Grimoire's Standard Code Library*

Shanghai Jiao Tong University

Dated: 2018 年 9 月 25 日

*https://github.com/kzoacn/Grimoire

目录

Chapter 1

代数

1.1 $O(n^2 \log n)$ 求线性递推数列第 n 项

Given $a_0, a_1, \cdots, a_{m-1}$
$a_n = c_0 * a_{n-m} + \cdots + c_{m-1} * a_0$
a_0 *is the nth element*, $\cdots, a_{m-1}$ *is the* $n + m - 1$*th element*

```
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];
    }
```

```
}
```

1.2 任意模数快速傅里叶变换

```
// double 精度对 10^9 + 7 取模最多可以做到 2^20
const int MOD = 1000003;
const double PI = acos(-1);
typedef complex<double> Complex;
const int N = 65536, L = 15, MASK = (1 << L) - 1;
Complex w[N];
void FFTInit() {
    for (int i = 0; i < N; ++i)
        w[i] = Complex(cos(2 * i * PI / N), sin(2 * i * PI / N));
}
void FFT(Complex p[], int n) {
    for (int i = 1, j = 0; i < n - 1; ++i) {
        for (int s = n; j ^= s >>= 1, ~j & s;);
        if (i < j) swap(p[i], p[j]);
    }
    for (int d = 0; (1 << d) < n; ++d) {
        int m = 1 << d, m2 = m * 2, rm = n >> (d + 1);
        for (int i = 0; i < n; i += m2) {
            for (int j = 0; j < m; ++j) {
                Complex &p1 = p[i + j + m], &p2 = p[i + j];
                Complex t = w[rm * j] * p1;
                p1 = p2 - t, p2 = p2 + t;
            } } }
}
Complex A[N], B[N], C[N], D[N];
void mul(int a[N], int b[N]) {
    for (int i = 0; i < N; ++i) {
        A[i] = Complex(a[i] >> L, a[i] & MASK);
        B[i] = Complex(b[i] >> L, b[i] & MASK);
    }
    FFT(A, N), FFT(B, N);
    for (int i = 0; i < N; ++i) {
        int j = (N - i) % N;
        Complex da = (A[i] - conj(A[j])) * Complex(0, -0.5),
                db = (A[i] + conj(A[j])) * Complex(0.5, 0),
                dc = (B[i] - conj(B[j])) * Complex(0, -0.5),
                dd = (B[i] + conj(B[j])) * Complex(0.5, 0);
        C[j] = da * dd + da * dc * Complex(0, 1);
        D[j] = db * dd + db * dc * Complex(0, 1);
    }
    FFT(C, N), FFT(D, N);
    for (int i = 0; i < N; ++i) {
        long long da = (long long)(C[i].imag() / N + 0.5) % MOD,
                  db = (long long)(C[i].real() / N + 0.5) % MOD,
                  dc = (long long)(D[i].imag() / N + 0.5) % MOD,
                  dd = (long long)(D[i].real() / N + 0.5) % MOD;
        a[i] = ((dd << (L * 2)) + ((db + dc) << L) + da) % MOD;
    }
}
```

1.3 快速傅里叶变换

```
int prepare(int n) {
    int len = 1;
```

```
    for (; len <= 2 * n; len <<= 1);
    for (int i = 0; i < len; i++) {
        e[0][i] = Complex(cos(2 * pi * i / len), sin(2 * pi * i / len));
        e[1][i] = Complex(cos(2 * pi * i / len), -sin(2 * pi * i / len));
    }
    return len;
}
void DFT(Complex *a, int n, int f) {
    for (int i = 0, j = 0; i < n; i++) {
        if (i > j) std::swap(a[i], a[j]);
        for (int t = n >> 1; (j ^= t) < t; t >>= 1);
    }
    for (int i = 2; i <= n; i <<= 1)
        for (int j = 0; j < n; j += i)
            for (int k = 0; k < (i >> 1); k++) {
                Complex A = a[j + k];
                Complex B = e[f][n / i * k] * a[j + k + (i >> 1)];
                a[j + k] = A + B;
                a[j + k + (i >> 1)] = A - B;
            }
    if (f == 1) {
        for (int i = 0; i < n; i++)
            a[i].a /= n;
    }
}
```

1.4 闪电数论变换与魔力 CRT

```
#define meminit(A, l, r) memset(A + (l), 0, sizeof(*A) * ((r) - (l)))
#define memcopy(B, A, l, r) memcpy(B, A + (l), sizeof(*A) * ((r) - (l)))
void DFT(int *a, int n, int f) { //f=1 逆 DFT
    for (register int i = 0, j = 0; i < n; i++) {
        if (i > j) std::swap(a[i], a[j]);
        for (register int t = n >> 1; (j ^= t) < t; t >>= 1);
    }
    for (register int i = 2; i <= n; i <<= 1) {
        static int exp[MAXN];
        exp[0] = 1; exp[1] = fpm(PRT, (MOD - 1) / i, MOD);
        if (f == 1) exp[1] = fpm(exp[1], MOD - 2, MOD);
        for (register int k = 2; k < (i >> 1); k++) {
            exp[k] = 1ll * exp[k - 1] * exp[1] % MOD;
        }
        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 long long B = 1ll * pB * exp[k];
                pB = (pA - B) % MOD;
                pA = (pA + B) % MOD;
            }
        }
    }
    if (f == 1) {
        register int rev = fpm(n, MOD - 2, MOD);
        for (register int i = 0; i < n; i++) {
            a[i] = 1ll * a[i] * rev % MOD;
            if (a[i] < 0) { a[i] += MOD; }
        }
    }
}
```

```
// 在不写高精度的情况下合并 FFT 所得结果对 MOD 取模后的答案
// 值得注意的是，这个东西不能最后再合并，而是应该每做一次多项式乘法就 CRT 一次
int CRT(int *a) {
    static int x[3];
    for (int i = 0; i < 3; i++) {
        x[i] = a[i];
        for (int j = 0; j < i; j++) {
            int t = (x[i] - x[j] + FFT[i] -> MOD) % FFT[i] -> MOD;
            if (t < 0) t += FFT[i] -> MOD;
            x[i] = 1LL * t * inv[j][i] % FFT[i] -> MOD;
        }
    }
    int sum = 1, ret = x[0] % MOD;
    for (int i = 1; i < 3; i ++) {
        sum = 1LL * sum * FFT[i - 1] -> MOD % MOD;
        ret += 1LL * x[i] * sum % MOD;
        if(ret >= MOD) ret -= MOD;
    }
    return ret;
}
for (int i = 0; i < 3; i++) // inv 数组的预处理过程，inverse(x, p) 表示求 x 在 p 下逆元
    for (int j = 0; j < 3; j++)
        inv[i][j] = inverse(FFT[i] -> MOD, FFT[j] -> MOD);
```

1.5 多项式求逆

Given polynomial a and n, b is the polynomial such that $a * b \equiv 1 \pmod{x^n}$

```
void getInv(int *a, int *b, int n) {
    static int tmp[MAXN];
    b[0] = fpm(a[0], MOD - 2, MOD);
    for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
        for (; M <= 3 * (c - 1); M <<= 1);
        meminit(b, c, M);
        meminit(tmp, c, M);
        memcopy(tmp, a, 0, c);
        DFT(tmp, M, 0);
        DFT(b, M, 0);
        for (int i = 0; i < M; i++) {
            b[i] = 1ll * b[i] * (2ll - 1ll * tmp[i] * b[i] % MOD + MOD) % MOD;
        }
        DFT(b, M, 1);
        meminit(b, c, M);
    }
}
```

1.6 多项式除法

d is quotient and r is remainder

```
void divide(int n, int m, int *a, int *b, int *d, int *r) { // n、m 分别为多项式 A（被除数）
    ↪ 和 B（除数）的指数 + 1
    static int M, tA[MAXN], tB[MAXN], inv[MAXN], tD[MAXN];
    for (; n > 0 && a[n - 1] == 0; n--);
    for (; m > 0 && b[m - 1] == 0; m--);
    for (int i = 0; i < n; i++) tA[i] = a[n - i - 1];
    for (int i = 0; i < m; i++) tB[i] = b[m - i - 1];
    for (M = 1; M <= n - m + 1; M <<= 1);
    if (m < M) meminit(tB, m, M);
```

```
    getInv(tB, inv, M);
    for (M = 1; M <= 2 * (n - m + 1); M <<= 1);
    meminit(inv, n - m + 1, M);
    meminit(tA, n - m + 1, M);
    DFT(inv, M, 0);
    DFT(tA, M, 0);
    for (int i = 0; i < M; i++) {
        d[i] = 1ll * inv[i] * tA[i] % MOD;
    }
    DFT(d, M, 1);
    std::reverse(d, d + n - m + 1);
    for (M = 1; M <= n; M <<= 1);
    memcopy(tB, b, 0, m);
    if (m < M) meminit(tB, m, M);
    memcopy(tD, d, 0, n - m + 1);
    meminit(tD, n - m + 1, M);
    DFT(tD, M, 0);
    DFT(tB, M, 0);
    for (int i = 0; i < M; i++) {
        r[i] = 1ll * tD[i] * tB[i] % MOD;
    }
    DFT(r, M, 1);
    meminit(r, n, M);
    for (int i = 0; i < n; i++) {
        r[i] = (a[i] - r[i] + MOD) % MOD;
    }
}
```

1.7 多项式取指数取对数

Given polynomial a and n, b is the polynomial such that $b \equiv e^a \pmod{x^n}$ or $b \equiv \ln a \pmod{x^n}$

```
void getDiff(int *a, int *b, int n) { // 多项式取微分
    for (int i = 0; i + 1 < n; i++) {
        b[i] = 1ll * (i + 1) * a[i + 1] % MOD;
    }
    b[n - 1] = 0;
}
void getInt(int *a, int *b, int n) { // 多项式取积分，积分常数为 0
    static int inv[MAXN];
    inv[1] = 1;
    for (int i = 2; i < n; i++) {
        inv[i] = 1ll * (MOD - MOD / i) * inv[MOD % i] % MOD;
    }
    b[0] = 0;
    for (int i = 1; i < n; i++) {
        b[i] = 1ll * a[i - 1] * inv[i] % MOD;
    }
}
void getLn(int *a, int *b, int n) {
    static int inv[MAXN], d[MAXN];
    int M = 1;
    for (; M <= 2 * (n - 1); M <<= 1);
    getInv(a, inv, n);
    getDiff(a, d, n);
    meminit(d, n, M);
    meminit(inv, n, M);
    DFT(d, M, 0); DFT(inv, M, 0);
    for (int i = 0; i < M; i++) {
```

```
        d[i] = 1ll * d[i] * inv[i] % MOD;
    }
    DFT(d, M, 1);
    getInt(d, b, n);
}
void getExp(int *a, int *b, int n) {
    static int ln[MAXN], tmp[MAXN];
    b[0] = 1;
    for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
        for (; M <= 2 * (c - 1); M <<= 1);
        int bound = std::min(c, n);
        memcopy(tmp, a, 0, bound);
        meminit(tmp, bound, M);
        meminit(b, c, M);
        getLn(b, ln, c);
        meminit(ln, c, M);
        DFT(b, M, 0);
        DFT(tmp, M, 0);
        DFT(ln, M, 0);
        for (int i = 0; i < M; i++) {
            b[i] = 1ll * b[i] * (1ll - ln[i] + tmp[i] + MOD) % MOD;
        }
        DFT(b, M, 1);
        meminit(b, c, M);
    }
}
```

1.8 快速沃尔什变换

```
void FWT(LL a[],int n,int ty){
    for(int d=1;d<n;d<<=1){
        for(int m=(d<<1),i=0;i<n;i+=m){
            if(ty==1){
                for(int j=0;j<d;j++){
                    LL x=a[i+j],y=a[i+j+d];
                    a[i+j]=x+y;
                    a[i+j+d]=x-y;
                    //xor:a[i+j]=x+y,a[i+j+d]=x-y;
                    //and:a[i+j]=x+y;
                    //or:a[i+j+d]=x+y;
                }
            }else{
                for(int j=0;j<d;j++){
                    LL x=a[i+j],y=a[i+j+d];
                    a[i+j]=(x+y)/2;
                    a[i+j+d]=(x-y)/2;
                    //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
                    //and:a[i+j]=x-y;
                    //or:a[i+j+d]=y-x;
                }
            }
        }
    }
}
    FWT(a,1<<n,1);
    FWT(b,1<<n,1);
    for(int i=0;i<(1<<n);i++)
        c[i]=a[i]*b[i];
```

```
    FWT(c,1<<n,-1);
```

1.9 单纯形

```
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;
    }
}
```

Chapter 2

数论

2.1 大整数相乘取模

```
// x 与 y 须非负
long long mult(long long x, long long y, long long MODN) {
    long long t = (x * y - (long long)((long double)x / MODN * y + 1e-3) * MODN) % MODN;
    return t < 0 ? t + MODN : t;
}
```

2.2 EX-GCD

```
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;
    }
}
```

2.3 Miller-rabin

```
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;
}
```

2.4 Pollard-rho.cpp

```
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);
}
```

2.5 非互质 CRT

first is remainder, second is module

```
inline void fix(LL &x, LL y) {
    x = (x % y + y) % y;
}
bool solve(int n, std::pair<LL, LL> a[],
                  std::pair<LL, LL> &ans) {
    ans = std::make_pair(1, 1);
    for (int i = 0; i < n; ++i) {
        LL num, y;
        euclid(ans.second, a[i].second, num, y);
        LL divisor = std::__gcd(ans.second, a[i].second);
        if ((a[i].first - ans.first) % divisor) {
            return false;
        }
        num *= (a[i].first - ans.first) / divisor;
        fix(num, a[i].second);
        ans.first += ans.second * num;
        ans.second *= a[i].second / divisor;
        fix(ans.first, ans.second);
    }
    return true;
}
```

2.6 非互质 CRT -zky

```
//merge Ax=B and ax=b to A'x=B'
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;
}
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;
}
```

2.7 Pell 方程

```
// x_{k+1} = x_0 x_k + n y_0 y_k
// y_{k+1} = x_0 y_k + y_0 x_k
// n is not the index of which you want
pair<ll, ll> pell(ll n) {
    static ll p[N], q[N], g[N], h[N], a[N];
    p[1] = q[0] = h[1] = 1; p[0] = q[1] = g[1] = 0;
    a[2] = (ll)(floor(sqrtl(n) + 1e-7L));
    for(int i = 2; ; i ++) {
        g[i] = -g[i - 1] + a[i] * h[i - 1];
        h[i] = (n - g[i] * g[i]) / h[i - 1];
        a[i + 1] = (g[i] + a[2]) / h[i];
        p[i] = a[i] * p[i - 1] + p[i - 2];
        q[i] = a[i] * q[i - 1] + q[i - 2];
        if(p[i] * p[i] - n * q[i] * q[i] == 1)
            return {p[i], q[i]};
    }
} // x^2 - n*y^2 = 1 最小正整数根，n 为完全平方数时无解
```

2.8 Simpson

```
// 三次函数，两倍精度拟合
// error = (r-l)^5/6480 |f^{(4)}|
// \int_a^b f(x) dx ≈ (b-a)/8 [f(a) + 3f((2a+b)/3) + 3f((a+2b)/3) + f(b)]
// 三次函数拟合 error = 1/90 ((r-l)/2)^5 |f^{(4)}|
d simpson(d fl,d fr,d fmid,d l,d r) {
    return (fl+fr+4.0*fmid)*(r-l)/6.0; }
d rsimpson(d slr,d fl,d fr,d fmid,d l,d r) {
    d mid = (l+r)/2,fml = f((l+mid)/2),fmr = f((mid+r)/2);
    d slm = simpson(fl,fmid,fml,l,mid);
    d smr = simpson(fmid,fr,fmr,mid,r);
    if(fabs(slr - smr - slm) / slr < eps)return slm + smr;
    return rsimpson(slm,fl,fmid,fml,l,mid)+
        rsimpson(smr,fmid,fr,fmr,mid,r);
}
```

2.9 解一元三次方程

听说极端情况精度不够

2.10 线段下整点

solve for $\sum_{i=0}^{n-1} \lfloor \frac{a+bi}{m} \rfloor$, $n, m, a, b > 0$

```
LL solve(LL n,LL a,LL b,LL m){
    if(b==0) return n*(a/m);
    if(a>=m) return n*(a/m)+solve(n,a%m,b,m);
    if(b>=m) return (n-1)*n/2*(b/m)+solve(n,a,b%m,m);
    return solve((a+b*n)/m,(a+b*n)%m,m,b);
}
```

2.11 线性同余不等式

2.12 EX-BSGS -zzq

```
/*
 * EX_BSGS
 * a^x = b (mod p)
 * p may not be a prime
 */

ll qpow(ll a, ll x, ll Mod) {
    ll res = 1;
    for (; x; x >>= 1) {
        if (x & 1) res = res * a % Mod;
        a = a * a % Mod;
    }
    return res;
}

std::unordered_map<int, int> mp;

ll exbsgs(ll a, ll b, ll p) {
    if (b == 1) return 0;
    ll t, d = 1, k = 0;
    while ((t = std::__gcd(a, p)) != 1) {
        if (b % t) return -1;
        ++k, b /= t, p /= t, d = d * (a / t) % p;
        if (b == d) return k;
    }
    mp.clear();
    ll m = std::ceil(std::sqrt(p));
    ll a_m = qpow(a, m, p);
    ll mul = b;
    for (ll j = 1; j <= m; ++j) {
        mul = mul * a % p;
        mp[mul] = j;
    }
    for (ll i = 1; i <= m; ++i) {
        d = d * a_m % p;
        if (mp.count(d)) return i * m - mp[d] + k;
    }
    return -1;
}
```

2.13 EX-BSGS -zky

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

2.14 分治乘法

```
(a+b)(c+d) = ac+(bc+ad)+bd = 2ac-(a-b)(c-d)+2bd

x = x^m m=(n+1)/2
(ax+b)(cx+d) = x^2ac + x(bc+ad) + bd = x^2ac + x(ac + bd - (a-b)(c-d)) + bd
```

2.15 组合数模 p^k

```
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;
}
```

Chapter 3

图论

3.1 图论基础

```
struct Graph {  // Remember to call .init()!
    int e, nxt[M], v[M], adj[N], n;
    bool base;
    __inline void init(bool _base, int _n = 0) {
        n = _n; base = _base;
        e = 0; memset(adj + base, -1, sizeof(*adj) * n);
    }
    __inline int new_node() {
        adj[n + base] = -1;
        return n++ + base;
    }
    __inline void ins(int u0, int v0) {  // directional
        v[e] = v0; nxt[e] = adj[u0]; adj[u0] = e++;
    }
    __inline void bi_ins(int u0, int v0) {  // bi-directional
        ins(u0, v0); ins(v0, u0);
    }
};
```

