Talisman

November 26, 2018

目录

1 计算几何

1.1 二维计算几何-wrz

```
#include<bits/stdc++.h>
using namespace std;
const double inf = 1e9;
const double eps = 1e-9;
const double pi = acos(-1.0);
bool le(double x, double y){return x < y - eps;} // x严格小于y
bool leq(double x, double y){return x < y + eps;} // x小于等于y
bool equ(double x, double y){return fabs(x - y) < eps;} // x等于y
double mysqrt(double x) {return x < eps ? 0 : sqrt(x);} // 开根号
double sqr(double x) {return x * x;} // 平方
struct point // 点或向量
{
    double x, y;
    double operator * (point that){return x*that.x + y*that.y;}
    double operator ^ (point that){return x*that.y - y*that.x;}
    point operator * (double t){return (point){x*t, y*t};}
    point operator / (double t){return (point){x/t, y/t};}
    point operator + (point that) {return (point){x + that.x, y + that.y};}
    point operator - (point that) {return (point){x - that.x, y - that.y};}
    double len(){return mysqrt(x*x+y*y);} // 到原点距离/向量长度
    point reset_len(double t) // 改变向量长度为t，t为正则方向不变，t为负则方向相反
    {double p = len();return (point){x*t/p, y*t/p};}
    point rot90() {return (point){-y, x};} // 逆时针旋转90度
    point rotate(double angle) // 使向量逆时针旋转angle弧度
    {double c = cos(angle), s = sin(angle);return (point){c * x  - s * y, s * x + c * y};}
};
struct line // 参数方程表示，p为线上一点，v为方向向量
{
    point p, v; // p为线上一点，v为方向向量
    double angle; // 半平面交用，用atan2计算，此时v的左侧为表示的半平面。注意有的函数声明一个新的line时没有初始化这个值!
    bool operator < (const line &that) const {return angle < that.angle;} // 半平面交用，按与x轴夹角排序
};
struct circle{point c; double r;};
double distance(point a, point b) // a，b两点距离
{return mysqrt(sqr(a.x - b.x) + sqr(a.y - b.y));}
circle make_circle(point a, point b) // 以a，b两点为直径作圆
{double d = distance(a, b);return (circle){(a+b)/2, d/2};}
double point_to_line(point a, line b) // 点a到直线b距离
{return fabs((b.v ^ (a - b.p)) / b.v.len());}
point project_to_line(point a, line b) // 点a到直线b的垂足/投影
{return b.v.reset_len((a - b.p) * b.v / b.v.len()) + b.p;}
vector<point> circle_inter(circle a, circle b) // 圆a和圆b的交点，需保证两圆不重合，圆的半径必须大于0
{
    double d = distance(a.c, b.c);
    vector<point> ret;
    if(le(a.r + b.r, d) || le(a.r + d, b.r) || le(b.r + d, a.r)) return vector<point>(); // 相离或内含
    point r = (b.c - a.c).reset_len(1);
    double x = ((sqr(a.r) - sqr(b.r)) / d + d) / 2;
    double h = mysqrt(sqr(a.r) - sqr(x));
    if(equ(h, 0)) return vector<point>({a.c + r * x}); // 内切或外切
    else return vector<point>({a.c + r*x + r.rot90()*h, a.c + r*x - r.rot90()*h}); // 相交两点
}
vector<point> line_circle_inter(line a, circle b) // 直线a和圆b的交点
{
    double d = point_to_line(b.c, a);
    if(le(b.r, d)) return vector<point>(); // 不交
    double x = mysqrt(sqr(b.r) - sqr(d));
    point p = project_to_line(b.c, a);
    if(equ(x, 0)) return vector<point> ({p}); // 相切
    else return vector<point> ({p + a.v.reset_len(x), p - a.v.reset_len(x)}); // 相交两点
}
point line_inter(line a, line b) // 直线a和直线b的交点，需保证两直线不平行
{double s1 = a.v ^ (b.p - a.p);double s2 = a.v ^ (b.p + b.v - a.p);return (b.p * s2 - (b.p + b.v) * s1) / (s2 - s1);}
vector<point> tangent(point p, circle a) // 过点p的圆a的切线的切点，圆的半径必须大于0
{circle c = make_circle(p, a.c);return circle_inter(a, c);}
vector<line> intangent(circle a, circle b) // 圆a和圆b的内公切线
{
    point p = (b.c * a.r + a.c * b.r) / (a.r + b.r);
    vector<point> va = tangent(p, a), vb = tangent(p, b);
    vector<line> ret;
    if(va.size() == 2 && vb.size() == 2){ret.push_back((line){va[0], vb[0] - va[0]});ret.push_back((line){va[1], vb[1] - va[1]});}
    else if(va.size() == 1 && vb.size() == 1){ret.push_back((line){p, (a.c - b.c).rot90()});}
    return ret;
}
// 判断半平面交是否有解，若有解需保证半平面交必须有界，可以通过外加四个大半平面解决
// lcnt为半平面数量，l为需要做的所有半平面的数组，p为存交点的临时数组，h为时刻更新的合法的半平面数组，下标均从1开始
bool HP(int lcnt, line *l, line *h, point *p)
{
    sort(l+1, l+1+lcnt);
    int head = 1, tail = 1;
    h[1] = l[1];
    for(int i = 2; i <= lcnt; i++)
    {
        line cur = l[i];
        for(; head < tail && le(cur.v ^ (p[tail-1]-cur.p), 0); tail--); // 先删队尾再删队头，顺序不能换
        for(; head < tail && le(cur.v ^ (p[head]-cur.p), 0); head++);
        h[++tail] = cur;
        if(equ(h[tail].v ^ h[tail-1].v, 0)) // 平行
        {
            if(le(h[tail].v * h[tail-1].v, 0)) return false; // 方向相反的平行直线，显然已经不可能围出有界半平面了
            tail--;
            if(le(h[tail+1].v ^ (h[tail].p - h[tail+1].p), 0)) h[tail] = h[tail+1];
        }
        if(head < tail) p[tail-1] = line_inter(h[tail-1], h[tail]);
    }
    for(; head < tail && le(h[head].v ^ (p[tail-1]-h[head].p), 0); tail--);
    return tail - head > 1;
}
double calc(double X){return 0;} // 计算给定X坐标上的覆盖的长度，配合辛普森积分使用
// 自适应辛普森积分，参数分别为(左端点x坐标，中点x坐标，右端点x坐标，左端点答案，中点答案，右端点答案)
// 改变计算深度应调整eps
double simpson(double l, double mid, double r, double fl, double fm, double fr)
{
    double lmid = (l+mid)/2, rmid = (r+mid)/2, flm = calc(lmid), frm = calc(rmid);
    double ans = (r-l) * (fl + 4*fm + fr), ansl = (mid-l) * (fl + 4*flm + fm), ansr = (r-mid) * (fm + 4*frm + fr);
    if(fabs(ansl + ansr - ans) < eps) return ans / 6;
    else return simpson(l,lmid,mid,fl,flm,fm) + simpson(mid,rmid,r,fm,frm,fr);
}
int main(){}
```

1.2 basis

```
struct Point {
    DB x, y;
    Point rotate(DB ang) const {return Point(cos(ang) * x - sin(ang) * y, cos(ang) * y + sin(ang) * x);} // 逆时针旋转 ang 弧度
    Point turn90() const {return Point(-y, x);} // 逆时针旋转 90 度
    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) { // 直线与直线交点
    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;
```

```cpp
    p = (l1.a * s2 + l1.b * s1) / (s1 + s2);
    return true;
}
bool onSeg(const Line& l, const Point& p) {  // 点在线段上
    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) {  // 点到直线距离
    return fabs(det(p - l.a, l.b - l.a) / (l.b - l.a).len());}
DB disToSeg(const Line& l, const Point& p) {  // 点到线段距离
    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返回两个点
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;
}
// 求圆到圆的外共切线，按关于 c1.o 的顺时针方向返回两条线
vector<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 的顺时针方向返回两条线
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;
}
```

1.3 圆交

```cpp
struct Event {
    Point p; double ang; int delta;
    Event (Point p = Point(0, 0), double ang = 0, double delta = 0) : p(p), ang(ang), delta(delta) {}
};
bool operator < (const Event &a, const Event &b) {
    return a.ang < b.ang;}
void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
    double d2 = (a.o - b.o).len2(),
           dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
           pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4));
    Point d = b.o - a.o, p = d.rotate(PI / 2),
          q0 = a.o + d * dRatio + p * pRatio,
          q1 = a.o + d * dRatio - p * pRatio;
    double ang0 = (q0 - a.o).ang(),
           ang1 = (q1 - a.o).ang();
    evt.push_back(Event(q1, ang1, 1));
    evt.push_back(Event(q0, ang0, -1));
    cnt += ang1 > ang0;
}
bool issame(const Circle &a, const Circle &b) { return sign((a.o - b.o).len()) == 0 && sign(a.r - b.r) == 0; }
bool overlap(const Circle &a, const Circle &b) {
    return sign(a.r - b.r - (a.o - b.o).len()) >= 0; }
bool intersect(const Circle &a, const Circle &b) {
    return sign((a.o - b.o).len() - a.r - b.r) < 0; }
Circle c[N];
double area[N];  // area[k] -> area of intersections >= k.
Point centroid[N];
bool keep[N];
void add(int cnt, DB a, Point c) {area[cnt] += a; centroid[cnt] = centroid[cnt] + c * a;}
void solve(int C) {
    for (int i = 1; i <= C; ++ i) {area[i] = 0; centroid[i] = Point(0, 0);}
    for (int i = 0; i < C; ++i) {
        int cnt = 1;
        vector<Event> evt;
        for (int j = 0; j < i; ++j) if (issame(c[i], c[j])) ++cnt;
        for (int j = 0; j < C; ++j) {
            if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])){++cnt;}
        }
        for (int j = 0; j < C; ++j) {
            if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i], c[j])) {addEvent(c[i], c[j], evt, cnt);}}
        if (evt.size() == 0u) { add(cnt, PI * c[i].r * c[i].r, c[i].o);}
        else {
            sort(evt.begin(), evt.end());
            evt.push_back(evt.front());
```

```
            for (int j = 0; j + 1 < (int)evt.size(); ++j) {
                cnt += evt[j].delta;
                add(cnt, det(evt[j].p, evt[j + 1].p) / 2, (evt[j].p + evt[j + 1].p) / 3);
                double ang = evt[j + 1].ang - evt[j].ang;
                if (ang < 0) { ang += PI * 2;}
                if (sign(ang) == 0) continue;
                add(cnt, ang * c[i].r * c[i].r / 2, c[i].o +
                    Point(sin(ang1) - sin(ang0), -cos(ang1) + cos(ang0)) * (2 / (3 * ang) * c[i].r));
                add(cnt, -sin(ang) * c[i].r * c[i].r / 2, (c[i].o + evt[j].p + evt[j + 1].p) / 3);
            }
        }
    }
    for (int i = 1; i <= C; ++ i)if (sign(area[i])) {centroid[i] = centroid[i] / area[i];}
}
```

1.4 圆的面积并

```
//n^2*logn
struct point {
    point rotate(const double &ang) {return point(cos(ang) * x - sin(ang) * y, cos(ang) * y + sin(ang) * x);}
    double ang() {return atan2(y, x);}
};
struct Circle {
    point o; double r;
    int tp; // 正圆为1 反向圆为-1
    Circle (point o = point(0, 0), double r = 0, int tp = 0) : o(o), r(r), tp(tp) {}
};
struct Event {
    point p; double ang; int delta;
    Event (point p = point(0, 0), double ang = 0, double delta = 0) : p(p), ang(ang), delta(delta) {}
};
bool operator < (const Event &a, const Event &b) {return a.ang < b.ang;}
void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
    double d2 = (a.o - b.o).len2(),
        dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
        pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r))) / d2 / 2;
    point d = b.o - a.o, p = d.rotate(PI / 2),
        q0 = a.o + d * dRatio + p * pRatio,
        q1 = a.o + d * dRatio - p * pRatio;
    double ang0 = (q0 - a.o).ang(),
        ang1 = (q1 - a.o).ang();
    evt.push_back(Event(q1, ang1, b.tp));
    evt.push_back(Event(q0, ang0, -b.tp));
    cnt += (ang1 > ang0) * b.tp;
}
bool issame(const Circle &a, const Circle &b) {return sign((a.o - b.o).len()) == 0 && sign(a.r - b.r) == 0;}
bool overlap(const Circle &a, const Circle &b) {return sign(a.r - b.r - (a.o - b.o).len()) >= 0;}
bool intersect(const Circle &a, const Circle &b) {return sign((a.o - b.o).len() - a.r - b.r) < 0;}
int C;
Circle c[N];
double area[N];
void solve() { // area[1]..area[C]
    memset(area, 0, sizeof(double) * (C + 1));
    for (int i = 0; i < C; ++i) {
        int cnt = (c[i].tp > 0);
        vector<Event> evt;
        for (int j = 0; j < i; ++j) if (issame(c[i], c[j])) cnt += c[j].tp;
        for (int j = 0; j < C; ++j)
            if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) cnt += c[j].tp;
        for (int j = 0; j < C; ++j)
            if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i], c[j]))
                addEvent(c[i], c[j], evt, cnt);
        if (evt.size() == 0) area[cnt] += c[i].tp * PI * c[i].r * c[i].r;
        else {
            sort(evt.begin(), evt.end());
            evt.push_back(evt.front());
            for (int j = 0; j + 1 < (int)evt.size(); ++j) {
                cnt += evt[j].delta;
                area[cnt] += c[i].tp * det(evt[j].p, evt[j + 1].p) / 2;
                double ang = evt[j + 1].ang - evt[j].ang;
                if (ang < 0) ang += PI * 2;
                area[cnt] += c[i].tp * (ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2);
            }
        }
    }
}
```

1.5 凸包

```
// 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;
    }
    // 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;
    }
};
```

1.6 三角形内心，外心，垂心

```
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);
}
```

1.7 费马点

Find a point P that minimizes $|PA| + |PB| + |PC|$.

```
point fermat_point (cp a, cp b, cp c) {
    if (a == b) return a; if (b == c) return b; if (c == a) return c;
    double ab = dis (a, b), bc = dis (b, c), ca = dis (c, a);
    double cosa = dot (b - a, c - a) / ab / ca;
    double cosb = dot (a - b, c - b) / ab / bc;
    double cosc = dot (b - c, a - c) / ca / bc;
    double sq3 = PI / 3.0; point mid;
    if (sgn (cosa + 0.5) < 0) mid = a;
    else if (sgn (cosb + 0.5) < 0) mid = b;
    else if (sgn (cosc + 0.5) < 0) mid = c;
    else if (sgn (det (b - a, c - a)) < 0) mid = line_intersect (line (a, b + (c - b).rot (sq3)), line (b, c + (a - c).rot (sq3)));
    else mid = line_intersect (line (a, c + (b - c).rot (sq3)), line (c, b + (a - b).rot (sq3)));
    return mid; }
```

1.8 多边形和圆的交

```
double sector_area (const point &a, const point &b, const double &r) {
    double c = (2.0 * r * r - (a - b).norm2 ()) / (2.0 * r * r);
    double al = acos (c);
    return r * r * al / 2.0; }
double area(const point &a,const point &b,const double &r) {
    double dA = dot (a, a), dB = dot (b, b), dC = point_to_segment (point (), line (a, b)), ans = 0.0;
    if (sgn (dA - r * r) <= 0 && sgn (dB - r * r) <= 0) return det (a, b) / 2.0;
    point tA = a.unit () * r;
    point tB = b.unit () * r;
    if (sgn (dC - r) > 0) return sector_area (tA, tB, r);
    std::pair <point, point> ret = line_circle_intersect (line (a, b), circle (point (), r));
    if (sgn (dA - r * r) > 0 && sgn (dB - r * r) > 0) {
        ans += sector_area (tA, ret.first, r);
        ans += det (ret.first, ret.second) / 2.0;
        ans += sector_area (ret.second, tB, r);
        return ans; }
    if (sgn (dA - r * r) > 0)
        return det (ret.first, b) / 2.0 + sector_area (tA, ret.first, r);
    else
        return det (a, ret.second) / 2.0 + sector_area (ret.second, tB, r); }
double solve(const std::vector<point> &p, const circle &c) {
    double ret = 0.0;
    for (int i = 0; i < (int) p.size (); ++i) {
        int s = sgn (det (p[i] - c.c, p[ (i + 1) % p.size ()] - c.c));
        if (s > 0)
            ret += area (p[i] - c.c, p[ (i + 1) % p.size ()] - c.c, c.r);
        else
            ret -= area (p[ (i + 1) % p.size ()] - c.c, p[i] - c.c, c.r); }
    return fabs (ret); }
```

1.9 点到凸包切线

```
typedef vector<vector<P>> Convex;
#define sz(x) ((int) x.size())
int lb(P x, const vector<P> & v, int le, int ri, int sg) {
    if (le > ri) le = ri;
    int s(le), t(ri);
    while (le != ri) {
        int mid((le + ri) / 2);
        if (sign(det(v[mid] - x, v[mid + 1] - v[mid])) == sg)
            le = mid + 1; else ri = mid; }
    return le; } // le 即为下标，按需返回
// v[0] 为顺时针上凸壳，v[1] 为顺时针下凸壳，均允许起始两个点横坐标相同
// 返回值为真代表严格在凸包外，顺时针旋转在 d1 方向先碰到凸包
bool getTan(P x, const Convex & v, int & d1, int & d2) {
    if (x.x < v[0][0].x) {
        d1 = lb(x, v[0], 0, sz(v[0]) - 1, 1);
        d2 = lb(x, v[1], 0, sz(v[1]) - 1, -1) + (int) v[0].size() - 1;
        return true;
    } else if(x.x > v[0].back().x) {
        d1 = lb(x, v[1], 0, sz(v[1]) - 1, 1) + (int) v[0].size() - 1;
        d2 = lb(x, v[0], 0, sz(v[0]) - 1, -1);
        return true;
    } else {
        for(int d(0); d < 2; d++) {
            int id(lower_bound(v[d].begin(), v[d].end(), x, [&](const P & a, const P & b) {
                return d == 0 ? a < b : b < a; }) - v[d].begin());
            if (id && (id == sz(v[d]) || det(v[d][id - 1] - x, v[d][id] - x) > 0)) {
                d1 = lb(x, v[d], id, sz(v[d]) - 1, 1);
                d2 = lb(x, v[d], 0, id, -1);
                if (d) {
                    d1 += (int) v[0].size() - 1;
                    d2 += (int) v[0].size() - 1; }
                return true; } } }
    return false; }
```

