Dreadnought

Standard Code Library

May 11, 2016

Contents

二维几何

二维几何基本操作

```cpp
struct Point {
  Point rotate(const double ang) { // 逆时针旋转 ang 弧度
    return Point(cos(ang) * x - sin(ang) * y, cos(ang) * y + sin(ang) * x);
  }
  Point turn90() { // 逆时针旋转 90 度
    return Point(-y, x);
  }
};
Point isLL(const Line &l1, const Line &l2) {
  double s1 = det(l2.b - l2.a, l1.a - l2.a),
         s2 = -det(l2.b - l2.a, l1.b - l2.a);
  return (l1.a * s2 + l1.b * s1) / (s1 + s2);
}
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) { // 点到直线投影
  return l.a + (l.b - l.a) * (dot(p - l.a, l.b - l.a) / (l.b - l.a).len2());
}
double disToLine(const Line &l, const Point &p) {
  return abs(det(p - l.a, l.b - l.a) / (l.b - l.a).len());
}
double 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) : min((p - l.a).len(), (p - l.b).len());
}
Point symmetryPoint(const Point a, const Point b) { // 点 b 关于点 a 的中心对称点
  return a + a - b;
}
Point reflection(const Line &l, const Point &p) { // 点关于直线的对称点
  return symmetryPoint(projection(l, p), p);
}
// 求圆与直线的交点
bool isCL(Circle a, Line l, Point &p1, Point &p2) {
  double 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;
  d = max(d, 0.0);
  Point p = l.a - ((l.b - l.a) * (x / y)), delta = (l.b - l.a) * (sqrt(d) / y);
  p1 = p + delta, p2 = p - delta;
  return true;
}
// 求圆与圆的交面积
double areaCC(const Circle &c1, const Circle &c2) {
  double d = (c1.o - c2.o).len();
  if (sign(d - (c1.r + c2.r)) >= 0) {
    return 0;
  }
  if (sign(d - abs(c1.r - c2.r)) <= 0) {
    double r = min(c1.r, c2.r);
    return r * r * PI;
  }
  double 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);
}
// 求圆与圆的交点，注意调用前要先判定重圆
bool isCC(Circle a, Circle b, Point &p1, Point &p2) {
  double s1 = (a.o - b.o).len();
  if (sign(s1 - a.r - b.r) > 0 || sign(s1 - abs(a.r - b.r)) < 0) return false;
  double s2 = (a.r * a.r - b.r * b.r) / s1;
  double aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
  Point o = (b.o - a.o) * (aa / (aa + bb)) + a.o;
  Point delta = (b.o - a.o).unit().turn90() * newSqrt(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;
}
```

```cpp
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

```cpp
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 }
```

三角形的心

```cpp
Point inCenter(const Point &A, const Point &B, const Point &C) { // 内心
  double a = (B - C).len(), b = (C - A).len(), c = (A - B).len(),
    s = fabs(det(B - A, C - A)),
    r = s / p;
  return (A * a + B * b + C * c) / (a + b + c);
}
Point circumCenter(const Point &a, const Point &b, const Point &c) { // 外心
  Point bb = b - a, cc = c - a;
  double db = bb.len2(), dc = cc.len2(), d = 2 * det(bb, cc);
  return a - Point(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
}
Point othroCenter(const Point &a, const Point &b, const Point &c) { // 垂心
  Point ba = b - a, ca = c - a, bc = b - c;
  double Y = ba.y * ca.y * bc.y,
      A = ca.x * ba.y - ba.x * ca.y,
      x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
      y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
  return Point(x0, y0);
}
```

圆与多边形面积交

```cpp
double areaCT(Point pa, Point pb, double r) {
  if (pa.len() < pb.len()) swap(pa, pb);
  if (sign(pb.len()) == 0) return 0;
  double a = pb.len(), b = pa.len(), c = (pb - pa).len();
  double sinB = fabs(det(pb, pb - pa) / a / c),
       cosB = dot(pb, pb - pa) / a / c,
       sinC = fabs(det(pa, pb) / a/ b),
       cosC = dot(pa, pb) / a / b;
  double B = atan2(sinB, cosB), C = atan2(sinC, cosC);
  if (a > r) {
    S = C / 2 * r * r;
    h = a * b * sinC / c;
    if (h < r && B < PI / 2) {
      S -= (acos(h / r) * r * r - h * sqrt(r * r - h * h));
    }
  } else if (b > r) {
    double theta = PI - B - asin(sinB / r * a);
    S = a * r * sin(theta) / 2 + (C - theta) / 2 * r * r;
  } else {
    S = sinC * a * b / 2;
  }
  return S;
}
```

圆的面积模板 $(n^2 \log n)$