3.2 坚固无敌的点双 -zzq

```
typedef std::pair<int, int> pii;
#define mkpair std::make_pair
int n, m;
std::vector<int> G[MAXN];
int dfn[MAXN], low[MAXN], bcc_id[MAXN], bcc_cnt, stamp;
bool iscut[MAXN];
std::vector<int> bcc[MAXN]; // Unnecessary
pii stk[MAXN]; int stk_top;
// Use a handwritten structure to get higher efficiency
void Tarjan(int now, int fa) {
    int child = 0;
    dfn[now] = low[now] = ++stamp;
    for (int to: G[now]) {
        if (!dfn[to]) {
            stk[++stk_top] = mkpair(now, to); ++child;
            Tarjan(to, now);
            low[now] = std::min(low[now], low[to]);
            if (low[to] >= dfn[now]) {
                iscut[now] = 1;
                bcc[++bcc_cnt].clear();
                while (1) {
                    pii tmp = stk[stk_top--];
```

```
                    if (bcc_id[tmp.first] != bcc_cnt) {
                        bcc[bcc_cnt].push_back(tmp.first);
                        bcc_id[tmp.first] = bcc_cnt;
                    }
                    if (bcc_id[tmp.second] != bcc_cnt) {
                        bcc[bcc_cnt].push_back(tmp.second);
                        bcc_id[tmp.second] = bcc_cnt;
                    }
                    if (tmp.first == now && tmp.second == to)
                        break;
                }
            }
        }
        else if (dfn[to] < dfn[now] && to != fa) {
            stk[++stk_top] = mkpair(now, to);
            low[now] = std::min(low[now], dfn[to]);
        }
    }
    if (!fa && child == 1) iscut[now] = 0;
}

void PBCC() {
    memset(dfn, 0, sizeof dfn);
    memset(low, 0, sizeof low);
    memset(iscut, 0, sizeof iscut);
    memset(bcc_id, 0, sizeof bcc_id);
    stamp = bcc_cnt = stk_top = 0;
    for (int i = 1; i <= n; ++i)
        if (!dfn[i]) Tarjan(i, 0);
}
```

3.3 坚固无敌的边双 -zzq

```
int n, m;
int head[MAXN], nxt[MAXM << 1], to[MAXM << 1], ed;
// Opposite edge exists, set head[] to -1.
int dfn[MAXN], low[MAXN], bcc_id[MAXN], bcc_cnt, stamp;
bool isbridge[MAXM << 1], vis[MAXN];
std::vector<int> bcc[MAXN];
void Tarjan(int now, int fa) {
    dfn[now] = low[now] = ++stamp;
    for (int i = head[now]; ~i; i = nxt[i]) {
        if (!dfn[to[i]]) {
            Tarjan(to[i], now);
            low[now] = std::min(low[now], low[to[i]]);
            if (low[to[i]] > dfn[now])
                isbridge[i] = isbridge[i ^ 1] = 1;
        }
        else if (dfn[to[i]] < dfn[now] && to[i] != fa)
            low[now] = std::min(low[now], dfn[to[i]]);
    }
}
void DFS(int now) {
    vis[now] = 1;
    bcc[bcc_id[now] = bcc_cnt].push_back(now);
    for (int i = head[now]; ~i; i = nxt[i]) {
        if (isbridge[i]) continue;
        if (!vis[to[i]]) DFS(to[i]);
    }
```

```
}
void EBCC() {
    memset(dfn, 0, sizeof dfn);
    memset(low, 0, sizeof low);
    memset(isbridge, 0, sizeof isbridge);
    memset(bcc_id, 0, sizeof bcc_id);
    bcc_cnt = stamp = 0;
    for (int i = 1; i <= n; ++i)
        if (!dfn[i]) Tarjan(i, 0);
    memset(vis, 0, sizeof vis);
    for (int i = 1; i <= n; ++i)
        if (!vis[i]) {
            ++bcc_cnt; DFS(i);
        }
}
```

3.4 坚固无敌的点双 -jzh

```
const bool BCC_VERTEX = 0, BCC_EDGE = 1;
struct BCC {  // N = N0 + M0. Remember to call init(&raw_graph).
    Graph *g, forest; // g is raw graph ptr.
    int dfn[N], DFN, low[N];
    int stack[N], top;
    int expand_to[M];       // Where edge i is expanded to in expaned graph.
    // Vertex i expaned to i.
    int compress_to[N];  // Where vertex i is compressed to.
    bool cut[N], compress_cut[N], branch[M], vis[N], flag;
    //std::vector<int> BCC_component[N];  // Cut vertex belongs to none.
    __inline void init(Graph *raw_graph) {
        g = raw_graph;
    }
    void DFS(int u, int pe) {
        dfn[u] = low[u] = ++DFN; cut[u] = false;
        if (!~g->adj[u]) {
            cut[u] = 1;
            compress_to[u] = forest.new_node();
            compress_cut[compress_to[u]] = 1;
        }
        for (int e = g->adj[u]; ~e; e = g->nxt[e]) {
            int v = g->v[e];
            if ((e ^ pe) > 1 && dfn[v] > 0 && dfn[v] < dfn[u]) {
                stack[top++] = e;
                low[u] = std::min(low[u], dfn[v]);
            }
            else if (!dfn[v]) {
                stack[top++] = e; branch[e] = 1;
                DFS(v, e);
                low[u] = std::min(low[v], low[u]);
                if (low[v] >= dfn[u]) {
                    if ((pe == -1 && flag || pe != -1) && !cut[u]) {
                        cut[u] = 1;
                        compress_to[u] = forest.new_node();
                        compress_cut[compress_to[u]] = 1;
                    }
                    int cc = forest.new_node();
                    if (cut[u]) forest.bi_ins(compress_to[u], cc);
                    compress_cut[cc] = 0;
                    //BCC_component[cc].clear();
                    do {
```

```
                        int cur_e = stack[--top];
                        compress_to[expand_to[cur_e]] = cc;
                        compress_to[expand_to[cur_e^1]] = cc;
                        if (branch[cur_e]) {
                            int v = g->v[cur_e];
                            if (cut[v]) {
                                forest.bi_ins(cc, compress_to[v]);
                            } else {
                                //BCC_component[cc].push_back(v);
                                compress_to[v] = cc;
                            }
                        }
                    } while (stack[top] != e);
                    if (pe == -1 && !flag) {
                        compress_to[u] = cc;
                    }
                }
            }
        }
    }
    inline bool dfs(int u, int pe) {
        vis[u] = 1;
        int d = 0;
        for (int e = g->adj[u]; ~e; e = g->nxt[e]) {
            int v = g -> v[e];
            if (!vis[v]) {
                ++d; dfs(v, e);
            }
        }
        return pe == -1 ? d > 1 : 0;
    }
    void solve() {
        forest.init(g->base);
        int n = g->n;
        for (int i = 0; i < g->e; i++) {
            expand_to[i] = g->new_node();
        }
        memset(vis + g -> base, 0, sizeof(*vis) * n);
        memset(branch, 0, sizeof(*branch) * g->e);
        memset(dfn + g->base, 0, sizeof(*dfn) * n); DFN = 0;
        for (int i = 0; i < n; i++)
            if (!dfn[i + g->base]) {
                top = 0;
                flag = dfs(i + g -> base, -1);
                DFS(i + g->base, -1);
            }
    }
} bcc;
```

3.5 坚固无敌的边双 -jzh

```
struct BCC {
    Graph *g, forest;
    int dfn[N], low[N], stack[N], tot[N], belong[N], vis[N], top, dfs_clock;
    // tot[] is the size of each BCC, belong[] is the BCC that each node belongs to
    pair<int, int > ori[M]; // bridge in raw_graph(raw node)
    bool is_bridge[M];
    __inline void init(Graph *raw_graph) {
        g = raw_graph;
```

```
        memset(is_bridge, false, sizeof(*is_bridge) * g -> e);
        memset(vis + g -> base, 0, sizeof(*vis) * g -> n);
    }
    void tarjan(int u, int from) {
        dfn[u] = low[u] = ++dfs_clock; vis[u] = 1; stack[++top] = u;
        for (int p = g -> adj[u]; ~p; p = g -> nxt[p]) {
            if ((p ^ 1) == from) continue;
            int v = g -> v[p];
            if (vis[v]) {
                if (vis[v] == 1) low[u] = min(low[u], dfn[v]);
            } else {
                tarjan(v, p);
                low[u] = min(low[u], low[v]);
                if (low[v] > dfn[u]) is_bridge[p / 2] = true;
            }
        }
        if (dfn[u] != low[u]) return;
        tot[forest.new_node()] = 0;
        do {
            belong[stack[top]] = forest.n;
            vis[stack[top]] = 2;
            tot[forest.n]++;
            --top;
        } while (stack[top + 1] != u);
    }
    void solve() {
        forest.init(g -> base);
        int n = g -> n;
        for (int i = 0; i < n; ++i)
            if (!vis[i + g -> base]) {
                top = dfs_clock = 0;
                tarjan(i + g -> base, -1);
            }
        for (int i = 0; i < g -> e / 2; ++i)
            if (is_bridge[i]) {
                int e = forest.e;
                forest.bi_ins(belong[g -> v[i * 2]], belong[g -> v[i * 2 + 1]], g -> w[i *
                   ↪ 2]);
                ori[e] = make_pair(g -> v[i * 2 + 1], g -> v[i * 2]);
                ori[e + 1] = make_pair(g -> v[i * 2], g -> v[i * 2 + 1]);
            }
    }
} bcc;
```

3.6 2-sat

```
//清点清边要两倍
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) {
        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);
    }
}
bool solve() {
    int counter = n + n + 1;
    stamp = top = comps = 0;
    std::fill(dfn, dfn + counter, 0);
    std::fill(comp, comp + counter, 0);
    for (int i = 0; i < counter; ++i) {
        if (!dfn[i]) {
            tarjan(i);
        }
    }
    for (int i = 0; i < n; ++i) {
        if (comp[i << 1] == comp[i << 1 | 1]) {
            return false;
        }
        answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
    }
    return true;
}
```

3.7 闪电二分图匹配

```
int matchx[N], matchy[N], level[N];
vector<int> edge[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() {
    memset(matchx, -1, sizeof(*matchx) * n);
    memset(matchy, -1, sizeof(*matchy) * m);
    for (int ans = 0; ; ) {
        std::vector<int> q;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1) {
                level[i] = 0;
                q.push_back(i);
            } else level[i] = -1;
        }
```

```
        for (int head = 0; head < (int)q.size(); ++head) {
            int x = q[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;
                    q.push_back(w);
                }
            }
        }
        int delta = 0;
        for (int i = 0; i < n; ++i)
            if (matchx[i] == -1 && dfs(i)) ++delta;
        if (delta == 0) return ans; else ans += delta;
    }
}
```

3.8 一般图匹配

```
// 0-base, match[u] is linked to u
vector<int> lnk[MAXN];
int match[MAXN], Queue[MAXN], pred[MAXN], base[MAXN], head, tail, sta, fin, nbase;
bool inQ[MAXN], inB[MAXN];
inline void push(int u) {
    Queue[tail++] = u; inQ[u] = 1;
}
inline int pop() {
    return Queue[head++];
}
inline int FindCA(int u, int v) {
    static bool inP[MAXN];
    fill(inP, inP + n, false);
    while (1) {
        u = base[u]; inP[u] = 1;
        if(u == sta) break;
        u = pred[match[u]];
    }
    while (1) {
        v = base[v];
        if (inP[v]) break;
        v = pred[match[v]];
    }
    return v;
}
inline void RT(int u) {
    int v;
    while (base[u] != nbase) {
        v = match[u];
        inB[base[u]] = inB[base[v]] = 1;
        u = pred[v];
        if (base[u] != nbase) pred[u] = v;
    }
}
inline void BC(int u, int v) {
    nbase = FindCA(u, v);
    fill(inB, inB + n, 0);
    RT(u); RT(v);
    if (base[u] != nbase) pred[u] = v;
```

```
    if (base[v] != nbase) pred[v] = u;
    for (int i = 0; i < n; ++i)
        if (inB[base[i]]) {
            base[i] = nbase;
            if (!inQ[i]) push(i);
        }
}
bool FindAP(int u) {
    bool found = false;
    for (int i = 0; i < n; ++i) {
        pred[i] = -1; base[i] = i; inQ[i] = 0;
    }
    sta = u; fin = -1; head = tail = 0; push(sta);
    while (head < tail) {
        int u = pop();
        for (int i = (int)lnk[u].size() - 1; i >= 0; --i) {
            int v = lnk[u][i];
            if (base[u] != base[v] && match[u] != v) {
                if (v == sta || match[v] >= 0 && pred[match[v]] >= 0) BC(u, v);
                else if (pred[v] == -1) {
                    pred[v] = u;
                    if (match[v] >= 0) push(match[v]);
                    else {
                        fin = v;
                        return true;
                    }
                }
            }
        }
    }
    return found;
}
inline void AP() {
    int u = fin, v, w;
    while (u >= 0) {
        v = pred[u]; w = match[v];
        match[v] = u; match[u] = v;
        u = w;
    }
}
inline int FindMax() {
    for (int i = 0; i < n; ++i) match[i] = -1;
    for (int i = 0; i < n; ++i)
        if (match[i] == -1 && FindAP(i)) AP();
    int ans = 0;
    for (int i = 0; i < n; ++i) ans += (match[i] != -1);
    return ans;
}
```

3.9 一般图最大权匹配

```
//maximum weight blossom,  change g[u][v].w to INF - g[u][v].w when minimum weight blossom
  ↪ is needed
//type of ans is long long
//replace all int to long long if weight of edge is long long

struct WeightGraph {
    static const int INF = INT_MAX;
    static const int MAXN = 400;
```

```cpp
struct edge{
    int u, v, w;
    edge() {}
    edge(int u, int v, int w): u(u), v(v), w(w) {}
};
int n, n_x;
edge g[MAXN * 2 + 1][MAXN * 2 + 1];
int lab[MAXN * 2 + 1];
int match[MAXN * 2 + 1], slack[MAXN * 2 + 1], st[MAXN * 2 + 1], pa[MAXN * 2 + 1];
int flower_from[MAXN * 2 + 1][MAXN+1], S[MAXN * 2 + 1], vis[MAXN * 2 + 1];
vector<int> flower[MAXN * 2 + 1];
queue<int> q;
inline int e_delta(const edge &e){ // does not work inside blossoms
    return lab[e.u] + lab[e.v] - g[e.u][e.v].w * 2;
}
inline void update_slack(int u, int x){
    if(!slack[x] || e_delta(g[u][x]) < e_delta(g[slack[x]][x]))
        slack[x] = u;
}
inline void set_slack(int x){
    slack[x] = 0;
    for(int u = 1;u <= n; ++u)
        if(g[u][x].w > 0 && st[u] != x && S[st[u]] == 0)
            update_slack(u, x);
}
void q_push(int x){
    if(x <= n)q.push(x);
    else for(size_t i = 0;i < flower[x].size(); i++)
        q_push(flower[x][i]);
}
inline void set_st(int x, int b){
    st[x]=b;
    if(x > n) for(size_t i = 0;i < flower[x].size(); ++i)
                set_st(flower[x][i], b);
}
inline int get_pr(int b, int xr){
    int pr = find(flower[b].begin(), flower[b].end(), xr) - flower[b].begin();
    if(pr % 2 == 1){
        reverse(flower[b].begin() + 1, flower[b].end());
        return (int)flower[b].size() - pr;
    } else return pr;
}
inline void set_match(int u, int v){
    match[u]=g[u][v].v;
    if(u > n){
        edge e=g[u][v];
        int xr = flower_from[u][e.u], pr=get_pr(u, xr);
        for(int i = 0;i < pr; ++i)
            set_match(flower[u][i], flower[u][i ^ 1]);
        set_match(xr, v);
        rotate(flower[u].begin(), flower[u].begin()+pr, flower[u].end());
    }
}
inline void augment(int u, int v){
    for(; ; ){
        int xnv=st[match[u]];
        set_match(u, v);
        if(!xnv)return;
        set_match(xnv, st[pa[xnv]]);
        u=st[pa[xnv]], v=xnv;
```

```
        }
    }
    inline int get_lca(int u, int v){
        static int t=0;
        for(++t; u || v; swap(u, v)){
            if(u == 0)continue;
            if(vis[u] == t)return u;
            vis[u] = t;
            u = st[match[u]];
            if(u) u = st[pa[u]];
        }
        return 0;
    }
    inline void add_blossom(int u, int lca, int v){
        int b = n + 1;
        while(b <= n_x && st[b]) ++b;
        if(b > n_x) ++n_x;
        lab[b] = 0, S[b] = 0;
        match[b] = match[lca];
        flower[b].clear();
        flower[b].push_back(lca);
        for(int x = u, y; x != lca; x = st[pa[y]]) {
            flower[b].push_back(x),
            flower[b].push_back(y = st[match[x]]),
            q_push(y);
        }
        reverse(flower[b].begin() + 1, flower[b].end());
        for(int x = v, y; x != lca; x = st[pa[y]]) {
            flower[b].push_back(x),
            flower[b].push_back(y = st[match[x]]),
            q_push(y);
        }
        set_st(b, b);
        for(int x = 1; x <= n_x; ++x) g[b][x].w = g[x][b].w = 0;
        for(int x = 1; x <= n; ++x) flower_from[b][x] = 0;
        for(size_t i = 0 ; i < flower[b].size(); ++i){
            int xs = flower[b][i];
            for(int x = 1; x <= n_x; ++x)
                if(g[b][x].w == 0 || e_delta(g[xs][x]) < e_delta(g[b][x]))
                    g[b][x] = g[xs][x], g[x][b] = g[x][xs];
            for(int x = 1;x <= n; ++x)
                if(flower_from[xs][x]) flower_from[b][x] = xs;
        }
        set_slack(b);
    }
    inline void expand_blossom(int b){ // S[b]  ==  1
        for(size_t i = 0; i < flower[b].size(); ++i)
            set_st(flower[b][i], flower[b][i]);
        int xr = flower_from[b][g[b][pa[b]].u], pr = get_pr(b, xr);
        for(int i = 0; i < pr; i += 2){
            int xs = flower[b][i], xns = flower[b][i + 1];
            pa[xs] = g[xns][xs].u;
            S[xs] = 1, S[xns] = 0;
            slack[xs] = 0, set_slack(xns);
            q_push(xns);
        }
        S[xr] = 1, pa[xr] = pa[b];
        for(size_t i = pr + 1;i < flower[b].size(); ++i){
            int xs = flower[b][i];
            S[xs] = -1, set_slack(xs);
```

```
        }
        st[b] = 0;
    }
    inline bool on_found_edge(const edge &e){
        int u = st[e.u], v = st[e.v];
        if(S[v] == -1){
            pa[v] = e.u, S[v] = 1;
            int nu = st[match[v]];
            slack[v] = slack[nu] = 0;
            S[nu] = 0, q_push(nu);
        }else if(S[v] == 0){
            int lca = get_lca(u, v);
            if(!lca) return augment(u, v), augment(v, u), true;
            else add_blossom(u, lca, v);
        }
        return false;
    }
    inline bool matching(){
        memset(S + 1, -1, sizeof(int) * n_x);
        memset(slack + 1, 0, sizeof(int) * n_x);
        q = queue<int>();
        for(int x = 1;x <= n_x; ++x)
            if(st[x] == x && !match[x]) pa[x]=0, S[x]=0, q_push(x);
        if(q.empty())return false;
        for(;;){
            while(q.size()){
                int u = q.front();q.pop();
                if(S[st[u]] == 1)continue;
                for(int v = 1;v <= n; ++v)
                    if(g[u][v].w > 0 && st[u] != st[v]){
                        if(e_delta(g[u][v]) == 0){
                            if(on_found_edge(g[u][v]))return true;
                        }else update_slack(u, st[v]);
                    }
            }
            int d = INF;
            for(int b = n + 1; b <= n_x;++b)
                if(st[b] == b && S[b] == 1)d = min(d, lab[b]/2);
            for(int x = 1; x <= n_x; ++x)
                if(st[x] == x && slack[x]){
                    if(S[x] == -1)d = min(d, e_delta(g[slack[x]][x]));
                    else if(S[x] == 0)d = min(d, e_delta(g[slack[x]][x])/2);
                }
            for(int u = 1; u <= n; ++u){
                if(S[st[u]] == 0){
                    if(lab[u] <= d)return 0;
                    lab[u] -= d;
                }else if(S[st[u]] == 1)lab[u] += d;
            }
            for(int b = n+1; b <= n_x; ++b)
                if(st[b] == b){
                    if(S[st[b]] == 0) lab[b] += d * 2;
                    else if(S[st[b]] == 1) lab[b] -= d * 2;
                }
            q=queue<int>();
            for(int x = 1; x <= n_x; ++x)
                if(st[x] == x && slack[x] && st[slack[x]] != x && e_delta(g[slack[x]][x]) == 0)
                    if(on_found_edge(g[slack[x]][x]))return true;
            for(int b = n + 1; b <= n_x; ++b)
```

```
                if(st[b] == b && S[b] == 1 && lab[b] == 0)expand_blossom(b);
        }
        return false;
    }
    inline pair<long long, int> solve(){
        memset(match + 1, 0, sizeof(int) * n);
        n_x = n;
        int n_matches = 0;
        long long tot_weight = 0;
        for(int u = 0; u <= n; ++u) st[u] = u, flower[u].clear();
        int w_max = 0;
        for(int u = 1; u <= n; ++u)
            for(int v = 1; v <= n; ++v){
                flower_from[u][v] = (u == v ? u : 0);
                w_max = max(w_max, g[u][v].w);
            }
        for(int u = 1; u <= n; ++u) lab[u] = w_max;
        while(matching()) ++n_matches;
        for(int u = 1; u <= n; ++u)
            if(match[u] && match[u] < u)
                tot_weight += g[u][match[u]].w;
        return make_pair(tot_weight, n_matches);
    }
    inline void init(){
        for(int u = 1; u <= n; ++u)
            for(int v = 1; v <= n; ++v)
                g[u][v]=edge(u, v, 0);
    }
};
```