1.10 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<point> mnk;
mnk.push_back(cv[0][0] + cv[1][0]);
do {
    int d = (det(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]);
```

1.11 最近点对

```
double solve(point *p, int l, int r) { // 左闭右开 返回距离平方
    if(l + 1 >= r) return INF;
    int m = (l + r) / 2; double mx = p[m].x; vector <point> v;
    double ret = min(solve(p, l, m), solve(p, m, r));
    for(int i = l; i < r; i ++)
        if(sqr(p[i].x - mx) < ret) v.push_back(p[i]);
    sort(v.begin(), v.end(), by_y);
    for(int i = 0; i < v.size(); i ++)
        for(int j = i + 1; j < v.size(); j ++) {
            if(sqr(v[i].y - v[j].y) > ret) break;
            ret = min(ret, (v[i] - v[j]).len2()); }
    return ret; } // 需先对p[]按x进行排序
```

1.12 最小覆盖圆

```
circle minimum_circle (std::vector <point> p) { circle ret;
    std::random_shuffle (p.begin (), p.end ());
    for (int i = 0; i < (int) p.size (); ++i)
        if (!in_circle (p[i], ret)) { ret = circle (p[i], 0);
            for (int j = 0; j < i; ++j)
                if (!in_circle (p[j], ret)) {
                    ret = make_circle (p[j], p[i]);
                    for (int k = 0; k < j; ++k)
                        if (!in_circle (p[k], ret))
                            ret = make_circle(p[i],p[j],p[k]); } }
    return ret; }
```

1.13 最小覆盖球

```
vector<point3D> vec;
Circle calc() {
    if(vec.empty()) { return Circle(point3D(0, 0, 0), 0);
    }else if(1 == (int)vec.size()) {return Circle(vec[0], 0);
    }else if(2 == (int)vec.size()) {
        return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
    }else if(3 == (int)vec.size()) {
        double r = (vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] - vec[0]).len() / 2 / fabs(cross(vec[0] - vec[2], vec[1] - vec[2]).len());
        Plane ppp1 = Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0]));
        return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])), Plane(vec[2] - vec[1], 0.5 * (vec[2] + vec[1])), Plane(cross(vec[1] - vec[0], vec[2] - vec[0]), vec[0])), r);
    }else {
        point3D o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])), Plane(vec[2] - vec[0], 0.5 * (vec[2] + vec[0])), Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0]))));
        return Circle(o, (o - vec[0]).len()); } }
Circle miniBall(int n) {
    Circle res(calc());
    for(int i(0); i < n; i++) {
        if(!in_circle(a[i], res)) { vec.push_back(a[i]);
            res = miniBall(i); vec.pop_back();
            if(i) { point3D tmp(a[i]);
                memmove(a + 1, a, sizeof(point3D) * i);
                a[0] = tmp; } } }
    return res; }
int main() {
    int n; scanf("%d", &n);
    for(int i(0); i < n; i++) a[i].scan();
    sort(a, a + n); n = unique(a, a + n) - a;
    vec.clear(); random_shuffle(a, a + n);
    printf("%.10f\n", miniBall(n).r); }
```

1.14 阿波罗尼茨圆

硬币问题: 易知两两相切的圆半径为 r1, r2, r3, 求与他们都相切的圆的半径 r4 分母取负号, 答案再取绝对值, 为外切圆半径分母取正号为内切圆半径 // $r_4^{\pm} = \frac{r_1r_2r_3}{r_1r_2+r_1r_3+r_2r_3 \pm 2\sqrt{r_1r_2r_3(r_1+r_2+r_3)}}$

1.15 圆幂

圆幂: 半径为 R 的圆 O, 任意一点 P 到 O 的幂为 $h = OP^2 - R^2$
圆幂定理: 过 P 的直线交圆在 A 和 B 两点, 则 $PA \cdot PB = |h|$
根轴: 到两圆等幂点的轨迹是一条垂直于连心线的直线
反演: 已知一圆 C, 圆心为 O, 半径为 r, 如果 P 与 P' 在过圆心 O 的直线上, 且 $OP \cdot OP' = r^2$, 则称 P 与 P' 关于 O 互为反演. 一般 C 取单位圆.
反演的性质:
不过反演中心的直线反形是过反演中心的圆, 反之亦然.
不过反演中心的圆, 它的反形是一个不过反演中心的圆.
两条直线在交点 A 的夹角, 等于它们的反形在相应点 A' 的夹角, 但方向相反.
两个相交圆周在交点 A 的夹角等于它们的反形在相应点 A' 的夹角, 但方向相反.
直线和圆周在交点 A 的夹角等于它们的反演图形在相应点 A' 的夹角, 但方向相反.
正交圆反形也正交. 相切圆反形也相切, 当切点为反演中心时, 反形为两条平行线.

1.16 球面

球面距离: 连接球面两点的大圆劣弧 (所有曲线中最短)
球面角: 球面两个大圆弧所在半平面形成的二面角
球面凸多边形: 把一个球面多边形任意一边向两方无限延长成大圆, 其余边都在此大圆的同旁.
球面角盈 E: 球面凸 n 边形的内角和与 $(n-2)\pi$ 的差
离北极夹角 θ, 距离 h 的球冠: $S = 2\pi Rh = 2\pi R^2(1-\cos\theta)$, $V = \frac{\pi h^2}{3}(3R-h)$
球面凸 n 边形面积: $S = ER^2$

1.17 公式

1.17.1 Heron's Formula

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$
$$p = \frac{a+b+c}{2}$$

1.17.2 四面体内接球球心

假设 s_i 是第 i 个顶点相对面的面积, 则有

$$\begin{cases} x = \dfrac{s_1x_1 + s_2x_2 + s_3x_3 + s_4x_4}{s_1+s_2+s_3+s_4} \\ y = \dfrac{s_1y_1 + s_2y_2 + s_3y_3 + s_4y_4}{s_1+s_2+s_3+s_4} \\ z = \dfrac{s_1z_1 + s_2z_2 + s_3z_3 + s_4z_4}{s_1+s_2+s_3+s_4} \end{cases}$$

体积可以使用 1/6 混合积求, 内接球半径为

$$r = \frac{3V}{s_1+s_2+s_3+s_4}$$

1.17.3 三角形内心

$$\frac{a\vec{A} + b\vec{B} + c\vec{C}}{a+b+c}$$

1.17.4 三角形外心

$$\frac{\vec{A} + \vec{B} - \frac{\vec{BC}\cdot\vec{CA}}{\vec{AB}\times\vec{BC}}\vec{AB}^T}{2}$$

1.17.5 三角形垂心

$$\vec{H} = 3\vec{G} - 2\vec{O}$$

1.17.6 三角形偏心

$$\frac{-a\vec{A} + b\vec{B} + c\vec{C}}{-a+b+c}$$

剩余两点的同理.

1.17.7 三角形内接外接圆半径

$$r = \frac{2S}{a+b+c}, \ R = \frac{abc}{4S}$$

1.17.8 Pick's Theorem

$$S = I + \frac{B}{2} - 1$$

S is the area of lattice polygon, I is the number of lattice interior points, and B is the number of lattice boundary points.

1.17.9 Euler's Formula

For convex polyhedron: $V - E + F = 2$.
For planar graph: $|F| = |E| - |V| + n + 1$, n denotes the number of connected components.

1.18 三角公式

$$\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b$$
$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$
$$\tan(a \pm b) = \frac{\tan(a) \pm \tan(b)}{1 \mp \tan(a)\tan(b)}$$
$$\tan(a) \pm \tan(b) = \frac{\sin(a \pm b)}{\cos(a)\cos(b)}$$
$$\sin(a) + \sin(b) = 2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2})$$
$$\sin(a) - \sin(b) = 2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$$
$$\cos(a) + \cos(b) = 2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})$$
$$\cos(a) - \cos(b) = -2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$$

$\sin(na) = n\cos^{n-1} a \sin a - \binom{n}{3}\cos^{n-3} a \sin^3 a + \binom{n}{5}\cos^{n-5} a \sin^5 a - \ldots$

$$\cos(na) = \cos^n a - \binom{n}{2}\cos^{n-2} a \sin^2 a + \binom{n}{4}\cos^{n-4} a \sin^4 a - \ldots$$

1.18.1 超球坐标系

$$\begin{aligned} x_1 &= r\cos(\phi_1) \\ x_2 &= r\sin(\phi_1)\cos(\phi_2) \\ \cdots & \\ x_{n-1} &= r\sin(\phi_1)\cdots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\ x_n &= r\sin(\phi_1)\cdots\sin(\phi_{n-2})\sin(\phi_{n-1}) \\ \phi_{n-1} &\in [0, 2\pi] \\ \forall i = 1..n-1 \phi_i &\in [0, \pi] \end{aligned}$$

1.18.2 三维旋转公式

绕着 $(0,0,0) - (ux, uy, uz)$ 旋转 θ, (ux, uy, uz) 是单位向量

$$R = \begin{matrix} \cos\theta + u_x^2(1-\cos\theta) & u_xu_y(1-\cos\theta) - u_z\sin\theta & u_xu_z(1-\cos\theta) + u_y\sin\theta \\ u_yu_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_yu_z(1-\cos\theta) - u_x\sin\theta \\ u_zu_x(1-\cos\theta) - u_y\sin\theta & u_zu_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{matrix}.$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

1.18.3 立体角公式

$$\phi: \text{二面角}$$

$$\Omega = (\phi_{ab} + \phi_{bc} + \phi_{ac})\ \mathrm{rad} - \pi\ \mathrm{sr}$$

$$\tan\left(\frac{1}{2}\Omega/\mathrm{rad}\right) = \frac{\left|\vec{a}\ \vec{b}\ \vec{c}\right|}{abc + \left(\vec{a}\cdot\vec{b}\right)c + (\vec{a}\cdot\vec{c})\,b + \left(\vec{b}\cdot\vec{c}\right)a}$$

$$\theta_s = \frac{\theta_a + \theta_b + \theta_c}{2}$$

1.18.4 常用体积公式

- Pyramid $V = \frac{1}{3}Sh$.
- Sphere $V = \frac{4}{3}\pi R^3$.
- Frustum $V = \frac{1}{3}h(S_1 + \sqrt{S_1 S_2} + S_2)$.
- Ellipsoid $V = \frac{4}{3}\pi abc$.

1.18.5 高维球体积

$$V_2 = \pi R^2,\ S_2 = 2\pi R$$

$$V_3 = \frac{4}{3}\pi R^3,\ S_3 = 4\pi R^2$$

$$V_4 = \frac{1}{2}\pi^2 R^4,\ S_4 = 2\pi^2 R^3$$

$$\text{Generally, } V_n = \frac{2\pi}{n}V_{n-2},\ S_{n-1} = \frac{2\pi}{n-2}S_{n-3}$$

$$\text{Where, } S_0 = 2,\ V_1 = 2,\ S_1 = 2\pi,\ V_2 = \pi$$

1.19 三维计算几何

```
/* 大拇指指向x轴正方向时，4指弯曲由y轴正方向指向z轴正方向
   大拇指沿着原点到点(x, y, z)的向量，4指弯曲方向旋转w度 */
/* (x, y, z) * A = (x_new, y_new, z_new), 行向量右乘转移矩阵 */
void calc(D x, D y, D z, D w) { // 三维绕轴旋转
    w = w * pi / 180;
    memset(a, 0, sizeof(a));
    s1 = x * x + y * y + z * z;
    a[0][0] = ((y*y+z*z)*cos(w)+x*x)/s1; a[0][1] = x*y*(1-cos(w))/s1+z*sin(w)/sqrt(s1); a[0][2] = x*z*(1-cos(w))/s1-y*sin(w)/sqrt(s1);
    a[1][0] = x*y*(1-cos(w))/s1-z*sin(w)/sqrt(s1); a[1][1] = ((x*x+z*z)*cos(w)+y*y)/s1; a[1][2] = y*z*(1-cos(w))/s1+x*sin(w)/sqrt(s1);
    a[2][0] = x*z*(1-cos(w))/s1+y*sin(w)/sqrt(s1); a[2][1] = y*z*(1-cos(w))/s1-x*sin(w)/sqrt(s1); a[2][2] = ((x*x+y*y)*cos(w)+z*z)/s1;
}
point3D cross (const point3D & a, const point3D & b) {
    return point3D(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x); }
double mix(point3D a, point3D b, point3D c) {
    return dot(cross(a, b), c); }
struct Line { point3D s, t; };
struct Plane { // nor 为单位法向量，离原点距离 m
    point3D nor; double m;
    Plane(point3D r, point3D a) : nor(r){
        nor = 1 / r.len() * r;
        m = dot(nor, a); } };
// 以下函数注意除以0的情况
// 点到平面投影
point3D project_to_plane(point3D a, Plane b) {
return a + (b.m - dot(a, b.nor)) * b.nor; }
// 点到直线投影
point3D project_to_line(point3D a, Line b) {
return b.s + dot(a - b.s, b.t - b.s) / dot(b.t - b.s, b.t - b.s) * (b.t - b.s); }
// 直线与直线最近点
pair<point3D, point3D> closest_two_lines(Line x, Line y) {
double a = dot(x.t - x.s, x.t - x.s);
double b = dot(x.t - x.s, y.t - y.s);
double e = dot(y.t - y.s, y.t - y.s);
double d = a*e - b*b; point3D r = x.s - y.s;
double c = dot(x.t - x.s, r), f = dot(y.t - y.s, r);
double s = (b*f - c*e) / d, t = (a*f - c*b) / d;
return {x.s + s*(x.t - x.s), y.s + t*(y.t - y.s)}; }
// 直线与平面交点
point3D intersect(Plane a, Line b) {
double t = dot(a.nor, a.m * a.nor - b.s) / dot(a.nor, b.t - b.s);
return b.s + t * (b.t - b.s); }
// 平面与平面求交线
Line intersect(Plane a, Plane b) {
point3D d=cross(a.nor,b.nor), d2=cross(b.nor,d);
double t = dot(d2, a.nor);
point3D s = 1 / t * (a.m - dot(b.m * b.nor, a.nor)) * d2 + b.m * b.nor;
return (Line) {s, s + d}; }
// 三个平面求交点
point3D intersect(Plane a, Plane b, Plane c) {
return intersect(a, intersect(b, c));
point3D c1 (a.nor.x, b.nor.x, c.nor.x);
point3D c2 (a.nor.y, b.nor.y, c.nor.y);
point3D c3 (a.nor.z, b.nor.z, c.nor.z);
point3D c4 (a.m, b.m, c.m);
return 1 / mix(c1, c2, c3) * point3D(mix(c4, c2, c3), mix(c1, c4, c3), mix(c1, c2, c4)); }
```

1.20 三维凸包

```
__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);
}
```

2 数据结构

2.1 KD 树-wrz

```
// 这里写的是询问三维空间中的一个长方体内有没有点
int D;
struct point
{
    int d[3];
    bool operator < (const point &that) const {return d[D] < that.d[D];}
}p[N];
struct node
{
    int d[2][3]; // 0 min 1 max
}t[N<<2];
void pushup(int x)
{
    for(int i = 0; i < 3; i++)
    {
        t[x].d[0][i] = min(t[x<<1].d[0][i], t[x<<1|1].d[0][i]);
        t[x].d[1][i] = max(t[x<<1].d[1][i], t[x<<1|1].d[1][i]);
    }
```

```
}
void build(int x, int l, int r, int d)
{
    if(l == r)
    {
        for(int i = 0; i < 3; i++)
            t[x].d[0][i] = t[x].d[1][i] = p[l].d[i];
        return;
    }
    D = d; int mid = (l+r) >> 1;
    nth_element(p+l, p+mid, p+r+1);
    (++d) %= 3;
    build(x<<1, l, mid, d);
    build(x<<1|1, mid+1, r, d);
    pushup(x);
}
int d[2][3];
bool query(int x, int l, int r)
{
    int in = 1, out = 0;
    for(int i = 0; i < 3; i++)
    {
        if(!(d[0][i] <= t[x].d[0][i] && t[x].d[1][i] <= d[1][i])) in = 0; // 包含
        if(d[1][i] < t[x].d[0][i] || t[x].d[1][i] < d[0][i]) out = 1; // 不交
    }
    if(in) return true; if(out) return false;
    int mid = (l+r) >> 1;
    return query(x<<1, l, mid) || query(x<<1|1, mid+1, r);
}
```

2.2 KD 树-gwx

