

Dracarys

Team Reference Library

May 11, 2016

|                         |    |                      |    |
|-------------------------|----|----------------------|----|
| Contents                |    | 后缀自动机                | 17 |
| 多边形与圆面积交                | 2  | 后缀树 (With Pop Front) | 17 |
| 二维几何                    | 2  | 字符串最小表示              | 19 |
| $n \log n$ 半平面交         | 3  | 轻重链剖分                | 19 |
| Delaunay 三角剖分           | 4  | Splay Tree           | 20 |
| 三维几何操作合并                | 5  | Link Cut Tree        | 20 |
| 三维凸包                    | 6  | Dominator Tree       | 21 |
| 凸包上快速询问                 | 7  | DancingLinks         | 21 |
| 圆的面积模板 ( $n^2 \log n$ ) | 8  | 环状最长公共子序列            | 22 |
| 三角形的心                   | 9  | 直线下有多少个格点            | 23 |
| 最小覆盖球                   | 9  | 费用流                  | 23 |
| 经纬度求球面最短距离              | 9  | 积分表                  | 24 |
| 长方体表面两点最短距离             | 9  | Java                 | 24 |
| 最大团                     | 10 | Vimrc                | 25 |
| KM                      | 11 |                      |    |
| 最小树形图                   | 11 |                      |    |
| 无向图最小割                  | 11 |                      |    |
| 带花树                     | 12 |                      |    |
| Hopcroft                | 13 |                      |    |
| 素数判定                    | 13 |                      |    |
| 启发式分解                   | 13 |                      |    |
| 二次剩余                    | 14 |                      |    |
| Pell 方程                 | 14 |                      |    |
| 日期公式                    | 14 |                      |    |
| Schreier-Sims           | 14 |                      |    |
| 线性规划                    | 15 |                      |    |
| FFT                     | 16 |                      |    |
| Manacher/ 扩展 KMP        | 16 |                      |    |
| 后缀数组 ( 倍增 )             | 17 |                      |    |

## 多边形与圆面积交

```

1 double areaCT(Point pa, Point pb, double r) {
2     if (pa.len() < pb.len()) swap(pa, pb);
3     if (sign(pb.len()) == 0) return 0;
4     double a = pb.len(), b = pa.len(), c = (pb - pa).len();
5     double sinB = fabs(det(pb, pb - pa) / a / c),
6         cosB = dot(pb, pb - pa) / a / c,
7         sinC = fabs(det(pa, pb) / a / b),
8         cosC = dot(pa, pb) / a / b;
9     double B = atan2(sinB, cosB), C = atan2(sinC, cosC);
10    if (a > r) {
11        S = C / 2 * r * r;
12        h = a * b * sinC / c;
13        if (h < r && B < PI / 2) {
14            S -= (acos(h / r) * r * r - h * sqrt(r * r - h * h));
15        }
16    } else if (b > r) {
17        double theta = PI - B - asin(sinB / r * a);
18        S = a * r * sin(theta) / 2 + (C - theta) / 2 * r * r;
19    } else {
20        S = sinC * a * b / 2;
21    }
22    return S;
23 }

```

## 二维几何

```

1 struct Point {
2     Point rotate(const double ang) { // 逆时针旋转 ang 弧度
3         return Point(cos(ang) * x - sin(ang) * y, cos(ang) * y + sin(ang) * x);
4     }
5     Point turn90() { // 逆时针旋转 90 度
6         return Point(-y, x);
7     }
8 };
9 Point isLL(const Line &l1, const Line &l2) {
10    double s1 = det(l2.b - l2.a, l1.a - l2.a),
11        s2 = -det(l2.b - l2.a, l1.b - l2.a);
12    return (l1.a * s2 + l1.b * s1) / (s1 + s2);
13 }
14 bool onSeg(const Line &l, const Point &p) { // 点在线段上
15     return sign(det(p - l.a, l.b - l.a)) == 0 && sign(dot(p - l.a, p - l.b)) <= 0;
16 }
17 Point projection(const Line &l, const Point &p) { // 点到直线投影
18     return l.a + (l.b - l.a) * (dot(p - l.a, l.b - l.a) / (l.b - l.a).len2());
19 }

```

```

20 double disToLine(const Line &l, const Point &p) {
21     return abs(det(p - l.a, l.b - l.a) / (l.b - l.a).len());
22 }
23 double disToSeg(const Line &l, const Point &p) { // 点到线段距离
24     return sign(dot(p - l.a, l.b - l.a)) * sign(dot(p - l.b, l.a - l.b)) != 1 ?
25         disToLine(l, p) : min((p - l.a).len(), (p - l.b).len());
26 }
27 Point symmetryPoint(const Point a, const Point b) { // 点 b 关于点 a 的中心对称点
28     return a + a - b;
29 }
30 Point reflection(const Line &l, const Point &p) { // 点关于直线的对称点
31     return symmetryPoint(projection(l, p), p);
32 }
33 // 求圆与直线的交点
34 bool isCL(Circle a, Line l, Point &p1, Point &p2) {
35     double x = dot(l.a - a.o, l.b - l.a),
36         y = (l.b - l.a).len2(),
37         d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
38     if (sign(d) < 0) return false;
39     d = max(d, 0.0);
40     Point p = l.a - ((l.b - l.a) * (x / y)), delta = (l.b - l.a) * (sqrt(d) / y);
41     p1 = p + delta, p2 = p - delta;
42     return true;
43 }
44 // 求圆与圆的交面积
45 double areaCC(const Circle &c1, const Circle &c2) {
46     double d = (c1.o - c2.o).len();
47     if (sign(d - (c1.r + c2.r)) >= 0) {
48         return 0;
49     }
50     if (sign(d - abs(c1.r - c2.r)) <= 0) {
51         double r = min(c1.r, c2.r);
52         return r * r * PI;
53     }
54     double x = (d * d + c1.r * c1.r - c2.r * c2.r) / (2 * d),
55         t1 = acos(x / c1.r), t2 = acos((d - x) / c2.r);
56     return c1.r * c1.r * t1 + c2.r * c2.r * t2 - d * c1.r * sin(t1);
57 }
58 // 求圆与圆的交点, 注意调用前要先判定重圆
59 bool isCC(Circle a, Circle b, Point &p1, Point &p2) {
60     double s1 = (a.o - b.o).len();
61     if (sign(s1 - a.r - b.r) > 0 || sign(s1 - abs(a.r - b.r)) < 0) return false;
62     double s2 = (a.r * a.r - b.r * b.r) / s1;
63     double aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
64     Point o = (b.o - a.o) * (aa / (aa + bb)) + a.o;

```

```

65 Point delta = (b.o - a.o).unit().turn90() * newSqrt(a.r * a.r - aa * aa);
66 p1 = o + delta, p2 = o - delta;
67 return true;
68 }
69 // 求点到圆的切点, 按关于点的顺时针方向返回两个点
70 bool tanCP(const Circle &c, const Point &p0, Point &p1, Point &p2) {
71     double x = (p0 - c.o).len2(), d = x - c.r * c.r;
72     if (d < EPS) return false; // 点在圆上认为没有切点
73     Point p = (p0 - c.o) * (c.r * c.r / x);
74     Point delta = ((p0 - c.o) * (-c.r * sqrt(d) / x)).turn90();
75     p1 = c.o + p + delta;
76     p2 = c.o + p - delta;
77     return true;
78 }
79 // 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
80 vector<Line> extanCC(const Circle &c1, const Circle &c2) {
81     vector<Line> ret;
82     if (sign(c1.r - c2.r) == 0) {
83         Point dir = c2.o - c1.o;
84         dir = (dir * (c1.r / dir.len())).turn90();
85         ret.push_back(Line(c1.o + dir, c2.o + dir));
86         ret.push_back(Line(c1.o - dir, c2.o - dir));
87     } else {
88         Point p = (c1.o * -c2.r + c2.o * c1.r) / (c1.r - c2.r);
89         Point p1, p2, q1, q2;
90         if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) {
91             if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
92             ret.push_back(Line(p1, q1));
93             ret.push_back(Line(p2, q2));
94         }
95     }
96     return ret;
97 }
98 // 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
99 vector<Line> intanCC(const Circle &c1, const Circle &c2) {
100     vector<Line> ret;
101     Point p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
102     Point p1, p2, q1, q2;
103     if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) { // 两圆相切认为没有切线
104         ret.push_back(Line(p1, q1));
105         ret.push_back(Line(p2, q2));
106     }
107     return ret;
108 }
109 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

```

```

110 int ret = 0, n = polygon.size();
111 for(int i = 0; i < n; ++i) {
112     Point u = polygon[i], v = polygon[(i + 1) % n];
113     if (onSeg(Line(u, v), p)) return true;
114     if (sign(u.y - v.y) <= 0) swap(u, v);
115     if (sign(p.y - u.y) > 0 || sign(p.y - v.y) <= 0) continue;
116     ret += sign(det(p, v, u)) > 0;
117 }
118 return ret & 1;
119 }
120 vector<Point> convexCut(const vector<Point>&ps, Line l) { // 用半平面 (q1,q2)
    ↪ 的逆时针方向去切凸多边形
121     vector<Point> qs;
122     int n = ps.size();
123     for (int i = 0; i < n; ++i) {
124         Point p1 = ps[i], p2 = ps[(i + 1) % n];
125         int d1 = sign(det(l.a, l.b, p1)), d2 = sign(det(l.a, l.b, p2));
126         if (d1 >= 0) qs.push_back(p1);
127         if (d1 * d2 < 0) qs.push_back(isLL(Line(p1, p2), l));
128     }
129     return qs;
130 }
131 vector<Point> convexHull(vector<Point> ps) { // 求点集 ps 组成的凸包
132     int n = ps.size(); if (n <= 1) return ps;
133     sort(ps.begin(), ps.end());
134     vector<Point> qs;
135     for (int i = 0; i < n; qs.push_back(ps[i++]))
136         while (qs.size() > 1 && sign(det(qs[qs.size()-2], qs.back(), ps[i])) <= 0)
            ↪ qs.pop_back();
137     for (int i = n - 2, t = qs.size(); i >= 0; qs.push_back(ps[i--]))
138         while ((int)qs.size() > t && sign(det(qs[(int)qs.size()-2], qs.back(), ps[i])) <= 0)
            ↪ qs.pop_back();
139     qs.pop_back(); return qs;
140 }

```

### $n \log n$ 半平面交

```

1 struct Point {
2     int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0); }
3 };
4 struct Line {
5     bool include(const Point &p) const { return sign(det(b - a, p - a)) > 0; }
6     Line push() const { // 将半平面向外推 eps
7         const double eps = 1e-6;
8         Point delta = (b - a).turn90().norm() * eps;

```

```

9     return Line(a - delta, b - delta);
10 }
11 };
12 bool sameDir(const Line &l0, const Line &l1) { return parallel(l0, l1) && sign(dot(l0.b
    ↪ - l0.a, l1.b - l1.a)) == 1; }
13 bool operator < (const Point &a, const Point &b) {
14     if (a.quad() != b.quad()) {
15         return a.quad() < b.quad();
16     } else {
17         return sign(det(a, b)) > 0;
18     }
19 }
20 bool operator < (const Line &l0, const Line &l1) {
21     if (sameDir(l0, l1)) {
22         return l1.include(l0.a);
23     } else {
24         return (l0.b - l0.a) < (l1.b - l1.a);
25     }
26 }
27 bool check(const Line &u, const Line &v, const Line &w) { return w.include(intersect(u,
    ↪ v)); }
28 vector<Point> intersection(vector<Line> &l) {
29     sort(l.begin(), l.end());
30     deque<Line> q;
31     for (int i = 0; i < (int)l.size(); ++i) {
32         if (i && sameDir(l[i], l[i - 1])) {
33             continue;
34         }
35         while (q.size() > 1 && !check(q[q.size() - 2], q[q.size() - 1], l[i])) q.pop_back();
36         while (q.size() > 1 && !check(q[1], q[0], l[i])) q.pop_front();
37         q.push_back(l[i]);
38     }
39     while (q.size() > 2 && !check(q[q.size() - 2], q[q.size() - 1], q[0])) q.pop_back();
40     while (q.size() > 2 && !check(q[1], q[0], q[q.size() - 1])) q.pop_front();
41     vector<Point> ret;
42     for (int i = 0; i < (int)q.size(); ++i) ret.push_back(intersect(q[i], q[(i + 1) %
    ↪ q.size()]));
43     return ret;
44 }

```

## Delaunay 三角剖分

```

1  /*
2  Delaunay Triangulation 随机增量算法 :
3  节点数至少为点数的 6 倍, 空间消耗较大注意计算内存使用
4  建图的过程在 build 中, 注意初始化内存池和初始三角形的坐标范围 (Triangulation::LOTS)