3.10 有根树 hash

```
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++;
    }
}
```

3.11 无向图最小割

3.12 必经点 Dominator-tree

```
//solve(s, n, raw_g): s is the root and base accords to base of raw_g
//idom[x] will be x if x does not have a dominator,and will be -1 if x is not reachable from s.
struct dominator_tree {
    int base, dfn[N], sdom[N], idom[N], id[N], f[N], fa[N], smin[N], stamp;
    Graph *g;
    void predfs(int u) {
        id[dfn[u] = stamp++] = u;
        for (int i = g -> adj[u]; ~i; i = g -> nxt[i]) {
            int v = g -> v[i];
            if (dfn[v] < 0) f[v] = u, predfs(v);
        }
    }
    int getfa(int u) {
        if (fa[u] == u) return u;
        int ret = getfa(fa[u]);
        if (dfn[sdom[smin[fa[u]]]] < dfn[sdom[smin[u]]])
            smin[u] = smin[fa[u]];
        return fa[u] = ret;
    }
    void solve (int s, int n, Graph *raw_graph) {
        g = raw_graph;
        base = g -> base;
        memset(dfn + base, -1, sizeof(*dfn) * n);
        memset(idom + base, -1, sizeof(*idom) * n);
        static Graph pred, tmp;
        pred.init(base, n);
        for (int i = 0; i < n; ++i) {
            for (int p = g -> adj[i + base]; ~p; p = g -> nxt[p])
                pred.ins(g -> v[p], i + base);
        }
        stamp = 0; tmp.init(base, n); predfs(s);
        for (int i = 0; i < stamp; ++i) {
            fa[id[i]] = smin[id[i]] = id[i];
        }
        for (int o = stamp - 1; o >= 0; --o) {
            int x = id[o];
            if (o) {
                sdom[x] = f[x];
                for (int i = pred.adj[x]; ~i; i = pred.nxt[i]) {
                    int p = pred.v[i];
```

```
                    if (dfn[p] < 0) continue;
                    if (dfn[p] > dfn[x]) {
                        getfa(p);
                        p = sdom[smin[p]];
                    }
                    if (dfn[sdom[x]] > dfn[p]) sdom[x] = p;
                }
                tmp.ins(sdom[x], x);
            }
            while (~tmp.adj[x]) {
                int y = tmp.v[tmp.adj[x]];
                tmp.adj[x] = tmp.nxt[tmp.adj[x]];
                getfa(y);
                if (x != sdom[smin[y]]) idom[y] = smin[y];
                else idom[y] = x;
            }
            for (int i = g -> adj[x]; ~i; i = g -> nxt[i])
                if (f[g -> v[i]] == x) fa[g -> v[i]] = x;
        }
        idom[s] = s;
        for (int i = 1; i < stamp; ++i) {
            int x = id[i];
            if (idom[x] != sdom[x]) idom[x] = idom[idom[x]];
        }
    }
};
```

3.13 K 短路

```
//需保证 GivenEdge 里面边的顺序和 Edge 中一样
//两个优先队列要考虑大根还是小根
//heap 总是小根堆
//dij 不能求正权最长路
//INF or -INF

typedef long long LL;
MAXN, MAXK, MAXN, INF //int or LL, it depends
const int MAXNODE = MAXN + MAXM * 2;    // m + nlgm ???
bool used[MAXN];
int n, m, cnt, S, T, Kth, N;// m is number of all edges
int rt[MAXN], seq[MAXN], adj[MAXN], from[MAXN], dep[MAXN];
LL dist[MAXN], w[MAXM], ans[MAXK];
struct GivenEdge {  //edge given from origin input
    int u, v, w;
    GivenEdge() {};
    GivenEdge(int _u, int _v, int _w): u(_u), v(_v), w(_w) {};
} edge[MAXM];
struct Edge {
    int v, nxt, w;
    Edge() {};
    Edge(int _v, int _nxt, int _w): v(_v), nxt(_nxt), w(_w) {};
} e[MAXM];
inline void addedge(int u, int v, int w) {
    e[++cnt] = Edge(v, adj[u], w); adj[u] = cnt;
}
inline void dij(int S) {  //dij in original graph, spfa if needed
    for (int i = 1; i <= N; ++i) {
        dist[i] = INF; dep[i] = INF; used[i] = false; from[i] = 0;
    }
```

```
    static priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > >
      ↪ hp;
    while (!hp.empty()) hp.pop();
    hp.push(make_pair(dist[S] = 0, S));
    dep[S] = 1;
    while (!hp.empty()) {
        pair<LL, int> now = hp.top(); hp.pop();
        int u = now.second;
        if (used[u]) continue;
            else used[u] = true;
        for (int p = adj[u]; p; p = e[p].nxt) {
            int v = e[p].v;
            if (dist[u] + e[p].w < dist[v]) {  //different when max or min
                dist[v] = dist[u] + e[p].w;
                dep[v] = dep[u] + 1;
                from[v] = p;
                hp.push(make_pair(dist[v], v));
            }
        }
    }
    for (int i = 1; i <= m; ++i) w[i] = 0;
    for (int i = 1; i <= N; ++i)
        if (from[i]) w[from[i]] = -1;
    for (int i = 1; i <= m; ++i) {
        if (~w[i] && dist[edge[i].u] < INF && dist[edge[i].v] < INF) {
            w[i] = -dist[edge[i].u] + (dist[edge[i].v] + edge[i].w);    //different when
              ↪ max or min
        } else {
            w[i] = -1;
        }
    }
}
inline bool cmp_dep(int p, int q) {
    return dep[p] < dep[q];
}
struct Heap {
    LL key;
    int id, lc, rc, dist;
    Heap() {};
    Heap(LL k, int i, int l, int r, int d): key(k), id(i), lc(l), rc(r), dist(d) {};
    inline void clear() {
        key = 0;
        id = lc = rc = dist = 0;
    }
} hp[MAXNODE];

inline int merge_simple(int u, int v) {
    if (!u) return v;
    if (!v) return u;
    if (hp[u].key > hp[v].key) {
        swap(u, v);
    }
    hp[u].rc = merge_simple(hp[u].rc, v);
    if (hp[hp[u].lc].dist < hp[hp[u].rc].dist) {
        swap(hp[u].lc, hp[u].rc);
    }
    hp[u].dist = hp[hp[u].rc].dist + 1;
    return u;
}

```

```cpp
inline int merge_full(int u, int v) {
    if (!u) return v;
    if (!v) return u;
    if (hp[u].key > hp[v].key) {
        swap(u, v);
    }
    int nownode = ++cnt;
    hp[nownode] = hp[u];
    hp[nownode].rc = merge_full(hp[nownode].rc, v);
    if (hp[hp[nownode].lc].dist < hp[hp[nownode].rc].dist) {
        swap(hp[nownode].lc, hp[nownode].rc);
    }
    hp[nownode].dist = hp[hp[nownode].rc].dist + 1;
    return nownode;
}
priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > > Q;
int main() {
    scanf("%d%d%d", &n, &m, &Kth);
    for (int i = 1; i <= m; ++i) {
        int u, v, w;
        scanf("%d%d%d", &u, &v, &w);
        edge[i] = {u, v, w};
    }
    N = ; S = ; T = ;
    memset(adj, 0, sizeof(*adj) * (N + 1));
    cnt = 0;
    for (int i = 1; i <= m; ++i) {
        addedge(edge[i].v, edge[i].u, edge[i].w); // important!!! reverse the edge
    }
    dij(T);
    if (dist[S] == INF) {   //must judge before building heaps; -INF if max kth
        ...
        return 0;
    }
    for (int i = 1; i <= N; ++i) {
        seq[i] = i;
    }
    sort(seq + 1, seq + N + 1, cmp_dep);

    cnt = 0;
    memset(adj, 0, sizeof(*adj) * (N + 1));
    memset(rt, 0, sizeof(*rt) * (N + 1));
    for (int i = 1; i <= m; ++i) {
        addedge(edge[i].u, edge[i].v, edge[i].w);
    }
    rt[T] = cnt = 0;  // now cnt is total nodes in heaps
    hp[0].dist = -1;
    for (int i = 1; i <= N; ++i) {
        int u = seq[i], v = edge[from[u]].v;
        rt[u] = 0;
        for (int p = adj[u]; p; p = e[p].nxt) {
            if (~w[p]) {
                hp[++cnt] = Heap(w[p], p, 0, 0, 0);
                rt[u] = merge_simple(rt[u], cnt);
            }
        }
        if (i == 1) continue;
        rt[u] = merge_full(rt[u], rt[v]);
    }
    while (!Q.empty()) Q.pop();
```

```
    Q.push(make_pair(dist[S], 0));
    edge[0].v = S;
    for (int kth = 1; kth <= Kth; ++kth) {
        if (Q.empty()) {
            ans[kth] = -1;
            continue;
        }
        pair<LL, int> now = Q.top(); Q.pop();
        ans[kth] = now.first;
        int p = now.second;
        if (hp[p].lc) {
            Q.push(make_pair(+hp[hp[p].lc].key + now.first - hp[p].key,
              ↪ hp[p].lc));//different when max or min
        }
        if (hp[p].rc) {
            Q.push(make_pair(+hp[hp[p].rc].key + now.first - hp[p].key,
              ↪ hp[p].rc));//different when max or min
        }
        if (rt[edge[hp[p].id].v]) {
            Q.push(make_pair(hp[rt[edge[hp[p].id].v]].key + now.first,
              ↪ rt[edge[hp[p].id].v]));//different when max or min
        }
    }
    ...
    for (int i = 1; i <= cnt; ++i) {
        hp[i].clear();
    }
}
```

3.14 最大团搜索

```
// Super Fast Maximum Clique
// To Build Graph: Maxclique(Edges, Number of Nodes)
// To Get Answer: mcqdyn(AnswerNodes Index Array, AnswserLength)
typedef bool BB[N];
struct Maxclique {
    const BB* e; int pk, level; const float Tlimit;
    struct Vertex{ int i, d; Vertex(int i):i(i),d(0){} };
    typedef vector<Vertex> Vertices; typedef vector<int> ColorClass;
    Vertices V; vector<ColorClass> C; ColorClass QMAX, Q;
    static bool desc_degree(const Vertex &vi, const Vertex &vj){
        return vi.d > vj.d;
    }
    void init_colors(Vertices &v){
        const int max_degree = v[0].d;
        for(int i = 0; i < (int)v.size(); i++) v[i].d = min(i, max_degree) + 1;
    }
    void set_degrees(Vertices &v){
        for(int i = 0, j; i < (int)v.size(); i++)
            for(v[i].d = j = 0; j < int(v.size()); j++)
                v[i].d += e[v[i].i][v[j].i];
    }
    struct StepCount{ int i1, i2; StepCount():i1(0),i2(0){} };
    vector<StepCount> S;
    bool cut1(const int pi, const ColorClass &A){
        for(int i = 0; i < (int)A.size(); i++) if (e[pi][A[i]]) return true;
        return false;
    }
    void cut2(const Vertices &A, Vertices &B){
```

```
        for(int i = 0; i < (int)A.size() - 1; i++)
            if(e[A.back().i][A[i].i])
                B.push_back(A[i].i);
    }
    void color_sort(Vertices &R){
        int j = 0, maxno = 1, min_k = max((int)QMAX.size() - (int)Q.size() + 1, 1);
        C[1].clear(), C[2].clear();
        for(int i = 0; i < (int)R.size(); i++) {
            int pi = R[i].i, k = 1;
            while(cut1(pi, C[k])) k++;
            if(k > maxno) maxno = k, C[maxno + 1].clear();
            C[k].push_back(pi);
            if(k < min_k) R[j++].i = pi;
        }
        if(j > 0) R[j - 1].d = 0;
        for(int k = min_k; k <= maxno; k++)
            for(int i = 0; i < (int)C[k].size(); i++)
                R[j].i = C[k][i], R[j++].d = k;
    }
    void expand_dyn(Vertices &R){// diff -> diff with no dyn
        S[level].i1 = S[level].i1 + S[level - 1].i1 - S[level].i2;//diff
        S[level].i2 = S[level - 1].i1;//diff
        while((int)R.size()) {
            if((int)Q.size() + R.back().d > (int)QMAX.size()){
                Q.push_back(R.back().i); Vertices Rp; cut2(R, Rp);
                if((int)Rp.size()){
                    if((float)S[level].i1 / ++pk < Tlimit) degree_sort(Rp);//diff
                    color_sort(Rp);
                    S[level].i1++, level++;//diff
                    expand_dyn(Rp);
                    level--;//diff
                }
                else if((int)Q.size() > (int)QMAX.size()) QMAX = Q;
                Q.pop_back();
            }
            else return;
            R.pop_back();
        }
    }
    void mcqdyn(int* maxclique, int &sz){
        set_degrees(V); sort(V.begin(),V.end(), desc_degree); init_colors(V);
        for(int i = 0; i < (int)V.size() + 1; i++) S[i].i1 = S[i].i2 = 0;
        expand_dyn(V); sz = (int)QMAX.size();
        for(int i = 0; i < (int)QMAX.size(); i++) maxclique[i] = QMAX[i];
    }
    void degree_sort(Vertices &R){
        set_degrees(R); sort(R.begin(), R.end(), desc_degree);
    }
    Maxclique(const BB* conn, const int sz, const float tt = 0.025) \
     : pk(0), level(1), Tlimit(tt){
        for(int i = 0; i < sz; i++) V.push_back(Vertex(i));
        e = conn, C.resize(sz + 1), S.resize(sz + 1);
    }
};
```

3.15 极大团计数

3.16 欧拉回路

```
//从一个奇度点 dfs, sqn 即为回路/路径
//first 存点, second 存边的编号, 正反边编号一致
//清空 cur、used 数组
void getCycle(int u) {
    for(int &i=cur[u]; i < (int)adj[u].size(); ++ i) {
        int id = adj[u][i].second;
        if (used[id]) continue;
        used[id] = true;
        getCycle(adj[u][i].first);
    }
    sqn.push_back(u);
}
```

