Wizards

Standard Code Library

2017 年 11 月 20 日

目录

1 数论 2
1.1 $O(m^2 \log n)$ 线性递推 2
1.2 求逆元 2
1.3 中国剩余定理 2
1.4 素性测试 2
1.5 质因数分解 2
1.6 佩尔方程 2
1.7 二次剩余 3
1.8 一元三次方程 3
1.9 线下整点 3
1.10 线性同余不等式 3
1.11 组合数取模 3

2 代数 4
2.1 快速傅里叶变换 4
2.2 快速数论变换 4
2.3 快速沃尔什变换 4
2.4 自适应辛普森积分 4
2.5 单纯形 4

3 计算几何 5
3.1 二维 5
3.1.1 点类 5
3.1.2 凸包 6
3.1.3 凸包最近点对 7
3.1.4 三角形的心 7
3.1.5 半平面交 8
3.1.6 最大空凸包 8
3.1.7 平面最近点对 8
3.1.8 最小覆盖圆 9
3.1.9 多边形内部可视 9
3.2 三维 9
3.2.1 三维点类 9
3.2.2 凸包 10
3.2.3 最小覆盖球 10

4 字符串 11
4.1 AC 自动机 11
4.2 后缀数组 11
4.3 后缀自动机 11
4.4 广义后缀自动机 12
4.5 manacher 12
4.6 回文自动机 12
4.7 循环串的最小表示 12

5 数据结构 13
5.1 可并堆 13
5.2 KD-Tree 13
5.3 Treap 14
5.4 Splay 14
5.5 Link cut Tree 15
5.6 树上莫队 15
5.7 CDQ 分治 16
5.8 整体二分 16

6 图论 16
6.1 2-SAT tarjan 16
6.2 KM 16
6.3 点双连通分量 17
6.4 边双连通分量 17
6.5 树上路径求交 17
6.6 最小树形图 18
6.7 带花树 18
6.8 支配树 18
6.9 随机最大权匹配 19
6.10 无向图最小割 19
6.11 最大团搜索 20
6.12 弦图判定 20
6.13 斯坦纳树 20
6.14 虚树 21
6.15 点分治 21
6.16 最小割最大流 21
6.17 最小费用流 21
6.18 zkw 费用流 22
6.19 最小割树 22
6.20 上下界网络流建图 23
6.20.1 无源汇的上下界可行流 23
6.20.2 有源汇的上下界可行流 23
6.20.3 有源汇的上下界最大流 23
6.20.4 有源汇的上下界最小流 23

7 其他 23
7.1 Dancing Links 23
7.1.1 精确覆盖 23
7.1.2 重复覆盖 23
7.2 蔡勒公式 24
7.3 五边形数定理 24
7.4 凸包闵可夫斯基和 24

8 技巧 24
8.1 STL 归还空间 24
8.2 大整数取模 24
8.3 读入优化 24
8.4 汇编技巧 24

9 提示 25
9.1 线性规划转对偶 25
9.2 NTT 素数及其原根 25
9.3 积分表 25

1. 数论

1.1 $O(m^2 \log n)$ 线性递推

Given $a_0, a_1, \ldots, a_{m-1}$
$a_n = c_0 \times a_{n-m} + \cdots + c_{m-1} \times a_{n-1}$
Solve for $a_n = v_0 \times a_0 + v_1 \times a_1 + \cdots + v_{m-1} \times a_{m-1}$

```
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 求逆元

```
void ex_gcd(long long a, long long b, long long &x, long long &y) {
  if (b == 0) {
    x = 1;
    y = 0;
    return;
  }
  long long xx, yy;
  ex_gcd(b, a % b, xx, yy);
  y = xx - a / b * yy;
  x = yy;
}
long long inv(long long x, long long MODN) {
  long long inv_x, y;
  ex_gcd(x, MODN, inv_x, y);
  return (inv_x % MODN + MODN) % MODN;
}
```

1.3 中国剩余定理

```
//返回 (ans, M), 其中 ans 是模 M 意义下的解
std::pair<long long, long long> CRT(const std::vector<long long>& m, const std::vector<long long>& a) {
  long long M = 1, ans = 0;
  int n = m.size();
  for (int i = 0; i < n; i++) M *= m[i];
  for (int i = 0; i < n; i++) {
    ans = (ans + (M / m[i]) * a[i] % M * inv(M / m[i], m[i])) % M;  // 可能需要大整数相乘取模
  }
  return std::make_pair(ans, M);
}
```

1.4 素性测试

```
int strong_pseudo_primetest(long long n,int base) {
    long long n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=powmod(base,n2,n);
    if((res==1)||(res==n-1)) return 1;
    s--;
    while(s>=0) {
        res=mulmod(res,res,n);
        if(res==n-1) return 1;
        s--;
    }
    return 0; // n is not a strong pseudo prime
}
int isprime(long long n) {
  static LL testNum[]={2,3,5,7,11,13,17,19,23,29,31,37};
  static LL lim[]={4,0,1373653LL,25326001LL,25000000000LL,
  2152302898747LL,3474749660383LL,341550071728321LL,0,0,0,0};
  if(n<2||n==3215031751LL) return 0;
  for(int i=0;i<12;++i){
    if(n<lim[i]) return 1;
    if(strong_pseudo_primetest(n,testNum[i])==0) return 0;
  }
  return 1;
}
```

1.5 质因数分解

```
int ansn; LL ans[1000];
LL func(LL x,LL n){ return(mod_mul(x,x,n)+1)%n; }
LL Pollard(LL n){
  LL i,x,y,p;
  if(Rabin_Miller(n)) return n;
  if(!(n&1)) return 2;
  for(i=1;i<20;i++){
    x=i; y=func(x,n); p=gcd(y-x,n);
    while(p==1) {x=func(x,n); y=func(func(y,n),n); p=gcd((y-x+n)%n,n)%n;}
    if(p==0||p==n) continue;
    return p;
  }
}
void factor(LL n){
  LL x;
  x=Pollard(n);
  if(x==n){ ans[ansn++]=x; return; }
  factor(x), factor(n/x);
}
```

1.6 佩尔方程

```
import java.math.BigInteger;
import java.util.Scanner;
```

```java
public class Main {
    public static BigInteger p, q;
    public static void solve(int n) {
        BigInteger N, p1, p2, q1, q2, a0, a1, a2, g1, g2, h1, h2;
        g1 = q2 = p1 = BigInteger.ZERO;
        h1 = q1 = p2 = BigInteger.ONE;
        a0 = a1 = BigInteger.valueOf((long)Math.sqrt(1.0*n));
        N = BigInteger.valueOf(n);
        while (true) {
            g2 = a1.multiply(h1).subtract(g1);
            h2 = N.subtract(g2.pow(2)).divide(h1);
            a2 = g2.add(a0).divide(h2);
            p = a1.multiply(p2).add(p1);
            q = a1.multiply(q2).add(q1);
            if (p.pow(2).subtract(N.multiply(q.pow(2)))
        .compareTo(BigInteger.ONE) == 0) return;
      g1 = g2;h1 = h2;a1 = a2;
            p1 = p2;p2 = p;
            q1 = q2;q2 = q;
        }
    }
    public static void main(String[] args) {
        Scanner cin = new Scanner(System.in);
        int t=cin.nextInt();
        while (t--!=0) {
            solve(cin.nextInt());
            System.out.println(p + " " + q);
        }
    }
}
```

1.7 二次剩余

```cpp
// x^2 = a (mod p), 0 <= a < p, 返回 true or false 代表是否存在解
// p 必须是质数，若是多个单次质数的乘积，可以分别求解再用 CRT 合并
void multiply(ll &c, ll &d, ll a, ll b, ll w) {
    int cc = (a * c + b * d % MOD * w) % MOD;
    int dd = (a * d + b * c) % MOD;
    c = cc, d = dd;
}
bool solve(int n, int &x) {
    if (MOD == 2) return x = 1, true;
    if (power(n, MOD / 2, MOD) == MOD - 1) return false;
    ll c = 1, d = 0, b = 1, a, w;
    do { a = rand() % MOD;
        w = (a * a - n + MOD) % MOD;
        if (w == 0) return x = a, true;
    } while (power(w, MOD / 2, MOD) != MOD - 1);
    for (int times = (MOD + 1) / 2; times; times >>= 1) {
        if (times & 1) multiply(c, d, a, b, w);
        multiply(a, b, a, b, w);
    }
    // x = (a + sqrt(w)) ^ ((p + 1) / 2)
    return x = c, true;
}
```

1.8 一元三次方程

```cpp
double a(p[3]), b(p[2]), c(p[1]), d(p[0]);
double k(b / a), m(c / a), n(d / a);
double p(-k * k / 3. + m);
double q(2. * k * k * k / 27 - k * m / 3. + n);
Complex omega[3] = {Complex(1, 0), Complex(-0.5, 0.5 * sqrt(3)), Complex(-0.5, -0.5 * sqrt(3))};
Complex r1, r2;
double delta(q * q / 4 + p * p * p / 27);
if (delta > 0) {
    r1 = cubrt(-q / 2. + sqrt(delta));
    r2 = cubrt(-q / 2. - sqrt(delta));
} else {
    r1 = pow(-q / 2. + pow(Complex(delta), 0.5), 1. / 3);
    r2 = pow(-q / 2. - pow(Complex(delta), 0.5), 1. / 3);
}
for(int _(0); _ < 3; _++) {
    Complex x = -k / 3. + r1 * omega[_ * 1] + r2 * omega[_ * 2 % 3];
}
```

1.9 线下整点

```cpp
// $\sum_{i=0}^{n-1}\lfloor\frac{a+bi}{m}\rfloor$, $n,m,a,b>0$
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);
}
```

1.10 线性同余不等式

```cpp
// Find the minimal non-negative solutions for $l \le d \cdot x \bmod m \le r$
// $0 \le d,l,r < m; l \le r, O(\log n)$
ll cal(ll m, ll d, ll l, ll r) {
    if (l == 0) return 0;
    if (d == 0) return MXL; // 无解
    if (d * 2 > m) return cal(m, m - d, m - r, m - l);
    if ((l - 1) / d < r / d) return (l - 1) / d + 1;
    ll k = cal(d, (-m % d + d) % d, l % d, r % d);
    return k == MXL ? MXL : (k * m + l - 1) / d + 1;
}
```

1.11 组合数取模

```cpp
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);
    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;
}
```

2. 代数

2.1 快速傅里叶变换

```
void fft(Complex a[], int n, int f) {
  for (int i = 0; i < n; ++i)
    if (R[i] < i) swap(a[i], a[R[i]]);
  for (int i = 1, h = 0; i < n; i <<= 1, h++) {
    Complex wn = Complex(cos(pi / i), f * sin(pi / i));
    Complex w = Complex(1, 0);
    for (int k = 0; k < i; ++k, w = w * wn) tmp[k] = w;
    for (int p = i << 1, j = 0; j < n; j += p) {
      for (int k = 0; k < i; ++k) {
        Complex x = a[j + k], y = a[j + k + i] * tmp[k];
        a[j + k] = x + y; a[j + k + i] = x - y;
      }
    }
  }
}
```

2.2 快速数论变换

```
// n 必须是 2 的次幂
void fft(Complex a[], int n, int f) {
  for (int i = 0; i < n; ++i)
    if (R[i] < i) swap(a[i], a[R[i]]);
  for (int i = 1, h = 0; i < n; i <<= 1, h++) {
    Complex wn = Complex(cos(pi / i), f * sin(pi / i));
    Complex w = Complex(1, 0);
    for (int k = 0; k < i; ++k, w = w * wn) tmp[k] = w;
    for (int p = i << 1, j = 0; j < n; j += p) {
      for (int k = 0; k < i; ++k) {
        Complex x = a[j + k], y = a[j + k + i] * tmp[k];
        a[j + k] = x + y; a[j + k + i] = x - y;
      }
    }
  }
}
```

2.3 快速沃尔什变换

```
void FWT(LL a[],int n,int ty){ //the length is 2^n
  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;
                  //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;
                  //and:a[i+j]=x-y; or:a[i+j+d]=y-x;
        }
      }
    }
  }
}
```

2.4 自适应辛普森积分

```
namespace adaptive_simpson {
  template<typename function>
  inline double area(function f, const double &left, const
      double &right) {
    double mid = (left + right) / 2;
    return (right - left) * (f(left) + 4 * f(mid) +
        f(right)) / 6;
  }
  template<typename function>
  inline double simpson(function f, const double &left,
      const double &right, const double &eps, const
      double &area_sum) {
    double mid = (left + right) / 2;
    double area_left = area(f, left, mid);
    double area_right = area(f, mid, right);
    double area_total = area_left + area_right;
    if (fabs(area_total - area_sum) <= 15 * eps) {
      return area_total + (area_total - area_sum) / 15;
    }
    return simpson(f, left, right, eps / 2, area_left) +
        simpson(f, mid, right, eps / 2, area_right);
  }
  template<typename function>
  inline double simpson(function f, const double &left,
      const double &right, const double &eps) {
    return simpson(f, left, right, eps, area(f, left,
        right));
  }
}
```

2.5 单纯形