```
struct Point
{
    double x, y;
    int id;
    Point operator - (const Point &a) const {
        return (Point){x - a.x, y - a.y, id};
    }
} b[maxn], c[maxn];
struct node
{
    Point p;
    int ch[2];
} a[maxn];
struct rev
{
    int id;
    double dis;
    bool operator < (const rev &a) const{
        int tmp = sign(dis - a.dis);
        if(tmp)
            return tmp < 0;
        return id < a.id;
    }
};
typedef pr priority_queue <rev>;
pr p0;

int build(int l, int r, int f)
{
    if(l > r)
        return 0;
    int x = (l + r) >> 1;
    if(f == 0)
        nth_element(a + l, a + x, a + r + 1, cmp0); //按x排序
    else
        nth_element(a + l, a + x, a + r + 1, Cmp1); //按y排序
    a[x].ch[0] = build(l, x - 1, f ^ 1);
    a[x].ch[1] = build(x + 1, r, f ^ 1);
    return x;
}
void update(pr &a, rev x)
{
    if(a.size() < K)
        a.push(x);
    else if(x < a.top())
    {
        a.pop();
        a.push(x);
    }
}
pr merge(pr a, pr b)
{
    int s1 = a.size(), s2 = b.size();
    if(s1 < s2)
    {
        while(!a.empty())
        {
            update(b, a.top());
            a.pop();
        }
        return b;
    }
    else
    {
        while(!b.empty())
        {
            update(a, b.top());
            b.pop();
        }
        return a;
    }
}
pr query(int u, Point x, int f)
{
    if(!u)
        return p0;//empty priority_queue
    int d = (dis(a[a[u].ch[0]].p, x) > dis(a[a[u].ch[1]].p, x));
    double dx;
    pr res = query(a[u].ch[d], x, f ^ 1);
    update(res, (rev){a[u].p.id, dis(a[u].p, x)});
    if(f == 0)
        dx = abs(x.x - a[u].p.x);
    else
        dx = abs(x.y - a[u].p.y);
    if(dx > res.top().dis)
        return res;
    res = merge(res, query(a[u].ch[d ^ 1], x, f ^ 1));
    return res;
}
pr solve(Point p)   //离p最近的K个点
{
    int root = build(1, tot, 0);
    return query(root, p, 0);
}
```

2.3 LCT-wrz

```
struct node
{
    node *ch[2], *fa;
    uint v, sum, k, b; int rev, siz;
}mem[N], *tot, *null, *pos[N];
void init()
{
    null = tot = mem;
    null->ch[0] = null->ch[1] = null->fa = null;
    null->v = null->sum = null->b = null->rev = null->siz = 0; null->k = 1;
    for(int i = 1; i <= n; i++) pos[i] = ++tot, *pos[i] = *null, pos[i]->v = pos[i]->sum = 1;
}
int type(node *x){return x->fa->ch[1]==x?1:0;}
int isroot(node *x){return x->fa->ch[type(x)] != x;}
void mswap(node *&x, node *&y){node *t = x; x = y; y = t;}
void pushup(node *x)
{
    x->sum = (x->v + x->ch[0]->sum + x->ch[1]->sum) % MOD;
    x->siz = (x->ch[0]->siz + x->ch[1]->siz + 1) % MOD;
}
void pushdown(node *x)
{
    if(x->rev)
    {
        x->rev = 0, x->ch[0]->rev ^= 1, x->ch[1]->rev ^= 1;
        mswap(x->ch[0]->ch[0], x->ch[0]->ch[1]);
        mswap(x->ch[1]->ch[0], x->ch[1]->ch[1]);
    }
    for(int i = 0; i <= 1; i++)
    {
        x->ch[i]->v = (x->k * x->ch[i]->v % MOD + x->b) % MOD;
        x->ch[i]->sum = (x->ch[i]->sum * x->k % MOD + x->b * x->ch[i]->siz % MOD) % MOD;
        (x->ch[i]->k *= x->k) %= MOD;
        (x->ch[i]->b *= x->k) %= MOD, (x->ch[i]->b += x->b) %= MOD;
    }
    x->k = 1; x->b = 0;
}
void update(node *x){if(!isroot(x))update(x->fa); pushdown(x);}
void rotate(node *x)
{
    node *f = x->fa; int d = type(x);
    x->fa = f->fa, !isroot(f) ? x->fa->ch[type(f)] = x : 0;
    (f->ch[d] = x->ch[d^1]) != null ? f->ch[d]->fa = f : 0;
    f->fa = x, x->ch[d^1] = f; pushup(f);
}
void splay(node *x)
{
    update(x);
    for(; !isroot(x); )
    {
        if(isroot(x->fa)) rotate(x);
        else if(type(x) == type(x->fa)) rotate(x->fa), rotate(x);
        else rotate(x),rotate(x);
    }
```

```
    pushup(x);
}
void access(node *x)
{
    node *tmp = null;
    for(; x != null; )
    {
        splay(x);
        x->ch[1] = tmp;
        pushup(x);
        tmp = x;
        x = x->fa;
    }
}
void makeroot(node *x)
{
    access(x);
    splay(x);
    x->rev ^= 1;
    swap(x->ch[0], x->ch[1]);
}
void link(node *x, node *y)
{
    makeroot(x);
    x->fa = y;
}
void cut(node *x, node *y)
{
    makeroot(x); access(y);
    splay(y); y->ch[0] = x->fa = null;
    pushup(y);
}
```

2.4 左偏树-wrz

```
struct heap
{
    heap *ch[2];
    int dis, siz, v;
}mem[N*2], *h[N], *null, *tot;
heap* newheap()
{
    heap *p = ++tot;
    *p = *null;
    return p;
}
void init()
{
    null = tot = mem;
    null->ch[0] = null->ch[1] = null;
    null->v = null->dis = null->siz = 0;
    for(int i = 1; i <= n; i++) h[i] = null;
}
heap *merge(heap *x, heap *y) // big
{
    if(x == null) return y;
    if(y == null) return x;
    if(x->v < y->v) swap(x, y);
    x->ch[1] = merge(x->ch[1], y);
    if(x->ch[0]->dis < x->ch[1]->dis) swap(x->ch[0], x->ch[1]);
    x->dis = x->ch[1]->dis + 1;
    x->siz = x->ch[0]->siz + x->ch[1]->siz + 1;
    return x;
}
heap *pop(heap *x){return merge(x->ch[0], x->ch[1]);}
int main()
{
    init();
    heap *a = newheap(); a->siz = 1; a->v = 233;
    heap *b = newheap(); b->siz = 1; b->v = 233;
    heap *c = merge(a, b);
}
```

2.5 splay-wrz

```
struct node
{
    node *ch[2], *fa;
    ll key; int siz, tag;
}mem[N*20], *tot, *null, *root;
void init()
{
    root = null = tot = mem;
    null->ch[0] = null->ch[1] = null->fa = null;
    null->key = null->siz = null->tag = 0;
}
int type(node *x){return x->fa->ch[1]==x;}
node *newnode(ll key)
{
    node *p = ++tot; *p = *null;
    p->key = key; p->siz = 1;
    return p;
}
void pushup(node *x)
{
    x->siz = x->ch[0]->siz + x->ch[1]->siz + 1;
}
void rotate(node *x)
{
    node *f =  x->fa; int d = type(x);
    (x->fa = f->fa) != null ? x->fa->ch[type(f)] = x : 0;
    (f->ch[d] = x->ch[!d]) != null ? f->ch[d]->fa = f : 0;
    x->ch[!d] = f, f->fa = x, pushup(f);
}
void pushdown(node *x)
{
    if(x->tag)
    {
        int &tag = x->tag;
        if(x->ch[0] != null) x->ch[0]->key += tag, x->ch[0]->tag += tag;
        if(x->ch[1] != null) x->ch[1]->key += tag, x->ch[1]->tag += tag;
        tag = 0;
    }
}
void update(node *x)
{
    if(x==null) return;
    update(x->fa);
    pushdown(x);
}
void splay(node *x, node *top)
{
    update(x);
    for(;x->fa!=top;)
    {
        if(x->fa->fa == top) rotate(x);
        else if(type(x) == type(x->fa)) rotate(x->fa), rotate(x);
        else rotate(x), rotate(x);
    }
    if(top == null) root = x;
    pushup(x);
}
void insert(node *x, node *f, node *p, int d)
{
    if(x == null)
    {
        p->fa = f, f->ch[d] = p;
        return;
    }
    pushdown(x);
    if(p->key < x->key) insert(x->ch[0], x, p ,0);
    else insert(x->ch[1], x, p, 1);
    pushup(x);
}

void insert(node *p)
{
    if(root == null) root = p, p->fa = p->ch[0] = p->ch[1] = null;
    else insert(root, null, p, 0), splay(p, null);
}
node *findl(node *x){return x->ch[0]==null?x:findl(x->ch[0]);}
node *findr(node *x){return x->ch[1]==null?x:findr(x->ch[1]);}
void insertlr()
{
    insert(newnode(-INF));
    insert(newnode(INF));
}
void delet(node *p)
{
    splay(p, null);
    node *lp = findr(p->ch[0]), *rp = findl(p->ch[1]);
    if(lp == null && rp != null) root = p->ch[1], root->fa = null;
    else if(lp != null && rp == null) root = p->ch[0], root->fa = null;
    else if(lp == null && rp == null)root = null;
    else
    {
        splay(rp, null); splay(lp,rp);
        lp->ch[1] = null; splay(lp,null);
    }
}
node* findk(node *p, int k)
{
    for(; ; )
    {
        pushdown(p);
        if(p->ch[0]->siz >= k) p = p->ch[0];
        else if(p->ch[0]->siz + 1 == k) {splay(p, null); return p;}
        else k -= p->ch[0]->siz + 1, p = p->ch[1];
    }
}
node* findv(node *p, int v)
{
    node* ret = null;
    for(; p!=null; )
    {
        pushdown(p);
        if(p->key >= v) ret = p, p = p->ch[0];
        else p = p->ch[1];
    }
    splay(ret, null);
```

```
    return ret;
}
void addv(node *p, int v)
{
    if(p == null) return;
    p->key += v;
    p->tag += v;
}
```

2.6 treap-gwx

```
//srand()
struct node
{
    int pri, val, c, s;     //pri: random value; c: times of showing; s: size of subtree
    int ch[2];
    int cmp(int x) const {
        if(x == val) return -1;
        return x < val ? 0 : 1;
    }
} a[maxn];
void maintain(int u)  {
    a[u].s = a[u].c + a[a[u].ch[0]].s + a[a[u].ch[1]].s;
}
void rotate(int &u, int d)
{
    int tmp = a[u].ch[d ^ 1];
    a[u].ch[d ^ 1] = a[tmp].ch[d];
    a[tmp].ch[d] = u;
    maintain(u); maintain(tmp);
    u = tmp;
}
void insert(int &u, int val)
{
    if(!u)
    {
        u = ++cnt;
        a[cnt] = (node){rand(), val, 1, 1};
        return;
    }
    a[u].s++;
    int d = a[u].cmp(val);
    if(d == -1)  {a[u].c++; return;}
    insert(a[u].ch[d], val);
    if(a[a[u].ch[d]].pri > a[u].pri) rotate(u, d ^ 1);
}
int find(int u, int val, int comp, int &res)
{
    int d = a[u].cmp(val);
    if(!u) return -1;
    if(d == -1) return u;
    if(d == comp)
    {
        if(d) res = max(res, a[u].val);
        else res = min(res, a[u].val);
    }
    return find(a[u].ch[d], val, comp, res);
}
void remove(int &u)
{
    if(!a[u].ch[0]) u = a[u].ch[1];
    else if(!a[u].ch[1]) u = a[u].ch[0];
    else
    {
        int d = a[a[u].ch[0]].pri < a[a[u].ch[1]].pri ? 0 : 1;
        rotate(u, d); remove(a[u].ch[d]);
    }
}
void del(int &u, int val)
{
    if(find(root, val, -2, val) == -1) return;
    a[u].s--;
    int d = a[u].cmp(val);
    if(d == -1)
    {
        a[u].c--;
        if(!a[u].c) remove(u);
    }
    else del(a[u].ch[d], val);
}
int find_rank(int u, int val)
{
    int d = a[u].cmp(val);
    if(d == -1) return 1 + a[a[u].ch[0]].s;
    if(d == 0) return find_rank(a[u].ch[0], val);
    return a[u].s - a[a[u].ch[1]].s + find_rank(a[u].ch[1], val);
}
int find_kth(int u, int k)
{
    if(k <= a[a[u].ch[0]].s) return find_kth(a[u].ch[0], k);
    if(k > a[a[u].ch[0]].s + a[u].c) return find_kth(a[u].ch[1], k - a[a[u].ch[0]].s - a[u].c);
    return a[u].val;
}
int pre(int val)
{
    int ans = -inf;
    int pos = find(root, val, 1, ans);
    if(pos != -1 && a[pos].ch[0])
    {
        pos = a[pos].ch[0];
        while(a[pos].ch[1]) pos = a[pos].ch[1];
        ans = max(ans, a[pos].val);
    }
    return ans;
}
int post(int val)
{
    int ans = inf;
    int pos = find(root, val, 0, ans);
    if(pos != -1 && a[pos].ch[1])
    {
        pos = a[pos].ch[1];
        while(a[pos].ch[0]) pos = a[pos].ch[0];
        ans = min(ans, a[pos].val);
    }
    return ans;
}
```

2.7 可持久化平衡树

```
int Copy(int x){// 可持久化
    id++;sz[id]=sz[x];L[id]=L[x];R[id]=R[x];
    v[id]=v[x];return id;}
int merge(int x,int y){
    // 合并x 和y 两颗子树，可持久化到z 中
    if(!x||!y)return x+y;int z;
    int o=rand()%(sz[x]+sz[y]);// 注意rand 上限
    if(o<sz[x])z=Copy(y),L[z]=merge(x,L[y]);
    else z=Copy(x),R[z]=merge(R[x],y);
    ps(z);return z;
}
void split(int x,int&y,int&z,int k){
    // 将x 分成y 和z 两颗子树，y 的大小为k
    y=z=0;if(!x)return;
    if(sz[L[x]]>=k)z=Copy(x),split(L[x],y,L[z],k),ps(z);
    else y=Copy(x),split(R[x],R[y],z,k-sz[L[x]]-1),ps(y);}
```

3 图论

3.1 匹配

Tutte-Berge formula The theorem states that the size of a maximum matching of a graph $G = (V, E)$ equals

$$\frac{1}{2}\min_{U\subseteq V}\left(|U| - \operatorname{odd}(G-U) + |V|\right),$$

where $\operatorname{odd}(H)$ counts how many of the connected components of the graph H have an odd number of vertices.

Tutte theorem A graph, $G = (V, E)$, has a perfect matching if and only if for every subset U of V, the subgraph induced by $V - U$ has at most $|U|$ connected components with an odd number of vertices.

Hall's marriage theorem A family S of finite sets has a transversal if and only if S satisfies the marriage condition.

3.2 Hopcoft-Karp

```
// O(sqrt(n)m)
template <int MAXN = 100000, int MAXM = 100000>
struct hopcoft_karp {
    int mx[MAXN], my[MAXM], lv[MAXN];
    bool dfs (edge_list <MAXN, MAXM> &e, int x) {
        for (int i = e.begin[x]; ~i; i = e.next[i]) {
            int y = e.dest[i], w = my[y];
            if (!~w || (lv[x] + 1 == lv[w] && dfs (e, w))) {
                mx[x] = y; my[y] = x; return true; } }
        lv[x] = -1; return false; }
    int solve (edge_list <MAXN, MAXM> &e, int n, int m) {
        std::fill (mx, mx + n, -1); std::fill (my, my + m, -1);
        for (int ans = 0; ; ) {
            std::vector <int> q;
            for (int i = 0; i < n; ++i)
                if (mx[i] == -1) {
                    lv[i] = 0; q.push_back (i);
                } else lv[i] = -1;
            for (int head = 0; head < (int) q.size(); ++head) {
                int x = q[head];
                for (int i = e.begin[x]; ~i; i = e.next[i]) {
                    int y = e.dest[i], w = my[y];
                    if (~w && lv[w] < 0) { lv[w] = lv[x] + 1; q.push_back (w); } } }
            int d = 0; for (int i = 0; i < n; ++i) if (!~mx[i] && dfs (e, i)) ++d;
            if (d == 0) return ans; else ans += d; } }
};
```

3.3 KM-truly-n3

```
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;
```

3.4 tarjan-gwx

```
//cut[i]: i是否为割点
//bridge[i]: e[i]是否为桥
void dfs(int u, int pa)
{
    d[u] = l[u] = ++timer;
    st.push(u); vst[u] = 1;
    int child = 0;
    for(int i = tail[u]; i; i = e[i].next)
        if(!d[e[i].v])
        {
            child++;
            dfs(e[i].v, u);
            l[u] = min(l[u], l[e[i].v]);
            if(l[e[i].v] >= d[u])
            {
                cut[u] = 1;
                if(l[e[i].v] > d[u])
                    bridge[i] = 1;
            }
        }
        else if(vst[e[i].v]) l[u] = min(l[u], d[e[i].v]);
    if(!pa && child < 2) cut[u] = 0;
    if(l[u] == d[u])
    {
        int v; scc++;
        while(true)
        {
            v = st.top(); st.pop();
            id[v] = scc; vst[v] = 0; size[scc]++;
            if(u == v) break;
        }
    }
}
```

3.5 边双联通-gwx

```
//G[i]: 第i个边双联通分量中有哪些点
void tarjan(int u, int pa)
{
    d[u] = l[u] = ++timer;
    for(int i = tail[u]; i; i = e[i].next)
    {
        if(!d[e[i].v])
        {
            st[++top] = i;
            tarjan(e[i].v, u);
            l[u] = min(l[u], l[e[i].v]);
            if(l[e[i].v] >= d[u])
            {
                bcc++;
                while(true)
                {
                    int now = st[top--];
                    if(vst[e[now].u] != bcc)
                    {
                        vst[e[now].u] = bcc;
                        G[bcc].push_back(e[now].u);
                    }
                    if(vst[e[now].v] != bcc)
                    {
                        vst[e[now].v] = bcc;
                        G[bcc].push_back(e[now].v);
                    }
                    if(now == i) break;
                }
            }
        }
        else if(e[i].v != pa)
            l[u] = min(l[u], d[e[i].v]);
    }
}
```

3.6 最大团

