{
                                        f[i*p[j]]=f[i/minpw[i]]*f[minpw[i]*p[j]];
                                }
                                break;
                        }else{
                                //Case 3 coprime
                                minpw[i*p[j]]=p[j];
                                f[i*p[j]]=f[i]*f[p[j]];
                                phi[i*p[j]]=phi[i]*(p[j]-1);
                                mu[i*p[j]]=-mu[i];
                        }
                }
        }
}
```

1.16 直线下整点个数

返回结果：

$$\sum_{0\le i<n}\lfloor\frac{a+b\cdot i}{m}\rfloor$$

使用条件：$n,m>0$，$a,b\geq 0$
时间复杂度：$\mathcal{O}(nlogn)$

```
//calc \sum_{i=0}^{n-1} [(a+bi)/m]
// n,a,b,m >0
LL solve(LL n,LL a,LL b,LL m){
    if(b==0)
        return n*(a/m);
    if(a>=m || b>=m)
        return n*(a/m)+(n-1)*n/2*(b/m)+solve(n,a%m,b%m,m);
    return solve((a+b*n)/m,(a+b*n)%m,m,b);
}
```

2 数值

2.1 高斯消元

```
void Gauss(){
        int r,k;
        for(int i=0;i<n;i++){
                r=i;
                for(int j=i+1;j<n;j++)
                        if(fabs(A[j][i])>fabs(A[r][i]))r=j;
                if(r!=i)for(int j=0;j<=n;j++)swap(A[i][j],A[r][j]);
                for(int k=i+1;k<n;k++){
                        double f=A[k][i]/A[i][i];
                        for(int j=i;j<=n;j++)A[k][j]-=f*A[i][j];
                }
        }
        for(int i=n-1;i>=0;i--){
                for(int j=i+1;j<n;j++)
                        A[i][n]-=A[j][n]*A[i][j];
                A[i][n]/=A[i][i];
        }
        for(int i=0;i<n-1;i++)
                cout<<fixed<<setprecision(3)<<A[i][n]<<" ";
        cout<<fixed<<setprecision(3)<<A[n-1][n];
}
bool Gauss(){
        for(int i=1;i<=n;i++){
                int r=0;
                for(int j=i;j<=m;j++)
```

```
                if(a[j][i]){r=j;break;}
                if(!r)return 0;
                ans=max(ans,r);
                swap(a[i],a[r]);
                for(int j=i+1;j<=m;j++)
                if(a[j][i])a[j]^=a[i];
        }for(int i=n;i>=1;i--){
                for(int j=i+1;j<=n;j++)if(a[i][j])
                a[i][n+1]=a[i][n+1]^a[j][n+1];
        }return 1;
}
LL Gauss(){
        for(int i=0;i<n;i++)for(int j=0;j<n;j++)A[i][j]%=m;
        for(int i=0;i<n;i++)for(int j=0;j<n;j++)A[i][j]=(A[i][j]+m)%m;
        LL ans=n%2?-1:1;
        for(int i=0;i<n;i++){
                for(int j=i+1;j<n;j++){
                        while(A[j][i]){
                                LL t=A[i][i]/A[j][i];
                                for(int k=0;k<n;k++)
                                A[i][k]=(A[i][k]-A[j][k]*t%m+m)%m;
                                swap(A[i],A[j]);
                                ans=-ans;
                        }
                }ans=ans*A[i][i]%m;
        }return (ans%m+m)%m;
}
int Gauss(){//求秩
        int r,now=-1;
        int ans=0;
        for(int i = 0; i <n; i++){
                r = now + 1;
                for(int j = now + 1; j < m; j++)
                        if(fabs(A[j][i]) > fabs(A[r][i]))
                                r = j;
                if (!sgn(A[r][i])) continue;
                ans++;
                now++;
                if(r != now)
                        for(int j = 0; j < n; j++)
                                swap(A[r][j], A[now][j]);

```

```
                for(int k = now + 1; k < m; k++){
                        double t = A[k][i] / A[now][i];
                        for(int j = 0; j < n; j++){
                                A[k][j] -= t * A[now][j];
                        }
                }
        }
        return ans;
}
```

2.2 快速傅立叶变换

返回结果：

$$c_i = \sum_{0 \le j \le i} a_j \cdot b_{i-j} \ \ (0 \le i < n)$$

时间复杂度：$\mathcal{O}(nlogn)$

```
typedef complex<double> cp;
const double pi = acos(-1);
void FFT(vector<cp>&num,int len,int ty){
    for(int i=1,j=0;i<len-1;i++){
        for(int k=len;j^=k>>=1,~j&k;);
        if(i<j)
            swap(num[i],num[j]);
    }
    for(int h=0;(1<<h)<len;h++){
        int step=1<<h,step2=step<<1;
        cp w0(cos(2.0*pi/step2),ty*sin(2.0*pi/step2));
        for(int i=0;i<len;i+=step2){
            cp w(1,0);
            for(int j=0;j<step;j++){
                cp &x=num[i+j+step];
                cp &y=num[i+j];
                cp d=w*x;
                x=y-d;
                y=y+d;
                w=w*w0;
            }
        }
    }
    if(ty==-1)
        for(int i=0;i<len;i++)
            num[i]=cp(num[i].real()/(double)len,num[i].imag());
```

```
}
vector<cp> mul(vector<cp>a,vector<cp>b){
    int len=a.size()+b.size();
    while((len&-len)!=len)len++;
    while(a.size()<len)a.push_back(cp(0,0));
    while(b.size()<len)b.push_back(cp(0,0));
    FFT(a,len,1);
    FFT(b,len,1);
    vector<cp>ans(len);
    for(int i=0;i<len;i++)
        ans[i]=a[i]*b[i];
    FFT(ans,len,-1);
    return ans;
}
```

2.3 单纯形法求解线性规划

返回结果：

$$max\{c_{1\times m}\cdot x_{m\times 1}\mid x_{m\times 1}\geq 0_{m\times 1}, a_{n\times m}\cdot x_{m\times 1}\leq b_{n\times 1}\}$$

```
namespace LP{
        const int maxn=233;
        double a[maxn][maxn];
        int Ans[maxn],pt[maxn];
        int n,m;
        void pivot(int l,int i){
                double t;
                swap(Ans[l+n],Ans[i]);
                t=-a[l][i];
                a[l][i]=-1;
                for(int j=0;j<=n;j++)a[l][j]/=t;
                for(int j=0;j<=m;j++){
                        if(a[j][i]&&j!=l){
                                t=a[j][i];
                                a[j][i]=0;
                                for(int k=0;k<=n;k++)a[j][k]+=t*a[l][k];
                        }
                }
        }
        vector<double> solve(vector<vector<double> >A,vector<double>B,vector<double>C){
                n=C.size();
                m=B.size();
                for(int i=0;i<C.size();i++)
```

```
        a[0][i+1]=C[i];
    for(int i=0;i<B.size();i++)
        a[i+1][0]=B[i];

    for(int i=0;i<m;i++)
        for(int j=0;j<n;j++)
            a[i+1][j+1]=-A[i][j];

    for(int i=1;i<=n;i++)Ans[i]=i;

    double t;
    for(;;){
        int l=0;t=-eps;
        for(int j=1;j<=m;j++)if(a[j][0]<t)t=a[l=j][0];
        if(!l)break;
        int i=0;
        for(int j=1;j<=n;j++)if(a[l][j]>eps){i=j;break;}
        if(!i){
            puts("Infeasible");
            return vector<double>();
        }
        pivot(l,i);
    }
    for(;;){
        int i=0;t=eps;
        for(int j=1;j<=n;j++)if(a[0][j]>t)t=a[0][i=j];
        if(!i)break;
        int l=0;
        t=1e30;
        for(int j=1;j<=m;j++)if(a[j][i]<-eps){
            double tmp;
            tmp=-a[j][0]/a[j][i];
            if(t>tmp)t=tmp,l=j;
        }
        if(!l){
            puts("Unbounded");
            return vector<double>();
        }
        pivot(l,i);
    }
    vector<double>x;
    for(int i=n+1;i<=n+m;i++)pt[Ans[i]]=i-n;
```

```
                for(int i=1;i<=n;i++)x.push_back(pt[i]?a[pt[i]][0]:0);
                return x;
        }
}
```

2.4 自适应辛普森

```
double area(const double &left, const double &right) {
    double mid = (left + right) / 2;
    return (right - left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
}

double simpson(const double &left, const double &right,
               const double &eps, const double &area_sum) {
    double mid = (left + right) / 2;
    double area_left = area(left, mid);
    double area_right = area(mid, right);
    double area_total = area_left + area_right;
    if (std::abs(area_total - area_sum) < 15 * eps) {
        return area_total + (area_total - area_sum) / 15;
    }
    return simpson(left, mid, eps / 2, area_left)
         + simpson(mid, right, eps / 2, area_right);
}

double simpson(const double &left, const double &right, const double &eps) {
    return simpson(left, right, eps, area(left, right));
}
```

2.5 多项式求根

```
const double eps=1e-12;
double a[10][10];
typedef vector<double> vd;
int sgn(double x) { return x < -eps ? -1 : x > eps; }
double mypow(double x,int num){
        double ans=1.0;
        for(int i=1;i<=num;++i)ans*=x;
        return ans;
}
double f(int n,double x){
```

```
        double ans=0;
        for(int i=n;i>=0;--i)ans+=a[n][i]*mypow(x,i);
        return ans;
}
double getRoot(int n,double l,double r){
        if(sgn(f(n,l))==0)return l;
        if(sgn(f(n,r))==0)return r;
        double temp;
        if(sgn(f(n,l))>0)temp=-1;else temp=1;
        double m;
        for(int i=1;i<=10000;++i){
                m=(l+r)/2;
                double mid=f(n,m);
                if(sgn(mid)==0){
                        return m;
                }
                if(mid*temp<0)l=m;else r=m;
        }
        return (l+r)/2;
}
vd did(int n){
        vd ret;
        if(n==1){
                ret.push_back(-1e10);
                ret.push_back(-a[n][0]/a[n][1]);
                ret.push_back(1e10);
                return ret;
        }
        vd mid=did(n-1);
        ret.push_back(-1e10);
        for(int i=0;i+1<mid.size();++i){
                int t1=sgn(f(n,mid[i])),t2=sgn(f(n,mid[i+1]));
                if(t1*t2>0)continue;
                ret.push_back(getRoot(n,mid[i],mid[i+1]));
        }
        ret.push_back(1e10);
        return ret;
}
int main(){
        int n; scanf("%d",&n);
        for(int i=n;i>=0;--i){
                scanf("%lf",&a[n][i]);
```

```
        }
        for(int i=n-1;i>=0;--i)
                for(int j=0;j<=i;++j)a[i][j]=a[i+1][j+1]*(j+1);
        vd ans=did(n);
        sort(ans.begin(),ans.end());
        for(int i=1;i+1<ans.size();++i)printf("%.10f\n",ans[i]);
        return 0;
}
```

3 数据结构

3.1 平衡的二叉查找树

3.1.1 Treap

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e5+5;
#define sz(x) (x?x->siz:0)
struct Treap{
        struct node{
                int key,val;
                int siz,s;
                node *c[2];
                node(int v=0){
                        val=v;
                        key=rand();
                        siz=1,s=1;
                        c[0]=c[1]=0;
                }
                void rz(){siz=s;if(c[0])siz+=c[0]->siz;if(c[1])siz+=c[1]->siz;}
        }pool[maxn],*cur,*root;
        Treap(){cur=pool;}
        node* newnode(int val){return *cur=node(val),cur++;}
        void rot(node *&t,int d){
                if(!t->c[d])t=t->c[!d];
                else{
                        node *p=t->c[d];t->c[d]=p->c[!d];
                        p->c[!d]=t;t->rz();p->rz();t=p;
                }
        }
        void insert(node *&t,int x){
```

```
28              if(!t){t=newnode(x);return;}
29              if(t->val==x){t->s++;t->siz++;return;}
30              insert(t->c[x>t->val],x);
31              if(t->key<t->c[x>t->val]->key)
32                      rot(t,x>t->val);
33              else t->rz();
34      }
35      void del(node *&t,int x){
36              if(!t)return;
37              if(t->val==x){
38                      if(t->s>1){t->s--;t->siz--;return;}
39                      if(!t->c[0]||!t->c[1]){
40                              if(!t->c[0])t=t->c[1];
41                              else t=t->c[0];
42                              return;
43                      }
44                      int d=t->c[0]->key<t->c[1]->key;
45                      rot(t,d);
46                      del(t,x);
47                      return;
48              }
49              del(t->c[x>t->val],x);
50              t->rz();
51      }
52      int pre(node *t,int x){
53              if(!t)return INT_MIN;
54              int ans=pre(t->c[x>t->val],x);
55              if(t->val<x)ans=max(ans,t->val);
56              return ans;
57      }
58      int nxt(node *t,int x){
59              if(!t)return INT_MAX;
60              int ans=nxt(t->c[x>=t->val],x);
61              if(t->val>x)ans=min(ans,t->val);
62              return ans;
63      }
64      int rank(node *t,int x){
65              if(!t)return 0;
66              if(t->val==x)return sz(t->c[0]);
67              if(t->val<x)return sz(t->c[0])+t->s+rank(t->c[1],x);
68              if(t->val>x)return rank(t->c[0],x);
69      }
```

```
        int kth(node *t,int x){
                if(sz(t->c[0])>=x)return kth(t->c[0],x);
                if(sz(t->c[0])+t->s>=x)return t->val;
                return kth(t->c[1],x-t->s-sz(t->c[0]));
        }
        void deb(node *t){
                if(!t)return;
                deb(t->c[0]);
                printf("%d ",t->val);
                deb(t->c[1]);
        }
        void insert(int x){insert(root,x);}
        void del(int x){del(root,x);}
        int pre(int x){return pre(root,x);}
        int nxt(int x){return nxt(root,x);}
        int rank(int x){return rank(root,x);}
        int kth(int x){return kth(root,x);}
        void deb(){deb(root);puts("");}
}T;
int main(){
    srand(12121);
    int m;
    scanf("%d",&m);
    while(m--){
        int opt,x;
        scanf("%d",&opt);
        switch(opt){
            case 1:
                scanf("%d",&x);
                T.insert(x);
                break;
            case 2:
                scanf("%d",&x);
                T.del(x);
                break;
            case 3:
                scanf("%d",&x);
                printf("%d\n",T.rank(x)+1);
                break;
            case 4:
                scanf("%d",&x);
                printf("%d\n",T.kth(x));
```

```
                break;
            case 5:
                scanf("%d",&x);
                printf("%d\n",T.pre(x));
                break;
            case 6:
                scanf("%d",&x);
                printf("%d\n",T.nxt(x));
                break;
        }
    }
        return 0;
}
```

3.1.2 Splay

```
void Rotate(int x, int c){
        int y = T[x].c[c];
        int z = T[y].c[1 - c];

        if (T[x].fa){
                if (T[T[x].fa].c[0] == x) T[T[x].fa].c[0] = y;
                else T[T[x].fa].c[1] = y;
        }

        T[z].fa = x; T[x].c[c] = z;
        T[y].fa = T[x].fa; T[x].fa = y; T[y].c[1 - c] = x;

        Update(x);
        Update(y);
}

int stack[M], fx[M];

void Splay(int x, int fa){
        int top = 0;
        for (int  u = x; u != fa; u = T[u].fa)
                stack[++top] = u;
        for (int i = 2; i <= top; i++)
                if (T[stack[i]].c[0] == stack[i - 1]) fx[i] = 0;
                else fx[i] = 1;

```

```
        for (int i = 2; i <= top; i += 2){
                if (i == top) Rotate(stack[i], fx[i]);
                else {
                        if (fx[i] == fx[i + 1]){
                                Rotate(stack[i + 1], fx[i + 1]);
                                Rotate(stack[i], fx[i]);
                        } else {
                                Rotate(stack[i], fx[i]);
                                Rotate(stack[i + 1], fx[i + 1]);
                        }
                }
        }

        if (fa == 0) Root = x;
}
```

3.2 坚固的数据结构

3.2.1 坚固的平衡树

```
#define sz(x) (x?x->siz:0)
struct node{
    int siz,key;
    LL val,sum;
    LL mu,a,d;
    node *c[2],*f;
    void split(int ned,node *&p,node *&q);
    node* rz(){
        sum=val;siz=1;
        if(c[0])sum+=c[0]->sum,siz+=c[0]->siz;
        if(c[1])sum+=c[1]->sum,siz+=c[1]->siz;
        return this;
    }
    void make(LL _mu,LL _a,LL _d){
        sum=sum*_mu+_a*siz+_d*siz*(siz-1)/2;
        val=val*_mu+_a+_d*sz(c[0]);
        mu*=_mu;a=a*_mu+_a;d=d*_mu+_d;
    }
    void pd(){
        if(mu==1&&a==0&&d==0)return;
        if(c[0])c[0]->make(mu,a,d);
        if(c[1])c[1]->make(mu,a+d+d*sz(c[0]),d);
```

```
        mu=1;a=d=0;
    }
    node(){mu=1;}
}nd[maxn*2],*root;
node *merge(node *p,node *q){
    if(!p||!q)return p?p->rz():(q?q->rz():0);
    p->pd();q->pd();
    if(p->key<q->key){
        p->c[1]=merge(p->c[1],q);
        return p->rz();
    }else{
        q->c[0]=merge(p,q->c[0]);
        return q->rz();
    }
}
void node::split(int ned,node *&p,node *&q){
    if(!ned){p=0;q=this;return;}
    if(ned==siz){p=this;q=0;return;}
    pd();
    if(sz(c[0])>=ned){
        c[0]->split(ned,p,q);c[0]=0;rz();
        q=merge(q,this);
    }else{
        c[1]->split(ned-sz(c[0])-1,p,q);c[1]=0;rz();
        p=merge(this,p);
    }
}
int main(){
    for(int i=1;i<=n;i++){
        nd[i].val=in();
        nd[i].key=rand();
        nd[i].rz();
        root=merge(root,nd+i);
    }
}
```

3.2.2 坚固的字符串

1. ext 库中的 rope

```
#include <ext/rope>

using __gnu_cxx::crope;
```

```cpp
using __gnu_cxx::rope;

crope a, b;

int main(void) {
    a = b.substr(pos, len);     // [pos, pos + len)
    a = b.substr(pos);          // [pos, pos]
    b.c_str();                  // might lead to memory leaks
    b.delete_c_str();           // delete the c_str that created before
    a.insert(pos, text);        // insert text before position pos
    a.erase(pos, len);          // erase [pos, pos + len)
}
```