```

```

5  Triangulation::find 返回包含某点的三角形
6  Triangulation::add_point 将某点加入三角剖分
7  某个 Triangle 在三角剖分中当且仅当它的 has_children 为 0
8  如果要找到三角形 u 的邻域, 则枚举它的所有 u.edge[i].tri, 该条边的两个点为 u.p[(i+1)%3],
    ↪ u.p[(i+2)%3]
9  */
10 const int N = 100000 + 5, MAX_TRIS = N * 6;
11 const double EPSILON = 1e-6, PI = acos(-1.0);
12 struct Point {
13     double x,y; Point():x(0),y(0){} Point(double x, double y):x(x),y(y){}
14     bool operator ==(Point const& that)const {return x==that.x&&y==that.y;}
15 };
16 inline double sqr(double x) { return x*x; }
17 double dist_sqr(Point const& a, Point const& b){return sqr(a.x-b.x)+sqr(a.y-b.y);}
18 bool in_circumcircle(Point const& p1, Point const& p2, Point const& p3, Point const& p4)
    ↪ {
19     double u11 = p1.x - p4.x, u21 = p2.x - p4.x, u31 = p3.x - p4.x;
20     double u12 = p1.y - p4.y, u22 = p2.y - p4.y, u32 = p3.y - p4.y;
21     double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
22     double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
23     double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
24     double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33 +
    ↪ u11*u22*u33;
25     return det > EPSILON;
26 }
27 double side(Point const& a, Point const& b, Point const& p) { return (b.x-a.x)*(p.y-a.y)
    ↪ - (b.y-a.y)*(p.x-a.x);}
28 typedef int SideRef; struct Triangle; typedef Triangle* TriangleRef;
29 struct Edge {
30     TriangleRef tri; SideRef side; Edge() : tri(0), side(0) {}
31     Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {}
32 };
33 struct Triangle {
34     Point p[3]; Edge edge[3]; TriangleRef children[3]; Triangle() {}
35     Triangle(Point const& p0, Point const& p1, Point const& p2) {
36         p[0]=p0;p[1]=p1;p[2]=p2;children[0]=children[1]=children[2]=0;
37     }
38     bool has_children() const { return children[0] != 0; }
39     int num_children() const {
40         return children[0] == 0 ? 0
41             : children[1] == 0 ? 1
42             : children[2] == 0 ? 2 : 3;
43     }
44     bool contains(Point const& q) const {
45         double a=side(p[0],p[1],q), b=side(p[1],p[2],q), c=side(p[2],p[0],q);

```

```

46     return a >= -EPSILON && b >= -EPSILON && c >= -EPSILON;
47 }
48 } triange_pool[MAX_TRIS], *tot_triangles;
49 void set_edge(Edge a, Edge b) {
50     if (a.tri) a.tri->edge[a.side] = b;
51     if (b.tri) b.tri->edge[b.side] = a;
52 }
53 class Triangulation {
54 public:
55     Triangulation() {
56         const double LOTS = 1e6;
57         the_root = new(tot_triangles++)
58             ↪ Triangle(Point(-LOTS, -LOTS), Point(+LOTS, -LOTS), Point(0, +LOTS));
59     }
60     TriangleRef find(Point p) const { return find(the_root, p); }
61     void add_point(Point const& p) { add_point(find(the_root, p), p); }
62 private:
63     TriangleRef the_root;
64     static TriangleRef find(TriangleRef root, Point const& p) {
65         for(;;) {
66             if (!root->has_children()) return root;
67             else for (int i = 0; i < 3 && root->children[i]; ++i)
68                 if (root->children[i]->contains(p))
69                     {root = root->children[i]; break;}
70         }
71     }
72     void add_point(TriangleRef root, Point const& p) {
73         TriangleRef tab, tbc, tca;
74         tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
75         tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
76         tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
77         set_edge(Edge(tab, 0), Edge(tbc, 1)); set_edge(Edge(tbc, 0), Edge(tca, 1));
78         set_edge(Edge(tca, 0), Edge(tab, 1)); set_edge(Edge(tab, 2), root->edge[2]);
79         set_edge(Edge(tbc, 2), root->edge[0]); set_edge(Edge(tca, 2), root->edge[1]);
80         root->children[0] = tab; root->children[1] = tbc; root->children[2] = tca;
81         flip(tab, 2); flip(tbc, 2); flip(tca, 2);
82     }
83     void flip(TriangleRef tri, SideRef pi) {
84         TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
85         if (!trj || !in_circumcircle(tri->p[0], tri->p[1], tri->p[2], trj->p[pj])) return;
86         TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj],
87             ↪ tri->p[pi]);
88         TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi],
89             ↪ trj->p[pj]);
90         set_edge(Edge(trk, 0), Edge(trl, 0));

```

```

88     set_edge(Edge(trk, 1), tri->edge[(pi+2)%3]); set_edge(Edge(trk, 2),
89         ↪ trj->edge[(pj+1)%3]);
90     set_edge(Edge(trl, 1), trj->edge[(pj+2)%3]); set_edge(Edge(trl, 2),
91         ↪ tri->edge[(pi+1)%3]);
92     tri->children[0] = trk; tri->children[1] = trl; tri->children[2] = 0;
93     trj->children[0] = trk; trj->children[1] = trl; trj->children[2] = 0;
94     flip(trk, 1); flip(trk, 2); flip(trl, 1); flip(trl, 2);
95 }
96 };
97 int n; Point ps[N];
98 void build(){
99     tot_triangles = triange_pool; cin >> n;
100     for(int i = 0; i < n; ++i) scanf("%lf%lf", &ps[i].x, &ps[i].y);
101     random_shuffle(ps, ps + n); Triangulation tri;
102     for(int i = 0; i < n; ++i) tri.add_point(ps[i]);
103 }

```

### 三维几何操作合并

```

1 struct Point3D {
2     double x, y, z;
3 };
4 Point3D det(const Point3D &a, const Point3D &b) {
5     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);
6 }
7 // 平面法向量 : 平面上两个向量叉积
8 // 点共平面 : 平面上一点与之的向量点积法向量为 0
9 // 点在线段 ( 直线 ) 上 : 共线且两边点积非正
10 // 点在三角形内 ( 不包含边界, 需再判断是与某条边共线 )
11 bool pointInTri(const Point3D &a, const Point3D &b, const Point3D &c, const Point3D &p)
12     ↪ {
13     return sign(det(a - b, a - c).len() - det(p - a, p - b).len() - det(p - b, p -
14         ↪ c).len() - det(p - c, p - a).len()) == 0;
15 }
16 // 共平面的两点是否在这平面上一条直线的同侧
17 bool sameSide(const Point3D &a, const Point3D &b, const Point3D &p0, const Point3D &p1)
18     ↪ {
19     return sign(dot(det(a - b, p0 - b), det(a - b, p1 - b))) > 0;
20 }
21 // 两点在平面同侧 : 点积法向量符号相同
22 // 两直线平行 / 垂直 : 同二维
23 // 平面平行 / 垂直 : 判断法向量
24 // 线面垂直 : 法向量和直线平行
25 // 判断空间线段是否相交 : 四点共面两线段不平行相互在异侧
26 // 线段和三角形是否相交 : 线段在三角形平面不同侧
27     ↪ 三角形任意两点在线段和第三点组成的平面的不同侧

```

```

24 // 求空间直线交点
25 Point3D intersection(const Point3D &a0, const Point3D &b0, const Point3D &a1, const
    ↪ Point3D &b1) {
26     double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) / ((a0.x -
    ↪ b0.x) * (a1.y - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));
27     return a0 + (b0 - a0) * t;
28 }
29 // 求平面和直线的交点
30 Point3D intersection(const Point3D &a, const Point3D &b, const Point3D &c, const Point3D
    ↪ &l0, const Point3D &l1) {
31     Point3D p = pVec(a, b, c); // 平面法向量
32     double t = (p.x * (a.x - l0.x) + p.y * (a.y - l0.y) + p.z * (a.z - l0.z)) / (p.x *
    ↪ (l1.x - l0.x) + p.y * (l1.y - l0.y) + p.z * (l1.z - l0.z));
33     return l0 + (l1 - l0) * t;
34 }
35 // 求平面交线：取不平行的一条直线的一个交点，以及法向量叉积得到直线方向
36 // 点到直线距离：叉积得到三角形的面积除以底边
37 // 点到平面距离：点积法向量
38 // 直线间距离：平时随便取一点求距离，否则叉积方向向量得到方向点积计算长度
39 // 直线夹角：点积 平面夹角：法向量点积
40 // 三维向量旋转操作(绕向量 s 旋转 ang 角度)，对于右手系 s 指向观察者时逆时针
41 // 矩阵版
42 void rotate(const Point3D &s, double ang) {
43     double l = s.len(), x = s.x / l, y = s.y / l, z = s.z / l, sinA = sin(ang), cosA =
    ↪ cos(ang);
44     double p[4][4] = {cosA + (1 - cosA) * x * x, (1 - cosA) * x * y - sinA * z, (1 - cosA)
    ↪ * x * z + sinA * y, 0,
45         (1 - cosA) * y * x + sinA * z, cosA + (1 - cosA) * y * y, (1 - cosA) * y * z - sinA
    ↪ * x, 0,
46         (1 - cosA) * z * x - sinA * y, (1 - cosA) * z * y + sinA * x, cosA + (1 - cosA) * z
    ↪ * z, 0,
47         0, 0, 0, 1 };
48 }
49 // 计算版：把需要旋转的向量按照 s 分解，做二维旋转，再回到三维

```

### 三维凸包

```

1 #define SIZE(X) (int(X.size()))
2 #define PI 3.14159265358979323846264338327950288
3 struct Point {
4     Point cross(const Point &p) const
5     { return Point(y * p.z - z * p.y, z * p.x - x * p.z, x * p.y - y * p.x); }
6 } info[1005];
7 int mark[1005][1005], n, cnt;;
8 double mix(const Point &a, const Point &b, const Point &c)
9 { return a.dot(b.cross(c)); }

```

```

10 double area(int a, int b, int c)
11 { return ((info[b] - info[a]).cross(info[c] - info[a])).length(); }
12 double volume(int a, int b, int c, int d)
13 { return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]); }
14 struct Face {
15     int a, b, c; Face() {}
16     Face(int a, int b, int c): a(a), b(b), c(c) {}
17     int &operator [](int k)
18     { if (k == 0) return a; if (k == 1) return b; return c; }
19 };
20 vector <Face> face;
21 inline void insert(int a, int b, int c) { face.push_back(Face(a, b, c)); }
22 void add(int v) {
23     vector <Face> tmp; int a, b, c; cnt++;
24     for (int i = 0; i < SIZE(face); i++) {
25         a = face[i][0]; b = face[i][1]; c = face[i][2];
26         if (Sign(volume(v, a, b, c)) < 0)
27             mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
28         else tmp.push_back(face[i]);
29     } face = tmp;
30     for (int i = 0; i < SIZE(tmp); i++) {
31         a = face[i][0]; b = face[i][1]; c = face[i][2];
32         if (mark[a][b] == cnt) insert(b, a, v);
33         if (mark[b][c] == cnt) insert(c, b, v);
34         if (mark[c][a] == cnt) insert(a, c, v);
35     }
36 int Find() {
37     for (int i = 2; i < n; i++) {
38         Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
39         if (ndir == Point()) continue; swap(info[i], info[2]);
40         for (int j = i + 1; j < n; j++) if (Sign(volume(0, 1, 2, j)) != 0) {
41             swap(info[j], info[3]); insert(0, 1, 2); insert(0, 2, 1); return 1;
42         } return 0; }
43 int main() {
44     for (; scanf("%d", &n) == 1; ) {
45         for (int i = 0; i < n; i++) info[i].Input();
46         sort(info, info + n); n = unique(info, info + n) - info;
47         face.clear(); random_shuffle(info, info + n);
48         if (Find()) { memset(mark, 0, sizeof(mark)); cnt = 0;
49             for (int i = 3; i < n; i++) add(i); vector<Point> Ndir;
50             for (int i = 0; i < SIZE(face); ++i) {
51                 Point p = (info[face[i][0]] - info[face[i][1]]).cross(
52                     info[face[i][2]] - info[face[i][1]]);
53                 p = p / p.length(); Ndir.push_back(p);
54             } sort(Ndir.begin(), Ndir.end());