```
/*
Int g[][]为图的邻接矩阵。
    MC(V)表示点集V的最大团
    令Si={vi, vi+1, ..., vn}, mc[i]表示MC(Si)
    倒着算mc[i], 那么显然MC(V)=mc[1]
    此外有mc[i]=mc[i+1] or mc[i]=mc[i+1]+1
*/
void init(){
    int i, j;
    for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%d", &g[i][j]);
}
void dfs(int size){
    int i, j, k;
    if (len[size]==0) {
        if (size>ans) {
            ans=size; found=true;
        }
        return;
    }
    for (k=0; k<len[size] && !found; ++k) {
        if (size+len[size]-k<=ans) break;
        i=list[size][k];
        if (size+mc[i]<=ans) break;
        for (j=k+1, len[size+1]=0; j<len[size]; ++j)
        if (g[i][list[size][j]]) list[size+1][len[size+1]++]=list[size][j];
        dfs(size+1);
    }
}
void work(){
    int i, j;
    mc[n]=ans=1;
    for (i=n-1; i; --i) {
        found=false;
        len[1]=0;
        for (j=i+1; j<=n; ++j) if (g[i][j]) list[1][len[1]++]=j;
        dfs(1);
        mc[i]=ans;
    }
}
```

3.7 欧拉回路-wrz

```
#include<cstdio>
#define N 100005
#define M 200005
using namespace std;
int last[N], ecnt = 1, cnt, ans[M], in_deg[N], out_deg[N];
bool vis[M];
struct edge{int next,to;}e[M<<1];
void addedge(int a, int b)
```

```
{
    e[++ecnt] = (edge){last[a], b};
    last[a] = ecnt;
}
void dfs(int x)
{
    for(int &i = last[x]; i; i = e[i].next)
    {
        int y = e[i].to, j = i;
        if(!vis[j>>1])
        {
            vis[j>>1] = 1;
            dfs(y);
            ans[++cnt] = j;
        }
    }
}
int main()
{
    int t, n, m, a, b;
    scanf("%d%d%d",&t,&n,&m);
    for(int i = 1; i <= m; i++)
    {
        scanf("%d%d",&a,&b);
        addedge(a,b);
        if(t == 1)addedge(b,a), in_deg[a]++, in_deg[b]++;
        else ecnt++, in_deg[b]++, out_deg[a]++;
    }

    if(t == 1) // 无向
    {
        for(int i = 1; i <= n; i++)
            if((in_deg[i]+out_deg[i]) & 1)
                return !printf("NO\n");
    }
    else // 有向
    {
        for(int i = 1; i <= n; i++)
            if(in_deg[i] != out_deg[i])
                return !printf("NO\n");
    }
    dfs(a);
    if(cnt != m)
    {
        puts("NO");
    }
    else
    {
        puts("YES");
        for(int i = cnt; i; i--)
        {
            printf("%d ",ans[i]&1?-(ans[i]>>1):(ans[i]>>1));
        }
    }
}
```

3.8 SPFA 判负环-wrz

```
int inq[N], inqt[N], dis[N];
bool SPFA()
{
    queue<int> q;
    for(int i = 1; i <= n; i++) dis[i] = 0, q.push(i), inq[i] = 1; // 全部入队
    for(; !q.empty(); )
    {
        int x = q.front(); q.pop(); inq[x] = 0;
        for(int i = last[x]; i; i = e[i].next)
        {
            int y = e[i].to;
            if(dis[x] + e[i].val < dis[y])
            {
                dis[y] = dis[x] + e[i].val;
                if(!inq[y])
                {
                    if(++inqt[y] > n) return false; // 入队n次即有负环
                    inq[y] = 1;
                    q.push(y);
                }
            }
        }
    }
    return true;
}
/*
    步骤:
    1.建好原图
    2.SPFA() // 若返回为true表示无负环, false表示有负环

    多次调用时记得清空inqt等数组
    有负环时理论复杂度是O(n^2)的
*/
```

3.9 k 短路 a 星-gwx

```
const int maxn = 1005;
int n, m;
int S, T, K;
int dist[maxn], cnt[maxn];
bool vst[maxn];
vector<pair<int, int>> G[maxn], H[maxn];    //正图&反图
struct node
{
    ll d;
    int id;
    node(){}
    node(ll d, int id): d(d), id(id) {}
    bool operator< (const node &other) const{
        return d + dist[id] > other.d + dist[other.id];
    }
};

priority_queue <pair<ll, int>> q;
priority_queue <node> Q;

void init()
{
    for(int i = 1; i <= n; ++i)
        G[i].clear(), H[i].clear(), cnt[i] = 0;
}

void dijkstra(int S)
{
    memset(dist, 127, sizeof(dist));
    memset(vst, 0, sizeof(vst));
    while(!q.empty()) q.pop();
    dist[S] = 0;
    q.push(make_pair(0, S));
    for(int i = 1; i <= n; ++i)
    {
        if(q.empty()) break;
        while(vst[q.top().second]) q.pop();
        int u = q.top().second; q.pop();
        vst[u] = 1;
        for(auto i: H[u])
        {
            if(dist[i.first] > dist[u] + i.second)
            {
                dist[i.first] = dist[u] + i.second;
                q.push(make_pair(-dist[i.first], i.first));
            }
        }
    }
}

int solve()
{
    while(!Q.empty()) Q.pop();
    Q.push(node(0, S));
    while(!Q.empty())
    {
        auto u = Q.top(); Q.pop();
        if(++cnt[u.id] > K) continue;
        if(u.d + dist[u.id] > ti) continue;
        if(u.id == T && cnt[T] == K)
            return u.d;
        for(auto i: G[u.id])
            Q.push(node(u.d + i.second, i.first));
    }
    return -1;
}
```

3.10 K 短路可并堆

```
//Kth Shortest Path via Persistable Mergeable Heap
//可持久化可并堆求k短路 O(SSSP+(m+k)\log n)
//By ysf
//通过题目: USACO Mar08 牛跑步 (板子题)

//注意这是个多项式算法, 在k比较大时很有优势, 但k比较小时最好还是用A*
//DAG和有环的情况都可以, 有重边或自环也无所谓, 但不能有零环
//以下代码以Dijkstra+可持久化左偏树为例

const int maxn=1005,maxe=10005,maxm=maxe*30;//点数, 边数, 左偏树结点数

//需要用到的结构体定义
struct A{//用来求最短路
    int x,d;
    A(int x,int d):x(x),d(d){}
    bool operator<(const A &a)const{return d>a.d;}
};

struct node{//左偏树结点
    int w,i,d;//i: 最后一条边的编号 d: 左偏树附加信息
    node *lc,*rc;
    node(){}
    node(int w,int i):w(w),i(i),d(0){}
    void refresh(){d=rc->d+1;}
}null[maxm],*ptr=null,*root[maxn];
```

```cpp

struct B{//维护答案用
    int x,w;//x是结点编号，w表示之前已经产生的权值
    node *rt;//这个答案对应的堆顶，注意可能不等于任何一个结点的堆
    B(int x,node *rt,int w):x(x),w(w),rt(rt){}
    bool operator<(const B &a)const{return w+rt->w>a.w+a.rt->w;}
};

//全局变量和数组定义
vector<int>G[maxn],W[maxn],id[maxn];//最开始要存反向图，然后把G清空作为儿子列表
bool vis[maxn],used[maxe];//used表示边是否在最短路树上
int u[maxe],v[maxe],w[maxe];//存下每条边，注意是有向边
int d[maxn],p[maxn];//p表示最短路树上每个点的父边
int n,m,k,s,t;//s,t分别表示起点和终点

//以下是主函数中较关键的部分
for(int i=0;i<=n;i++)root[i]=null;//一定要加上！！！
//(读入&建反向图)
Dijkstra();
//(清空G,W,id)
for(int i=1;i<=n;i++)
    if(p[i]){
        used[p[i]]=true;//在最短路树上
        G[v[p[i]]].push_back(i);
    }
for(int i=1;i<=m;i++){
    w[i]-=d[u[i]]-d[v[i]];//现在的w[i]表示这条边能使路径长度增加多少
    if(!used[i])
        root[u[i]]=merge(root[u[i]],newnode(w[i],i));
}
dfs(t);
priority_queue<B>heap;
heap.push(B(s,root[s],0));//初始状态是找贡献最小的边加进去
printf("%d\n",d[s]);//第1短路需要特判
while(--k){//其余k-1短路径用二叉堆维护
    if(heap.empty())printf("-1\n");
    else{
        int x=heap.top().x,w=heap.top().w;
        node *rt=heap.top().rt;
        heap.pop();
        printf("%d\n",d[s]+w+rt->w);
        if(rt->lc!=null||rt->rc!=null)
            heap.push(B(x,merge(rt->lc,rt->rc),w));//pop掉当前边，换成另一条贡献大一点的边
        if(root[v[rt->i]]!=null)
            heap.push(B(v[rt->i],root[v[rt->i]],w+rt->w));//保留当前边，往后面再接上另一条边
    }
}
//主函数到此结束

//Dijkstra预处理最短路 O(m\log n)
void Dijkstra(){
    memset(d,63,sizeof(d));
    d[t]=0;
    priority_queue<A>heap;
    heap.push(A(t,0));
    while(!heap.empty()){
        int x=heap.top().x;
        heap.pop();
        if(vis[x])continue;
        vis[x]=true;
        for(int i=0;i<(int)G[x].size();i++)
            if(!vis[G[x][i]]&&d[G[x][i]]>d[x]+W[x][i]){
                d[G[x][i]]=d[x]+W[x][i];
                p[G[x][i]]=id[x][i];
                heap.push(A(G[x][i],d[G[x][i]]));
            }
    }
}

//dfs求出每个点的堆 总计O(m\log n)
//需要调用merge，同时递归调用自身
void dfs(int x){
    root[x]=merge(root[x],root[v[p[x]]]);
    for(int i=0;i<(int)G[x].size();i++)
        dfs(G[x][i]);
}

//包装过的new node() O(1)
node *newnode(int w,int i){
    *++ptr=node(w,i);
    ptr->lc=ptr->rc=null;
    return ptr;
}

//带可持久化的左偏树合并 总计O(\log n)
//递归调用自身
node *merge(node *x,node *y){
    if(x==null)return y;
    if(y==null)return x;
    if(x->w>y->w)swap(x,y);
    node *z=newnode(x->w,x->i);
    z->lc=x->lc;
    z->rc=merge(x->rc,y);
    if(z->lc->d>z->rc->d)swap(z->lc,z->rc);
    z->refresh();
    return z;
}
```

3.11 上下界网络流

有源汇上下界费用流 转换为求无源汇上下界最小费用可行循环流, 通过 $T \to S$ 连边, 流量上下界为 (原总流量, ∞)。

无源汇上下界最小费用可行循环流 在原基础上再新增一个超级源点 $supS, supT$, 构造只有上界的网络。

对于原图的每一条边 (u, v) , 再新图中添加一条 $supS \to v$ 流量为 u, v 流量下界的边, 一条 $u \to supT$ 流量为 u, v 流量下界的边, 一条 $u \to v$ 流量为 u, v 流量上界-流量下界的边。

做从 $supS \to supT$ 的最小费用流, 限定到达 $supT$ 的流量为满流（即 $supS$ 所有出边的流量和）。此即为答案。

HINT：原图中所有未提及的边费用都应记为 0 。新图中的重新构造的边的费用等同原图中对应边的费用。

上下界网络流 $B(u, v)$ 表示边 (u, v) 流量的下界,$C(u, v)$ 表示边 (u, v) 流量的上界,$F(u, v)$ 表示边 (u, v) 的流量。设 $G(u, v) = F(u, v) - B(u, v)$, 显然有 $0 \le G(u, v) \le C(u, v) - B(u, v)$

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 $G(u, v) + B(u, v)$。

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。

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 ∞, 下界为 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。

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。

3.12 zkw 费用流

```cpp
//稠密图、二分图中较快，稀疏图中不如SPFA
int flow, cost, price;

int dfs(int u, int f)
{
    if(u == t)
    {
        flow += f;
        cost += price * f;
        return f;
    }
    vst[u] = 1;
    int used = 0;
    for(int i = tail[u]; i; i = e[i].next)
        if(!vst[e[i].v] && e[i].c > 0 && e[i].w == 0)
        {
            int w = dfs(e[i].v, min(e[i].c, f - used));
            e[i].c -= w; e[i ^ 1].c += w; used += w;
            if(used == f) return f;
        }
    return used;
}
bool modlabel()
{
    int d = inf;
    for(int u = s; u <= t; u++)
        if(vst[u])
            for(int i = tail[u]; i; i = e[i].next)
                if(e[i].c > 0 && !vst[e[i].v]) d = min(d, e[i].w);
    if(d == inf) return 0;
    for(int u = s; u <= t; u++)
        if(vst[u])
            for(int i = tail[u]; i; i = e[i].next)
                e[i].w -= d, e[i ^ 1].w += d;
    price += d;
    return 1;
}
void zkw()
{
```

```cpp
    do
        do memset(vst, 0, sizeof(vst));
        while(dfs(s, inf) > 0);
    while(modlabel());
}
```

3.13 stoer-wagner 无向图最小割树

```cpp
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);
}
```

3.14 朱刘算法-gwx

```cpp
//时间复杂度: O(nm)
int N, m;
int pre[maxn], in[maxn], f[maxn], id[maxn];
struct node {int u, v, w;} a[maxm * 2]; //边表

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

int mst()
{
    long long res = 0;
    int root = 1;
    int n = N;
    while(true)
    {
        for(int i = 1; i <= n; i++) in[i] = INT_MAX, pre[i] = 0;
        for(int i = 1; i <= m; i++)
            if(a[i].u != a[i].v  && in[a[i].v] > a[i].w)
                in[a[i].v] = a[i].w, pre[a[i].v] = a[i].u;
        for(int i = 1; i <= n; i++)
            if(in[i] == INT_MAX && i != root) return 0;
        int cnt = 0;
        for(int i = 1; i <= n; i++) f[i] = i, id[i] = 0;
        for(int i = 1; i <= n; i++)
        {
            if(i == root) continue;
            res += in[i];
            if(find(i) != find(pre[i])) f[f[i]] = f[pre[i]];
            else
            {
                cnt++;
                for(int j = i; j && !id[j]; j = pre[j])
                    id[j] = cnt;
            }
        }
        if(!cnt) break;
        for(int i = 1; i <= n; i++)
            if(!id[i]) id[i] = ++cnt;
        for(int i = 1; i <= m; i++)
        {
            if(id[a[i].u] != id[a[i].v]) a[i].w -= in[a[i].v];
            a[i].u = id[a[i].u];
            a[i].v = id[a[i].v];
        }
        n = cnt;
        root = id[root];
    }
    return res;
}
```

3.15 树哈希

A[n] is the hash of the sub-tree with root n.
B[n] is the hash of the whole tree with root n.

```cpp
template <int MAXN = 100000, int MAXM = 200000, long long MOD = 1000000000000000003ll>
struct tree_hash {
    static long long ra[MAXN];
    tree_hash () {
        std::mt19937_64 mt (time (0));
        std::uniform_int_distribution <long long> uid (0, MOD - 1);
        for (int i = 0; i < MAXN; ++i) ra[i] = uid (mt); }
    struct node {
        std::vector <long long> s; int d1, d2; long long h1, h2;
        node () { d1 = d2 = 0; }
        void add (int d, long long v) {
            s.push_back (v);
            if (d > d1) d2 = d1, d1 = d; else if (d > d2) d2 = d; }
        long long hash () {
            h1 = h2 = 1; for (long long i : s) {
                h1 = mul_mod (h1, ra[d1] + i, MOD);
                h2 = mul_mod (h2, ra[d2] + i, MOD); }
                return h1; }
    std::pair <int, long long> del (int d, long long v) {
        if (d == d1) return { d2 + 1, mul_mod (h2, inverse (ra[d2] + v, MOD), MOD) };
        return { d1 + 1, mul_mod (h1, inverse (ra[d1] + v, MOD), MOD) }; } };
    std::pair <int, long long> u[MAXN]; node tree[MAXN];
    long long A[MAXN], B[MAXN];
    void dfs1 (const edge_list <MAXN, MAXM> &e, int x, int p = -1) {
        tree[x] = node ();
        for (int i = e.begin[x]; ~i; i = e.next[i]) {
            int c = e.dest[i]; if (c != p) {
                dfs1 (e, c, x); tree[x].add (tree[c].d1 + 1, tree[c].h1); } }
        A[x] = tree[x].hash (); }
    void dfs2 (const edge_list <MAXN, MAXM> &e, int x, int p = -1) {
        if (~p) tree[x].add (u[x].first, u[x].second);
        B[x] = tree[x].hash ();
    for (int i = e.begin[x]; ~i; i = e.next[i]) {
            int c = e.dest[i]; if (c != p) {
                u[c] = tree[x].del (tree[c].d1 + 1, tree[c].h1);
                dfs2 (e, c, x); } } }
    void solve (const edge_list <MAXN, MAXM> &e, int root) {
        dfs1 (e, root); dfs2 (e, root); } };
template <int MAXN, int MAXM, long long MOD>
long long tree_hash <MAXN, MAXM, MOD>::ra[MAXN];
```

3.16 矩阵树定理

C = 度数矩阵-邻接矩阵
无向图 G 的生成树个数 = C 的任意 $n-1$ 阶主子式 (对角线的乘积)

3.17 带花树

```cpp
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();
}
```

3.18 支配树-gwx