2. 可持久化平衡树实现的 rope

```cpp
class Rope {
private:
    class Node {
    public:
        Node *left, *right;
        int size;
        char key;

        Node(char key = 0, Node *left = NULL, Node *right = NULL)
               : key(key), left(left), right(right) {
            update();
        }

        void update() {
            size = (left ? left->size : 0) + 1 + (right ? right->size : 0);
        }

        std::string to_string() {
            return (left ? left->to_string() : "") + key
                 + (right ? right->to_string() : "");
        }
    };

    bool random(int a, int b) {
        return rand() % (a + b) < a;
    }

    Node* merge(Node *x, Node *y) {
```

```
29          if (!x) {
30              return y;
31          }
32          if (!y) {
33              return x;
34          }
35          if (random(x->size, y->size)) {
36              return new Node(x->key, x->left, merge(x->right, y));
37          } else {
38              return new Node(y->key, merge(x, y->left), y->right);
39          }
40      }
41
42      std::pair<Node*, Node*> split(Node *x, int size) {
43          if (!x) {
44              return std::make_pair<Node*, Node*>(NULL, NULL);
45          }
46          if (size == 0) {
47              return std::make_pair<Node*, Node*>(NULL, x);
48          }
49          if (size > x->size) {
50              return std::make_pair<Node*, Node*>(x, NULL);
51          }
52          if (x->left && size <= x->left->size) {
53              std::pair<Node*, Node*> part =
54                  split(x->left, size);
55              return std::make_pair(part.first, new Node(x->key, part.second, x->right));
56          } else {
57              std::pair<Node*, Node*> part =
58                  split(x->right, size - (x->left ? x->left->size : 0) - 1);
59              return std::make_pair(new Node(x->key, x->left, part.first), part.second);
60          }
61      }
62
63      Node* build(const std::string &text, int left, int right) {
64          if (left > right) {
65              return NULL;
66          }
67          int mid = left + right >> 1;
68          return new Node(text[mid],
69                          build(text, left, mid - 1),
70                          build(text, mid + 1, right));
```

```
    }

public:
    Node *root;

    Rope() {
        root = NULL;
    }

    Rope(const std::string &text) {
        root = build(text, 0, (int)text.length() - 1);
    }

    Rope(const Rope &other) {
        root = other.root;
    }

    Rope& operator = (const Rope &other) {
        if (this == &other) {
            return *this;
        }
        root = other.root;
        return *this;
    }

    int size() {
        return root ? root->size : 0;
    }

    void insert(int pos, const std::string &text) {
        if (pos < 0 || pos > size()) {
            throw "Out of range";
        }
        std::pair<Node*, Node*> part = split(root, pos);
        root = merge(merge(part.first, build(text, 0, (int)text.length() - 1)),
                     part.second);
    }

    void erase(int left, int right) {
        if (left < 0 || left >= size() ||
            right < 1 || right > size()) {
            throw "Out of range";
```

```
113             }
114             if (left >= right) {
115                 return;
116             }
117             std::pair<Node*, Node*> part = split(root, left);
118             root = merge(part.first, split(part.second, right - left).second);
119         }
120 
121         std::string substr(int left, int right) {
122             if (left < 0 || left >= size() ||
123                 right < 1 || right > size()) {
124                 throw "Out of range";
125             }
126             if (left >= right) {
127                 return "";
128             }
129             return split(split(root, left).second, right - left).first->to_string();
130         }
131 
132         void copy(int left, int right, int pos) {
133             if (left < 0 || left >= size() ||
134                 right < 1 || right > size() ||
135                 pos < 0 || pos > size()) {
136                 throw "Out of range";
137             }
138             if (left >= right) {
139                 return;
140             }
141             std::pair<Node*, Node*> part = split(root, pos);
142             root = merge(merge(part.first,
143                                split(split(root, left).second, right - left).first),
144                          part.second);
145         }
146     };
```

3.2.3 坚固的左偏树

```
1  int Merge(int x, int y){
2    if (x == 0 || y == 0) return x + y;
3    if (Heap[x].Key < Heap[y].Key) swap(x, y);
4    Heap[x].Ri = Merge(Heap[x].Ri, y);
5    if (Heap[Heap[x].Le].Dis < Heap[Heap[x].Ri].Dis) swap(Heap[x].Le, Heap[x].Ri);
```

```
  if (Heap[x].Ri == 0) Heap[x].Dis = 0;
  else Heap[x].Dis = Heap[Heap[x].Ri].Dis + 1;
  return x;
}

for (int i = 0; i <= n; i++){
        Heap[i].Le = Heap[i].Ri = 0;
        Heap[i].Dis = 0;
        Heap[i].Key = Cost[i];
}
Heap[0].Dis = -1;
```

3.2.4 不坚固的斜堆

```
struct node;
node *Null,*root[maxn];
struct node{
        node* c[2];
        int val,ind;
        node(int _val=0,int _ind=0){
                val=_val;c[0]=c[1]=Null;ind=_ind;
        }
};
node* merge(node *p,node *q){
        if(p==Null)return q;
        if(q==Null)return p;
        if(p->val>q->val)swap(p,q);
        p->c[1]=merge(p->c[1],q);
        swap(p->c[0],p->c[1]);
        return p;
}

Null=new node(0);
Null->c[0]=Null->c[1]=Null;
```

3.3 树上的魔术师

3.3.1 轻重树链剖分 (zky)

```
vector<int>G[maxn];
int fa[maxn],top[maxn],siz[maxn],son[maxn],mp[maxn],z,dep[maxn];
void dfs(int u){
```

```
        siz[u]=1;
        for(int i=0;i<G[u].size();i++){
                int v=G[u][i];
                if(v!=fa[u]){
                        fa[v]=u;dep[v]=dep[u]+1;
                        dfs(v);
                        siz[u]+=siz[v];
                        if(siz[son[u]]<siz[v])son[u]=v;
                }
        }
}
void build(int u,int tp){
        top[u]=tp;mp[u]=++z;
        if(son[u])build(son[u],tp);
        for(int v,i=0;i<G[u].size();i++)if((v=G[u][i])!=son[u]&&v!=fa[u])build(v,v);
}
```

3.3.2 轻重树链剖分 (lyx)

```
void Prep(int x){
        dep[x] = dep[fa[x]] + 1;
        size[x] = 1;
        son[x] = 0;
        for (int i = g[x]; i; i = nxt[i]){
                int y = adj[i];
                if (y == fa[x]) continue;
                fa[y] = x;
                Prep(y);
                size[x] += size[y];
                if (size[y] > size[son[x]]) son[x] = y;
        }
}
void Dfs(int x){
        dfn[x] = ++dfc;
        if (son[x] != 0){
                top[son[x]] = top[x];
                Dfs(son[x]);
        }
        for (int i = g[x]; i; i = nxt[i]){
                int y = adj[i];
                if (y != fa[x] && y != son[x]){
                        top[y] = y;
```

```
                        Dfs(y);
                }
                if (y != fa[x]){
                        Bel[(i + 1) >> 1] = dfn[y];
                        val[dfn[y]] = len[i];
                }
        }
}
int Ask(int x, int y){
        int Ret = -1000000001;
        while (top[x] != top[y]){
                if (dep[top[y]] > dep[top[x]]) swap(x, y);
                Ret = max(Ret, Query(1, 1, n, dfn[top[x]], dfn[x]));
                x = fa[top[x]];
        }
        if (dep[y] > dep[x]) swap(x, y);
        if (x != y)
                Ret = max(Ret, Query(1, 1, n, dfn[y] + 1, dfn[x]));
        return Ret;
}
//Hints : Ask 部分具体的求值或者修改要稍作变动
```

3.3.3 Link Cut Tree(zky)

```
struct LCT{
    struct node{
        bool rev;
        int mx,val;
        node *f,*c[2];
        bool d(){return this==f->c[1];}
        bool rt(){return !f||(f->c[0]!=this&&f->c[1]!=this);}
        void sets(node *x,int d){pd();if(x)x->f=this;c[d]=x;rz();}
        void makerv(){rev^=1;swap(c[0],c[1]);}
        void pd(){
            if(rev){
                if(c[0])c[0]->makerv();
                if(c[1])c[1]->makerv();
                rev=0;
            }
        }
        void rz(){
            mx=val;
```

```
            if(c[0])mx=max(mx,c[0]->mx);
            if(c[1])mx=max(mx,c[1]->mx);
        }
    }nd[int(1e4)+1];
    void rot(node *x){
        node *y=x->f;if(!y->rt())y->f->pd();
        y->pd();x->pd();bool d=x->d();
        y->sets(x->c[!d],d);
        if(y->rt())x->f=y->f;
        else y->f->sets(x,y->d());
        x->sets(y,!d);
    }
    void splay(node *x){
        while(!x->rt())
            if(x->f->rt())rot(x);
            else if(x->d()==x->f->d())rot(x->f),rot(x);
            else rot(x),rot(x);
    }
    node* access(node *x){
        node *y=0;
        for(;x;x=x->f){
            splay(x);
            x->sets(y,1);y=x;
        }return y;
    }
    void makert(node *x){
        access(x)->makerv();
        splay(x);
    }
    void link(node *x,node *y){
        makert(x);
        x->f=y;
        access(x);
    }
    void cut(node *x,node *y){
        makert(x);access(y);splay(y);
        y->c[0]=x->f=0;
        y->rz();
    }
    void link(int x,int y){link(nd+x,nd+y);}
    void cut(int x,int y){cut(nd+x,nd+y);}
}T;
```

3.3.4 Link Cut Tree(lyx)

```
struct node{
        bool Rev;
        int c[2], fa;
}T[N];


inline void Rev(int x){
        if (!x) return;
        swap(T[x].c[0], T[x].c[1]);
        T[x].Rev ^= 1;
}

inline void Lazy_Down(int x){
        if (!x) return;
        if (T[x].Rev) Rev(T[x].c[0]), Rev(T[x].c[1]), T[x].Rev = 0;
}

void Rotate(int x, int c){
        int y = T[x].c[c];
        int z = T[y].c[1 - c];

        if (T[x].fa){
                if (T[T[x].fa].c[0] == x) T[T[x].fa].c[0] = y;
                else T[T[x].fa].c[1] = y;
        }

        T[z].fa = x; T[x].c[c] = z;
        T[y].fa = T[x].fa; T[x].fa = y; T[y].c[1 - c] = x;

        //Update(x);
        //Update(y);
}

int stack[N], fx[N];

void Splay(int x){
        int top = 0;
        for (int  u = x; u; u = T[u].fa)
                stack[++top] = u;
        for (int i = top; i >= 1; i--)
```

```
                Lazy_Down(stack[i]);
        for (int i = 2; i <= top; i++)
                if (T[stack[i]].c[0] == stack[i - 1]) fx[i] = 0;
                else fx[i] = 1;

        for (int i = 2; i <= top; i += 2){
                if (i == top) Rotate(stack[i], fx[i]);
                else {
                        if (fx[i] == fx[i + 1]){
                                Rotate(stack[i + 1], fx[i + 1]);
                                Rotate(stack[i], fx[i]);
                        } else {
                                Rotate(stack[i], fx[i]);
                                Rotate(stack[i + 1], fx[i + 1]);
                        }
                }
        }

        if (x != stack[top]) Par[x] = Par[stack[top]], Par[stack[top]] = 0;
}

inline int Access(int u){
        int Nxt = 0;
        while (u){
                Splay(u);
                if (T[u].c[1]){
                        T[T[u].c[1]].fa = 0;
                        Par[T[u].c[1]] = u;
                }
                T[u].c[1] = Nxt;
                if (Nxt){
                        T[Nxt].fa = u;
                        Par[Nxt] = 0;
                }
                //Update(u)
                Nxt = u;
                u = Par[u];
        }
        return Nxt;
}

inline void Root(int u){
```

```
        Access(u);
        Splay(u);
        Rev(u);
}

inline void Link(int u, int v){
        Root(u);
        Par[u] = v;
}

inline void Cut(int u, int v){
        Access(u);
        Splay(v);
        if (Par[v] != u){
                swap(u, v);
                Access(u);
                Splay(v);
        }
        Par[v] = 0;
}

inline int Find_Root(int x){
        Access(x);
        Splay(x);

        int y = x;
        while (T[y].c[0]){
                Lazy_Down(y);
                y = T[y].c[0];
        }
        return y;
}
```

3.3.5 AAA Tree

```
#define rep(i,a,n) for(int i=a;i<n;i++)
int n,m;
struct info{
    int mx,mn,sum,sz;
    info(){}
    info(int mx,int mn,int sum,int sz):
        mx(mx),mn(mn),sum(sum),sz(sz){}
```

```
    void deb(){printf("sum:%d size:%d",(int)sum,sz);}
};
struct flag{
    int mul,add;
    flag(){mul=1;}
    flag(int mul,int add):
        mul(mul),add(add){}
    bool empty(){return mul==1&&add==0;}
};
info operator+(const info &a,const flag &b) {
    return a.sz?info(a.mx*b.mul+b.add,a.mn*b.mul+b.add,a.sum*b.mul+b.add*a.sz,a.sz):a;
}
info operator+(const info &a,const info &b) {
    return info(max(a.mx,b.mx),min(a.mn,b.mn),a.sum+b.sum,a.sz+b.sz);
}
flag operator+(const flag &a,const flag &b) {
    return flag(a.mul*b.mul,a.add*b.mul+b.add);
}
struct node{
    node *c[4],*f;
    flag Cha,All;
    info cha,sub,all;
    bool rev,inr;
    int val;
    void makerev(){rev^=1;swap(c[0],c[1]);}
    void makec(const flag &a){
        Cha=Cha+a;cha=cha+a;val=val*a.mul+a.add;
        all=cha+sub;
    }
    void makes(const flag &a,bool _=1){
        All=All+a;all=all+a;sub=sub+a;
        if(_)makec(a);
    }
    void rz(){
        cha=all=sub=info(-(1<<30),1<<30,0,0);
        if(!inr)all=cha=info(val,val,val,1);
        rep(i,0,2)if(c[i])cha=cha+c[i]->cha,sub=sub+c[i]->sub;
        rep(i,0,4)if(c[i])all=all+c[i]->all;
        rep(i,2,4)if(c[i])sub=sub+c[i]->all;
    }
    void pd(){
        if(rev){
```

```
            if(c[0])c[0]->makerev();
            if(c[1])c[1]->makerev();
            rev=0;
        }
        if(!All.empty()){
            rep(i,0,4)if(c[i])c[i]->makes(All,i>=2);
            All=flag(1,0);
        }
        if(!Cha.empty()){
            rep(i,0,2)if(c[i])c[i]->makec(Cha);
            Cha=flag(1,0);
        }

    }
    node *C(int i){if(c[i])c[i]->pd();return c[i];}
    bool d(int ty){return f->c[ty+1]==this;}
    int D(){rep(i,0,4)if(f->c[i]==this)return i;}
    void sets(node *x,int d){if(x)x->f=this;c[d]=x;}
    bool rt(int ty){
        if(ty==0)return !f||(f->c[0]!=this&&f->c[1]!=this);
        else return !f||!f->inr||!inr;
    }
}nd[maxn*2],*cur=nd+maxn,*pool[maxn],**Cur=pool;
int _cnt;
node *newnode(){
    _cnt++;
    node *x=(Cur==pool)?cur++:*(--Cur);
    rep(i,0,4)x->c[i]=0;x->f=0;
    x->All=x->Cha=flag(1,0);
    x->all=x->cha=info(-(1<<30),(1<<30),0,0);
    x->inr=1;x->rev=0;x->val=0;
    return x;
}
void dele(node *x){*(Cur++)=x;}
void rot(node *x,int ty){
    node *p=x->f;int d=x->d(ty);
    if(!p->f)x->f=0;else p->f->sets(x,p->D());
    p->sets(x->c[!d+ty],d+ty);x->sets(p,!d+ty);p->rz();
}
void splay(node *x,int ty=0){
    while(!x->rt(ty)){
        if(x->f->rt(ty))rot(x,ty);
```

```
92          else if(x->d(ty)==x->f->d(ty))rot(x->f,ty),rot(x,ty);
93          else rot(x,ty),rot(x,ty);
94      }x->rz();
95  }
96  void add(node *u,node *w){
97      w->pd();
98      rep(i,2,4)if(!w->c[i]){w->sets(u,i);return;}
99      node *x=newnode(),*v;
100     for(v=w;v->c[2]->inr;v=v->C(2));
101     x->sets(v->c[2],2);x->sets(u,3);
102     v->sets(x,2);splay(x,2);
103 }
104 void del(node *w){
105     if(w->f->inr){
106         w->f->f->sets(w->f->c[5-w->D()],w->f->D());
107         dele(w->f);splay(w->f->f,2);
108     }else w->f->sets(0,w->D());
109     w->f=0;
110 }
111 void access(node *w){
112     static node *sta[maxn];
113     static int top=0;
114     node *v=w,*u;
115     for(u=w;u;u=u->f)sta[top++]=u;
116     while(top)sta[--top]->pd();
117     splay(w);
118     if(w->c[1])u=w->c[1],w->c[1]=0,add(u,w),w->rz();
119     while(w->f){
120         for(u=w->f;u->inr;u=u->f);
121         splay(u);
122         if(u->c[1])w->f->sets(u->c[1],w->D()),splay(w->f,2);
123         else del(w);
124         u->sets(w,1);
125         (w=u)->rz();
126     }splay(v);
127 }
128 void makert(node *x){
129     access(x);x->makerev();
130 }
131 node *findp(node *u){
132     access(u);u=u->C(0);
133     while(u&&u->c[1])u=u->C(1);
```

```
    return u;
}
node *findr(node *u){for(;u->f;u=u->f);return u;}
node* cut(node *u){
    node *v=findp(u);
    if(v)access(v),del(u),v->rz();
    return v;
}
void link(node *u,node *v) {
    node* p=cut(u);
    if(findr(u)!=findr(v))p=v;
    if(p)access(p),add(u,p),p->rz();
}
int main(){
//  freopen("bzoj3153.in","r",stdin);
    n=getint();m=getint();
    static int _u[maxn],_v[maxn];
    rep(i,1,n)_u[i]=getint(),_v[i]=getint();
    rep(i,1,n+1){
        nd[i].val=getint();
        nd[i].rz();
    }
    rep(i,1,n)makert(nd+_u[i]),link(nd+_u[i],nd+_v[i]);
    int root=getint();
    makert(nd+root);
//  deb();
    int x,y,z;
    node *u,*v;
    while(m--){
        int k=getint();x=getint();
        u=nd+x;
        if(k==0||k==3||k==4||k==5||k==11){
            access(u);
            if(k==3||k==4||k==11){
                int ans=u->val;
                rep(i,2,4)if(u->c[i]){
                    info res=u->c[i]->all;
                    if(k==3) ans=min(ans,res.mn);
                    else if(k==4) ans=max(ans,res.mx);
                    else if(k==11) ans+=res.sum;
                }printf("%d\n",ans);
            }else{
```

```
                y=getint();
                flag fg(k==5,y);
                u->val=u->val*fg.mul+fg.add;
                rep(i,2,4)if(u->c[i])u->c[i]->makes(fg);
                u->rz();
            }
        }else if(k==2||k==6||k==7||k==8||k==10){
            y=getint();
            makert(u),access(nd+y),splay(u);
            if (k==7||k==8||k==10) {
                info ans=u->cha;
                if (k==7) printf("%d\n",ans.mn);
                else if (k==8) printf("%d\n",ans.mx);
                else printf("%d\n",ans.sum);
            }else u->makec(flag(k==6,getint()));
            makert(nd+root);
        }else if(k==9)link(u,nd+getint());
        else if(k==1)makert(u),root=x;
    }
    return 0;
}
```

3.4 ST

```
for (int i = 1; i <= n; i++)
        Log[i] = int(log2(i));

for (int i = 1; i <= n; i++)
        Rmq[i][0] = i;

for (int k = 1; (1 << k) <= n; k++)
        for (int i = 1; i + (1 << k) - 1 <= n; i++){
                int x = Rmq[i][k - 1], y = Rmq[i + (1 << (k - 1))][k - 1];
                if (a[x] < a[y])
                        Rmq[i][k] = x;
                else
                        Rmq[i][k] = y;
        }

int Smallest(int l, int r){
        int k = Log[r - l + 1];

```

```
        int x = Rmq[l][k];
        int y = Rmq[r - (1 << k) + 1][k];

        if (a[x] < a[y]) return x;
        else return y;
}
```