```

```

55     int ans = unique(Ndir.begin(), Ndir.end()) - Ndir.begin();
56     printf("%d\n", ans);
57     } else printf("1\n");
58 } }
59 // 求重心
60 double calcDist(const Point &p, int a, int b, int c)
61 { return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c)); }
62 //compute the minimal distance of center of any faces
63 double findDist() { //compute center of mass
64     double totalWeight = 0; Point center(.0, .0, .0);
65     Point first = info[face[0][0]];
66     for (int i = 0; i < SIZE(face); ++i) {
67         Point p = (info[face[i][0]]+info[face[i][1]]+info[face[i][2]]+first)*.25;
68         double weight = mix(info[face[i][0]] - first, info[face[i][1]]
69             - first, info[face[i][2]] - first);
70         totalWeight += weight; center = center + p * weight;
71     } center = center / totalWeight;
72     double res = 1e100; //compute distance
73     for (int i = 0; i < SIZE(face); ++i)
74         res = min(res, calcDist(center, face[i][0], face[i][1], face[i][2]));
75     return res; }

```

### 凸包上快速询问

```

1  /*
2   给定凸包, log n 内完成各种询问, 具体操作有 :
3   1. 判定一个点是否在凸包内
4   2. 询问凸包外的点到凸包的两个切点
5   3. 询问一个向量关于凸包的切点
6   4. 询问一条直线和凸包的交点
7   INF 为坐标范围, 需要定义点类大于号
8   改成实数只需修改 sign 函数, 以及把 long long 改为 double 即可
9   构造函数时传入凸包要求无重点, 面积非空, 以及 pair(x,y) 的最小点放在第一个
10 */
11 const int INF = 1000000000;
12 struct Convex
13 {
14     int n;
15     vector<Point> a, upper, lower;
16     Convex(vector<Point> _a) : a(_a) {
17         n = a.size();
18         int ptr = 0;
19         for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
20         for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
21         for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
22         upper.push_back(a[0]);

```

```

23     }
24     int sign(long long x) { return x < 0 ? -1 : x > 0; }
25     pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
26         int l = 0, r = (int)convex.size() - 2;
27         for( ; l + 1 < r; ) {
28             int mid = (l + r) / 2;
29             if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
30             else l = mid;
31         }
32         return max(make_pair(vec.det(convex[r]), r), make_pair(vec.det(convex[0]), 0));
33     }
34     void update_tangent(const Point &p, int id, int &i0, int &i1) {
35         if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
36         if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
37     }
38     void binary_search(int l, int r, Point p, int &i0, int &i1) {
39         if (l == r) return;
40         update_tangent(p, l % n, i0, i1);
41         int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
42         for( ; l + 1 < r; ) {
43             int mid = (l + r) / 2;
44             int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
45             if (smid == sl) l = mid;
46             else r = mid;
47         }
48         update_tangent(p, r % n, i0, i1);
49     }
50     int binary_search(Point u, Point v, int l, int r) {
51         int sl = sign((v - u).det(a[l % n] - u));
52         for( ; l + 1 < r; ) {
53             int mid = (l + r) / 2;
54             int smid = sign((v - u).det(a[mid % n] - u));
55             if (smid == sl) l = mid;
56             else r = mid;
57         }
58         return l % n;
59     }
60     // 判定点是否在凸包内, 在边界返回 true
61     bool contain(Point p) {
62         if (p.x < lower[0].x || p.x > lower.back().x) return false;
63         int id = lower_bound(lower.begin(), lower.end(), Point(p.x, -INF)) - lower.begin();
64         if (lower[id].x == p.x) {
65             if (lower[id].y > p.y) return false;
66         } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;

```



```

67     id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF), greater<Point>()) -
        ⇨ upper.begin();
68     if (upper[id].x == p.x) {
69         if (upper[id].y < p.y) return false;
70     } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
71     return true;
72 }
73 // 求点 p 关于凸包的两个切点, 如果在凸包外则有序返回编号, 多解返回任意一个□ 否则返回
        ⇨ false
74 bool get_tangent(Point p, int &i0, int &i1) {
75     if (contain(p)) return false;
76     i0 = i1 = 0;
77     int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
78     binary_search(0, id, p, i0, i1);
79     binary_search(id, (int)lower.size(), p, i0, i1);
80     id = lower_bound(upper.begin(), upper.end(), p, greater<Point>()) - upper.begin();
81     binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
82     binary_search((int)lower.size() - 1 + id, (int)lower.size() - 1 + (int)upper.size(),
        ⇨ p, i0, i1);
83     return true;
84 }
85 // 求凸包上和向量 vec 叉积最大的点, 返回编号, 有多个返回任意一个
86 int get_tangent(Point vec) {
87     pair<long long, int> ret = get_tangent(upper, vec);
88     ret.second = (ret.second + (int)lower.size() - 1) % n;
89     ret = max(ret, get_tangent(lower, vec));
90     return ret.second;
91 }
92 // 求凸包和直线 u,v 的交点, 如果无严格相交返回 false 。如果有则是和 (i,next(i))
        ⇨ 的交点, 两个点无序, 交在点上不确定返回两条线段之一。
93 bool get_intersection(Point u, Point v, int &i0, int &i1) {
94     int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
95     if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
96         if (p0 > p1) swap(p0, p1);
97         i0 = binary_search(u, v, p0, p1);
98         i1 = binary_search(u, v, p1, p0 + n);
99         return true;
100     } else {
101         return false;
102     }
103 }
104 };

```

圆的面积模板 ( $n^2 \log n$ )

```

1 struct Event {
2     Point p;
3     double ang;
4     int delta;
5     Event (Point p = Point(0, 0), double ang = 0, double delta = 0) : p(p), ang(ang),
        ⇨ delta(delta) {}
6 };
7 bool operator < (const Event &a, const Event &b) {
8     return a.ang < b.ang;
9 }
10 void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
11     double d2 = (a.o - b.o).len2(),
12         dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
13         pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4));
14     Point d = b.o - a.o, p = d.rotate(PI / 2),
15         q0 = a.o + d * dRatio + p * pRatio,
16         q1 = a.o + d * dRatio - p * pRatio;
17     double ang0 = (q0 - a.o).ang(),
18         ang1 = (q1 - a.o).ang();
19     evt.push_back(Event(q1, ang1, 1));
20     evt.push_back(Event(q0, ang0, -1));
21     cnt += ang1 > ang0;
22 }
23 bool issame(const Circle &a, const Circle &b) { return sign((a.o - b.o).len()) == 0 &&
        ⇨ sign(a.r - b.r) == 0; }
24 bool overlap(const Circle &a, const Circle &b) { return sign(a.r - b.r - (a.o -
        ⇨ b.o).len()) >= 0; }
25 bool intersect(const Circle &a, const Circle &b) { return sign((a.o - b.o).len() - a.r -
        ⇨ b.r) < 0; }
26 int C;
27 Circle c[N];
28 double area[N];
29 void solve() {
30     memset(area, 0, sizeof(double) * (C + 1));
31     for (int i = 0; i < C; ++i) {
32         int cnt = 1;
33         vector<Event> evt;
34         for (int j = 0; j < i; ++j) if (issame(c[i], c[j])) ++cnt;
35         for (int j = 0; j < C; ++j) {
36             if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) {
37                 ++cnt;
38             }
39         }
40         for (int j = 0; j < C; ++j) {

```

```

41     if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i],
42         ↪ c[j])) {
43         addEvent(c[i], c[j], evt, cnt);
44     }
45     if (evt.size() == 0) {
46         area[cnt] += PI * c[i].r * c[i].r;
47     } else {
48         sort(evt.begin(), evt.end());
49         evt.push_back(evt.front());
50         for (int j = 0; j + 1 < (int)evt.size(); ++j) {
51             cnt += evt[j].delta;
52             area[cnt] += det(evt[j].p, evt[j + 1].p) / 2;
53             double ang = evt[j + 1].ang - evt[j].ang;
54             if (ang < 0) {
55                 ang += PI * 2;
56             }
57             area[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2;
58         }}}

```

### 三角形的心

```

1 Point inCenter(const Point &A, const Point &B, const Point &C) { // 内心
2     double a = (B - C).len(), b = (C - A).len(), c = (A - B).len(),
3         s = fabs(det(B - A, C - A)),
4         r = s / p;
5     return (A * a + B * b + C * c) / (a + b + c);
6 }
7 Point circumCenter(const Point &a, const Point &b, const Point &c) { // 外心
8     Point bb = b - a, cc = c - a;
9     double db = bb.len2(), dc = cc.len2(), d = 2 * det(bb, cc);
10    return a - Point(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
11 }
12 Point othroCenter(const Point &a, const Point &b, const Point &c) { // 垂心
13     Point ba = b - a, ca = c - a, bc = b - c;
14     double Y = ba.y * ca.y * bc.y,
15         A = ca.x * ba.y - ba.x * ca.y,
16         x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
17         y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
18     return Point(x0, y0);
19 }

```

### 最小覆盖球

```

1 int nouter; Tpoint outer[4], res; double radius;
2 void ball() {
3     Tpoint q[3]; double m[3][3], sol[3], L[3], det;

```

```

4     int i, j; res.x = res.y = res.z = radius = 0;
5     for (i=0; i<3; ++i) q[i]=outer[i+1]-outer[0], sol[i]=dot(q[i], q[i]);
6     for (i=0; i<3; ++i) for (j=0; j<3; ++j) m[i][j]=dot(q[i], q[j])*2;
7     det= m[0][0]*m[1][1]*m[2][2]
8     + m[0][1]*m[1][2]*m[2][0]
9     + m[0][2]*m[2][1]*m[1][0]
10    - m[0][2]*m[1][1]*m[2][0]
11    - m[0][1]*m[1][0]*m[2][2]
12    - m[0][0]*m[1][2]*m[2][1];
13    if ( fabs(det)<eps ) return;
14    for (j=0; j<3; ++j) {
15        for (i=0; i<3; ++i) m[i][j]=sol[i];
16        L[j]=( m[0][0]*m[1][1]*m[2][2]
17            + m[0][1]*m[1][2]*m[2][0]
18            + m[0][2]*m[2][1]*m[1][0]
19            - m[0][2]*m[1][1]*m[2][0]
20            - m[0][1]*m[1][0]*m[2][2]
21            - m[0][0]*m[1][2]*m[2][1]
22        ) / det;
23        for (i=0; i<3; ++i) m[i][j]=dot(q[i], q[j])*2;
24    } res=outer[0];
25    for (i=0; i<3; ++i) res = res + q[i] * L[i];
26    radius=dist2(res, outer[0]);
27 }

```

### 经纬度求球面最短距离

```

1 double sphereDis(double lon1, double lat1, double lon2, double lat2, double R) {
2     return R * acos(cos(lat1) * cos(lat2) * cos(lon1 - lon2) + sin(lat1) * sin(lat2));
3 }

```

### 长方体表面两点最短距离

```

1 int r;
2 void turn(int i, int j, int x, int y, int z, int x0, int y0, int L, int W, int H) {
3     if (z==0) { int R = x*x+y*y; if (R<r) r=R;
4     } else {
5         if(i>=0 && i< 2) turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);
6         if(j>=0 && j< 2) turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);
7         if(i<=0 && i>-2) turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
8         if(j<=0 && j>-2) turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
9     }
10 int main(){
11     int L, H, W, x1, y1, z1, x2, y2, z2;
12     cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
13     if (z1!=0 && z1!=H) if (y1==0 || y1==W)

```

```

14 swap(y1,z1), std::swap(y2,z2), std::swap(W,H);
15 else swap(x1,z1), std::swap(x2,z2), std::swap(L,H);
16 if (z1==H) z1=0, z2=H-z2;
17 r=0x3fffffff; turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
18 cout<<r<<endl; return 0;
19 }

```

### 最大团

```

1 // Super Fast Maximum Clique
2 // To Build Graph: Maxclique(Edges, Number of Nodes)
3 // To Get Answer: mcqdyn(AnswerNodes Index Array, AnswerLength)
4 typedef bool BB[N];
5 struct Maxclique {
6     const BB* e; int pk, level; const float Tlimit;
7     struct Vertex{ int i, d; Vertex(int i):i(i),d(0){} };
8     typedef vector<Vertex> Vertices; typedef vector<int> ColorClass;
9     Vertices V; vector<ColorClass> C; ColorClass QMAX, Q;
10    static bool desc_degree(const Vertex &vi, const Vertex &vj){
11        return vi.d > vj.d;
12    }
13    void init_colors(Vertices &v){
14        const int max_degree = v[0].d;
15        for(int i = 0; i < (int)v.size(); i++) v[i].d = min(i, max_degree) + 1;
16    }
17    void set_degrees(Vertices &v){
18        for(int i = 0, j; i < (int)v.size(); i++)
19            for(v[i].d = j = 0; j < int(v.size()); j++)
20                v[i].d += e[v[i].i][v[j].i];
21    }
22    struct StepCount{ int i1, i2; StepCount():i1(0),i2(0){} };
23    vector<StepCount> S;
24    bool cut1(const int pi, const ColorClass &A){
25        for(int i = 0; i < (int)A.size(); i++) if (e[pi][A[i]]) return true;
26        return false;
27    }
28    void cut2(const Vertices &A, Vertices &B){
29        for(int i = 0; i < (int)A.size() - 1; i++)
30            if(e[A.back().i][A[i].i])
31                B.push_back(A[i].i);
32    }
33    void color_sort(Vertices &R){
34        int j = 0, maxno = 1, min_k = max((int)QMAX.size() - (int)Q.size() + 1, 1);
35        C[1].clear(), C[2].clear();
36        for(int i = 0; i < (int)R.size(); i++) {
37            int pi = R[i].i, k = 1;