```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; }
int C;
Circle c[N];
double area[N];
void solve() {
  memset(area, 0, sizeof(double) * (C + 1));
  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);
      }
    }
```

```
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 }}}}
```

凸包快速询问

```
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，多解返回任意一个口 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 };
```

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++)
           ↪ Triangle(Point(-LOTS,-LOTS),Point(+LOTS,-LOTS),Point(0,+LOTS));
58     }
59     TriangleRef find(Point p) const { return find(the_root,p); }
60     void add_point(Point const& p) { add_point(find(the_root,p),p); }
61   private:
62     TriangleRef the_root;
63     static TriangleRef find(TriangleRef root, Point const& p) {
64       for( ; ; ) {
65         if (!root->has_children()) return root;
66         else for (int i = 0; i < 3 && root->children[i] ; ++i)
67           if (root->children[i]->contains(p))
```

```cpp
68          {root = root->children[i]; break;}
69      }
70    }
71    void add_point(TriangleRef root, Point const& p) {
72      TriangleRef tab,tbc,tca;
73      tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
74      tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
75      tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
76      set_edge(Edge(tab,0),Edge(tbc,1));set_edge(Edge(tbc,0),Edge(tca,1));
77      set_edge(Edge(tca,0),Edge(tab,1));set_edge(Edge(tab,2),root->edge[2]);
78      set_edge(Edge(tbc,2),root->edge[0]);set_edge(Edge(tca,2),root->edge[1]);
79      root->children[0]=tab;root->children[1]=tbc;root->children[2]=tca;
80      flip(tab,2); flip(tbc,2); flip(tca,2);
81    }
82    void flip(TriangleRef tri, SideRef pi) {
83      TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
84      if(!trj||!in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
85      TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj],
          ↪ tri->p[pi]);
86      TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi],
          ↪ trj->p[pj]);
87      set_edge(Edge(trk,0), Edge(trl,0));
88      set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2),
          ↪ trj->edge[(pj+1)%3]);
89      set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2),
          ↪ tri->edge[(pi+1)%3]);
90      tri->children[0]=trk;tri->children[1]=trl;tri->children[2]=0;
91      trj->children[0]=trk;trj->children[1]=trl;trj->children[2]=0;
92      flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2);
93    }
94 };
95 int n; Point ps[N];
96 void build(){
97   tot_triangles = triange_pool; cin >> n;
98   for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
99   random_shuffle(ps, ps + n); Triangulation tri;
100  for(int i = 0; i < n; ++ i) tri.add_point(ps[i]);
101 }
```

三维几何
三维几何基本操作

```cpp
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   return sign(det(a - b, a - c).len() - det(p - a, p - b).len() - det(p - b, p -
       ↪ c).len() - det(p - c, p - a).len()) == 0;
13 }
14 // 共平面的两点是否在这平面上一条直线的同侧
15 bool sameSide(const Point3D &a, const Point3D &b, const Point3D &p0, const Point3D &p1)
     ↪ {
16   return sign(dot(det(a - b, p0 - b), det(a - b, p1 - b))) > 0;
17 }
18 // 两点在平面同侧 ： 点积法向量符号相同
19 // 两直线平行 / 垂直 ： 同二维
20 // 平面平行 / 垂直 ： 判断法向量
21 // 线面垂直 ： 法向量和直线平行
22 // 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
23 // 线段和三角形是否相交 ： 线段在三角形平面不同侧
          ↪ 三角形任意两点在线段和第三点组成的平面的不同侧
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 ： 平行时随便取一点求距离，否则叉积方向向量得到方向点积计算长度
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
```

