Contents

1 计算几何

1.1 二维计算几何基本操作

```cpp
const double PI = 3.14159265358979323846264338327950288;
double arcSin(const double &a) {
  return (a <= -1.0) ? (-PI / 2) : ((a >= 1.0) ? (PI / 2) : (asin(a))); }
double arcCos(const double &a) {
  return (a <= -1.0) ? (PI) : ((a >= 1.0) ? (0) : (acos(a))); }
struct point { double x, y; // something omitted
  point rot(const double &a) const { // counter-clockwise
    return point(x * cos(a) - y * sin(a), x * sin(a) + y * cos(a)); }
  point rot90()  const { return point(-y,  x); } // counter-clockwise
  point project(const point &p1, const point &p2) const {
    const point &q = *this; return p1 + (p2 - p1) * (dot(p2 - p1, q - p1) / (p2 - p1).norm()); }
  bool onSeg(const point &a, const point &b) const { // a, b inclusive
    const point &c = *this; return sign(dot(a - c, b - c)) <= 0 && sign(det(b - a, c - a)) == 0; }
  double distLP(const point &p1, const point &p2) const { // dist from *this to line p1->p2
    const point &q = *this; return fabs(det(p2 - p1, q - p1)) / (p2 - p1).len(); }
  double distSP(const point &p1, const point &p2) const { // dist from *this to segment [p1, p2]
    const point &q = *this;
    if (dot(p2 - p1, q - p1) < EPS) return (q - p1).len();
    if (dot(p1 - p2, q - p2) < EPS) return (q - p2).len();
    return distLP(p1, p2);
  }
  bool inAngle(const point &p1, const point &p2) const { // det(p1, p2) ≥ 0
    const point &q = *this; return det(p1, q) > -EPS && det(p2, q) < EPS;
  }
};
bool lineIntersect(const point &a, const point &b, const point &c, const point &d, point &e) {
  double s1 = det(c - a, d - a), s2 = det(d - b, c - b);
  if (!sign(s1 + s2)) return false; e = (b - a) * (s1 / (s1 + s2)) + a; return true;
}
int segIntersectCheck(const point &a, const point &b, const point &c, const point &d, point &o) {
  static double s1, s2, s3, s4;
  static int iCnt;
  int d1 = sign(s1 = det(b - a, c - a)), d2 = sign(s2 = det(b - a, d - a));
  int d3 = sign(s3 = det(d - c, a - c)), d4 = sign(s4 = det(d - c, b - c));
  if ((d1 ^ d2) == -2 && (d3 ^ d4) == -2) {
    o = (c * s2 - d * s1) / (s2 - s1); return true;
  } iCnt = 0;
  if (d1 == 0 && c.onSeg(a, b)) o = c, ++iCnt;
  if (d2 == 0 && d.onSeg(a, b)) o = d, ++iCnt;
  if (d3 == 0 && a.onSeg(c, d)) o = a, ++iCnt;
  if (d4 == 0 && b.onSeg(c, d)) o = b, ++iCnt;
  return iCnt ? 2 : 0; // 不相交返回 0, 严格相交返回 1, 非严格相交返回 2
}
struct circle {
  point o; double r, rSquare;
  bool inside(const point &a) { return (a - o).len() < r + EPS; } // 非严格
  bool contain(const circle &b) const { return sign(b.r + (o - b.o).len() - r) <= 0; } // 非严格
  bool disjunct(const circle &b) const { return sign(b.r + r - (o - b.o).len()) <= 0; } // 非严格
  int isCL(const point &p1, const point &p2, point &a, point &b) const {
    double x = dot(p1 - o, p2 - p1), y = (p2 - p1).norm();
    double d = x * x - y * ((p1 - o).norm() - rSquare);
    if (d < -EPS) return 0; if (d < 0) d = 0;
    point q1 = p1 - (p2 - p1) * (x / y);
    point q2 = (p2 - p1) * (sqrt(d) / y);
    a = q1 - q2; b = q1 + q2; return q2.len() < EPS ? 1 : 2;
  }
  int tanCP(const point &p, point &a, point &b) const { // 返回切点, 注意可能与 p 重合
    double x = (p - o).norm(), d = x - rSquare;
    if (d < -EPS) return 0; if (d < 0) d = 0;
    point q1 = (p - o) * (rSquare / x), q2 = ((p - o) * (-r * sqrt(d) / x)).rot90();
    a = o + (q1 - q2); b = o + (q1 + q2); return q2.len() < EPS ? 1 : 2;
  }
};
bool checkCrossCS(const circle &cir, const point &p1, const point &p2) { // 非严格
  const point &c = cir.o; const double &r = cir.r;
  return c.distSP(p1, p2) < r + EPS && (r < (c - p1).len() + EPS || r < (c - p2).len() + EPS);
}
bool checkCrossCC(const circle &cir1, const circle &cir2) { // 非严格
  double &r1 = cir1.r, &r2 = cir2.r, d = (cir1.o - cir2.o).len();
  return d < r1 + r2 + EPS && fabs(r1 - r2) < d + EPS;
}
int isCC(const circle &cir1, const circle &cir2, point &a, point &b) {
  const point &c1 = cir1.o, &c2 = cir2.o;
  double x = (c1 - c2).norm(), y = ((cir1.rSquare - cir2.rSquare) / x + 1) / 2;
  double d = cir1.rSquare / x - y * y;
  if (d < -EPS) return 0; if (d < 0) d = 0;
  point q1 = c1 + (c2 - c1) * y, q2 = ((c2 - c1) * sqrt(d)).rot90();
  a = q1 - q2; b = q1 + q2; return q2.len() < EPS ? 1 : 2;
}
vector<pair<point, point> > tanCC(const circle &cir1, const circle &cir2) {
```

```cpp
// 注意: 如果只有三条切线, 即 s1 = 1, s2 = 1, 返回的切线可能重复, 切点没有问题
  vector<pair<point, point> > list;
  if (cir1.contain(cir2) || cir2.contain(cir1)) return list;
  const point &c1 = cir1.o, &c2 = cir2.o;
  double r1 = cir1.r, r2 = cir2.r; point p, a1, b1, a2, b2; int s1, s2;
  if (sign(r1 - r2) == 0) {
    p = c2 - c1; p = (p * (r1 / p.len())).rot90();
    list.push_back(make_pair(c1 + p, c2 + p)); list.push_back(make_pair(c1 - p, c2 - p));
  } else {
    p = (c2 * r1 - c1 * r2) / (r1 - r2);
    s1 = cir1.tanCP(p, a1, b1); s2 = cir2.tanCP(p, a2, b2);
    if (s1 >= 1 && s2 >= 1)
      list.push_back(make_pair(a1, a2)), list.push_back(make_pair(b1, b2));
  } p = (c1 * r2 + c2 * r1) / (r1 + r2);
  s1 = cir1.tanCP(p, a1, b1); s2 = cir2.tanCP(p, a2, b2);
  if (s1 >= 1 && s2 >= 1)
    list.push_back(make_pair(a1, a2)), list.push_back(make_pair(b1, b2));
  return list;
}
bool distConvexPIn(const point &p1, const point &p2, const point &p3, const point &p4, const point &q) {
  point o12 = (p1 - p2).rot90(), o23 = (p2 - p3).rot90(), o34 = (p3 - p4).rot90();
  return (q - p2).inAngle(o12, o23) || (q - p3).inAngle(o23, o34)
    || ((q - p2).inAngle(o23, p3 - p2) && (q - p3).inAngle(p2 - p3, o23));
}
double distConvexP(int n, point ps[], const point &q) { // 外部点到多边形的距离
  int left = 0, right = n; while (right - left > 1) { int mid = (left + right) / 2;
    if (distConvexPIn(ps[(left + n - 1) % n], ps[left], ps[mid], ps[(mid + 1) % n], q))
      right = mid; else left = mid;
  } return q.distSP(ps[left], ps[right % n]);
}
double areaCT(const circle &cir, point pa, point pb) {
  pa = pa - cir.o; pb = pb - cir.o; double R = cir.r;
  if (pa.len() < pb.len()) swap(pa, pb); if (pb.len() < EPS) return 0;
  point pc = pb - pa; double a = pa.len(), b = pb.len(), c = pc.len(), S, h, theta;
  double cosB = dot(pb, pc) / b / c, B = acos(cosB);
  double cosC = dot(pa, pb) / a / b, C = acos(cosC);
  if (b > R) {
    S = C * 0.5 * R * R; h = b * a * sin(C) / c;
    if (h < R && B < PI * 0.5) S -= acos(h / R) * R * R - h * sqrt(R * R - h * h);
  } else if (a > R) {
    theta = PI - B - asin(sin(B) / R * b);
    S = 0.5 * b * R * sin(theta) + (C - theta) * 0.5 * R * R;
  } else S = 0.5 * sin(C) * b * a;
  return S;
}
circle minCircle(const point &a, const point &b) {
  return circle((a + b) * 0.5, (b - a).len() * 0.5);
}
circle minCircle(const point &a, const point &b, const point &c) { // 钝角三角形没有被考虑
  double a2( (b - c).norm() ), b2( (a - c).norm() ), c2( (a - b).norm() );
  if (b2 + c2 <= a2 + EPS) return minCircle(b, c);
  if (a2 + c2 <= b2 + EPS) return minCircle(a, c);
  if (a2 + b2 <= c2 + EPS) return minCircle(a, b);
  double A = 2.0 * (a.x - b.x), B = 2.0 * (a.y - b.y);
  double D = 2.0 * (a.x - c.x), E = 2.0 * (a.y - c.y);
  double C = a.norm() - b.norm(), F = a.norm() - c.norm();
  point p((C * E - B * F) / (A * E - B * D), (A * F - C * D) / (A * E - B * D));
  return circle(p, (p - a).len());
}
circle minCircle(point P[], int N) { // 1-based
  if (N == 1) return circle(P[1], 0.0);
  random_shuffle(P + 1, P + N + 1); circle O = minCircle(P[1], P[2]);
  Rep(i, 1, N) if(!O.inside(P[i])) { O = minCircle(P[1], P[i]);
    Foru(j, 1, i) if(!O.inside(P[j])) { O = minCircle(P[i], P[j]);
      Foru(k, 1, j) if(!O.inside(P[k])) O = minCircle(P[i], P[j], P[k]); }
  } return O;
}
```