```

```

38 while(cut1(pi, C[k])) k++;
39 if(k > maxno) maxno = k, C[maxno + 1].clear();
40 C[k].push_back(pi);
41 if(k < min_k) R[j++] .i = pi;
42 }
43 if(j > 0) R[j - 1].d = 0;
44 for(int k = min_k; k <= maxno; k++)
45     for(int i = 0; i < (int)C[k].size(); i++)
46         R[j].i = C[k][i], R[j++].d = k;
47 }
48 void expand_dyn(Vertices &R){// diff -> diff with no dyn
49     S[level].i1 = S[level].i1 + S[level - 1].i1 - S[level].i2;//diff
50     S[level].i2 = S[level - 1].i1;//diff
51     while((int)R.size()) {
52         if((int)Q.size() + R.back().d > (int)QMAX.size()){
53             Q.push_back(R.back().i); Vertices Rp; cut2(R, Rp);
54             if((int)Rp.size()){
55                 if((float)S[level].i1 / ++pk < Tlimit) degree_sort(Rp);//diff
56                 color_sort(Rp);
57                 S[level].i1++, level++;//diff
58                 expand_dyn(Rp);
59                 level--;//diff
60             }
61             else if((int)Q.size() > (int)QMAX.size()) QMAX = Q;
62             Q.pop_back();
63         }
64         else return;
65         R.pop_back();
66     }
67 }
68 void mcqdyn(int* maxclique, int &sz){
69     set_degrees(V); sort(V.begin(),V.end(), desc_degree); init_colors(V);
70     for(int i = 0; i < (int)V.size() + 1; i++) S[i].i1 = S[i].i2 = 0;
71     expand_dyn(V); sz = (int)QMAX.size();
72     for(int i = 0; i < (int)QMAX.size(); i++) maxclique[i] = QMAX[i];
73 }
74 void degree_sort(Vertices &R){
75     set_degrees(R); sort(R.begin(), R.end(), desc_degree);
76 }
77 Maxclique(const BB* conn, const int sz, const float tt = 0.025) \
78 : pk(0), level(1), Tlimit(tt){
79     for(int i = 0; i < sz; i++) V.push_back(Vertex(i));
80     e = conn, C.resize(sz + 1), S.resize(sz + 1);
81 }
82 };

```

## KM

```

1 // 最小匹配, 自带初始化 n <= m 方案存在 p[] 中
2 const int N = 105;
3 const int INF = 1000000000; // 严格大于最大边权
4 int n, m, a[N][N];
5 int u[N], v[N], p[N], fa[N], minv[N];
6 bool used[N];
7 int km() {
8     memset(u, 0, sizeof(int) * n);
9     for (int i = 0; i <= m; ++i) v[i] = 0, p[i] = n;
10    for (int i = 0; i < n; ++i) {
11        p[m] = i;
12        int j0 = m;
13        for (int j = 0; j <= m; ++j) minv[j] = INF, used[j] = false;
14        do {
15            used[j0] = true;
16            int i0 = p[j0], delta = INF, j1;
17            for (int j = 0; j < m; ++j) {
18                if (!used[j]) {
19                    int cur = a[i0][j] - u[i0] - v[j];
20                    if (cur < minv[j]) minv[j] = cur, fa[j] = j0;
21                    if (minv[j] < delta) delta = minv[j], j1 = j;
22                }
23            }
24            for (int j = 0; j <= m; ++j) {
25                if (used[j]) {
26                    u[p[j]] += delta, v[j] -= delta;
27                } else {
28                    minv[j] -= delta;
29                }
30            }
31            j0 = j1;
32        } while (p[j0] != n);
33        do {
34            int j1 = fa[j0];
35            p[j0] = p[j1];
36            j0 = j1;
37        } while (j0 != m);
38    }
39    return -v[m];
40 }

```

## 最小树形图

```

1 const int MAXN, INF; // INF >= sum( W_ij )
2 int from[MAXN + 10][MAXN * 2 + 10], n, m, edge[MAXN + 10][MAXN * 2 + 10];

```

```

3 int sel[MAXN * 2 + 10], fa[MAXN * 2 + 10], vis[MAXN * 2 + 10];
4 int getfa(int x){if(x == fa[x]) return x; return fa[x] = getfa(fa[x]);}
5 void liuzhu(){ // 1-base: root is 1, answer = (sel[i], i) for i in [2..n]
6     fa[1] = 1;
7     for(int i = 2; i <= n; ++i){
8         sel[i] = 1; fa[i] = i;
9         for(int j = 1; j <= n; ++j) if(fa[j] != i)
10            if(from[j][i] = i, edge[sel[i]][i] > edge[j][i]) sel[i] = j;
11    }
12    int limit = n;
13    while(1){
14        int prelimit = limit; memset(vis, 0, sizeof(vis)); vis[1] = 1;
15        for(int i = 2; i <= prelimit; ++i) if(fa[i] == i && !vis[i]){
16            int j = i; while(!vis[j]) vis[j] = i, j = getfa(sel[j]);
17            if(j == 1 || vis[j] != i) continue; vector<int> C; int k = j;
18            do C.push_back(k), k = getfa(sel[k]); while(k != j);
19            ++limit;
20            for(int i = 1; i <= n; ++i){
21                edge[i][limit] = INF, from[i][limit] = limit;
22            }
23            fa[limit] = vis[limit] = limit;
24            for(int i = 0; i < int(C.size()); ++i){
25                int x = C[i], fa[x] = limit;
26                for(int j = 1; j <= n; ++j)
27                    if(edge[j][x] != INF && edge[j][limit] > edge[j][x] - edge[sel[x]][x]){
28                        edge[j][limit] = edge[j][x] - edge[sel[x]][x];
29                        from[j][limit] = x;
30                    }
31            }
32            for(int j=1;j<=n;++j) if(getfa(j)==limit) edge[j][limit] = INF;
33            sel[limit] = 1;
34            for(int j = 1; j <= n; ++j)
35                if(edge[sel[limit]][limit] > edge[j][limit]) sel[limit] = j;
36        }
37        if(prelimit == limit) break;
38    }
39    for(int i = limit; i > 1; --i) sel[from[sel[i]][i]] = sel[i];
40 }

```

## 无向图最小割

```

1 int cost[maxn][maxn], seq[maxn], len[maxn], n, m, pop, ans;
2 bool used[maxn];
3 void Init(){
4     int i, j, a, b, c;

```

```

5  for(i=0;i<n;i++) for(j=0;j<n;j++) cost[i][j]=0;
6  for(i=0;i<m;i++){
7      scanf("%d %d %d",&a,&b,&c); cost[a][b]+=c; cost[b][a]+=c;
8  }
9  pop=n; for(i=0;i<n;i++) seq[i]=i;
10 }
11 void Work(){
12     ans=inf; int i,j,k,l,mm,sum,pk;
13     while(pop > 1){
14         for(i=1;i<pop;i++) used[seq[i]]=0; used[seq[0]]=1;
15         for(i=1;i<pop;i++) len[seq[i]]=cost[seq[0]][seq[i]];
16         pk=0; mm=-inf; k=-1;
17         for(i=1;i<pop;i++) if(len[seq[i]] > mm){ mm=len[seq[i]]; k=i; }
18         for(i=1;i<pop;i++){
19             used[seq[l=k]]=1;
20             if(i==pop-2) pk=k;
21             if(i==pop-1) break;
22             mm=-inf;
23             for(j=1;j<pop;j++) if(!used[seq[j]])
24                 if((len[seq[j]]+cost[seq[l]][seq[j]]) > mm)
25                     mm=len[seq[j]], k=j;
26         }
27         sum=0;
28         for(i=0;i<pop;i++) if(i != k) sum+=cost[seq[k]][seq[i]];
29         ans=min(ans,sum);
30         for(i=0;i<pop;i++)
31             cost[seq[k]][seq[i]]=cost[seq[i]][seq[k]]+cost[seq[pk]][seq[i]];
32         seq[pk]=seq[--pop];
33     }
34     printf("%d\n",ans);
35 }

```

## 带花树

```

1  vector<int> link[maxn];
2  int n,match[maxn],Queue[maxn],head,tail;
3  int pred[maxn],base[maxn],start,finish,newbase;
4  bool InQueue[maxn],InBlossom[maxn];
5  void push(int u){ Queue[tail++]=u;InQueue[u]=true; }
6  int pop(){ return Queue[head++]; }
7  int FindCommonAncestor(int u,int v){
8      bool InPath[maxn];
9      for(int i=0;i<n;i++) InPath[i]=0;
10     while(true){ u=base[u];InPath[u]=true;if(u==start) break;u=pred[match[u]]; }
11     while(true){ v=base[v];if(InPath[v]) break;v=pred[match[v]]; }
12     return v;

```

```

13 }
14 void ResetTrace(int u){
15     int v;
16     while(base[u]!=newbase){
17         v=match[u];
18         InBlossom[base[u]]=InBlossom[base[v]]=true;
19         u=pred[v];
20         if(base[u]!=newbase) pred[u]=v;
21     }
22 }
23 void BlossomContract(int u,int v){
24     newbase=FindCommonAncestor(u,v);
25     for (int i=0;i<n;i++)
26         InBlossom[i]=0;
27     ResetTrace(u);ResetTrace(v);
28     if(base[u]!=newbase) pred[u]=v;
29     if(base[v]!=newbase) pred[v]=u;
30     for(int i=0;i<n;++i)
31         if(InBlossom[base[i]]){
32             base[i]=newbase;
33             if(!InQueue[i]) push(i);
34         }
35 }
36 bool FindAugmentingPath(int u){
37     bool found=false;
38     for(int i=0;i<n;++i) pred[i]=-1,base[i]=i;
39     for (int i=0;i<n;i++) InQueue[i]=0;
40     start=u;finish=-1; head=tail=0; push(start);
41     while(head<tail){
42         int u=pop();
43         for(int i=link[u].size()-1;i>=0;i--){
44             int v=link[u][i];
45             if(base[u]!=base[v]&&match[u]!=v)
46                 if(v==start|| (match[v]>=0&&pred[match[v]]>=0))
47                     BlossomContract(u,v);
48             else if(pred[v]==-1){
49                 pred[v]=u;
50                 if(match[v]>=0) push(match[v]);
51                 else{ finish=v; return true; }
52             }
53         }
54     }
55     return found;
56 }
57 void AugmentPath(){

```

```

58 int u=finish,v,w;
59 while(u>=0){ v=pred[u];w=match[v];match[v]=u;match[u]=v;u=w; }
60 }
61 void FindMaxMatching(){
62     for(int i=0;i<n;++i) match[i]=-1;
63     for(int i=0;i<n;++i) if(match[i]==-1) if(FindAugmentingPath(i)) AugmentPath();
64 }

```

## Hopcroft

```

1 // 左侧 N 个点, 右侧 K 个点, 1-based, 初始化将 matx[],maty[] 都置为 0
2 int N, K;
3 int que[N], dx[N], dy[N], matx[N], maty[N];
4 int BFS()
5 {
6     int flag = 0, qt = 0, qh = 0;
7     for(int i = 1; i <= K; ++ i) dy[i] = 0;
8     for(int i = 1; i <= N; ++ i) {
9         dx[i] = 0;
10        if (! matx[i]) que[qt ++] = i;
11    }
12    while (qh < qt) {
13        int u = que[qh ++];
14        for(Edge *e = E[u]; e; e = e->n)
15            if (! dy[e->t]) {
16                dy[e->t] = dx[u] + 1;
17                if (! maty[e->t]) flag = true;
18                else {
19                    dx[maty[e->t]] = dx[u] + 2;
20                    que[qt ++] = maty[e->t];
21                }
22            }
23    }
24    return flag;
25 }
26 int DFS(int u)
27 {
28     for(Edge *e = E[u]; e; e = e->n)
29         if (dy[e->t] == dx[u] + 1) {
30             dy[e->t] = 0;
31             if (! maty[e->t] || DFS(maty[e->t])) {
32                 matx[u] = e->t; maty[e->t] = u;
33                 return true;
34             }
35         }
36     return false;