3.5 可持久化线段树

```
struct node1 {
        int L, R, Lson, Rson, Sum;
} tree[N * 40];
int root[N], a[N], b[N];
int tot, n, m;
int Real[N];
int Same(int x) {
        ++tot;
        tree[tot] = tree[x];
        return tot;
}
int build(int L, int R) {
        ++tot;
        tree[tot].L = L;
        tree[tot].R = R;
        tree[tot].Lson = tree[tot].Rson = tree[tot].Sum = 0;
        if (L == R) return tot;
        int s = tot;
        int mid = (L + R) >> 1;
        tree[s].Lson = build(L, mid);
        tree[s].Rson = build(mid + 1, R);
        return s;
}
int Ask(int Lst, int Cur, int L, int R, int k) {
        if (L == R) return L;
        int Mid = (L + R) >> 1;
        int Left = tree[tree[Cur].Lson].Sum - tree[tree[Lst].Lson].Sum;
        if (Left >= k) return Ask(tree[Lst].Lson, tree[Cur].Lson, L, Mid, k);
        k -= Left;
        return Ask(tree[Lst].Rson, tree[Cur].Rson, Mid + 1, R, k);
}
int Add(int Lst, int pos) {
        int root = Same(Lst);
```

```
        tree[root].Sum++;
        if (tree[root].L == tree[root].R) return root;
        int mid = (tree[root].L + tree[root].R) >> 1;
        if (pos <= mid) tree[root].Lson = Add(tree[root].Lson, pos);
        else tree[root].Rson = Add(tree[root].Rson, pos);
        return root;
}
int main() {
        scanf("%d%d", &n, &m);
        int up = 0;
        for (int i = 1; i <= n; i++){
                scanf("%d", &a[i]);
                b[i] = a[i];
        }
        sort(b + 1, b + n + 1);
        up = unique(b + 1, b + n + 1) - b - 1;
        for (int i = 1; i <= n; i++){
                int tmp = lower_bound(b + 1, b + up + 1, a[i]) - b;
                Real[tmp] = a[i];
                a[i] = tmp;
        }
        tot = 0;
        root[0] = build(1, up);
        for (int i = 1; i <= n; i++){
                root[i] = Add(root[i - 1], a[i]);
        }
        for (int i = 1; i <= m; i++){
                int u, v, w;
                scanf("%d%d%d", &u, &v, &w);
                printf("%d\n", Real[Ask(root[u - 1], root[v], 1, up, w)]);
        }
        return 0;
}
```

3.6 可持久化 Trie

```
int Pre[N];
int n, q, Len, cnt, Lstans;
char s[N];
int First[N], Last[N];
int Root[N];
int Trie_tot;
```

```
struct node{
    int To[30];
    int Lst;
}Trie[N];
int tot;
struct node1{
    int L, R, Lson, Rson, Sum;
}tree[N * 25];
int Build(int L, int R){
    ++tot;
    tree[tot].L = L;
    tree[tot].R = R;
    tree[tot].Lson = tree[tot].Rson = tree[tot].Sum = 0;
    if (L == R) return tot;
    int s = tot;
    int mid = (L + R) >> 1;
    tree[s].Lson = Build(L, mid);
    tree[s].Rson = Build(mid + 1, R);
    return s;
}
int Same(int x){
    ++tot;
    tree[tot] = tree[x];
    return tot;
}
int Add(int Lst, int pos){
    int s = Same(Lst);
    tree[s].Sum++;
    if (tree[s].L == tree[s].R) return s;
    int Mid = (tree[s].L + tree[s].R) >> 1;
    if (pos <= Mid) tree[s].Lson = Add(tree[Lst].Lson, pos);
    else tree[s].Rson = Add(tree[Lst].Rson, pos);
    return s;
}

int Ask(int Lst, int Cur, int L, int R, int pos){
    if (L >= pos) return 0;
    if (R < pos) return tree[Cur].Sum - tree[Lst].Sum;
    int Mid = (L + R) >> 1;
    int Ret = Ask(tree[Lst].Lson, tree[Cur].Lson, L, Mid, pos);
    Ret += Ask(tree[Lst].Rson, tree[Cur].Rson, Mid + 1, R, pos);
    return Ret;
```

```
}

int main(){
    while (scanf("%d", &n) == 1){
        for (int i = 1; i <= Trie_tot; i++){
            for (int j = 1; j <= 26; j++)
                Trie[i].To[j] = 0;
            Trie[i].Lst = 0;
        }
        Trie_tot = 1;
        cnt = 0;
        for (int ii = 1; ii <= n; ii++){
            scanf("%s", s + 1);
            Len = strlen(s + 1);
            int Cur = 1;
            First[ii] = cnt + 1;
            for (int i = 1; i <= Len; i++){
                int ch = s[i] - 'a' + 1;
                if (Trie[Cur].To[ch] == 0){
                    ++Trie_tot;
                    Trie[Cur].To[ch] = Trie_tot;
                }
                Cur = Trie[Cur].To[ch];
                Pre[++cnt] = Trie[Cur].Lst;
                Trie[Cur].Lst = ii;
            }
            Last[ii] = cnt;
        }
        tot = 0;
        Root[0] = Build(0, n);
        for (int i = 1; i <= cnt; i++){
            Root[i] = Add(Root[i - 1], Pre[i]);
        }
        Lstans = 0;
        scanf("%d", &q);
        for (int ii = 1; ii <= q; ii++){
            int L, R;
            scanf("%d%d", &L, &R);
            L = (L + Lstans) % n + 1;
            R = (R + Lstans) % n + 1;
            if (L > R) swap(L, R);
            int Ret = Ask(Root[First[L] - 1], Root[Last[R]], 0, n, L);
```

```
            printf("%d\n", Ret);
            Lstans = Ret;
        }
    }
    return 0;
}
```

3.7 k-d 树

```
long long norm(const long long &x) {
    //    For manhattan distance
    return std::abs(x);
    //    For euclid distance
    return x * x;
}

struct Point {
    int x, y, id;

    const int& operator [] (int index) const {
        if (index == 0) {
            return x;
        } else {
            return y;
        }
    }

    friend long long dist(const Point &a, const Point &b) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            result += norm(a[i] - b[i]);
        }
        return result;
    }
} point[N];

struct Rectangle {
    int min[2], max[2];

    Rectangle() {
        min[0] = min[1] = INT_MAX;
        max[0] = max[1] = INT_MIN;
```

```
    }

    void add(const Point &p) {
        for (int i = 0; i < 2; ++i) {
            min[i] = std::min(min[i], p[i]);
            max[i] = std::max(max[i], p[i]);
        }
    }

    long long dist(const Point &p) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            //    For minimum distance
            result += norm(std::min(std::max(p[i], min[i]), max[i]) - p[i]);
            //    For maximum distance
            result += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
        }
        return result;
    }
};

struct Node {
    Point seperator;
    Rectangle rectangle;
    int child[2];

    void reset(const Point &p) {
        seperator = p;
        rectangle = Rectangle();
        rectangle.add(p);
        child[0] = child[1] = 0;
    }
} tree[N << 1];

int size, pivot;

bool compare(const Point &a, const Point &b) {
    if (a[pivot] != b[pivot]) {
        return a[pivot] < b[pivot];
    }
    return a.id < b.id;
}
```

```cpp

int build(int l, int r, int type = 1) {
    pivot = type;
    if (l >= r) {
        return 0;
    }
    int x = ++size;
    int mid = l + r >> 1;
    std::nth_element(point + l, point + mid, point + r, compare);
    tree[x].reset(point[mid]);
    for (int i = l; i < r; ++i) {
        tree[x].rectangle.add(point[i]);
    }
    tree[x].child[0] = build(l, mid, type ^ 1);
    tree[x].child[1] = build(mid + 1, r, type ^ 1);
    return x;
}

int insert(int x, const Point &p, int type = 1) {
    pivot = type;
    if (x == 0) {
        tree[++size].reset(p);
        return size;
    }
    tree[x].rectangle.add(p);
    if (compare(p, tree[x].seperator)) {
        tree[x].child[0] = insert(tree[x].child[0], p, type ^ 1);
    } else {
        tree[x].child[1] = insert(tree[x].child[1], p, type ^ 1);
    }
    return x;
}

//    For minimum distance
void query(int x, const Point &p, std::pair<long long, int> &answer, int type = 1) {
    pivot = type;
    if (x == 0 || tree[x].rectangle.dist(p) > answer.first) {
        return;
    }
    answer = std::min(answer,
             std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
    if (compare(p, tree[x].seperator)) {
```

```
        query(tree[x].child[0], p, answer, type ^ 1);
        query(tree[x].child[1], p, answer, type ^ 1);
    } else {
        query(tree[x].child[1], p, answer, type ^ 1);
        query(tree[x].child[0], p, answer, type ^ 1);
    }
}

std::priority_queue<std::pair<long long, int> > answer;

void query(int x, const Point &p, int k, int type = 1) {
    pivot = type;
    if (x == 0 ||
        (int)answer.size() == k && tree[x].rectangle.dist(p) > answer.top().first) {
        return;
    }
    answer.push(std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
    if ((int)answer.size() > k) {
        answer.pop();
    }
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, k, type ^ 1);
        query(tree[x].child[1], p, k, type ^ 1);
    } else {
        query(tree[x].child[1], p, k, type ^ 1);
        query(tree[x].child[0], p, k, type ^ 1);
    }
}
```

3.8 莫队算法

```
struct node{
        int l, r, id;
        friend bool operator < (const node &a, const node &b){
                if (a.l / Block == b.l / Block) return a.r / Block < b.r / Block;
                return a.l / Block < b.l / Block;
        }
}q[N];
Block = int(sqrt(n));
for (int i = 1; i <= m; i++){
        scanf("%d%d", &q[i].l, &q[i].r);
        q[i].id = i;
```

```
}
sort(q + 1, q + 1 + m);
Cur = a[1]; /// Hints: adjust by yourself
Le = Ri = 1;
for (int i = 1; i <= m; i++){
        while (q[i].r > Ri) Ri++, ChangeRi(1, Le, Ri);
        while (q[i].l > Le) ChangeLe(-1, Le, Ri), Le++;
        while (q[i].l < Le) Le--, ChangeLe(1, Le, Ri);
        while (q[i].r < Ri) ChangeRi(-1, Le, Ri), Ri--;
        Ans[q[i].id] = Cur;
}
```

3.9 树上在线莫队

```
bool operator<(qes a,qes b){
    if(dfn[a.x]/B!=dfn[b.x]/B)return dfn[a.x]/B<dfn[b.x]/B;
    if(dfn[a.y]/B!=dfn[b.y]/B)return dfn[a.y]/B<dfn[b.y]/B;
    if(a.tm/B!=b.tm/B)return a.tm/B<b.tm/B;
    return a.tm<b.tm;
}
void vxor(int x){
    if(vis[x])ans-=(LL)W[cnt[col[x]]]*V[col[x]],cnt[col[x]]--;
    else cnt[col[x]]++,ans+=(LL)W[cnt[col[x]]]*V[col[x]];
    vis[x]^=1;
}
void change(int x,int y){
    if(vis[x]){
        vxor(x);col[x]=y;vxor(x);
    }else col[x]=y;
}
void TimeMachine(int tar){//XD
    for(int i=now+1;i<=tar;i++)change(C[i].x,C[i].y);
    for(int i=now;i>tar;i--)change(C[i].x,C[i].pre);
    now=tar;
}
void vxor(int x,int y){
    while(x!=y)if(dep[x]>dep[y])vxor(x),x=fa[x];
    else vxor(y),y=fa[y];
}
    for(int i=1;i<=q;i++){
        int ty=getint(),x=getint(),y=getint();
        if(ty&&dfn[x]>dfn[y])swap(x,y);
```

```
        if(ty==0) C[++Csize]=(oper){x,y,pre[x],i},pre[x]=y;
        else Q[Qsize+1]=(qes){x,y,Qsize+1,Csize},Qsize++;
    }sort(Q+1,Q+1+Qsize);
    int u=Q[1].x,v=Q[1].y;
    TimeMachine(Q[1].tm);
    vxor(Q[1].x,Q[1].y);
    int LCA=lca(Q[1].x,Q[1].y);
    vxor(LCA);anss[Q[1].id]=ans;vxor(LCA);
    for(int i=2;i<=Qsize;i++){
        TimeMachine(Q[i].tm);
        vxor(Q[i-1].x,Q[i].x);
        vxor(Q[i-1].y,Q[i].y);
        int LCA=lca(Q[i].x,Q[i].y);
        vxor(LCA);
        anss[Q[i].id]=ans;
        vxor(LCA);
    }
```

3.10 整体二分

```
struct BIT{
        LL d[maxn];
        inline int lowbit(int x){return x&-x;}
        LL get(int x){
                LL ans=0;
                while(x)ans+=d[x],x-=lowbit(x);
                return ans;
        }
        void updata(int x,LL f){
                while(x<=m)d[x]+=f,x+=lowbit(x);
        }
        void add(int l,int r,LL f){
                updata(l,f);
                updata(r+1,-f);
        }
}T,T2;
int anss[maxn],wana[maxn];
struct qes{
        LL x,y,z;
        qes(LL _x=0,LL _y=0,LL _z=0):
                x(_x),y(_y),z(_z){}
}q[maxn],p[maxn];
```

```
bool part(qes &q){
        if(q.y+q.z>=wana[q.x])return 1;
        q.z+=q.y;q.y=0;return 0;
}
void solve(int lef,int rig,int l,int r){
        if(l==r){
                for(int i=lef;i<=rig;i++)if(anss[p[i].x]!=-1)
                anss[p[i].x]=l;return;
        }int mid=(l+r)>>1;
        for(int i=l;i<=mid;i++){
                if(q[i].x<=q[i].y)T.add(q[i].x,q[i].y,q[i].z);
                else T.add(1,q[i].y,q[i].z),T.add(q[i].x,m,q[i].z);
        }for(int i=lef;i<=rig;i++){
                p[i].y=0;
                for(int j=0;j<O[p[i].x].size()&&p[i].y<=int(1e9)+1;j++)
                p[i].y+=T.get(O[p[i].x][j]);
        }for(int i=l;i<=mid;i++){
                if(q[i].x<=q[i].y)T.add(q[i].x,q[i].y,-q[i].z);
                else T.add(1,q[i].y,-q[i].z),T.add(q[i].x,m,-q[i].z);
        }int dv=stable_partition(p+lef,p+rig+1,part)-p-1;
        if(lef<=dv)
        solve(lef,dv,l,mid);
        if(dv+1<=rig)
        solve(dv+1,rig,mid+1,r);
}
```

3.11 树状数组 kth

```
int find(int k){
    int cnt=0,ans=0;
    for(int i=22;i>=0;i--){
        ans+=(1<<i);
        if(ans>n || cnt+d[ans]>=k)ans-=(1<<i);
        else cnt+=d[ans];
    }
    return ans+1;
}
```

3.12 虚树

```cpp
int a[maxn*2],sta[maxn*2];
int top=0,k;
void build(){
    top=0;
    sort(a,a+k,bydfn);
    k=unique(a,a+k)-a;
    sta[top++]=1;_n=k;
    for(int i=0;i<k;i++){
        int LCA=lca(a[i],sta[top-1]);
        while(dep[LCA]<dep[sta[top-1]]){
            if(dep[LCA]>=dep[sta[top-2]]){
                add_edge(LCA,sta[--top]);
                if(sta[top-1]!=LCA)sta[top++]=LCA;
                break;
            }add_edge(sta[top-2],sta[top-1]);top--;
        }if(sta[top-1]!=a[i])sta[top++]=a[i];
    }
    while(top>1)
        add_edge(sta[top-2],sta[top-1]),top--;
        for(int i=0;i<k;i++)inr[a[i]]=1;
}
```

3.13 点分治 (zky)

```cpp
int siz[maxn],f[maxn],dep[maxn],cant[maxn],root,All,d[maxn];
void makert(int u,int fa){
    siz[u]=1;f[u]=0;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(e.v!=fa&&!cant[e.v]){
            dep[e.v]=dep[u]+1;
            makert(e.v,u);
            siz[u]+=siz[e.v];
            f[u]=max(f[u],siz[e.v]);
        }
    }f[u]=max(f[u],All-f[u]);
    if(f[root]>f[u])root=u;
}
void dfs(int u,int fa){
        //Gain data
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
```

```
        if(e.v==fa||cant[e.v])continue;
        d[e.v]=d[u]+e.w;
        dfs(e.v,u);
    }
}
void calc(int u){
        d[u]=0;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(cant[e.v])continue;
        d[e.v]=e.w;
        dfs(e.v,u);

    }
}
void solve(int u){
    calc(u);cant[u]=1;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(cant[e.v])continue;
        All=siz[e.v];
        f[root=0]=n+1;
        makert(e.v,0);
        solve(root);
    }
}
All=n
f[root=0]=n+1;
makert(1,1);
solve(root);
```

3.14 元芳树

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e4+1e4+233;
const int BIT=18;
int n,m,q;
struct edge{
        int u,v,w;
        bool operator==(edge oth)const{
                return u==oth.u && v==oth.v && w==oth.w;
```

```
        }
        bool operator!=(edge oth)const{
                return !(*this==oth);
        }
};
vector<edge>G[maxn],T[maxn];

int dfn[maxn],low[maxn],tot,rlen[maxn];
bool ins[maxn];
stack<edge>S;
int Rcnt=0;
vector<edge>ring[maxn];
vector<int>bel[maxn],sum[maxn],dis[maxn];
int fa[maxn][BIT];
int dep[maxn],dep2[maxn],fw[maxn];
vector<pair<int,int> >ind[maxn];
map<pair<int,int>,int>Mw;
pair<int,int>pack(int a,int b){
        if(a>b)swap(a,b);
        return make_pair(a,b);
}
void tarjan(int u){
        dfn[u]=low[u]=++tot;
        for(int i=0;i<G[u].size();i++){
                edge e=G[u][i];
                if(dfn[e.v])
                        low[u]=min(low[u],dfn[e.v]);
                else{
                        S.push(e);
                        tarjan(e.v);
                        if(low[e.v]==dfn[u]){

                                if(S.top()==e){
                                        fa[e.v][0]=u;
                                        fw[e.v]=e.w;
                                        S.pop();
                                        continue;
                                }

                                Rcnt++;
                                edge ed;
                                do{
```

```
                    ed=S.top();S.pop();
                    ring[Rcnt].push_back(ed);
                }while(ed!=e);
                    reverse(ring[Rcnt].begin(),ring[Rcnt].end());
                int last=ring[Rcnt].back().v;
                    ring[Rcnt].push_back((edge){last,u,Mw[pack(last,u)]});
                        }
                        low[u]=min(low[u],low[e.v]);
                }
        }
}
void up(int u){
        if(dep[u]||u==1)return ;
        if(fa[u][0])up(fa[u][0]);
        dep[u]=dep[fa[u][0]]+1;
        fw[u]+=fw[fa[u][0]];
}
void build(){
        S.push((edge){0,1,0});
        tarjan(1);

        for(int i=1;i<=Rcnt;i++){
                rlen[i]=0;
                sum[i].resize(ring[i].size());
                dis[i].resize(ring[i].size());
                for(int j=0;j<ring[i].size();j++){
                        rlen[i]+=ring[i][j].w;
                        ind[i].push_back(make_pair(ring[i][j].u,j));
                }
                sum[i][0]=0;
                fw[i+n]=0;
                fa[i+n][0]=ring[i][0].u;
                for(int j=1;j<ring[i].size();j++){
                        sum[i][j]=sum[i][j-1]+ring[i][j-1].w;
                        dis[i][j]=min(sum[i][j],rlen[i]-sum[i][j]);
                        fw[ring[i][j].u]=dis[i][j];
                        fa[ring[i][j].u][0]=i+n;
                }
                sort(ind[i].begin(),ind[i].end());
        }

        for(int i=1;i<=n+Rcnt;i++)
```

```
                up(i);

        for(int j=1;j<BIT;j++)
        for(int i=1;i<=n+Rcnt;i++)if(fa[i][j-1])
                fa[i][j]=fa[fa[i][j-1]][j-1];

}
pair<int,int>second_lca;
int lca(int u,int v){
        if(dep[u]<dep[v])swap(u,v);
        int d=dep[u]-dep[v];
        for(int i=0;i<BIT;i++)if(d>>i&1)
                u=fa[u][i];
        if(u==v)return u;
        for(int i=BIT-1;i>=0;i--)if(fa[u][i]!=fa[v][i]){
                u=fa[u][i];
                v=fa[v][i];
        }
        second_lca=make_pair(u,v);
        return fa[u][0];
}
int main(){

        freopen("bzoj2125.in","r",stdin);

        scanf("%d%d%d",&n,&m,&q);
        for(int i=1;i<=m;i++){
                int u,v,w;scanf("%d%d%d",&u,&v,&w);
                G[u].push_back((edge){u,v,w});
                G[v].push_back((edge){v,u,w});
                Mw[pack(u,v)]=w;
        }

        build();
        while(q--){
                int u,v;
                scanf("%d%d",&u,&v);
                int LCA=lca(u,v);
                if(LCA<=n)printf("%d\n",fw[u]+fw[v]-2*fw[LCA]);
                else{
                        if(dep[u]<dep[v])swap(u,v);
                        int R=LCA-n;
```