3.17 朱刘最小树形图

```
struct D_MT {
    struct Edge {
        int u, v, w;
        inline Edge() {}
        inline Edge(int _u, int _v, int _w):u(_u), v(_v), w(_w) {}
    };
    int nn, mm, n, m, vis[maxn], pre[maxn], id[maxn], in[maxn];
    Edge edges[maxn], bac[maxn];
    void init(int _n) {
        n = _n; m = 0;
    }
    void AddEdge(int u, int v, int w) {
        edges[m++] = Edge(u, v, w);
    }
    int work(int root) {
        int ret = 0;
        while(true) {
            for (int i = 0; i < n; i++) in[i] = inf + 1;
            for (int i = 0; i < m; i++) {
                int u = edges[i].u, v = edges[i].v;
                if(edges[i].w < in[v] && u != v){
                    in[v] = edges[i].w;
                    pre[v] = u;
                }
            }
            for (int i = 0; i < n; i++) {
                if(i == root) continue;
                if(in[i] == inf + 1) return inf;
            }
            int cnt = 0;
            for (int i = 0; i < n; i++) id[i] = vis[i] = -1;
            in[root] = 0;
            for (int i = 0; i < n; i++) {
                ret += in[i];
                int v = i;
                while (vis[v] != i&& id[v] == -1 && v != root ){
                    vis[v] = i; v = pre[v];
                }
                if (v != root && id[v] == -1) {
```

```
                    for (int u = pre[v]; u != v; u = pre[u]) id[u] = cnt;
                    id[v] = cnt++;
                }
            }
            if (!cnt) break;
            for (int i = 0; i < n; i++)
                if (id[i] == -1) id[i] = cnt++;
            for (int i = 0; i < m; i++){
                int u = edges[i].u, v = edges[i].v;
                edges[i].v = id[v]; edges[i].u = id[u];
                if(id[u] != id[v]) edges[i].w -= in[v];
            }
            n = cnt; root = id[root];
        }
        return ret;
    }
} MT;
```

Chapter 4

数据结构

4.1 Kd-tree

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

struct P{
    int a[2],val;
    int id;
    int& operator[](int s){return a[s];}
    const int& operator[](int s)const{return a[s];}

    LL dis(const P &b)const{
        LL ans=0;
        for (int i = 0; i < 2; ++i) {
            ans += norm(a[i] - b[i]);
        }
        return ans;
    }
}p[maxn];

bool operator==(const P &a,const P &b){
    for(int i=0;i<DIM;i++)
        if(a[i]!=b[i])
            return false;
    return true;
}
bool byVal(P a,P b){
    return a.val!=b.val ? a.val<b.val : a.id<b.id;
}

struct Rec{
    int mn[DIM],mx[DIM];
    Rec(){}
    Rec(const P &p){
        for(int i=0;i<DIM;i++){
            mn[i]=mx[i]=p[i];
        }
    }
    void add(const P &p){
        for(int i=0;i<DIM;i++){
            mn[i]=min(p[i],mn[i]);
```

```
            mx[i]=max(p[i],mx[i]);
        }
    }

    LL dis(const P &p) {
        LL ans = 0;
        for (int i = 0; i < 2; ++i) {
            //    For minimum distance
            ans += norm(min(max(p[i], mn[i]), mx[i]) - p[i]);
            //    For maximum distance
            //ans += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
        }
        return ans;
    }
};
inline Rec operator+(const Rec &ls,const Rec &rs){
    static Rec rec;
    for(int i=0;i<DIM;i++){
        rec.mn[i]=min(ls.mn[i],rs.mn[i]);
        rec.mx[i]=max(ls.mx[i],rs.mx[i]);
    }
    return rec;
}
struct node{
    Rec rec;
    P sep;
    int sum,siz;
    node *c[2];
    node *rz(){
        sum=sep.val;
        rec=Rec(sep);
        siz=1;
        if(c[0]){
            sum+=c[0]->sum;
            rec=rec+c[0]->rec;
            siz+=c[0]->siz;
        }
        if(c[1]){
            sum+=c[1]->sum;
            rec=rec+c[1]->rec;
            siz+=c[1]->siz;
        }
        return this;
    }
    node(){sum=0;siz=1;c[0]=c[1]=0;}
}*root,*re,pool[maxn],*cur=pool;
node *sta[maxn];
P tmp[maxn];
int D,si;
void init(){
    si=0;
    cur=pool;
    root=0;
}
bool cmp(const P &A,const P &B){

    if(!(A[D]==B[D]))
        return A[D]<B[D];
    return A.id<B.id;
}
```

```
int top;
node *newnode(){
    if(si)return sta[si--];
    return cur++;
}
node* build(P *p,int l,int r,int d){
    int mid=(l+r)>>1;D=d;
    nth_element(p+l,p+mid,p+r+1,cmp);
    node *t=newnode();
    t->sep=p[mid];
    if(l<=mid-1)
        t->c[0]=build(p,l,mid-1,d^1);
    if(mid+1<=r)
        t->c[1]=build(p,mid+1,r,d^1);
    return t->rz();
}
void dfs(node *&t){
    if(t->c[0])dfs(t->c[0]);
    tmp[++top]=t->sep;
    if(t->c[1])dfs(t->c[1]);
    sta[++si]=t;*t=node();
    //delete t;
}
node* rebuild(node *&t){
    if(!t)return 0;
    top=0;dfs(t);
    return build(tmp,1,top,0);
}
#define siz(x) (x?x->siz:0)
void Add(node *&t,const P &p,int d=0){//调用前 re=0; 调用后 rebuild(re);
    D=d;
    if(!t){
        t=newnode();
        t->sep=p;t->rz();
        return;
    }
    if(t->sep==p){
        t->sep.val+=p.val;
        t->rz();
        return;
    }
    if(p[D]<t->sep[D])
        Add(t->c[0],p,d^1);
    else
        Add(t->c[1],p,d^1);

    t->rz();

    if(max(siz(t->c[0]),siz(t->c[1]))>0.7*t->siz)
        re=t;
}
int ans;

bool Out(const Rec &a,const Rec &b){
    for(int i=0;i<DIM;i++){
        int l=max(a.mn[i],b.mn[i]);
        int r=min(a.mx[i],b.mx[i]);
        if(l>r)
            return true;
    }
```

```
    return false;
}
bool In(const Rec &a,const Rec &b){
    for(int i=0;i<DIM;i++){
        if(a.mn[i]<b.mn[i])
            return false;
        if(a.mx[i]>b.mx[i])
            return false;
    }
    return true;
}

bool In(const P &a,const Rec &b){
    for(int i=0;i<DIM;i++){
        if(!(b.mn[i]<=a[i]&&a[i]<=b.mx[i]))
            return false;
    }
    return true;
}

void Q(node *t,const Rec &R){
    if(Out(t->rec,R))return ;
    if(In(t->rec,R)){
        ans+=t->sum;
        return;
    }
    if(In(t->sep,R))
        ans+=t->sep.val;
    if(t->c[0])
        Q(t->c[0],R);
    if(t->c[1])
        Q(t->c[1],R);
}

priority_queue<pair<long long, int> > kNN;
void query(node *t, const P &p, int k, int d = 0) {//用钱清空 kNN
    D=d;
    if (!t || ((int)kNN.size() == k && t->rec.dis(p) > kNN.top().first)) {
        return;
    }
    kNN.push(make_pair(t->sep.dis(p), t->sep.id));
    if ((int)kNN.size() > k) {
        kNN.pop();
    }
    if (cmp(p, t->sep)) {
        query(t->c[0], p, k, d ^ 1);
        query(t->c[1], p, k, d ^ 1);
    } else {
        query(t->c[1], p, k, d ^ 1);
        query(t->c[0], p, k, d ^ 1);
    }
}
```

4.2 LCT

```
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;
```

4.3 树状数组上二分第 k 大

```
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;
}
```

4.4 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){
        if(!t){t=newnode(x);return;}
        if(t->val==x){t->s++;t->siz++;return;}
        insert(t->c[x>t->val],x);
        if(t->key<t->c[x>t->val]->key)
            rot(t,x>t->val);
        else t->rz();
    }
    void del(node *&t,int x){
        if(!t)return;
        if(t->val==x){
            if(t->s>1){t->s--;t->siz--;return;}
            if(!t->c[0]||!t->c[1]){
                if(!t->c[0])t=t->c[1];
                else t=t->c[0];
                return;
            }
            int d=t->c[0]->key<t->c[1]->key;
```

```
            rot(t,d);
            del(t,x);
            return;
        }
        del(t->c[x>t->val],x);
        t->rz();
    }
    int pre(node *t,int x){
        if(!t)return INT_MIN;
        int ans=pre(t->c[x>t->val],x);
        if(t->val<x)ans=max(ans,t->val);
        return ans;
    }
    int nxt(node *t,int x){
        if(!t)return INT_MAX;
        int ans=nxt(t->c[x>=t->val],x);
        if(t->val>x)ans=min(ans,t->val);
        return ans;
    }
    int rank(node *t,int x){
        if(!t)return 0;
        if(t->val==x)return sz(t->c[0]);
        if(t->val<x)return sz(t->c[0])+t->s+rank(t->c[1],x);
        if(t->val>x)return rank(t->c[0],x);
    }
    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;
```

4.5 FHQ-Treap

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn=1e5+5;
int in(){
    int r=0,f=1;char c=getchar();
    while(!isdigit(c))f=c=='-'?-1:f,c=getchar();
    while(isdigit(c))r=r*10+c-'0',c=getchar();
    return r*f;
}
int n,m;
#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 tot;
void C(int l,int r,int v){
    node *p,*q,*x,*y;
    root->split(l-1,p,q);
    q->split(r-l+1,x,y);
    x->make(0,v,0);x->pd();
    root=merge(p,merge(x,y));
}
void A(int l,int r,int d){
    node *p,*q,*x,*y;
    root->split(l-1,p,q);
    q->split(r-l+1,x,y);
```

```
    x->make(1,d,d);x->pd();
    root=merge(p,merge(x,y));
}
void I(int ps,int v){
    node *p,*q;
    root->split(ps-1,p,q);
    node *x=nd+(++tot);
    x->key=rand();x->val=v;x->rz();
    root=merge(merge(p,x),q);
}
LL Q(int l,int r){
    node *p,*q,*x,*y;
    root->split(l-1,p,q);
    q->split(r-l+1,x,y);
    LL ans=x->sum;
    root=merge(p,merge(x,y));
    return ans;
}
int main(){
//  freopen("bzoj3188.in","r",stdin);
    n=in();m=in();
    for(int i=1;i<=n;i++){
        nd[i].val=in();
        nd[i].key=rand();
        nd[i].rz();
        root=merge(root,nd+i);
    }tot=n;
    while(m--){
        int ty=in();
        int l,r;
        if(ty==1){
            l=in();r=in();
            C(l,r,in());
        }else if(ty==2){
            l=in();r=in();
            A(l,r,in());
        }else if(ty==3){
            int ps=in();
            I(ps,in());
        }else if(ty==4){
            l=in();r=in();
            printf("%lld\n",Q(l,r));
        }
    }
    return 0;
}
```

4.6 真-FHQTreap

```
const int mo=1e9+7;
int rnd(){
    static int x=1;
    return x=(x*23333+233);
}
int rnd(int n){
    int x=rnd();
    if(x<0)x=-x;
    return x%n+1;
}
```

```
struct node{
    int siz,key;
    int val;
    LL sum;
    node *c[2];
    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;
    }
    node(){}
    node(int v){
        siz=1;key=rnd();
        val=v;sum=v;
        c[0]=c[1]=0;
    }

}pool[maxn*8],*root,*cur=pool,*old_root,*stop;
node *newnode(int v=0){
    *cur=node(v);
    return cur++;
}
node *old_merge(node *p,node *q){
    if(!p&&!q)return 0;
    node *u=0;
    if(!p||!q)return u=p?p->rz():(q?q->rz():0);
    if(rnd(sz(p)+sz(q))<sz(p)){
        u=p;
        u->c[1]=old_merge(u->c[1],q);
    }else{
        u=q;
        u->c[0]=old_merge(p,u->c[0]);
    }
    return u->rz();
}
node *merge(node *p,node *q){
    if(!p&&!q)return 0;
    node *u=newnode();
    if(!p||!q)return u=p?p->rz():(q?q->rz():0);
    if(rnd(sz(p)+sz(q))<sz(p)){
        *u=*p;
        u->c[1]=merge(u->c[1],q);
    }else{
        *u=*q;
        u->c[0]=merge(p,u->c[0]);
    }
    return u->rz();
}
node *split(node *u,int l,int r){
    if(l>r||!u)return 0;
    node *x=0;
    if(l==1&&r==sz(u)){
        x=newnode();
        *x=*u;
        return x->rz();
    }
    int lsz=sz(u->c[0]);
    if(r<=lsz)
        return split(u->c[0],l,r);
```

```
    if(l>lsz+1)
        return split(u->c[1],l-lsz-1,r-lsz-1);
    x=newnode();
    *x=*u;
    x->c[0]=split(u->c[0],l,lsz);
    x->c[1]=split(u->c[1],1,r-lsz-1);
    return x->rz();
}
```

4.7 带修改莫队上树

```
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);
    }
```

4.8 虚树

```
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;
}
```

Chapter 5

字符串

5.1 Manacher

```
//prime is the origin string(0-base)
//-10,-1,-20 are added to s
//length of s is exactly 2 * l + 3
inline void manacher(char prime[]) {
    int l = strlen(prime), n = 0;
    s[n++] = -10;
    s[n++] = -1;
    for (int i = 0; i < l; ++i) {
        s[n++] = prime[i];
        s[n++] = -1;
    }
    s[n++] = -20; f[0] = 1;
    int mx = 0, id = 0;
    for (int i = 1; i + 1 < n; ++i) {
        f[i] = i > mx ? 1 : min(f[id * 2 - i], mx - i + 1);
        while (s[i + f[i]] == s[i - f[i]]) ++f[i];
        if (i + f[i] - 1 > mx) {
            mx = i + f[i] - 1;
            id = i;
        }
    }
}
```

5.2 指针版回文自动机

```
/*
 * Palindrome Automaton - pointer version
 * PAMPAMPAM? PAMPAMPAM!
 */

namespace PAM {
    struct Node *pool_pointer;
    struct Node {
        Node *fail, *to[26];
        int cnt, len;

        Node() {}
        Node(int len): len(len) {
            memset(to, 0, sizeof(to));
            fail = 0;
            cnt = 0;
        }

```

```
        void *operator new (size_t) {
            return pool_pointer++;
        }
    } pool[100005], *root[2], *last;
    int pam_len, str[100005];

    void init() {
        pool_pointer = pool;
        root[0] = new Node(0);
        root[1] = new Node(-1);
        root[0]->fail = root[1]->fail = root[1];
        str[pam_len = 0] = -1; // different from all characters
        last = root[0];
    }

    void extend(char ch) {
        static Node *p, *np, *q;

        int x = str[++pam_len] = ch - 'a';

        p = last;
        while (str[pam_len - p->len - 1] != x)
            p = p->fail;
        if (!p->to[x]) {
            np = new Node(p->len + 2), q = p->fail;
            while (str[pam_len - q->len - 1] != x) q = q->fail;
            np->fail = q->to[x] ? q->to[x] : root[0];
            p->to[x] = np;
        }
        last = p->to[x];
        ++last->cnt;
    }
}
```

5.3 后缀数组

```
const int maxl=1e5+1e4+5;
const int maxn=maxl*2;
int a[maxn],x[maxn],y[maxn],c[maxn],sa[maxn],rank[maxn],height[maxn];
void calc_sa(int n){
    int m=alphabet,k=1;
    memset(c,0,sizeof(*c)*(m+1));
    for(int i=1;i<=n;i++)c[x[i]=a[i]]++;
    for(int i=1;i<=m;i++)c[i]+=c[i-1];
    for(int i=1;i<=n;i++)sa[c[x[i]]--]=i;
    for(;k<=n;k<<=1){
        int tot=k;
        for(int i=n-k+1;i<=n;i++)y[i-n+k]=i;
        for(int i=1;i<=n;i++)
            if(sa[i]>k)y[++tot]=sa[i]-k;
        memset(c,0,sizeof(*c)*(m+1));
        for(int i=1;i<=n;i++)c[x[i]]++;
        for(int i=1;i<=m;i++)c[i]+=c[i-1];
        for(int i=n;i>=1;i--)sa[c[x[y[i]]]--]=y[i];
        for(int i=1;i<=n;i++)y[i]=x[i];
        tot=1;x[sa[1]]=1;
        for(int i=2;i<=n;i++){
            if(max(sa[i],sa[i-1])+k>n||y[sa[i]]!=y[sa[i-1]]||y[sa[i]+k]!=y[sa[i-1]+k])
                ++tot;
```

```
            x[sa[i]]=tot;
        }
        if(tot==n)break;else m=tot;
    }
}
void calc_height(int n){
    for(int i=1;i<=n;i++)rank[sa[i]]=i;
    for(int i=1;i<=n;i++){
        height[rank[i]]=max(0,height[rank[i-1]]-1);
        if(rank[i]==1)continue;
        int j=sa[rank[i]-1];
        while(max(i,j)+height[rank[i]]<=n&&a[i+height[rank[i]]]==a[j+height[rank[i]]])
            ++height[rank[i]];
    }
}
```

5.4 最小表示法

```
int solve(char *text, int length) {//0-base , 多解答案为起点最小
    int i = 0, j = 1, delta = 0;
    while (i < length && j < length && delta < length) {
        char tokeni = text[(i + delta) % length];
        char tokenj = text[(j + delta) % length];
        if (tokeni == tokenj) {
            delta++;
        } else {
            if (tokeni > tokenj) {
                i += delta + 1;
            } else {
                j += delta + 1;
            }
            if (i == j) {
                j++;
            }
            delta = 0;
        }
    }
    return std::min(i, j);
}
```

Chapter 6

计算几何

6.1 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();
}
int dir(P p,L l){
    int t=sgn((p-l.b)*(l.b-l.a));
    if(t<0)return -1;
    if(t>0)return 1;
    return 0;
}
bool segIntersect(L l1,L l2){//strictly
    if(dir(l2.a,l1)*dir(l2.b,l1)<0&&dir(l1.a,l2)*dir(l1.b,l2)<0)
        return true;
    return false;
}
bool in_tri(P pt,P *p){//change p
    if((p[1]-p[0])*(p[2]-p[0])<0)
        reverse(p,p+3);
    for(int i=0;i<3;i++){
        if(dir(pt,L(p[i],p[(i+1)%3]))==1)
            return false;
    }
    return true;
}

vector<P> convexCut(const vector<P>&ps, L l) { // 用半平面 l 的逆时针方向去切凸多边形
    vector<P> qs;
    int n = ps.size();
    for (int i = 0; i < n; ++i) {
        Point p1 = ps[i], p2 = ps[(i + 1) % n];
```

```
        int d1 = sgn(l.b * (p1 - l.a)), d2 = sign(l.b * (p2 - l.a));
        if (d1 >= 0) qs.push_back(p1);
        if (d1 * d2 < 0) qs.push_back(intersect(L(p1, p2 - p1), l));
    }
    return qs;
}
```