```

```

37 }
38 void Hopcroft()
39 {
40     while (BFS()) for(int i = 1; i <= N; ++ i) if (! matx[i]) DFS(i);
41 }

```

## 素数判定

```

1 int strong_pseudo_primetest(long long n,int base) {
2     long long n2=n-1,res;
3     int s=0;
4     while(n%2==0) n2>>=1,s++;
5     res=powmod(base,n2,n);
6     if((res==1)|| (res==n-1)) return 1;
7     s--;
8     while(s>0) {
9         res=mulmod(res,res,n);
10        if(res==n-1) return 1;
11        s--;
12    }
13    return 0; // n is not a strong pseudo prime
14 }
15 int isprime(long long n) {
16     static LL testNum[]={2,3,5,7,11,13,17,19,23,29,31,37};
17     static LL lim[]={4,0,1373653LL,25326001LL,25000000000LL,2152302898747LL, \
18         3474749660383LL,341550071728321LL,0,0,0,0};
19     if(n<2||n==3215031751LL) return 0;
20     for(int i=0;i<12;++i){
21         if(n<lim[i]) return 1;
22         if(strong_pseudo_primetest(n,testNum[i])==0) return 0;
23     }
24     return 1;
25 }

```

## 启发式分解

```

1 int ansn; LL ans[1000];
2 LL func(LL x,LL n){ return(mod_mul(x,x,n)+1)%n; }
3 LL Pollard(LL n){
4     LL i,x,y,p;
5     if(Rabin_Miller(n)) return n;
6     if(!(n&1)) return 2;
7     for(i=1;i<20;i++){
8         x=i; y=func(x,n); p=gcd(y-x,n);
9         while(p==1) {x=func(x,n); y=func(func(y,n),n); p=gcd((y-x+n)%n,n)%n;}
10        if(p==0||p==n) continue;

```

```

11     return p;
12 }
13 }
14 void factor(LL n){
15     LL x;
16     x=Pollard(n);
17     if(x==n){ ans[ansn++]=x; return; }
18     factor(x), factor(n/x);
19 }

```

## 二次剩余

```

1 void calcH(int &t, int &h, const int p) {
2     int tmp = p - 1; for (t = 0; (tmp & 1) == 0; tmp /= 2) t++; h = tmp;
3 }
4 // solve equation  $x^2 \bmod p = a$ 
5 bool solve(int a, int p, int &x, int &y) {
6     srand(19920225);
7     if (p == 2) { x = y = 1; return true; }
8     int p2 = p / 2, tmp = power(a, p2, p);
9     if (tmp == p - 1) return false;
10    if ((p + 1) % 4 == 0) {
11        x = power(a, (p + 1) / 4, p); y = p - x; return true;
12    } else {
13        int t, h, b, pb; calcH(t, h, p);
14        if (t >= 2) {
15            do {b = rand() % (p - 2) + 2;
16            } while (power(b, p / 2, p) != p - 1);
17            pb = power(b, h, p);
18        } int s = power(a, h / 2, p);
19        for (int step = 2; step <= t; step++) {
20            int ss = (((long long)(s * s) % p) * a) % p;
21            for (int i = 0; i < t - step; i++) ss = ((long long)ss * ss) % p;
22            if (ss + 1 == p) s = (s * pb) % p; pb = ((long long)pb * pb) % p;
23        } x = ((long long)s * a) % p; y = p - x;
24    } return true;
25 }

```

## Pell 方程

```

1 ULL A,B,p[maxn],q[maxn],a[maxn],g[maxn],h[maxn];
2 int main() {
3     for (int test=1, n;scanf("%d",&n) && n;++test) {
4         printf("Case %d: ",test);
5         if (fabs(sqrt(n)-floor(sqrt(n)+1e-7))<=1e-7) {
6             int a=(int)(floor(sqrt(n)+1e-7)); printf("%d %d\n",a,1);
7         } else {

```

```

8         // 求  $x^2 - ny^2 = 1$  的最小正整数根, n 不是完全平方数
9         p[1]=q[0]=h[1]=1;p[0]=q[1]=g[1]=0;
10        a[2]=(int)(floor(sqrt(n)+1e-7));
11        for (int i=2;i;++i) {
12            g[i]=-g[i-1]+a[i]*h[i-1]; h[i]=(n-sqr(g[i]))/h[i-1];
13            a[i+1]=(g[i]+a[2])/h[i]; p[i]=a[i]*p[i-1]+p[i-2];
14            q[i]=a[i]*q[i-1]+q[i-2];
15            if (sqr((ULL)(p[i]))-n*sqr((ULL)(q[i]))==1){
16                A=p[i];B=q[i];break; }
17        } cout << A << ' ' << B <<endl;
18    }
}

```

## 日期公式

```

1 int zeller(int y,int m,int d) {
2     if (m<=2) y--,m+=12; int c=y/100; y%=100;
3     int w=((c>>2)-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
4     if (w<0) w+=7; return(w);
5 }
6 int getId(int y, int m, int d) {
7     if (m < 3) {y --; m += 12};
8     return 365 * y + y / 4 - y / 100 + y / 400 + (153 * m + 2) / 5 + d;
9 }

```

## Schreier-Sims

```

1 struct Permutation{
2     vector<int> P;Permutation(){} Permutation(int n){ P.resize(n); }
3     Permutation inv()const{
4         Permutation ret(P.size());
5         for(int i = 0; i < int(P.size()); ++i) ret.P[P[i]] = i;
6         return ret;
7     }
8     int &operator [] (const int &dn){ return P[dn]; }
9     void resize(const size_t &sz){ P.resize(sz); }
10    size_t size()const{ return P.size(); }
11    const int &operator [] (const int &dn)const{ return P[dn]; }
12 };
13 Permutation operator *(const Permutation &a, const Permutation &b){
14     Permutation ret(a.size());
15     for(int i = 0; i < (int)a.size(); ++i) ret[i] = b[a[i]];
16     return ret;
17 }
18 typedef vector<Permutation> Bucket;
19 typedef vector<int> Table; typedef pair<int,int> pii;
20 int n, m;

```

```

21 vector<Bucket> buckets, bucketsInv; vector<Table> lookupTable;
22 int fastFilter(const Permutation &g, bool addToGroup = true){
23     int n = buckets.size();
24     Permutation p;
25     for(int i = 0; i < n; ++i){
26         int res = lookupTable[i][p[i]];
27         if(res == -1){
28             if(addToGroup){
29                 buckets[i].push_back(p); bucketsInv[i].push_back(p.inv());
30                 lookupTable[i][p[i]] = (int)buckets[i].size() - 1;
31             }
32             return i;
33         }
34         p = p * bucketsInv[i][res]; swap(i1,i2);
35     }
36     return -1;
37 }
38 long long calcTotalSize(){
39     long long ret = 1;
40     for(int i = 0; i < n; ++i) ret *= buckets[i].size();
41     return ret;
42 }
43 bool inGroup(const Permutation &g){ return fastFilter(g, false) == -1; }
44 void solve(const Bucket &gen,int _n){// m perm[0..n - 1]s
45     n = _n, m = gen.size();
46     //clear all
47     vector<Bucket> _buckets(n); swap(buckets, _buckets);
48     vector<Bucket> _bucketsInv(n); swap(bucketsInv, _bucketsInv);
49     vector<Table> _lookupTable(n); swap(lookupTable, _lookupTable);
50 }
51 for(int i = 0; i < n; ++i){
52     lookupTable[i].resize(n);
53     fill(lookupTable[i].begin(), lookupTable[i].end(), -1);
54 }
55 Permutation id(n);
56 for(int i = 0; i < n; ++i) id[i] = i;
57 for(int i = 0; i < n; ++i){
58     buckets[i].push_back(id); bucketsInv[i].push_back(id);
59     lookupTable[i][i] = 0;
60 }
61 for(int i = 0; i < m; ++i) fastFilter(gen[i]);
62 queue<pair<point,point>> toUpdate;
63 for(int i = 0; i < n; ++i)
64     for(int j = i; j < n; ++j)
65         for(int k = 0; k < (int)buckets[i].size(); ++k)

```

```

66         for(int l = 0; l < (int)buckets[j].size(); ++l)
67             toUpdate.push(make_pair(pii(i,k), pii(j,l)));
68 while(!toUpdate.empty()){
69     pii a = toUpdate.front().first, b = toUpdate.front().second;
70     toUpdate.pop();
71     int res=fastFilter(buckets[a.first][a.second]*buckets[b.first][b.second]);
72     if(res==-1) continue;
73     pii newPair(res, (int)buckets[res].size() - 1);
74     for(int i = 0; i < n; ++i)
75         for(int j = 0; j < (int)buckets[i].size(); ++j){
76             if(i <= res) toUpdate.push(make_pair(pii(i, j), newPair));
77             if(res <= i) toUpdate.push(make_pair(newPair, pii(i, j)));
78         }
79 }
80 }

```

### 线性规划

```

1 // 求 $\max\{cx \mid Ax \leq b, x \geq 0\}$ 的解
2 typedef vector<double> VD;
3 VD simplex(vector<VD> A, VD b, VD c) {
4     int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
5     vector<VD> D(n + 2, VD(m + 1, 0)); vector<int> ix(n + m);
6     for (int i = 0; i < n + m; ++ i) ix[i] = i;
7     for (int i = 0; i < n; ++ i) {
8         for (int j = 0; j < m - 1; ++ j) D[i][j] = -A[i][j];
9         D[i][m - 1] = 1; D[i][m] = b[i];
10        if (D[r][m] > D[i][m]) r = i;
11    }
12    for (int j = 0; j < m - 1; ++ j) D[n][j] = c[j];
13    D[n + 1][m - 1] = -1;
14    for (double d; ; ) {
15        if (r < n) {
16            int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t;
17            D[r][s] = 1.0 / D[r][s]; vector<int> speedUp;
18            for (int j = 0; j <= m; ++ j) if (j != s) {
19                D[r][j] *= -D[r][s];
20                if(D[r][j]) speedUp.push_back(j);
21            }
22            for (int i = 0; i <= n + 1; ++ i) if (i != r) {
23                for(int j = 0; j < speedUp.size(); ++ j)
24                    D[i][speedUp[j]] += D[r][speedUp[j]] * D[i][s];
25                D[i][s] *= D[r][s];
26            } r = -1; s = -1;
27        } for (int j = 0; j < m; ++ j) if (s < 0 || ix[s] > ix[j])
28            if (D[n + 1][j] > EPS || (D[n + 1][j] > -EPS && D[n][j] > EPS)) s = j;

```



```

29     if (s < 0) break;
30     for (int i = 0; i < n; ++i) if (D[i][s] < -EPS)
31         if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -EPS
32             || (d < EPS && ix[r + m] > ix[i + m])) r = i;
33     if (r < 0) return VD(); // 无边界
34 }
35 if (D[n + 1][m] < -EPS) return VD(); // 无解
36 VD x(m - 1);
37 for (int i = m; i < n + m; ++i) if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
38 return x; // 最优值在 D[n][m]
39 }

```

## FFT

```

1 // double 精度对 $10^9 + 7$  取模最多可以做到 $2^{20}$ 
2 const int MOD = 1000003;
3 const double PI = acos(-1);
4 typedef complex<double> Complex;
5 const int N = 65536, L = 15, MASK = (1 << L) - 1;
6 Complex w[N];
7 void FFTInit() {
8     for (int i = 0; i < N; ++i) {
9         w[i] = Complex(cos(2 * i * PI / N), sin(2 * i * PI / N));
10    }
11 }
12 void FFT(Complex p[], int n) {
13     for (int i = 1, j = 0; i < n - 1; ++i) {
14         for (int s = n; j ^= s >= 1, ~j & s;);
15         if (i < j) {
16             swap(p[i], p[j]);
17         }
18     }
19     for (int d = 0; (1 << d) < n; ++d) {
20         int m = 1 << d, m2 = m * 2, rm = n >> (d + 1);
21         for (int i = 0; i < n; i += m2) {
22             for (int j = 0; j < m; ++j) {
23                 Complex &p1 = p[i + j + m], &p2 = p[i + j];
24                 Complex t = w[rm * j] * p1;
25                 p1 = p2 - t;
26                 p2 = p2 + t;
27             }
28         }
29     }
30     Complex A[N], B[N], C[N], D[N];
31     void mul(int a[N], int b[N]) {
32         for (int i = 0; i < N; ++i) {
33             A[i] = Complex(a[i] >> L, a[i] & MASK);
34             B[i] = Complex(b[i] >> L, b[i] & MASK);