三维凸包求重心

```
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     }
7   } info[1005];
8   int mark[1005][1005],n, cnt;;
9   double mix(const Point &a, const Point &b, const Point &c) {
10    return a.dot(b.cross(c));
11  }
12  double area(int a, int b, int c) {
13    return ((info[b] - info[a]).cross(info[c] - info[a])).length();
14  }
15  double volume(int a, int b, int c, int d) {
16    return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);
17  }
18  struct Face {
19    int a, b, c; Face() {}
20    Face(int a, int b, int c): a(a), b(b), c(c) {}
21    int &operator [](int k) {
22      if (k == 0) return a; if (k == 1) return b; return c;
23    }
24  };
25  vector <Face> face;
26  inline void insert(int a, int b, int c) {
27    face.push_back(Face(a, b, c));
28  }
29  void add(int v) {
30    vector <Face> tmp; int a, b, c; cnt++;
31    for (int i = 0; i < SIZE(face); i++) {
32      a = face[i][0]; b = face[i][1]; c = face[i][2];
33      if (Sign(volume(v, a, b, c)) < 0)
34        mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
35      else tmp.push_back(face[i]);
36    } face = tmp;
37    for (int i = 0; i < SIZE(tmp); i++) {
38      a = face[i][0]; b = face[i][1]; c = face[i][2];
39      if (mark[a][b] == cnt) insert(b, a, v);
40      if (mark[b][c] == cnt) insert(c, b, v);
41      if (mark[c][a] == cnt) insert(a, c, v);
42    }
43  }
44  int Find() {
45    for (int i = 2; i < n; i++) {
46      Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
47      if (ndir == Point()) continue; swap(info[i], info[2]);
48      for (int j = i + 1; j < n; j++) if (Sign(volume(0, 1, 2, j)) != 0) {
49        swap(info[j], info[3]); insert(0, 1, 2); insert(0, 2, 1); return 1;
50      }
51    }
52    return 0;
53  }
54  int main() {
55    for (; scanf("%d", &n) == 1; ) {
56      for (int i = 0; i < n; i++) info[i].Input();
57      sort(info, info + n); n = unique(info, info + n) - info;
58      face.clear(); random_shuffle(info, info + n);
59      if (Find()) {
60        memset(mark, 0, sizeof(mark)); cnt = 0;
61        for (int i = 3; i < n; i++) add(i); vector<Point> Ndir;
62        for (int i = 0; i < SIZE(face); ++i) {
63          Point p = (info[face[i][0]] - info[face[i][1]]).cross(info[face[i][2]] -
                  ↪ info[face[i][1]]);
64          p = p / p.length(); Ndir.push_back(p);
65        }
66        sort(Ndir.begin(), Ndir.end());
67        int ans = unique(Ndir.begin(), Ndir.end()) - Ndir.begin();
68        printf("%d\n", ans);
69      } else printf("1\n");
70    }
71  }
72  // 求重心
73  double calcDist(const Point &p, int a, int b, int c) {
74    return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c));
75  }
76  //compute the minimal distance of center of any faces
77  double findDist() { //compute center of mass
78    double totalWeight = 0;
79    Point center(.0, .0, .0);
80    Point first = info[face[0][0]];
81    for (int i = 0; i < SIZE(face); ++i) {
82      Point p = (info[face[i][0]]+info[face[i][1]]+info[face[i][2]]+first)*.25;
83      double weight = mix(info[face[i][0]] - first, info[face[i][1]] - first,
                  ↪ info[face[i][2]] - first);
84      totalWeight += weight; center = center + p * weight;
85    }
86    center = center / totalWeight;
87    double res = 1e100; //compute distance
88    for (int i = 0; i < SIZE(face); ++i)
89      res = min(res, calcDist(center, face[i][0], face[i][1], face[i][2]));
90      return res;
91  }
```

最小覆盖球

```cpp
int nouter; Tpoint outer[4], res; double radius;
void ball() {
  Tpoint q[3]; double m[3][3], sol[3], L[3], det;
  int i,j; res.x = res.y = res.z = radius = 0;
  for (i=0; i<3; ++i) q[i]=outer[i+1]-outer[0], sol[i]=dot(q[i], q[i]);
  for (i=0;i<3;++i) for(j=0;j<3;++j) m[i][j]=dot(q[i],q[j])*2;
  det= m[0][0]*m[1][1]*m[2][2]
  + m[0][1]*m[1][2]*m[2][0]
  + m[0][2]*m[2][1]*m[1][0]
  - m[0][2]*m[1][1]*m[2][0]
  - m[0][1]*m[1][0]*m[2][2]
  - m[0][0]*m[1][2]*m[2][1];
  if ( fabs(det)<eps ) return;
  for (j=0; j<3; ++j) {
    for (i=0; i<3; ++i) m[i][j]=sol[i];
    L[j]=( m[0][0]*m[1][1]*m[2][2]
    + m[0][1]*m[1][2]*m[2][0]
    + m[0][2]*m[2][1]*m[1][0]
    - m[0][2]*m[1][1]*m[2][0]
    - m[0][1]*m[1][0]*m[2][2]
    - m[0][0]*m[1][2]*m[2][1]
    ) / det;
    for (i=0; i<3; ++i) m[i][j]=dot(q[i], q[j])*2;
  } res=outer[0];
  for (i=0; i<3; ++i ) res = res + q[i] * L[i];
  radius=dist2(res, outer[0]);
}
```