```
const double eps = 1e-8;
// max{c * x | Ax <= b, x >= 0} 的解, 无解返回空的
    vector, 否则就是解.
vector<double> simplex(vector<vector<double> > &A,
    vector<double> b, vector<double> c) {
  int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
  vector<vector<double> > D(n + 2, vector<double>(m + 1));
  vector<int> ix(n + m);
  for(int i = 0; i < n + m; i++)  {
    ix[i] = i;
  }
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < m - 1; j++) {
      D[i][j] = -A[i][j];
    }
    D[i][m - 1] = 1;
    D[i][m] = b[i];
    if (D[r][m] > D[i][m]) {
      r = i;
    }
  }
  for(int j = 0; j < m - 1; j++) {
    D[n][j] = c[j];
  }
  D[n + 1][m - 1] = -1;
  for(double d; ;) {
    if (r < n) {
      swap(ix[s], ix[r + m]);
      D[r][s] = 1. / D[r][s];
      for(int j = 0; j <= m; j++) {
        if (j != s) {
          D[r][j] *= -D[r][s];
        }
      }
      for(int i = 0; i <= n + 1; i++) {
        if (i != r) {
          for(int j = 0; j <= m; j++) {
            if (j != s) {
```

```
              D[i][j] += D[r][j] * D[i][s];
            }
          }
          D[i][s] *= D[r][s];
        }
      }
    }
    r = -1, s = -1;
    for(int j = 0; j < m; j++) {
      if (s < 0 || ix[s] > ix[j]) {
        if (D[n + 1][j] > eps || D[n + 1][j] > -eps && D[n][j] > eps) {
          s = j;
        }
      }
    }
    if (s < 0) break;
    for(int i = 0; i < n; i++) {
      if (D[i][s] < -eps) {
        if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -eps
          || d < eps && ix[r + m] > ix[i + m]) {
          r = i;
        }
      }
    }
    if (r < 0) return vector<double> ();
  }
  if (D[n + 1][m] < -eps) return vector<double> ();
  vector<double> x(m - 1);
  for(int i = m; i < n + m; i++) {
    if (ix[i] < m - 1) {
      x[ix[i]] = D[i - m][m];
    }
  }
  return x;
}
```

3. 计算几何

3.1 二维

3.1.1 点类

```
int sign(DB x) {
  return (x > eps) - (x < -eps);
}
DB msqrt(DB x) {
  return sign(x) > 0 ? sqrt(x) : 0;
}
struct Point {
  DB x, y;
  Point rotate(DB ang) const {  // 逆时针旋转 ang 弧度
    return Point(cos(ang) * x - sin(ang) * y, cos(ang) * y + sin(ang) * x);
  }
  Point turn90() const {  // 逆时针旋转 90 度
    return Point(-y, x);
  }
  Point unit() const {
    return *this / len();
  }
};
DB dot(const Point& a, const Point& b) {
  return a.x * b.x + a.y * b.y;
}
DB det(const Point& a, const Point& b) {
  return a.x * b.y - a.y * b.x;
}
#define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x-p1.x)*(p2.y-p1.y))
#define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))
bool isLL(const Line& l1, const Line& l2, Point& p) {
  DB s1 = det(l2.b - l2.a, l1.a - l2.a),
     s2 = -det(l2.b - l2.a, l1.b - l2.a);
  if (!sign(s1 + s2)) return false;
  p = (l1.a * s2 + l1.b * s1) / (s1 + s2);
  return true;
}
bool onSeg(const Line& l, const Point& p) {
  return sign(det(p - l.a, l.b - l.a)) == 0 && sign(dot(p - l.a, p - l.b)) <= 0;
}
Point projection(const Line & l, const Point& p) {
  return l.a + (l.b - l.a) * (dot(p - l.a, l.b - l.a) / (l.b - l.a).len2());
}
DB disToLine(const Line& l, const Point& p) {
  return fabs(det(p - l.a, l.b - l.a) / (l.b - l.a).len());
}
DB disToSeg(const Line& l, const Point& p) {
  return sign(dot(p - l.a, l.b - l.a)) * sign(dot(p - l.b, l.a - l.b)) == 1 ? disToLine(l, p) : std::min((p - l.a).len(), (p - l.b).len());
}
// 圆与直线交点
bool isCL(Circle a, Line l, Point& p1, Point& p2) {
  DB x = dot(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 (sign(d) < 0) return false;
  Point p = l.a - ((l.b - l.a) * (x / y)), delta = (l.b - l.a) * (msqrt(d) / y);
  p1 = p + delta; p2 = p - delta;
  return true;
}
//圆与圆的交面积
DB areaCC(const Circle& c1, const Circle& c2) {
  DB d = (c1.o - c2.o).len();
  if (sign(d - (c1.r + c2.r)) >= 0) return 0;
  if (sign(d - std::abs(c1.r - c2.r)) <= 0) {
    DB r = std::min(c1.r, c2.r);
    return r * r * PI;
  }
  DB x = (d * d + c1.r * c1.r - c2.r * c2.r) / (2 * d),
    t1 = acos(x / c1.r), t2 = acos((d - x) / c2.r);
  return c1.r * c1.r * t1 + c2.r * c2.r * t2 - d * c1.r * sin(t1);
}
// 圆与圆交点
bool isCC(Circle a, Circle b, P& p1, P& p2) {
  DB s1 = (a.o - b.o).len();
  if (sign(s1 - a.r - b.r) > 0 || sign(s1 - std::abs(a.r - b.r)) < 0) return false;
  DB s2 = (a.r * a.r - b.r * b.r) / s1;
  DB 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).unit().turn90() * msqrt(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
bool tanCP(const Circle &c, const Point &p0, Point &p1, Point &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
  Point p = (p0 - c.o) * (c.r * c.r / x);
  Point delta = ((p0 - c.o) * (-c.r * sqrt(d) / x)).turn90();
  p1 = c.o + p + delta;
  p2 = c.o + p - delta;
  return true;
}
```

```cpp
// 求圆到圆的外共切线，按关于 c1.o 的顺时针方向
vector<Line> extanCC(const Circle &c1, const Circle &c2) {
  vector<Line> ret;
  if (sign(c1.r - c2.r) == 0) {
    Point dir = c2.o - c1.o;
    dir = (dir * (c1.r / dir.len())).turn90();
    ret.push_back(Line(c1.o + dir, c2.o + dir));
    ret.push_back(Line(c1.o - dir, c2.o - dir));
  } else {
    Point p = (c1.o * -c2.r + c2.o * c1.r) / (c1.r - c2.r);
    Point p1, p2, q1, q2;
    if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) {
      if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
      ret.push_back(Line(p1, q1));
      ret.push_back(Line(p2, q2));
    }
  }
  return ret;
}
// 求圆到圆的内共切线，按关于 c1.o 的顺时针方向
std::vector<Line> intanCC(const Circle &c1, const Circle &c2) {
  std::vector<Line> ret;
  Point p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
  Point p1, p2, q1, q2;
  if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) { // 两圆相切认为没有切线
    ret.push_back(Line(p1, q1));
    ret.push_back(Line(p2, q2));
  }
  return ret;
}
bool contain(vector<Point> polygon, Point p) { // 判断点 p 是否被多边形包含，包括落在边界上
  int ret = 0, n = polygon.size();
  for(int i = 0; i < n; ++ i) {
    Point u = polygon[i], v = polygon[(i + 1) % n];
    if (onSeg(Line(u, v), p)) return true;  // Here I guess.
    if (sign(u.y - v.y) <= 0) swap(u, v);
    if (sign(p.y - u.y) > 0 || sign(p.y - v.y) <= 0) continue;
    ret += sign(det(p, v, u)) > 0;
  }
  return ret & 1;
}
// 用半平面 (q1,q2) 的逆时针方向去切凸多边形
std::vector<Point> convexCut(const std::vector<Point>&ps, Point q1, Point q2) {
  std::vector<Point> qs; int n = ps.size();
  for (int i = 0; i < n; ++i) {
    Point p1 = ps[i], p2 = ps[(i + 1) % n];
    int d1 = crossOp(q1,q2,p1), d2 = crossOp(q1,q2,p2);
    if (d1 >= 0) qs.push_back(p1);
    if (d1 * d2 < 0) qs.push_back(isSS(p1, p2, q1, q2));
  }
  return qs;
}
// 求凸包
std::vector<Point> convexHull(std::vector<Point> ps) {
  int n = ps.size(); if (n <= 1) return ps;
  std::sort(ps.begin(), ps.end());
  std::vector<Point> qs;
  for (int i = 0; i < n; qs.push_back(ps[i ++]))
    while (qs.size() > 1 && sign(det(qs[qs.size() - 2], qs.back(), ps[i])) <= 0)
      qs.pop_back();
  for (int i = n - 2, t = qs.size(); i >= 0; qs.push_back(ps[i --]))
    while ((int)qs.size() > t && sign(det(qs[qs.size() - 2], qs.back(), ps[i])) <= 0)
      qs.pop_back();
  return qs;
}
```

3.1.2 凸包

```cpp
// 凸包中的点按逆时针方向
struct Convex {
  int n;
  std::vector<Point> a, upper, lower;
  void make_shell(const std::vector<Point>& p,
      std::vector<Point>& shell) {  // p needs to be sorted.
    clear(shell); int n = p.size();
    for (int i = 0, j = 0; i < n; i++, j++) {
      for (; j >= 2 && sign(det(shell[j-1] - shell[j-2],
              p[i] - shell[j-2])) <= 0; --j) shell.pop_back();
      shell.push_back(p[i]);
    }
  }
  void make_convex() {
    std::sort(a.begin(), a.end());
    make_shell(a, lower);
    std::reverse(a.begin(), a.end());
    make_shell(a, upper);
    a = lower; a.pop_back();
    a.insert(a.end(), upper.begin(), upper.end());
    if ((int)a.size() >= 2) a.pop_back();
    n = a.size();
  }
  void init(const std::vector<Point>& _a) {
    clear(a); a = _a; n = a.size();
    make_convex();
  }
  void read(int _n) {  // Won't make convex.
    clear(a); n = _n; a.resize(n);
    for (int i = 0; i < n; i++)
      a[i].read();
  }
  std::pair<DB, int> get_tangent(
      const std::vector<Point>& convex, const Point& vec) {
    int l = 0, r = (int)convex.size() - 2;
    assert(r >= 0);
    for (; l + 1 < r; ) {
      int mid = (l + r) / 2;
      if (sign(det(convex[mid + 1] - convex[mid], vec)) > 0)
        r = mid;
      else l = mid;
    }
    return std::max(std::make_pair(det(vec, convex[r]), r),
        std::make_pair(det(vec, convex[0]), 0));
  }
  int binary_search(Point u, Point v, int l, int r) {
    int s1 = sign(det(v - u, a[l % n] - u));
    for (; l + 1 < r; ) {
      int mid = (l + r) / 2;
      int smid = sign(det(v - u, a[mid % n] - u));
      if (smid == s1) l = mid;
      else r = mid;
    }
    return l % n;
  }
  // 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) {
    std::pair<DB, int> ret = get_tangent(upper, vec);
    ret.second = (ret.second + (int)lower.size() - 1) % n;
    ret = std::max(ret, get_tangent(lower, vec));
    return ret.second;
```

```cpp
  }
  // 求凸包和直线 u, v 的交点，如果不相交返回 false，如果有则是和 (i, next(i)) 的交点，交在点上不确定返回前后两条边其中之一
  bool get_intersection(Point u, Point v, int &i0, int &i1) {
    int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
    if (sign(det(v - u, a[p0] - u)) * sign(det(v - u, a[p1] - u)) <= 0) {
      if (p0 > p1) std::swap(p0, p1);
      i0 = binary_search(u, v, p0, p1);
      i1 = binary_search(u, v, p1, p0 + n);
      return true;
    }
    else return false;
  }
};
```

3.1.3 凸包最近点对