```

```

33     }
34     FFT(A, N), FFT(B, N);
35     for (int i = 0; i < N; ++i) {
36         int j = (N - i) % N;
37         Complex da = (A[i] - conj(A[j])) * Complex(0, -0.5),
38             db = (A[i] + conj(A[j])) * Complex(0.5, 0),
39             dc = (B[i] - conj(B[j])) * Complex(0, -0.5),
40             dd = (B[i] + conj(B[j])) * Complex(0.5, 0);
41         C[j] = da * dd + da * dc * Complex(0, 1);
42         D[j] = db * dd + db * dc * Complex(0, 1);
43     }
44     FFT(C, N), FFT(D, N);
45     for (int i = 0; i < N; ++i) {
46         long long da = (long long)(C[i].imag() / N + 0.5) % MOD,
47             db = (long long)(C[i].real() / N + 0.5) % MOD,
48             dc = (long long)(D[i].imag() / N + 0.5) % MOD,
49             dd = (long long)(D[i].real() / N + 0.5) % MOD;
50         a[i] = ((dd << (L * 2)) + ((db + dc) << L) + da) % MOD;
51     }
52 }

```

## Manacher/ 扩展 KMP

```

1 void Manacher(char text[], int n, int palindrome[]) {
2     palindrome[0] = 1;
3     for (int i = 1, j = 0, i < (n << 1) - 1; ++i) {
4         int p = i >> 1;
5         int q = i - p;
6         int r = (j + 1 >> 1) + palindrome[j] - 1;
7         palindrome[i] = r < q ? 0 : min(r - q + 1, palindrome[(j << 1) - i]);
8         while (0 <= p - palindrome[i] && q + palindrome[i] < n && text[p - palindrome[i]] ==
9             text[q + palindrome[i]]) {
10             palindrome[i] ++;
11         }
12         if (q + palindrome[i] - 1 > r) {
13             j = i;
14         }
15     }
16 }
17 void ExtendedKMP(char *s, int next[]) {
18     int l = strlen(s), i = 0, j = 0, k = 1;
19     while (1 + j < l && s[j] == s[1 + j]) {
20         ++j;
21     }
22     next[1] = j;

```

```

23     for (int i = 2; i < 1; ++i) {
24         int len = k + next[k], ll = next[i - k];
25         if (ll < len - i) {
26             next[i] = ll;
27         } else {
28             j = max(0, len - i);
29             while (i + j < 1 && s[j] == s[i + j]) {
30                 ++j;
31             }
32             next[i] = j;
33             k = i;
34         }
35     }
36 }

```

### 后缀数组 (倍增)

```

1  int rank[MAX_N], height[MAX_N];
2  int cmp(int *x, int a, int b, int d){
3      return x[a]==x[b]&&x[a+d]==x[b+d];
4  }
5  void doubling(int *a, int N, int M){
6      static int sRank[MAX_N], tmpA[MAX_N], tmpB[MAX_N];
7      int *x=tmpA, *y=tmpB;
8      for(int i=0; i<M; ++i) sRank[i]=0;
9      for(int i=0; i<N; ++i) ++sRank[x[i]=a[i]];
10     for(int i=1; i<M; ++i) sRank[i] += sRank[i-1];
11     for(int i=N-1; i>=0; --i) sa[--sRank[x[i]]]=i;
12     for(int d=1, p=0; p<N; M=p, d<=1){
13         p=0; for(int i=N-d; i<N; ++i) y[p++] = i;
14         for(int i=0; i<N; ++i) if(sa[i]>=d) y[p++] = sa[i]-d;
15         for(int i=0; i<M; ++i) sRank[i]=0;
16         for(int i=0; i<N; ++i) ++sRank[x[i]];
17         for(int i=1; i<M; ++i) sRank[i] += sRank[i-1];
18         for(int i=N-1; i>=0; --i) sa[--sRank[x[y[i]]]]=y[i];
19         swap(x, y); x[sa[0]]=0; p=1;
20         for(int i=1; i<N; ++i) x[sa[i]] = cmp(y, sa[i], sa[i-1], d)?p-1:p++;
21     }
22 }
23 void calcHeight(){
24     for(int i=0; i<N; ++i) rank[sa[i]] = i;
25     int cur=0; for(int i=0; i<N; ++i)
26     if(rank[i]){
27         if(cur) cur--;
28         for(; a[i+cur]==a[sa[rank[i]-1]+cur]; ++cur);
29         height[rank[i]] = cur;

```

```

30 }
31 }

```

### 后缀自动机

```

1  struct State {
2      int length;
3      State *parent, *go[C];
4      State(int length = 0):length(length), parent(NULL){
5          memset(go, 0, sizeof(go));
6      }
7      State* extend(State*, int token);
8  } node_pool[V], *tot_node;
9  State* State::extend(State *start, int token){
10     State *p=this;
11     State *np=new(tot_node++) State(this->length+1);
12     while(p!=NULL&&p->go[token]==NULL)
13         p->go[token]=np, p=p->parent;
14     if(p==NULL) np->parent=start;
15     else{
16         State *q=p->go[token];
17         if(p->length+1==q->length) np->parent=q;
18         else{
19             State *nq=new(tot_node++) State(p->length+1);
20             memcpy(nq->go, q->go, sizeof(q->go));
21             nq->parent=q->parent;
22             np->parent=q->parent=nq;
23             while(p!=NULL&&p->go[token]==q)
24                 p->go[token]=nq, p=p->parent;
25         }
26     }
27     return np;
28 }

```

### 后缀树 (With Pop Front)

```

1  int pos, text[N];
2  struct Node {
3      int l, r;
4      Node *suf, *ch[C];
5      int dgr;
6      Node *fa;
7      Node (int l = -1, int r = INF) : l(l), r(r) {
8          suf = fa = NULL;
9          memset(ch, 0, sizeof(ch));
10         dgr = 0;

```

```

11 }
12 Node* addEdge(Node *t) {
13     int c = text[t->l];
14     dgr += !ch[c];
15     ch[c] = t;
16     t->fa = this;
17     return t;
18 }
19 int len() {
20     return min(r, pos + 1) - 1;
21 }
22 };
23
24 int top;
25 Node pool[N << 1], *root, *nxtSuf, *cur;
26 int remCnt, curP, curLen;
27 long long size;
28 queue<Node*> leaves;
29 void init() {
30     top = 0, pos = -1;
31     remCnt = 0, curP = 0, curLen = 0;
32     nxtSuf = NULL;
33     root = cur = new(pool + (top++)) Node(-1, -1);
34     size = 0;
35     while (leaves.size()) leaves.pop();
36 }
37 void link(Node *u) {
38     if (nxtSuf) nxtSuf->suf = u;
39     nxtSuf = u;
40 }
41 bool walk(Node *u) {
42     int len = u->len();
43     if (curLen >= len) {
44         curP += len, curLen -= len, cur = u;
45         return true;
46     }
47     return false;
48 }
49 void extend(int c) {
50     text[++pos] = c;
51     nxtSuf = NULL;
52     ++remCnt;
53     while (remCnt) {
54         curP = curLen ? curP : pos;
55         int curE = text[curP];

```

```

56     if (!cur->ch[curE]) {
57         leaves.push(cur->addEdge(new(pool + (top++)) Node(pos)));
58         link(cur);
59     } else {
60         Node *nxt = cur->ch[curE];
61         if (walk(nxt)) continue;
62         if (text[nxt->l + curLen] == c) {
63             ++curLen;
64             link(cur);
65             break;
66         }
67         Node *split = new(pool + (top++)) Node(nxt->l, nxt->l + curLen);
68         cur->addEdge(split);
69         leaves.push(split->addEdge(new(pool + (top++)) Node(pos)));
70         nxt->l += curLen;
71         split->addEdge(nxt);
72         link(split);
73     }
74     --remCnt;
75     if (cur == root && curLen > 0) {
76         curP = pos - (--curLen);
77     } else {
78         cur = cur->suf ? cur->suf : root;
79     }
80 }
81 size += leaves.size();
82 }
83 void finish() {
84     nxtSuf = NULL;
85     for (int i = 0; i < top; ++i) if (pool[i].r == INF) link(pool + i);
86     while (remCnt > 0) {
87         if (curLen) {
88             int curE = text[curP];
89             Node *nxt = cur->ch[curE];
90             if (walk(nxt)) continue;
91             Node *split = new(pool + (top++)) Node(nxt->l, nxt->l + curLen);
92             leaves.push(cur->addEdge(split));
93             nxt->l += curLen;
94             split->addEdge(nxt);
95             link(split);
96         } else {
97             leaves.push(cur);
98             link(cur);
99         }
100     }
    --remCnt;

```

```

101     if (cur == root && curLen > 0) {
102         --curLen;
103         curP = pos - remCnt + 1;
104     } else {
105         cur = cur->suf ? cur->suf : root;
106     }
107 }
108 if (nxtSuf != root) link(root);
109 }
110 void eraseUp(Node *u) {
111     size -= u->len();
112     int ch = text[u->l];
113     u = u->fa;
114     u->ch[ch] = NULL;
115     --(u->dgr);
116 }
117 void erase() {
118     Node *u = leaves.front();
119     leaves.pop();
120     while (u->dgr == 0 && u != cur) eraseUp(u);
121     if (u == cur) {
122         if (cur->dgr == 0 && curLen == 0) {
123             int len = u->len();
124             curLen = len;
125             curP = pos - len + 1;
126             cur = cur->fa;
127             eraseUp(u);
128         }
129         if (curLen) {
130             int curE = text[curP];
131             if (!cur->ch[curE]) {
132                 Node *leaf = new(pool + (top++)) Node(pos - curLen + 1);
133                 leaves.push(cur->addEdge(leaf));
134                 size += leaf->len();
135                 --remCnt;
136                 if (cur == root && curLen > 0) {
137                     curP = pos - (--curLen) + 1;
138                 } else {
139                     cur = cur->suf ? cur->suf : root;
140                 }
141                 while (curLen && walk(cur->ch[text[curP]])) continue;
142             }
143         }
144     }
145     int n;
146     char s[N], buf[N];
147     int ord[N], stop, sord[N << 1];

```

```

146 void dfs(Node *u) {
147     sord[u - pool] = stop++;
148     for (int i = 0; i < C; ++i) {
149         if (u->ch[i]) {
150             dfs(u->ch[i]);
151         }
152     }
153 }
154 void getOrd() {
155     init();
156     for (int i = 0; i < n; ++i) extend(s[i] - 'a');
157     finish();
158     stop = 0;
159     dfs(root);
160     int i = 0;
161     while (leaves.size()) {
162         ord[i++] = sord[leaves.front() - pool];
163         leaves.pop();
164     }
165 }

```

### 字符串最小表示

```

1 std::string find(std::string s) {
2     int i,j,k,l,N=s.length(); s+=s;
3     for(i=0,j=1;j<N;){
4         for(k=0;k<N&&s[i+k]==s[j+k];k++);
5         if(k>=N) break;
6         if(s[i+k]<s[j+k]) j+=k+1;
7         else l=i+k,i=j,j=max(l,j)+1;
8     }
9     return s.substr(i,N);
10 }

```

### 轻重链剖分

```

1 struct Tree(){*root[N];
2 int father[N],size[N],depth[N];
3 int bfsOrd[N],pathId[N],ordInPath[N],sqn[N];
4 void doBfs(int s){
5     int qh=0,qt=0,*que=bfsOrd; father[s]=-1; depth[s]=0;
6     for(que[qt++]=s;qh<qt;){
7         int u=que[qh++];
8         foreach(iter,adj[u]){
9             int v=*iter; if(v==father[u]) continue;
10            father[v]=u; depth[v]=depth[u]+1; que[qt++]=v;
11        }

```

```

12 }
13 }
14 void doSplit(){
15     for(int i=N-1;i>=0;--i){
16         int u=bfs0rd[i]; size[u]=1;
17         foreach(iter,adj[u]){
18             int v=*iter; if(v==father[u]) continue; size[u]+=size[v];
19         }
20     }
21     memset(pathId,-1,sizeof pathId);
22     for(int i=0;i<N;++i){
23         int top=bfs0rd[i],cnt=0;
24         if(pathId[top]!=-1) continue;
25         for(int next,u=top;u!=-1;u=next){
26             sqn[cnt]=val[u]; ordInPath[u]=cnt; pathId[u]=top; ++cnt;
27             next=-1;
28             foreach(iter,adj[u]){
29                 int v=*iter; if(v==father[u]) continue;
30                 if(next<0||size[next]<size[v]) next=v;
31             }
32         }
33         root[top]=new Tree(0,cnt,sqn);
34     }
35 }
36 void prepare(){ doBfs(0); doSplit(); }