1.2 圆的面积模板

```cpp
struct Event { point p; double alpha; int add; // 构造函数省略
  bool operator < (const Event &other) const { return alpha < other.alpha; } };
void circleKCover(circle *c, int N, double *area) { // area[k] : 至少被覆盖 k 次
  static bool overlap[MAXN][MAXN], g[MAXN][MAXN];
  Rep(i, 0, N + 1) area[i] = 0.0; Rep(i, 1, N) Rep(j, 1, N) overlap[i][j] = c[i].contain(c[j]);
  Rep(i, 1, N) Rep(j, 1, N) g[i][j] = !(overlap[i][j] || overlap[j][i] || c[i].disjunct(c[j]));
  Rep(i, 1, N) { static Event events[MAXN * 2 + 1]; int totE = 0, cnt = 1;
    Rep(j, 1, N) if (j != i && overlap[j][i]) ++cnt;
    Rep(j, 1, N) if (j != i && g[i][j]) {
      circle &a = c[i], &b = c[j]; double l = (a.o - b.o).norm();
      double s = ((a.r - b.r) * (a.r + b.r) / l + 1) * 0.5;
      double t = sqrt(-(l - sqr(a.r - b.r)) * (l - sqr(a.r + b.r)) / (l * l * 4.0));
```

```
13        point dir = b.o - a.o, nDir = point(-dir.y, dir.x);
14        point aa = a.o + dir * s + nDir * t;
15        point bb = a.o + dir * s - nDir * t;
16        double A = atan2(aa.y - a.o.y, aa.x - a.o.x);
17        double B = atan2(bb.y - a.o.y, bb.x - a.o.x);
18        events[totE++] = Event(bb, B, 1); events[totE++] = Event(aa, A, -1); if (B > A) ++cnt;
19      } if (totE == 0) { area[cnt] += PI * c[i].rSquare; continue; }
20      sort(events, events + totE); events[totE] = events[0];
21      Foru(j, 0, totE) {
22        cnt += events[j].add; area[cnt] += 0.5 * det(events[j].p, events[j + 1].p);
23        double theta = events[j + 1].alpha - events[j].alpha; if (theta < 0) theta += 2.0 * PI;
24        area[cnt] += 0.5 * c[i].rSquare * (theta - sin(theta));
25  }}}
```