图论
Hungarian

```cpp
// 最小匹配, 自带初始化 n <= m 方案存在 p[] 中
const int N = 105;
const int INF = 1000000000; // 严格大于最大边权
int n, m, a[N][N];
int u[N], v[N], p[N], fa[N], minv[N];
bool used[N];
int km() {
  memset(u, 0, sizeof(int) * n);
  for (int i = 0; i <= m; ++i) v[i] = 0, p[i] = n;
  for (int i = 0; i < n; ++i) {
    p[m] = i;
    int j0 = m;
    for (int j = 0; j <= m; ++j) minv[j] = INF, used[j] = false;
    do {
      used[j0] = true;
      int i0 = p[j0], delta = INF, j1;
      for (int j = 0; j < m; ++j) {
        if (!used[j]) {
          int cur = a[i0][j] - u[i0] - v[j];
          if (cur < minv[j]) minv[j] = cur, fa[j] = j0;
          if (minv[j] < delta) delta = minv[j], j1 = j;
        }
      }
      for (int j = 0; j <= m; ++j) {
        if (used[j]) {
          u[p[j]] += delta, v[j] -= delta;
        } else {
          minv[j] -= delta;
        }
      }
      j0 = j1;
    } while (p[j0] != n);
    do {
      int j1 = fa[j0];
      p[j0] = p[j1];
      j0 = j1;
    } while (j0 != m);
  }
  return -v[m];
}
```

Hopcroft

```cpp
// 左侧 N 个点, 右侧 K 个点 , 1-based, 初始化将 matx[],maty[] 都置为 0
int N, K;
int que[N], dx[N], dy[N], matx[N], maty[N];
int BFS()
{
  int flag = 0, qt = 0, qh = 0;
  for(int i = 1; i <= K; ++ i) dy[i] = 0;
  for(int i = 1; i <= N; ++ i) {
    dx[i] = 0;
    if (! matx[i]) que[qt ++] = i;
  }
  while (qh < qt) {
    int u = que[qh ++];
    for(Edge *e = E[u]; e; e = e->n)
      if (! dy[e->t]) {
        dy[e->t] = dx[u] + 1;
        if (! maty[e->t]) flag = true;
        else {
          dx[maty[e->t]] = dx[u] + 2;
          que[qt ++] = maty[e->t];
        }
      }
  }
}
```

```
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  }
```

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, AnswserLength)
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 };
```

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 }
```

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 }
```

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 }
```

主流

```
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 }
```

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 }
```

数论
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(n2%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,
       ↪ 3474749660383LL,341550071728321LL,0,0,0,0};
18    if(n<2||n==3215031751LL) return 0;
19    for(int i=0;i<12;++i){
20      if(n<lim[i]) return 1;
21      if(strong_pseudo_primetest(n,testNum[i])==0) return 0;
22    }
23    return 1;
24 }
```

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 }
```

直线下整点个数

```
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  }
```

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 mod 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         }
```

```
18        }
19        cout << A << ' ' << B <<endl;
20      }
21    }
22 }
```

代数
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 }
31 Complex A[N], B[N], C[N], D[N];
32 void mul(int a[N], int b[N]) {
33   for (int i = 0; i < N; ++i) {
34     A[i] = Complex(a[i] >> L, a[i] & MASK);
35     B[i] = Complex(b[i] >> L, b[i] & MASK);
36   }
37   FFT(A, N), FFT(B, N);
38   for (int i = 0; i < N; ++i) {
39     int j = (N - i) % N;
40     Complex da = (A[i] - conj(A[j])) * Complex(0, -0.5),
41         db = (A[i] + conj(A[j])) * Complex(0.5, 0),
```