```cpp
//判断点是否在多边形内
int isPointInPolygon(point p, point *a, int n) {
    int cnt = 0;
    for(int i=0; i<n; ++i) {
        if(OnSegment(p, a[i], a[(i+1)%n])) return -1;
        double k = cross(a[(i+1)%n]-a[i], p-a[i]);
        double d1 = a[i].y - p.y;
    double  d2 = a[(i+1)].y - p.y;
        if(k>0 &&d1<=0 &&d2>0) cnt++;
        if(k<0 &&d2<=0 &&d1>0) cnt++;
        //k==0，点和线段共线的情况不考虑
    }
    if(cnt&1)return 1;
    return 0;
}
//判断凸包是否相离
bool two_getaway_ConvexHull(point *cha, int n1, point *chb, int m1) {
    if(n1==1 && m1==1) {
        if(cha[0]==chb[0])
            return false;
    } else if(n1==1 && m1==2) {
        if(OnSegment(cha[0], chb[0], chb[1]))
            return false;
    } else if(n1==2 && m1==1) {
        if(OnSegment(chb[0], cha[0], cha[1]))
            return false;
    } else if(n1==2 && m1==2) {
        if(SegmentIntersection(cha[0], cha[1], chb[0], chb[1]))
            return false;
    } else if(n1==2) {
        for(int i=0; i<n1; ++i)
            if(isPointInPolygon(cha[i], chb, m1))
                return false;
    } else if(m1==2) {
        for(int i=0; i<m1; ++i)
            if(isPointInPolygon(chb[i], cha, n1))
                return false;
    } else {
        for(int i=0; i<n1; ++i) {
            for(int j=0; j<m1; ++j) {
                if(SegmentIntersection(cha[i], cha[(i+1)%n1], chb[j], chb[(j+1)%m1]))
                    return false;
            }
        }
        for(int i=0; i<n1; ++i)
            if(isPointInPolygon(cha[i], chb, m1))
                return false;
        for(int i=0; i<m1; ++i)
            if(isPointInPolygon(chb[i], cha, n1))
                return false;
    }
    return true;
}
//旋转卡壳求两个凸包最近距离
double solve(point *P, point *Q, int n, int m) {
    if(n==1 && m==1) {
        return length(P[0] - Q[0]);
    } else if(n==1 && m==2) {
        return DistanceToSegment(P[0], Q[0], Q[1]);
    } else if(n==2 && m==1) {
        return DistanceToSegment(Q[0], P[0], P[1]);
    } else if(n==2 && m==2) {
        return SegmentToSegment(P[0], P[1], Q[0], Q[1]);
    }

    int yminP = 0, ymaxQ = 0;
    for(int i=0; i<n; ++i) if(P[i].y < P[yminP].y) yminP = i;
    for(int i=0; i<m; ++i) if(Q[i].y > Q[ymaxQ].y) ymaxQ = i;
    P[n] = P[0];
    Q[n] = Q[0];
    double INF2 = 1e100;
    double arg, ans = INF2;
    for(int i=0; i<n; ++i) {
        //当叉积负正转正时，说明点 ymaxQ 就是对踵点
        while((arg=cross(P[yminP] - P[yminP+1],Q[ymaxQ+1] - Q[ymaxQ])) < -eps)
            ymaxQ = (ymaxQ+1)%m;
        double ret;
        if(arg > eps) { //卡住第二个凸包上的点。
            ret = DistanceToSegment(Q[ymaxQ], P[yminP], P[yminP+1]);
            ans  = min(ans,ret);
        } else { //arg==0，卡住第二个凸包的边
            ret = SegmentToSegment(P[yminP],P[yminP+1],
          Q[ymaxQ],Q[ymaxQ+1]);
            ans = min(ans,ret);
        }
        yminP = (yminP+1)%n;
    }
    return ans;
}
double mindis_twotubao(point *P, point *Q, int n, int m){
    //return min(solve(P, Q, n, m),solve(Q,P,m,n));
    if(two_getaway_ConvexHull(P,n,Q,m)==true) return min(solve(P, Q, n, m),solve(Q,P,m,n));
    else return 0.0;
}
```

3.1.4 三角形的心

```cpp
Point inCenter(const Point &A, const Point &B, const Point &C) { // 内心
  double a = (B - C).len(), b = (C - A).len(), c = (A - B).len(),
    s = fabs(det(B - A, C - A)),
    r = s / p;
  return (A * a + B * b + C * c) / (a + b + c);
}
Point circumCenter(const Point &a, const Point &b, const Point &c) { // 外心
  Point bb = b - a, cc = c - a;
  double db = bb.len2(), dc = cc.len2(), d = 2 * det(bb, cc);
  return a - Point(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
}
Point othroCenter(const Point &a, const Point &b, const Point &c) { // 垂心
```

```
  Point ba = b - a, ca = c - a, bc = b - c;
  double Y = ba.y * ca.y * bc.y,
      A = ca.x * ba.y - ba.x * ca.y,
      x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
      y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
  return Point(x0, y0);
}
```

3.1.5 半平面交

```
struct Point {
  int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0);}
};
struct Line {
  bool include(const Point &p) const { return sign(det(b - a, p - a)) > 0; }
  Line push() const{ // 将半平面向外推 eps
    const double eps = 1e-6;
    Point delta = (b - a).turn90().norm() * eps;
    return Line(a - delta, b - delta);
  }
};
bool sameDir(const Line &l0, const Line &l1) { return parallel(l0, l1) && sign(dot(l0.b - l0.a, l1.b - l1.a)) == 1; }
bool operator < (const Point &a, const Point &b) {
  if (a.quad() != b.quad()) {
    return a.quad() < b.quad();
  } else {
    return sign(det(a, b)) > 0;
  }
}
bool operator < (const Line &l0, const Line &l1) {
  if (sameDir(l0, l1)) {
    return l1.include(l0.a);
  } else {
    return (l0.b - l0.a) < (l1.b - l1.a);
  }
}
bool check(const Line &u, const Line &v, const Line &w) { return w.include(intersect(u, v)); }
vector<Point> intersection(vector<Line> &l) {
  sort(l.begin(), l.end());
  deque<Line> 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<Point> ret;
  for (int i = 0; i < (int)q.size(); ++i) ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
  return ret;
}
```

3.1.6 最大空凸包

```
inline double eq(double x, double y) {
    return fabs(x-y)<eps;
}
double xmult(point a, point b, point o) {
    return (a.x-o.x)*(o.y-b.y)-(a.y-o.y)*(o.x-b.x);
}
double dist(point a, point b) {
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
point o;
bool cmp_angle(point a,point b) {
    if(eq(xmult(a,b,o),0.0)) {
        return dist(a,o)<dist(b,o);
    }
    return xmult(a,o,b)>0;
}
double empty_convex(point *p, int pn) {
    double ans=0;
    for(int i=0; i<pn; i++) {
        for(int j=0; j<pn; j++) {
            dp[i][j]=0;
        }
    }
    for(int i=0; i<pn; i++) {
        int j = i-1;
        while(j>=0 && eq(xmult(p[i], p[j], o),0.0))j--;//coline
        bool flag= j==i-1;
        while(j>=0) {
            int k = j-1;
            while(k >= 0 && xmult(p[i],p[k],p[j])>0)k--;
            double area = fabs(xmult(p[i],p[j],o))/2;
            if(k >= 0)area+=dp[j][k];
            if(flag) dp[i][j]=area;
            ans=max(ans,area);
            j=k;
        }
        if(flag) {
            for(int j=1; j<i; j++) {
                dp[i][j] = max(dp[i][j],dp[i][j-1]);
            }
        }
    }
    return ans;
}
double largest_empty_convex(point *p, int pn) {
    point data[maxn];
    double ans=0;
    for(int i=0; i<pn; i++) {
        o=p[i];
        int dn=0;
        for(int j=0; j<pn; j++) {
            if(p[j].y>o.y||(p[j].y==o.y&&p[j].x>=o.x)) {
                data[dn++]=p[j];
            }
        }
        sort(data, data+dn, cmp_angle);
        ans=max(ans, empty_convex(data, dn));
    }
    return ans;
}
```

3.1.7 平面最近点对

```
double Dis(Point a, Point b) {
  return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}
double Closest_Pair(int left, int right) {
  double d = INF;
  if(left == right) return d;
  if(left +1 == right)
    return Dis(p[left],p[right]);
  int mid = (left+right)>>1;
  double d1 = Closest_Pair(left,mid);
  double d2 = Closest_Pair(mid,right);
```

```cpp
  d = min(d1,d2);
  int k = 0;
  for(int i = left; i <= right; i++) {
    if(fabs(p[mid].x - p[i].x) <= d)
      temp[k++] = p[i];
  }
  sort(temp,temp+k,cmpy);
  for(int i = 0; i < k; i++) {
    for(int j = i+1; j < k && temp[j].y - temp[i].y < d;
      j++) {
      double d3 = Dis(temp[i],temp[j]);
      d = min(d,d3);
    }
  }
  return d;
}
```

3.1.8 最小覆盖圆

```cpp
#include<cmath>
#include<cstdio>
#include<algorithm>
using namespace std;
const double eps=1e-6;
struct couple {
  double x, y;
  couple(){}
  couple(const double &xx, const double &yy) {
    x = xx; y = yy;
  }
} a[100001];
int n;
//dis means distance, dis2 means square of it
struct circle {
  double r; couple c;
} cir;
inline bool inside(const couple & x) {
  return di2(x, cir.c) < cir.r*cir.r+eps;
}
inline void p2c(int x, int y) {
  cir.c.x = (a[x].x+a[y].x)/2;
  cir.c.y = (a[x].y+a[y].y)/2;
  cir.r = dis(cir.c, a[x]);
}
inline void p3c(int i, int j, int k) {
  couple x = a[i], y = a[j], z = a[k];
  cir.r =
    sqrt(di2(x,y)*di2(y,z)*di2(z,x))/fabs(x*y+y*z+z*x)/2;
  couple t1((x-y).x, (y-z).x), t2((x-y).y, (y-z).y),
    t3((len(x)-len(y))/2, (len(y)-len(z))/2);
  cir.c = couple(t3*t2, t1*t3)/(t1*t2);
}
inline circle mi() {
  sort(a + 1, a + 1 + n);
  n = unique(a + 1, a + 1 + n) - a - 1;
  if(n == 1) {
    cir.c = a[1];
    cir.r = 0;
    return cir;
  }
  random_shuffle(a + 1, a + 1 + n);
  p2c(1, 2);
  for(int i = 3; i <= n; i++)
    if(!inside(a[i])) {
      p2c(1, i);
      for(int j = 2; j < i; j++)
        if(!inside(a[j])) {
          p2c(i, j);
          for(int k = 1; k < j; k++)
            if(!inside(a[k]))
              p3c(i,j, k);
        }
    }
  return cir;
}
```

3.1.9 多边形内部可视

```cpp
int C(const Point & P, const Point & A, const Point & Q,
  const Point & B) {
  Point C = GetIntersection(P, A - P, Q, Q - B);
  return OnLine(Q, C, B);
}
int Onleft(const Point & a, const Point &b, const Point &
  c) {
  return dcmp(Cross(b - c, a - c)) > 0;
}
int visible(int x, int y) {
  int P = (x + n - 1) % n, Q = (x + 1) % n;
  Point u = p[y] - p[x], v = p[x] - p[P], w = p[x] - p[Q];
  if (Onleft(p[Q], p[x], p[P])) {
    return dcmp(Cross(v, u)) > 0 && dcmp(Cross(w, u)) < 0;
  } else {
    return !(dcmp(Cross(v, u)) < 0 && dcmp(Cross(w, u)) >
      0);
  }
}
int solve(int x, int y) {
  if (vis[x][y] == dfn) return g[x][y];
  vis[x][y] = dfn;
  if (x == y || y == x + 1) return g[x][y] = 1;
  for (int i = x; i + 1 <= y; i++) {
    if (C(p[x], p[y], p[i], p[i + 1])) return g[x][y] = 0;
  }
  for (int i = x + 1; i < y; i++) {
    if (OnLine(p[x], p[i], p[y])) {
      return g[x][y] = solve(x, i) && solve(i, y);
    }
  }
  if (!visible(x, y) || !visible(y, x)) return g[x][y] =
    0;
  return g[x][y] = 1;
}
```

3.2 三维

3.2.1 三维点类

```cpp
// 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
Point rotate(const Point& s, const Point& axis, DB w) {
  DB x = axis.x, y = axis.y, z = axis.z;
  DB s1 = x * x + y * y + z * z, ss1 = msqrt(s1),
     cosw = cos(w), sinw = sin(w);
  DB 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;
  DB 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 Point(ans[0], ans[1], ans[2]);
}
```

3.2.2 凸包

```
__inline P cross(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
        );
}
__inline DB mix(const P& a, const P& b, const P& c) {
  return dot(cross(a, b), c);
}
__inline DB volume(const P& a, const P& b, const P& c,
   ↪ const P& d) {
  return mix(b - a, c - a, d - a);
}
struct Face {
  int a, b, c;
  __inline Face() {}
  __inline Face(int _a, int _b, int _c):
    a(_a), b(_b), c(_c) {}
  __inline DB area() const {
    return 0.5 * cross(p[b] - p[a], p[c] - p[a]).len();
  }
  __inline P normal() const {
    return cross(p[b] - p[a], p[c] - p[a]).unit();
  }
  __inline DB dis(const P& p0) const {
    return dot(normal(), p0 - p[a]);
  }
};
std::vector<Face> face, tmp;  // Should be O(n).
int mark[N][N], Time, n;
__inline void add(int v) {
  ++ Time;
  clear(tmp);
  for (int i = 0; i < (int)face.size(); ++ i) {
    int a = face[i].a, b = face[i].b, c = face[i].c;
    if (sign(volume(p[v], p[a], p[b], p[c])) > 0) {
      mark[a][b] = mark[b][a] = mark[a][c] =
        mark[c][a] = mark[b][c] = mark[c][b] = Time;
    } else {
      tmp.push_back(face[i]);
    }
  }
  clear(face); face = tmp;
  for (int i = 0; i < (int)tmp.size(); ++ i) {
    int a = face[i].a, b = face[i].b, c = face[i].c;
    if (mark[a][b] == Time) face.emplace_back(v, b, a);
    if (mark[b][c] == Time) face.emplace_back(v, c, b);
    if (mark[c][a] == Time) face.emplace_back(v, a, c);
    assert(face.size() < 500u);
  }
}
void reorder() {
  for (int i = 2; i < n; ++ i) {
    P tmp = cross(p[i] - p[0], p[i] - p[1]);
    if (sign(tmp.len())) {
      std::swap(p[i], p[2]);
      for (int j = 3; j < n; ++ j)
        if (sign(volume(p[0], p[1], p[2], p[j]))) {
          std::swap(p[j], p[3]);
          return;
        }
    }
  }
}
void build_convex() {
  reorder();
  clear(face);
  face.emplace_back(0, 1, 2);
  face.emplace_back(0, 2, 1);
  for (int i = 3; i < n; ++ i)
    add(i);
}
```

3.2.3 最小覆盖球