1.3 多边形相关

```
1   struct Polygon { // stored in [0, n)
2     int n; point list[MAXN];
3     Polygon cut(const point &a, const point &b) {
4       static Polygon res;
5       static point o;
6       res.n = 0;
7       for (int i = 0; i < n; ++i) {
8         int s1 = sign(det(list[i] - a, b - a));
9         int s2 = sign(det(list[(i + 1) % n] - a, b - a));
10        if (s1 <= 0) res.list[res.n++] = list[i];
11        if (s1 * s2 < 0) {
12          lineIntersect(a, b, list[i], list[(i + 1) % n], o);
13          res.list[res.n++] = o;
14        }
15      } return res;
16    }
17    bool contain(const point &p) const { // 1 if on border or inner, 0 if outter
18      static point A, B;
19      int res = 0;
20      for (int i = 0; i < n; ++i) {
21        A = list[i]; B = list[(i + 1) % n];
22        if (p.onSeg(A, B)) return 1;
23        if (sign(A.y - B.y) <= 0) swap(A, B);
24        if (sign(p.y - A.y) > 0) continue;
25        if (sign(p.y - B.y) <= 0) continue;
26        res += (int)(sign(det(B - p, A - p)) > 0);
27      }
28      return res & 1;
29    }
30    bool convexContain(const point &p) const { // sort by polar angle
31      for (int i = 1; i < n; ++i) list[i] = list[i] - list[0];
32      point q = p - list[0];
33      if (sign(det(list[1], q)) < 0 || sign(det(list[n - 1], q)) > 0) return false;
34      int l = 2, r = n - 1;
35      while (l <= r) {
36        int mid = (l + r) >> 1;
37        double d1 = sign(det(list[mid], q)), d2 = sign(det(list[mid - 1], q));
38        if (d1 <= 0) {
39          if (d2 <= 0) {
40            if (sign(det(q - list[mid - 1], list[mid] - list[mid - 1]) <= 0) <= 0)
41              return true;
42          } else r = mid - 1;
43        } else l = mid + 1;
44      }
45      return false;
46    }
47    double isPLAtan2(const point &a, const point &b) {
48      double k = (b - a).alpha(); if (k < 0) k += 2 * PI;
49      return k;
50    }
51    point isPL_Get(const point &a, const point &b, const point &s1, const point &s2) {
52      double k1 = det(b - a, s1 - a), k2 = det(b - a, s2 - a);
53      if (sign(k1) == 0) return s1;
54      if (sign(k2) == 0) return s2;
55      return (s1 * k2 - s2 * k1) / (k2 - k1);
56    }
57    int isPL_Dic(const point &a, const point &b, int l, int r) {
58      int s = (det(b - a, list[l] - a) < 0) ? -1 : 1;
59      while (l <= r) {
60        int mid = (l + r) / 2;
61        if (det(b - a, list[mid] - a) * s <= 0) r = mid - 1;
62        else l = mid + 1;
63      }
64      return r + 1;
65    }
66    int isPL_Find(double k, double w[]) {
67      if (k <= w[0] || k > w[n - 1]) return 0;
```

```
68      int l = 0, r = n - 1, mid;
69      while (l <= r) {
70        mid = (l + r) / 2;
71        if (w[mid] >= k) r = mid - 1;
72        else l = mid + 1;
73      } return r + 1;
74    }
75    bool isPL(const point &a, const point &b, point &cp1, point &cp2) {  // O(logN)
76      static double w[MAXN * 2]; // pay attention to the array size
77      for (int i = 0; i <= n; ++i) list[i + n] = list[i];
78      for (int i = 0; i < n; ++i) w[i] = w[i + n] = isPLAtan2(list[i], list[i + 1]);
79      int i = isPL_Find(isPLAtan2(a, b), w);
80      int j = isPL_Find(isPLAtan2(b, a), w);
81      double k1 = det(b - a, list[i] - a), k2 = det(b - a, list[j] - a);
82      if (sign(k1) * sign(k2) > 0) return false; // no intersection
83      if (sign(k1) == 0 || sign(k2) == 0) { // intersect with a point or a line in the convex
84        if (sign(k1) == 0) {
85          if (sign(det(b - a, list[i + 1] - a)) == 0) cp1 = list[i], cp2 = list[i + 1];
86          else cp1 = cp2 = list[i];
87          return true;
88        }
89        if (sign(k2) == 0) {
90          if (sign(det(b - a, list[j + 1] - a)) == 0) cp1 = list[j], cp2 = list[j + 1];
91          else cp1 = cp2 = list[j];
92        }
93        return true;
94      }
95      if (i > j) swap(i, j);
96      int x = isPL_Dic(a, b, i, j), y = isPL_Dic(a, b, j, i + n);
97      cp1 = isPL_Get(a, b, list[x - 1], list[x]);
98      cp2 = isPL_Get(a, b, list[y - 1], list[y]);
99      return true;
100   }
101   double getI(const point &O) const {
102     if (n <= 2) return 0;
103     point G(0.0, 0.0);
104     double S = 0.0, I = 0.0;
105     for (int i = 0; i < n; ++i) {
106       const point &x = list[i], &y = list[(i + 1) % n];
107       double d = det(x, y);
108       G = G + (x + y) * d / 3.0;
109       S += d;
110     } G = G / S;
111     for (int i = 0; i < n; ++i) {
112       point x = list[i] - G, y = list[(i + 1) % n] - G;
113       I += fabs(det(x, y)) * (x.norm() + dot(x, y) + y.norm());
114     }
115     return I = I / 12.0 + fabs(S * 0.5) * (O - G).norm();
116   }
117 };
```