```
/*
    用ins()加边
    build前设置n为点数，s为源点
    树中的i号点对应原图的id[i]号点
*/
struct Dominator_Tree {
    int n, s, cnt;
    int dfn[N], id[N], pa[N], semi[N], idom[N], p[N], mn[N];
    vector<int>e[N], dom[N], be[N];
    void ins(int x, int y) {e[x].push_back(y);}
    void dfs(int x) {
        dfn[x] = ++cnt; id[cnt] = x;
        for (int i : e[x]) {
            if (!dfn[i])dfs(i), pa[dfn[i]] = dfn[x];
            be[dfn[i]].push_back(dfn[x]);
        }
    }
    int get(int x) {
        if (p[x] != p[p[x]]) {
            if (semi[mn[x]] > semi[get(p[x])])mn[x] = get(p[x]);
            p[x] = p[p[x]];
        }
        return mn[x];
    }
    void LT() {
        for (int i = cnt; i > 1; i--) {
            for (int j : be[i])semi[i] = min(semi[i], semi[get(j)]);
            dom[semi[i]].push_back(i);
            int x = p[i] = pa[i];
            for (int j : dom[x])idom[j] = (semi[get(j)] < x ? get(j) : x);
            dom[x].clear();
        }
        for (int i = 2; i <= cnt; i++) {
            if (idom[i] != semi[i])idom[i] = idom[idom[i]];
            dom[id[idom[i]]].push_back(id[i]);
        }
    }
    void build() {
        for(int i = 1; i <= n; ++i)
            dfn[i] = 0, dom[i].clear(), be[i].clear(), p[i] = mn[i] = semi[i] = i;
        cnt = 0; dfs(s); LT();
    }
};
```

3.19 斯坦纳树

```
//N点数，M边数，P关键点数
const int inf = 0x3f3f3f3f;
int n, m, p, status, idx[P], f[1 << P][N];
priority_queue<pair<int, int> > q; //int top, h[N];
void dijkstra(int dis[]) {}
void Steiner_Tree() {
    for (int i = 1; i < status; i++) {
        while (!q.empty()) q.pop(); //top = 0;
        memset(vis, 0, sizeof(vis));
        for (int j = 1; j <= n; j++) {
            for (int k = i & (i - 1); k; (--k) &= i)
                f[i][j] = min(f[i][j], f[k][j] + f[i ^ k][j]);
            if (f[i][j] != inf)
                q.push(make_pair(-f[i][j], j)); //h[++top] = j, vis[j] = 1;
        }
        dijkstra(f[i]); //SPFA(f[i]);
    }
}
int main() {
    scanf("%d%d%d", &n, &m, &p);
    status = 1 << p;
    tot = 0; memset(lst, 0, sizeof(lst));
    /*求最小生成森林
      每棵生成树中至少选择一个点，点权为代价
      新开一个空白关键点0作为源
      Add(0, i, val[i]); Add(i, 0, val[i]);*/
    for (int i = 1; i <= p; i++) scanf("%d", &idx[i]);
    memset(f, 0x3f, sizeof(f));
    for (int i = 1; i <= n; i++) f[0][i] = 0;
    for (int i = 1; i <= p; i++) f[1 << (i - 1)][idx[i]] = 0;
    Steiner_Tree();
    int ans = inf;
    for (int i = 1; i <= n; i++) ans = min(ans, f[status - 1][i]);
}
```

3.20 弦图

```
template <int MAXN = 100000, int MAXM = 100000>
struct chordal_graph {
    int n; edge_list <MAXN, MAXM> e;
    int id[MAXN], seq[MAXN];
    void init () {
        struct point {
            int lab, u;
            point (int lab = 0, int u = 0) : lab (lab), u (u) {}
            bool operator < (const point &a) const { return lab < a.lab; } };
        std::fill (id, id + n, -1);
        static int label[MAXN]; std::fill (label, label + n, 0);
        std::priority_queue <point> q;
        for (int i = 0; i < n; ++i) q.push (point (0, i));
        for (int i = n - 1; i >= 0; --i) {
            for (; ~id[q.top ().u]; ) q.pop ();
            int u = q.top ().u; q.pop (); id[u] = i;
            for (int j = e.begin[u], v; ~j; j = e.next[j])
                if (v = e.dest[j], !~id[v]) ++label[v], q.push (point (label[v], v)); }
        for (int i = 0; i < n; ++i) seq[id[i]] = i; }
    bool is_chordal () {
        static int vis[MAXN], q[MAXN]; std::fill (vis, vis + n, -1);
        for (int i = n - 1; i >= 0; --i) {
            int u = seq[i], t = 0, v;
            for (int j = e.begin[u]; ~j; j = e.next[j])
                if (v = e.dest[j], id[v] > id[u]) q[t++] = v;
            if (!t) continue; int w = q[0];
            for (int j = 0; j < t; ++j) if (id[q[j]] < id[w]) w = q[j];
            for (int j = e.begin[w]; ~j; j = e.next[j]) vis[e.dest[j]] = i;
            for (int j = 0; j < t; ++j) if (q[j] != w && vis[q[j]] != i) return 0;
        }
        return 1; }
    int min_color () {
        int res = 0;
        static int vis[MAXN], c[MAXN];
        std::fill (vis, vis + n, -1);
        std::fill (c, c + n, n);
        for (int i = n - 1; i >= 0; --i) {
            int u = seq[i];
            for (int j = e.begin[u]; ~j; j = e.next[j]) vis[c[e.dest[j]]] = i;
            int k; for (k = 0; vis[k] == i; ++k);
            c[u] = k; res = std::max (res, k + 1);
        }
        return res; } };
```

4 数学

4.1 杜教筛

```
// 用之前必须先init(); 如果n很大, 求和记得开long long; 如果有取模, 求和记得改取模
#define N 1000005 // (10^9)^(2/3)
#define M 3333331 // hash siz
int prime[N], notprime[N], pcnt, mu[N], pre[N];
int hash[M], nocnt; struct node{int id, f, next;}no[1000000];
int F(int n) // calculate mu[1]+mu[2]+...+mu[n]
{
    if(n<N) return pre[n];
    int h = n%M; for(int i = hash[h]; i; i = no[i].next) if(no[i].id == n) return no[i].f;
    int ret = 1;
    for(int i = 2, j; i <= n; i = j + 1)
    {
        j = n/(n/i);
        ret -= F(n/i) * (j-i+1);
    }
    no[++nocnt] = (node){n, ret, hash[h]};
    hash[h] = nocnt;
    return ret;
}
void init()
{
    mu[1] = 1;
    for(int i = 2; i < N; i++)
    {
        if(!notprime[i]) prime[++pcnt] = i, mu[i] = -1;
        for(int j = 1; j <= pcnt && prime[j] * i < N; j++)
        {
            notprime[prime[j] * i] = 1;
            if(i % prime[j]) mu[prime[j] * i] = -mu[i];
            else {mu[prime[j] * i] = 0; break;}
        }
    }
    for(int i = 1; i < N; i++) pre[i] = pre[i-1] + mu[i];
}
```

4.2 直线下整点

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

```
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);
}
```

4.3 拉格朗日插值

```
// 用之前必须先init(); 如果所有的逆元都能预处理就是O(n)的, 否则是O(nlogn)的
#define MOD 1000000007
int inv[N], invf[N], f[N];
int fpow(int a, int b)
{
    int r = 1;
    for(; b; b >>= 1)
    {
        if(b & 1) r = 1ll*r*a%MOD;
        a = 1ll*a*a%MOD;
    }
    return r;
}
int la(int x, int k) // k次, 求f(x)
{
    int lim = k+2, ff = 1;
    for(int i = 1; i <= lim; i++)
        ff = 1ll * ff * (x-i) % MOD;
    for(int i = 1; i <= lim; i++)
        f[i] = (f[i-1] + fpow(i, k)) % MOD; // 预处理f(1),f(2),...,f(lim), 注意修改
    if(x <= lim) return f[x];
    int ret = 0;
    for(int i = 1; i <= lim; i++)
    {
        (ret += 1ll * f[i]
                * ff % MOD * (x-i < N ? inv[x-i] : fpow(x-i, MOD-2)) % MOD // 复杂度
                * invf[i-1] % MOD * invf[lim-i] % MOD * ((lim-i) % 2 ? MOD-1 : 1) % MOD
        ) %= MOD;
    }
    return ret;
}
void init()
{
    inv[1] = 1;
    for(int i = 2; i < N; i++) inv[i] = 1ll * (MOD - MOD / i) * inv[MOD % i] % MOD;
    invf[0] = 1;
    for(int i = 1; i < N; i++) invf[i] = 1ll * invf[i-1] * inv[i] % MOD;
}
```

4.4 FFT-wrz

```
typedef complex<double> comp;
int len;
comp w[N<<1], a[N<<1], b[N<<1], c[N<<1]; // 数组记得至少开两倍
void init()
{
    double pi = acos(-1.0);
    for(int i = 0; i < len; i++) w[i] = (comp){cos(2*pi*i/len), sin(2*pi*i/len)};
}
void FFT(comp *a, comp *w)
{
    for(int i = 0, j = 0; i < len; i++)
    {
        if(i<j) swap(a[i], a[j]);
        for(int l = len>>1; (j^=l)<l; l >>= 1);
    }
    for(int i = 2; i <= len; i <<= 1)
    {
        int m = i >> 1;
        for(int j = 0; j < len; j += i)
        {
            for(int k = 0; k < m; k++)
            {
                comp tmp = w[len/i*k] * a[j+k+m];
                a[j+k+m] = a[j+k] - tmp;
                a[j+k] = a[j+k] + tmp;
            }
        }
    }
}
void mul(comp *a, comp *b, comp *c, int l) // 多项式乘法, c = a * b , c的长度为l
{
    for(len = 1; len <= l; len <<= 1);
    init(); FFT(a, w); FFT(b, w);
    for(int i = 0; i < len; i++) c[i] = a[i] * b[i];
    reverse(c+1, c+len); FFT(c, w);
    for(int i = 0; i < len; i++) c[i].r /= len; // 转化为int等时应加0.5, 如int(c[i].r+0.5)
}
```

4.5 NTT-gwx

```
const int G;
int n, m, inm;
int rev[maxn], a[maxn], b[maxn];    //maxn > 2 ^ k

ll power(ll b, int k)
{
    ll res = 1;
    for(; k; k >>= 1, b = b * b % mod)
        if(k & 1)
            res = res * b % mod;
    return res;
}

void ntt(ll*a, int f)
{
    for(int i = 0; i < m; i++)
        if(rev[i] < i)
            swap(a[i], a[rev[i]]);
    for(int l = 2, h = 1; l <= m; h = l, l <<= 1)
    {
        int ur;
        if(f == 1)
            ur = power(G, (mod - 1) / l);
        else
            ur = power(G, mod - 1 - (mod - 1) / l);
        for(int i = 0; i < m; i += l)
        {
            ll w = 1;
            for(int k = i; k < i + h; k++, w = w * ur % mod)
            {
                int x = a[k], y = a[k + h] * w % mod;
                a[k] = (x + y) % mod;
                a[k + h] = (x - y + mod) % mod;
            }
        }
    }
    if(f == -1)
        for(int i = 0; i < m; i++)
            a[i] = a[i] * inm % mod;
}

void multi()
{
    ntt(a, 1); ntt(b, 1);
    for(int i = 0; i < m; i++)
        a[i] = a[i] * b[i] % mod;
    ntt(a, -1);
}
```

```cpp
void init()
{
    for(m = 1; m <= 2 * n; m <<= 1) ;
    for(int i = 1; i < m; i++)
    {
        rev[i] = rev[i - 1];
        for(int j = m >> 1; (rev[i] ^= j) < j; j >>= 1) ;
    }
}
```

4.6 FWT

```cpp
void FWT(int a[],int n)
{
    for (int d = 1; d < n; d <<= 1)
        for (int m = d << 1, i = 0; i < n; i += m)
            for (int j = 0; j < d; j++) {
                int x = a[i + j], y = a[i + j + d];

                a[i + j] = (x + y) % mod,
                a[i + j + d] = (x - y + mod) % mod;

                //xor: a[i + j] = x + y, a[i + j + d] = (x - y + mod) % mod;
                //and: a[i + j] = x + y;
                //or:  a[i + j + d] = x + y;
            }
}

void UFWT(int a[], int n)
{
    for (int d = 1; d < n; d <<= 1)
        for (int m = d << 1, i = 0; i < n; i += m)
            for(int j = 0; j < d; j++) {
                int x = a[i + j], y = a[i + j + d];

                a[i + j] = 1LL * (x + y) * rev % mod,
                a[i + j + d] = (1LL * (x - y) * rev % mod + mod) % mod;

                //xor: a[i + j] = (x + y) / 2, a[i + j + d] = (x - y) / 2;
                //and: a[i + j] = x - y;
                //or:  a[i + j + d] = y - x;
            }
}

void solve(int a[], int b[], int n)
{
    FWT(a, n);
    FWT(b, n);
    for (int i = 0; i < n; i++)
        a[i] = 1LL * a[i] * b[i] % mod;
    UFWT(a, n);
}
```

4.7 高精度-wrz

```cpp
#include<bits/stdc++.h>
#define BASE 10000
#define L 20005
using namespace std;
int p; char s[10*L];
struct bigint
{
    int num[L], len;
    bigint(int x = 0){memset(num,0,sizeof(num)); len = 1; num[0] = x;}
    bigint operator + (bigint b)
    {
        bigint c;
        c.len = max(b.len, len);
        for(int i = 0; i < c.len; i++)
        {
            c.num[i] += num[i] + b.num[i];
            c.num[i+1] = c.num[i] / BASE;
            c.num[i] %= BASE;
        }
        if(c.num[c.len])c.len++;
        return c;
    }
    bigint operator - (bigint b)
    {
        bigint c;
        c.len = max(len, b.len);
        for(int i = 0; i < c.len; i++)
        {
            c.num[i] += num[i] - b.num[i];
            if(c.num[i] < 0) {c.num[i] += BASE; c.num[i+1]--;}
        }
        while(!c.num[c.len-1] && c.len > 1)c.len--;
        return c;
    }
    void operator -= (int b)
    {
        num[0] -= b;
        for(int i = 0; i < len; i++)
        {
            num[i+1] += num[i] / BASE;
            num[i] %= BASE;
            if(num[i] < 0)num[i] += BASE, num[i+1]--;
        }
        while(!num[len-1] && len > 1) len--;
    }
    bigint operator * (bigint b)
    {
        bigint c;
        c.len = len + b.len;
        for(int i = 0; i < len; i++)
            for(int j = 0; j < b.len; j++)
            {
                c.num[i+j] += num[i] * b.num[j];
                c.num[i+j+1] += c.num[i+j] / BASE;
                c.num[i+j] %= BASE;
            }
        if(!c.num[c.len-1] && c.len > 1)c.len--;
        return c;
    }
    bigint operator * (int b)
    {
        bigint c;
        for(int i = 0; i < len; i++) c.num[i] = num[i] * b; // long long
        for(int i = 0; i < len; i++){c.num[i+1] += c.num[i] / BASE;c.num[i] %= BASE;}
        c.len = len;
        while(c.num[c.len])c.len++;
        return c;
    }
    bool substract(bigint b, int pos)
    {
        if(len < b.len - pos)return false;
        else if(len == b.len-pos)
            for(int i = len-1; i>=0; i--)
                if(num[i] < b.num[i+pos])return false;
                else if(num[i] > b.num[i+pos])break;
        for(int i = 0; i < len; i++)
        {
            num[i] -= b.num[i+pos];
            if(num[i] < 0){num[i] += BASE;num[i+1]--;}
        }
        while(!num[len-1] && len > 1)len--;
        return true;
    }
    // remember to change [BASE] to 10 !!!
    // [this] is the remainder
    bigint operator / (bigint b)
    {
        bigint c;
        if(len < b.len)return c;
        int k = len - b.len;
        c.len = k + 1;
        for(int i = len-1; i>=0; i--)
        {
            if(i>=k)b.num[i] = b.num[i-k];
            else b.num[i] = 0;
        }
        b.len = len;
        for(int i = 0; i <= k; i++) while(this->substract(b,i)) c.num[k-i]++;
        for(int i = 0; i < c.len; i++)
        {
            c.num[i+1] += c.num[i] / BASE;
            c.num[i] %= BASE;
        }
        while(!c.num[c.len-1] && c.len > 0) c.len--;
        return c;
    }
    // [this] is not the remainder
    bigint operator / (int b)
    {
        bigint c; int tmp = 0;
        for(int i = len-1; i>=0; i--)
        {
            tmp = tmp * BASE + num[i];
            c.num[i] = tmp / b;
            tmp %= b;
        }
        for(c.len = len; !c.num[c.len-1] && c.len > 1; c.len--);
        return c;
    }
    bool scan()
    {
        int n = -1; char ch = getchar();
        while(ch < '0' || ch > '9') if(ch == EOF) return false; else ch = getchar();
        while(ch >= '0' && ch <= '9') s[++n] = ch - '0', ch = getchar();
        len = 0;
        for(int i = n; i >= 0; i-=4)
        {
            num[len] += s[i];
            if(i>=1)num[len] += s[i-1] * 10;
            if(i>=2)num[len] += s[i-2] * 100;
            if(i>=3)num[len] += s[i-3] * 1000;
            ++len;
        }
        return true;
```

```
    }
    void clr(){memset(num,0,sizeof(num));}
    void print()
    {
        printf("%d",num[len-1]);
        for(int i = len-2; i>=0; i--) printf("%04d",num[i]);
        printf("\n");
    }
};
int main(){}
```

4.8 线性基-gwx