```
                        int uu=second_lca.first;
                        int vv=second_lca.second;
                        int ans=fw[u]-fw[uu]+fw[v]-fw[vv];
                        int uid,vid;
                        uid=lower_bound(ind[R].begin(),ind[R].end(),make_pair(uu,-1))->secc
                        vid=lower_bound(ind[R].begin(),ind[R].end(),make_pair(vv,-1))->secc
                        ans+=min(abs(sum[R][uid]-sum[R][vid]),rlen[R]-abs(sum[R][uid]-sum[R
                        printf("%d\n",ans);
                }
        }
        return 0;
}
```

4 图论

4.1 强连通分量

```
int stamp, comps, top;
int dfn[N], low[N], comp[N], stack[N];

void tarjan(int x) {
    dfn[x] = low[x] = ++stamp;
    stack[top++] = x;
    for (int i = 0; i < (int)edge[x].size(); ++i) {
        int y = edge[x][i];
        if (!dfn[y]) {
            tarjan(y);
            low[x] = std::min(low[x], low[y]);
        } else if (!comp[y]) {
            low[x] = std::min(low[x], dfn[y]);
        }
    }
    if (low[x] == dfn[x]) {
        comps++;
        do {
            int y = stack[--top];
            comp[y] = comps;
        } while (stack[top] != x);
    }
}

void solve() {
```

```
    stamp = comps = top = 0;
    std::fill(dfn, dfn + n, 0);
    std::fill(comp, comp + n, 0);
    for (int i = 0; i < n; ++i) {
        if (!dfn[i]) {
            tarjan(i);
        }
    }
}
```

4.1.1 点双连通分量 (lyx)

```
void Dfs(int x, int lst){
        dfn[x] = ++dfc;
        low[x] = dfn[x];
        stack[++cnt] = x;
        int son = 0;

        for (int i = g[x]; i; i = nxt[i]){
                if (!dfn[adj[i]]){
                        ++son;
                        Dfs(adj[i], i);
                        low[x] = min(low[x], low[adj[i]]);
                        if (low[adj[i]] >= dfn[x]){
                                int Tmp;
                                iscut[x] = 1;
                                ++block;
                                E[x].push_back(block + n);
                                do{
                                        Tmp = stack[cnt --];
                                        belong[Tmp] = block + n;

                                        E[Tmp].push_back(block + n);
                                }while (Tmp != adj[i]);
                        }
                }
                else
                if (i  ^ lst != 1) low[x] = min(low[x], dfn[adj[i]]);
        }
        if (x == Root && son == 1) iscut[x] = 0, belong[x] = E[x][0];
        if (x == Root && son == 0){
                ++block;
```

```
                belong[x] = block + n;
        }
}

        tot = 1;//!!!!!!!!!!!!!!!!!!!!!!!!!1
        block = 0;
        cnt = 0;
        for (int i = 1; i <= n * 3; i++){
                g[i] = 0;
                iscut[i] = 0;
                G[i].clear();
                E[i].clear();
                belong[i] = 0;
                size[i] = 0;
                low[i] = dfn[i] =0;
        }
        for (int i = 1; i <= m; i++){
                int x, y;
                scanf("%d%d", &x, &y);
                Ins(x,y);
        }

        dfc = 0;
        for (int i = 1; i <= n; i++)
                if (dfn[i] == 0){
                        Root = i;
                        Dfs(i, 0);
                }
```

4.2 2-SAT 问题

```
int stamp, comps, top;
int dfn[N], low[N], comp[N], stack[N];

void add(int x, int a, int y, int b) {
    edge[x << 1 | a].push_back(y << 1 | b);
}

void tarjan(int x) {
    dfn[x] = low[x] = ++stamp;
    stack[top++] = x;
    for (int i = 0; i < (int)edge[x].size(); ++i) {
```

```
12          int y = edge[x][i];
13          if (!dfn[y]) {
14              tarjan(y);
15              low[x] = std::min(low[x], low[y]);
16          } else if (!comp[y]) {
17              low[x] = std::min(low[x], dfn[y]);
18          }
19      }
20      if (low[x] == dfn[x]) {
21          comps++;
22          do {
23              int y = stack[--top];
24              comp[y] = comps;
25          } while (stack[top] != x);
26      }
27  }
28
29  bool solve() {
30      int counter = n + n + 1;
31      stamp = top = comps = 0;
32      std::fill(dfn, dfn + counter, 0);
33      std::fill(comp, comp + counter, 0);
34      for (int i = 0; i < counter; ++i) {
35          if (!dfn[i]) {
36              tarjan(i);
37          }
38      }
39      for (int i = 0; i < n; ++i) {
40          if (comp[i << 1] == comp[i << 1 | 1]) {
41              return false;
42          }
43          answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
44      }
45      return true;
46  }
```

4.3 二分图最大匹配

4.3.1 Hungary 算法

时间复杂度：$\mathcal{O}(V \cdot E)$

```cpp
vector<int>G[maxn];
int Link[maxn],vis[maxn],T;
bool find(int x){
        for(int i=0;i<G[x].size();i++){
                int v=G[x][i];
                if(vis[v]==T)continue;
                vis[v]=T;
                if(!Link[v]||find(Link[v])){
                        Link[v]=x;
                        return 1;
                }
        }return 0;
}
int Hungarian(int n){
        int ans=0;
        memset(Link,0,sizeof Link);
        for(int i=1;i<=n;i++){
                T++;
                ans+=find(i);
        }return ans;
}
```

4.3.2 Hopcroft Karp 算法

时间复杂度：$\mathcal{O}(\sqrt{V} \cdot E)$

```cpp
int matchx[N], matchy[N], level[N];

bool dfs(int x) {
    for (int i = 0; i < (int)edge[x].size(); ++i) {
        int y = edge[x][i];
        int w = matchy[y];
        if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
            matchx[x] = y;
            matchy[y] = x;
            return true;
        }
    }
    level[x] = -1;
    return false;
}

int solve() {
```

```
    std::fill(matchx, matchx + n, -1);
    std::fill(matchy, matchy + m, -1);
    for (int answer = 0; ; ) {
        std::vector<int> queue;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1) {
                level[i] = 0;
                queue.push_back(i);
            } else {
                level[i] = -1;
            }
        }
        for (int head = 0; head < (int)queue.size(); ++head) {
            int x = queue[head];
            for (int i = 0; i < (int)edge[x].size(); ++i) {
                int y = edge[x][i];
                int w = matchy[y];
                if (w != -1 && level[w] < 0) {
                    level[w] = level[x] + 1;
                    queue.push_back(w);
                }
            }
        }
        int delta = 0;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1 && dfs(i)) {
                delta++;
            }
        }
        if (delta == 0) {
            return answer;
        } else {
            answer += delta;
        }
    }
}
```

4.4 二分图最大权匹配

时间复杂度：$\mathcal{O}(V^4)$

```
int labelx[N], labely[N], match[N], slack[N];
bool visitx[N], visity[N];
```

```
3
4  bool dfs(int x) {
5      visitx[x] = true;
6      for (int y = 0; y < n; ++y) {
7          if (visity[y]) {
8              continue;
9          }
10         int delta = labelx[x] + labely[y] - graph[x][y];
11         if (delta == 0) {
12             visity[y] = true;
13             if (match[y] == -1 || dfs(match[y])) {
14                 match[y] = x;
15                 return true;
16             }
17         } else {
18             slack[y] = std::min(slack[y], delta);
19         }
20     }
21     return false;
22 }
23
24 int solve() {
25     for (int i = 0; i < n; ++i) {
26         match[i] = -1;
27         labelx[i] = INT_MIN;
28         labely[i] = 0;
29         for (int j = 0; j < n; ++j) {
30             labelx[i] = std::max(labelx[i], graph[i][j]);
31         }
32     }
33     for (int i = 0; i < n; ++i) {
34         while (true) {
35             std::fill(visitx, visitx + n, 0);
36             std::fill(visity, visity + n, 0);
37             for (int j = 0; j < n; ++j) {
38                 slack[j] = INT_MAX;
39             }
40             if (dfs(i)) {
41                 break;
42             }
43             int delta = INT_MAX;
44             for (int j = 0; j < n; ++j) {
```

```
                if (!visity[j]) {
                    delta = std::min(delta, slack[j]);
                }
            }
            for (int j = 0; j < n; ++j) {
                if (visitx[j]) {
                    labelx[j] -= delta;
                }
                if (visity[j]) {
                    labely[j] += delta;
                } else {
                    slack[j] -= delta;
                }
            }
        }
    }
    int answer = 0;
    for (int i = 0; i < n; ++i) {
        answer += graph[match[i]][i];
    }
    return answer;
}
```

4.5 最大流 (dinic)

时间复杂度：$\mathcal{O}(V^2 \cdot E)$

```
struct edge{int u,v,cap,flow;};
vector<edge>edges;
vector<int>G[maxn];
int s,t;
int cur[maxn],d[maxn];
void add(int u,int v,int cap){
        edges.push_back((edge){u,v,cap,0});
        G[u].push_back(edges.size()-1);
        edges.push_back((edge){v,u,0,0});
        G[v].push_back(edges.size()-1);
}
bool bfs(){
        static int vis[maxn];
        memset(vis,0,sizeof vis);vis[s]=1;
        queue<int>q;q.push(s);d[s]=0;
        while(!q.empty()){
```

```
                int u=q.front();q.pop();
                for(int i=0;i<G[u].size();i++){
                        edge e=edges[G[u][i]];if(vis[e.v]||e.cap==e.flow)continue;
                        d[e.v]=d[u]+1;vis[e.v]=1;q.push(e.v);
                }
        }return vis[t];
}
int dfs(int u,int a){
        if(u==t||!a)return a;
        int flow=0,f;
        for(int &i=cur[u];i<G[u].size();i++){
                edge e=edges[G[u][i]];
                if(d[e.v]==d[u]+1&&(f=dfs(e.v,min(a,e.cap-e.flow)))>0){
                        edges[G[u][i]].flow+=f;
                        edges[G[u][i]^1].flow-=f;
                        flow+=f;a-=f;if(!a)break;
                }
        }return flow;
}
int dinic(){
        int flow=0,x;
        while(bfs()){
                memset(cur,0,sizeof cur);
                while(x=dfs(s,INT_MAX)){
                        flow+=x;
                        memset(cur,0,sizeof cur);
                }
        }return flow;
}
```

4.6 最大流 (sap)

时间复杂度：$\mathcal{O}(V^2 \cdot E)$

```
int g[T], adj[M], nxt[M], f[M];
int cnt[T], dist[T], cur[T], fa[T], dat[T];
void Ins(int x, int y, int ff, int rf){
        adj[++tot] = y; nxt[tot] = g[x]; g[x] = tot; f[tot] = ff;
        adj[++tot] = x; nxt[tot] = g[y]; g[y] = tot; f[tot] = rf;
}
int sap(int s, int t){
        int x, sum;
        for (int i = 1; i <= t; i++){
```

```
                dist[i] = 1;
                cur[i] = g[i];
                fa[i] = 0;
                dat[i] = 0;
                cnt[i] = 0;
        }
        cnt[0] = 1; cnt[1] = t - 1;
        dist[t] = 0;
        dat[s] = INF;
        x = s;
        sum = 0;
        while (1){
                int p;
                for (p = cur[x]; p; p = nxt[p]){
                        if (f[p] > 0 && dist[adj[p]] == dist[x] - 1) break;
                }
                if (p > 0){
                        cur[x] = p;
                        fa[adj[p]] = p;
                        dat[adj[p]] = min(dat[x], f[p]);
                        x = adj[p];
                        if (x == t){
                                sum += dat[x];
                                while (x != s){
                                        f[fa[x]] -= dat[t];
                                        f[fa[x] ^ 1] += dat[t];
                                        x = adj[fa[x] ^ 1];
                                 }
                        }
                } else {
                        cnt[dist[x]] --;
                        if (cnt[dist[x]] == 0) return sum;
                        dist[x] = t + 1;
                        for (int p = g[x]; p; p = nxt[p]){
                                if (f[p] > 0 && dist[adj[p]] + 1 < dist[x]){
                                        dist[x] = dist[adj[p]] + 1;
                                        cur[x] = p;
                                }
                        }
                        cnt[dist[x]]++;
                        if (dist[s] > t) return sum;
                        if (x != s) x = adj[fa[x] ^ 1];
```

```
                }
        }
}
/*
tot = 1
edges' id start from 2
remember to clean g
t is the number of points
*/
```

4.7 上下界网络流

$B(u,v)$ 表示边 (u,v) 流量的下界，$C(u,v)$ 表示边 (u,v) 流量的上界，$F(u,v)$ 表示边 (u,v) 的流量。设 $G(u,v)=F(u,v)-B(u,v)$，显然有

$$0 \le G(u,v) \le C(u,v)-B(u,v)$$

4.7.1 无源汇的上下界可行流

建立超级源点 S^* 和超级汇点 T^*，对于原图每条边 (u,v) 在新网络中连如下三条边：$S^* \to v$，容量为 $B(u,v)$；$u \to T^*$，容量为 $B(u,v)$；$u \to v$，容量为 $C(u,v)-B(u,v)$。最后求新网络的最大流，判断从超级源点 S^* 出发的边是否都满流即可，边 (u,v) 的最终解中的实际流量为 $G(u,v)+B(u,v)$。

4.7.2 有源汇的上下界可行流

从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边。按照无源汇的上下界可行流一样做即可，流量即为 $T \to S$ 边上的流量。

4.7.3 有源汇的上下界最大流

1. 在有源汇的上下界可行流中，从汇点 T 到源点 S 的边改为连一条上界为 ∞，下届为 x 的边。x 满足二分性质，找到最大的 x 使得新网络存在无源汇的上下界可行流即为原图的最大流。

2. 从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边，变成无源汇的网络。按照无源汇的上下界可行流的方法，建立超级源点 S^* 和超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S \to T$ 的最大流即可。

4.7.4 有源汇的上下界最小流

1. 在有源汇的上下界可行流中，从汇点 T 到源点 S 的边改为连一条上界为 x，下界为 0 的边。x 满足二分性质，找到最小的 x 使得新网络存在无源汇的上下界可行流即为原图的最小流。

2. 按照无源汇的上下界可行流的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，但是注意这一次不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0，所以 S^*，T^* 无影响，再直接求一次 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \to S$ 边上的流量即为原图的最小流，否则无解。

4.8 最小费用最大流

4.8.1 稀疏图

时间复杂度：$\mathcal{O}(V \cdot E^2)$

```
struct EdgeList {
    int size;
    int last[N];
    int succ[M], other[M], flow[M], cost[M];
    void clear(int n) {
        size = 0;
        std::fill(last, last + n, -1);
    }
    void add(int x, int y, int c, int w) {
        succ[size] = last[x];
        last[x] = size;
        other[size] = y;
        flow[size] = c;
        cost[size++] = w;
    }
} e;

int n, source, target;
int prev[N];

void add(int x, int y, int c, int w) {
    e.add(x, y, c, w);
    e.add(y, x, 0, -w);
}

bool augment() {
    static int dist[N], occur[N];
    std::vector<int> queue;
    std::fill(dist, dist + n, INT_MAX);
    std::fill(occur, occur + n, 0);
    dist[source] = 0;
```

```
    occur[source] = true;
    queue.push_back(source);
    for (int head = 0; head < (int)queue.size(); ++head) {
        int x = queue[head];
        for (int i = e.last[x]; ~i; i = e.succ[i]) {
            int y = e.other[i];
            if (e.flow[i] && dist[y] > dist[x] + e.cost[i]) {
                dist[y] = dist[x] + e.cost[i];
                prev[y] = i;
                if (!occur[y]) {
                    occur[y] = true;
                    queue.push_back(y);
                }
            }
        }
        occur[x] = false;
    }
    return dist[target] < INT_MAX;
}

std::pair<int, int> solve() {
    std::pair<int, int> answer = std::make_pair(0, 0);
    while (augment()) {
        int number = INT_MAX;
        for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
            number = std::min(number, e.flow[prev[i]]);
        }
        answer.first += number;
        for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
            e.flow[prev[i]] -= number;
            e.flow[prev[i] ^ 1] += number;
            answer.second += number * e.cost[prev[i]];
        }
    }
    return answer;
}
```

4.8.2 稠密图

使用条件：费用非负

时间复杂度：$\mathcal{O}(V \cdot E^2)$

```
struct EdgeList {
    int size;
    int last[N];
    int succ[M], other[M], flow[M], cost[M];
    void clear(int n) {
        size = 0;
        std::fill(last, last + n, -1);
    }
    void add(int x, int y, int c, int w) {
        succ[size] = last[x];
        last[x] = size;
        other[size] = y;
        flow[size] = c;
        cost[size++] = w;
    }
} e;

int n, source, target, flow, cost;
int slack[N], dist[N];
bool visit[N];

void add(int x, int y, int c, int w) {
    e.add(x, y, c, w);
    e.add(y, x, 0, -w);
}

bool relabel() {
    int delta = INT_MAX;
    for (int i = 0; i < n; ++i) {
        if (!visit[i]) {
            delta = std::min(delta, slack[i]);
        }
        slack[i] = INT_MAX;
    }
    if (delta == INT_MAX) {
        return true;
    }
    for (int i = 0; i < n; ++i) {
        if (visit[i]) {
            dist[i] += delta;
        }
    }
```

```
    return false;
}

int dfs(int x, int answer) {
    if (x == target) {
        flow += answer;
        cost += answer * (dist[source] - dist[target]);
        return answer;
    }
    visit[x] = true;
    int delta = answer;
    for (int i = e.last[x]; ~i; i = e.succ[i]) {
        int y = e.other[i];
        if (e.flow[i] > 0 && !visit[y]) {
            if (dist[y] + e.cost[i] == dist[x]) {
                int number = dfs(y, std::min(e.flow[i], delta));
                e.flow[i] -= number;
                e.flow[i ^ 1] += number;
                delta -= number;
                if (delta == 0) {
                    dist[x] = INT_MIN;
                    return answer;
                }
            } else {
                slack[y] = std::min(slack[y], dist[y] + e.cost[i] - dist[x]);
            }
        }
    }
    return answer - delta;
}

std::pair<int, int> solve() {
    flow = cost = 0;
    std::fill(dist, dist + n, 0);
    do {
        do {
            fill(visit, visit + n, 0);
        } while (dfs(source, INT_MAX));
    } while (!relabel());
    return std::make_pair(flow, cost);
}
```

4.9 一般图最大匹配

时间复杂度：$\mathcal{O}(V^3)$

```cpp
int match[N], belong[N], next[N], mark[N], visit[N];
std::vector<int> queue;

int find(int x) {
    if (belong[x] != x) {
        belong[x] = find(belong[x]);
    }
    return belong[x];
}

void merge(int x, int y) {
    x = find(x);
    y = find(y);
    if (x != y) {
        belong[x] = y;
    }
}

int lca(int x, int y) {
    static int stamp = 0;
    stamp++;
    while (true) {
        if (x != -1) {
            x = find(x);
            if (visit[x] == stamp) {
                return x;
            }
            visit[x] = stamp;
            if (match[x] != -1) {
                x = next[match[x]];
            } else {
                x = -1;
            }
        }
        std::swap(x, y);
    }
}

void group(int a, int p) {
```

```
    while (a != p) {
        int b = match[a], c = next[b];
        if (find(c) != p) {
            next[c] = b;
        }
        if (mark[b] == 2) {
            mark[b] = 1;
            queue.push_back(b);
        }
        if (mark[c] == 2) {
            mark[c] = 1;
            queue.push_back(c);
        }
        merge(a, b);
        merge(b, c);
        a = c;
    }
}

void augment(int source) {
    queue.clear();
    for (int i = 0; i < n; ++i) {
        next[i] = visit[i] = -1;
        belong[i] = i;
        mark[i] = 0;
    }
    mark[source] = 1;
    queue.push_back(source);
    for (int head = 0; head < (int)queue.size() && match[source] == -1; ++head) {
        int x = queue[head];
        for (int i = 0; i < (int)edge[x].size(); ++i) {
            int y = edge[x][i];
            if (match[x] == y || find(x) == find(y) || mark[y] == 2) {
                continue;
            }
            if (mark[y] == 1) {
                int r = lca(x, y);
                if (find(x) != r) {
                    next[x] = y;
                }
                if (find(y) != r) {
                    next[y] = x;
```