1.4 半平面交

```
1   struct Border {
2     point p1, p2; double alpha;
3     Border() : p1(), p2(), alpha(0.0) {}
4     Border(const point &a, const point &b): p1(a), p2(b), alpha( atan2(p2.y - p1.y, p2.x - p1.x) ) {}
5     bool operator == (const Border &b) const { return sign(alpha - b.alpha) == 0; }
6     bool operator < (const Border &b) const {
7       int c = sign(alpha - b.alpha); if (c != 0) return c > 0;
8       return sign(det(b.p2 - b.p1, p1 - b.p1)) >= 0;
9     }
10  };
11  point isBorder(const Border &a, const Border &b) { // a and b should not be parallel
12    point is; lineIntersect(a.p1, a.p2, b.p1, b.p2, is); return is;
13  }
14  bool checkBorder(const Border &a, const Border &b, const Border &me) {
15    point is; lineIntersect(a.p1, a.p2, b.p1, b.p2, is);
16    return sign(det(me.p2 - me.p1, is - me.p1)) > 0;
17  }
18  double HPI(int N, Border border[]) {
19    static Border que[MAXN * 2 + 1]; static point ps[MAXN];
20    int head = 0, tail = 0, cnt = 0; // [head, tail)
21    sort(border, border + N); N = unique(border, border + N) - border;
22    for (int i = 0; i < N; ++i) {
23      Border &cur = border[i];
24      while (head + 1 < tail && !checkBorder(que[tail - 2], que[tail - 1], cur)) --tail;
25      while (head + 1 < tail && !checkBorder(que[head], que[head + 1], cur)) ++head;
26      que[tail++] = cur;
27    } while (head + 1 < tail && !checkBorder(que[tail - 2], que[tail - 1], que[head])) --tail;
28    while (head + 1 < tail && !checkBorder(que[head], que[head + 1], que[tail - 1])) ++head;
29    if (tail - head <= 2) return 0.0;
```

```
30      Foru(i, head, tail) ps[cnt++] = isBorder(que[i], que[(i + 1 == tail) ? (head) : (i + 1)]);
31      double area = 0; Foru(i, 0, cnt) area += det(ps[i], ps[(i + 1) % cnt]);
32      return fabs(area * 0.5); // or (-area * 0.5)
33  }
```

1.5 最大面积空凸包

```
1   inline bool toUpRight(const point &a, const point &b) {
2     int c = sign(b.y - a.y); if (c > 0) return true;
3     return c == 0 && sign(b.x - a.x) > 0;
4   }
5   inline bool cmpByPolarAngle(const point &a, const point &b) { // counter-clockwise, shorter first if they
            share the same polar angle
6     int c = sign(det(a, b)); if (c != 0) return c > 0;
7     return sign(b.len() - a.len()) > 0;
8   }
9   double maxEmptyConvexHull(int N, point p[]) {
10    static double dp[MAXN][MAXN];
11    static point vec[MAXN];
12    static int seq[MAXN]; // empty triangles formed with (0,0),vec[o],vec[seq[i]]
13    double ans = 0.0;
14    Rep(o, 1, N) {
15      int totVec = 0;
16      Rep(i, 1, N) if (toUpRight(p[o], p[i])) vec[++totVec] = p[i] - p[o];
17      sort(vec + 1, vec + totVec + 1, cmpByPolarAngle);
18      Rep(i, 1, totVec) Rep(j, 1, totVec) dp[i][j] = 0.0;
19      Rep(k, 2, totVec) {
20        int i = k - 1;
21        while (i > 0 && sign( det(vec[k], vec[i]) ) == 0) --i;
22        int totSeq = 0;
23        for (int j; i > 0; i = j) {
24          seq[++totSeq] = i;
25          for (j = i - 1; j > 0 && sign(det(vec[i] - vec[k], vec[j] - vec[k])) > 0; --j);
26          double v = det(vec[i], vec[k]) * 0.5;
27          if (j > 0) v += dp[i][j];
28          dp[k][i] = v;
29          cMax(ans, v);
30        } for (int i = totSeq - 1; i >= 1; --i) cMax( dp[k][ seq[i] ], dp[k][seq[i + 1]] );
31      }
32    } return ans;
33  }
```

1.6 最近点对

```
1   int N; point p[maxn];
2   bool cmpByX(const point &a, const point &b) { return sign(a.x - b.x) < 0;  }
3   bool cmpByY(const int &a, const int &b) { return p[a].y < p[b].y;  }
4   double minimalDistance(point *c, int n, int *ys) {
5     double ret = 1e+20;
6     if (n < 20) {
7       Foru(i, 0, n) Foru(j, i + 1, n) cMin(ret, (c[i] - c[j]).len() );
8       sort(ys, ys + n, cmpByY); return ret;
9     } static int mergeTo[maxn];
10    int mid = n / 2; double xmid = c[mid].x;
11    ret = min(minimalDistance(c, mid, ys), minimalDistance(c + mid, n - mid, ys + mid));
12    merge(ys, ys + mid, ys + mid, ys + n, mergeTo, cmpByY);
13    copy(mergeTo, mergeTo + n, ys);
14    Foru(i, 0, n) {
15      while (i < n && sign(fabs(p[ys[i]].x - xmid) - ret) > 0) ++i;
16      int cnt = 0;
17      Foru(j, i + 1, n)
18        if (sign(p[ys[j]].y - p[ys[i]].y - ret) > 0) break;
19        else if (sign(fabs(p[ys[j]].x - xmid) - ret) <= 0) {
20          ret = min(ret, (p[ys[i]] - p[ys[j]]).len());
21          if (++cnt >= 10) break;
22        }
23    } return ret;
24  }
25  double work() {
26    sort(p, p + n, cmpByX); Foru(i, 0, n) ys[i] = i; return minimalDistance(p, n, ys);
27  }
```