```
const int eps = 1e-8;
struct Tpoint {
  double x, y, z;
};
int npoint, nouter;
Tpoint pt[200000], outer[4],res;
double radius,tmp;
inline double dist(Tpoint p1, Tpoint p2) {
  double dx=p1.x-p2.x, dy=p1.y-p2.y, dz=p1.z-p2.z;
  return ( dx*dx + dy*dy + dz*dz );
}
inline double dot(Tpoint p1, Tpoint p2) {
  return p1.x*p2.x + p1.y*p2.y + p1.z*p2.z;
}
void ball() {
  Tpoint q[3]; double m[3][3], sol[3], L[3], det;
  int i,j;
  res.x = res.y = res.z = radius = 0;
  switch ( nouter ) {
    case 1: res=outer[0]; break;
    case 2:
        res.x=(outer[0].x+outer[1].x)/2;
        res.y=(outer[0].y+outer[1].y)/2;
        res.z=(outer[0].z+outer[1].z)/2;
        radius=dist(res, outer[0]);
        break;
    case 3:
        for (i=0; i<2; ++i ) {
          q[i].x=outer[i+1].x-outer[0].x;
          q[i].y=outer[i+1].y-outer[0].y;
          q[i].z=outer[i+1].z-outer[0].z;
        }
        for (i=0; i<2; ++i) for(j=0; j<2; ++j)
          m[i][j]=dot(q[i], q[j])*2;
        for (i=0; i<2; ++i ) sol[i]=dot(q[i], q[i]);
        if (fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0])<eps)
          return;
        L[0]=(sol[0]*m[1][1]-sol[1]*m[0][1])/det;
        L[1]=(sol[1]*m[0][0]-sol[0]*m[1][0])/det;
        res.x=outer[0].x+q[0].x*L[0]+q[1].x*L[1];
        res.y=outer[0].y+q[0].y*L[0]+q[1].y*L[1];
        res.z=outer[0].z+q[0].z*L[0]+q[1].z*L[1];
        radius=dist(res, outer[0]);
        break;
    case 4:
        for (i=0; i<3; ++i) {
          q[i].x=outer[i+1].x-outer[0].x;
          q[i].y=outer[i+1].y-outer[0].y;
          q[i].z=outer[i+1].z-outer[0].z;
          sol[i]=dot(q[i], q[i]);
        }
        for (i=0;i<3;++i)
          for(j=0;j<3;++j) m[i][j]=dot(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)<eps ) 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]=dot(q[i], q[j])*2;
      }
      res=outer[0];
      for (i=0; i<3; ++i ) {
        res.x += q[i].x * L[i];
        res.y += q[i].y * L[i];
        res.z += q[i].z * L[i];
      }
      radius=dist(res, outer[0]);
  }
}
void minball(int n) {
  ball();
  if ( nouter<4 )
    for (int i=0; i<n; ++i)
      if (dist(res, pt[i])-radius>eps) {
        outer[nouter]=pt[i];
        ++nouter;
        minball(i);
        --nouter;
        if (i>0) {
          Tpoint Tt = pt[i];
          memmove(&pt[1], &pt[0], sizeof(Tpoint)*i);
          pt[0]=Tt;
        }
      }
}
void solve() {
  for (int i=0;i<npoint;i++)
    scanf("%lf%lf%lf",&pt[i].x,&pt[i].y,&pt[i].z);
  random_shuffle(pt, pt + npoint);
  radius=-1;
  for (int i=0;i<npoint;i++){
    if (dist(res,pt[i])-radius>eps){
      nouter=1;
      outer[0]=pt[i];
      minball(i);
    }
  }
  printf("%.5f\n",sqrt(radius));
}
int main(){
  for( ; cin >> npoint && npoint; )
    solve();
  return 0;
}
```

4. 字符串

4.1 AC 自动机

```
int newnode() {
  ++tot;
  memset(ch[tot], 0, sizeof(ch[tot]));
  fail[tot] = dep[tot] = par[tot] = 0;
  return tot;
}
void insert(char *s,int x) {
  if(*s == '\0') return;
  else {
    int &y = ch[x][*s - 'a'];
    if(y == 0) y = newnode(), par[y] = x, dep[y] = dep[x]
      + 1;
    insert(s + 1, y);
  }
}
void build() {
  int line[maxn];
  int f = 0, r = 0;
  fail[root] = root;
  for(int i = 0; i < alpha; i++) {
    if(ch[root][i]) fail[ch[root][i]] = root, line[r++] =
      ch[root][i];
    else ch[root][i] = root;
  }
  while(f != r) {
    int x = line[f++];
    for(int i = 0; i < alpha; i++) {
      if(ch[x][i]) fail[ch[x][i]] = ch[fail[x]][i],
        line[r++] = ch[x][i];
      else ch[x][i] = ch[fail[x]][i];
    }
  }
}
```

4.2 后缀数组

```
const int MAXN = MAXL * 2 + 1;
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 = n; i; --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; --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]];
  }
}
```

4.3 后缀自动机

```
static const int MAXL = MAXN * 2;  // MAXN is original
  length
static const int alphabet = 26;  // sometimes need
  changing
int l, last, cnt, trans[MAXL][alphabet], par[MAXL],
  sum[MAXL], seq[MAXL], mxl[MAXL], size[MAXL];  // mxl
  is maxlength, size is the size of right
char str[MAXL];
inline void init() {
  l = strlen(str + 1); cnt = last = 1;
  for (int i = 0; i <= l * 2; ++i) memset(trans[i], 0,
    sizeof(trans[i]));
```

```cpp
  memset(par, 0, sizeof(*par) * (l * 2 + 1));
  memset(mxl, 0, sizeof(*mxl) * (l * 2 + 1));
  memset(size, 0, sizeof(*size) * (l * 2 + 1));
}
inline void extend(int pos, int c) {
  int p = last, np = last = ++cnt;
  mxl[np] = mxl[p] + 1; size[np] = 1;
  for (; p && !trans[p][c]; p = par[p]) trans[p][c] = np;
  if (!p) par[np] = 1;
  else {
    int q = trans[p][c];
    if (mxl[p] + 1 == mxl[q]) par[np] = q;
    else {
      int nq = ++cnt;
      mxl[nq] = mxl[p] + 1;
      memcpy(trans[nq], trans[q], sizeof(trans[nq]));
      par[nq] = par[q];
      par[np] = par[q] = nq;
      for (; trans[p][c] == q; p = par[p]) trans[p][c] = nq;
    }
  }
}
inline void buildsam() {
  for (int i = 1; i <= l; ++i) extend(i, str[i] - 'a');
  memset(sum, 0, sizeof(*sum) * (l * 2 + 1));
  for (int i = 1; i <= cnt; ++i) sum[mxl[i]]++;
  for (int i = 1; i <= l; ++i) sum[i] += sum[i - 1];
  for (int i = cnt; i; --i) seq[sum[mxl[i]]--] = i;
  for (int i = cnt; i; --i) size[par[seq[i]]] += size[seq[i]];
}
```

4.4 广义后缀自动机

```cpp
inline void add_node(int x, int &last) {
  int lastnode = last;
  if (c[lastnode][x]) {
    int nownode = c[lastnode][x];
    if (l[nownode] == l[lastnode] + 1) last = nownode;
    else {
      int auxnode = ++cnt; l[auxnode] = l[lastnode] + 1;
      for (int i = 0; i < alphabet; ++i) c[auxnode][i] = c[nownode][i];
      par[auxnode] = par[nownode]; par[nownode] = auxnode;
      for (; lastnode && c[lastnode][x] == nownode; lastnode = par[lastnode]) {
        c[lastnode][x] = auxnode;
      }
      last = auxnode;
    }
  } else {
    int newnode = ++cnt; l[newnode] = l[lastnode] + 1;
    for (; lastnode && !c[lastnode][x]; lastnode = par[lastnode]) c[lastnode][x] = newnode;
    if (!lastnode) par[newnode] = 1;
    else {
      int nownode = c[lastnode][x];
      if (l[lastnode] + 1 == l[nownode]) par[newnode] = nownode;
      else {
        int auxnode = ++cnt; l[auxnode] = l[lastnode] + 1;
        for (int i = 0; i < alphabet; ++i) c[auxnode][i] = c[nownode][i];
        par[auxnode] = par[nownode]; par[nownode] = par[newnode] = auxnode;
        for (; lastnode && c[lastnode][x] == nownode; lastnode = par[lastnode]) {
          c[lastnode][x] = auxnode;
        }
      }
    }
    last = newnode;
  }
}
```

4.5 manacher

```cpp
void Manacher(std::string s,int p[]) {
    std::string t = "$#";
    for (int i = 0; i < s.size(); i++) {
        t += s[i];
        t += "#";
    }
    int mx = 0, id = 0;
    for (int i = 1; i < t.size(); i++) {
        p[i] = mx > i ? min(p[2 * id - i], mx - i) : 1;
        while (t[i + p[i]] == t[i - p[i]]) ++p[i];
        if (mx < i + p[i]) {
            mx = i + p[i];
            id = i;
        }
    }
}
```

4.6 回文自动机

```cpp
int nT, nStr, last, c[MAXT][26], fail[MAXT], r[MAXN], l[MAXN], s[MAXN];
int allocate(int len) {
  l[nT] = len;
  r[nT] = 0;
  fail[nT] = 0;
  memset(c[nT], 0, sizeof(c[nT]));
  return nT++;
}
void init() {
  nT = nStr = 0;
  int newE = allocate(0);
  int newO = allocate(-1);
  last = newE;
  fail[newE] = newO;
  fail[newO] = newE;
  s[0] = -1;
}
void add(int x) {
  s[++nStr] = x;
  int now = last;
  while (s[nStr - l[now] - 1] != s[nStr]) now = fail[now];
  if (!c[now][x]) {
    int newnode = allocate(l[now] + 2), &newfail = fail[newnode];
    newfail = fail[now];
    while (s[nStr - l[newfail] - 1] != s[nStr]) newfail = fail[newfail];
    newfail = c[newfail][x];
    c[now][x] = newnode;
  }
  last = c[now][x];
  r[last]++;
}
void count() {
  for (int i = nT - 1; i >= 0; i--) {
    r[fail[i]] += r[i];
  }
}
```

4.7 循环串的最小表示

```cpp
std::string find(std::string s) {
  int i, j, k, l, n = s.length(); s += s;
  for(i = 0, j = 1; j < n;) {
    for (k = 0; k < n && s[i + k] == s[j + k]; k++);
```

```
    if (k >= n) break;
    if (s[i + k] < s[j + k]) j += k + 1; // 如果求最大表示，换成 '>'
    else l = i + k, i = j, j = max(l, j) + 1;
  }
  return s.substr(i, n);
}
```

5. 数据结构

5.1 可并堆

```
int merge(int x,int y) {
//p[i] 结点 i 的权值，这里是维护大根堆
//d[i] 在 i 的子树中，i 到右叶子结点的最远距离.
    if(!x) return y;
    if(!y) return x;
    if(p[x] < p[y]) std::swap(x, y);
    r[x] = merge(r[x], y);
    if(r[x]) fa[r[x]] = x;
    if(d[l[x]] < d[r[x]]) std::swap(l[x], r[x]);//调整树的结构，使其满足左偏性质
    d[x] = d[r[x]] + 1;
    return x;
}
```

5.2 KD-Tree

```
long long norm(const long long &x) {
    return std::abs(x);
    return x * x;
}
struct Point {
    int x, y, id;
    const int& operator [] (int index) const {
        if (index == 0) {
            return x;
        } else {
            return y;
        }
    }
    friend long long dist(const Point &a, const Point &b) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            result += norm(a[i] - b[i]);
        }
        return result;
    }
} point[N];
struct Rectangle {
    int min[2], max[2];
    Rectangle() {
        min[0] = min[1] = INT_MAX;  // sometimes int is not enough
        max[0] = max[1] = INT_MIN;
    }
    void add(const Point &p) {
        for (int i = 0; i < 2; ++i) {
            min[i] = std::min(min[i], p[i]);
            max[i] = std::max(max[i], p[i]);
        }
    }
    long long dist(const Point &p) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            result += norm(std::min(std::max(p[i], min[i]), max[i]) - p[i]);
            result += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
        }
        return result;
    }
};
struct Node {
    Point seperator;
    Rectangle rectangle;
    int child[2];
    void reset(const Point &p) {
        seperator = p;
        rectangle = Rectangle();
        rectangle.add(p);
        child[0] = child[1] = 0;
    }
} tree[N << 1];
int size, pivot;
bool compare(const Point &a, const Point &b) {
    if (a[pivot] != b[pivot]) {
        return a[pivot] < b[pivot];
    }
    return a.id < b.id;
}
// 左闭右開: build(1, n + 1)
int build(int l, int r, int type = 1) {
    pivot = type;
    if (l >= r) {
        return 0;
    }
    int x = ++size;
    int mid = l + r >> 1;
    std::nth_element(point + l, point + mid, point + r, compare);
    tree[x].reset(point[mid]);
    for (int i = l; i < r; ++i) {
        tree[x].rectangle.add(point[i]);
    }
    tree[x].child[0] = build(l, mid, type ^ 1);
    tree[x].child[1] = build(mid + 1, r, type ^ 1);
    return x;
}
int insert(int x, const Point &p, int type = 1) {
    pivot = type;
    if (x == 0) {
        tree[++size].reset(p);
        return size;
    }
    tree[x].rectangle.add(p);
    if (compare(p, tree[x].seperator)) {
        tree[x].child[0] = insert(tree[x].child[0], p, type ^ 1);
    } else {
        tree[x].child[1] = insert(tree[x].child[1], p, type ^ 1);
    }
    return x;
}
// For minimum distance
// For maximum: 下面递归 query 时 0, 1 换顺序;< and >;min and max
void query(int x, const Point &p, std::pair<long long, int> &answer, int type = 1) {
    pivot = type;
    if (x == 0 || tree[x].rectangle.dist(p) > answer.first) {
        return;
    }
    answer = std::min(answer,
             std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, answer, type ^ 1);
        query(tree[x].child[1], p, answer, type ^ 1);
    } else {
        query(tree[x].child[1], p, answer, type ^ 1);
```

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

5.3 Treap