6.2 圆基础

```
struct C{
    P o;
    double r;
    C(){}
    C(P _o,double _r):o(_o),r(_r){}
};
// 求圆与直线的交点
//turn90() P(-y,x)
double fix(double x){return x>=0?x:0;}
bool intersect(C a, L l, P &p1, P &p2) {
    double x = ((l.a - a.o)^ (l.b - l.a)),
        y = (l.b - l.a).len2(),
        d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
    if (sgn(d) < 0) return false;
    d = max(d, 0.0);
    P p = l.a - ((l.b - l.a) * (x / y)), delta = (l.b - l.a) * (sqrt(d) / y);
    p1 = p + delta, p2 = p - delta;
    return true;
}
// 求圆与圆的交点，注意调用前要先判定重圆
bool intersect(C a, C b, P &p1, P &p2) {
    double s1 = (a.o - b.o).len();
    if (sgn(s1 - a.r - b.r) > 0 || sgn(s1 - fabs(a.r - b.r)) < 0) return false;
    double s2 = (a.r * a.r - b.r * b.r) / s1;
    double aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
    P o = (b.o - a.o) * (aa / (aa + bb)) + a.o;
    P delta = (b.o - a.o).norm().turn90() * sqrt(fix(a.r * a.r - aa * aa));
    p1 = o + delta, p2 = o - delta;
    return true;
}
// 求点到圆的切点，按关于点的顺时针方向返回两个点
bool tang(const C &c, const P &p0, P &p1, P &p2) {
    double x = (p0 - c.o).len2(), d = x - c.r * c.r;
    if (d < eps) return false; // 点在圆上认为没有切点
    P p = (p0 - c.o) * (c.r * c.r / x);
    P delta = ((p0 - c.o) * (-c.r * sqrt(d) / x)).turn90();
    p1 = c.o + p + delta;
    p2 = c.o + p - delta;
    return true;
}
// 求圆到圆的外共切线，按关于 c1.o 的顺时针方向返回两条线
vector<L> extan(const C &c1, const C &c2) {
    vector<L> ret;
    if (sgn(c1.r - c2.r) == 0) {
        P dir = c2.o - c1.o;
        dir = (dir * (c1.r / dir.len())).turn90();
        ret.push_back(L(c1.o + dir, c2.o + dir));
        ret.push_back(L(c1.o - dir, c2.o - dir));
    } else {
        P p = (c1.o * -c2.r + c2.o * c1.r) / (c1.r - c2.r);
```

```
        P p1, p2, q1, q2;
        if (tang(c1, p, p1, p2) && tang(c2, p, q1, q2)) {
//            if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
            ret.push_back(L(p1, q1));
            ret.push_back(L(p2, q2));
        }
    }
    return ret;
}
// 求圆到圆的内共切线，按关于 c1.o 的顺时针方向返回两条线
vector<L> intan(const C &c1, const C &c2) {
    vector<L> ret;
    P p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
    P p1, p2, q1, q2;
    if (tang(c1, p, p1, p2) && tang(c2, p, q1, q2)) { // 两圆相切认为没有切线
        ret.push_back(L(p1, q1));
        ret.push_back(L(p2, q2));
    }
    return ret;
}
```

6.3 点在多边形内

```
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(onSeg(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;
}
```

6.4 二维最小覆盖圆

```
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
```

6.5 圆并

```
double ans[2001];
struct Point {
    double x, y;
    Point(){}
    Point(const double & x, const double & y) : x(x), y(y) {}
    void scan() {scanf("%lf%lf", &x, &y);}
    double sqrlen() {return sqr(x) + sqr(y);}
    double len() {return sqrt(sqrlen());}
    Point rev() {return Point(y, -x);}
    void print() {printf("%f %f\n", x, y);}
    Point zoom(const double & d) {double lambda = d / len(); return Point(lambda * x,
      ↪ lambda * y);}
} dvd, a[2001];
Point centre[2001];
double atan2(const Point & x) {
    return atan2(x.y, x.x);
}
Point operator - (const Point & a, const Point & b) {
    return Point(a.x - b.x, a.y - b.y);
}
Point operator + (const Point & a, const Point & b) {
    return Point(a.x + b.x, a.y + b.y);
}
double operator * (const Point & a, const Point & b) {
    return a.x * b.y - a.y * b.x;
}
Point operator * (const double & a, const Point & b) {
    return Point(a * b.x, a * b.y);
}
double operator % (const Point & a, const Point & b) {
    return a.x * b.x + a.y * b.y;
}
struct circle {
    double r; Point o;
    circle() {}
```

```
    void scan() {
        o.scan();
        scanf("%lf", &r);
    }
} cir[2001];
struct arc {
    double theta;
    int delta;
    Point p;
    arc() {};
    arc(const double & theta, const Point & p, int d) : theta(theta), p(p), delta(d) {}
} vec[4444];
int nV;
inline bool operator < (const arc & a, const arc & b) {
    return a.theta + eps < b.theta;
}
int cnt;
inline void psh(const double t1, const Point p1, const double t2, const Point p2) {
    if(t2 + eps < t1)
        cnt++;
    vec[nV++] = arc(t1, p1, 1);
    vec[nV++] = arc(t2, p2, -1);
}
inline double cub(const double & x) {
    return x * x * x;
}
inline void combine(int d, const double & area, const Point & o) {
    if(sign(area) == 0) return;
    centre[d] = 1 / (ans[d] + area) * (ans[d] * centre[d] + area * o);
    ans[d] += area;
}
bool equal(const double & x, const double & y) {
    return x + eps>  y and y + eps > x;
}
bool equal(const Point & a, const Point & b) {
    return equal(a.x, b.x) and equal(a.y, b.y);
}
bool equal(const circle & a, const circle & b) {
    return equal(a.o, b.o) and equal(a.r, b.r);
}
bool f[2001];
int main() {
    //freopen("hdu4895.in", "r", stdin);
    int n, m, index;
    while(EOF != scanf("%d%d%d", &m, &n, &index)) {
        index--;
        for(int i(0); i < m; i++) {
            a[i].scan();
        }
        for(int i(0); i < n; i++) {
            cir[i].scan();//n 个圆
        }
        for(int i(0); i < n; i++) {//这一段在去重圆 能加速 删掉不会错
            f[i] = true;
            for(int j(0); j < n; j++) if(i != j) {
                if(equal(cir[i], cir[j]) and i < j or !equal(cir[i], cir[j]) and cir[i].r <
                  cir[j].r + eps and (cir[i].o - cir[j].o).sqrlen() < sqr(cir[i].r -
                  cir[j].r) + eps) {
                    f[i] = false;
                    break;
```

```
                }
            }
        }
        int n1(0);
        for(int i(0); i < n; i++)
            if(f[i])
                cir[n1++] = cir[i];
        n = n1;//去重圆结束
        fill(ans, ans + n + 1, 0);//ans[i] 表示被圆覆盖至少 i 次的面积
        fill(centre, centre + n + 1, Point(0, 0));//centre[i] 表示上面 ans[i] 部分的重心
        for(int i(0); i < m; i++)
            combine(0, a[i] * a[(i + 1) % m] * 0.5, 1. / 3 * (a[i] + a[(i + 1) % m]));
        for(int i(0); i < n; i++) {
            dvd = cir[i].o - Point(cir[i].r, 0);
            nV = 0;
            vec[nV++] = arc(-pi, dvd, 1);
            cnt = 0;
            for(int j(0); j < n; j++) if(j != i) {
                double d = (cir[j].o - cir[i].o).sqrlen();
                if(d < sqr(cir[j].r - cir[i].r) + eps) {
                    if(cir[i].r + i * eps < cir[j].r + j * eps)
                        psh(-pi, dvd, pi, dvd);
                }else if(d + eps < sqr(cir[j].r + cir[i].r)) {
                    double lambda = 0.5 * (1 + (sqr(cir[i].r) - sqr(cir[j].r)) / d);
                    Point cp(cir[i].o + lambda * (cir[j].o - cir[i].o));
                    Point nor((cir[j].o - cir[i].o).rev().zoom(sqrt(sqr(cir[i].r) - (cp - cir[i].o).sqrlen())));
                    Point frm(cp + nor);
                    Point to(cp - nor);
                    psh(atan2(frm - cir[i].o), frm, atan2(to - cir[i].o), to);
                }
            }
            sort(vec + 1, vec + nV);
            vec[nV++] = arc(pi, dvd, -1);
            for(int j = 0; j + 1 < nV; j++) {
                cnt += vec[j].delta;
                //if(cnt == 1) {//如果只算 ans[1] 和 centre[1], 可以加这个 if 加速.
                    double theta(vec[j + 1].theta - vec[j].theta);
                    double area(sqr(cir[i].r) * theta * 0.5);
                    combine(cnt, area, cir[i].o + 1. / area / 3 * cub(cir[i].r) * Point(sin(vec[j + 1].theta) - sin(vec[j].theta), cos(vec[j].theta) - cos(vec[j + 1].theta)));
                    combine(cnt, -sqr(cir[i].r) * sin(theta) * 0.5, 1. / 3 * (cir[i].o + vec[j].p + vec[j + 1].p));
                    combine(cnt, vec[j].p * vec[j + 1].p * 0.5, 1. / 3 * (vec[j].p + vec[j + 1].p));
                //}
            }
        }//板子部分结束 下面是题目
        combine(0, -ans[1], centre[1]);
        for(int i = 0; i < m; i++) {
            if(i != index)
                (a[index] - Point((a[i] - a[index]) * (centre[0] - a[index]), (a[i] - a[index]) % (centre[0] - a[index])).zoom((a[i] - a[index]).len())).print();
            else
                a[i].print();
        }
    }
    fclose(stdin);
```

```
    return 0;
}
```

6.6 经典阿波罗尼斯圆

```
硬币问题：易知两两相切的圆半径为 r1，r2，r3，求与他们都相切的圆的半径 r4
分母取负号，答案再取绝对值，为外切圆半径
分母取正号为内切圆半径
// $r_4^{\pm} = \frac{r_1r_2r_3}{r_1r_2+r_1r_3+r_2r_3\pm2\sqrt{r_1r_2r_3(r_1+r_2+r_3)}}$
```

6.7 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;
}
```

6.8 求凸包

```
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;
}
```

6.9 凸包游戏

```
/*
   给定凸包，log n 内完成各种询问，具体操作有 :
   1. 判定一个点是否在凸包内
   2. 询问凸包外的点到凸包的两个切点
   3. 询问一个向量关于凸包的切点
   4. 询问一条直线和凸包的交点
   INF 为坐标范围，需要定义点类大于号
   改成实数只需修改 sign 函数，以及把 long long 改为 double 即可
   构造函数时传入凸包要求无重点，面积非空，以及 pair(x,y) 的最小点放在第一个
*/
const int INF = 1000000000;
struct Convex
{
    int n;
    vector<Point> a, upper, lower;
    Convex(vector<Point> _a) : a(_a) {
        n = a.size();
        int ptr = 0;
        for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
        for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
        for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
```

```
        upper.push_back(a[0]);
    }
    int sign(long long x) { return x < 0 ? -1 : x > 0; }
    pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
        int l = 0, r = (int)convex.size() - 2;
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
            else l = mid;
        }
        return max(make_pair(vec.det(convex[r]), r)
            , make_pair(vec.det(convex[0]), 0));
    }
    void update_tangent(const Point &p, int id, int &i0, int &i1) {
        if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
        if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
    }
    void binary_search(int l, int r, Point p, int &i0, int &i1) {
        if (l == r) return;
        update_tangent(p, l % n, i0, i1);
        int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
            if (smid == sl) l = mid;
            else r = mid;
        }
        update_tangent(p, r % n, i0, i1);
    }
    int binary_search(Point u, Point v, int l, int r) {
        int sl = sign((v - u).det(a[l % n] - u));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((v - u).det(a[mid % n] - u));
            if (smid == sl) l = mid;
            else r = mid;
        }
        return l % n;
    }
    // 判定点是否在凸包内，在边界返回 true
    bool contain(Point p) {
        if (p.x < lower[0].x || p.x > lower.back().x) return false;
        int id = lower_bound(lower.begin(), lower.end()
            , Point(p.x, -INF)) - lower.begin();
        if (lower[id].x == p.x) {
            if (lower[id].y > p.y) return false;
        } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
        id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF)
            , greater<Point>()) - upper.begin();
        if (upper[id].x == p.x) {
            if (upper[id].y < p.y) return false;
        } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
        return true;
    }
    // 求点 p 关于凸包的两个切点，如果在凸包外则有序返回编号
    // 共线的多个切点返回任意一个，否则返回 false
    bool get_tangent(Point p, int &i0, int &i1) {
        if (contain(p)) return false;
        i0 = i1 = 0;
        int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
```

```
        binary_search(0, id, p, i0, i1);
        binary_search(id, (int)lower.size(), p, i0, i1);
        id = lower_bound(upper.begin(), upper.end(), p
            , greater<Point>()) - upper.begin();
        binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
        binary_search((int)lower.size() - 1 + id
            , (int)lower.size() - 1 + (int)upper.size(), p, i0, i1);
        return true;
    }
    // 求凸包上和向量 vec 叉积最大的点，返回编号，共线的多个切点返回任意一个
    int get_tangent(Point vec) {
        pair<long long, int> ret = get_tangent(upper, vec);
        ret.second = (ret.second + (int)lower.size() - 1) % n;
        ret = max(ret, get_tangent(lower, vec));
        return ret.second;
    }
    // 求凸包和直线 u,v 的交点，如果无严格相交返回 false.
    //如果有则是和 (i,next(i)) 的交点，两个点无序，交在点上不确定返回前后两条线段其中之一
    bool get_intersection(Point u, Point v, int &i0, int &i1) {
        int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
        if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
            if (p0 > p1) swap(p0, p1);
            i0 = binary_search(u, v, p0, p1);
            i1 = binary_search(u, v, p1, p0 + n);
            return true;
        } else {
            return false;
        }
    }
};
```

6.10 平面最近点

```
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;
}
```

6.11 无敌面积并（多圆多多边形），n^3

```
/*
   n^3 计算多边形圆的面积并
```

```
   注意先去重圆

*/

double form(double x){
    while(x>=2*pi)x-=2*pi;
    while(x<0)x+=2*pi;
    return x;
}
double calcCir(C cir){
    vector<double>ang;
    ang.push_back(0);
    ang.push_back(pi);
    double ans=0;
    for(int i=1;i<=n;i++){
        if(cir==c[i])continue;
        P p1,p2;
        if(intersect(cir,c[i],p1,p2)){
            ang.push_back(form(cir.ang(p1)));
            ang.push_back(form(cir.ang(p2)));
        }
    }

    for(int i=1;i<=m;i++){
        vector<P>tmp;
        tmp=intersect(poly[i],cir);
        for(int j=0;j<tmp.size();j++){
            ang.push_back(form(cir.ang(tmp[j])));
        }
    }
    sort(ang.begin(),ang.end());
    for(int i=0;i<ang.size();i++){
        double t1=ang[i],t2=(i+1==ang.size()?ang[0]+2*pi:ang[i+1]);
        P p=cir.at((t1+t2)/2);
        int ok=1;
        for(int j=1;j<=n;j++){
            if(cir==c[j])continue;
            if(inC(p,c[j],true)){
                ok=0;
                break;
            }
        }
        for(int j=1;j<=m&&ok;j++){
            if(inPoly(p,poly[j],true)){
                ok=0;
                break;
            }
        }
        if(ok){
            double r=cir.r,x0=cir.o.x,y0=cir.o.y;
            ans+=(r*r*(t2-t1)+r*x0*(sin(t2)-sin(t1))-r*y0*(cos(t2)-cos(t1)))/2;

        }
    }
    return ans;
}
P st;
bool bySt(P a,P b){
    return dis(a,st)<dis(b,st);
}
```

```
double calcSeg(L l){
    double ans=0;
    vector<P>pt;
    pt.push_back(l.a);
    pt.push_back(l.b);
    for(int i=1;i<=n;i++){
        P p1,p2;
        if(intersect(c[i],l,p1,p2)){
            if(onSeg(p1,l))
                pt.push_back(p1);
            if(onSeg(p2,l))
                pt.push_back(p2);
        }
    }
    st=l.a;
    sort(pt.begin(),pt.end(),bySt);
    for(int i=0;i+1<pt.size();i++){
        P p1=pt[i],p2=pt[i+1];
        P p=(p1+p2)/2;
        int ok=1;
        for(int j=1;j<=n;j++){
            if(sgn(dis(p,c[j].o),c[j].r)<0){
                ok=0;
                break;
            }
        }
        if(ok){
            double x1=p1.x,y1=p1.y,x2=p2.x,y2=p2.y;
            double res=(x1*y2-x2*y1)/2;
            ans+=res;
        }
    }
    return ans;
}
```

6.12 Farmland

```
const int N = 11111, M = 111111 * 4;

struct eglist {
    int other[M], succ[M], last[M], sum;
    void clear() {
        memset(last, -1, sizeof(last));
        sum = 0;
    }
    void addEdge(int a, int b) {
        other[sum] = b, succ[sum] = last[a], last[a] = sum++;
        other[sum] = a, succ[sum] = last[b], last[b] = sum++;
    }
}e;

int n, m;
struct point {
    int x, y;
    point(int x, int y) : x(x), y(y) {}
    point() {}
    friend point operator -(point a, point b) {
        return point(a.x - b.x, a.y - b.y);
    }
```