1.7 凸包与点集直径

```
1   vector<point> convexHull(int n, point ps[]) { // counter-clockwise, strict
2     static point qs[MAXN * 2];
3     sort(ps, ps + n, cmpByXY);
4     if (n <= 2) return vector(ps, ps + n);
5     int k = 0;
6     for (int i = 0; i < n; qs[k++] = ps[i++])
7       while (k > 1 && det(qs[k - 1] - qs[k - 2], ps[i] - qs[k - 1]) < EPS) --k;
8     for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
9       while (k > t && det(qs[k - 1] - qs[k - 2], ps[i] - qs[k - 1]) < EPS) --k;
10    return vector<point>(qs, qs + k);
11  }
12  double convexDiameter(int n, point ps[]) {
13    if (n < 2) return 0; if (n == 2) return (ps[1] - ps[0]).len();
14    double k, ans = 0;
15    for (int x = 0, y = 1, nx, ny; x < n; ++x) {
16      for(nx = (x == n - 1) ? (0) : (x + 1); ; y = ny) {
17        ny = (y == n - 1) ? (0) : (y + 1);
18        if ( sign(k = det(ps[nx] - ps[x], ps[ny] - ps[y])) <= 0) break;
19      } ans = max(ans, (ps[x] - ps[y]).len());
20      if (sign(k) == 0) ans = max(ans, (ps[x] - ps[ny]).len());
21    } return ans;
22  }
```

1.8 Farmland

```
1   struct node { int begin[MAXN], *end; } a[MAXN]; // 按对 p[i] 的极角的 atan2 值排序
2   bool check(int n, point p[], int b1, int b2, bool vis[MAXN][MAXN]) {
3     static pii l[MAXN * 2 + 1]; static bool used[MAXN];
4     int tp(0), *k, p, p1, p2; double area(0.0);
5     for (l[0] = pii(b1, b2); ; ) {
6       vis[p1 = l[tp].first][p2 = l[tp].second] = true;
7       area += det(p[p1], p[p2]);
8       for (k = a[p2].begin; k != a[p2].end; ++k) if (*k == p1) break;
9       k = (k == a[p2].begin) ? (a[p2].end - 1) : (k - 1);
10      if ((l[++tp] = pii(p2, *k)) == l[0]) break;
11    } if (sign(area) < 0 || tp < 3) return false;
12    Rep(i, 1, n) used[i] = false;
13    for (int i = 0; i < tp; ++i) if (used[p = l[i].first]) return false; else used[p] = true;
14    return true; // a face with tp vertices
15  }
16  int countFaces(int n, point p[]) {
17    static bool vis[MAXN][MAXN]; int ans = 0;
18    Rep(x, 1, n) Rep(y, 1, n) vis[x][y] = false;
19    Rep(x, 1, n) for (int *itr = a[x].begin; itr != a[x].end; ++itr) if (!vis[x][*itr])
20      if (check(n, p, x, *itr, vis)) ++ans;
21    return ans;
22  }
```