```
struct Node{
  int mn, key, size, tag;
  bool rev;
  Node* ch[2];
  Node(int mn, int key, int size): mn(mn), key(key), size(size), rev(0), tag(0){}
  void downtag();
  Node* update(){
    mn = min(ch[0] -> mn, min(key, ch[1] -> mn));
    size = ch[0] -> size + 1 + ch[1] -> size;
    return this;
  }
};
typedef pair<Node*, Node*> Pair;
Node *null, *root;
void Node::downtag(){
  if(rev){
    for(int i = 0; i < 2; i++)
      if(ch[i] != null){
        ch[i] -> rev ^= 1;
        swap(ch[i] -> ch[0], ch[i] -> ch[1]);
      }
    rev = 0;
  }
  if(tag){
    for(int i = 0; i < 2; i++)
      if(ch[i] != null){
        ch[i] -> key += tag;
        ch[i] -> mn += tag;
        ch[i] -> tag += tag;
      }
    tag = 0;
  }
}
int r(){
  static int s = 3023192386;
  return (s += (s << 3) + 1) & (~0u >> 1);
}
bool random(int x, int y){
  return r() % (x + y) < x;
}
Node* merge(Node *p, Node *q){
  if(p == null) return q;
  if(q == null) return p;
  p -> downtag();
  q -> downtag();
  if(random(p -> size, q -> size)){
    p -> ch[1] = merge(p -> ch[1], q);
    return p -> update();
  }else{
    q -> ch[0] = merge(p, q -> ch[0]);
    return q -> update();
  }
}
Pair split(Node *x, int n){
  if(x == null) return make_pair(null, null);
  x -> downtag();
  if(n <= x -> ch[0] -> size){
    Pair ret = split(x -> ch[0], n);
    x -> ch[0] = ret.second;
    return make_pair(ret.first, x -> update());
  }
  Pair ret = split(x -> ch[1], n - x -> ch[0] -> size - 1);
  x -> ch[1] = ret.first;
  return make_pair(x -> update(), ret.second);
}
pair<Node*, Pair> get_segment(int l, int r){
  Pair ret = split(root, l - 1);
  return make_pair(ret.first, split(ret.second, r - l + 1));
}
int main(){
  null = new Node(INF, INF, 0);
  null -> ch[0] = null -> ch[1] = null;
  root = null;
}
```

5.4 Splay

```
template<class T>void checkmin(T &x,T y) { if(y < x) x = y; }
struct Node {
  Node *c[2], *fa;
  int size, rev; LL val, add, min;

  Node *init(LL v) {
    val = min = v, add = rev = 0;
    c[0] = c[1] = fa = NULL, size = 1;
    return this;
  }
  void rvs() { std::swap(c[0], c[1]), rev ^= 1; }
  void inc(LL x) { val += x, add += x, min += x; }
  void pushdown() {
    if(rev) {
      if(c[0]) c[0]->rvs();
      if(c[1]) c[1]->rvs();
      rev = 0;
    }
    if(add) {
      if(c[0]) c[0]->inc(add);
      if(c[1]) c[1]->inc(add);
      add = 0;
    }
  }
  void update() {
    min = val, size = 1;
    if(c[0]) checkmin(min, c[0]->min), size += c[0]->size;
    if(c[1]) checkmin(min, c[1]->min), size += c[1]->size;
  }

} *root;
Node* newnode(LL x) {
  static Node pool[maxs], *p = pool;
  return (++p)->init(x);
}
void setc(Node *x,int t,Node *y) {
  x->c[t] = y;
  if(y) y->fa = x;
```

```cpp
}
Node *find(int k) {
  Node *now = root;
  while(true) {
    now->pushdown();
    int t = (now->c[0] ? now->c[0]->size : 0) + 1;
    if(t == k) break;
    if(t > k) now = now->c[0];
    else now = now->c[1], k -= t;
  }
  return now;
}
void rotate(Node *x,Node* &k) {
  Node *y = x->fa, *z = y->fa;
  if(y != k) z->c[z->c[1] == y] = x;
  else k = x;
  x->fa = z;
  int i = (y->c[1] == x);
  setc(y, i, x->c[i ^ 1]);
  setc(x, i ^ 1, y);
  y->update(), x->update();
}
void spaly(Node *x,Node* &k) {
  static Node *st[maxs], *y, *z;
  int top = 0;
  y = x;
  while(y != k) st[++top] = y, y = y->fa;
  st[++top] = y;
  while(top) st[top]->pushdown(), top--;
  while(x != k) {
    y = x->fa, z = y->fa;
    if(y != k) {
      if((y == z->c[1]) ^ (x == y->c[1])) rotate(x, k);
      else rotate(y, k);
    }
    rotate(x, k);
  }
}
Node *subtree(int l,int r) {
  assert((++l) <= (++r));
  spaly(find(l - 1), root), spaly(find(r + 1),
    ↪ root->c[1]);
  return root->c[1]->c[0];
}
void ins(int pos,int v) {
  pos++;
  spaly(find(pos), root), spaly(find(pos + 1),
    ↪ root->c[1]);
  setc(root->c[1], 0, newnode(v));
  root->c[1]->update(), root->update();
}
void del(int pos) {
  pos++;
  spaly(find(pos - 1), root), spaly(find(pos + 1),
    ↪ root->c[1]);
  root->c[1]->c[0] = NULL, root->c[1]->update(),
    ↪ root->update();
}
void init() {
  root = newnode(0);
  setc(root, 1, newnode(0));
  root->update();
}
```

5.5 Link cut Tree

```cpp
inline void reverse(int x) {
  tr[x].rev ^= 1; swap(tr[x].c[0], tr[x].c[1]);
}
inline void rotate(int x, int k) {
  int y = tr[x].fa, z = tr[y].fa;
    tr[x].fa = z; tr[z].c[tr[z].c[1] == y] = x;
    tr[tr[x].c[k ^ 1]].fa = y; tr[y].c[k] = tr[x].c[k ^
      ↪ 1];
    tr[x].c[k ^ 1] = y; tr[y].fa = x;
}
inline void splay(int x, int w) {
  int z = x; pushdown(x);
  while (tr[x].fa != w) {
    int y = tr[x].fa; z = tr[y].fa;
    if (z == w) {
      pushdown(z = y); pushdown(x);
      rotate(x, tr[y].c[1] == x);
      update(y); update(x);
    } else {
      pushdown(z); pushdown(y); pushdown(x);
      int t1 = tr[y].c[1] == x, t2 = tr[z].c[1] == y;
      if (t1 == t2) rotate(y, t2), rotate(x, t1);
      else rotate(x, t1), rotate(x, t2);
      update(z); update(y); update(x);
    }
  }
  update(x);
  if (x != z) par[x] = par[z], par[z] = 0;
}
inline void access(int x) {
  for (int y = 0; x; y = x, x = par[x]) {
    splay(x, 0);
    if (tr[x].c[1]) par[tr[x].c[1]] = x, tr[tr[x].c[1]].fa
      ↪ = 0;
    tr[x].c[1] = y; par[y] = 0; tr[y].fa = x; update(x);
  }
}
inline void makeroot(int x) {
  access(x); splay(x, 0); reverse(x);
}
inline void link(int x, int y) {
  makeroot(x); par[x] = y;
}
inline void cut(int x, int y) {
  access(x); splay(y, 0);
  if (par[y] != x) swap(x, y), access(x), splay(y, 0);
  par[y] = 0;
}
inline void split(int x, int y) {  // x will be the root
  ↪ of the tree
  makeroot(y); access(x); splay(x, 0);
}
```

5.6 树上莫队

```cpp
void dfs(int u) {
  dep[u] = dep[fa[u][0]] + 1, stk.push(u);
  for(int i = 1; i < logn; i++) fa[u][i] = fa[fa[u][i -
    ↪ 1]][i - 1];
  for(int i = 0; i < vec[u].size(); i++) {
    int v = vec[u][i];
    if(v == fa[u][0]) continue;
    fa[v][0] = u, dfs(v), size[u] += size[v];
    if(size[u] >= bufsize) {
      ++bcnt;
      while(stk.top() != u) {
        block[stk.top()] = bcnt;
        stk.pop();
      }
      size[u] = 0;
    }
  }
  size[u]++;
}
void prework() {
  dfs(1), ++bcnt;
  while(!stk.empty()) block[stk.top()] = bcnt, stk.pop();
}
void rev(int u) {
```

```
  now -= (cnt[val[u]] > 0);
  if(used[u]) cnt[val[u]]--, used[u] = false;
  else cnt[val[u]]++, used[u] = true;
  now += (cnt[val[u]] > 0);
}
void move(int &x,int y,int z) {
  int fwd = y;
  rev(getlca(x, z)), rev(getlca(y, z));
  while(x != y) {
    if(dep[x] < dep[y]) std::swap(x, y);
    rev(x), x = fa[x][0];
  }
  x = fwd;
}
void solve() {
  int L = 1, R = 1;
  std::sort(query + 1, query + m + 1);
  rev(1);
  for(int i = 1; i <= m; i++) {
    int l = query[i].u, r = query[i].v;
    move(L, l, R), move(R, r, L);
    ans[query[i].t] = now;
  }
}
```

5.7 CDQ 分治

```
struct Node {
  int x, y, z, idx;
  friend bool operator == (const Node &a,const Node &b) { return a.x == b.x && a.y == b.y && a.z == b.z; }
  friend bool operator < (const Node &a,const Node &b) { return a.y < b.y; }
} triple[maxn];
bool cmpx(const Node &a,const Node &b) {
  if(a.x != b.x) return a.x < b.x;
  if(a.y != b.y) return a.y < b.y;
  return a.z < b.z;
}
void solve(int l,int r) {
  if(l == r) return;
  int mid = (l + r) >> 1;
  solve(l, mid);
  static std::pair<Node,int> Lt[maxn], Rt[maxn];
  int Ls = 0, Rs = 0, pos = 1;
  for(int i = l; i <= mid; i++) Lt[++Ls] = std::make_pair(triple[i], i);
  for(int i = mid + 1; i <= r; i++) Rt[++Rs] = std::make_pair(triple[i], i);
  std::sort(Lt + 1, Lt + Ls + 1);
  std::sort(Rt + 1, Rt + Rs + 1);
  backup.clear();
  for(int i = 1; i <= Rs; i++) {
    while(pos <= Ls && !(Rt[i].first < Lt[pos].first)) {
      insert(Lt[pos].first.z, 1);
      pos++;
    }
    f[Rt[i].second] += query(Rt[i].first.z);
  }
  for(int i = 0; i < backup.size(); i++) pre[backup[i]] = 0;
  solve(mid + 1, r);
}
```

5.8 整体二分

```
void solve(int l,int r,std::vector<int> q) {
  if(l == r || q.empty()) {
    for(int i = 0; i < q.size(); i++) ans[q[i]] = l;
  }
  else {
    int mid = (l + r) >> 1;
    std::vector<int> qL, qR;
    backup.clear();
    for(int i = 1; i <= mid; i++) {
      Event e = event[i];
      if(e.l <= e.r) add(e.l, e.v), add(e.r + 1, -e.v);
      else add(1, e.v), add(e.r + 1, -e.v), add(e.l, e.v);
    }
    for(int i = 0; i < q.size(); i++) {
      LL val = 0;
      for(int j = 0; j < vec[q[i]].size(); j++) {
        val += count(vec[q[i]][j]);
        if(val >= p[q[i]]) break;
      }
      if(cnt[q[i]] + val >= p[q[i]]) qL.push_back(q[i]);
      else cnt[q[i]] += val, qR.push_back(q[i]);
    }
    for(int i = 0; i < backup.size(); i++) sum[backup[i]] = 0;
    solve(l, mid, qL), solve(mid + 1, r, qR);
  }
}
```

6. 图论

6.1 2-SAT tarjan