```
    double arg() {
        return atan2(y, x);
    }
}points[N];

vector<pair<int, double> > vecs;
vector<int> ee[M];
vector<pair<double, pair<int, int> > > edges;
double length[M];
int tot, father[M], next[M], visit[M];

int find(int x) {
    return father[x] == x ? x : father[x] = find(father[x]);
}

long long det(point a, point b) {
    return 1LL * a.x * b.y - 1LL * b.x * a.y;
}

double dist(point a, point b) {
    return sqrt(1.0 * (a.x - b.x) * (a.x - b.x) + 1.0 * (a.y - b.y) * (a.y - b.y));
}

int main() {
    scanf("%d %d", &n, &m);
    e.clear();
    for(int i = 1; i <= n; i++) {
        scanf("%d %d", &points[i].x, &points[i].y);
    }
    for(int i = 1; i <= m; i++) {
        int a, b;
        scanf("%d %d", &a, &b);
        e.addEdge(a, b);
    }
    for(int x = 1; x <= n; x++) {
        vector<pair<double, int> > pairs;
        for(int i = e.last[x]; ~i; i = e.succ[i]) {
            int y = e.other[i];
            pairs.push_back(make_pair((points[y] - points[x]).arg(), i));
        }
        sort(pairs.begin(), pairs.end());
        for(int i = 0; i < (int)pairs.size(); i++) {
            next[pairs[(i + 1) % (int)pairs.size()].second ^ 1] = pairs[i].second;
        }
    }
    memset(visit, 0, sizeof(visit));
    tot = 0;
    for(int start = 0; start < e.sum; start++) {
        if (visit[start])
            continue;
        long long total = 0;
        int now = start;
        vecs.clear();
        while(!visit[now]) {
            visit[now] = 1;
            total += det(points[e.other[now ^ 1]], points[e.other[now]]);
            vecs.push_back(make_pair(now / 2, dist(points[e.other[now ^ 1]],
              ↪points[e.other[now]])));
            now = next[now];
        }
```

```
        if (now == start && total > 0) {
            ++tot;
            for(int i = 0; i < (int)vecs.size(); i++) {
                ee[vecs[i].first].push_back(tot);
            }
        }
    }

    for(int i = 0; i < e.sum / 2; i++) {
        int a = 0, b = 0;
        if (ee[i].size() == 0)
            continue;
        else if (ee[i].size() == 1) {
            a = ee[i][0];
        } else if (ee[i].size() == 2) {
            a = ee[i][0], b = ee[i][1];
        }
        edges.push_back(make_pair(dist(points[e.other[i * 2]], points[e.other[i * 2 + 1]]),
          ↪make_pair(a, b)));
    }
    sort(edges.begin(), edges.end());
    for(int i = 0; i <= tot; i++)
        father[i] = i;
    double ans = 0;
    for(int i = 0; i < (int)edges.size(); i++) {
        int a = edges[i].second.first, b = edges[i].second.second;
        double v = edges[i].first;
        if (find(a) != find(b)) {
            ans += v;
            father[father[a]] = father[b];
        }
    }
    printf("%.5f\n", ans);
}
```

6.13 三维基础

```
struct P {
    double x, y, z;
    P(){}
    P(double _x,double _y,double _z):x(_x),y(_y),z(_z){}
    double len2(){
        return (x*x+y*y+z*z);
    }
    double len(){
        return sqrt(x*x+y*y+z*z);
    }
};
bool operator==(P a,P b){
    return sgn(a.x-b.x)==0 && sgn(a.y-b.y)==0 && sgn(a.z-b.z)==0 ;
}
bool operator<(P a,P b){
    return sgn(a.x-b.x) ? a.x<b.x :(sgn(a.y-b.y)?a.y<b.y :a.z<b.z);
}
P operator+(P a,P b){
    return P(a.x+b.x,a.y+b.y,a.z+b.z);
}
P operator-(P a,P b){
    return P(a.x-b.x,a.y-b.y,a.z-b.z);
```

```
}
P operator*(P a,double b){
    return P(a.x*b,a.y*b,a.z*b);
}
P operator/(P a,double b){
    return P(a.x/b,a.y/b,a.z/b);
}
P operator*(const P &a, const P &b) {
    return P(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
}
double operator^(const P &a, const P &b) {
    return a.x*b.x+a.y*b.y+a.z*b.z;
}

double dis(P a,P b){return (b-a).len();}
double dis2(P a,P b){return (b-a).len2();}


// 平面法向量 : 平面上两个向量叉积
// 点共平面 : 平面上一点与之的向量点积法向量为 0
// 点在线段 ( 直线 ) 上 : 共线且两边点积非正
// 点在三角形内 ( 不包含边界，需再判断是与某条边共线 )
bool in_tri(const P &a, const P &b, const P &c, const P &p) {
    return sgn(((a - b)*(a - c)).len() - ((p - a)*(p - b)).len() - ((p - b)*(p - c)).len() - ((p - c)*(p - a)).len()) == 0;
}
// 共平面的两点是否在这平面上一条直线的同侧
bool sameSide(const P &a, const P &b, const P &p0, const P &p1) {
    return sgn(((a - b)*(p0 - b)) ^ ((a - b)*(p1 - b))) > 0;
}
// 两点在平面同侧 : 点积法向量符号相同
// 两直线平行 / 垂直 : 同二维
// 平面平行 / 垂直 : 判断法向量
// 线面垂直 : 法向量和直线平行
// 判断空间线段是否相交 : 四点共面两线段不平行相互在异侧
// 线段和三角形是否相交 : 线段在三角形平面不同侧 三角形任意两点在线段和第三点组成的平面的不同侧
// 求空间直线交点
P intersect(const P &a0, const P &b0, const P &a1, const P &b1) {

double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) / ((a0.x - b0.x) * (a1.y - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));

    //double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) / ((a0.x - b0.x) * (a1.y - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));
    return a0 + (b0 - a0) * t;
}
// 求平面和直线的交点
P intersect(const P &a, const P &b, const P &c, const P &l0, const P &l1) {

    P p = (b-a)*(c-a); // 平面法向量
    double t = (p^(a-l0)) / (p^(l1-l0));
    return l0 + (l1 - l0) * t;
//    P p = pVec(a, b, c); // 平面法向量
//    double t = (p.x * (a.x - l0.x) + p.y * (a.y - l0.y) + p.z * (a.z - l0.z)) / (p.x * (l1.x - l0.x) + p.y * (l1.y - l0.y) + p.z * (l1.z - l0.z));
//    return l0 + (l1 - l0) * t;
}
// 求平面交线 : 取不平行的一条直线的一个交点，以及法向量叉积得到直线方向
// 点到直线距离 : 叉积得到三角形的面积除以底边
// 点到平面距离 : 点积法向量
```

```
// 直线间距离 : 平行时随便取一点求距离，否则叉积方向向量得到方向点积计算长度
// 直线夹角 : 点积 平面夹角 : 法向量点积
```

6.14 三维凸包

```
int mark[1005][1005],n, cnt;;
double mix(const P &a, const P &b, const P &c) {
    return a^(b*c);
}
double area(int a, int b, int c) {
    return ((info[b] - info[a])*(info[c] - info[a])).len();
}
double volume(int a, int b, int c, int d) {
    return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);
}
struct Face {
    int a, b, c; Face() {}
    Face(int a, int b, int c): a(a), b(b), c(c) {}
    int &operator [](int k) {
        if (k == 0) return a; if (k == 1) return b; return c;
    }
};
vector <Face> face;
inline void insert(int a, int b, int c) {
    face.push_back(Face(a, b, c));
}
void add(int v) {
    vector <Face> tmp; int a, b, c; cnt++;
    for (int i = 0; i < SIZE(face); i++) {
        a = face[i][0]; b = face[i][1]; c = face[i][2];
        if (sgn(volume(v, a, b, c)) < 0)
        mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
        else tmp.push_back(face[i]);
    } face = tmp;
    for (int i = 0; i < SIZE(tmp); i++) {
        a = face[i][0]; b = face[i][1]; c = face[i][2];
        if (mark[a][b] == cnt) insert(b, a, v);
        if (mark[b][c] == cnt) insert(c, b, v);
        if (mark[c][a] == cnt) insert(a, c, v);
    }
}
int Find() {
    for (int i = 2; i < n; i++) {
        P ndir = (info[0] - info[i])*(info[1] - info[i]);
        if (ndir == P()) continue; swap(info[i], info[2]);
        for (int j = i + 1; j < n; j++) if (sgn(volume(0, 1, 2, j)) != 0) {
            swap(info[j], info[3]); insert(0, 1, 2); insert(0, 2, 1); return 1;
        }
    }
    return 0;
}

// 求重心
double calcDist(const P &p, int a, int b, int c) {
    return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c));
}
//compute the minimal distance of center of any faces
P findCenter() { //compute center of mass
    double totalWeight = 0;
```

```
    P center(.0, .0, .0);
    P first = info[face[0][0]];
    for (int i = 0; i < SIZE(face); ++i) {
        P p = (info[face[i][0]]+info[face[i][1]]+info[face[i][2]]+first)*.25;
        double weight = mix(info[face[i][0]] - first, info[face[i][1]] - first,
          ↪info[face[i][2]] - first);
        totalWeight += weight; center = center + p * weight;
    }
    center = center / totalWeight;
    return center;
}
double minDis(P p) {
    double res = 1e100; //compute distance
    for (int i = 0; i < SIZE(face); ++i)
        res = min(res, calcDist(p, face[i][0], face[i][1], face[i][2]));
    return res;
}

void findConvex(P *info,int n) {
    sort(info, info + n); n = unique(info, info + n) - info;
    face.clear(); random_shuffle(info, info + n);
    if(!Find())return abort();
    memset(mark, 0, sizeof(mark)); cnt = 0;
    for (int i = 3; i < n; i++) add(i);
}
// 三维绕轴旋转，大拇指指向 axis 向量方向，四指弯曲方向转 w 弧度
P rotate(const P& s, const P& axis, double w) {
    double x = axis.x, y = axis.y, z = axis.z;
    double s1 = x * x + y * y + z * z, ss1 = msqrt(s1),
       cosw = cos(w), sinw = sin(w);
    double a[4][4];
    memset(a, 0, sizeof a);
    a[3][3] = 1;
    a[0][0] = ((y * y + z * z) * cosw + x * x) / s1;
    a[0][1] = x * y * (1 - cosw) / s1 + z * sinw / ss1;
    a[0][2] = x * z * (1 - cosw) / s1 - y * sinw / ss1;
    a[1][0] = x * y * (1 - cosw) / s1 - z * sinw / ss1;
    a[1][1] = ((x * x + z * z) * cosw + y * y) / s1;
    a[1][2] = y * z * (1 - cosw) / s1 + x * sinw / ss1;
    a[2][0] = x * z * (1 - cosw) / s1 + y * sinw / ss1;
    a[2][1] = y * z * (1 - cosw) / s1 - x * sinw / ss1;
    a[2][2] = ((x * x + y * y) * cos(w) + z * z) / s1;
    double ans[4] = {0, 0, 0, 0}, c[4] = {s.x, s.y, s.z, 1};
    for (int i = 0; i < 4; ++ i)
        for (int j = 0; j < 4; ++ j)
            ans[i] += a[j][i] * c[j];
    return P(ans[0], ans[1], ans[2]);
}
```

6.15 三角剖分与 V 图

```
/*
Delaunay Triangulation 随机增量算法 :
节点数至少为点数的 6 倍，空间消耗较大注意计算内存使用
建图的过程在 build 中，注意初始化内存池和初始三角形的坐标范围 (Triangulation::LOTS)
Triangulation::find 返回包含某点的三角形
Triangulation::add_point 将某点加入三角剖分
某个 Triangle 在三角剖分中当且仅当它的 has_children 为 0
```

```
如果要找到三角形 u 的邻域，则枚举它的所有 u.edge[i].tri，该条边的两个点为 u.p[(i+1)%3], u.p[(i+2)%3]
通过三角剖分构造 V 图：连接相邻三角形外接圆圆心即可
复杂度好像是 O(nlogn)
*/
const int N = 100000 + 5, MAX_TRIS = N * 6;
const double eps = 1e-6, PI = acos(-1.0);
struct P {
    double x,y; P():x(0),y(0){}
    P(double x, double y):x(x),y(y){}
    bool operator ==(P const& that)const {return x==that.x&&y==that.y;}
};
inline double sqr(double x) { return x*x; }
double dist_sqr(P const& a, P const& b){return sqr(a.x-b.x)+sqr(a.y-b.y);}
bool in_circumcircle(P const& p1, P const& p2, P const& p3, P const& p4) {//p4 in C(p1,p2,p3)
    double u11 = p1.x - p4.x, u21 = p2.x - p4.x, u31 = p3.x - p4.x;
    double u12 = p1.y - p4.y, u22 = p2.y - p4.y, u32 = p3.y - p4.y;
    double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
    double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
    double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
    double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33 + u11*u22*u33;
    return det > eps;
}
double side(P const& a, P const& b, P const& p) { return (b.x-a.x)*(p.y-a.y) - (b.y-a.y)*(p.x-a.x);}
typedef int SideRef; struct Triangle; typedef Triangle* TriangleRef;
struct Edge {
    TriangleRef tri; SideRef side; Edge() : tri(0), side(0) {}
    Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {}
};
struct Triangle {
    P p[3]; Edge edge[3]; TriangleRef children[3]; Triangle() {}
    Triangle(P const& p0, P const& p1, P const& p2) {
        p[0] = p0; p[1] = p1; p[2] = p2;
        children[0] = children[1] = children[2] = 0;
    }
    bool has_children() const { return children[0] != 0; }
    int num_children() const {
        return children[0] == 0 ? 0
            : children[1] == 0 ? 1
            : children[2] == 0 ? 2 : 3;
    }
    bool contains(P const& q) const {
        double a=side(p[0],p[1],q), b=side(p[1],p[2],q), c=side(p[2],p[0],q);
        return a >= -eps && b >= -eps && c >= -eps;
    }
} triange_pool[MAX_TRIS], *tot_triangles;
void set_edge(Edge a, Edge b) {
    if (a.tri) a.tri->edge[a.side] = b;
    if (b.tri) b.tri->edge[b.side] = a;
}
class Triangulation {
    public:
        Triangulation() {
            const double LOTS = 1e6;//初始为极大三角形
            the_root = new(tot_triangles++) Triangle(P(-LOTS,-LOTS),P(+LOTS,-LOTS),P(0,+LOTS));
        }
```

```
        TriangleRef find(P p) const { return find(the_root,p); }
        void add_point(P const& p) { add_point(find(the_root,p),p); }
    private:
        TriangleRef the_root;
        static TriangleRef find(TriangleRef root, P const& p) {
            for( ; ; ) {
                if (!root->has_children()) return root;
                else for (int i = 0; i < 3 && root->children[i] ; ++i)
                        if (root->children[i]->contains(p))
                            {root = root->children[i]; break;}
            }
        }
        void add_point(TriangleRef root, P const& p) {
            TriangleRef tab,tbc,tca;
            tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
            tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
            tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
            set_edge(Edge(tab,0),Edge(tbc,1)); set_edge(Edge(tbc,0),Edge(tca,1));
            set_edge(Edge(tca,0),Edge(tab,1)); set_edge(Edge(tab,2),root->edge[2]);
            set_edge(Edge(tbc,2),root->edge[0]); set_edge(Edge(tca,2),root->edge[1]);
            root->children[0]=tab; root->children[1]=tbc; root->children[2]=tca;
            flip(tab,2); flip(tbc,2); flip(tca,2);
        }
        void flip(TriangleRef tri, SideRef pi) {
            TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
            if(!trj || !in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
            TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj], tri->p[pi]);
            TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi], trj->p[pj]);
            set_edge(Edge(trk,0), Edge(trl,0));
            set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
            set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
            tri->children[0]=trk; tri->children[1]=trl; tri->children[2]=0;
            trj->children[0]=trk; trj->children[1]=trl; trj->children[2]=0;
            flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2);
        }
};
int n; P ps[N];
void build(){
    tot_triangles = triange_pool; cin >> n;
    for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
    random_shuffle(ps, ps + n); Triangulation tri;
    for(int i = 0; i < n; ++ i) tri.add_point(ps[i]);
}
```

6.16 三维最小覆盖球

```
bool equal(const double & x, const double & y) {
    return x + eps > y and y + eps > x;
}
double operator % (const Point & a, const Point & b) {
    return a.x * b.x + a.y * b.y + a.z * b.z;
}
Point operator * (const Point & a, const Point & b) {
    return Point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
}
```

```
struct Circle {
    double r; Point o;
};
struct Plane {
    Point nor;
    double m;
    Plane(const Point & nor, const Point & a) : nor(nor){
        m = nor % a;
    }
};
Point intersect(const Plane & a, const Plane & b, const Plane & c) {
    Point c1(a.nor.x, b.nor.x, c.nor.x), c2(a.nor.y, b.nor.y, c.nor.y), c3(a.nor.z,
      ↪b.nor.z, c.nor.z), c4(a.m, b.m, c.m);
    return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
}
bool in(const Point & a, const Circle & b) {
    return sign((a - b.o).len() - b.r) <= 0;
}
bool operator < (const Point & a, const Point & b) {
    if(!equal(a.x, b.x)) {
        return a.x < b.x;
    }
    if(!equal(a.y, b.y)) {
        return a.y < b.y;
    }
    if(!equal(a.z, b.z)) {
        return a.z < b.z;
    }
    return false;
}
bool operator == (const Point & a, const Point & b) {
    return equal(a.x, b.x) and equal(a.y, b.y) and equal(a.z, b.z);
}
vector<Point> vec;
Circle calc() {
    if(vec.empty()) {
        return Circle(Point(0, 0, 0), 0);
    }else if(1 == (int)vec.size()) {
        return Circle(vec[0], 0);
    }else if(2 == (int)vec.size()) {
        return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
    }else if(3 == (int)vec.size()) {
        double r((vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] -
          ↪vec[0]).len() / 2 / fabs(((vec[0] - vec[2]) * (vec[1] - vec[2])).len()));
        return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
                           Plane(vec[2] - vec[1], 0.5 * (vec[2] + vec[1])),
                    Plane((vec[1] - vec[0]) * (vec[2] - vec[0]), vec[0])), r);
    }else {
        Point o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
                  Plane(vec[2] - vec[0], 0.5 * (vec[2] + vec[0])),
                  Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0]))));
        return Circle(o, (o - vec[0]).len());
    }
}
Circle miniBall(int n) {
    Circle res(calc());
    for(int i(0); i < n; i++) {
        if(!in(a[i], res)) {
            vec.push_back(a[i]);
            res = miniBall(i);
```

```
            vec.pop_back();
            if(i) {
                Point tmp(a[i]);
                memmove(a + 1, a, sizeof(Point) * i);
                a[0] = tmp;
            }
        }
    }
    return res;
}
int main() {
    int n;
    sort(a, a + n);
    n = unique(a, a + n) - a;
    vec.clear();
    printf("%.10f\n", miniBall(n).r);
}
```

6.17 空间四点外接球