1.9 Voronoi 图

不能有重点, 点数应当不小于 2

```
1   #define Oi(e) ((e)->oi)
2   #define Dt(e) ((e)->dt)
3   #define On(e) ((e)->on)
4   #define Op(e) ((e)->op)
5   #define Dn(e) ((e)->dn)
6   #define Dp(e) ((e)->dp)
7   #define Other(e, p) ((e)->oi == p ? (e)->dt : (e)->oi)
8   #define Next(e, p) ((e)->oi == p ? (e)->on : (e)->dn)
9   #define Prev(e, p) ((e)->oi == p ? (e)->op : (e)->dp)
10  #define V(p1, p2, u, v) (u = p2->x - p1->x, v = p2->y - p1->y)
11  #define C2(u1, v1, u2, v2) (u1 * v2 - v1 * u2)
12  #define C3(p1, p2, p3) ((p2->x - p1->x) * (p3->y - p1->y) - (p2->y - p1->y) * (p3->x - p1->x))
13  #define Dot(u1, v1, u2, v2) (u1 * u2 + v1 * v2)
14  #define dis(a,b) (sqrt( (a->x - b->x) * (a->x - b->x) + (a->y - b->y) * (a->y - b->y) ))
15  const int maxn = 110024;
16  const int aix = 4;
17  const double eps = 1e-7;
18  int n, M, k;
19  struct gEdge {
20    int u, v; double w;
21    bool operator <(const gEdge &e1) const { return w < e1.w - eps; }
22  } E[aix * maxn], MST[maxn];
23  struct point {
24    double x, y; int index; edge *in;
25    bool operator <(const point &p1) const { return x < p1.x - eps || (abs(x - p1.x) <= eps && y < p1.y -
            eps); }
```

```
26  };
27  struct edge { point *oi, *dt; edge *on, *op, *dn, *dp; };
28
29  point p[maxn], *Q[maxn];
30  edge mem[aix * maxn], *elist[aix * maxn];
31  int nfree;
32  void Alloc_memory() { nfree = aix * n; edge *e = mem; for (int i = 0; i < nfree; i++) elist[i] = e++; }
33  void Splice(edge *a, edge *b, point *v) {
34      edge *next;
35      if (Oi(a) == v) next = On(a), On(a) = b; else next = Dn(a), Dn(a) = b;
36      if (Oi(next) == v) Op(next) = b; else Dp(next) = b;
37      if (Oi(b) == v) On(b) = next, Op(b) = a; else Dn(b) = next, Dp(b) = a;
38  }
39  edge *Make_edge(point *u, point *v) {
40      edge *e = elist[--nfree];
41      e->on = e->op = e->dn = e->dp = e; e->oi = u; e->dt = v;
42      if (!u->in) u->in = e;
43      if (!v->in) v->in = e;
44      return e;
45  }
46  edge *Join(edge *a, point *u, edge *b, point *v, int side) {
47      edge *e = Make_edge(u, v);
48      if (side == 1) {
49          if (Oi(a) == u) Splice(Op(a), e, u);
50          else Splice(Dp(a), e, u);
51          Splice(b, e, v);
52      } else {
53          Splice(a, e, u);
54          if (Oi(b) == v) Splice(Op(b), e, v);
55          else Splice(Dp(b), e, v);
56      } return e;
57  }
58  void Remove(edge *e) {
59      point *u = Oi(e), *v = Dt(e);
60      if (u->in == e) u->in = e->on;
61      if (v->in == e) v->in = e->dn;
62      if (Oi(e->on) == u) e->on->op = e->op; else e->on->dp = e->op;
63      if (Oi(e->op) == u) e->op->on = e->on; else e->op->dn = e->on;
64      if (Oi(e->dn) == v) e->dn->op = e->dp; else e->dn->dp = e->dp;
65      if (Oi(e->dp) == v) e->dp->on = e->dn; else e->dp->dn = e->dn;
66      elist[nfree++] = e;
67  }
68  void Low_tangent(edge *e_l, point *o_l, edge *e_r, point *o_r, edge **l_low, point **OL, edge **r_low,
        point **OR) {
69      for (point *d_l = Other(e_l, o_l), *d_r = Other(e_r, o_r); ; )
70          if (C3(o_l, o_r, d_l) < -eps)       e_l = Prev(e_l, d_l), o_l = d_l, d_l = Other(e_l, o_l);
71          else if (C3(o_l, o_r, d_r) < -eps) e_r = Next(e_r, d_r), o_r = d_r, d_r = Other(e_r, o_r);
72          else break;
73      *OL = o_l, *OR = o_r; *l_low = e_l, *r_low = e_r;
74  }
75  void Merge(edge *lr, point *s, edge *rl, point *u, edge **tangent) {
76      double l1, l2, l3, l4, r1, r2, r3, r4, cot_L, cot_R, u1, v1, u2, v2, n1, cot_n, P1, cot_P;
77      point *O, *D, *OR, *OL; edge *B, *L, *R;
78      Low_tangent(lr, s, rl, u, &L, &OL, &R, &OR);
79      for (*tangent = B = Join(L, OL, R, OR, 0), O = OL, D = OR; ; ) {
80          edge *El = Next(B, O), *Er = Prev(B, D), *next, *prev;
81          point *l = Other(El, O), *r = Other(Er, D);
82          V(l, O, l1, l2); V(l, D, l3, l4); V(r, O, r1, r2); V(r, D, r3, r4);
83          double cl = C2(l1, l2, l3, l4), cr = C2(r1, r2, r3, r4);
84          bool BL = cl > eps, BR = cr > eps;
85          if (!BL && !BR) break;
86          if (BL) {
87              double dl = Dot(l1, l2, l3, l4);
88              for (cot_L = dl / cl; ; Remove(El), El = next, cot_L = cot_n) {
89                  next = Next(El, O); V(Other(next, O), O, u1, v1); V(Other(next, O), D, u2, v2);
90                  n1 = C2(u1, v1, u2, v2); if (!(n1 > eps)) break;
91                  cot_n = Dot(u1, v1, u2, v2) / n1;
92                  if (cot_n > cot_L) break;
93              }
94          } if (BR) {
95              double dr = Dot(r1, r2, r3, r4);
96              for (cot_R = dr / cr; ; Remove(Er), Er = prev, cot_R = cot_P) {
97                  prev = Prev(Er, D); V(Other(prev, D), O, u1, v1); V(Other(prev, D), D, u2, v2);
98                  P1 = C2(u1, v1, u2, v2); if (!(P1 > eps)) break;
99                  cot_P = Dot(u1, v1, u2, v2) / P1;
100                 if (cot_P > cot_R) break;
101             }
102         } l = Other(El, O); r = Other(Er, D);
103         if (!BL || (BL && BR && cot_R < cot_L)) B = Join(B, O, Er, r, 0), D = r;
104         else B = Join(El, l, B, D, 0), O = l;
105     }
106 }
107 void Divide(int s, int t, edge **L, edge **R) {
108     edge *a, *b, *c, *ll, *lr, *rl, *rr, *tangent;
109     int n = t - s + 1;
110     if (n == 2) *L = *R = Make_edge(Q[s], Q[t]);
111     else if (n == 3) {
```

```
112         a = Make_edge(Q[s], Q[s + 1]), b = Make_edge(Q[s + 1], Q[t]);
113         Splice(a, b, Q[s + 1]);
114         double v = C3(Q[s], Q[s + 1], Q[t]);
115         if (v > eps)      c = Join(a, Q[s], b, Q[t], 0), *L = a, *R = b;
116         else if (v < -eps) c = Join(a, Q[s], b, Q[t], 1), *L = c, *R = c;
117         else *L = a, *R = b;
118     } else if (n > 3) {
119         int split = (s + t) / 2;
120         Divide(s, split, &ll, &lr); Divide(split + 1, t, &rl, &rr);
121         Merge(lr, Q[split], rl, Q[split + 1], &tangent);
122         if (Oi(tangent) == Q[s]) ll = tangent;
123         if (Dt(tangent) == Q[t]) rr = tangent;
124         *L = ll; *R = rr;
125     }
126 }
127 void Make_Graph() {
128     edge *start, *e; point *u, *v;
129     for (int i = 0; i < n; i++) {
130         start = e = (u = &p[i])->in;
131         do{ v = Other(e, u);
132             if (u < v) E[M++].u = (u - p, v - p, dis(u, v)); // M < aix * maxn
133         } while ((e = Next(e, u)) != start);
134     }
135 }
136 int b[maxn];
137 int Find(int x) { while (x != b[x]) { b[x] = b[b[x]]; x = b[x]; } return x; }
138 void Kruskal() {
139     memset(b, 0, sizeof(b)); sort(E, E + M);
140     for (int i = 0; i < n; i++) b[i] = i;
141     for (int i = 0, kk = 0; i < M && kk < n - 1; i++) {
142         int m1 = Find(E[i].u), m2 = Find(E[i].v);
143         if (m1 != m2) b[m1] = m2, MST[kk++] = E[i];
144     }
145 }
146 void solve() {
147     scanf("%d", &n);
148     for (int i = 0; i < n; i++) scanf("%lf%lf", &p[i].x, &p[i].y), p[i].index = i, p[i].in = NULL;
149     Alloc_memory(); sort(p, p + n);
150     for (int i = 0; i < n; i++) Q[i] = p + i;
151     edge *L, *R; Divide(0, n - 1, &L, &R);
152     M = 0; Make_Graph(); Kruskal();
153 }
154 int main() { solve(); return 0; }
```