```
template<class TAT>void checkmin(TAT &x,TAT y) { if(y < x) x = y; }
void tarjan(int u) {
  dfn[u] = low[u] = ++dt, flag[u] = true, stk.push(u);
  for(int i = 0; i < vec[u].size(); i++) {
    int v = vec[u][i];
    if(!dfn[v]) tarjan(v), checkmin(low[u], low[v]);
    else if(flag[v]) checkmin(low[u], dfn[v]);
  }

  if(low[u] == dfn[u]) {
    ++bcnt;
    while(stk.top() != u) block[stk.top()] = bcnt, flag[stk.top()] = false, stk.pop();
    block[u] = bcnt, flag[u] = false, stk.pop();
  }
}
bool solve() {
  bool ans = true;
  for(int i = 1; i <= 2 * n; i++) if(!dfn[i]) tarjan(i);
  for(int i = 1; i <= n; i++) if(block[2 * i] == block[2 * i - 1]) { ans = false; break;  }
    return ans;
}
```

6.2 KM

```
struct KM {
  // Truly O(n^3)
  // 邻接矩阵，不能连的边设为 -INF，求最小权匹配时边权取负，但不能连的还是 -INF，使用时先对 1 -> n 调用 hungary() ，再 get_ans() 求值
  int w[N][N];
  int lx[N], ly[N], match[N], way[N], slack[N];
  bool used[N];
  void init() {
    for (int i = 1; i <= n; i++) {
      match[i] = 0;
      lx[i] = 0;
      ly[i] = 0;
      way[i] = 0;
    }
  }
  void hungary(int x) {
    match[0] = x;
```

```
    int j0 = 0;
    for (int j = 0; j <= n; j++) {
      slack[j] = INF;
      used[j] = false;
    }
    do {
      used[j0] = true;
      int i0 = match[j0], delta = INF, j1 = 0;
      for (int j = 1; j <= n; j++) {
        if (used[j] == false) {
          int cur = -w[i0][j] - lx[i0] - ly[j];
          if (cur < slack[j]) {
            slack[j] = cur;
            way[j] = j0;
          }
          if (slack[j] < delta) {
            delta = slack[j];
            j1 = j;
          }
        }
      }
      for (int j = 0; j <= n; j++) {
        if (used[j]) {
          lx[match[j]] += delta;
          ly[j] -= delta;
        }
        else slack[j] -= delta;
      }
      j0 = j1;
    } while (match[j0] != 0);
    do {
      int j1 = way[j0];
      match[j0] = match[j1];
      j0 = j1;
    } while (j0);
  }
  int get_ans() {
    int sum = 0;
    for(int i = 1; i <= n; i++) {
      if (w[match[i]][i] == -INF) ; // 无解
      if (match[i] > 0) sum += w[match[i]][i];
    }
    return sum;
  }
} km;
```

6.3 点双连通分量

```
void tarjan(int u,int f) {
  low[u] = dfn[u] = ++df;
  for(int i = head[u]; i ; i = edge[i].next) {
    int v = edge[i].v;
    if(v == f) continue;
    if(!dfn[v]) {
      stack[++top] = i, tarjan(v, u);
      low[u] = std::min(low[u], low[v]);
      if(low[v] >= dfn[u]) {
        ++tot;
        while(stack[top] != i) pbc[stack[top--]] = tot;
        pbc[stack[top--]] = tot;
      }
    }
    else if(dfn[v] < dfn[u]) {
      stack[++top] = i;
      low[u] = std::min(low[u], dfn[v]);
    }
  }
}
void work() {
  for(int i = 1; i <= n; i++) if(!dfn[i]) tarjan(i, 0);
  for(int i = 1; i <= m; i++) pbc[i * 2 - 1] = pbc[i * 2] = std::max(pbc[i * 2 - 1], pbc[i * 2]);
}
```

6.4 边双连通分量

```
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;
```

6.5 树上路径求交

```
bool cmp(int x,int y) {
  return dep[x] > dep[y];
}
std::pair<int,int> path_intersection(int a,int b,int c,int d) {
  std::vector<int> poi;
  poi.push_back(getlca(a, c));
  poi.push_back(getlca(a, d));
  poi.push_back(getlca(b, c));
```

```
  poi.push_back(getlca(b, d));
  std::sort(poi.begin(), poi.end(), cmp);
  return std::make_pair(poi[0], poi[1]);
}
```

6.6 最小树形图

```
const int MAXN,INF;// INF >= sum( W_ij )
int from[MAXN + 10][MAXN * 2 + 10],n,m,edge[MAXN + 10][MAXN * 2 + 10];
int sel[MAXN * 2 + 10],fa[MAXN * 2 + 10],vis[MAXN * 2 + 10];
int getfa(int x){if(x == fa[x]) return x; return fa[x] = getfa(fa[x]);}
void liuzhu(){ // 1-base: root is 1, answer = (sel[i], i) for i in [2..n]
  fa[1] = 1;
  for(int i = 2; i <= n; ++i){
    sel[i] = 1; fa[i] = i;
    for(int j = 1; j <= n; ++j) if(fa[j] != i)
      if(from[j][i] = i, edge[sel[i]][i] > edge[j][i]) sel[i] = j;
  }
  int limit = n;
  while(1){
    int prelimit = limit; memset(vis, 0, sizeof(vis)); vis[1] = 1;
    for(int i = 2; i <= prelimit; ++i) if(fa[i] == i && !vis[i]){
      int j = i; while(!vis[j]) vis[j] = i, j = getfa(sel[j]);
      if(j == 1 || vis[j] != i) continue; vector<int> C; int k = j;
      do C.push_back(k), k = getfa(sel[k]); while(k != j);
      ++limit;
      for(int i = 1; i <= n; ++i){
        edge[i][limit] = INF, from[i][limit] = limit;
      }
      fa[limit] = vis[limit] = limit;
      for(int i = 0; i < int(C.size()); ++i){
        int x = C[i], fa[x] = limit;
        for(int j = 1; j <= n; ++j)
          if(edge[j][x] != INF && edge[j][limit] > edge[j][x] - edge[sel[x]][x]){
            edge[j][limit] = edge[j][x] - edge[sel[x]][x];
            from[j][limit] = x;
          }
      }
      for(int j=1;j<=n;++j) if(getfa(j)==limit) edge[j][limit] = INF;
      sel[limit] = 1;
      for(int j = 1; j <= n; ++j)
        if(edge[sel[limit]][limit] > edge[j][limit]) sel[limit] = j;
    }
    if(prelimit == limit) break;
  }
  for(int i = limit; i > 1; --i) sel[from[sel[i]][i]] = sel[i];
}
```

6.7 带花树

```
vector<int> link[maxn];
int n,match[maxn],Queue[maxn],head,tail;
int pred[maxn],base[maxn],start,finish,newbase;
bool InQueue[maxn],InBlossom[maxn];
void push(int u){ Queue[tail++]=u;InQueue[u]=true; }
int pop(){ return Queue[head++]; }
int FindCommonAncestor(int u,int v){
  bool InPath[maxn];
  for(int i=0;i<n;i++) InPath[i]=0;
  while(true){ u=base[u];InPath[u]=true;if(u==start) break;u=pred[match[u]]; }
  while(true){ v=base[v];if(InPath[v]) break;v=pred[match[v]]; }
  return v;
}
void ResetTrace(int u){
  int v;
  while(base[u]!=newbase){
    v=match[u];
    InBlossom[base[u]]=InBlossom[base[v]]=true;
    u=pred[v];
    if(base[u]!=newbase) pred[u]=v;
  }
}
void BlossomContract(int u,int v){
  newbase=FindCommonAncestor(u,v);
  for (int i=0;i<n;i++)
  InBlossom[i]=0;
  ResetTrace(u);ResetTrace(v);
  if(base[u]!=newbase) pred[u]=v;
  if(base[v]!=newbase) pred[v]=u;
  for(int i=0;i<n;++i)
  if(InBlossom[base[i]]){
    base[i]=newbase;
    if(!InQueue[i]) push(i);
  }
}
bool FindAugmentingPath(int u){
  bool found=false;
  for(int i=0;i<n;++i) pred[i]=-1,base[i]=i;
  for (int i=0;i<n;i++) InQueue[i]=0;
  start=u;finish=-1; head=tail=0; push(start);
  while(head<tail){
    int u=pop();
    for(int i=link[u].size()-1;i>=0;i--){
      int v=link[u][i];
      if(base[u]!=base[v]&&match[u]!=v)
        if(v==start||(match[v]>=0&&pred[match[v]]>=0))
          BlossomContract(u,v);
        else if(pred[v]==-1){
          pred[v]=u;
          if(match[v]>=0) push(match[v]);
          else{ finish=v; return true; }
        }
    }
  }
  return found;
}
void AugmentPath(){
  int u=finish,v,w;
  while(u>=0){ v=pred[u];w=match[v];match[v]=u;match[u]=v;u=w; }
}
void FindMaxMatching(){
  for(int i=0;i<n;++i) match[i]=-1;
  for(int i=0;i<n;++i) if(match[i]==-1) if(FindAugmentingPath(i)) AugmentPath();
}
```

6.8 支配树

```
vector<int> prec[N], succ[N];
vector<int> ord;
int stamp, vis[N];
int num[N];
int fa[N];
void dfs(int u) {
  vis[u] = stamp;
  num[u] = ord.size();
  ord.push_back(u);
  for (int i = 0; i < (int)succ[u].size(); ++i) {
```

```
    int v = succ[u][i];
    if (vis[v] != stamp) {
      fa[v] = u;
      dfs(v);
    }
  }
}
int fs[N], mins[N], dom[N], sem[N];
int find(int u) {
  if (u != fs[u]) {
    int v = fs[u];
    fs[u] = find(fs[u]);
    if (mins[v] != -1 && num[sem[mins[v]]] <
      ↪ num[sem[mins[u]]]) {
      mins[u] = mins[v];
    }
  }
  return fs[u];
}
void merge(int u, int v) { fs[u] = v; }
vector<int> buf[N];
int buf2[N];
void mark(int source) {
  ord.clear();
  ++stamp;
  dfs(source);
  for (int i = 0; i < (int)ord.size(); ++i) {
    int u = ord[i];
    fs[u] = u, mins[u] = -1, buf2[u] = -1;
  }
  for (int i = (int)ord.size() - 1; i > 0; --i) {
    int u = ord[i], p = fa[u];
    sem[u] = p;
    for (int j = 0; j < (int)prec[u].size(); ++j) {
      int v = prec[u][j];
      if (use[v] != stamp) continue;
      if (num[v] > num[u]) {
        find(v); v = sem[mins[v]];
      }
      if (num[v] < num[sem[u]]) {
        sem[u] = v;
      }
    }
    buf[sem[u]].push_back(u);
    mins[u] = u;
    merge(u, p);
    while (buf[p].size()) {
      int v = buf[p].back();
      buf[p].pop_back();
      find(v);
      if (sem[v] == sem[mins[v]]) {
        dom[v] = sem[v];
      } else {
        buf2[v] = mins[v];
      }
    }
  }
  dom[ord[0]] = ord[0];
  for (int i = 0; i < (int)ord.size(); ++i) {
    int u = ord[i];
    if (~buf2[u]) {
      dom[u] = dom[buf2[u]];
    }
  }
}
```

6.9 随机最大权匹配

```
bool dfs(int i) {
    path[len++] = i;
    if (v[i]) return true;
    v[i] = true;
    for (int j = 0; j < k; ++j) {
        if (i != j && match[i] != j && !v[j]) {
            int kok = match[j];
            if (d[kok] < d[i] + w[i][j] - w[j][kok]) {
                d[kok] = d[i] + w[i][j] - w[j][kok];
                if (dfs(kok)) return true;
            }
        }
    }
    --len; v[i] = false;
    return false;
}
void solve() {
    for (int i = 0; i < k; ++i) p[i] = i, match[i] = i ^
      ↪ 1;
    int cnt = 0;
    for (;;) {
        len = 0;
        bool flag = false;
        memset(d, 0, sizeof(d));
        memset(v, 0, sizeof(v));
        for (int i = 0; i < k; ++i) {
            if (dfs(p[i])) {
                flag = true;
                int t = match[path[len - 1]], j = len - 2;
                while (path[j] != path[len - 1]) {
                    match[t] = path[j];
                    swap(t, match[path[j]]);
                    --j;
                }
                match[t] = path[j];
                match[path[j]] = t;
                break;
            }
        }
        if (!flag) {
            if (++cnt >= 3) break;
            random_shuffle(p, p+k);
        }
    }
}
```

6.10 无向图最小割

```
int cost[maxn][maxn],seq[maxn],len[maxn],n,m,pop,ans;
bool used[maxn];
void Init(){
  int i,j,a,b,c;
  for(i=0;i<n;i++) for(j=0;j<n;j++) cost[i][j]=0;
  for(i=0;i<m;i++){
    scanf("%d %d %d",&a,&b,&c); cost[a][b]+=c;
      ↪ cost[b][a]+=c;
  }
  pop=n; for(i=0;i<n;i++) seq[i]=i;
}
void Work(){
  ans=inf; int i,j,k,l,mm,sum,pk;
  while(pop > 1){
    for(i=1;i<pop;i++) used[seq[i]]=0; used[seq[0]]=1;
    for(i=1;i<pop;i++) len[seq[i]]=cost[seq[0]][seq[i]];
    pk=0; mm=-inf; k=-1;
    for(i=1;i<pop;i++) if(len[seq[i]] > mm){
      ↪ mm=len[seq[i]]; k=i; }
    for(i=1;i<pop;i++){
      used[seq[l=k]]=1;
      if(i==pop-2) pk=k;
      if(i==pop-1) break;
      mm=-inf;
      for(j=1;j<pop;j++) if(!used[seq[j]])
        if((len[seq[j]]+=cost[seq[l]][seq[j]]) > mm)
          mm=len[seq[j]], k=j;
    }
    sum=0;
```