```
// 注意，无法处理小于四点的退化情况
pair<P,double> ball(P outer[4]) {
    P res; double radius;
    P q[3]; double m[3][3], sol[3], L[3], det;
    int i,j; res.x = res.y = res.z = radius = 0;
    for (i=0; i<3; ++i) q[i]=outer[i+1]-outer[0], sol[i]=(q[i]^q[i]);
    for (i=0;i<3;++i) for(j=0;j<3;++j) m[i][j]=(q[i]^q[j])*2;
    det= m[0][0]*m[1][1]*m[2][2]
    + m[0][1]*m[1][2]*m[2][0]
    + m[0][2]*m[2][1]*m[1][0]
    - m[0][2]*m[1][1]*m[2][0]
    - m[0][1]*m[1][0]*m[2][2]
    - m[0][0]*m[1][2]*m[2][1];
    if ( fabs(det)<1e-10) return;
    for (j=0; j<3; ++j) {
        for (i=0; i<3; ++i) m[i][j]=sol[i];
        L[j]=( m[0][0]*m[1][1]*m[2][2]
        + m[0][1]*m[1][2]*m[2][0]
        + m[0][2]*m[2][1]*m[1][0]
        - m[0][2]*m[1][1]*m[2][0]
        - m[0][1]*m[1][0]*m[2][2]
        - m[0][0]*m[1][2]*m[2][1]
        ) / det;
        for (i=0; i<3; ++i) m[i][j]=(q[i]^q[j])*2;
    }
    res=outer[0];
    for (i=0; i<3; ++i ) res = res + q[i] * L[i];
    radius=dis(res, outer[0]);
    return make_pair(res,radius);
}
```

Chapter 7

技巧

7.1 无敌的读入优化

```
// getchar() 读入优化 << 关同步 cin << 此优化
// 用 isdigit() 会小幅变慢
// 返回 false 表示读到文件尾
namespace Reader {
    const int L = (1 << 15) + 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; bool neg = 0;
        for (; getchar(ch) && (ch < '0' || ch > '9'); ) neg ^= ch == '-';
        if (ch == EOF) return false;
        x = ch - '0';
        for (; getchar(ch), ch >= '0' && ch <= '9'; )
            x = x * 10 + ch - '0';
        if (neg) x = -x;
        return true;
    }
}
```

7.2 真正释放 STL 内存

```
template <typename T>
__inline void clear(T& container) {
    container.clear(); // 或者删除了一堆元素
    T(container).swap(container);
}
```

7.3 梅森旋转算法

```
#include <random>

int main() {
```

```
    std::mt19937 g(seed);  // std::mt19937_64
    std::cout << g() << std::endl;
}
```

7.4 蔡勒公式

```
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;
}
```

7.5 开栈

```
register char *_sp __asm__("rsp");
int main() {
    const int size = 400 << 20;//400MB
    static char *sys, *mine(new char[size] + size - 4096);
    sys = _sp; _sp = mine; _main(); _sp = sys;
}
```

7.6 Size 为 k 的子集

```
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;
    }
}
```

7.7 长方体表面两点最短距离

7.8 经纬度求球面最短距离

7.9 32-bit/64-bit 随机素数

32-bit	64-bit
73550053	1249292846855685773
148898719	1701750434419805569
189560747	3605499878424114901
459874703	5648316673387803781
1202316001	6125342570814357977
1431183547	6215155308775851301
1438011109	6294606778040623451
1538762023	6347330550446020547
1557944263	7429632924303725207
1981315913	8524720079480389849

7.10 NTT 素数及其原根

Prime	Primitive root
1053818881	7
1051721729	6
1045430273	3
1012924417	5
1007681537	3
1000000000622593	5

7.11 Formulas

7.11.1 Arithmetic Function

$$\sigma_k(n) = \sum_{d|n} d^k = \prod_{i=1}^{\omega(n)} \frac{p_i^{(a_i+1)k} - 1}{p_i^k - 1}$$

$$J_k(n) = n^k \prod_{p|n} (1 - \frac{1}{p^k})$$

$J_k(n)$ is the number of k-tuples of positive integers all less than or equal to n that form a coprime $(k+1)$-tuple together with n.

$$\sum_{\delta|n} J_k(\delta) = n^k$$

$$\sum_{\delta|n} \delta^s J_r(\delta) J_s(\frac{n}{\delta}) = J_{r+s}(n)$$

$$\sum_{\delta|n} \varphi(\delta) d(\frac{n}{\delta}) = \sigma(n), \ \sum_{\delta|n} |\mu(\delta)| = 2^{\omega(n)}$$

$$\sum_{\delta|n} 2^{\omega(\delta)} = d(n^2), \ \sum_{\delta|n} d(\delta^2) = d^2(n)$$

$$\sum_{\delta|n} d(\frac{n}{\delta}) 2^{\omega(\delta)} = d^2(n), \ \sum_{\delta|n} \frac{\mu(\delta)}{\delta} = \frac{\varphi(n)}{n}$$

$$\sum_{\delta|n} \frac{\mu(\delta)}{\varphi(\delta)} = d(n), \ \sum_{\delta|n} \frac{\mu^2(\delta)}{\varphi(\delta)} = \frac{n}{\varphi(n)}$$

$$n|\varphi(a^n - 1)$$

$$\sum_{\substack{1 \le k \le n \\ \gcd(k,n)=1}} f(\gcd(k-1, n)) = \varphi(n) \sum_{d|n} \frac{(\mu * f)(d)}{\varphi(d)}$$

$$\varphi(\operatorname{lcm}(m,n)) \varphi(\gcd(m,n)) = \varphi(m) \varphi(n)$$

$$\sum_{\delta|n} d^3(\delta) = (\sum_{\delta|n} d(\delta))^2$$

$$d(uv) = \sum_{\delta|\ \gcd(u,v)} \mu(\delta) d(\frac{u}{\delta}) d(\frac{v}{\delta})$$

$$\sigma_k(u)\sigma_k(v) = \sum_{\delta|\ \gcd(u,v)} \delta^k \sigma_k(\frac{uv}{\delta^2})$$

$$\mu(n) = \sum_{k=1}^{n} [\gcd(k,n)=1] \cos 2\pi \frac{k}{n}$$

$$\varphi(n) = \sum_{k=1}^{n} [\gcd(k,n)=1] = \sum_{k=1}^{n} \gcd(k,n) \cos 2\pi \frac{k}{n}$$

$$\begin{cases} S(n) = \sum_{k=1}^{n} (f*g)(k) \\ \sum_{k=1}^{n} S(\lfloor \frac{n}{k} \rfloor) = \sum_{i=1}^{n} f(i) \sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} (g*1)(j) \end{cases}$$

$$\begin{cases} S(n) = \sum_{k=1}^{n} (f \cdot g)(k), g \text{ completely multiplicative} \\ \sum_{k=1}^{n} S(\lfloor \frac{n}{k} \rfloor) g(k) = \sum_{k=1}^{n} (f*1)(k) g(k) \end{cases}$$

7.11.2 Binomial Coefficients

$$\binom{n}{k} = (-1)^k \binom{k-n-1}{k}$$

$$\sum_{k \le n} \binom{r+k}{k} = \binom{r+n+1}{n}$$

$$\sum_{k=0}^{n} \binom{k}{m} = \binom{n+1}{m+1}$$

$$\sqrt{1+z} = 1 + \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k \times 2^{2k-1}} \binom{2k-2}{k-1} z^k$$

$$\sum_{k=0}^{r} \binom{r-k}{m} \binom{s+k}{n} = \binom{r+s+1}{m+n+1}$$

$$C_{n,m} = \binom{n+m}{m} - \binom{n+m}{m-1}, n \ge m$$

$$\binom{n}{k} \equiv [n\&k = k] \pmod 2$$

7.11.3 Fibonacci Numbers

$$F(z) = \frac{z}{1-z-z^2}$$

$$f_n = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}}, \phi = \frac{1+\sqrt{5}}{2}, \hat{\phi} = \frac{1-\sqrt{5}}{2}$$

$$\sum_{k=1}^{n} f_k = f_{n+2} - 1$$

$$\sum_{k=1}^{n} f_k^2 = f_n f_{n+1}$$

$$\sum_{k=0}^{n} f_k f_{n-k} = \frac{1}{5}(n-1) f_n + \frac{2}{5} n f_{n-1}$$

$$f_n^2 + (-1)^n = f_{n+1} f_{n-1}$$

$$f_{n+k} = f_n f_{k+1} + f_{n-1} f_k$$

$$f_{2n+1} = f_n^2 + f_{n+1}^2$$
$$(-1)^k f_{n-k} = f_n f_{k-1} - f_{n-1} f_k$$
$$\text{Modulo } f_n, f_{mn+r} \equiv \begin{cases} f_r, & m \bmod 4 = 0; \\ (-1)^{r+1} f_{n-r}, & m \bmod 4 = 1; \\ (-1)^n f_r, & m \bmod 4 = 2; \\ (-1)^{r+1+n} f_{n-r}, & m \bmod 4 = 3. \end{cases}$$

7.11.4 Stirling Cycle Numbers

$$\begin{bmatrix} n+1 \\ k \end{bmatrix} = n \begin{bmatrix} n \\ k \end{bmatrix} + \begin{bmatrix} n \\ k-1 \end{bmatrix}, \quad \begin{bmatrix} n+1 \\ 2 \end{bmatrix} = n! H_n$$
$$x^{\underline{n}} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} (-1)^{n-k} x^k, \quad x^{\overline{n}} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} x^k$$

7.11.5 Stirling Subset Numbers

$$\begin{Bmatrix} n+1 \\ k \end{Bmatrix} = k \begin{Bmatrix} n \\ k \end{Bmatrix} + \begin{Bmatrix} n \\ k-1 \end{Bmatrix}$$
$$x^n = \sum_k \begin{Bmatrix} n \\ k \end{Bmatrix} x^{\underline{k}} = \sum_k \begin{Bmatrix} n \\ k \end{Bmatrix} (-1)^{n-k} x^{\overline{k}}$$
$$m! \begin{Bmatrix} n \\ m \end{Bmatrix} = \sum_k \binom{m}{k} k^n (-1)^{m-k}$$

7.11.6 Eulerian Numbers

$$\left\langle {n \atop k} \right\rangle = (k+1) \left\langle {n-1 \atop k} \right\rangle + (n-k) \left\langle {n-1 \atop k-1} \right\rangle$$
$$x^n = \sum_k \left\langle {n \atop k} \right\rangle \binom{x+k}{n}$$
$$\left\langle {n \atop m} \right\rangle = \sum_{k=0}^{m} \binom{n+1}{k} (m+1-k)^n (-1)^k$$

7.11.7 Harmonic Numbers

$$\sum_{k=1}^{n} H_k = (n+1) H_n - n$$
$$\sum_{k=1}^{n} k H_k = \frac{n(n+1)}{2} H_n - \frac{n(n-1)}{4}$$
$$\sum_{k=1}^{n} \binom{k}{m} H_k = \binom{n+1}{m+1} (H_{n+1} - \frac{1}{m+1})$$

7.11.8 Pentagonal Number Theorem

$$\prod_{n=1}^{\infty} (1 - x^n) = \sum_{n=-\infty}^{\infty} (-1)^k x^{k(3k-1)/2}$$
$$p(n) = p(n-1) + p(n-2) - p(n-5) - p(n-7) + \cdots$$
$$f(n,k) = p(n) - p(n-k) - p(n-2k) + p(n-5k) + p(n-7k) - \cdots$$

7.11.9 Bell Numbers

$$B_{n+1}=\sum_{k=0}^{n}\binom{n}{k}B_k$$

$$B_{p^m+n}\equiv mB_n+B_{n+1}\pmod p$$

7.11.10 Bernoulli Numbers

$$B_n=1-\sum_{k=0}^{n-1}\binom{n}{k}\frac{B_k}{n-k+1}$$

$$G(x)=\sum_{k=0}^{\infty}\frac{B_k}{k!}x^k=\frac{1}{\sum_{k=0}^{\infty}\frac{x^k}{(k+1)!}}$$

$$S_m(n)=\frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k}B_k n^{m-k+1}$$

7.11.11 Tetrahedron Volume

$$V=\frac{\sqrt{4u^2v^2w^2-\sum_{cyc}u^2(v^2+w^2-U^2)^2+\prod_{cyc}(v^2+w^2-U^2)}}{12}$$

7.11.12 BEST Thoerem

Counting the number of different Eulerian circuits in directed graphs.

$$\mathrm{ec}(G)=t_w(G)\prod_{v\in V}(\deg(v)-1)!$$

When calculating $t_w(G)$ for directed multigraphs, the entry $q_{i,j}$ for distinct i and j equals $-m$, where m is the number of edges from i to j, and the entry $q_{i,i}$ equals the indegree of i minus the number of loops at i. It is a property of Eulerian graphs that $\mathrm{tv}(G)=\mathrm{tw}(G)$ for every two vertices v and w in a connected Eulerian graph G.

7.11.13 重心

半径为 r，圆心角为 θ 的扇形重心与圆心的距离为 $\frac{4r\sin(\theta/2)}{3\theta}$
半径为 r，圆心角为 θ 的圆弧重心与圆心的距离为 $\frac{4r\sin^3(\theta/2)}{3(\theta-\sin(\theta))}$

7.11.14 Others

$$S_j=\sum_{k=1}^{n}x_k^j$$

$$h_m=\sum_{1\le j_1<\cdots<j_m\le n}x_{j_1}\cdots x_{j_m}$$

$$H_m=\sum_{1\le j_1\le\cdots\le j_m\le n}x_{j_1}\cdots x_{j_m}$$

$$h_n=\frac{1}{n}\sum_{k=1}^{n}(-1)^{k+1}S_k h_{n-k}$$

$$H_n=\frac{1}{n}\sum_{k=1}^{n}S_k H_{n-k}$$

$$\sum_{k=0}^{n}kc^k=\frac{nc^{n+2}-(n+1)c^{n+1}+c}{(c-1)^2}$$

$$n!=\sqrt{2\pi n}(\frac{n}{e})^n(1+\frac{1}{12n}+\frac{1}{288n^2}+O(\frac{1}{n^3}))$$

$$\begin{aligned}&\max\{x_a-x_b,y_a-y_b,z_a-z_b\}-\min\{x_a-x_b,y_a-y_b,z_a-z_b\}\\=&\frac{1}{2}\sum_{cyc}|(x_a-y_a)-(x_b-y_b)|\end{aligned}$$

$$(a+b)(b+c)(c+a) = \frac{(a+b+c)^3 - a^3 - b^3 - c^3}{3}$$

Integrals of Rational Functions

$$\int \frac{1}{1+x^2}dx = \tan^{-1} x \tag{1}$$

$$\int \frac{1}{a^2+x^2}dx = \frac{1}{a}\tan^{-1}\frac{x}{a} \tag{2}$$

$$\int \frac{x}{a^2+x^2}dx = \frac{1}{2}\ln|a^2+x^2| \tag{3}$$

$$\int \frac{x^2}{a^2+x^2}dx = x - a\tan^{-1}\frac{x}{a} \tag{4}$$

$$\int \frac{x^3}{a^2+x^2}dx = \frac{1}{2}x^2 - \frac{1}{2}a^2\ln|a^2+x^2| \tag{5}$$

$$\int \frac{1}{ax^2+bx+c}dx = \frac{2}{\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} \tag{6}$$

$$\int \frac{1}{(x+a)(x+b)}dx = \frac{1}{b-a}\ln\frac{a+x}{b+x},\ a \neq b \tag{7}$$

$$\int \frac{x}{(x+a)^2}dx = \frac{a}{a+x} + \ln|a+x| \tag{8}$$

$$\begin{aligned}\int \frac{x}{ax^2+bx+c}dx &= \frac{1}{2a}\ln|ax^2+bx+c| \\ &- \frac{b}{a\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}}\end{aligned} \tag{9}$$

Integrals with Roots

$$\int \frac{x}{\sqrt{x\pm a}}dx = \frac{2}{3}(x\mp 2a)\sqrt{x\pm a} \tag{10}$$

$$\int \sqrt{\frac{x}{a-x}}dx = -\sqrt{x(a-x)} - a\tan^{-1}\frac{\sqrt{x(a-x)}}{x-a} \tag{11}$$

$$\int \sqrt{\frac{x}{a+x}}dx = \sqrt{x(a+x)} - a\ln\left[\sqrt{x}+\sqrt{x+a}\right] \tag{12}$$

$$\int x\sqrt{ax+b}dx = \frac{2}{15a^2}(-2b^2+abx+3a^2x^2)\sqrt{ax+b} \tag{13}$$

$$\begin{aligned}\int \sqrt{x(ax+b)}dx &= \frac{1}{4a^{3/2}}\left[(2ax+b)\sqrt{ax(ax+b)}\right. \\ &\left.-b^2\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right|\right]\end{aligned} \tag{14}$$

$$\begin{aligned}\int \sqrt{x^3(ax+b)}dx &= \left[\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3}\right]\sqrt{x^3(ax+b)} \\ &+ \frac{b^3}{8a^{5/2}}\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right|\end{aligned} \tag{15}$$

$$\int \sqrt{x^2\pm a^2}dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \pm \frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{16}$$

$$\int \sqrt{a^2-x^2}dx = \frac{1}{2}x\sqrt{a^2-x^2} + \frac{1}{2}a^2\tan^{-1}\frac{x}{\sqrt{a^2-x^2}} \tag{17}$$

$$\int x\sqrt{x^2\pm a^2}dx = \frac{1}{3}\left(x^2\pm a^2\right)^{3/2} \tag{18}$$

$$\int \frac{1}{\sqrt{x^2\pm a^2}}dx = \ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{19}$$

$$\int \frac{1}{\sqrt{a^2-x^2}}dx = \sin^{-1}\frac{x}{a} \tag{20}$$

$$\int \frac{x}{\sqrt{x^2\pm a^2}}dx = \sqrt{x^2\pm a^2} \tag{21}$$

$$\int \frac{x}{\sqrt{a^2-x^2}}dx = -\sqrt{a^2-x^2} \tag{22}$$

$$\int \frac{x^2}{\sqrt{x^2\pm a^2}}dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \mp \frac{1}{2}a^2\ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{23}$$

$$\begin{aligned}\int \sqrt{ax^2+bx+c}dx &= \frac{b+2ax}{4a}\sqrt{ax^2+bx+c} \\ &+ \frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx^+c)}\right|\end{aligned} \tag{24}$$

$$\begin{aligned}\int x\sqrt{ax^2+bx+c} &= \frac{1}{48a^{5/2}}\left(2\sqrt{a}\sqrt{ax^2+bx+c}\right. \\ &\times\left(-3b^2+2abx+8a(c+ax^2)\right) \\ &\left.+3(b^3-4abc)\ln\left|b+2ax+2\sqrt{a}\sqrt{ax^2+bx+c}\right|\right)\end{aligned} \tag{25}$$

$$\int \frac{1}{\sqrt{ax^2+bx+c}}dx = \frac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right| \tag{26}$$

$$\begin{aligned}\int \frac{x}{\sqrt{ax^2+bx+c}}dx &= \frac{1}{a}\sqrt{ax^2+bx+c} \\ &- \frac{b}{2a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right|\end{aligned} \tag{27}$$

$$\int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2+x^2}} \tag{28}$$

Integrals with Logarithms

$$\int \frac{\ln ax}{x}dx = \frac{1}{2}(\ln ax)^2 \tag{29}$$

$$\int \ln(ax+b)dx = \left(x+\frac{b}{a}\right)\ln(ax+b) - x, a \neq 0 \tag{30}$$

$$\int \ln(x^2+a^2)\,\mathrm{dx} = x\ln(x^2+a^2) + 2a\tan^{-1}\frac{x}{a} - 2x \tag{31}$$

$$\int \ln(x^2-a^2)\,\mathrm{dx} = x\ln(x^2-a^2) + a\ln\frac{x+a}{x-a} - 2x \tag{32}$$

$$\begin{aligned}\int \ln\left(ax^2+bx+c\right)dx &= \frac{1}{a}\sqrt{4ac-b^2}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} \\ &- 2x + \left(\frac{b}{2a}+x\right)\ln\left(ax^2+bx+c\right)\end{aligned} \tag{33}$$

$$\begin{aligned}\int x\ln(ax+b)dx &= \frac{bx}{2a} - \frac{1}{4}x^2 \\ &+ \frac{1}{2}\left(x^2-\frac{b^2}{a^2}\right)\ln(ax+b)\end{aligned} \tag{34}$$

$$\begin{aligned}\int x\ln\left(a^2-b^2x^2\right)dx &= -\frac{1}{2}x^2 + \\ &\frac{1}{2}\left(x^2-\frac{a^2}{b^2}\right)\ln\left(a^2-b^2x^2\right)\end{aligned} \tag{35}$$

Integrals with Exponentials

$$\int x^n e^{ax}\,\mathrm{d}x = \frac{x^n e^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{d}x \tag{36}$$

$$\int xe^{-ax^2}\,\mathrm{dx} = -\frac{1}{2a}e^{-ax^2} \tag{37}$$

Integrals with Trigonometric Functions

$$\int \sin^3 ax dx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a} \tag{38}$$

$$\int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a} \tag{39}$$

$$\int \cos^3 ax dx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a} \tag{40}$$

$$\int \cos ax\sin bx dx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b \tag{41}$$

$$\begin{aligned}\int \sin^2 ax\cos bx dx &= -\frac{\sin[(2a-b)x]}{4(2a-b)} \\ &+ \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)}\end{aligned} \tag{42}$$

$$\int \sin^2 x\cos x dx = \frac{1}{3}\sin^3 x \tag{43}$$

$$\int \cos^2 ax \sin bx dx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)} \tag{44}$$

$$\int \cos^2 ax \sin ax dx = -\frac{1}{3a}\cos^3 ax \tag{45}$$

$$\int \sin^2 ax \cos^2 bx dx = \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)} \tag{46}$$

$$\int \sin^2 ax \cos^2 ax dx = \frac{x}{8} - \frac{\sin 4ax}{32a} \tag{47}$$

$$\int \tan ax dx = -\frac{1}{a}\ln \cos ax \tag{48}$$

$$\int \tan^2 ax dx = -x + \frac{1}{a}\tan ax \tag{49}$$

$$\int \tan^3 ax dx = \frac{1}{a}\ln \cos ax + \frac{1}{2a}\sec^2 ax \tag{50}$$

$$\int \sec x dx = \ln|\sec x + \tan x| = 2\tanh^{-1}\left(\tan \frac{x}{2}\right) \tag{51}$$

$$\int \sec^2 ax dx = \frac{1}{a}\tan ax \tag{52}$$

$$\int \sec^3 x \, \mathrm{dx} = \frac{1}{2}\sec x \tan x + \frac{1}{2}\ln|\sec x + \tan x| \tag{53}$$

$$\int \sec x \tan x dx = \sec x \tag{54}$$

$$\int \sec^2 x \tan x dx = \frac{1}{2}\sec^2 x \tag{55}$$

$$\int \sec^n x \tan x dx = \frac{1}{n}\sec^n x, n \neq 0 \tag{56}$$

$$\int \csc x dx = \ln\left|\tan \frac{x}{2}\right| = \ln|\csc x - \cot x| + C \tag{57}$$

$$\int \csc^2 ax dx = -\frac{1}{a}\cot ax \tag{58}$$

$$\int \csc^3 x dx = -\frac{1}{2}\cot x \csc x + \frac{1}{2}\ln|\csc x - \cot x| \tag{59}$$

$$\int \csc^n x \cot x dx = -\frac{1}{n}\csc^n x, n \neq 0 \tag{60}$$

$$\int \sec x \csc x dx = \ln|\tan x| \tag{61}$$

Products of Trigonometric Functions and Monomials

$$\int x \cos x dx = \cos x + x \sin x \tag{62}$$

$$\int x \cos ax dx = \frac{1}{a^2}\cos ax + \frac{x}{a}\sin ax \tag{63}$$

$$\int x^2 \cos x dx = 2x \cos x + \left(x^2 - 2\right)\sin x \tag{64}$$

$$\int x^2 \cos ax dx = \frac{2x \cos ax}{a^2} + \frac{a^2x^2 - 2}{a^3}\sin ax \tag{65}$$

$$\int x \sin x dx = -x \cos x + \sin x \tag{66}$$

$$\int x \sin ax dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2} \tag{67}$$

$$\int x^2 \sin x dx = \left(2 - x^2\right)\cos x + 2x \sin x \tag{68}$$

$$\int x^2 \sin ax dx = \frac{2 - a^2x^2}{a^3}\cos ax + \frac{2x \sin ax}{a^2} \tag{69}$$

Products of Trigonometric Functions and Exponentials

$$\int e^x \sin x dx = \frac{1}{2}e^x(\sin x - \cos x) \tag{70}$$

$$\int e^{bx} \sin ax dx = \frac{1}{a^2+b^2}e^{bx}(b \sin ax - a \cos ax) \tag{71}$$

$$\int e^x \cos x dx = \frac{1}{2}e^x(\sin x + \cos x) \tag{72}$$

$$\int e^{bx} \cos ax dx = \frac{1}{a^2+b^2}e^{bx}(a \sin ax + b \cos ax) \tag{73}$$

$$\int xe^x \sin x dx = \frac{1}{2}e^x(\cos x - x \cos x + x \sin x) \tag{74}$$

$$\int xe^x \cos x dx = \frac{1}{2}e^x(x \cos x - \sin x + x \sin x) \tag{75}$$

7.12 Java