```

## Splay Tree

```

1 // 注意初始化内存池和 null 节点
2 struct Node{
3     int rev,size; Node *ch[2],*p;
4     void set(Node*,int); int dir(); void update(); void relax(); void appRev();
5 } nodePool[MAX_NODE],*curNode,*null;
6 Node *newNode(){
7     Node *t=curNode++; t->rev=0, t->size=1;
8     t->ch[0]=t->ch[1]=t->p=null; return t;
9 }
10 struct Splay{
11     Node *root;
12     Splay(){ root=newNode(); root->set(newNode(),0); root->update(); }
13     void rot(Node *t){
14         Node *p=t->p; int d=t->dir();
15         p->relax(); t->relax();
16         if(p==root) root=t;
17         p->set(t->ch[!d],d); p->p->set(t,p->dir()); t->set(p,!d);
18         p->update();

```

```

19     }
20     void splay(Node *t,Node *f=null){
21         for(t->relax();t->p!=f;){
22             if(t->p->p==f) rot(t);
23             else t->dir()==t->p->dir()?rot(t->p),rot(t):(rot(t),rot(t));
24             t->update();
25         }
26     };
27     void initNull(){ curNode=nodePool;null=curNode++;null->size=0; }
28     void Node::set(Node *t,int _d){ ch[_d]=t; t->p=this; }
29     int Node::dir(){ return this==p->ch[1]; }
30     void Node::update(){ size=ch[0]->size+ch[1]->size+1; }
31     void Node::relax(){ if(rev) ch[0]->appRev(), ch[1]->appRev(), rev=false; }
32     void Node::appRev(){ if(this==null) return; rev^=true; swap(ch[0],ch[1]); }

```

## Link Cut Tree

```

1 // 注意初始化 null 节点, 单点的 is_root 初始为 true
2 struct Node{
3     Node *ch[2], *p;
4     int is_root, rev;
5     bool dir();
6     void set(Node*, bool);
7     void update();
8     void relax();
9     void app_rev();
10 } *null;
11 void rot(Node *t){
12     Node *p=t->p; bool d=t->dir();
13     p->relax(); t->relax(); p->set(t->ch[!d],d);
14     if(p->is_root) t->p=p->p,swap(p->is_root,t->is_root);
15     else p->p->set(t,p->dir());
16     t->set(p,!d); p->update();
17 }
18 void splay(Node *t){
19     for(t->relax();!t->is_root;){
20         if(t->p->is_root) rot(t);
21         else t->dir()==t->p->dir()?rot(t->p),rot(t):(rot(t),rot(t));
22         t->update();
23     }
24     void access(Node *t){
25         for(Node *s=null; t!=null; s=t,t=t->p){
26             splay(t);
27             if (t->p == null) { /*TODO*/ }
28             t->ch[1]->is_root=true; s->is_root=false;
29             t->ch[1]=s; t->update();

```

```

30 }
31 }
32 bool Node::dir(){ return this==p->ch[1]; }
33 void Node::set(Node *t,bool _d){ ch[_d]=t; t->p=this; }
34 void Node::update(){ }
35 void Node::app_rev(){ if (this == null) return; rev ^= true; swap(ch[0], ch[1]); }
36 void Node::relax() { if(this==null) return; if (rev) { ch[0]->app_rev();
    ↪ ch[1]->app_rev(); rev = false; } }
37 void make_root(Node *u) { access(u); splay(u); u->app_rev(); }

```

## Dominator Tree

```

1 vector<int> prec[N], succ[N];
2 vector<int> ord;
3 int stamp, vis[N];
4 int num[N];
5 int fa[N];
6 void dfs(int u) {
7     vis[u] = stamp;
8     num[u] = ord.size();
9     ord.push_back(u);
10    for (int i = 0; i < (int)succ[u].size(); ++i) {
11        int v = succ[u][i];
12        if (vis[v] != stamp) {
13            fa[v] = u;
14            dfs(v);
15        }
16    }
17 }
18 int fs[N], mins[N], dom[N], sem[N];
19 int find(int u) {
20     if (u != fs[u]) {
21         int v = fs[u];
22         fs[u] = find(fs[u]);
23         if (mins[v] != -1 && num[sem[mins[v]]] < num[sem[mins[u]]]) {
24             mins[u] = mins[v];
25         }
26     }
27     return fs[u];
28 }
29 void merge(int u, int v) { fs[u] = v; }
30 vector<int> buf[N];
31 int buf2[N];
32 void mark(int source) {
33     ord.clear();
34     ++stamp;

```

```

35 dfs(source);
36 for (int i = 0; i < (int)ord.size(); ++i) {
37     int u = ord[i];
38     fs[u] = u, mins[u] = -1, buf2[u] = -1;
39 }
40 for (int i = (int)ord.size() - 1; i > 0; --i) {
41     int u = ord[i], p = fa[u];
42     sem[u] = p;
43     for (int j = 0; j < (int)prec[u].size(); ++j) {
44         int v = prec[u][j];
45         if (use[v] != stamp) continue;
46         if (num[v] > num[u]) {
47             find(v); v = sem[mins[v]];
48         }
49         if (num[v] < num[sem[u]]) {
50             sem[u] = v;
51         }
52     }
53     buf[sem[u]].push_back(u);
54     mins[u] = u;
55     merge(u, p);
56     while (buf[p].size()) {
57         int v = buf[p].back();
58         buf[p].pop_back();
59         find(v);
60         if (sem[v] == sem[mins[v]]) {
61             dom[v] = sem[v];
62         } else {
63             buf2[v] = mins[v];
64         }
65     }
66 }
67 dom[ord[0]] = ord[0];
68 for (int i = 0; i < (int)ord.size(); ++i) {
69     int u = ord[i];
70     if (~buf2[u]) {
71         dom[u] = dom[buf2[u]];
72     }
73 }
74 }

```

## DancingLinks

```

1 struct node{
2     node *left,*right,*up,*down,*col; int row,cnt;
3 }*head,*col[MAXC],*Node[MAXNODE],*ans[MAXNODE];

```

```

4  int totNode;
5  void insert(const std::vector<int> &V,int rownum){
6      std::vector<node*> N;
7      for(int i=0;i<int(V.size());++i){
8          node* now=Node+(totNode++); now->row=rownum;
9          now->col=now->up=col[V[i]], now->down=col[V[i]]->down;
10         now->up->down=now, now->down->up=now;
11         now->col->cnt++; N.push_back(now);
12     }
13     for(int i=0;i<int(V.size());++i)
14         N[i]->right=N[(i+1)%V.size()], N[i]->left=N[(i-1+V.size())%V.size()];
15 }
16 void Remove(node *x){
17     x->left->right=x->right, x->right->left=x->left;
18     for(node *i=x->down;i!=x;i=i->down)
19         for(node *j=i->right;j!=i;j=j->right)
20             j->up->down=j->down, j->down->up=j->up, --(j->col->cnt);
21 }
22 void Resume(node *x){
23     for(node *i=x->up;i!=x;i=i->up)
24         for(node *j=i->left;j!=i;j=j->left)
25             j->up->down=j->down->up=j, ++(j->col->cnt);
26     x->left->right=x, x->right->left=x;
27 }
28 bool search(int tot){
29     if(head->right==head) return true;
30     node *choose=NULL;
31     for(node *i=head->right;i!=head;i=i->right){
32         if(choose==NULL||choose->cnt>i->cnt) choose=i;
33         if(choose->cnt<2) break;
34     }
35     Remove(choose);
36     for(node *i=choose->down;i!=choose;i=i->down){
37         for(node *j=i->right;j!=i;j=j->right) Remove(j->col);
38         ans[tot]=i;
39         if(search(tot+1)) return true;
40         ans[tot]=NULL;
41         for(node *j=i->left;j!=i;j=j->left) Resume(j->col);
42     }
43     Resume(choose);
44     return false;
45 }
46 void prepare(int totC){
47     head=Node+totC;
48     for(int i=0;i<totC;++i) col[i]=Node+i;

```

```

49     totNode=totC+1;
50     for(int i=0;i<=totC;++i){
51         (Node+i)->right=Node+(i+1)%totC;
52         (Node+i)->left=Node+(i+totC)%totC;
53         (Node+i)->up=(Node+i)->down=Node+i;
54     }
55 }

```

### 环状最长公共子序列

```

1  int n, a[N << 1], b[N << 1];
2  bool has(int i, int j) { return a[(i - 1) % n] == b[(j - 1) % n];}
3  const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1, 0}};
4  int from[N][N];
5  int solve() {
6      memset(from, 0, sizeof(from));
7      int ret = 0;
8      for (int i = 1; i <= 2 * n; ++ i) {
9          from[i][0] = 2;
10         int left = 0, up = 0;
11         for (int j = 1; j <= n; ++ j) {
12             int upleft = up + 1 + !from[i - 1][j];
13             if (!has(i, j)) upleft = INT_MIN;
14             int max = std::max(left, std::max(upleft, up));
15             if (left == max) {
16                 from[i][j] = 0;
17             } else if (upleft == max) {
18                 from[i][j] = 1;
19             } else {
20                 from[i][j] = 2;
21             }
22             left = max;
23         }
24         if (i >= n) {
25             int count = 0;
26             for (int x = i, y = n; y;) {
27                 int t = from[x][y];
28                 count += t == 1;
29                 x += DELTA[t][0];
30                 y += DELTA[t][1];
31             }
32             ret = std::max(ret, count);
33             int x = i - n + 1, y = 0;
34             from[x][0] = 0;
35             while (y <= n && from[x][y] == 0) y++;
36             for (; x <= i; ++ x) {

```

```

37     from[x][y] = 0;
38     if (x == i) break;
39     for (; y <= n; ++ y) {
40         if (from[x + 1][y] == 2) break;
41         if (y + 1 <= n && from[x + 1][y + 1] == 1) {
42             y ++;
43             break;
44         }}}}
45     return ret;
46 }

```

### 直线下有多少个格点

```

1 LL solve(LL n,LL a,LL b,LL m){
2     // 计算 for (int i=0;i<n;++i) s+=floor((a+b*i)/m)
3     //n,m,a,b>0
4     if(b==0) return n*(a/m);
5     if(a>m) return n*(a/m)+solve(n,a%m,b,m);
6     if(b>m) return (n-1)*n/2*(b/m)+solve(n,a,b%m,m);
7     return solve((a+b*n)/m,(a+b*n)%m,m,b);
8 }

```

### 费用流

```

1 // Q is a priority_queue<PII, vector<PII>, greater<PII>>
2 // for an edge(s, t): u is the capacity, v is the cost, nxt is the next edge,
3 // op is the opposite edge
4 // this code can not deal with negative cycles
5 typedef pair<int,int> PII;
6 struct edge{ int t,u,v; edge *nxt,*op; }E[MAXE],*V[MAXV];
7 int D[MAXN], dist[MAXN], maxflow, mincost; bool in[MAXN];
8 bool modlabel(){
9     while(!Q.empty()) Q.pop();
10    for(int i=S;i<=T;++i) if(in[i]) D[i]=0,Q.push(PII(0,i)); else D[i]=inf;
11    while(!Q.empty()){
12        int x=Q.top().first,y=Q.top().second; Q.pop();
13        if(y==T) break; if(D[y]<x) continue;
14        for(edge *ii=V[y];ii;ii=ii->nxt) if(ii->u)

```

```

15        if(x+(ii->v+dist[ii->t]-dist[y])<D[ii->t]){
16            D[ii->t]=x+(ii->v+dist[ii->t]-dist[y]);
17            Q.push(PII(D[ii->t],ii->t));
18        }
19    }
20    if(D[T]==inf) return false;
21    for(int i=S;i<=T;++i) if(D[i]>D[T]) dist[i]+=D[T]-D[i];
22    return true;
23 }
24 int aug(int p,int limit){
25     if(p==T) return maxflow+=limit,mincost+=limit*dist[S],limit;
26     in[p]=1; int kk,ll=limit;
27     for(edge *ii=V[p];ii;ii=ii->nxt) if(ii->u){
28         if(!in[ii->t]&&dist[ii->t]+ii->v==dist[p]){
29             kk=aug(ii->t,min(ii->u,ll)); ll-=kk,ii->u-=kk,ii->op->u+=kk;
30             if(!ll) return in[p]=0,limit;
31         }
32     }
33     return limit-ll;
34 }
35 PII mincostFlow(){
36     for(int i=S;i<=T;++i) dist[i]=i==T?inf:0;
37     while(!Q.empty()) Q.pop(); Q.push(PII(0,T));
38     while(!Q.empty()){
39         int x=Q.top().first,y=Q.top().second; Q.pop(); if(dist[y]<x) continue;
40         for(edge *ii=V[y];ii;ii=ii->nxt) if(ii->op->u&&ii->v+x<dist[ii->t]
41             dist[ii->t]=ii->v+x,Q.push(PII(dist[ii->t],ii->t));
42     }
43     maxflow=mincost=0;
44     do{
45         do{
46             memset(in,0,sizeof(in));
47         }while(aug(S,maxflow));
48     }while(modlabel());
49     return PII(maxflow,mincost);
50 }