```
                }
                group(x, r);
                group(y, r);
            } else if (match[y] == -1) {
                next[y] = x;
                for (int u = y; u != -1; ) {
                    int v = next[u];
                    int mv = match[v];
                    match[v] = u;
                    match[u] = v;
                    u = mv;
                }
                break;
            } else {
                next[y] = x;
                mark[y] = 2;
                mark[match[y]] = 1;
                queue.push_back(match[y]);
            }
        }
    }
}

int solve() {
    std::fill(match, match + n, -1);
    for (int i = 0; i < n; ++i) {
        if (match[i] == -1) {
            augment(i);
        }
    }
    int answer = 0;
    for (int i = 0; i < n; ++i) {
        answer += (match[i] != -1);
    }
    return answer;
}
```

4.10 无向图全局最小割

时间复杂度：$\mathcal{O}(V^3)$

注意事项：处理重边时，应该对边权累加

```cpp
int node[N], dist[N];
bool visit[N];

int solve(int n) {
    int answer = INT_MAX;
    for (int i = 0; i < n; ++i) {
        node[i] = i;
    }
    while (n > 1) {
        int max = 1;
        for (int i = 0; i < n; ++i) {
            dist[node[i]] = graph[node[0]][node[i]];
            if (dist[node[i]] > dist[node[max]]) {
                max = i;
            }
        }
        int prev = 0;
        memset(visit, 0, sizeof(visit));
        visit[node[0]] = true;
        for (int i = 1; i < n; ++i) {
            if (i == n - 1) {
                answer = std::min(answer, dist[node[max]]);
                for (int k = 0; k < n; ++k) {
                    graph[node[k]][node[prev]] =
                        (graph[node[prev]][node[k]] += graph[node[k]][node[max]]);
                }
                node[max] = node[--n];
            }
            visit[node[max]] = true;
            prev = max;
            max = -1;
            for (int j = 1; j < n; ++j) {
                if (!visit[node[j]]) {
                    dist[node[j]] += graph[node[prev]][node[j]];
                    if (max == -1 || dist[node[max]] < dist[node[j]]) {
                        max = j;
                    }
                }
            }
        }
    }
    return answer;
```

```
}
```

4.11 有根树的同构

时间复杂度：$\mathcal{O}(VlogV)$

```
const unsigned long long MAGIC = 4423;

unsigned long long magic[N];
std::pair<unsigned long long, int> hash[N];

void solve(int root) {
    magic[0] = 1;
    for (int i = 1; i <= n; ++i) {
        magic[i] = magic[i - 1] * MAGIC;
    }
    std::vector<int> queue;
    queue.push_back(root);
    for (int head = 0; head < (int)queue.size(); ++head) {
        int x = queue[head];
        for (int i = 0; i < (int)son[x].size(); ++i) {
            int y = son[x][i];
            queue.push_back(y);
        }
    }
    for (int index = n - 1; index >= 0; --index) {
        int x = queue[index];
        hash[x] = std::make_pair(0, 0);

        std::vector<std::pair<unsigned long long, int> > value;
        for (int i = 0; i < (int)son[x].size(); ++i) {
            int y = son[x][i];
            value.push_back(hash[y]);
        }
        std::sort(value.begin(), value.end());

        hash[x].first = hash[x].first * magic[1] + 37;
        hash[x].second++;
        for (int i = 0; i < (int)value.size(); ++i) {
            hash[x].first = hash[x].first * magic[value[i].second] + value[i].first;
            hash[x].second += value[i].second;
        }
        hash[x].first = hash[x].first * magic[1] + 41;
```

```
        hash[x].second++;
    }
}
```

4.12 哈密尔顿回路（ORE 性质的图）

ORE 性质：

$$\forall x, y \in V \wedge (x, y) \notin E \quad s.t. \quad deg_x + deg_y \geq n$$

返回结果：从顶点 1 出发的一个哈密尔顿回路

使用条件：$n \geq 3$

```
int left[N], right[N], next[N], last[N];

void cover(int x) {
    left[right[x]] = left[x];
    right[left[x]] = right[x];
}

int adjacent(int x) {
    for (int i = right[0]; i <= n; i = right[i]) {
        if (graph[x][i]) {
            return i;
        }
    }
    return 0;
}

std::vector<int> solve() {
    for (int i = 1; i <= n; ++i) {
        left[i] = i - 1;
        right[i] = i + 1;
    }
    int head, tail;
    for (int i = 2; i <= n; ++i) {
        if (graph[1][i]) {
            head = 1;
            tail = i;
            cover(head);
            cover(tail);
            next[head] = tail;
            break;
        }
```

```
    }
    while (true) {
        int x;
        while (x = adjacent(head)) {
            next[x] = head;
            head = x;
            cover(head);
        }
        while (x = adjacent(tail)) {
            next[tail] = x;
            tail = x;
            cover(tail);
        }
        if (!graph[head][tail]) {
            for (int i = head, j; i != tail; i = next[i]) {
                if (graph[head][next[i]] && graph[tail][i]) {
                    for (j = head; j != i; j = next[j]) {
                        last[next[j]] = j;
                    }
                    j = next[head];
                    next[head] = next[i];
                    next[tail] = i;
                    tail = j;
                    for (j = i; j != head; j = last[j]) {
                        next[j] = last[j];
                    }
                    break;
                }
            }
        }
        next[tail] = head;
        if (right[0] > n) {
            break;
        }
        for (int i = head; i != tail; i = next[i]) {
            if (adjacent(i)) {
                head = next[i];
                tail = i;
                next[tail] = 0;
                break;
            }
        }
```

```
    }
    std::vector<int> answer;
    for (int i = head; ; i = next[i]) {
        if (i == 1) {
            answer.push_back(i);
            for (int j = next[i]; j != i; j = next[j]) {
                answer.push_back(j);
            }
            answer.push_back(i);
            break;
        }
        if (i == tail) {
            break;
        }
    }
    return answer;
}
```

4.13 必经点树

```
vector<int>G[maxn],rG[maxn],dom[maxn];
int n,m;
int dfn[maxn],rdfn[maxn],dfs_c,semi[maxn],idom[maxn],fa[maxn];
struct ufsets{
    int fa[maxn],best[maxn];
    int find(int x){
        if(fa[x]==x)
            return x;
        int f=find(fa[x]);
        if(dfn[semi[best[x]]]>dfn[semi[best[fa[x]]]])
            best[x]=best[fa[x]];
        fa[x]=f;
        return f;
    }
    int getbest(int x){
        find(x);
        return best[x];
    }
    void init(){
        for(int i=1;i<=n;i++)
            fa[i]=best[i]=i;
    }
```

```
}uf;
void init(){
    uf.init();
    for(int i=1;i<=n;i++){
        semi[i]=i;
        idom[i]=0;
        fa[i]=0;
        dfn[i]=rdfn[i]=0;
    }
    dfs_c=0;
}
void dfs(int u){
    dfn[u]=++dfs_c;
    rdfn[dfn[u]]=u;
    for(int i=0;i<G[u].size();i++){
        int v=G[u][i];
        if(!dfn[v]){
            fa[v]=u;
            dfs(v);
        }
    }
}

void tarjan(){
    for(int i=n;i>1;i--){
        int tmp=1e9;
        int y=rdfn[i];
        for(int i=0;i<rG[y].size();i++){
            int x=rG[y][i];
            tmp=min(tmp,dfn[semi[uf.getbest(x)]]);
        }
        semi[y]=rdfn[tmp];
        int x=fa[y];
        dom[semi[y]].push_back(y);
        uf.fa[y]=x;
        for(int i=0;i<dom[x].size();i++){
            int z=dom[x][i];
            if(dfn[semi[uf.getbest(z)]]<dfn[x])
                idom[z]=uf.getbest(z);
            else
                idom[z]=semi[z];
        }
```

```
        dom[x].clear();
    }
    semi[rdfn[1]]=1;
    for(int i=2;i<=n;i++){
        int x=rdfn[i];
        if(idom[x]!=semi[x])
            idom[x]=idom[idom[x]];

    }
    idom[rdfn[1]]=0;
}
init();
dfs(1);
tarjan();
```

5 字符串

5.1 模式匹配

5.1.1 KMP 算法

```
void build(char *pattern) {
    int length = (int)strlen(pattern + 1);
    fail[0] = -1;
    for (int i = 1, j; i <= length; ++i) {
        for (j = fail[i - 1]; j != -1 && pattern[i] != pattern[j + 1]; j = fail[j]);
        fail[i] = j + 1;
    }
}

void solve(char *text, char *pattern) {
    int length = (int)strlen(text + 1);
    for (int i = 1, j; i <= length; ++i) {
        for (j = match[i - 1]; j != -1 && text[i] != pattern[j + 1]; j = fail[j]);
        match[i] = j + 1;
    }
}
///Hint: 1 - Base
```

5.1.2 扩展 KMP 算法

返回结果：

$$next_i = lcp(text, text_{i...n-1})$$

```
void solve(char *text, int length, int *next) {
    int j = 0, k = 1;
    for (; j + 1 < length && text[j] == text[j + 1]; j++);
    next[0] = length - 1;
    next[1] = j;
    for (int i = 2; i < length; ++i) {
        int far = k + next[k] - 1;
        if (next[i - k] < far - i + 1) {
            next[i] = next[i - k];
        } else {
            j = std::max(far - i + 1, 0);
            for (; i + j < length && text[j] == text[i + j]; j++);
            next[i] = j;
            k = i;
        }
    }
}
/// 0 - Base
```

5.1.3 AC 自动机

```
struct Node{
        int Next[30], fail, mark;
}Tree[N];

void Init(){
        memset(Tree, 0, sizeof Tree);
        cnt = 1;

        for (int i = 1; i <= n; i++){
                char c;
                int now = 1;
                scanf("%s", s + 1);
                int Length = strlen(s + 1);
                for (int j = 1; j <= Length; j++){
                        c = s[j];
                        if (Tree[now].Next[c - 'a']) now = Tree[now].Next[c - 'a']; else
                                Tree[now].Next[c - 'a'] = ++ cnt, now = cnt;
```

```
                }
        }
}

void Build_Ac(){
        int en = 0;
        Q[0] = 1;
        for (int fi = 0; fi <= en; fi++){
                int now = Q[fi];
                for (int next = 0; next < 26; next++)
                        if (Tree[now].Next[next])
                        {
                                int k = Tree[now].Next[next];
                                if (now == 1) Tree[k].fail = 1; else
                                {
                                        int h = Tree[now].fail;
                                        while (h && !Tree[h].Next[next]) h = Tree[h].fail;
                                        if (!h) Tree[k].fail = 1;
                                        else Tree[k].fail = Tree[h].Next[next];
                                }
                                Q[++ en] = k;
                        }
        }
}

/// Hints : when not match , fail = 1
```

5.2 后缀三姐妹

5.2.1 后缀数组

```
struct Sa{
        int heap[N],s[N],sa[N],r[N],tr[N],sec[N],m,cnt;
        int h[19][N];

        void Prep(){
                for (int i=1; i<=m; i++) heap[i]=0;
                for (int i=1; i<=n; i++) heap[s[i]]++;
                for (int i=2; i<=m; i++) heap[i]+=heap[i-1];
                for (int i=n; i>=1; i--) sa[heap[s[i]]--]=i;
                r[sa[1]]=1; cnt=1;
                for (int i=2; i<=n; i++){
```

```
                if (s[sa[i]]!=s[sa[i-1]]) cnt++;
                r[sa[i]]=cnt;
        }
        m=cnt;
    }

    void Suffix(){
        int j=1;
        while (cnt<n){
            cnt=0;
            for (int i=n-j+1; i<=n; i++) sec[++cnt]=i;
            for (int i=1; i<=n; i++) if (sa[i]>j)
                sec[++cnt]=sa[i]-j;
            for (int i=1; i<=n; i++) tr[i]=r[sec[i]];
            for (int i=1; i<=m; i++) heap[i]=0;
            for (int i=1; i<=n; i++) heap[tr[i]]++;
            for (int i=2; i<=m; i++) heap[i]+=heap[i-1];
            for (int i=n; i>=1; i--)
                sa[heap[tr[i]]--]=sec[i];
            tr[sa[1]]=1; cnt=1;
            for (int i=2; i<=n; i++){
                if ((r[sa[i]]!=r[sa[i-1]]) || (r[sa[i]+j]!=r[sa[i-1]+j]))
                    cnt++;
                tr[sa[i]]=cnt;
            }
            for (int i=1; i<=n; i++) r[i]=tr[i];
            m=cnt; j=j+j;
        }
    }

    void Calc(){
        int k=0;
        for (int i=1; i<=n; i++){
            if (r[i]==1) continue;
            int j=sa[r[i]-1];
            while ((i+k<=n) && (j+k<=n) && (s[i+k]==s[j+k])) k++;
            h[0][r[i]]=k;
            if (k) k--;
        }
        for (int i=1; i<19; i++)
            for (int j=1; j+(1 << i)-1<=n; j++)
                h[i][j]=min(h[i-1][j],h[i-1][j + (1 << (i - 1)) + 1]);
```

```
        }

        int Query(int L,int R){
                L=r[L], R=r[R];
                if (L>R) swap(L,R);
                L++;
                int l0 = Lg[R-L+1];
                return min(h[l0][L],h[l0][R-(1 << l0)+1]);
        }

        void Work(){
                Prep(); Suffix(); Calc();
        }
}P,S;

/// Hints : 1 - Base
```

5.2.2 后缀数组 (dc3)

```
//`DC3 待排序的字符串放在 r 数组中，从 r[0] 到 r[n-1]，长度为 n，且最大值小于 m.`
//`约定除 r[n-1] 外所有的 r[i] 都大于 0, r[n-1]=0。`
//`函数结束后，结果放在 sa 数组中，从 sa[0] 到 sa[n-1]。`
//`r 必须开长度乘 3`
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)

int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b)
{
        return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
}
int c12(int k,int *r,int a,int b)
{
        if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
        else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m)
{
        int i;
        for(i=0;i<n;i++) wv[i]=r[a[i]];
```

```
        for(i=0;i<m;i++) wss[i]=0;
        for(i=0;i<n;i++) wss[wv[i]]++;
        for(i=1;i<m;i++) wss[i]+=wss[i-1];
        for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m)
{
        int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
        r[n]=r[n+1]=0;
        for(i=0;i<n;i++)
                if(i%3!=0) wa[tbc++]=i;
        sort(r+2,wa,wb,tbc,m);
        sort(r+1,wb,wa,tbc,m);
        sort(r,wa,wb,tbc,m);
        for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
                rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
        if (p<tbc) dc3(rn,san,tbc,p);
        else for (i=0;i<tbc;i++) san[rn[i]]=i;
        for (i=0;i<tbc;i++)
                if(san[i]<tb) wb[ta++]=san[i]*3;
        if(n%3==1) wb[ta++]=n-1;
        sort(r,wb,wa,ta,m);
        for(i=0;i<tbc;i++)
                wv[wb[i]=G(san[i])]=i;
        for(i=0,j=0,p=0;i<ta && j<tbc;p++)
                sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
        for(;i<ta;p++) sa[p]=wa[i++];
        for(;j<tbc;p++) sa[p]=wb[j++];
}

int main(){
        int n,m=0;
        scanf("%d",&n);
        for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
        printf("%d\n",m);
        s[n++]=0;
        dc3(s,sa,n,m);
        for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
}
```

5.2.3 后缀自动机-多串 LCS

对一个串建后缀自动机，其他串在上面匹配，因为是求所有串的公共子串，所以每个点记录每个串最长匹配长度的最小值，最后找到所有点中最长的一个即可。一个注意事项就是，当走到一个点时，还要更新它的 parent 树上的祖先的匹配长度，数组开两倍啦啦啦！

```
struct Node{
        int len, fail;
        int To[30];
}T[N];
int Lst, Root, tot, ans;
char s[N];
int Len[N], Ans[N], Ord[N];
void Add(int x, int l){
        int Nt = ++tot, p = Lst;
        T[Nt].len = l;
        for (;p && !T[p].To[x]; p = T[p].fail) T[p].To[x] = Nt;
        if (!p) T[Nt].fail = Root; else
        if (T[T[p].To[x]].len == T[p].len + 1) T[Nt].fail = T[p].To[x];
        else{
                int q = ++tot, qt = T[p].To[x];
                T[q] = T[qt];
                T[q].len = T[p].len + 1;
                T[qt].fail = T[Nt].fail = q;
                for (;p && T[p].To[x] == qt; p = T[p].fail) T[p].To[x] = q;
        }
        Lst = Nt;
}
bool cmp(int a, int b){
        return T[a].len < T[b].len;
}
int main(){
        scanf("%s", s + 1);
        int n = strlen(s + 1);
        ans = n;
        Root = tot = Lst = 1;
        for (int i = 1; i <= n; i++)
                Add(s[i] - 'a' + 1, i);
        for (int i = 1; i <= tot; i++)
                Ord[i] = i;
        sort(Ord + 1, Ord + tot + 1, cmp);
        for (int i = 1; i <= tot; i++)
                Ans[i] = T[i].len;
```

```
        bool flag = 0;
        while (scanf("%s", s + 1) != EOF){
                flag = 1;
                int n = strlen(s + 1);
                int p = Root, len = 0;
                for (int i = 1; i <= tot; i++) Len[i] = 0;
                for (int i = 1; i <= n; i++){
                        int x = s[i] - 'a' + 1;
                        if (T[p].To[x]) len++, p = T[p].To[x];
                        else {
                                while (p && !T[p].To[x]) p = T[p].fail;
                                if (!p) p = Root, len = 0;
                                else len = T[p].len + 1, p = T[p].To[x];
                        }
                        Len[p] = max(Len[p], len);
                }
                for (int i = tot; i >= 1; i--){
                        int Cur = Ord[i];
                        Ans[Cur] = min(Ans[Cur], Len[Cur]);
                        if (Len[Cur] && T[Cur].fail)
                                Len[T[Cur].fail] = T[T[Cur].fail].len;
                }
        }
        if (flag){
                ans = 0;
                for (int i = 1; i <= tot; i++){
                        ans = max(ans, Ans[i]);
                }
        }
        printf("%d\n", ans);
        return 0;
}
```

5.2.4 后缀自动机-各长度字串出现次数最大值

给一个字符串 S，令 F(x) 表示 S 的所有长度为 x 的子串中，出现次数的最大值。 构建字符串的自动机，对于每个节点，right 集合大小就是出现次数，maxs 就是它代表的最长长度，那么我们用 |right(x)| 去更新 f[maxs[x]] 的值，最后从大到小用 f[i] 去更新 f[i-1] 的值即可

```
struct Node{
        int len, fail;
        int To[30];
```

```
}T[N];
int Lst, Root, tot, n;
char s[N];
int Ord[N], Ans[N], Ways[N], heap[N];
void Add(int x, int l){
        int Nt = ++tot, p = Lst;
        T[Nt].len = l;
        for (;p && !T[p].To[x]; p = T[p].fail) T[p].To[x] = Nt;
        if (!p) T[Nt].fail = Root; else
        if (T[T[p].To[x]].len == T[p].len + 1) T[Nt].fail = T[p].To[x];
        else{
                int q = ++tot, qt = T[p].To[x];
                T[q] = T[qt];
                T[q].len = T[p].len + 1;
                T[qt].fail = T[Nt].fail = q;
                for (;p && T[p].To[x] == qt; p = T[p].fail) T[p].To[x] = q;
        }
        Lst = Nt;
}
bool cmp(int a, int b){
        return T[a].len < T[b].len;
}
void sort(){
        for (int i = 1; i <= tot; i++) heap[T[i].len]++;
        for (int i = 1; i <= n; i++) heap[i] += heap[i-1];
        for (int i = 1; i <= tot; i++) Ord[heap[T[i].len]--]=i;
}
int main(){
        scanf("%s", s + 1);
        n = strlen(s + 1);
        Root = tot = Lst = 1;
        for (int i = 1; i <= n; i++)
                Add(s[i] - 'a' + 1, i);
        sort();
        memset(Ways , 0, sizeof(Ways));
        for (int i = 1, p = Root; i <= n; i++)
                p = T[p].To[s[i] - 'a' + 1], Ways[p] = 1;
        for (int i = tot; i >= 1; i--){
                int Cur = Ord[i];
                if (T[Cur].fail == 0) continue;
                Ways[T[Cur].fail] += Ways[Cur];
        }
```