```
    for(i=0;i<pop;i++) if(i != k) sum+=cost[seq[k]][seq[i]];
    ans=min(ans,sum);
    for(i=0;i<pop;i++)
      cost[seq[k]][seq[i]]=cost[seq[i]][seq[k]]+=cost[seq[pk]][seq[i]];
    seq[pk]=seq[--pop];
  }
  printf("%d\n",ans);
}
```

6.11 最大团搜索

```
const int N = 1000 + 7;
vector<vector<bool> > adj;
class MaxClique {
    const vector<vector<bool> > adj;
    const int n;
    vector<int> result, cur_res;
    vector<vector<int> > color_set;
    const double t_limit; // MAGIC
  int para, level;
  vector<pair<int, int> > steps;
public:
    class Vertex {
    public:
        int i, d;
        Vertex(int i, int d = 0) : i(i), d(d) {}
    };
    void reorder(vector<Vertex> &p) {
        for (auto &u : p) {
            u.d = 0;
            for (auto v : p) u.d += adj[v.i][u.i];
        }
        sort(p.begin(), p.end(), [&](const Vertex &a, const Vertex &b) { return a.d > b.d; } );
    }
    void init_color(vector<Vertex> &p) {
        int maxd = p[0].d;
        for (int i = 0; i < p.size(); i++) p[i].d = min(i, maxd) + 1;
    }
    bool bridge(const vector<int> &s, int x) {
        for (auto v : s) if (adj[v][x]) return true;
        return false;
    }
    void color_sort(vector<Vertex> &cur) {
        int totc = 0, ptr = 0, mink = max((int)result.size() - (int)cur_res.size(), 0);
        for (int i = 0; i < cur.size(); i++) {
            int x = cur[i].i, k = 0;
            while (k < totc && bridge(color_set[k], x)) k++;
            if (k == totc) color_set[totc++].clear();
            color_set[k].push_back(x);
            if (k < mink) cur[ptr++].i = x;
        }
        if (ptr) cur[ptr - 1].d = 0;
        for (int i = mink; i < totc; i ++) {
            for (auto v : color_set[i]) {
                cur[ptr++] = Vertex(v, i + 1);
            }
        }
    }
    void expand(vector<Vertex> &cur) {
    steps[level].second = steps[level].second - steps[level].first + steps[level - 1].first;
    steps[level].first = steps[level - 1].second;
        while (cur.size()) {
            if (cur_res.size() + cur.back().d <= result.size()) return ;
            int x = cur.back().i;
            cur_res.push_back(x); cur.pop_back();
            vector<Vertex> remain;
            for (auto v : cur) {
                if (adj[v.i][x]) remain.push_back(v.i);
            }
            if (remain.size() == 0) {
                if (cur_res.size() > result.size()) result = cur_res;
            } else {
        if (1. * steps[level].second / ++para < t_limit) reorder(remain);
                color_sort(remain);
        steps[level++].second++;
                expand(remain);
        level--;
            }
            cur_res.pop_back();
        }
    }
public:
    MaxClique(const vector<vector<bool> > &_adj, int n, double tt = 0.025) : adj(_adj), n(n), t_limit(tt) {
        result.clear();
        cur_res.clear();
        color_set.resize(n);
    steps.resize(n + 1);
    fill(steps.begin(), steps.end(), make_pair(0, 0));
    level = 1;
    para = 0;
    }
    vector<int> solve() {
        vector<Vertex> p;
        for (int i = 0; i < n; i++) p.push_back(Vertex(i));
        reorder(p);
        init_color(p);
        expand(p);
        return result;
    }
};
```

6.12 弦图判定

```
// n 必须是 2 的次幂
void fft(Complex a[], int n, int f) {
  for (int i = 0; i < n; ++i)
    if (R[i] < i) swap(a[i], a[R[i]]);
  for (int i = 1, h = 0; i < n; i <<= 1, h++) {
    Complex wn = Complex(cos(pi / i), f * sin(pi / i));
    Complex w = Complex(1, 0);
    for (int k = 0; k < i; ++k, w = w * wn) tmp[k] = w;
    for (int p = i << 1, j = 0; j < n; j += p) {
      for (int k = 0; k < i; ++k) {
        Complex x = a[j + k], y = a[j + k + i] * tmp[k];
        a[j + k] = x + y; a[j + k + i] = x - y;
      }
    }
  }
}
```

6.13 斯坦纳树

```
void SPFA(int *dist) {
    static int line[maxn + 5];
    static bool hash[maxn + 5];
    int f = 0, r = 0;
    for(int i = 1; i <= N; i++) if(dist[i] < inf) line[r] = i, hash[i] = true, r = (r + 1) % (N + 1);
    while(f != r) {
        int t = line[f]; hash[t] = false, f = (f + 1) % (N + 1);
```

```cpp
        for(int i = head[t]; i ; i = edge[i].next) {
            int v = edge[i].v, dt = dist[t] + edge[i].w;
            if(dt < dist[v]) {
                dist[v] = dt;
                if(!hash[v]) {
                    if(dist[v] < dist[line[f]]) f = (f + N) % (N + 1), line[f] = v;
                    else line[r] = v, r = (r + 1) % (N + 1);
                    hash[v] = true;
                }
            }
        }
    }
}
void solve()
{
    for(int i = 1; i <= S; i++) {
        for(int j = 1; j <= N; j++)
            for(int k = (i - 1) & i; k ; k = (k - 1) & i) G[i][j] = std::min(G[i][j], G[k][j] + G[k ^ i][j]);
        SPFA(G[i]);
    }
}
```

6.14 虚树

```cpp
bool cmp(const int lhs,const int rhs) { return dfn[lhs] < dfn[rhs]; }
void build()
{
  int top = 0;
  std::sort(h + 1, h + 1 + m, cmp);
  for (int i = 1; i <= m; i++) {
    if (!top) father[st[++top] = h[i]] = 0;
    else {
        int p = h[i], lca = LCA(h[i],st[top]);
        while(d[st[top]] > d[lca]) {
            if (d[st[top - 1]] <= d[lca]) father[st[top]] = lca;
            top--;
        }
        if (st[top] != lca) t[++tot] = lca, father[lca] = st[top], st[++top] = lca;
        father[p] = lca, st[++top] = p;
    }
  }
}
```

6.15 点分治

```cpp
template<class TAT>void checkmax(TAT &x,TAT y) { if(x < y) x = y; }
template<class TAT>void checkmin(TAT &x,TAT y) { if(y < x) x = y; }
void getsize(int u,int fa) {
  size[u] = 1, smax[u] = 0;
  for(int i = 0; i < G[u].size(); i++) {
    int v = G[u][i];
    if(v == fa || ban[v]) continue;
    getsize(v, u);
    size[u] += size[v];
    checkmax(smax[u], size[v]);
  }
}
int getroot(int u,int ts,int fa) {
  int res = u;
  checkmax(smax[u], ts - size[u]);
  for(int i = 0; i < G[u].size(); i++) {
    int v = G[u][i];
    if(v == fa || ban[v]) continue;
    int w = getroot(v, ts, u);
    if(smax[w] < smax[res]) res = w;
  }
  return res;
}
void solve() {
  static int line[maxn];
  static std::vector<int> vec;
  int f = 0, r = 0; line[r++] = 1;
  while(f != r) {
    int u = line[f++];
    getsize(u, 0), u = getroot(u, size[u], 0);
    ban[u] = true, vec.clear();
    for(int i = 0; i < G[u].size(); i++)
      if(!ban[G[u][i]]) vec.push_back(G[u][i]);
    /*do something you like...*/
    for(int i = 0; i < vec.size(); i++) line[r++] = vec[i];
  }
}
```

6.16 最小割最大流

```cpp
bool BFS() {
    for(int i = 1; i <= ind; i++) dep[i] = 0;
    dep[S] = 1, line.push(S);
    while(!line.empty()) {
        int now = line.front(); line.pop();
        for(int i = head[now], p; i ; i = edge[i].next)
            if(edge[i].cap && !dep[p = edge[i].v])
                dep[p] = dep[now] + 1, line.push(p);
    }
    if(dep[T]) {
        for(int i = 1; i <= ind; i++) cur[i] = head[i];
        return true;
    }
    else return false;
}
int DFS(int a,int flow) {
    if(a == T) return flow;
    int ret = 0;
    for(int &i = cur[a], p; i ; i = edge[i].next)
        if(dep[p = edge[i].v] == dep[a] + 1 && edge[i].cap) {
            int ff = DFS(p, std::min(flow, edge[i].cap));
            flow -= ff, edge[i].cap -= ff;
            ret += ff, edge[i ^ 1].cap += ff;
            if(!flow) break;
        }
  return ret;
}
int solve() {
    int totflow = 0;
    while(BFS()) {
        totflow += DFS(S, INF);
    }
    return totflow;
}
```

6.17 最小费用流

```cpp
bool SPFA() {
    static int line[maxv];
    static bool hash[maxv];
    register int f = 0, r = 0;
  for(int i = 1; i <= ind; i++) dist[i] = inf, from[i] = 0;
    dist[S] = 0, line[r] = S, r = (r + 1) % maxv, hash[S] = true;
    while(f != r) {
        int x = line[f];
```

```
        line[f] = 0, f = (f + 1) % maxv, hash[x] = false;
        for(int i = head[x]; i; i = edge[i].next)
            if(edge[i].cap) {
                int v = edge[i].v, w = dist[x] + edge[i].cost;
                if(w < dist[v]) {
                    dist[v] = w, from[v] = i;
                    if(!hash[v]) {
                        if(f != r && dist[v] <= dist[line[f]]) f = (f - 1 + maxv) % maxv, line[f] = v;
                        else line[r] = v, r = (r + 1) % maxv;
                        hash[v] = true;
                    }
                }
            }
    }
    return from[T];
}
int back(int x,int flow) {
  if(from[x]) {
    flow = back(edge[from[x] ^ 1].v, std::min(flow, edge[from[x]].cap));
    edge[from[x]].cap -= flow, edge[from[x] ^ 1].cap += flow;
  }
  return flow;
}
int solve() {
    int mincost = 0, maxflow = 0;
    while(SPFA()) {
        int flow = back(T, inf);
        mincost += dist[T] * flow, maxflow += flow;
    }
    return mincost;
}
```

6.18 zkw 费用流

```
int S, T, totFlow, totCost;
int dis[N], slack[N], visit[N];
int modlable () {
    int delta = INF;
    for (int i = 1; i <= T; i++) {
        if (!visit[i] && slack[i] < delta) delta = slack[i];
        slack[i] = INF;
    }
    if (delta == INF) return 1;
    for (int i = 1; i <= T; i++)
        if (visit[i]) dis[i] += delta;
    return 0;
}
int dfs (int x, int flow) {
    if (x == T) {
        totFlow += flow;
        totCost += flow * (dis[S] - dis[T]);
        return flow;
    }
    visit[x] = 1;
    int left = flow;
    for (int i = e.last[x]; ~i; i = e.succ[i])
        if (e.cap[i] > 0 && !visit[e.other[i]]) {
            int y = e.other[i];
            if (dis[y] + e.cost[i] == dis[x]) {
                int delta = dfs (y, min (left, e.cap[i]));
                e.cap[i] -= delta;
                e.cap[i ^ 1] += delta;
                left -= delta;
                if (!left) { visit[x] = 0; return flow; }
            } else {
                slack[y] = min (slack[y], dis[y] + e.cost[i] - dis[x]);
            }
        }
    return flow - left;
}
pair <int, int> minCost () {
    totFlow = 0; totCost = 0;
    fill (dis + 1, dis + T + 1, 0);
    do {
        do {
            fill (visit + 1, visit + T + 1, 0);
        } while (dfs (S, INF));
    } while (!modlable ());
    return make_pair (totFlow, totCost);
}
```

6.19 最小割树