```



积分表

|  |  |  |  |   |   |
|--|--|--|--|---|---|
| <b>Integrals of Rational Functions</b>   | $\int \frac{1}{1+x^2} dx = \tan^{-1} x$  | $\int \frac{1}{a^2+x^2} dx = \frac{1}{a} \tan^{-1} \frac{x}{a}$  | $\int \frac{x}{a^2+x^2} dx = \frac{1}{2} \ln  a^2 + x^2 $  | $\int \frac{x^2}{a^2+x^2} dx = x - a \tan^{-1} \frac{x}{a}$   | $\int \frac{x^3}{a^2+x^2} dx = \frac{1}{2} x^2 - \frac{1}{2} a^2 \ln  a^2 + x^2 $   |
| $\int \frac{1}{ax^2+bx+c} dx = \frac{2}{\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}}$  | $\int \frac{1}{(x+a)(x+b)} dx = \frac{1}{b-a} \ln \frac{a+x}{b+x}, a \neq b$   | $\int \frac{x}{(x+a)^2} dx = \frac{a}{a+x} + \ln  a+x $  | $\int \frac{x}{ax^2+bx+c} dx = \frac{1}{2a} \ln  ax^2 + bx + c  - \frac{b}{a\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}}$  |   |   |
| <b>Integrals with Roots</b>  | $\int \frac{x}{\sqrt{x \pm a}} dx = \frac{2}{3} (x \mp 2a) \sqrt{x \pm a}$   | $\int \sqrt{\frac{x}{a-x}} dx = -\sqrt{x(a-x)} - a \tan^{-1} \frac{\sqrt{x(a-x)}}{x-a}$  | $\int \sqrt{\frac{x}{a+x}} dx = \sqrt{x(a+x)} - a \ln [\sqrt{x} + \sqrt{x+a}]$   | $\int x \sqrt{x^2 \pm a^2} dx = \frac{1}{3} (x^2 \pm a^2)^{3/2}$  |   |
| $\int x \sqrt{ax+b} dx = \frac{2}{15a^2} (-2b^2 + abx + 3a^2 x^2) \sqrt{ax+b}$   | $\int \sqrt{x(ax+b)} dx = \frac{1}{4a^{3/2}} \left[ (2ax+b) \sqrt{ax(ax+b)} - b^2 \ln \left  a\sqrt{x} + \sqrt{a(ax+b)} \right  \right]$ | $\int \sqrt{x^2 \pm a^2} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \pm \frac{1}{2} a^2 \ln \left  x + \sqrt{x^2 \pm a^2} \right $                          | $\int \sqrt{x^3(ax+b)} dx = \left[ \frac{b}{12a} - \frac{b^2}{8a^2 x} + \frac{x}{3} \right] \sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}} \ln \left  a\sqrt{x} + \sqrt{a(ax+b)} \right $ | $\int \sqrt{a^2 - x^2} dx = \frac{1}{2} x \sqrt{a^2 - x^2} + \frac{1}{2} a^2 \tan^{-1} \frac{x}{\sqrt{a^2 - x^2}}$                                      | $\int \frac{x^2}{\sqrt{x^2 \pm a^2}} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \mp \frac{1}{2} a^2 \ln \left  x + \sqrt{x^2 \pm a^2} \right $ |
| $\int \frac{1}{\sqrt{x^2 \pm a^2}} dx = \ln \left  x + \sqrt{x^2 \pm a^2} \right $   | $\int \frac{1}{\sqrt{a^2 - x^2}} dx = \sin^{-1} \frac{x}{a}$   | $\int \frac{x}{\sqrt{x^2 \pm a^2}} dx = \sqrt{x^2 \pm a^2}$  | $\int \frac{x}{\sqrt{a^2 - x^2}} dx = -\sqrt{a^2 - x^2}$   | $\int \sqrt{ax^2 + bx + c} dx = \frac{b+2ax}{4a} \sqrt{ax^2 + bx + c} + \frac{4ac-b^2}{8a^{3/2}} \ln \left  2ax + b + 2\sqrt{a(ax^2 + bx + c)} \right $ | $\int \frac{1}{\sqrt{ax^2 + bx + c}} dx = \frac{1}{\sqrt{a}} \ln \left  2ax + b + 2\sqrt{a(ax^2 + bx + c)} \right $                       |
| $\int \frac{x}{\sqrt{ax^2 + bx + c}} dx = \frac{1}{a} \sqrt{ax^2 + bx + c} - \frac{b}{2a^{3/2}} \ln \left  2ax + b + 2\sqrt{a(ax^2 + bx + c)} \right $       | $\int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2 \sqrt{a^2+x^2}}$   | <b>Integrals with Logarithms</b>   | $\int \ln(ax+b) dx = \left( x + \frac{b}{a} \right) \ln(ax+b) - x, a \neq 0$   |   |   |
| $\int \frac{\ln ax}{x} dx = \frac{1}{2} (\ln ax)^2$  | $\int \ln(x^2 + a^2) dx = x \ln(x^2 + a^2) + 2a \tan^{-1} \frac{x}{a} - 2x$  | $\int \ln(x^2 - a^2) dx = x \ln(x^2 - a^2) + a \ln \frac{x+a}{x-a} - 2x$   | $\int x \ln(ax+b) dx = \frac{bx}{2a} - \frac{1}{4} x^2 + \frac{1}{2} \left( x^2 - \frac{b^2}{a^2} \right) \ln(ax+b)$   | <b>Integrals with Exponentials</b>  |   |
| $\int \ln(ax^2 + bx + c) dx = \frac{1}{a} \sqrt{4ac - b^2} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} - 2x + \left( \frac{b}{2a} + x \right) \ln(ax^2 + bx + c)$ | $\int x \ln(a^2 - b^2 x^2) dx = -\frac{1}{2} x^2 + \frac{1}{2} \left( x^2 - \frac{a^2}{b^2} \right) \ln(a^2 - b^2 x^2)$                  |  |  |   |   |
| $\int x^n e^{ax} dx = \frac{x^n e^{ax}}{a} - \frac{n}{a} \int x^{n-1} e^{ax} dx$   | $\int x e^{-ax^2} dx = -\frac{1}{2a} e^{-ax^2}$  | <b>Integrals with Trigonometric Functions</b>  | $\int \sin^3 ax dx = -\frac{3 \cos ax}{4a} + \frac{\cos 3ax}{12a}$   | $\int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a}$   |   |
| $\int \cos^3 ax dx = \frac{3 \sin ax}{4a} + \frac{\sin 3ax}{12a}$  | $\int \cos ax \sin bxdx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b$   | $\int \sin^2 ax \cos bxdx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)}$                                       | $\int \sin^2 x \cos x dx = \frac{1}{3} \sin^3 x$   |   |   |
| $\int \cos^2 ax \sin bxdx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)}$  | $\int \cos^2 ax \sin ax dx = -\frac{1}{3a} \cos^3 ax$  | $\int \sin^2 ax \cos^2 bxdx = \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)}$ | $\int \sin^2 ax \cos^2 ax dx = \frac{x}{8} - \frac{\sin 4ax}{32a}$   |   |   |
| $\int \tan ax dx = -\frac{1}{a} \ln \cos ax$   | $\int \tan^2 ax dx = -x + \frac{1}{a} \tan ax$   | $\int \tan^3 ax dx = \frac{1}{a} \ln \cos ax + \frac{1}{2a} \sec^2 ax$   | $\int \sec x dx = \ln  \sec x + \tan x  = 2 \tanh^{-1} \left( \tan \frac{x}{2} \right)$  | $\int \sec^2 ax dx = \frac{1}{a} \tan ax$   |   |
| $\int \sec^3 x dx = \frac{1}{2} \sec x \tan x + \frac{1}{2} \ln  \sec x + \tan x $   | $\int \sec x \tan x dx = \sec x$   | $\int \sec^2 x \tan x dx = \frac{1}{2} \sec^2 x$   | $\int \sec^n x \tan x dx = \frac{1}{n} \sec^n x, n \neq 0$   | $\int \csc x dx = \ln \left  \tan \frac{x}{2} \right  = \ln  \csc x - \cot x  + C$  |   |
| $\int \csc^2 ax dx = -\frac{1}{a} \cot ax$   | $\int \csc^3 x dx = -\frac{1}{2} \cot x \csc x + \frac{1}{2} \ln  \csc x - \cot x $  | $\int \csc^n x \cot x dx = -\frac{1}{n} \csc^n x, n \neq 0$  | $\int \sec x \csc x dx = \ln  \tan x $   | <b>Products of Trigonometric Functions and Monomials</b>  |   |
| $\int x \cos x dx = \cos x + x \sin x$   | $\int x \cos ax dx = \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax$  | $\int x^2 \cos x dx = 2x \cos x + (x^2 - 2) \sin x$  | $\int x^2 \cos ax dx = \frac{2x \cos ax}{a^2} + \frac{a^2 x^2 - 2}{a^3} \sin ax$   | $\int x \sin x dx = -x \cos x + \sin x$   |   |
| $\int x \sin ax dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2}$   | $\int x^2 \sin x dx = (2 - x^2) \cos x + 2x \sin x$  | $\int x^2 \sin ax dx = \frac{2-a^2 x^2}{a^3} \cos ax + \frac{2x \sin ax}{a^2}$   | <b>Products of Trigonometric Functions and Exponentials</b>  |   |   |
| $\int e^x \sin x dx = \frac{1}{2} e^x (\sin x - \cos x)$   | $\int e^{bx} \sin ax dx = \frac{1}{a^2+b^2} e^{bx} (b \sin ax - a \cos ax)$  | $\int e^{bx} \cos ax dx = \frac{1}{a^2+b^2} e^{bx} (a \sin ax + b \cos ax)$  | $\int x e^x \sin x dx = \frac{1}{2} e^x (\cos x - x \cos x + x \sin x)$  |   |   |
| $\int x e^x \cos x dx = \frac{1}{2} e^x (x \cos x - \sin x + x \sin x)$  | $\int e^x \cos x dx = \frac{1}{2} e^x (\sin x + \cos x)$   |  |  |   |   |

Java

```

1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4
5 public class Main{
6     BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
7     PrintWriter writer = new PrintWriter(System.out);
8     StringTokenizer tokenizer = null;
9
10    void solve() throws Exception {
11    }
12    void run()throws Exception{
13        try{

```

```

14        while (true) {
15            solve();
16        }
17    }
18    catch(Exception e){
19    }
20    finally{
21        reader.close();
22        writer.close();
23    }
24 }
25 String next()throws Exception{
26     for(;tokenizer == null || !tokenizer.hasMoreTokens();){
27         tokenizer = new StringTokenizer(reader.readLine());

```

```
28     }
29     return tokenizer.nextToken();
30 }
31 int nextInt()throws Exception{
32     return Integer.parseInt(next());
33 }
34 double nextDouble()throws Exception{
35     return Double.parseDouble(next());
36 }
37 BigInteger nextBigInteger()throws Exception{
38     return new BigInteger(next());
39 }
40 public static void main(String args[])throws Exception{
41     (new Main()).run();
42 }
43 }
```

## Vimrc

```
1 \begin{lstlisting}
2 set nu ai ci si mouse=a ts=4 sts=4 sw=4
3
4 nmap<C-A> ggVG
5 vmap<C-C> "+y
6
7 nmap<F3> : vs %<.in <CR>
8 nmap<F5> : !./%< <CR>
9 nmap<F8> : !./%< <%<.in <CR>
10 nmap<F9> : !g++ % -o %< -Wall <CR>
11
12 "nmap<F4> : !gedit % <CR>
13 "autocmd BufNewFile *.cpp Or ~/temp.cpp
14 "set hlsearch incseach
15
16 "syntax on
17 "filetype plugin indent on
18 \end{lstlisting}
```