```
        for (int i = 1; i <= tot; i++)
                Ans[T[i].len] = max(Ans[T[i].len], Ways[i]);
        for (int i = n; i >= 1; i--)
                Ans[i] = max(Ans[i + 1], Ans[i]);
        for (int i = 1; i <= n; i++)
                printf("%d\n", Ans[i]);
        return 0;
}
```

5.2.5 后缀自动机-两串 LCS

```
struct node{
        int len, fail;
        int To[27];
}T[N];
char a[N], b[N];
int Lst, Root, tot;
void add(int x, int l){
        int Nt = ++tot, p = Lst;
        T[Nt].len = l;
        for (;p && !T[p].To[x]; p = T[p].fail) T[p].To[x] = Nt;
        if (!p) T[Nt].fail = Root;
        else
        if (T[T[p].To[x]].len == T[p].len + 1) T[Nt].fail = T[p].To[x];
        else{
                int q = ++tot, qt = T[p].To[x];
                T[q] = T[qt];
                T[q].len  = T[p].len + 1;
                T[qt].fail = T[Nt].fail = q;
                for (;p && T[p].To[x] == qt; p = T[p].fail) T[p].To[x] = q;
        }
        Lst = Nt;
}
int main(){
        while (scanf("%s%s", a + 1, b + 1) == 2){
                int n = strlen(a + 1);
                Lst = Root = tot = 1;
                for (int i = 1; i <= n; i++)
                        add(a[i] - 'a' + 1, i);
                int m = strlen(b + 1);
                int p = Root, len = 0;
                int Ans = 0;
```

```
                for (int i = 1; i <= m; i++){
                        int x = b[i] - 'a' + 1;
                        if (T[p].To[x]) len++, p = T[p].To[x];
                        else {
                                while (p && !T[p].To[x]) p = T[p].fail;
                                if (!p) p = Root, len = 0;
                                else len = T[p].len + 1, p = T[p].To[x];
                        }
                        if (len > Ans) Ans = len;
                }
                printf("%d\n", Ans);
                for (int i = 1; i <= tot; i++){
                        T[i].len = T[i].fail = 0;
                        for (int j = 1; j <= 26; j++)
                                T[i].To[j] = 0;
                }
        }
        return 0;
}
//Hints£ºSAM + Longest common subsequence
```

5.3 回文三兄弟

5.3.1 马拉车

```
void Manacher(){
        R[1] = 1;
        for (int i = 2, j = 1; i <= length; i++){
                if (j + R[j] <= i){
                        R[i] = 0;
                } else {
                        R[i] = min(R[j * 2 - i], j + R[j] - i);
                }
                while (i - R[i] >= 1 && i + R[i] <= length
                        && text[i - R[i]] == text[i + R[i]]){
                        R[i]++;
                }
                if (i + R[i] > j + R[j]){
                        j = i;
                }
        }
}
```

```
        length = 0;
        int n = strlen(s + 1);
        for (int i = 1; i <= n; i++){
                text[++length] = '*';
                text[++length] = s[i];
        }
        text[++length] = '*';
/// Hints: 1 - Base
```

5.3.2 回文树 (lyx)

```
const int N = 400005;

char s[N];
int Len;

struct Palindromic_Tree {
    int next[N][27];
    int fail[N];
    int cnt[N];
    int num[N];
    int len[N];
    char S[N];
    int last;
    int n;
    int p;

    int newnode(int l)
    {
        for(int i = 1; i <= 26; i++) next[p][i] = 0;
        cnt[p] = 0;
        num[p] = 0;
        len[p] = l;
                fail[p] = 0;
        return p++;
    }
    void init()
    {
        p = 0;
        newnode(0);
        newnode(-1);
        last = 0;
```

```
        n = 0;
        S[n] = -1;
        fail[0] = 1;
    }
    int get_fail(int x)
    {
        while (S[n - len[x] - 1] != S[n]) x = fail[x];
        return x;
    }
    void add(char c, int pos)
    {
        c = c - 'a' + 1;
        S[++ n] = c ;
        int cur = get_fail(last);
        if (!next[cur][c])
        {
            int now = newnode(len[cur] + 2);
            fail[now] = next[get_fail(fail[cur])][c];
            next[cur][c] = now;
            num[now] = num[fail[now]] + 1;
        }
                last = next[cur][c] ;
        cnt[last]++;
    }
    void count()
    {
        for (int i = p - 1 ; i >= 0 ; -- i) cnt[fail[i]] += cnt[i] ;
    }
}T;

Len = strlen(s + 1);

T.init();
for (int i = 1; i <= Len; i++)
        T.add(s[i], i);
T.count();
```

5.3.3 回文自动机 (zky)

```
struct PAM{
        int tot,last,str[maxn],nxt[maxn][26],n;
        int len[maxn],suf[maxn],cnt[maxn];
```

```
        int newnode(int l){
                len[tot]=l;
                return tot++;
        }
        void init(){
                tot=0;
                newnode(0);// tree0 is node 0
                newnode(-1);// tree-1 is node 1
                str[0]=-1;
                suf[0]=1;
        }
        int find(int x){
                while(str[n-len[x]-1]!=str[n])x=suf[x];
                return x;
        }
        void add(int c){
                str[++n]=c;
                int u=find(last);
                if(!nxt[u][c]){
                        int v=newnode(len[u]+2);
                        suf[v]=nxt[find(suf[u])][c];
                        nxt[u][c]=v;
                }last=nxt[u][c];
                cnt[last]++;
        }
        void count(){
                for(int i=tot-1;i>=0;i--)cnt[suf[i]]+=cnt[i];
        }
}P;
int main(){
        P.init();
        for(int i=0;i<n;i++)
                P.add(s[i]-'a');
        P.count();
```

5.4 循环串最小表示

```
string sol(char *s){
    int n=strlen(s);
    int i=0,j=1,k=0,p;
    while(i<n&&j<n&&k<n){
        int t=s[(i+k)%n]-s[(j+k)%n];
```

```
        if(t==0)k++;
        else if(t<0)j+=k+1,k=0;
        else i+=k+1,k=0;
        if(i==j)j++;
    }p=min(i,j);
    string S;
    for(int i=p;i<p+n;i++)S.push_back(s[i%n]);
    return S;
}
```

6 计算几何

6.1 二维基础

6.1.1 点类

```
int sgn(double x){return (x>eps)-(x<-eps);}
int sgn(double a,double b){return sgn(a-b);}
double sqr(double x){return x*x;}
struct P{
        double x,y;
        P(){}
        P(double x,double y):x(x),y(y){}
        double len2(){
                return sqr(x)+sqr(y);
        }
        double len(){
                return sqrt(len2());
        }
        void print(){
                printf("(%.3f,%.3f)\n",x,y);
        }
        P turn90(){return P(-y,x);}
        P norm(){return P(x/len(),y/len());}
};
bool operator==(P a,P b){
        return !sgn(a.x-b.x) and !sgn(a.y-b.y);
}
P operator+(P a,P b){
        return P(a.x+b.x,a.y+b.y);
}
P operator-(P a,P b){
```

```
        return P(a.x-b.x,a.y-b.y);
}
P operator*(P a,double b){
        return P(a.x*b,a.y*b);
}
P operator/(P a,double b){
        return P(a.x/b,a.y/b);
}
double operator^(P a,P b){
        return a.x*b.x + a.y*b.y;
}
double operator*(P a,P b){
        return a.x*b.y - a.y*b.x;
}
double det(P a,P b,P c){
        return (b-a)*(c-a);
}
double dis(P a,P b){
        return (b-a).len();
}
double Area(vector<P>poly){
        double ans=0;
        for(int i=1;i<poly.size();i++)
                ans+=(poly[i]-poly[0])*(poly[(i+1)%poly.size()]-poly[0]);
        return fabs(ans)/2;
}
struct L{
        P a,b;
        L(){}
        L(P a,P b):a(a),b(b){}
        P v(){return b-a;}
};
bool onLine(P p,L l){
        return sgn((l.a-p)*(l.b-p))==0;
}
bool onSeg(P p,L s){
        return onLine(p,s) and sgn((s.b-s.a)^(p-s.a))>=0 and sgn((s.a-s.b)^(p-s.b))>=0;
}
bool parallel(L l1,L l2){
        return sgn(l1.v()*l2.v())==0;
}
P intersect(L l1,L l2){
```

```
        double s1=det(l1.a,l1.b,l2.a);
        double s2=det(l1.a,l1.b,l2.b);
        return (l2.a*s2-l2.b*s1)/(s2-s1);
}
P project(P p,L l){
        return l.a+l.v()*((p-l.a)^l.v())/l.v().len2();
}
double dis(P p,L l){
        return fabs((p-l.a)*l.v())/l.v().len();
}
```

6.1.2 凸包

```
vector<P> convex(vector<P>p){
        sort(p.begin(),p.end());
        vector<P>ans,S;
    for(int i=0;i<p.size();i++){
            while(S.size()>=2
                            && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
                                    S.pop_back();
            S.push_back(p[i]);
    }//dw
    ans=S;
    S.clear();
        for(int i=(int)p.size()-1;i>=0;i--){
            while(S.size()>=2
                            && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
                                    S.pop_back();
            S.push_back(p[i]);
        }//up
        for(int i=1;i+1<S.size();i++)
                ans.push_back(S[i]);
        return ans;
}
```

6.1.3 半平面交

```
struct P{
        int quad() const { return sgn(y) == 1 || (sgn(y) == 0 && sgn(x) >= 0);}
};
struct L{
```

```
        bool onLeft(const P &p) const { return sgn((b - a)*( p - a)) > 0; }
        L push() const{ // push out eps
                const double eps = 1e-10;
                P delta = (b - a).turn90().norm() * eps;
                return L(a - delta, b - delta);
        }
};
bool sameDir(const L &l0, const L &l1) {
        return parallel(l0, l1) && sgn((l0.b - l0.a)^(l1.b - l1.a)) == 1;
}
bool operator < (const P &a, const P &b) {
        if (a.quad() != b.quad())
                return a.quad() < b.quad();
        else
                return sgn((a*b)) > 0;
}
bool operator < (const L &l0, const L &l1) {
        if (sameDir(l0, l1))
                return l1.onLeft(l0.a);
        else
                return (l0.b - l0.a) < (l1.b - l1.a);
}
bool check(const L &u, const L &v, const L &w) {
        return w.onLeft(intersect(u, v));
}
vector<P> intersection(vector<L> &l) {
        sort(l.begin(), l.end());
        deque<L> q;
        for (int i = 0; i < (int)l.size(); ++i) {
                if (i && sameDir(l[i], l[i - 1])) {
                        continue;
                }
                while (q.size() > 1
                        && !check(q[q.size() - 2], q[q.size() - 1], l[i]))
                                q.pop_back();
                while (q.size() > 1
                        && !check(q[1], q[0], l[i]))
                                q.pop_front();
                q.push_back(l[i]);
        }
        while (q.size() > 2
                && !check(q[q.size() - 2], q[q.size() - 1], q[0]))
```

```
                        q.pop_back();
        while (q.size() > 2
                && !check(q[1], q[0], q[q.size() - 1]))
                        q.pop_front();
        vector<P> ret;
        for (int i = 0; i < (int)q.size(); ++i)
        ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
        return ret;
}
```

6.1.4 最近点对

```
bool byY(P a,P b){return a.y<b.y;}
LL solve(P *p,int l,int r){
        LL d=1LL<<62;
        if(l==r)
                return d;
        if(l+1==r)
                return dis2(p[l],p[r]);
        int mid=(l+r)>>1;
        d=min(solve(l,mid),d);
        d=min(solve(mid+1,r),d);
        vector<P>tmp;
        for(int i=l;i<=r;i++)
                if(sqr(p[mid].x-p[i].x)<=d)
                        tmp.push_back(p[i]);
        sort(tmp.begin(),tmp.end(),byY);
        for(int i=0;i<tmp.size();i++)
                for(int j=i+1;j<tmp.size()&&j-i<10;j++)
                        d=min(d,dis2(tmp[i],tmp[j]));
        return d;
}
```

6.1.5 最小圆覆盖

```
struct line{
        point p,v;
};
point Rev(point v){return point(-v.y,v.x);}
point operator*(line A,line B){
        point u=B.p-A.p;
```

```
        double t=(B.v*u)/(B.v*A.v);
        return A.p+A.v*t;
}
point get(point a,point b){
        return (a+b)/2;
}
point get(point a,point b,point c){
        if(a==b)return get(a,c);
        if(a==c)return get(a,b);
        if(b==c)return get(a,b);
        line ABO=(line){(a+b)/2,Rev(a-b)};
        line BCO=(line){(c+b)/2,Rev(b-c)};
        return ABO*BCO;
}
int main(){
        scanf("%d",&n);
        for(int i=1;i<=n;i++)scanf("%lf%lf",&p[i].x,&p[i].y);
        random_shuffle(p+1,p+1+n);
        O=p[1];r=0;
        for(int i=2;i<=n;i++){
                if(dis(p[i],O)<r+1e-6)continue;
                O=get(p[1],p[i]);r=dis(O,p[i]);
                for(int j=1;j<i;j++){
                        if(dis(p[j],O)<r+1e-6)continue;
                        O=get(p[i],p[j]);r=dis(O,p[i]);
                        for(int k=1;k<j;k++){
                                if(dis(p[k],O)<r+1e-6)continue;
                                O=get(p[i],p[j],p[k]);r=dis(O,p[i]);
                        }
                }
        }printf("%.2lf %.2lf %.2lf\n",O.x,O.y,r);
        return 0;
}s
```

6.2 多边形

6.2.1 判断点在多边形内部

```
bool InPoly(P p,vector<P>poly){
        int cnt=0;
        for(int i=0;i<poly.size();i++){
                P a=poly[i],b=poly[(i+1)%poly.size()];
```

```
                if(OnLine(p,L(a,b)))
                        return false;
                int x=sgn(det(a,p,b));
                int y=sgn(a.y-p.y);
                int z=sgn(b.y-p.y);
                cnt+=(x>0&&y<=0&&z>0);
                cnt-=(x<0&&z<=0&&y>0);
        }
        return cnt;
}
```

7 其他

7.1 斯坦纳树

```
priority_queue<pair<int, int> > Q;

// m is key point
// n is all point

for (int s = 0; s < (1 << m); s++){
        for (int i = 1; i <= n; i++){
                if (id[i]) continue;
                for (int s0 = 0; s0 < s; s0++)
                        if ( (s0 & s) == s0 ){
                                f[s][i] = min(f[s][i], f[s0][i] + f[s - s0][i]);
                        }
        }
        for (int i = 1; i <= n; i++) vis[i] = 0;
    while (!Q.empty()) Q.pop();
        for (int i = 1; i <= n; i++){
                if (id[i]) continue;
                Q.push(mp(-f[s][i], i));
        }
        while (!Q.empty()){
                while (!Q.empty() && Q.top().first != -f[s][Q.top().second]) Q.pop();
                        if (Q.empty()) break;
                        int Cur = Q.top().second; Q.pop();
                        for (int p = g[Cur]; p; p = nxt[p]){
                                int y = adj[p];
                                if ( f[s][y] > f[s][Cur] + 1){
                                        f[s][y] = f[s][Cur] + 1;
```

```
                                        Q.push(mp(-f[s][y], y));
                                }
                        }
        }
}
```

7.2 无敌的读入优化

```
namespace Reader {
        const int L = (1 << 20) + 5;
        char buffer[L], *S, *T;
        __inline bool getchar(char &ch) {
                if (S == T) {
                        T = (S = buffer) + fread(buffer, 1, L, stdin);
                        if (S == T) {
                                ch = EOF;
                                return false;
                        }
                }
                ch = *S ++;
                return true;
        }
        __inline bool getint(int &x) {
                char ch;
                for (; getchar(ch) && (ch < '0' || ch > '9'); );
                if (ch == EOF) return false;
                x = ch - '0';
                for (; getchar(ch), ch >= '0' && ch <= '9'; )
                        x = x * 10 + ch - '0';
                return true;
        }
}
Reader::getint(x);
Reader::getint(y);
```

7.3 最小树形图

```
const int maxn=1100;

int n,m , g[maxn][maxn] , used[maxn] , pass[maxn] , eg[maxn] , more , queue[maxn];

```

```
void combine (int id , int &sum ) {
        int tot = 0 , from , i , j , k ;
        for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
                queue[tot++]=id ; pass[id]=1;
        }
        for ( from=0; from<tot && queue[from]!=id ; from++);
        if  ( from==tot ) return ;
        more = 1 ;
        for ( i=from ; i<tot ; i++) {
                sum+=g[eg[queue[i]]][queue[i]] ;
                if ( i!=from ) {
                        used[queue[i]]=1;
                        for ( j = 1 ; j <= n ; j++) if ( !used[j] )
                                if ( g[queue[i]][j]<g[id][j] ) g[id][j]=g[queue[i]][j] ;
                }
        }
        for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
                for ( j=from ; j<tot ; j++){
                        k=queue[j];
                        if ( g[i][id]>g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg[k]][k];
                }
        }
}

int mdst( int root ) { // return the total length of MDST
        int i , j , k , sum = 0 ;
        memset ( used , 0 , sizeof ( used ) ) ;
        for ( more =1; more ; ) {
                more = 0 ;
                memset (eg,0,sizeof(eg)) ;
                for ( i=1 ; i <= n ; i ++) if ( !used[i] && i!=root ) {
                        for ( j=1 , k=0 ; j <= n ; j ++) if ( !used[j] && i!=j )
                                if ( k==0 || g[j][i] < g[k][i] ) k=j ;
                        eg[i] = k ;
                }
                memset(pass,0,sizeof(pass));
                for ( i=1; i<=n ; i++) if ( !used[i] && !pass[i] && i!= root ) combine ( i
        }
        for ( i =1; i<=n ; i ++) if ( !used[i] && i!= root ) sum+=g[eg[i]][i];
        return sum ;
}
```

7.4 DLX

```
int n,m,K;
struct DLX{
        int L[maxn],R[maxn],U[maxn],D[maxn];
        int sz,col[maxn],row[maxn],s[maxn],H[maxn];
        bool vis[233];
        int ans[maxn],cnt;
        void init(int m){
                for(int i=0;i<=m;i++){
                        L[i]=i-1;R[i]=i+1;
                        U[i]=D[i]=i;s[i]=0;
                }
                memset(H,-1,sizeof H);
                L[0]=m;R[m]=0;sz=m+1;
        }
        void Link(int r,int c){
                U[sz]=c;D[sz]=D[c];U[D[c]]=sz;D[c]=sz;
                if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
                else{
                        L[sz]=H[r];R[sz]=R[H[r]];
                        L[R[H[r]]]=sz;R[H[r]]=sz;
                }
                s[c]++;col[sz]=c;row[sz]=r;sz++;
        }
        void remove(int c){
                for(int i=D[c];i!=c;i=D[i])
                        L[R[i]]=L[i],R[L[i]]=R[i];
        }
        void resume(int c){
                for(int i=U[c];i!=c;i=U[i])
                        L[R[i]]=R[L[i]]=i;
        }
        int A(){
                int res=0;
                memset(vis,0,sizeof vis);
                for(int i=R[0];i;i=R[i])if(!vis[i]){
                        vis[i]=1;res++;
                        for(int j=D[i];j!=i;j=D[j])
                                for(int k=R[j];k!=j;k=R[k])
                                        vis[col[k]]=1;
                }
```