```
42         dc = (B[i] - conj(B[j])) * Complex(0, -0.5),
43         dd = (B[i] + conj(B[j])) * Complex(0.5, 0);
44     C[j] = da * dd + da * dc * Complex(0, 1);
45     D[j] = db * dd + db * dc * Complex(0, 1);
46   }
47   FFT(C, N), FFT(D, N);
48   for (int i = 0; i < N; ++i) {
49     long long da = (long long)(C[i].imag() / N + 0.5) % MOD,
50         db = (long long)(C[i].real() / N + 0.5) % MOD,
51         dc = (long long)(D[i].imag() / N + 0.5) % MOD,
52         dd = (long long)(D[i].real() / N + 0.5) % MOD;
53     a[i] = ((dd << (L * 2)) + ((db + dc) << L) + da) % MOD;
54   }
55 }
```

线性规划

```
1  // 求 max{cx | Ax ≤ b, x ≥ 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;
```

```cpp
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  }
```

Schreier-Sims

```cpp
 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  }
```

字符串
后缀数组（倍增）

```cpp
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 }
```

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]] ==
         ↪ text[q + palindrome[i]]) {
9        palindrome[i] ++;
10     }
11     if (q + palindrome[i] - 1 > r) {
12       j = i;
13     }
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 < l; ++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 < l && s[j] == s[i + j]) {
30         ++j;
31       }
32       next[i] = j;
33       k = i;
34     }
35   }
36 }
```

字符串最小表示

```cpp
std::string find(std::string s) {
  int i,j,k,l,N=s.length(); s+=s;
  for(i=0,j=1;j<N;){
    for(k=0;k<N&&s[i+k]==s[j+k];k++);
    if(k>=N) break;
    if(s[i+k]<s[j+k]) j+=k+1;
    else l=i+k,i=j,j=max(l,j)+1;
  }
  return s.substr(i,N);
}
```

后缀树 (With Pop Front)

```cpp
int pos, text[N];
struct Node {
  int l, r;
  Node *suf, *ch[C];
  int dgr;
  Node *fa;
  Node (int l = -1, int r = INF) : l(l), r(r) {
    suf = fa = NULL;
    memset(ch, 0, sizeof(ch));
    dgr = 0;
  }
  Node* addEdge(Node *t) {
    int c = text[t->l];
    dgr += !ch[c];
    ch[c] = t;
    t->fa = this;
    return t;
  }
  int len() {
    return min(r, pos + 1) - l;
  }
};

int top;
Node pool[N << 1], *root, *nxtSuf, *cur;
int remCnt, curP, curLen;
long long size;
queue<Node*> leaves;
void init() {
  top = 0, pos = -1;
  remCnt = 0, curP = 0, curLen = 0;
  nxtSuf = NULL;
  root = cur = new(pool + (top++)) Node(-1, -1);
  size = 0;
  while (leaves.size()) leaves.pop();
}
void link(Node *u) {
  if (nxtSuf) nxtSuf->suf = u;
  nxtSuf = u;
}
bool walk(Node *u) {
  int len = u->len();
  if (curLen >= len) {
    curP += len, curLen -= len, cur = u;
    return true;
  }
  return false;
}
void extend(int c) {
  text[++pos] = c;
  nxtSuf = NULL;
  ++remCnt;
  while (remCnt) {
    curP = curLen ? curP : pos;
    int curE = text[curP];
    if (!cur->ch[curE]) {
      leaves.push(cur->addEdge(new(pool + (top++)) Node(pos)));
      link(cur);
    } else {
      Node *nxt = cur->ch[curE];
      if (walk(nxt)) continue;
      if (text[nxt->l + curLen] == c) {
        ++curLen;
        link(cur);
        break;
      }
      Node *split = new(pool + (top++)) Node(nxt->l, nxt->l + curLen);
      cur->addEdge(split);
      leaves.push(split->addEdge(new(pool + (top++)) Node(pos)));
      nxt->l += curLen;
      split->addEdge(nxt);
      link(split);
    }
    --remCnt;
    if (cur == root && curLen > 0) {
      curP = pos - (--curLen);
    } else {
      cur = cur->suf ? cur->suf : root;
    }
  }
  size += leaves.size();
}
void finish() {
  nxtSuf = NULL;
  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   int n;
144   char s[N], buf[N];
145   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   }
```

数据结构
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(); }
```

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=bfsOrd[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=bfsOrd[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(); }
```