1.10 三维计算几何基本操作

```
1   struct point { double x, y, z; // something omitted
2       friend point det(const point &a, const point &b) {
3           return point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
4       }
5       friend double mix(const point &a, const point &b, const point &c) {
6           return a.x * b.y * c.z + a.y * b.z * c.x + a.z * b.x * c.y - a.z * b.y * c.x - a.y
            * b.x * c.z;
7       }
8       double distLP(const point &p1, const point &p2) const {
9           return det(p2 - p1, *this - p1).len() / (p2 - p1).len();
10      }
11      double distFP(const point &p1, const point &p2, const point &p3) const {
12          point n = det(p2 - p1, p3 - p1); return fabs( dot(n, *this - p1) / n.len() );
13      }
14  };
15  double distLL(const point &p1, const point &p2, const point &q1, const point &q2) {
16      point p = q1 - p1, u = p2 - p1, v = q2 - q1;
17      double d = u.norm() * v.norm() - dot(u, v) * dot(u, v);
18      if (sign(d) == 0) return p1.distLP(q1, q2);
19      double s = (dot(p, u) * v.norm() - dot(p, v) * dot(u, v)) / d;
20      return (p1 + u * s).distLP(q1, q2);
21  }
22  double distSS(const point &p1, const point &p2, const point &q1, const point &q2) {
23      point p = q1 - p1, u = p2 - p1, v = q2 - q1;
24      double d = u.norm() * v.norm() - dot(u, v) * dot(u, v);
25      if (sign(d) == 0) return min( min((p1 - q1).len(), (p1 - q2).len()),
26                                    min((p2 - q1).len(), (p2 - q2).len()));
27      double s1 = (dot(p, u) * v.norm() - dot(p, v) * dot(u, v)) / d;
28      double s2 = (dot(p, v) * u.norm() - dot(p, u) * dot(u, v)) / d;
29      if (s1 < 0.0) s1 = 0.0; if (s1 > 1.0) s1 = 1.0;
30      if (s2 < 0.0) s2 = 0.0; if (s2 > 1.0) s2 = 1.0;
31      point r1 = p1 + u * s1; point r2 = q1 + v * s2;
32      return (r1 - r2).len();
33  }
34  bool isFL(const point &p, const point &o, const point &q1, const point &q2, point &res) {
35      double a = dot(o, q2 - p), b = dot(o, q1 - p), d = a - b;
```

```
36    if (sign(d) == 0) return false;
37    res = (q1 * a - q2 * b) / d;
38    return true;
39  }
40  bool isFF(const point &p1, const point &o1, const point &p2, const point &o2, point &a, point &b) {
41    point e = det(o1, o2), v = det(o1, e);
42    double d = dot(o2, v); if (sign(d) == 0) return false;
43    point q = p1 + v * (dot(o2, p2 - p1) / d);
44    a = q; b = q + e;
45    return true;
46  }
```

1.11 凸多面体切割

```
1  vector<vector<point> > convexCut(const vector<vector<point> > &pss, const point &p, const point &o) {
2    vector<vector<point> > res;
3    vector<point> sec;
4    for (unsigned itr = 0, size = pss.size(); itr < size; ++itr) {
5      const vector<point> &ps = pss[itr];
6      int n = ps.size();
7      vector<point> qs;
8      bool dif = false;
9      for (int i = 0; i < n; ++i) {
10        int d1 = sign( dot(o, ps[i] - p) );
11        int d2 = sign( dot(o, ps[(i + 1) % n] - p) );
12        if (d1 <= 0) qs.push_back(ps[i]);
13        if (d1 * d2 < 0) {
14          point q;
15          isFL(p, o, ps[i], ps[(i + 1) % n], q); // must return true
16          qs.push_back(q);
17          sec.push_back(q);
18        }
19        if (d1 == 0) sec.push_back(ps[i]);
20        else dif = true;
21        dif |= dot(o, det(ps[(i + 1) % n] - ps[i], ps[(i + 2) % n] - ps[i])) < -EPS;
22      }
23      if (!qs.empty() && dif)
24        res.insert(res.end(), qs.begin(), qs.end());
25    }
26    if (!sec.empty()) {
27      vector<point> tmp( convexHull2D(sec, o) );
28      res.insert(res.end(), tmp.begin(), tmp.end());
29    }
30    return res;
31  }
32
33  vector<vector<point> > initConvex() {
34    vector<vector<point> > pss(6, vector<point>(4));
35    pss[0][0] = pss[1][0] = pss[2][0] = point(-INF, -INF, -INF);
36    pss[0][3] = pss[1][1] = pss[5][2] = point(-INF, -INF,  INF);
37    pss[0][1] = pss[2][3] = pss[4][2] = point(-INF,  INF, -INF);
38    pss[0][2] = pss[5][3] = pss[4][1] = point(-INF,  INF,  INF);
39    pss[1][3] = pss[2][1] = pss[3][2] = point( INF, -INF, -INF);
40    pss[1][2] = pss[5][1] = pss[3][3] = point( INF, -INF,  INF);
41    pss[2][2] = pss[4][3] = pss[3][1] = point( INF,  INF, -INF);
42    pss[5][0] = pss[4][0] = pss[3][0] = point( INF,  INF,  INF);
43    return pss;
44  }
```