```
ll solve()
{
    ll res = 0;
    memset(b, 0, sizeof(b));
    for(int i = 1; i <= tot; i++)
        for(int j = 60; j >= 0; j--)
            if((a[i] >> j) & 1)
            {
                if(!b[j])
                {
                    b[j] = a[i];
                    break;
                }
                a[i] ^= b[j];
            }
    for(int i = 60; i >= 0; i--)
        res = max(res, res ^ b[i]);
    return res;
}
```

4.9 线性递推

```
// O(m^2logn)
// Given a[0], a[1], ..., a[m - 1]
// a[n] = c[0] * a[n - m] + ... + c[m - 1] * a[n - 1]
// Solve for a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * 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);
    }
    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];
    }
}
```

4.10 单纯形

```
// max{c * x | Ax <= b, x >= 0}的解，无解返回空的vector，否则就是解．答案在an中
template <int MAXN = 100, int MAXM = 100>
struct simplex {
    int n, m; double a[MAXM][MAXN], b[MAXM], c[MAXN];
    bool infeasible, unbounded;
    double v, an[MAXN + MAXM]; int q[MAXN + MAXM];
    void pivot (int l, int e) {
        std::swap (q[e], q[l + n]);
        double t = a[l][e]; a[l][e] = 1; b[l] /= t;
        for (int i = 0; i < n; ++i) a[l][i] /= t;
        for (int i = 0; i < m; ++i) if (i != l && std::abs (a[i][e]) > EPS) {
            t = a[i][e]; a[i][e] = 0; b[i] -= t * b[l];
            for (int j = 0; j < n; ++j) a[i][j] -= t * a[l][j]; }
        if (std::abs (c[e]) > EPS) {
            t = c[e]; c[e] = 0; v += t * b[l];
            for (int j = 0; j < n; ++j) c[j] -= t * a[l][j]; } }
    bool pre () {
        for (int l, e; ; ) {
            l = e = -1;
            for (int i = 0; i < m; ++i) if (b[i] < -EPS && (!~l || rand () & 1)) l = i;
            if (!~l) return false;
            for (int i = 0; i < n; ++i) if (a[l][i] < -EPS && (!~e || rand () & 1)) e = i;
            if (!~e) return infeasible = true;
            pivot (l, e); } }
    double solve () {
        double p; std::fill (q, q + n + m, -1);
        for (int i = 0; i < n; ++i) q[i] = i;
        v = 0; infeasible = unbounded = false;
        if (pre ()) return 0;
        for (int l, e; ; pivot (l, e)) {
            l = e = -1; for (int i = 0; i < n; ++i) if (c[i] > EPS) { e = i; break; }
            if (!~e) break; p = INF;
            for (int i = 0; i < m; ++i) if (a[i][e] > EPS && p > b[i] / a[i][e])
                p = b[i] / a[i][e], l = i;
            if (!~l) return unbounded = true, 0; }
        for (int i = n; i < n + m; ++i) if (~q[i]) an[q[i]] = b[i - n];
        return v; } };
```

4.11 素数测试-gwx

```
ll multi(ll x, ll y, ll M) {
    ll res = 0;
    for(; y; y >>= 1, x = (x + x) % M)
        if(y & 1) res = (res + x) % M;
    return res;
}
ll power(ll x, ll y, ll p)
{
    ll res = 1;
    for(; y; y >>= 1, x = multi(x, x, p))
        if(y & 1) res = multi(res, x, p);
    return res;
}
int primetest(ll n, int base)
{
    ll n2 = n - 1, res;
    int s = 0;
    while(!(n2 & 1)) n2 >>= 1, s++;
    res = power(base, n2, n);
    if(res == 1 || res == n - 1) return 1;
    s--;
    while(s >= 0)
    {
        res = multi(res, res, n);
        if(res == n - 1) return 1;
        s--;
    }
    return 0;   // n is not a strong pseudo prime
}
int isprime(ll 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(!primetest(n, testNum[i])) return 0;
    }
    return 1;
}
ll pollard(ll n)
{
    ll i, x, y, p;
    if(isprime(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;
    }
```

```
}
```

4.12 原根-gwx

```cpp
bool check_force(int g, int p)
{
    int cnt = 0, prod = g;
    for(int i = 1; i <= p - 1; ++i, prod = prod * g % p)
        if(prod == 1) if(++cnt > 1) return 0;
    return 1;
}
//d[]: prime divisor of (p - 1)
bool check_fast(int g, int p)
{
    for(int i = 1; i <= m; ++i)
        if(power(g, (p - 1) / d[i], p) == 1) return 0;
    return 1;
}
int primitive_root(int p)
{
    for(int i = 2; i < p; ++i) if(check(i, p)) return i;
}
```

4.13 勾股数

$a = m^2 - n^2, b = 2mn, c = m^2 + n^2$, 其中 m 和 n 中有一个是偶数, 则 (a, b, c) 是素勾股数

4.14 Pell 方程

Find the smallest integer root of $x^2 - ny^2 = 1$ when n is not a square number, with the solution set $x_{k+1} = x_0x_k + ny_0y_k, y_{k+1} = x_0y_k + y_0x_k$.

```cpp
template <int MAXN = 100000>
struct pell {
    std::pair <long long, long long> solve (long long n) {
        static long long p[MAXN], q[MAXN], g[MAXN], h[MAXN], a[MAXN];
        p[1] = q[0] = h[1] = 1; p[0] = q[1] = g[1] = 0;
        a[2] = (long long) (floor (sqrtl (n) + 1e-7L));
        for (int i = 2; ; ++i) {
            g[i] = -g[i - 1] + a[i] * h[i - 1];
            h[i] = (n - g[i] * g[i]) / h[i - 1];
            a[i + 1] = (g[i] + a[2]) / h[i];
            p[i] = a[i] * p[i - 1] + p[i - 2];
            q[i] = a[i] * q[i - 1] + q[i - 2];
            if (p[i] * p[i] - n * q[i] * q[i] == 1)
                return { p[i], q[i] }; } } };
```

4.15 平方剩余

Solve $x^2 \equiv n \mod p (0 \le a < p)$ where p is prime in $O(\log p)$.

```cpp
struct quadric {
    void multiply(long long &c, long long &d, long long a, long long b, long long w, long long p) {
        int cc = (a * c + b * d % p * w) % p;
        int dd = (a * d + b * c) % p; c = cc, d = dd; }
    bool solve(int n, int p, int &x) {
        if (n == 0) return x = 0, true; if (p == 2) return x = 1, true;
        if (power (n, p / 2, p) == p - 1) return false;
        long long c = 1, d = 0, b = 1, a, w;
        do { a = rand() % p; w = (a * a - n + p) % p;
            if (w == 0) return x = a, true;
        } while (power (w, p / 2, p) != p - 1);
        for (int times = (p + 1) / 2; times; times >>= 1) {
            if (times & 1) multiply (c, d, a, b, w, p);
            multiply (a, b, a, b, w, p); }
        return x = c, true; } };
```

4.16 多点求值与快速插值

4.16.1 多点求值与快速插值

多点求值: 给出 $F(x)$ 和 $x_1, \cdots, x_n$, 求 $F(x_1), \cdots, F(x_n)$.
考虑分治, 设 $L(x) = \prod_{i=1}^{\lfloor n/2 \rfloor}(x - x_i)$, $R(x) = \prod_{i=\lfloor n/2 \rfloor+1}^{n}(x - x_i)$, 那么对于 $1 \le i \le \lfloor n/2 \rfloor$ 有 $F(x_i) = (F \mod L)(x_i)$, 对于 $\lfloor n/2 \rfloor < i \le n$ 有 $F(x_i) = (F \mod R)(x_i)$. 复杂度 $O(n \log^2 n)$.

快速插值: 给出 n 个 x_i 与 y_i, 求一个 $n-1$ 次多项式满足 $F(x_i) = y_i$.
考虑拉格朗日插值: $F(x) = \sum_{i=1}^{n} \frac{\prod_{i \ne j}(x - x_j)}{\prod_{i \ne j}(x_i - x_j)} y_i$.
对每个 i 先求出 $\prod_{i \ne j}(x_i - x_j)$. 设 $M(x) = \prod_{i=1}^{n}(x - x_i)$, 那么想要的是 $\frac{M(x)}{x - x_i}$. 取 $x = x_i$ 时, 上下都为 0, 使用洛必达法则, 则原式化为 $M'(x)$. 使用分治算出 $M(x)$, 使用多点求值算出每个 $\prod_{i \ne j}(x_i - x_j) = M'(x_i)$.
设 $\frac{y_i}{\prod_{i \ne j}(x_i - x_j)} = v_i$, 现在要求出 $\sum_{i=1}^{n} v_i \prod_{i \ne j}(x - x_j)$. 使用分治: 设 $L(x) = \prod_{i=1}^{\lfloor n/2 \rfloor}(x - x_i)$, $R(x) = \prod_{i=\lfloor n/2 \rfloor+1}^{n}(x - x_i)$, 则原式化为: $\left(\sum_{i=1}^{\lfloor n/2 \rfloor} v_i \prod_{i \ne j, j \le \lfloor n/2 \rfloor}(x - x_j)\right) R(x) + \left(\sum_{i=\lfloor n/2 \rfloor+1}^{n} v_i \prod_{i \ne j, j > \lfloor n/2 \rfloor}(x - x_j)\right) L(x)$, 递归计算. 复杂度 $O(n \log^2 n)$.

4.17 多项式牛顿法

4.17.1 多项式牛顿法

已知函数 $G(x)$, 求多项式 $F(x) \mod x^n$ 满足方程 $G(F(x)) \equiv 0 \mod x^n$.

当 $n = 1$ 时, 有 $G(F(x)) \equiv 0 \mod x$, 根据实际情况 (逆元, 二次剩余) 求解. 假设已经求出了 $G(F_0(x)) \equiv 0 \mod x^n$, 考虑扩展到 $\mod x^{2n}$ 下: 将 $G(F(x))$ 在 $F_0(x)$ 点泰勒展开, 有

$$G(F(x)) = G(F_0(x)) + \frac{G'(F_0(x))}{1!}(F(x) - F_0(x)) + \cdots$$

因为 $F(x)$ 和 $F_0(x)$ 的最后 n 项相同, 所以 $(F(x) - F_0(x))^2$ 的最低的非 0 项次数大于 $2n$, 经过推导可以得到

$$F(x) \equiv F_0(x) - \frac{G(F_0(x))}{G'(F_0(x))} \mod x^{2n}$$

应用 (以下复杂度均为 $O(n \log n)$) :

多项式求逆 (给定 $A(x)$, 求 $A(x)B(x) \equiv 1 \mod x^n$) : 构造方程 $A(x)B(x) - 1 \equiv 0 \mod x^n$, 初始解 $G_{invA}(B(x)) \equiv A[0]^{-1} \mod x$, 递推式 $F(x) \equiv 2F_0(x) - A(x)F_0^2(x) \mod x^{2n}$

多项式开方 (给定 $A(x)$, 求 $B^2(x) \equiv A(x) \mod x^n$) : 初始解 $G_{sqrtA}(B(x)) \equiv \sqrt{A[0]} \mod x$, 递推式 $F(x) \equiv \frac{F_0^2(x) + A(x)}{2F_0(x)} \mod x^{2n}$

多项式对数 (给定常数项为 1 的 $A(x)$, $B(x) \equiv \ln A(x)$) : 对 x 求导得 $(\ln A(x))' = \frac{A'(x)}{A(x)}$, 使用多项式求逆, 再积分回去 $\ln A(x) \equiv \int \frac{A'(x)}{A(x)}$

多项式指数 (给定常数项为 0 的 $A(x)$, 求 $B(x) \equiv e^{A(X)}$) : 初始解 $G_{expA}(B(x)) \equiv 1$, 递推式 $F(x) \equiv F_0(x)(1 - \ln F_0(x) + A(x))$

多项式任意幂次 (给定 $A(x)$, 求 $B(x) \equiv A^k(x), k \in Q$) : $A^k(x) \equiv e^{k \ln A(x)}$

5 字符串

5.1 AC 自动机-wrz

```cpp
struct ACAM
{
    ACAM *next[S], *fail;
    int ban;
}mem[N], *tot, *null, *root, *q[N];
ACAM *newACAM()
{
    ACAM *p = ++tot;
    *p = *null; return p;
}
void init()
{
    null = tot = mem;
    for(int i = 0; i < alpha; i++) null->next[i] = null;
    null->fail = null; null->ban = 0;
    root = newACAM();
}
void inser(char *s)
{
    ACAM *p = root;
    for(int i = 0; s[i]; i++)
    {
        int w = s[i] - 'a';
        if(p->next[w] == null) p->next[w] = newACAM();
        p = p->next[w];
    }
    p->ban = 1;
}
void build()
{
    root->fail = root; int head = 0, tail = 0;
    for(int i = 0; i < alpha; i++)
    {
        if(root->next[i] == null) root->next[i] = root;
        else root->next[i]->fail = root, q[tail++] = root->next[i];
    }
    for(; head < tail; head++)
    {
        ACAM *p = q[head];
        p->ban |= p->fail->ban;
        for(int i = 0; i < alpha; i++)
        {
            if(p->next[i] == null) p->next[i] = p->fail->next[i];
            else p->next[i]->fail = p->fail->next[i], q[tail++] = p->next[i];
        }
    }
}
```

5.2 扩展 KMP-gwx

```cpp
void get_next()
{
    int a = 0, p = 0;
    nxt[0] = m;
    for(int i = 1; i < m; i++)
    {
        if(i >= p || i + nxt[i - a] >= p)
        {
            if(i >= p)  p = i;
            while(p < m && t[p] == t[p - i]) p++;
            nxt[i] = p - i;
            a = i;
        }
        else nxt[i] = nxt[i - a];
    }
}

void exkmp()
{
    int a = 0, p = 0;
    get_next();
    for(int i = 0; i < n; i++)
    {
        if(i >= p || i + nxt[i - a] >= p) // i >= p 的作用：举个典型例子，s 和 t 无一字符相同
        {
            if(i >= p) p = i;
            while(p < n && p - i < m && s[p] == t[p - i]) p++;
            ext[i] = p - i;
            a = i;
        }
        else ext[i] = nxt[i - a];
    }
}
```

5.3 Manacher-gwx

```cpp
//maxn = 2 * n
void manacher(int n)
{
    int p = 0, r = 0;
    for(int i = 1; i <= n; i++)
    {
        if(i <= r) len[i] = min(len[2 * p - i], r - i + 1);
        else len[i] = 1;
        while(b[i + len[i]] == b[i - len[i]]) len[i]++;
        if(i + len[i] - 1 >= r)
            r = i + len[i] - 1, p = i;
    }
}

int main()
{
    scanf("%d\n%s", &n, a + 1);
    b[++tot] = '@'; b[++tot] = '#';
    for(int i = 1; i < n; i++)
        b[++tot] = a[i], b[++tot] = '#';
    b[++tot] = a[n];
    b[++tot] = '#'; b[++tot] = '$';
    manacher(tot);
}
```

5.4 最小表示-gwx

```cpp
//不保证起始位置最靠前(?)
string find(int N, string s) {
    int i, j, k, l;
    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;
        else l = i + k, i = j, j = max(l, j) + 1;
    }
    return s.substr(i, N);
}
```

5.5 回文树-wrz

```cpp
char s[N], out[N];
struct PT
{
    PT *fail, *next[A];
    int len;
}mem[N], *tot, *null, *root1, *root0, *last;
PT *newPT()
{
    PT *p = ++tot;
    *p = *null; return p;
}
void init()
{
    null = tot = mem;
    null->fail = null;
    for(int i = 0; i < A; i++) null->next[i] = null;
    null->len = 0;
    root1 = newPT(); root1->fail = root1; root1->len = -1;
    root0 = newPT(); root0->fail = root1; last = root1;
}
int extend(int c, int i) // 返回这一次是否多了一个回文子串
{
    PT *p = last;
    for(; s[i-p->len-1] != c+'a'; p = p->fail);
    if(p->next[c] != null) {last = p->next[c]; return 0;}
    PT *np = p->next[c] = last = newPT(); np->len = p->len + 2;
    if(p->len == -1) np->fail = root0;
    else
    {
        for(p=p->fail; s[i-p->len-1] != c+'a'; p = p->fail);
        np->fail = p->next[c];
    }
    return 1;
}
void main()
{
    scanf("%s",s+1); init();
    for(int i = 1, ii = strlen(s+1); i <= ii; i++)
        out[i] = extend(s[i]-'a', i)?'1':'0';
    puts(out+1);
}
```

5.6 后缀数组-wrz

```cpp
// 对都是数字的数组做SA时要保证数组中没有0，否则height等可能由于s[0]=s[n+1]=0出问题
// 多次使用要保证s[0]=s[n+1]=0
char s[N];
int n, t1[N], t2[N], sa[N], rank[N], sum[N], height[N], lef, rig; // 数组开两倍
void SA_build()
{
    int *x = t1, *y = t2, m = 30;
    for(int i = 1; i <= n; i++) sum[x[i] = s[i] - 'a' + 1]++;
    for(int i = 1; i <= m; i++) sum[i] += sum[i-1];
    for(int i = n; i >= 1; i--) sa[sum[x[i]]--] = i;
    for(int k = 1; k <= n; k <<= 1)
    {
        int p = 0;
        for(int i = n-k+1; i <= n; i++) y[++p] = i;
        for(int i = 1; i <= n; i++) if(sa[i] - k > 0)
            y[++p] = sa[i] - k;

        for(int i = 1; i <= m; i++) sum[i] = 0;
        for(int i = 1; i <= n; i++) sum[x[i]]++;
        for(int i = 1; i <= m; i++) sum[i] += sum[i-1];
        for(int i = n; i >= 1; i--) sa[sum[x[y[i]]]--] = y[i];

        swap(x, y);
        for(int i = 1; i <= n; i++)
            x[sa[i]] = x[sa[i-1]] + (y[sa[i]] == y[sa[i-1]] && y[sa[i]+k] == y[sa[i-1]+k] ? 0 : 1);
        m = x[sa[n]];
        if(m == n) break;
    }
    for(int i = 1; i <= n; i++) rank[sa[i]] = i;
    for(int i = 1, k = 0; i <= n; height[rank[i++]] = k?k--:k)
        for(; s[i+k] == s[sa[ran[i]-1]+k] && i+k <= n && sa[ran[i]-1]+k <= n; k++);
}
```

5.7 后缀数组 SAIS