综合

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+1);
52     (Node+i)->left=Node+(i+totC)%(totC+1);
53     (Node+i)->up=(Node+i)->down=Node+i;
54   }
55 }
```

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  }
```

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 ++; break;
43          }
44        }
45      }
46    }
47  }
48  return ret;
49 }
```

经纬度球面距离

```
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 }
11 int main(){
12   int L, H, W, x1, y1, z1, x2, y2, z2;
13   cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
14   if (z1!=0 && z1!=H) if (y1==0 || y1==W)
15       swap(y1,z1), std::swap(y2,z2), std::swap(W,H);
16   else swap(x1,z1), std::swap(x2,z2), std::swap(L,H);
17   if (z1==H) z1=0, z2=H-z2;
18   r=0x3fffffff;
19   turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
20   cout<<r<<endl;
21 }
```

其他
简易积分表

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

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

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

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

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

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

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

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

$\int x^n e^{ax} \, dx = \frac{x^n e^{ax}}{a} - \frac{n}{a} \int x^{n-1} e^{ax} \, dx$

$\int \sin^2 ax dx = \frac{x}{2} - \frac{1}{4a} \sin 2ax$ $\int \sin^3 ax dx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a}$

$\int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a}$ $\int \cos^3 ax dx = \frac{3\sin ax}{4a} + \frac{\sin 3ax}{12a}$

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

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

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

常用结论

弦图

设 $next(v)$ 表示 $N(v)$ 中最前的点 . 令 $w*$ 表示所有满足 $A \in B$ 的 w 中最后的一个点 , 判断 $v \cup N(v)$ 是否为极大团 , 只需判断是否存在一个 $w \in w*$, 满足 $Next(w) = v$ 且 $|N(v)| + 1 \leq |N(w)|$ 即可 .

五边形数

$\prod_{n=1}^{\infty} (1 - x^n) = \sum_{n=0}^{\infty} (-1)^n (1 - x^{2n+1}) x^{n(3n+1)/2}$

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 $\frac{4r\sin^3(\theta/2)}{3(\theta-\sin(\theta))}$

第二类 Bernoulli number

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

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

Stirling 数

第一类 :n 个元素的项目分作 k 个环排列的方法数目

$$s(n,k) = (-1)^{n+k} |s(n,k)|$$
$$|s(n,0)| = 0$$
$$|s(1,1)| = 1$$
$$|s(n,k)| = |s(n-1,k-1)| + (n-1) * |s(n-1,k)|$$

第二类 :n 个元素的集定义 k 个等价类的方法数

$$S(n,1) = S(n,n) = 1$$
$$S(n,k) = S(n-1,k-1) + k * S(n-1,k)$$

三角公式

$\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b \quad \cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$

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

$\sin(a) + \sin(b) = 2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2}) \quad \sin(a) - \sin(b) = 2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$

$\cos(a) + \cos(b) = 2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2}) \quad \cos(a) - \cos(b) = -2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$

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

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

Java 读入优化

```java
import java.io.*;
import java.util.*;
import java.math.*;

public class Main{
  BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
  PrintWriter writer = new PrintWriter(System.out);
  StringTokenizer tokenizer = null;
```

```java
  void solve() throws Exception {
  }
  void run()throws Exception{
    try{
      while (true) {
        solve();
      }
    }
    catch(Exception e){
    }
    finally{
      reader.close();
      writer.close();
    }
  }
  String next()throws Exception{
    for(;tokenizer == null || !tokenizer.hasMoreTokens();){
      tokenizer = new StringTokenizer(reader.readLine());
    }
    return tokenizer.nextToken();
  }
  int nextInt()throws Exception{
    return Integer.parseInt(next());
  }
  double nextDouble()throws Exception{
    return Double.parseDouble(next());
  }
  BigInteger nextBigInteger()throws Exception{
    return new BigInteger(next());
  }
  public static void main(String args[])throws Exception{
    (new Main()).run();
  }
}
```

Vimrc

```vim
set nu si mouse=a sw=4 ts=4 sts=4
set mp=g++\ -O2\ -Wall\ --std=c++11\ -Wno-unused-result\ %:r.cpp\ -o\ %:r
nmap <F2> :vs %:r.in <CR>
nmap <F8> :!time ./%:r < %:r.in <CR>
nmap <F9> :w <CR> :make<CR>
```