```
                return res;
        }
        void dfs(int d,int &ans){
                if(R[0]==0){ans=min(ans,d);return;}
                if(d+A()>=ans)return;
                int tmp=23333,c;
                for(int i=R[0];i;i=R[i])
                        if(tmp>s[i])tmp=s[i],c=i;
                for(int i=D[c];i!=c;i=D[i]){
                        remove(i);
                        for(int j=R[i];j!=i;j=R[j])remove(j);
                        dfs(d+1,ans);
                        for(int j=L[i];j!=i;j=L[j])resume(j);
                        resume(i);
                }
        }
        void del(int c){//exactly cover
        L[R[c]]=L[c];R[L[c]]=R[c];
                for(int i=D[c];i!=c;i=D[i])
                        for(int j=R[i];j!=i;j=R[j])
                                U[D[j]]=U[j],D[U[j]]=D[j],--s[col[j]];
    }
    void add(int c){  //exactly cover
        R[L[c]]=L[R[c]]=c;
                for(int i=U[c];i!=c;i=U[i])
                        for(int j=L[i];j!=i;j=L[j])
                                ++s[col[U[D[j]]=D[U[j]]=j]];
    }
        bool dfs2(int k){//exactly cover
        if(!R[0]){
            cnt=k;return 1;
        }
        int c=R[0];
                for(int i=R[0];i;i=R[i])
                        if(s[c]>s[i])c=i;
        del(c);
                for(int i=D[c];i!=c;i=D[i]){
                        for(int j=R[i];j!=i;j=R[j])
                                del(col[j]);
            ans[k]=row[i];if(dfs2(k+1))return true;
                        for(int j=L[i];j!=i;j=L[j])
                                add(col[j]);
```

```
        }
        add(c);
                return 0;
        }
}dlx;
int main(){
        dlx.init(n);
        for(int i=1;i<=m;i++)
                for(int j=1;j<=n;j++)
                        if(dis(station[i],city[j])<mid-eps)
                                dlx.Link(i,j);
                        dlx.dfs(0,ans);
}
```

7.5 插头 DP

```
int n,m,l;
struct L{
    int d[11];
    int& operator[](int x){return d[x];}
    int mc(int x){
        int an=1;
        if(d[x]==1){
            for(x++;x<l;x++)if(d[x]){
                an=an+(d[x]==1?1:-1);
                if(!an)return x;
            }
        }else{
            for(x--;x>=0;x--)if(d[x]){
                an=an+(d[x]==2?1:-1);
                if(!an)return x;
            }
        }
    }
    int h(){int an=0;for(int i=l-1;i>=0;i--)an=an*3+d[i];return an;}
    L s(int x,int y){
        L S=*this;
        S[x]=y;return S;
    }
    L operator>>(int _){
        L S=*this;
        for(int i=l-1;i>=1;i--)S[i]=S[i-1];
```

```
        S[0]=0;return S;
    }
};
struct Int{
    int len;
    int a[40];
    Int(){len=1;memset(a,0,sizeof a);}
    Int operator+=(const Int &o){
        int l=max(len,o.len);
        for(int i=0;i<l;i++)
            a[i]=a[i]+o.a[i];
        for(int i=0;i<l;i++)
            a[i+1]+=a[i]/10,a[i]%=10;
        if(a[l])l++;len=l;
        return *this;
    }
    void print(){
        for(int i=len-1;i>=0;i--)
            printf("%d",a[i]);
        puts("");
    }
};
struct hashtab{
    int sz;
    int tab[177147];
    Int w[177147];
    L s[177147];
    hashtab(){memset(tab,-1,sizeof tab);}
    void cl(){
        for(int i=0;i<sz;i++)tab[s[i].h()]=-1;
        sz=0;
    }
    Int& operator[](L S){
        int h=S.h();
        if(tab[h]==-1)tab[h]=sz,s[sz]=S,w[sz]=Int(),sz++;
        return w[tab[h]];
    }
}f[2];
bool check(L S){
    int cn1=0,cn2=0;
    for(int i=0;i<l;i++){
        cn1+=S[i]==1;
```

```
        cn2+=S[i]==2;
    }return cn1==1&&cn2==1;
}
int main(){
    Int One;One.a[0]=1;
    scanf("%d%d",&n,&m);if(n<m)swap(n,m);l=m+1;
    if(n==1||m==1){puts("1");return 0;}
    int cur=0;f[cur].cl();
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            if(i==1&&j==1){
                f[cur][L().s(0,1).s(1,2)]+=One;
                continue;
            }
            cur^=1;f[cur].cl();
            for(int k=0;k<f[!cur].sz;k++){
                L S=f[!cur].s[k];Int w=f[!cur][S];
                int d1=S[j-1],d2=S[j];
                if(d1==0&&d2==0){
                    if(i!=n&&j!=m)f[cur][S.s(j-1,1).s(j,2)]+=w;
                }else
                if(d1==0||d2==0){
                    if(i!=n)f[cur][S.s(j-1,d1|d2).s(j,0)]+=w;
                    if(j!=m)f[cur][S.s(j-1,0).s(j,d1|d2)]+=w;
                }else
                if(d1==1&&d2==2){
                    if(i==n&&j==m&&check(S))
                        (w+=w).print();
                }else
                if(d1==2&&d2==1){
                    f[cur][S.s(j-1,0).s(j,0)]+=w;
                }else
                if((d1==1&&d2==1)||(d1==2&&d2==2)){
                    int m1=S.mc(j),m2=S.mc(j-1);
                    f[cur][S.s(j-1,0).s(j,0).s(m1,1).s(m2,2)]+=w;
                }
            }
        }
        cur^=1;f[cur].cl();
        for(int k=0;k<f[!cur].sz;k++){
            L S=f[!cur].s[k];Int w=f[!cur][S];
            f[cur][S>>1]=w;
```

```
        }
    }
    return 0;
}
```

7.6 某年某月某日是星期几

```
int solve(int year, int month, int day) {
    int answer;
    if (month == 1 || month == 2) {
        month += 12;
        year--;
    }
    if ((year < 1752) || (year == 1752 && month < 9) ||
        (year == 1752 && month == 9 && day < 3)) {
        answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4 + 5) % 7;
    } else {
        answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4
                - year / 100 + year / 400) % 7;
    }
    return answer;
}
```

7.7 枚举大小为 k 的子集

使用条件：$k > 0$

```
void solve(int n, int k) {
    for (int comb = (1 << k) - 1; comb < (1 << n); ) {
        // ...
        int x = comb & -comb, y = comb + x;
        comb = (((comb & ~y) / x) >> 1) | y;
    }
}
```

7.8 环状最长公共子串

```
int n, a[N << 1], b[N << 1];

bool has(int i, int j) {
    return a[(i - 1) % n] == b[(j - 1) % n];
```

```
}

const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1, 0}};

int from[N][N];

int solve() {
    memset(from, 0, sizeof(from));
    int ret = 0;
    for (int i = 1; i <= 2 * n; ++i) {
        from[i][0] = 2;
        int left = 0, up = 0;
        for (int j = 1; j <= n; ++j) {
            int upleft = up + 1 + !!from[i - 1][j];
            if (!has(i, j)) {
                upleft = INT_MIN;
            }
            int max = std::max(left, std::max(upleft, up));
            if (left == max) {
                from[i][j] = 0;
            } else if (upleft == max) {
                from[i][j] = 1;
            } else {
                from[i][j] = 2;
            }
            left = max;
        }
        if (i >= n) {
            int count = 0;
            for (int x = i, y = n; y; ) {
                int t = from[x][y];
                count += t == 1;
                x += DELTA[t][0];
                y += DELTA[t][1];
            }
            ret = std::max(ret, count);
            int x = i - n + 1;
            from[x][0] = 0;
            int y = 0;
            while (y <= n && from[x][y] == 0) {
                y++;
            }
```

```
            for (; x <= i; ++x) {
                from[x][y] = 0;
                if (x == i) {
                    break;
                }
                for (; y <= n; ++y) {
                    if (from[x + 1][y] == 2) {
                        break;
                    }
                    if (y + 1 <= n && from[x + 1][y + 1] == 1) {
                        y++;
                        break;
                    }
                }
            }
        }
    }
    return ret;
}
```

7.9 LLMOD

```
LL multiplyMod(LL a, LL b, LL P) { // `需要保证 a 和 b 非负`
        LL t = (a * b - LL((long double)a / P * b + 1e-3) * P) % P;
        return t < 0 : t + P : t;
}
```

8 Java

8.1 基础模板

```
import java.io.*;
import java.util.*;
import java.math.*;
public class Main {
    public static void main(String[] args) {
        InputStream inputStream = System.in;
        OutputStream outputStream = System.out;
        InputReader in = new InputReader(inputStream);
        PrintWriter out = new PrintWriter(outputStream);
    }
```

```
}
public static class edge implements Comparable<edge>{
        public int u,v,w;
        public int compareTo(edge e){
                return w-e.w;
        }
}
public static class cmp implements Comparator<edge>{
        public int compare(edge a,edge b){
                if(a.w<b.w)return 1;
                if(a.w>b.w)return -1;
                return 0;
        }
}
class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;

    public InputReader(InputStream stream) {
        reader = new BufferedReader(new InputStreamReader(stream), 32768);
        tokenizer = null;
    }

    public String next() {
        while (tokenizer == null || !tokenizer.hasMoreTokens()) {
            try {
                tokenizer = new StringTokenizer(reader.readLine());
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
        return tokenizer.nextToken();
    }

    public int nextInt() {
        return Integer.parseInt(next());
    }

    public long nextLong() {
        return Long.parseLong(next());
    }
}
```

compact1, compact2, compact3

java.math

Class BigInteger

java.lang.Object
    java.lang.Number
        java.math.BigInteger

All Implemented Interfaces:

Serializable, Comparable<BigInteger>

```
public class BigInteger
extends Number
implements Comparable<BigInteger>
```

Immutable arbitrary-precision integers. All operations behave as if BigIntegers were represented in two's-complement notation (like Java's primitive integer types). BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. Additionally, BigInteger provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations.

Semantics of arithmetic operations exactly mimic those of Java's integer arithmetic operators, as defined in *The Java Language Specification*. For example, division by zero throws an `ArithmeticException`, and division of a negative by a positive yields a negative (or zero) remainder. All of the details in the Spec concerning overflow are ignored, as BigIntegers are made as large as necessary to accommodate the results of an operation.

Semantics of shift operations extend those of Java's shift operators to allow for negative shift distances. A right-shift with a negative shift distance results in a left shift, and vice-versa. The unsigned right shift operator (>>>) is omitted, as this operation makes little sense in combination with the "infinite word size" abstraction provided by this class.

Semantics of bitwise logical operations exactly mimic those of Java's bitwise integer operators. The binary operators (`and`, `or`, `xor`) implicitly perform sign extension on the shorter of the two operands prior to performing the operation.

Comparison operations perform signed integer comparisons, analogous to those performed by Java's relational and equality operators.

Modular arithmetic operations are provided to compute residues, perform exponentiation, and compute multiplicative inverses. These methods always return a non-negative result, between `0` and `(modulus - 1)`, inclusive.

Bit operations operate on a single bit of the two's-complement representation of their operand. If necessary, the operand is sign- extended so that it contains the designated bit. None of the single-bit operations can produce a BigInteger with a different sign from the BigInteger being operated on, as they affect only a single bit, and the "infinite word size" abstraction provided by this class ensures that there are infinitely many "virtual sign bits"

preceding each BigInteger.

For the sake of brevity and clarity, pseudo-code is used throughout the descriptions of BigInteger methods. The pseudo-code expression `(i + j)` is shorthand for "a BigInteger whose value is that of the BigInteger `i` plus that of the BigInteger `j`." The pseudo-code expression `(i == j)` is shorthand for "`true` if and only if the BigInteger `i` represents the same value as the BigInteger `j`." Other pseudo-code expressions are interpreted similarly.

All methods and constructors in this class throw `NullPointerException` when passed a null object reference for any input parameter. BigInteger must support values in the range $-2^{\texttt{Integer.MAX\_VALUE}}$ (exclusive) to $+2^{\texttt{Integer.MAX\_VALUE}}$ (exclusive) and may support values outside of that range. The range of probable prime values is limited and may be less than the full supported positive range of `BigInteger`. The range must be at least 1 to $2^{500000000}$.

Implementation Note:

`BigInteger constructors and operations throw ArithmeticException when the result is out of the supported range of` $-2^{\texttt{Integer.MAX\_VALUE}}$ `(exclusive) to` $+2^{\texttt{Integer.MAX\_VALUE}}$ `(exclusive).`

Since:

`JDK1.1`

See Also:

`BigDecimal`, `Serialized Form`

Field Summary

Fields

Modifier and Type	Field and Description
`static BigInteger`	**`ONE`** The BigInteger constant one.
`static BigInteger`	**`TEN`** The BigInteger constant ten.
`static BigInteger`	**`ZERO`** The BigInteger constant zero.

Constructor Summary

Constructors

Constructor and Description
`BigInteger``(byte[] val)` Translates a byte array containing the two's-complement binary representation of a BigInteger into a BigInteger.
`BigInteger``(int signum, byte[] magnitude)` Translates the sign-magnitude representation of a BigInteger into a BigInteger.

BigInteger(int bitLength, int certainty, **Random** rnd)
Constructs a randomly generated positive BigInteger that is probably prime, with the specified bitLength.

BigInteger(int numBits, **Random** rnd)
Constructs a randomly generated BigInteger, uniformly distributed over the range 0 to ($2^{numBits}$ - 1), inclusive.

BigInteger(**String** val)
Translates the decimal String representation of a BigInteger into a BigInteger.

BigInteger(**String** val, int radix)
Translates the String representation of a BigInteger in the specified radix into a BigInteger.

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
BigInteger	**abs**() Returns a BigInteger whose value is the absolute value of this BigInteger.
BigInteger	**add**(**BigInteger** val) Returns a BigInteger whose value is (this + val).
BigInteger	**and**(**BigInteger** val) Returns a BigInteger whose value is (this & val).
BigInteger	**andNot**(**BigInteger** val) Returns a BigInteger whose value is (this & ~val).
int	**bitCount**() Returns the number of bits in the two's complement representation of this BigInteger that differ from its sign bit.
int	**bitLength**() Returns the number of bits in the minimal two's-complement representation of this BigInteger, *excluding* a sign bit.
byte	**byteValueExact**() Converts this BigInteger to a byte, checking for lost information.
BigInteger	**clearBit**(int n) Returns a BigInteger whose value is equivalent to this BigInteger with the designated bit cleared.
int	**compareTo**(**BigInteger** val) Compares this BigInteger with the specified BigInteger.
BigInteger	**divide**(**BigInteger** val)

	Returns a BigInteger whose value is `(this / val)`.
`BigInteger[]`	**`divideAndRemainder(BigInteger val)`** Returns an array of two BigIntegers containing `(this / val)` followed by `(this % val)`.
`double`	**`doubleValue()`** Converts this BigInteger to a `double`.
`boolean`	**`equals(Object x)`** Compares this BigInteger with the specified Object for equality.
`BigInteger`	**`flipBit(int n)`** Returns a BigInteger whose value is equivalent to this BigInteger with the designated bit flipped.
`float`	**`floatValue()`** Converts this BigInteger to a `float`.
`BigInteger`	**`gcd(BigInteger val)`** Returns a BigInteger whose value is the greatest common divisor of `abs(this)` and `abs(val)`.
`int`	**`getLowestSetBit()`** Returns the index of the rightmost (lowest-order) one bit in this BigInteger (the number of zero bits to the right of the rightmost one bit).
`int`	**`hashCode()`** Returns the hash code for this BigInteger.
`int`	**`intValue()`** Converts this BigInteger to an `int`.
`int`	**`intValueExact()`** Converts this `BigInteger` to an `int`, checking for lost information.
`boolean`	**`isProbablePrime(int certainty)`** Returns `true` if this BigInteger is probably prime, `false` if it's definitely composite.
`long`	**`longValue()`** Converts this BigInteger to a `long`.
`long`	**`longValueExact()`** Converts this `BigInteger` to a `long`, checking for lost information.
`BigInteger`	**`max(BigInteger val)`** Returns the maximum of this BigInteger and `val`.
`BigInteger`	**`min(BigInteger val)`** Returns the minimum of this BigInteger and `val`.
`BigInteger`	**`mod(BigInteger m)`** Returns a BigInteger whose value is `(this mod m)`.

BigInteger	**modInverse**(**BigInteger** m) Returns a BigInteger whose value is $(this^{-1} \bmod m)$.
BigInteger	**modPow**(**BigInteger** exponent, **BigInteger** m) Returns a BigInteger whose value is $(this^{exponent} \bmod m)$.
BigInteger	**multiply**(**BigInteger** val) Returns a BigInteger whose value is (this * val).
BigInteger	**negate**() Returns a BigInteger whose value is (-this).
BigInteger	**nextProbablePrime**() Returns the first integer greater than this BigInteger that is probably prime.
BigInteger	**not**() Returns a BigInteger whose value is (~this).
BigInteger	**or**(**BigInteger** val) Returns a BigInteger whose value is (this | val).
BigInteger	**pow**(int exponent) Returns a BigInteger whose value is $(this^{exponent})$.
static **BigInteger**	**probablePrime**(int bitLength, **Random** rnd) Returns a positive BigInteger that is probably prime, with the specified bitLength.
BigInteger	**remainder**(**BigInteger** val) Returns a BigInteger whose value is (this % val).
BigInteger	**setBit**(int n) Returns a BigInteger whose value is equivalent to this BigInteger with the designated bit set.
BigInteger	**shiftLeft**(int n) Returns a BigInteger whose value is (this << n).
BigInteger	**shiftRight**(int n) Returns a BigInteger whose value is (this >> n).
short	**shortValueExact**() Converts this BigInteger to a short, checking for lost information.
int	**signum**() Returns the signum function of this BigInteger.
BigInteger	**subtract**(**BigInteger** val) Returns a BigInteger whose value is (this - val).
boolean	**testBit**(int n) Returns true if and only if the designated bit is set.
byte[]	**toByteArray**() Returns a byte array containing the two's-complement

	Returns a byte array containing the two's-complement representation of this BigInteger.
String	**toString**() Returns the decimal String representation of this BigInteger.
String	**toString**(int radix) Returns the String representation of this BigInteger in the given radix.
static **BigInteger**	**valueOf**(long val) Returns a BigInteger whose value is equal to that of the specified long.
BigInteger	**xor**(**BigInteger** val) Returns a BigInteger whose value is (this ^ val).

Methods inherited from class java.lang.Number

byteValue, shortValue

Methods inherited from class java.lang.Object

clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

ZERO

```
public static final BigInteger ZERO
```

The BigInteger constant zero.

Since:
1.2

ONE

```
public static final BigInteger ONE
```

The BigInteger constant one.

Since:
1.2

TEN

```
public static final BigInteger TEN
```

The BigInteger constant ten.

compact1, compact2, compact3

java.util

Class TreeMap<K,V>

java.lang.Object
    java.util.AbstractMap<K,V>
        java.util.TreeMap<K,V>

Type Parameters:

`K` - the type of keys maintained by this map

`V` - the type of mapped values

All Implemented Interfaces:

`Serializable`, `Cloneable`, `Map<K,V>`, `NavigableMap<K,V>`, `SortedMap<K,V>`