```cpp
// string is 0-based
// sa[] is 1-based
// s[n] < s[i] i = 0...n-1
namespace SA {
    int sa[MAXN], rk[MAXN], ht[MAXN], s[MAXN << 1], t[MAX << 1], p[MAXN], cnt[MAXN], cur[MAXN];
#define pushS(x) sa[cur[s[x]]--] = x
#define pushL(x) sa[cur[s[x]]++] = x
#define inducedSort(v) std::fill_n(sa, n, -1); std::fill_n(cnt, m, 0);\
    for (int i = 0; i < n; i++) cnt[s[i]]++;\
    for (int i = 1; i < m; i++) cnt[i] += cnt[i-1];\
    for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;\
    for (int i = n1-1; ~i; i--) pushS(v[i]);\
    for (int i = 1; i < m; i++) cur[i] = cnt[i-1];\
    for (int i = 0; i < n; i++) if (sa[i] > 0 &&  t[sa[i]-1]) pushL(sa[i]-1);\
    for (int i = 0; i < m; i++) cur[i] = cnt[i]-1;\
    for (int i = n-1;  ~i; i--) if (sa[i] > 0 && !t[sa[i]-1]) pushS(sa[i]-1)
    void sais(int n, int m, int *s, int *t, int *p) {
```

```
        int n1 = t[n-1] = 0, ch = rk[0] = -1, *s1 = s+n;
        for (int i = n-2; ~i; i--) t[i] = s[i] == s[i+1] ? t[i+1] : s[i] > s[i+1];
        for (int i = 1; i < n; i++) rk[i] = t[i-1] && !t[i] ? (p[n1] = i, n1++) : -1;
        inducedSort(p);
        for (int i = 0, x, y; i < n; i++) if (~(x = rk[sa[i]])) {
            if (ch < 1 || p[x+1] - p[x] != p[y+1] - p[y]) ch++;
            else for (int j = p[x], k = p[y]; j <= p[x+1]; j++, k++)
                if ((s[j]<<1|t[j]) != (s[k]<<1|t[k])) {ch++; break;}
            s1[y = x] = ch; }
        if (ch+1 < n1) sais(n1, ch+1, s1, t+n, p+n1);
        else for (int i = 0; i < n1; i++) sa[s1[i]] = i;
        for (int i = 0; i < n1; i++) s1[i] = p[sa[i]];
        inducedSort(s1); }
    int mapCharToInt(int n, const T *str) {
        int m = *std::max_element(str, str+n);
        std::fill_n(rk, m+1, 0);
        for (int i = 0; i < n; i++) rk[str[i]] = 1;
        for (int i = 0; i < m; i++) rk[i+1] += rk[i];
        for (int i = 0; i < n; i++) s[i] = rk[str[i]] - 1;
        return rk[m]; }
    void suffixArray(int n, const T *str) {
        int m = mapCharToInt(++n, str);
        sais(n, m, s, t, p);
        for (int i = 0; i < n; i++) rk[sa[i]] = i;
        for (int i = 0, h = ht[0] = 0; i < n-1; i++) {
            int j = sa[rk[i]-1];
            while (i+h < n && j+h < n && s[i+h] == s[j+h]) h++;
            if (ht[rk[i]] = h) h--; } } };
```

5.8 后缀自动机-wrz

```
struct SAM
{
    SAM *next[A], *fail;
    int len, mi, mx;
}mem[N], *tot, *null, *root, *last, *q[N];
SAM *newSAM(int len)
{
    SAM *p = ++tot;
    *p = *null;
    p->len = p->mi = len;
    p->mx = 0;
    return p;
}
void init()
{
    null = tot = mem;
    for(int i = 0; i < A; i++) null->next[i] = null;
    null->fail = null;
    null->len = null->mi = null->mx = 0;
    root = last = newSAM(0);
}
void extend(int v)
{
    SAM *p = last, *np = newSAM(p->len + 1); last = np;
    for(; p->next[v] == null && p != null; p = p->fail) p->next[v] = np;
    if(p==null) np->fail = root;
    else
    {
        SAM *q = p->next[v];
        if(q->len == p->len+1) np->fail = q;
        else
        {
            SAM *nq = newSAM(p->len+1);
            memcpy(nq->next, q->next, sizeof(nq->next));
            nq->fail = q->fail;
            q->fail = np->fail = nq;
            for(; p->next[v] == q && p != null; p = p->fail) p->next[v] = nq;
        }
    }
}
```

5.9 扩展后缀自动机-wrz

```
struct sam
{
    sam *fail, *next[A];
    int len;
}mem[N<<1], *tot, *null, *root;
sam* newsam()
{
    *++tot = *null;
    return tot;
}
void init()
{
    null = tot = mem; null->fail = null; null->len = 0;
    for(int i = 0; i < A; i++) null->next[i] = null;
    root = newsam();
}
sam* extend(sam *p, int v)
{
    if(p->next[v] != null)
    {
        sam *q = p->next[v];
        if(p->len + 1 == q->len) return q;
        else
        {
            sam *nq = newsam(); *nq = *q; nq->len = p->len + 1;
            q->fail = nq;
            for(; p->next[v] == q && p != null; p = p->fail) p->next[v] = nq;
            return nq;
        }
    }
    else
    {
        sam *np = newsam(); np->len = p->len + 1;
        for(; p->next[v] == null && p != null; p = p->fail) p->next[v] = np;
        if(p == null) np->fail = root;
        else
        {
            sam *q = p->next[v];
            if(p->len + 1 == q->len) np->fail = q;
            else
            {
                sam *nq = newsam(); *nq = *q; nq->len = p->len + 1;
                np->fail = q->fail = nq;
                for(; p->next[v] == q && p != null; p = p->fail) p->next[v] = nq;
            }
        }
        return np;
    }
}
void build_tree()
{
    for(sam *i = tot; i != mem; i--)
        addedge(i->fail - mem, i - mem);
}
```

5.10 结论

5.10.1 双回文串

如果 $s = x_1x_2 = y_1y_2 = z_1z_2, |x_1| < |y_1| < |z_1|, x_2, y_1, y_2, z_1$ 是回文串, 则 x_1 和 z_2 也是回文串.

5.10.2 Border 的结构

字符串 s 的所有不小于 $|s|/2$ 的 border 长度组成一个等差数列.
字符串 s 的所有 border 按长度排序后可分成 $O(\log|s|)$ 段, 每段是一个等差数列. 回文串的回文后缀同时也是它的 border.

5.10.3 子串最小后缀

设 $s[p..n]$ 是 $s[i..n], (l \le i \le r)$ 中最小者, 则 minsuf(l, r) 等于 $s[p..r]$ 的最短非空 border. minsuf(l, r) $= \min\{s[p..r], \text{minsuf}(r - 2^k + 1, \text{r})\}$, $(2^k < r - l + 1 \le 2^{k+1})$.

5.10.4 子串最大后缀

从左往右扫, 用 set 维护后缀的字典序递减的单调队列, 并在对应时刻添加"小于事件"点以便在之后修改队列; 查询直接在 set 里 lower_bound.

6 其他

6.1 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);
}
```

6.2 dancing-links

```
struct Node {
    Node *l, *r, *u, *d, *col;
    int size, line_no;
    Node() {
        size = 0; line_no = -1;
        l = r = u = d = col = NULL;
    }
} *root;
void cover(Node *c) {
    c->l->r = c->r; c->r->l = c->l;
    for (Node *u = c->d; u != c; u = u->d)
        for (Node *v = u->r; v != u; v = v->r) {
            v->d->u = v->u;
            v->u->d = v->d;
            -- v->col->size;
        }
}
void uncover(Node *c) {
    for (Node *u = c->u; u != c; u = u->u) {
        for (Node *v = u->l; v != u; v = v->l) {
```

```cpp
            ++ v->col->size;
            v->u->d = v;
            v->d->u = v;
        }
    }
    c->l->r = c; c->r->l = c;
}
std::vector<int> answer;
bool search(int k) {
    if (root->r == root) return true;
    Node *r = NULL;
    for (Node *u = root->r; u != root; u = u->r)
        if (r == NULL || u->size < r->size)
            r = u;
    if (r == NULL || r->size == 0) return false;
    else {
        cover(r);
        bool succ = false;
        for (Node *u = r->d; u != r && !succ; u = u->d) {
            answer.push_back(u->line_no);
            for (Node *v = u->r; v != u; v = v->r)   // Cover row
                cover(v->col);
            succ |= search(k + 1);
            for (Node *v = u->l; v != u; v = v->l)
                uncover(v->col);
            if (!succ) answer.pop_back();
        }
        uncover(r);
        return succ;
    }
}
bool entry[CR][CC];
Node *who[CR][CC];
int cr, cc;
void construct() {
    root = new Node();
    Node *last = root;
    for (int i = 0; i < cc; ++ i) {
        Node *u = new Node();
        last->r = u; u->l = last;
        Node *v = u; u->line_no = i;
        last = u;
        for (int j = 0; j < cr; ++ j)
            if (entry[j][i]) {
                ++ u->size;
                Node *cur = new Node();
                who[j][i] = cur;
                cur->line_no = j;
                cur->col = u;
                cur->u = v; v->d = cur;
                v = cur;
            }
        v->d = u; u->u = v;
    }
    last->r = root; root->l = last;
    for (int j = 0; j < cr; ++ j) {
        Node *last = NULL;
        for (int i = cc - 1; i >= 0; -- i)
            if (entry[j][i]) {
                last = who[j][i];
                break;
            }
        for (int i = 0; i < cc; ++ i)
            if (entry[j][i]) {
                last->r = who[j][i];
                who[j][i]->l = last;
                last = who[j][i];
            }
    }
}
void destruct() {
    for (Node *u = root->r; u != root; ) {
        for (Node *v = u->d; v != u; ) {
            Node *nxt = v->d;
            delete(v);
            v = nxt;
        }
        Node *nxt = u->r;
        delete(u); u = nxt;
    }
    delete root;
}
```

6.3 枚举子集

```cpp
for (int x = 1; x <= n; x++)
    for (int y = x & (x - 1); y; (--y) &= x) {
        //y is a subset of x
    }
```

6.4 梅森旋转

```cpp
#include <random>
int main() {
    std::mt19937 g(seed);  // std::mt19937_64
    std::cout << g() << std::endl;
}
```

6.5 大数乘法取模

```cpp
// 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;
}
```

7 提示

7.1 Vimrc

```vim
set ru nu ts=4 sts=4 sw=4 si sm hls is ar bs=2 mouse=a
syntax on
nm <F3> :vsplit %<.in <CR>
nm <F4> :!gedit % <CR>
au BufEnter *.cpp set cin
au BufEnter *.cpp nm <F5> :!time ./%< <CR>|nm <F7> :!gdb ./%< <CR>|nm <F8> :!time ./%< < %<.in <CR>|nm <F9> :!g++ % -o %< -g -std=gnu++14 -O2 -DLOCAL -Wall -Wconversion && size %< <CR>
au BufEnter *.java nm <F5> :!time java %< <CR>|nm <F8> :!time java %< < %<.in <CR>|nm <F9> :!javac % <CR>
```

7.2 make 支持 c++11

```bash
export CXXFLAGS='-std=c++11 -Wall'
source .bashrc
```

7.3 Java

```java
import java.util.*;
import java.math.*;
public class javaNote
{
    static BigInteger q[] = new BigInteger[5000000]; // 定义数组的正确姿势，记得分配内存

    public static void main(String[] args)
    {

        long currentTime = System.currentTimeMillis(); // 获取时间，单位是ms

        Scanner sc = new Scanner(System.in); // 定义输入
        int a = sc.nextInt(), b;
        System.out.println("integer = " + a); // 输出

        BigInteger x = new BigInteger("233"), y = new BigInteger("666");
        BigInteger.valueOf(1); // 将指定的表达式转化成BigInteger类型
        x.add(y); //x+y
        x.subtract(y); //x-y
        x.multiply(y); //x*y
        x.divide(y);

        x.pow(233); // x**233
        x.compareTo(y); // 比较x和y, x < y : -1, x = y : 0, x > y : 1

        BigDecimal n = new BigDecimal("233"), m = new BigDecimal("666");
        n.divide(m,a,RoundingMode.DOWN); //n/m并精确到小数点后第a位，a=0表示精确到个位，a为负数表示精确到小数点前-a+1位，可能变成科学计数法
        /*
            取整方式
            RoundingMode.CEILING: 取右边最近的整数，即向正无穷取整
            RoundingMode.FLOOR: 取左边最近的整数，即向负无穷取整
            RoundingMode.DOWN: 向0取整
            RoundingMode.UP: 远离0取整
            RoundingMode.HALF_UP:上取整的四舍五入，>=0.5会进位，<0.5会舍去，负数会先取绝对值再四舍五入再变回负数
            RoundingMode.HALF_DOWN:下取整的四舍五入，>0.5会进位，<=0.5会舍去，负数原理同上
            RoundingMode.HALF_EVEN:分奇偶的四舍五入，>0.5会进位，<0.5会舍去，=0.5会向最近的偶数取整，如2.5->2，(-2.5)->(-2)
        */

        Math.max(a, b);//取大
        Math.min(a, b);//取小
        Math.PI;//pi

        HashSet<BigInteger> hash = new HashSet<BigInteger>(); // hash table
        hash.contains(x); // hash table中是否有a，有则返回true，反之返回false
        hash.add(x); // 把x加进hash table
```

```
        hash.remove(x); // 从hash table中删去x

        Arrays.sort(arr, 1, n+1); // arr 是需要排序的数组，后两个参数分别是排序的起始位置和结束位置+1，还可以有第四个参数是比较函数
        // Arrays.sort(arr, a, b, cmp) = sort(arr+a, arr+b, cmp)
    }

    public static BigInteger sqrt (BigInteger x) {
    if (x.equals (BigInteger.ZERO) || x.equals (BigInteger.ONE)) return x;
    BigInteger d = BigInteger.ZERO.setBit (x.bitLength () / 2);
    BigInteger d2 = d;
    for (; ; ) {
        BigInteger y = d.add (x.divide (d)).shiftRight (1);
        if (y.equals (d) || y.equals (d2)) return d.min (d2);
        d2 = d; d = y; } }
}
```

7.4 cout 输出小数

```
std::cout << std::fixed << std::setprecision(5);
```

7.5 释放容器内存

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

7.6 tuple

```
mytuple = std::make_tuple (10, 2.6, 'a');          // packing values into tuple
std::tie (myint, std::ignore, mychar) = mytuple;   // unpacking tuple into variables
std::get<I>(mytuple) = 20;
std::cout << std::get<I>(mytuple) << std::endl;    // get the Ith(const) element
```

7.7 读入优化 & 手开 O3

```
// 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
#define __ __attribute__ ((optimize ("-O3")))
#define _ __ __inline __attribute__ ((__gnu_inline__, __always_inline__, __artificial__))
namespace Reader {
    const int L = (1 << 15) + 5;
    char buffer[L], *S, *T;
    __inline bool getchar(char &ch) {
        if (S == T) {
            T = (S = buffer) + fread(buffer, 1, L, stdin);
            if (S == T) {
                ch = EOF;
                return false;
            }
        }
        ch = *S++;
        return true;
    }
    __inline bool getint(int &x) {
        char ch; bool neg = 0;
        for (; getchar(ch) && (ch < '0' || ch > '9'); ) neg ^= ch == '-';
        if (ch == EOF) return false;
        x = ch - '0';
        for (; getchar(ch), ch >= '0' && ch <= '9'; )
            x = x * 10 + ch - '0';
        if (neg) x = -x;
        return true;
    }
}
```

7.8 手开栈

The following lines allow the program to use larger stack memory.

```
//C++
#pragma comment (linker, "/STACK:36777216")
//G++
int __size__ = 256 << 20;
char *__p__ = (char*) malloc(__size__) + __size__;
__asm__ ("movl␣%0,␣%%esp\n" :: "r"(__p__));
```

8 附录-数学公式

8.1

8.1.1 Mobius Inversion

$$F(n)=\sum_{d|n} f(d) \Rightarrow f(n)=\sum_{d|n}\mu(d)F(\frac{n}{d})$$

$$F(n)=\sum_{n|d} f(d) \Rightarrow f(n)=\sum_{n|d}\mu(\frac{d}{n})F(d)$$

$$[x=1]=\sum_{d|x}\mu(d),\quad x=\sum_{d|x}\mu(d)$$

8.1.2 Arithmetic Function

$$(p-1)! \equiv -1(mod\ p)$$

$$a>1, m,n>0, \text{ then } \gcd(a^m-1,a^n-1)=a^{\gcd(n,m)}-1$$

$$a>b, \gcd(a,b)=1, \text{ then } \gcd(a^m-b^m,a^n-b^n)=a^{\gcd(m,n)}-b^{\gcd(m,n)}$$

$$\prod_{k=1,gcd(k,m)=1}^{m} k \equiv \begin{cases} -1 & \bmod\ m,\ m=4,p^q,2p^q \\ 1 & \bmod\ m, \text{otherwise} \end{cases}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8.1.3 Binomial Coefficients

$$\binom{n}{k}=(-1)^k\binom{k-n-1}{k},\ \sum_{k\le n}\binom{r+k}{k}=\binom{r+n+1}{n}$$

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

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

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

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

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

$$\binom{n_1+\cdots+n_p}{m}=\sum_{k_1+\cdots+k_p=m}\binom{n_1}{k_1}\cdots\binom{n_p}{k_p}$$

8.1.4 Fibonacci Numbers

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

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

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

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

$$\frac{f_{2n}}{f_n}=f_{n-1}+f_{n+1}$$

$$f_1+2f_2+3f_3+\cdots+nf_n=nf_{n+2}-f_{n+3}+2]$$

$$\gcd(f_m,f_n)=f_{\gcd(m,n)}$$

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

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

$$f_{2n+1}=f_n^2+f_{n+1}^2$$

$$(-1)^kf_{n-k}=f_nf_{k-1}-f_{n-1}f_k$$

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

8.1.5 Lucas Numbers

$$L_0=2, L_1=1, L_n=L_{n-1}+L_{n-2}=(\frac{1+\sqrt{5}}{2})^n+\frac{1-\sqrt{5}}{2})^n$$

$$L(x)=\frac{2-x}{1-x-x^2}$$

8.1.6 Catlan Numbers

$$c_1=1, c_n=\sum_{i=0}^{n-1}c_ic_{n-1-i}=c_{n-1}\frac{4n-2}{n+1}=\frac{\binom{2n}{n}}{n+1}=\binom{2n}{n}-\binom{2n}{n-1}$$

$$c(x)=\frac{1-\sqrt{1-4x}}{2x}$$

Usage: n 对括号序列; n 个点满二叉树; $n*n$ 的方格左下到右上不过对角线方案数; 凸 $n+2$ 边形三角形分割数; n 个数的出栈方案数; $2n$ 个顶点连接, 线段两两不交的方案数.