```
int cnt,n,m,dis[N],last[N],a[N],tmp[N],ans[N][N],s,t,mark[N];
struct edge{int to,c,next;}e[N*200];
queue <int> q;
void addedge(int u,int v,int c) {
    e[++cnt].to=v;e[cnt].c=c;
  e[cnt].next=last[u];last[u]=cnt;
    e[++cnt].to=u;e[cnt].c=c;
  e[cnt].next=last[v];last[v]=cnt;
}
bool bfs() {
    memset(dis,0,sizeof(dis));
    dis[s]=2;
    while (!q.empty()) q.pop();
    q.push(s);
    while (!q.empty()) {
        int u=q.front();
        q.pop();
        for (int i=last[u];i;i=e[i].next)
            if (e[i].c&&!dis[e[i].to]) {
                dis[e[i].to]=dis[u]+1;
                if (e[i].to==t) return 1;
                q.push(e[i].to);
            }
    }
    return 0;
}
int dfs(int x,int maxf) {
    if (x==t||!maxf) return maxf;
    int ret=0;
    for (int i=last[x];i;i=e[i].next)
        if (e[i].c&&dis[e[i].to]==dis[x]+1) {
            int f=dfs(e[i].to,min(e[i].c,maxf-ret));
            e[i].c-=f;
            e[i^1].c+=f;
            ret+=f;
            if (ret==maxf) break;
        }
    if (!ret) dis[x]=0;
    return ret;
}
void dfs(int x) {
    mark[x]=1;
    for (int i=last[x];i;i=e[i].next)
        if (e[i].c&&!mark[e[i].to]) dfs(e[i].to);
}
void solve(int l,int r) {
    if (l==r) return;
    s=a[l];t=a[r];
    for (int i=2;i<=cnt;i+=2)
        e[i].c=e[i^1].c=(e[i].c+e[i^1].c)/2;
    int flow=0;
    while (bfs()) flow+=dfs(s,inf);
    memset(mark,0,sizeof(mark));
```

```
    dfs(s);
    for (int i=1;i<=n;i++)
        if (mark[i])
            for (int j=1;j<=n;j++)
                if (!mark[j])
            ans[i][j]=ans[j][i]=min(ans[i][j],flow);
    int i=l,j=r;
    for (int k=l;k<=r;k++)
        if (mark[a[k]]) tmp[i++]=a[k];
        else tmp[j--]=a[k];
    for (int k=l;k<=r;k++)
        a[k]=tmp[k];
    solve(l,i-1);
    solve(j+1,r);
}
```

6.20 上下界网络流建图

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

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

6.20.1 无源汇的上下界可行流

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

6.20.2 有源汇的上下界可行流

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

6.20.3 有源汇的上下界最大流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 ∞，下届为 x 的边。x 满足二分性质，找到最大的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最大流。
2. 从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边，变成无源汇的网络。按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 和超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S \to T$ 的最大流即可。

6.20.4 有源汇的上下界最小流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 x，下界为 0 的边。x 满足二分性质，找到最小的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最小流。
2. 按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，但是注意这一次不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0，所以 S^*，T^* 无影响，再直接求一次 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \to S$ 边上的流量即为原图的最小流，否则无解。

7. 其他

7.1 Dancing Links

7.1.1 精确覆盖

```
#pragma comment(linker, "/STACK:1024000000,1024000000")
int head,sz;
int U[maxn],D[maxn],L[maxn],R[maxn];
int H[maxn],ROW[maxn],C[maxn],S[maxn],O[maxn];
void remove(int c) {
    L[R[c]]=L[c];
    R[L[c]]=R[c];
    for(int i=D[c]; i!=c; i=D[i])
        for(int j=R[i]; j!=i; j=R[j]) {
            U[D[j]]=U[j];
            D[U[j]]=D[j];
            --S[C[j]];
        }
}
void resume(int c) {
    for(int i=U[c]; i!=c; i=U[i]) {
        for(int j=L[i]; j!=i; j=L[j]) {
            ++S[C[j]];
            U[D[j]]=j;
            D[U[j]]=j;
        }
    }
    L[R[c]]=c;
    R[L[c]]=c;
}
void init(int m) {
    head=0;//头指针为 0
    for(int i=0; i<=m; i++) {
        U[i]=i;D[i]=i;L[i]=i-1;R[i]=i+1;S[i]=0;
    }
    R[m]=0;L[0]=m;S[0]=INF+1;sz=m+1;
    memset(H,0,sizeof(H));
}
void insert(int i, int j) {
    if(H[i]) {
        L[sz] = L[H[i]];R[sz] = H[i];
      L[R[sz]] = sz;R[L[sz]] = sz;
    } else {
        L[sz] = sz;R[sz] = sz;
        H[i] = sz;
    }
    U[sz] = U[j];D[sz] = j;
    U[D[sz]] = sz;D[U[sz]] = sz;
    C[sz] = j;ROW[sz] = i;
    ++S[j];++sz;
}
bool dfs(int k,int len) {
    if(R[head]==head) return true;
    int s=INF,c;
    for (int t=R[head]; t!=head; t=R[t])
        if (S[t]<s) s=S[t],c=t;
    remove(c);
    for(int i=D[c]; i!=c; i=D[i]) {
        O[k]=ROW[i];
        for(int j=R[i]; j!=i; j=R[j])
      remove(C[j]);
        if(dfs(k+1,len)) return true;
        for(int j=L[i]; j!=i; j=L[j])
            resume(C[j]);
    }
    resume(c);
    return  false;
}
```

7.1.2 重复覆盖

```
int h() {
    int i,j,k,count=0;
    bool visit[N];
    memset(visit,0,sizeof(visit));
    for(i=R[0];i;i=R[i]) {
        if(visit[i]) continue;
        count++;
```

```
        visit[i]=1;
        for(j=D[i];j!=i;j=D[j]) {
            for(k=R[j];k!=j;k=R[k])
                visit[C[k]]=1;
        }
    }
    return count;
}
void Dance(int k) {
    int i,j,c,Min,ans;
    ans=h();
    if(k+ans>K || k+ans>=ak) return;
    if(!R[0]) {
        if(k<ak) ak=k;
        return;
    }
    for(Min=N,i=R[0];i;i=R[i])
        if(S[i]<Min) Min=S[i],c=i;
    for(i=D[c];i!=c;i=D[i]) {
        remove(i);
        for(j=R[i];j!=i;j=R[j])
            remove(j);
        Dance(k+1);
        for(j=L[i];j!=i;j=L[j])
            resume(j);
        resume(i);
    }
    return;
}
```

7.2 蔡勒公式

```
int zeller(int y,int m,int d) {
  if (m<=2) y--,m+=12; int c=y/100; y%=100;
  int w=((c>>2)-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
  if (w<0) w+=7; return(w);
}
```

7.3 五边形数定理

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

```
LL dp[N],fi[N];
LL five(LL x){ return (3*x*x-x)/2; }
void wbxs(){
    dp[0]=1;
    int t=1000; //其实可以等于 sqrt(N)
    for(int i=-t;i<=t;++i)
        fi[i+t]=five(i);    //Q
    for(int i=1;i<=100000;++i){
        int flag=1;
        for(int j=1;;++j){
            LL a=fi[j+t],b=fi[-j+t];
            if(a>i && b>i) break;
            if(a<=i) dp[i]=(dp[i]+dp[i-a]*flag+MOD)%MOD; //p
            if(b<=i) dp[i]=(dp[i]+dp[i-b]*flag+MOD)%MOD;
            flag*=-1;
        }
    }
}
```

7.4 凸包闵可夫斯基和

```
// cv[0..1] 为两个顺时针凸包，其中起点等于终点，求出的闵可夫斯基和不一定是严格凸包
int i[2] = {0, 0}, len[2] = {(int)cv[0].size() - 1, (int)cv[1].size() - 1};
vector<P> mnk;
mnk.push_back(cv[0][0] + cv[1][0]);
do {
  int d((cv[0][i[0] + 1] - cv[0][i[0]]) * (cv[1][i[1] + 1] - cv[1][i[1]]) >= 0);
  mnk.push_back(cv[d][i[d] + 1] - cv[d][i[d]] + mnk.back());
  i[d] = (i[d] + 1) % len[d];
} while(i[0] || i[1]);
```

8. 技巧

8.1 STL 归还空间

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

8.2 大整数取模

```
// 需要保证 x 和 y 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;
}
```

8.3 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;
    }
}
```

8.4 汇编技巧

```
O3优化
#define __ __attribute__ ((optimize("-O3")))
#define _ __ __inline __attribute__ ((__gnu_inline__, __always_inline__, __artificial__))

汇编开栈
#pragma comment(linker, "/STACK:256000000")

int __size = 256 << 20;
```

```
char* __p__ = (char *) malloc(__size__) + __size__;

int main() {
  __asm__("movl %0, %%esp\n" :: "r"(__p__));
  return 0;
}
```

9. 提示

9.1 线性规划转对偶

$$\begin{array}{l}\text{maximize } \mathbf{c}^T\mathbf{x}\\ \text{subject to } \mathbf{Ax}\le\mathbf{b},\mathbf{x}\ge 0\end{array} \Longleftrightarrow \begin{array}{l}\text{minimize } \mathbf{y}^T\mathbf{b}\\ \text{subject to } \mathbf{y}^T\mathbf{A}\ge\mathbf{c}^T,\mathbf{y}\ge 0\end{array}$$

9.2 NTT 素数及其原根

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

9.3 积分表

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

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

1. $\int \frac{dx}{\sqrt{(x-a)(b-x)}} = 2\arcsin\sqrt{\frac{x-a}{b-x}}+C \ (a<b)$
2.
$$\int \sqrt{(x-a)(b-x)}dx = \frac{2x-a-b}{4}\sqrt{(x-a)(b-x)} + \frac{(b-a)^2}{4}\arcsin\sqrt{\frac{x-a}{b-x}}+C,(a<b) \quad (1)$$

1. $\int \tan x dx = -\ln|\cos x|+C$
2. $\int \cot x dx = \ln|\sin x|+C$
3. $\int \sec x dx = \ln\left|\tan\left(\frac{\pi}{4}+\frac{x}{2}\right)\right|+C = \ln|\sec x+\tan x|+C$
4. $\int \csc x dx = \ln\left|\tan\frac{x}{2}\right|+C = \ln|\csc x-\cot x|+C$
5. $\int \sec^2 x dx = \tan x+C$
6. $\int \csc^2 x dx = -\cot x+C$
7. $\int \sec x\tan x dx = \sec x+C$
8. $\int \csc x\cot x dx = -\csc x+C$
9. $\int \sin^2 x dx = \frac{x}{2}-\frac{1}{4}\sin 2x+C$
10. $\int \cos^2 x dx = \frac{x}{2}+\frac{1}{4}\sin 2x+C$
11. $\int \sin^n x dx = -\frac{1}{n}\sin^{n-1}x\cos x+\frac{n-1}{n}\int\sin^{n-2}x dx$
12. $\int \cos^n x dx = \frac{1}{n}\cos^{n-1}x\sin x+\frac{n-1}{n}\int\cos^{n-2}x dx$
13. $\int \frac{dx}{\sin^n x} = -\frac{1}{n-1}\frac{\cos x}{\sin^{n-1}x}+\frac{n-2}{n-1}\int\frac{dx}{\sin^{n-2}x}$
14. $\int \frac{dx}{\cos^n x} = \frac{1}{n-1}\frac{\sin x}{\cos^{n-1}x}+\frac{n-2}{n-1}\int\frac{dx}{\cos^{n-2}x}$
15.
$$\begin{aligned}&\int\cos^m x\sin^n x dx\\ =&\frac{1}{m+n}\cos^{m-1}x\sin^{n+1}x+\frac{m-1}{m+n}\int\cos^{m-2}x\sin^n x dx\\ =&-\frac{1}{m+n}\cos^{m+1}x\sin^{n-1}x+\frac{n-1}{m+1}\int\cos^m x\sin^{n-2}x dx\end{aligned}$$
16. $\int\frac{dx}{a+b\sin x} = \begin{cases}\frac{2}{\sqrt{a^2-b^2}}\arctan\frac{a\tan\frac{x}{2}+b}{\sqrt{a^2-b^2}}+C & (a^2>b^2)\\ \frac{1}{\sqrt{b^2-a^2}}\ln\left|\frac{a\tan\frac{x}{2}+b-\sqrt{b^2-a^2}}{a\tan\frac{x}{2}+b+\sqrt{b^2-a^2}}\right|+C & (a^2<b^2)\end{cases}$
17. $\int\frac{dx}{a+b\cos x} = \begin{cases}\frac{2}{a+b}\sqrt{\frac{a+b}{a-b}}\arctan\left(\sqrt{\frac{a-b}{a+b}}\tan\frac{x}{2}\right)+C & (a^2>b\\ \frac{1}{a+b}\sqrt{\frac{a+b}{a-b}}\ln\left|\frac{\tan\frac{x}{2}+\sqrt{\frac{a+b}{b-a}}}{\tan\frac{x}{2}-\sqrt{\frac{a+b}{b-a}}}\right|+C & (a^2<b\end{cases}$
18. $\int\frac{dx}{a^2\cos^2x+b^2\sin^2x} = \frac{1}{ab}\arctan\left(\frac{b}{a}\tan x\right)+C$
19. $\int\frac{dx}{a^2\cos^2x-b^2\sin^2x} = \frac{1}{2ab}\ln\left|\frac{b\tan x+a}{b\tan x-a}\right|+C$
20. $\int x\sin ax dx = \frac{1}{a^2}\sin ax-\frac{1}{a}x\cos ax+C$
21. $\int x^2\sin ax dx = -\frac{1}{a}x^2\cos ax+\frac{2}{a^2}x\sin ax+\frac{2}{a^3}\cos ax+C$
22. $\int x\cos ax dx = \frac{1}{a^2}\cos ax+\frac{1}{a}x\sin ax+C$
23. $\int x^2\cos ax dx = \frac{1}{a}x^2\sin ax+\frac{2}{a^2}x\cos ax-\frac{2}{a^3}\sin ax+C$

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

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

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