```
public class TreeMap<K,V>
extends AbstractMap<K,V>
implements NavigableMap<K,V>, Cloneable, Serializable
```

A Red-Black tree based `NavigableMap` implementation. The map is sorted according to the natural ordering of its keys, or by a `Comparator` provided at map creation time, depending on which constructor is used.

This implementation provides guaranteed log(n) time cost for the `containsKey`, `get`, `put` and `remove` operations. Algorithms are adaptations of those in Cormen, Leiserson, and Rivest's *Introduction to Algorithms*.

Note that the ordering maintained by a tree map, like any sorted map, and whether or not an explicit comparator is provided, must be *consistent with `equals`* if this sorted map is to correctly implement the `Map` interface. (See `Comparable` or `Comparator` for a precise definition of *consistent with equals*.) This is so because the `Map` interface is defined in terms of the `equals` operation, but a sorted map performs all key comparisons using its `compareTo` (or `compare`) method, so two keys that are deemed equal by this method are, from the standpoint of the sorted map, equal. The behavior of a sorted map *is* well-defined even if its ordering is inconsistent with `equals`; it just fails to obey the general contract of the `Map` interface.

Note that this implementation is not synchronized. If multiple threads access a map concurrently, and at least one of the threads modifies the map structurally, it *must* be synchronized externally. (A structural modification is any operation that adds or deletes one or more mappings; merely changing the value associated with an existing key is not a structural modification.) This is typically accomplished by synchronizing on some object that naturally encapsulates the map. If no such object exists, the map should be "wrapped" using the `Collections.synchronizedSortedMap` method. This is best done at creation time, to prevent accidental unsynchronized access to the map:

```
SortedMap m = Collections.synchronizedSortedMap(new TreeMap(...));
```

The iterators returned by the `iterator` method of the collections returned by all of this

class's "collection view methods" are *fail-fast*: if the map is structurally modified at any time after the iterator is created, in any way except through the iterator's own `remove` method, the iterator will throw a `ConcurrentModificationException`. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

Note that the fail-fast behavior of an iterator cannot be guaranteed as it is, generally speaking, impossible to make any hard guarantees in the presence of unsynchronized concurrent modification. Fail-fast iterators throw `ConcurrentModificationException` on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: *the fail-fast behavior of iterators should be used only to detect bugs.*

All `Map.Entry` pairs returned by methods in this class and its views represent snapshots of mappings at the time they were produced. They do **not** support the `Entry.setValue` method. (Note however that it is possible to change mappings in the associated map using `put`.)

This class is a member of the Java Collections Framework.

Since:

1.2

See Also:

`Map`, `HashMap`, `Hashtable`, `Comparable`, `Comparator`, `Collection`, `Serialized Form`

Nested Class Summary

Nested classes/interfaces inherited from class java.util.AbstractMap

`AbstractMap.SimpleEntry<K,V>`, `AbstractMap.SimpleImmutableEntry<K,V>`

Constructor Summary

Constructors

Constructor and Description

`TreeMap()`
Constructs a new, empty tree map, using the natural ordering of its keys.

`TreeMap(Comparator<? super K> comparator)`
Constructs a new, empty tree map, ordered according to the given comparator.

`TreeMap(Map<? extends K,? extends V> m)`
Constructs a new tree map containing the same mappings as the given map, ordered according to the *natural ordering* of its keys.

`TreeMap(SortedMap<K,? extends V> m)`
Constructs a new tree map containing the same mappings and using the same ordering as the specified sorted map.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Map.Entry<K,V>`	**`ceilingEntry(K key)`** Returns a key-value mapping associated with the least key greater than or equal to the given key, or `null` if there is no such key.
`K`	**`ceilingKey(K key)`** Returns the least key greater than or equal to the given key, or `null` if there is no such key.
`void`	**`clear()`** Removes all of the mappings from this map.
`Object`	**`clone()`** Returns a shallow copy of this `TreeMap` instance.
`Comparator<? super K>`	**`comparator()`** Returns the comparator used to order the keys in this map, or `null` if this map uses the **natural ordering** of its keys.
`boolean`	**`containsKey(Object key)`** Returns `true` if this map contains a mapping for the specified key.
`boolean`	**`containsValue(Object value)`** Returns `true` if this map maps one or more keys to the specified value.
`NavigableSet<K>`	**`descendingKeySet()`** Returns a reverse order **`NavigableSet`** view of the keys contained in this map.
`NavigableMap<K,V>`	**`descendingMap()`** Returns a reverse order view of the mappings contained in this map.
`Set<Map.Entry<K,V>>`	**`entrySet()`** Returns a **`Set`** view of the mappings contained in this map.
`Map.Entry<K,V>`	**`firstEntry()`** Returns a key-value mapping associated with the least key in this map, or `null` if the map is empty.
`K`	**`firstKey()`** Returns the first (lowest) key currently in this map.
`Map.Entry<K,V>`	**`floorEntry(K key)`** Returns a key-value mapping associated with the greatest key less than or equal to the given key, or `null` if there is no such key.
`K`	**`floorKey(K key)`** Returns the greatest key less than or equal to the given key, or `null` if there is no such key.

	or null if there is no such key.
void	**forEach**(**BiConsumer**<? super **K**,? super **V**> action) Performs the given action for each entry in this map until all entries have been processed or the action throws an exception.
V	**get**(**Object** key) Returns the value to which the specified key is mapped, or `null` if this map contains no mapping for the key.
SortedMap<**K**,**V**>	**headMap**(**K** toKey) Returns a view of the portion of this map whose keys are strictly less than `toKey`.
NavigableMap<**K**,**V**>	**headMap**(**K** toKey, boolean inclusive) Returns a view of the portion of this map whose keys are less than (or equal to, if `inclusive` is true) `toKey`.
Map.Entry<**K**,**V**>	**higherEntry**(**K** key) Returns a key-value mapping associated with the least key strictly greater than the given key, or `null` if there is no such key.
K	**higherKey**(**K** key) Returns the least key strictly greater than the given key, or `null` if there is no such key.
Set<**K**>	**keySet**() Returns a **Set** view of the keys contained in this map.
Map.Entry<**K**,**V**>	**lastEntry**() Returns a key-value mapping associated with the greatest key in this map, or `null` if the map is empty.
K	**lastKey**() Returns the last (highest) key currently in this map.
Map.Entry<**K**,**V**>	**lowerEntry**(**K** key) Returns a key-value mapping associated with the greatest key strictly less than the given key, or `null` if there is no such key.
K	**lowerKey**(**K** key) Returns the greatest key strictly less than the given key, or `null` if there is no such key.
NavigableSet<**K**>	**navigableKeySet**() Returns a **NavigableSet** view of the keys contained in this map.
Map.Entry<**K**,**V**>	**pollFirstEntry**() Removes and returns a key-value mapping associated with the least key in this map, or `null` if the map is empty.
Map.Entry<**K**,**V**>	**pollLastEntry**() Removes and returns a key-value mapping associated with the greatest key in this map, or `null` if the map is empty

	the greatest key in this map, or null if the map is empty.
V	**put**(K key, V value) Associates the specified value with the specified key in this map.
void	**putAll**(Map<? extends K,? extends V> map) Copies all of the mappings from the specified map to this map.
V	**remove**(Object key) Removes the mapping for this key from this TreeMap if present.
V	**replace**(K key, V value) Replaces the entry for the specified key only if it is currently mapped to some value.
boolean	**replace**(K key, V oldValue, V newValue) Replaces the entry for the specified key only if currently mapped to the specified value.
void	**replaceAll**(BiFunction<? super K,? super V,? extends V> function) Replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception.
int	**size**() Returns the number of key-value mappings in this map.
NavigableMap<K,V>	**subMap**(K fromKey, boolean fromInclusive, K toKey, boolean toInclusive) Returns a view of the portion of this map whose keys range from fromKey to toKey.
SortedMap<K,V>	**subMap**(K fromKey, K toKey) Returns a view of the portion of this map whose keys range from fromKey, inclusive, to toKey, exclusive.
SortedMap<K,V>	**tailMap**(K fromKey) Returns a view of the portion of this map whose keys are greater than or equal to fromKey.
NavigableMap<K,V>	**tailMap**(K fromKey, boolean inclusive) Returns a view of the portion of this map whose keys are greater than (or equal to, if inclusive is true) fromKey.
Collection<V>	**values**() Returns a **Collection** view of the values contained in this map.

Methods inherited from class java.util.AbstractMap

equals, hashCode, isEmpty, toString

9 gedit

```
Compile:
#!/bin/sh
full=$GEDIT_CURRENT_DOCUMENT_NAME
name=`echo $full | cut -d. -f1`
g++ $full -o $name -g -Wall

Debug:
#!/bin/bash
name=`echo $GEDIT_CURRENT_DOCUMENT_NAME | cut -d. -f1`
gnome-terminal -x bash -c "gdb ./$name"

Run:
#!/bin/bash
name=`echo $GEDIT_CURRENT_DOCUMENT_NAME | cut -d. -f1`
gnome-terminal -x bash -c "time ./$name;echo 'Press any key to continue'; read"
```

10 数学

10.1 常用数学公式

10.1.1 求和公式

1. $\sum_{k=1}^{n}(2k-1)^2 = \frac{n(4n^2-1)}{3}$
2. $\sum_{k=1}^{n}k^3 = [\frac{n(n+1)}{2}]^2$
3. $\sum_{k=1}^{n}(2k-1)^3 = n^2(2n^2-1)$
4. $\sum_{k=1}^{n}k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$
5. $\sum_{k=1}^{n}k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$
6. $\sum_{k=1}^{n}k(k+1) = \frac{n(n+1)(n+2)}{3}$
7. $\sum_{k=1}^{n}k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$
8. $\sum_{k=1}^{n}k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

10.1.2 斐波那契数列

1. $fib_0 = 0, fib_1 = 1, fib_n = fib_{n-1} + fib_{n-2}$
2. $fib_{n+2} \cdot fib_n - fib_{n+1}^2 = (-1)^{n+1}$
3. $fib_{-n} = (-1)^{n-1} fib_n$

4. $fib_{n+k} = fib_k \cdot fib_{n+1} + fib_{k-1} \cdot fib_n$

5. $gcd(fib_m, fib_n) = fib_{gcd(m,n)}$

6. $fib_m | fib_n^2 \Leftrightarrow nfib_n | m$

10.1.3 错排公式

1. $D_n = (n-1)(D_{n-2} - D_{n-1})$

2. $D_n = n! \cdot (1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + \frac{(-1)^n}{n!})$

10.1.4 莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & 若n = 1 \\ (-1)^k & 若n无平方数因子，且n = p_1 p_2 \ldots p_k \\ 0 & 若n有大于1的平方数因数 \end{cases}$$

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & 若n = 1 \\ 0 & 其他情况 \end{cases}$$

$$g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$$

$$g(x) = \sum_{n=1}^{[x]} f(\frac{x}{n}) \Leftrightarrow f(x) = \sum_{n=1}^{[x]} \mu(n) g(\frac{x}{n})$$

10.1.5 伯恩赛德引理

设 G 是一个有限群，作用在集合 X 上。对每个 g 属于 G，令 X^g 表示 X 中在 g 作用下的不动元素，轨道数（记作 $|X/G|$）由如下公式给出：

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|.$$

10.1.6 五边形数定理

设 $p(n)$ 是 n 的拆分数，有

$$p(n) = \sum_{k \in \mathbb{Z} \setminus \{0\}} (-1)^{k-1} p\left(n - \frac{k(3k-1)}{2}\right)$$

10.1.7 树的计数

1. 有根树计数：$n+1$ 个结点的有根树的个数为

$$a_{n+1} = \frac{\sum_{j=1}^{n} j \cdot a_j \cdot S_{n,j}}{n}$$

其中，

$$S_{n,j}=\sum_{i=1}^{n/j}a_{n+1-ij}=S_{n-j,j}+a_{n+1-j}$$

2. 无根树计数：当 n 为奇数时，n 个结点的无根树的个数为

$$a_n-\sum_{i=1}^{n/2}a_ia_{n-i}$$

当 n 为偶数时，n 个结点的无根树的个数为

$$a_n-\sum_{i=1}^{n/2}a_ia_{n-i}+\frac{1}{2}a_{\frac{n}{2}}(a_{\frac{n}{2}}+1)$$

3. n 个结点的完全图的生成树个数为

$$n^{n-2}$$

4. 矩阵一树定理：图 G 由 n 个结点构成，设 $\boldsymbol{A}[G]$ 为图 G 的邻接矩阵、$\boldsymbol{D}[G]$ 为图 G 的度数矩阵，则图 G 的不同生成树的个数为 $\boldsymbol{C}[G]=\boldsymbol{D}[G]-\boldsymbol{A}[G]$ 的任意一个 $n-1$ 阶主子式的行列式值。

10.1.8 欧拉公式

平面图的顶点个数、边数和面的个数有如下关系：

$$V-E+F=C+1$$

其中，V 是顶点的数目，E 是边的数目，F 是面的数目，C 是组成图形的连通部分的数目。当图是单连通图的时候，公式简化为：

$$V-E+F=2$$

10.1.9 皮克定理

给定顶点坐标均是整点（或正方形格点）的简单多边形，其面积 A 和内部格点数目 i、边上格点数目 b 的关系：

$$A=i+\frac{b}{2}-1$$

10.1.10 牛顿恒等式

设

$$\prod_{i=1}^{n}(x-x_i)=a_n+a_{n-1}x+\cdots+a_1x^{n-1}+a_0x^n$$

$$p_k=\sum_{i=1}^{n}x_i^k$$

则

$$a_0p_k + a_1p_{k-1} + \cdots + a_{k-1}p_1 + ka_k = 0$$

特别地，对于

$$|\boldsymbol{A} - \lambda\boldsymbol{E}| = (-1)^n(a_n + a_{n-1}\lambda + \cdots + a_1\lambda^{n-1} + a_0\lambda^n)$$

有

$$p_k = Tr(\boldsymbol{A}^k)$$

10.2 平面几何公式

10.2.1 三角形

1. 半周长

$$p = \frac{a+b+c}{2}$$

2. 面积

$$S = \frac{a \cdot H_a}{2} = \frac{ab \cdot sinC}{2} = \sqrt{p(p-a)(p-b)(p-c)}$$

3. 中线

$$M_a = \frac{\sqrt{2(b^2+c^2)-a^2}}{2} = \frac{\sqrt{b^2+c^2+2bc \cdot cosA}}{2}$$

4. 角平分线

$$T_a = \frac{\sqrt{bc \cdot [(b+c)^2 - a^2]}}{b+c} = \frac{2bc}{b+c}cos\frac{A}{2}$$

5. 高线

$$H_a = bsinC = csinB = \sqrt{b^2 - (\frac{a^2+b^2-c^2}{2a})^2}$$

6. 内切圆半径

$$r = \frac{S}{p} = \frac{arcsin\frac{B}{2} \cdot sin\frac{C}{2}}{sin\frac{B+C}{2}} = 4R \cdot sin\frac{A}{2}sin\frac{B}{2}sin\frac{C}{2}$$
$$= \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p \cdot tan\frac{A}{2}tan\frac{B}{2}tan\frac{C}{2}$$

7. 外接圆半径

$$R = \frac{abc}{4S} = \frac{a}{2sinA} = \frac{b}{2sinB} = \frac{c}{2sinC}$$

10.2.2 四边形

D_1, D_2 为对角线，M 对角线中点连线，A 为对角线夹角，p 为半周长

1. $a^2 + b^2 + c^2 + d^2 = D_1^2 + D_2^2 + 4M^2$
2. $S = \frac{1}{2}D_1D_2sinA$

3. 对于圆内接四边形

$$ac + bd = D_1 D_2$$

4. 对于圆内接四边形

$$S = \sqrt{(p-a)(p-b)(p-c)(p-d)}$$

10.2.3 正 n 边形

R 为外接圆半径，r 为内切圆半径

1. 中心角

$$A = \frac{2\pi}{n}$$

2. 内角

$$C = \frac{n-2}{n}\pi$$

3. 边长

$$a = 2\sqrt{R^2 - r^2} = 2R \cdot sin\frac{A}{2} = 2r \cdot tan\frac{A}{2}$$

4. 面积

$$S = \frac{nar}{2} = nr^2 \cdot tan\frac{A}{2} = \frac{nR^2}{2} \cdot sinA = \frac{na^2}{4 \cdot tan\frac{A}{2}}$$

10.2.4 圆

1. 弧长

$$l = rA$$

2. 弦长

$$a = 2\sqrt{2hr - h^2} = 2r \cdot sin\frac{A}{2}$$

3. 弓形高

$$h = r - \sqrt{r^2 - \frac{a^2}{4}} = r(1 - cos\frac{A}{2}) = \frac{1}{2} \cdot arctan\frac{A}{4}$$

4. 扇形面积

$$S_1 = \frac{rl}{2} = \frac{r^2 A}{2}$$

5. 弓形面积

$$S_2 = \frac{rl - a(r-h)}{2} = \frac{r^2}{2}(A - sinA)$$

10.2.5 棱柱

1. 体积

$$V = Ah$$

A 为底面积，h 为高

2. 侧面积

$$S = lp$$

l 为棱长，p 为直截面周长

3. 全面积

$$T = S + 2A$$

10.2.6 棱锥

1. 体积

$$V = Ah$$

A 为底面积，h 为高

2. 正棱锥侧面积

$$S = lp$$

l 为棱长，p 为直截面周长

3. 正棱锥全面积

$$T = S + 2A$$

10.2.7 棱台

1. 体积

$$V = (A_1 + A_2 + \sqrt{A_1 A_2}) \cdot \frac{h}{3}$$

A_1, A_2 为上下底面积，h 为高

2. 正棱台侧面积

$$S = \frac{p_1 + p_2}{2} l$$

p_1, p_2 为上下底面周长，l 为斜高

3. 正棱台全面积

$$T = S + A_1 + A_2$$

10.2.8 圆柱

1. 侧面积

$$S = 2\pi r h$$

2. 全面积

$$T = 2\pi r(h + r)$$

3. 体积

$$V = \pi r^2 h$$

10.2.9 圆锥

1. 母线

$$l = \sqrt{h^2 + r^2}$$

2. 侧面积

$$S = \pi r l$$

3. 全面积

$$T = \pi r(l + r)$$

4. 体积

$$V = \frac{\pi}{3} r^2 h$$

10.2.10 圆台

1. 母线

$$l = \sqrt{h^2 + (r_1 - r_2)^2}$$

2. 侧面积

$$S = \pi(r_1 + r_2)l$$

3. 全面积

$$T = \pi r_1(l + r_1) + \pi r_2(l + r_2)$$

4. 体积

$$V = \frac{\pi}{3}(r_1^2 + r_2^2 + r_1 r_2)h$$

10.2.11 球

1. 全面积

$$T = 4\pi r^2$$

2. 体积

$$V = \frac{4}{3}\pi r^3$$

10.2.12 球台

1. 侧面积

$$S = 2\pi r h$$

2. 全面积

$$T = \pi(2rh + r_1^2 + r_2^2)$$

3. 体积

$$V = \frac{\pi h[3(r_1^2 + r_2^2) + h^2]}{6}$$

10.2.13 球扇形

1. 全面积

$$T = \pi r(2h + r_0)$$

h 为球冠高，r_0 为球冠底面半径

2. 体积

$$V = \frac{2}{3}\pi r^2 h$$

10.3 立体几何公式

10.3.1 球面三角公式

设 a, b, c 是边长，A, B, C 是所对的二面角，有余弦定理

$$cosa = cosb \cdot cosc + sinb \cdot sinc \cdot cosA$$

正弦定理

$$\frac{sinA}{sina} = \frac{sinB}{sinb} = \frac{sinC}{sinc}$$

三角形面积是 $A + B + C - \pi$

10.3.2 四面体体积公式

U, V, W, u, v, w 是四面体的 6 条棱，U, V, W 构成三角形，$(U, u), (V, v), (W, w)$ 互为对棱，则

$$V = \frac{\sqrt{(s-2a)(s-2b)(s-2c)(s-2d)}}{192uvw}$$

其中

$$\begin{cases} a &= \sqrt{xYZ}, \\ b &= \sqrt{yZX}, \\ c &= \sqrt{zXY}, \\ d &= \sqrt{xyz}, \\ s &= a+b+c+d, \\ X &= (w-U+v)(U+v+w), \\ x &= (U-v+w)(v-w+U), \\ Y &= (u-V+w)(V+w+u), \\ y &= (V-w+u)(w-u+V), \\ Z &= (v-W+u)(W+u+v), \\ z &= (W-u+v)(u-v+W) \end{cases}$$