8.1.7 Stirling Cycle Numbers

把 n 个元素集合分作 k 个**非空环**方案数.

$$s(n,0)=0, s(n,n)=1, s(n+1,k)=s(n,k-1)-nS(n,k)$$

$$s(n,k)=(-1)^{n-k}\begin{bmatrix}n\\k\end{bmatrix}$$

$$\begin{bmatrix}n+1\\k\end{bmatrix}=n\begin{bmatrix}n\\k\end{bmatrix}+\begin{bmatrix}n\\k-1\end{bmatrix},\ \begin{bmatrix}n+1\\2\end{bmatrix}=n!H_n$$

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

8.1.8 Stirling Subset Numbers

把 n 个元素集合分作 k 个**非空子集**方案数.

$$\begin{Bmatrix}n+1\\k\end{Bmatrix}=k\begin{Bmatrix}n\\k\end{Bmatrix}+\begin{Bmatrix}n\\k-1\end{Bmatrix}$$

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

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

For fixed k, generating functions :

$$\sum_{n=0}^{\infty}\begin{Bmatrix}n\\k\end{Bmatrix}x^{n-k}=\prod_{r=1}^{k}\frac{1}{1-rx}$$

8.1.9 Motzkin Numbers

圆上 n 点间画不相交弦的方案数. 选 n 个数 $k_1,k_2,...,k_n\in\{-1,0,1\}$, 保证 $\sum_i^a k_i(1\le a\le n)$ 非负且所有数总和为 0 的方案数.

$$M_{n+1}=M_n+\sum_{i}^{n-1}M_iM_{n-1-i}=\frac{(2n+3)M_n+3nM_{n-1}}{n+3}$$

$$M_n=\sum_{i=0}^{\lfloor\frac{n}{2}\rfloor}\binom{n}{2k}Catlan(k)$$

$$M(X)=\frac{1-x-\sqrt{1-2x-3x^2}}{2x^2}$$

8.1.10 Eulerian Numbers

$$\left\langle{n\atop k}\right\rangle=(k+1)\left\langle{n-1\atop k}\right\rangle+(n-k)\left\langle{n-1\atop k-1}\right\rangle$$

$$x^n=\sum_k\left\langle{n\atop k}\right\rangle\binom{x+k}{n}$$

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

8.1.11 Harmonic Numbers

$$\sum_{k=1}^{n}H_k=(n+1)H_n-n$$

$$\sum_{k=1}^{n}kH_k=\frac{n(n+1)}{2}H_n-\frac{n(n-1)}{4}$$

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

8.1.12 Pentagonal Number Theorem

$$\prod_{n=1}^{\infty}(1-x^n)=\sum_{n=-\infty}^{\infty}(-1)^kx^{k(3k-1)/2}$$

$$p(n)=p(n-1)+p(n-2)-p(n-5)-p(n-7)+\cdots$$

$$f(n,k)=p(n)-p(n-k)-p(n-2k)+p(n-5k)+p(n-7k)-\cdots$$

8.1.13 Bell Numbers

n 个元素集合划分的方案数.

$$B_n = \sum_{k=1}^{n} \left\{ {n \atop k} \right\}, \quad B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

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

$$B(x) = \sum_{n=0}^{\infty} \frac{B_n}{n!} x^n = \mathrm{e}^{\mathrm{e}^x - 1}$$

8.1.14 Bernoulli Numbers

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

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

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

8.1.15 Sum of Powers

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = (\frac{n(n+1)}{2})^2$$

$$\sum_{i=1}^{n} i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

$$\sum_{i=1}^{n} i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$$

8.1.16 Sum of Squares

$r_k(n)$ 表示用 k 个平方数组成 n 的方案数. 假设:

$$n = 2^{a_0} p_1^{2a_1} \cdots p_r^{2a_r} q_1 b_1 \cdots q_s b_s$$

其中 $p_i \equiv 3 \mod 4$, $q_i \equiv 1 \mod 4$, 那么

$$r_2(n) = \begin{cases} 0 & \text{if any } a_i \text{ is a half-integer} \\ 4\prod_{1}^{r}(b_i+1) & \text{if all } a_i \text{ are integers} \end{cases}$$

$r_3(n) > 0$ 当且仅当 n 不满足 $4^a(8b+7)$ 的形式 (a, b 为整数).

8.1.17 Pythagorean Triple

枚举 $x^2 + y^2 = z^2$ 的三元组: 可令 $x = m^2 - n^2, y = 2mn, z = m^2 + n^2$, 枚举 m 和 n 即可 $O(n)$ 枚举勾股数. 判断素勾股数方法: m, n 至少一个为偶数并且 m, n 互质, 那么 x, y, z 就是素勾股数.

8.1.18 Tetrahedron Volume

If U, V, W, u, v, w are lengths of edges of the tetrahedron (first three form a triangle; u opposite to U and so on)

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

8.1.19 杨氏矩阵与钩子公式

满足: 格子 (i,j) 没有元素, 则它右边和上边相邻格子也没有元素; 格子 (i,j) 有元素 $a[i][j]$, 则它右边和上边相邻格子要么没有元素, 要么有元素且比 $a[i][j]$ 大.

计数: $F_1 = 1, F_2 = 2, F_n = F_{n-1} + (n-1)F_{n-2}$, $F(x) = e^{x+\frac{x^2}{2}}$
钩子公式: 对于给定形状 λ, 不同杨氏矩阵的个数为:

$$d_\lambda = \frac{n!}{\prod h_\lambda(i,j)}$$

$h_\lambda(i,j)$ 表示该格子右边和上边的格子数量加 1.

8.1.20 重心

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

8.1.21 常见游戏

Anti-Nim 游戏 n 堆石子轮流拿, 谁走最后一步输。结论: 先手胜当且仅当 1. 所有堆石子数都为 1 且游戏的 SG 值为 0 (即有偶数个孤单堆-每堆只有 1 个石子数) 2. 存在某堆石子数大于 1 且游戏的 SG 值不为 0.

斐波那契博弈 有一堆物品, 两人轮流取物品, 先手最少取一个, 至多无上限, 但不能把物品取完, 之后每次取的物品数不能超过上一次取的物品数的二倍且至少为一件, 取走最后一件物品的人获胜. 结论: 先手胜当且仅当物品数 n 不是斐波那契数.

威佐夫博弈 有两堆石子, 博弈双方每次可以取一堆石子中的任意个, 不能不取, 或者取两堆石子中的相同个. 先取完者赢. 结论: 求出两堆石子 A 和 B 的差值 C, 如果 $\left\lfloor C * \frac{\sqrt{5}+1}{2} \right\rfloor = min(A, B)$ 那么后手赢, 否则先手赢.

约瑟夫环 令 n 个人标号为 $0, 1, 2, ..., n-1$, 令 $f_{i,m}$ 表示 i 个人报 m 胜利者的编号, 则 $f_{1,m} = 0, f_{i,m} = (f_{i-1,m} + m) mod\ i$.

8.1.22 错排公式

$$D_1 = 0, D_2 = 1, D_n = n!(\frac{1}{0!} - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + ... + \frac{(-1)^n}{n!})$$

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

8.1.23 概率相关

对于随机变量 X, 期望用 $E(X)$ 表示, 方差用 $D(X)$ 表示, 则 $D(X) = E(X - E(X))^2 = E(X^2) - (E(X))^2, D(X+Y) = D(X) + D(Y) D(aX) = a^2 D(X)$

$$E[x] = \sum_{i=1}^{\infty} P(X \geq i)$$

8.1.24 常用泰勒展开

$$f(x) = \frac{f(x_0)}{0!} + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots$$

$$\frac{1}{(1-x)^n} = 1 + \binom{n}{1}x + \binom{n+1}{2}x^2 + \binom{n+2}{3}x^3 + \cdots$$

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^i}{i!}$$

8.1.25 Others (某些近似数值公式在这里)

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

$$h_m = \sum_{1 \leq j_1 < \cdots < j_m \leq n} x_{j_1} \cdots x_{j_m}, \quad H_m = \sum_{1 \leq j_1 \leq \cdots \leq j_m \leq n} x_{j_1} \cdots x_{j_m}$$

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

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

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

$$\sum_{i=1}^{n} = ln(n) + \Gamma, (\Gamma \approx 0.577215664901532860606512090)$$

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

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

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

$$a^3 + b^3 = (a+b)(a^2 - ab + b^2), a^3 - b^3 = (a-b)(a^2 + ab + b^2)$$

$$a^n + b^n = (a+b)(a^{n-1} - a^{n-2}b + a^{n-3}b^2 - ... - ab^{n-2} + b^{n-1})(n\, mod\, 2 = 1$$

划分问题: n 个 $k-1$ 维向量最多把 k 维空间分为 $\sum_{i=0}^{k} C_n^i$ 份.

Binomial coefficients

$$\binom{n}{k} = (-1)^k \binom{k-n-1}{k}, \quad \sum_{k \leq n} \binom{r+k}{k} = \binom{r+n+1}{n}$$

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

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

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

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

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

$$\binom{n_1+\cdots+n_p}{m}=\sum_{k_1+\cdots+k_p=m}\binom{n_1}{k_1}\cdots\binom{n_p}{k_p}$$

Fibonacci numbers

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

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

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

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

$$\frac{f_{2n}}{f_n}=f_{n-1}+f_{n+1}$$

$$f_1+2f_2+3f_3+\cdots+nf_n=nf_{n+2}-f_{n+3}+2]$$

$$\gcd(f_m,f_n)=f_{\gcd(m,n)}$$

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

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

$$f_{2n+1}=f_n^2+f_{n+1}^2$$

$$(-1)^kf_{n-k}=f_nf_{k-1}-f_{n-1}f_k$$

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

Period modulo a prime p is a factor of $2p+2$ or $p-1$.
Only exception: $G(5)=20$.

Period modulo the power of a prime p^k: $G(p^k)=G(p)p^{k-1}$.

Period modulo $n=p_1^{k_1}...p_m^{k_m}$: $G(n)=lcm(G(p_1^{k_1}),...,G(p_m^{k_m}))$.

Lucas numbers

$$L_0=2, L_1=1, L_n=L_{n-1}+L_{n-2}=(\frac{1+\sqrt{5}}{2})^n+(\frac{1-\sqrt{5}}{2})^n$$

$$L(x)=\frac{2-x}{1-x-x^2}$$

Catlan numbers

$$c_1=1, c_n=\sum_{i=0}^{n-1}c_ic_{n-1-i}=c_{n-1}\frac{4n-2}{n+1}=\frac{\binom{2n}{n}}{n+1}$$

$$=\binom{2n}{n}-\binom{2n}{n-1}, c(x)=\frac{1-\sqrt{1-4x}}{2x}$$

Stirling cycle numbers

Divide n elements into k non-empty cycles.

$$s(n,0)=0, s(n,n)=1, s(n+1,k)=s(n,k-1)-ns(n,k)$$

$$s(n,k)=(-1)^{n-k}\begin{bmatrix}n\\k\end{bmatrix}$$

$$\begin{bmatrix}n+1\\k\end{bmatrix}=n\begin{bmatrix}n\\k\end{bmatrix}+\begin{bmatrix}n\\k-1\end{bmatrix}, \begin{bmatrix}n+1\\2\end{bmatrix}=n!H_n$$

$$x^{\underline{n}}=x(x-1)...(x-n+1)=\sum_{k=0}^{n}\begin{bmatrix}n\\k\end{bmatrix}(-1)^{n-k}x^k$$

$$x^{\overline{n}}=x(x+1)...(x+n-1)=\sum_{k=0}^{n}\begin{bmatrix}n\\k\end{bmatrix}x^k$$

Stirling subset numbers

Divide n elements into k non-empty subsets.

$$\begin{Bmatrix}n+1\\k\end{Bmatrix}=k\begin{Bmatrix}n\\k\end{Bmatrix}+\begin{Bmatrix}n\\k-1\end{Bmatrix}$$

$$x^n=\sum_{k=0}^{n}\begin{Bmatrix}n\\k\end{Bmatrix}x^{\underline{k}}=\sum_{k=0}^{n}\begin{Bmatrix}n\\k\end{Bmatrix}(-1)^{n-k}x^{\overline{k}}$$

$$m!\begin{Bmatrix}n\\m\end{Bmatrix}=\sum_{k=0}^{m}\binom{m}{k}k^n(-1)^{m-k}$$

$$\sum_{k=1}^{n}k^p=\sum_{k=0}^{p}\begin{Bmatrix}p\\k\end{Bmatrix}(n+1)^{\underline{k}}$$

For a fixed k, generating functions :

$$\sum_{n=0}^{\infty}\begin{Bmatrix}n\\k\end{Bmatrix}x^{n-k}=\prod_{r=1}^{k}\frac{1}{1-rx}$$

Motzkin numbers

Draw non-intersecting chords between n points on a circle.

Pick n numbers $k_1,k_2,...,k_n\in\{-1,0,1\}$ so that $\sum_i^a k_i(1\leq a\leq n)$ is non-negative and the sum of all numbers is 0.

$$M_{n+1}=M_n+\sum_{i}^{n-1}M_iM_{n-1-i}=\frac{(2n+3)M_n+3nM_{n-1}}{n+3}$$

$$M_n=\sum_{i=0}^{\lfloor\frac{n}{2}\rfloor}\binom{n}{2k}Catlan(k)$$

$$M(X)=\frac{1-x-\sqrt{1-2x-3x^2}}{2x^2}$$

Eulerian numbers

Permutations of the numbers 1 to n in which exactly k elements are greater than the previous element.

$$\left\langle{n\atop k}\right\rangle=(k+1)\left\langle{n-1\atop k}\right\rangle+(n-k)\left\langle{n-1\atop k-1}\right\rangle$$

$$x^n=\sum_k\left\langle{n\atop k}\right\rangle\binom{x+k}{n}$$

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

Harmonic numbers

Sum of the reciprocals of the first n natural numbers.

$$\sum_{k=1}^{n}H_k=(n+1)H_n-n$$

$$\sum_{k=1}^{n}kH_k=\frac{n(n+1)}{2}H_n-\frac{n(n-1)}{4}$$

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

Pentagonal number theorem

$$\prod_{n=1}^{\infty}(1-x^n)=\sum_{n=-\infty}^{\infty}(-1)^kx^{k(3k-1)/2}$$

$$p(n)=p(n-1)+p(n-2)-p(n-5)-p(n-7)+\cdots$$

$$f(n,k)=p(n)-p(n-k)-p(n-2k)+p(n-5k)+p(n-7k)-\cdots$$

Bell numbers

Divide a set that has exactly n elements.

$$B_n=\sum_{k=1}^{n}\begin{Bmatrix}n\\k\end{Bmatrix}, \quad B_{n+1}=\sum_{k=0}^{n}\binom{n}{k}B_k$$

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

$$B(x)=\sum_{n=0}^{\infty}\frac{B_n}{n!}x^n=e^{e^x-1}$$

Bernoulli numbers

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

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

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

Sum of powers

$$\sum_{i=1}^{n}i^2=\frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n}i^3=(\frac{n(n+1)}{2})^2$$

$$\sum_{i=1}^{n}i^4=\frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

$$\sum_{i=1}^{n}i^5=\frac{n^2(n+1)^2(2n^2+2n-1)}{12}$$

Sum of squares Denote $r_k(n)$ the ways to form n with k squares. If :

$$n = 2^{a_0} p_1^{2a_1} \cdots p_r^{2a_r} q_1 b_1 \cdots q_s b_s$$

where $p_i \equiv 3 \mod 4$, $q_i \equiv 1 \mod 4$, then

$$r_2(n) = \begin{cases} 0 & \text{if any } a_i \text{ is a half-integer} \\ 4\prod_1^r (b_i + 1) & \text{if all } a_i \text{ are integers} \end{cases}$$

$r_3(n) > 0$ when and only when n is not $4^a(8b+7)$.

Derangement

$$D_1 = 0, D_2 = 1, D_n = n!(\frac{1}{0!} - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + ... + \frac{(-1)^n}{n!})$$

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

Tetrahedron volume If U, V, W, u, v, w are lengths of edges of the tetrahedron (first three form a triangle; u opposite to U and so on)

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