```java
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());
    }
}
```

Other methods may have slightly different rounding semantics. For example, the result of the pow method using the specified algorithm can occasionally differ from the rounded mathematical result by more than one unit in the last place, one *ulp*.

Two types of operations are provided for manipulating the scale of a BigDecimal: scaling/rounding operations and decimal point motion operations. Scaling/rounding operations (setScale and round) return a BigDecimal whose value is approximately (or exactly) equal to that of the operand, but whose scale or precision is the specified value; that is, they increase or decrease the precision of the stored number with minimal effect on its value. Decimal point motion operations (movePointLeft and movePointRight) return a BigDecimal created from the operand by moving the decimal point a specified distance in the specified direction.

For the sake of brevity and clarity, pseudo-code is used throughout the descriptions of BigDecimal methods. The pseudo-code expression (i + j) is shorthand for "a BigDecimal whose value is that of the BigDecimal i added to that of the BigDecimal j." The pseudo-code expression (i == j) is shorthand for "true if and only if the BigDecimal i represents the same value as the BigDecimal j." Other pseudo-code expressions are interpreted similarly. Square brackets are used to represent the particular BigInteger and scale pair defining a BigDecimal value; for example [19, 2] is the BigDecimal numerically equal to 0.19 having a scale of 2.

Note: care should be exercised if BigDecimal objects are used as keys in a SortedMap or elements in a SortedSet since BigDecimal's *natural ordering* is *inconsistent with equals*. See Comparable, SortedMap or SortedSet for more information.

All methods and constructors for this class throw NullPointerException when passed a null object reference for any input parameter.

See Also:

BigInteger, MathContext, RoundingMode, SortedMap, SortedSet, Serialized Form

Field Summary

Fields

Modifier and Type	Field and Description
static BigDecimal	**ONE** The value 1, with a scale of 0.
static int	**ROUND_CEILING** Rounding mode to round towards positive infinity.
static int	**ROUND_DOWN** Rounding mode to round towards zero.
static int	**ROUND_FLOOR** Rounding mode to round towards negative infinity.
static int	**ROUND_HALF_DOWN** Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round down.
static int	**ROUND_HALF_EVEN**

	Rounding mode to round towards the "nearest neighbor" unless both neighbors are equidistant, in which case, round towards the even neighbor.
static int	**ROUND_HALF_UP** Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up.
static int	**ROUND_UNNECESSARY** Rounding mode to assert that the requested operation has an exact result, hence no rounding is necessary.
static int	**ROUND_UP** Rounding mode to round away from zero.
static **BigDecimal**	**TEN** The value 10, with a scale of 0.
static **BigDecimal**	**ZERO** The value 0, with a scale of 0.

Constructor Summary

Constructors

Constructor and Description

BigDecimal(**BigInteger** val)
Translates a BigInteger into a BigDecimal.

BigDecimal(**BigInteger** unscaledVal, int scale)
Translates a BigInteger unscaled value and an int scale into a BigDecimal.

BigDecimal(**BigInteger** unscaledVal, int scale, **MathContext** mc)
Translates a BigInteger unscaled value and an int scale into a BigDecimal, with rounding according to the context settings.

BigDecimal(**BigInteger** val, **MathContext** mc)
Translates a BigInteger into a BigDecimal rounding according to the context settings.

BigDecimal(char[] in)
Translates a character array representation of a BigDecimal into a BigDecimal, accepting the same sequence of characters as the **BigDecimal(String)** constructor.

BigDecimal(char[] in, int offset, int len)
Translates a character array representation of a BigDecimal into a BigDecimal, accepting the same sequence of characters as the **BigDecimal(String)** constructor, while allowing a sub-array to be specified.

BigDecimal(char[] in, int offset, int len, **MathContext** mc)
Translates a character array representation of a BigDecimal into a BigDecimal, accepting the same sequence of characters as the **BigDecimal(String)** constructor, while allowing a sub-array to be specified and with rounding according to the context settings.

BigDecimal(char[] in, **MathContext** mc)
Translates a character array representation of a BigDecimal into a BigDecimal, accepting the same sequence of characters as the **BigDecimal(String)** constructor and with rounding according to the context settings.

BigDecimal(double val)
Translates a double into a BigDecimal which is the exact decimal representation of the double's binary floating-point value.

BigDecimal(double val, **MathContext** mc)
Translates a double into a BigDecimal, with rounding according to the context settings.

BigDecimal(int val)
Translates an int into a BigDecimal.

BigDecimal(int val, **MathContext** mc)
Translates an int into a BigDecimal, with rounding according to the context settings.

BigDecimal(long val)
Translates a long into a BigDecimal.

BigDecimal(long val, **MathContext** mc)
Translates a long into a BigDecimal, with rounding according to the context settings.

BigDecimal(**String** val)
Translates the string representation of a BigDecimal into a BigDecimal.

BigDecimal(**String** val, **MathContext** mc)
Translates the string representation of a BigDecimal into a BigDecimal, accepting the same strings as the **BigDecimal(String)** constructor, with rounding according to the context settings.

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
BigDecimal	**abs**() Returns a BigDecimal whose value is the absolute value of this BigDecimal, and whose scale is this.scale().
BigDecimal	**abs**(**MathContext** mc) Returns a BigDecimal whose value is the absolute value of this BigDecimal, with rounding according to the context settings.
BigDecimal	**add**(**BigDecimal** augend) Returns a BigDecimal whose value is (this + augend), and whose scale is max(this.scale(), augend.scale()).
BigDecimal	**add**(**BigDecimal** augend, **MathContext** mc) Returns a BigDecimal whose value is (this + augend), with rounding according to the context settings.

byte	**byteValueExact**() Converts this `BigDecimal` to a `byte`, checking for lost information.
int	**compareTo**(**BigDecimal** val) Compares this `BigDecimal` with the specified `BigDecimal`.
BigDecimal	**divide**(**BigDecimal** divisor) Returns a `BigDecimal` whose value is `(this / divisor)`, and whose preferred scale is `(this.scale() - divisor.scale())`; if the exact quotient cannot be represented (because it has a non-terminating decimal expansion) an `ArithmeticException` is thrown.
BigDecimal	**divide**(**BigDecimal** divisor, int roundingMode) Returns a `BigDecimal` whose value is `(this / divisor)`, and whose scale is `this.scale()`.
BigDecimal	**divide**(**BigDecimal** divisor, int scale, int roundingMode) Returns a `BigDecimal` whose value is `(this / divisor)`, and whose scale is as specified.
BigDecimal	**divide**(**BigDecimal** divisor, int scale, **RoundingMode** roundingMode) Returns a `BigDecimal` whose value is `(this / divisor)`, and whose scale is as specified.
BigDecimal	**divide**(**BigDecimal** divisor, **MathContext** mc) Returns a `BigDecimal` whose value is `(this / divisor)`, with rounding according to the context settings.
BigDecimal	**divide**(**BigDecimal** divisor, **RoundingMode** roundingMode) Returns a `BigDecimal` whose value is `(this / divisor)`, and whose scale is `this.scale()`.
BigDecimal[]	**divideAndRemainder**(**BigDecimal** divisor) Returns a two-element `BigDecimal` array containing the result of `divideToIntegralValue` followed by the result of `remainder` on the two operands.
BigDecimal[]	**divideAndRemainder**(**BigDecimal** divisor, **MathContext** mc) Returns a two-element `BigDecimal` array containing the result of `divideToIntegralValue` followed by the result of `remainder` on the two operands calculated with rounding according to the context settings.
BigDecimal	**divideToIntegralValue**(**BigDecimal** divisor) Returns a `BigDecimal` whose value is the integer part of the quotient `(this / divisor)` rounded down.
BigDecimal	**divideToIntegralValue**(**BigDecimal** divisor, **MathContext** mc) Returns a `BigDecimal` whose value is the integer part of `(this / divisor)`.
double	**doubleValue**()

	Converts this `BigDecimal` to a `double`.
`boolean`	**`equals(Object x)`** Compares this `BigDecimal` with the specified `Object` for equality.
`float`	**`floatValue()`** Converts this `BigDecimal` to a `float`.
`int`	**`hashCode()`** Returns the hash code for this `BigDecimal`.
`int`	**`intValue()`** Converts this `BigDecimal` to an `int`.
`int`	**`intValueExact()`** Converts this `BigDecimal` to an `int`, checking for lost information.
`long`	**`longValue()`** Converts this `BigDecimal` to a `long`.
`long`	**`longValueExact()`** Converts this `BigDecimal` to a `long`, checking for lost information.
`BigDecimal`	**`max(BigDecimal val)`** Returns the maximum of this `BigDecimal` and `val`.
`BigDecimal`	**`min(BigDecimal val)`** Returns the minimum of this `BigDecimal` and `val`.
`BigDecimal`	**`movePointLeft(int n)`** Returns a `BigDecimal` which is equivalent to this one with the decimal point moved `n` places to the left.
`BigDecimal`	**`movePointRight(int n)`** Returns a `BigDecimal` which is equivalent to this one with the decimal point moved `n` places to the right.
`BigDecimal`	**`multiply(BigDecimal multiplicand)`** Returns a `BigDecimal` whose value is `(this × multiplicand)`, and whose scale is `(this.scale() + multiplicand.scale())`.
`BigDecimal`	**`multiply(BigDecimal multiplicand, MathContext mc)`** Returns a `BigDecimal` whose value is `(this × multiplicand)`, with rounding according to the context settings.
`BigDecimal`	**`negate()`** Returns a `BigDecimal` whose value is `(-this)`, and whose scale is `this.scale()`.
`BigDecimal`	**`negate(MathContext mc)`** Returns a `BigDecimal` whose value is `(-this)`, with rounding according to the context settings.
`BigDecimal`	**`plus()`**

	Returns a `BigDecimal` whose value is `(+this)`, and whose scale is `this.scale()`.
BigDecimal	**plus**(**MathContext** mc) Returns a `BigDecimal` whose value is `(+this)`, with rounding according to the context settings.
BigDecimal	**pow**(int n) Returns a `BigDecimal` whose value is $(this^n)$, The power is computed exactly, to unlimited precision.
BigDecimal	**pow**(int n, **MathContext** mc) Returns a `BigDecimal` whose value is $(this^n)$.
int	**precision**() Returns the *precision* of this `BigDecimal`.
BigDecimal	**remainder**(**BigDecimal** divisor) Returns a `BigDecimal` whose value is `(this % divisor)`.
BigDecimal	**remainder**(**BigDecimal** divisor, **MathContext** mc) Returns a `BigDecimal` whose value is `(this % divisor)`, with rounding according to the context settings.
BigDecimal	**round**(**MathContext** mc) Returns a `BigDecimal` rounded according to the `MathContext` settings.
int	**scale**() Returns the *scale* of this `BigDecimal`.
BigDecimal	**scaleByPowerOfTen**(int n) Returns a BigDecimal whose numerical value is equal to (`this` * 10^n).
BigDecimal	**setScale**(int newScale) Returns a `BigDecimal` whose scale is the specified value, and whose value is numerically equal to this `BigDecimal`'s.
BigDecimal	**setScale**(int newScale, int roundingMode) Returns a `BigDecimal` whose scale is the specified value, and whose unscaled value is determined by multiplying or dividing this `BigDecimal`'s unscaled value by the appropriate power of ten to maintain its overall value.
BigDecimal	**setScale**(int newScale, **RoundingMode** roundingMode) Returns a `BigDecimal` whose scale is the specified value, and whose unscaled value is determined by multiplying or dividing this `BigDecimal`'s unscaled value by the appropriate power of ten to maintain its overall value.
short	**shortValueExact**() Converts this `BigDecimal` to a `short`, checking for lost information.
int	**signum**()

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