1.12 三维凸包

不能有重点

```
1  namespace ConvexHull3D {
2    #define volume(a, b, c, d) (mix(ps[b] - ps[a], ps[c] - ps[a], ps[d] - ps[a]))
3    vector<Facet> getHull(int n, point ps[]) {
4      static int mark[MAXN][MAXN], a, b, c; int stamp = 0; bool exist = false;
5      vector<Facet> facet; random_shuffle(ps, ps + n);
6      for (int i = 2; i < n && !exist; i++) {
7        point ndir = det(ps[0] - ps[i], ps[1] - ps[i]);
8        if (ndir.len() < EPS) continue;
9        swap(ps[i], ps[2]); for (int j = i + 1; j < n && !exist; j++)
10          if (sign(volume(0, 1, 2, j)) != 0) {
11            exist = true; swap(ps[j], ps[3]);
12            facet.push_back(Facet(0, 1, 2)); facet.push_back(Facet(0, 2, 1));
13          }
14      } if (!exist) return ConvexHull2D(n, ps);
15      for (int i = 0; i < n; ++i) for (int j = 0; j < n; ++j) mark[i][j] = 0;
16      stamp = 0; for (int v = 3; v < n; ++v) {
17        vector<Facet> tmp; ++stamp;
18        for (unsigned i = 0; i < facet.size(); i++) {
19          a = facet[i].a; b = facet[i].b; c = facet[i].c;
20          if (sign(volume(v, a, b, c)) < 0)
21            mark[a][b] = mark[a][c] = mark[b][a] = mark[b][c] =  mark[c][a] = mark[c][b] = stamp;
22          else tmp.push_back(facet[i]);
23        } facet = tmp;
24        for (unsigned i = 0; i < tmp.size(); i++) {
25          a = facet[i].a; b = facet[i].b; c = facet[i].c;
26          if (mark[a][b] == stamp) facet.push_back(Facet(b, a, v));
27          if (mark[b][c] == stamp) facet.push_back(Facet(c, b, v));
28          if (mark[c][a] == stamp) facet.push_back(Facet(a, c, v));
29        }
30      } return facet;
31    }
32    #undef volume
33  }
34  namespace Gravity {
35    using ConvexHull3D::Facet;
36    point findG(point ps[], const vector<Facet> &facet) {
37      double ws = 0; point res(0.0, 0.0, 0.0), o = ps[ facet[0].a ];
38      for (int i = 0, size = facet.size(); i < size; ++i) {
39        const point &a = ps[ facet[i].a ], &b = ps[ facet[i].b ], &c = ps[ facet[i].c ];
40        point p = (a + b + c + o) * 0.25; double w = mix(a - o, b - o, c - o);
41        ws += w; res = res + p * w;
42      } res = res / ws;
43      return res;
44    }
45  }
```

1.13 长方体表面点距离

```
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) r = min(r, x * x + y * y);
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 calc(int L, int H, int W, int x1, int y1, int z1, int x2, int y2, int z2) {
12   if (z1 != 0 && z1 != H)
13     if (y1 == 0 || y1 == W) swap(y1, z1), swap(y2, z2), swap(W, H);
14     else                    swap(x1, z1), swap(x2, z2), swap(L, H);
15   if (z1 == H) z1 = 0, z2 = H - z2;
16   r = INF; turn(0, 0, x2 - x1, y2 - y1, z2, -x1, -y1, L, W, H);
17   return r;
18 }
```

1.14 最小覆盖球

```
1  namespace MinBall {
2    int outCnt;
3    point out[4], res;
4    double radius;
5    void ball() {
6      static point q[3];
7      static double m[3][3], sol[3], L[3], det;
8      int i, j;
9      res = point(0.0, 0.0, 0.0);
10     radius = 0.0;
11     switch (outCnt) {
12     case 1:
13       res = out[0];
14       break;
15     case 2:
16       res = (out[0] + out[1]) * 0.5;
17       radius = (res - out[0]).norm();
18       break;
19     case 3:
20       q[0] = out[1] - out[0];
21       q[1] = out[2] - out[0];
22       for (i = 0; i < 2; ++i)
23         for (j = 0; j < 2; ++j)
24           m[i][j] = dot(q[i], q[j]) * 2.0;
```

```
25    for (i = 0; i < 2; ++i)
26       sol[i] = dot(q[i], q[i]);
27    det = m[0][0] * m[1][1] - m[0][1] * m[1][0];
28    if (sign(det) == 0)
29       return;
30    L[0] = (sol[0] * m[1][1] - sol[1] * m[0][1]) / det;
31    L[1] = (sol[1] * m[0][0] - sol[0] * m[1][0]) / det;
32    res = out[0] + q[0] * L[0] + q[1] * L[1];
33    radius = (res - out[0]).norm();
34    break;
35  case 4:
36    q[0] = out[1] - out[0];
37    q[1] = out[2] - out[0];
38    q[2] = out[3] - out[0];
39    for (i = 0; i < 3; ++i)
40       for (j = 0; j < 3; ++j)
41          m[i][j] = dot(q[i], q[j]) * 2;
42    for (i = 0; i < 3; ++i)
43       sol[i] = dot(q[i], q[i]);
44    det = m[0][0] * m[1][1] * m[2][2] + m[0][1] * m[1][2] * m[2][0]
45       + m[0][2] * m[2][1] * m[1][0] - m[0][2] * m[1][1] * m[2][0]
46       - m[0][1] * m[1][0] * m[2][2] - m[0][0] * m[1][2] * m[2][1];
47    if (sign(det) == 0)
48       return;
49    for (j = 0; j < 3; ++j) {
50       for (i = 0; i < 3; ++i)
51          m[i][j] = sol[i];
52       L[j] = (m[0][0] * m[1][1] * m[2][2] + m[0][1] * m[1][2] * m[2][0]
53          + m[0][2] * m[2][1] * m[1][0] - m[0][2] * m[1][1] * m[2][0]
54          - m[0][1] * m[1][0] * m[2][2] - m[0][0] * m[1][2] * m[2][1])
55          / det;
56       for (i = 0; i < 3; ++i)
57          m[i][j] = dot(q[i], q[j]) * 2;
58    }
59    res = out[0];
60    for (i = 0; i < 3; ++i)
61       res += q[i] * L[i];
62    radius = (res - out[0]).norm();
63    }
64  }
65
66  void minball(int n, point pt[]) {
67    ball();
68    if (outCnt < 4)
69       for (int i = 0; i < n; ++i)
70          if ((res - pt[i]).norm() > +radius + EPS) {
71             out[outCnt] = pt[i];
72             ++outCnt;
73             minball(i, pt);
74             --outCnt;
75             if (i > 0) {
76                point Tt = pt[i];
77                memmove(&pt[1], &pt[0], sizeof(point) * i);
78                pt[0] = Tt;
79             }
80          }
81  }
82
83  pair<point, double> main(int npoint, point pt[]) { // 0-based
84    random_shuffle(pt, pt + npoint);
85    radius = -1;
86    for (int i = 0; i < npoint; i++) {
87       if ((res - pt[i]).norm() > EPS + radius) {
88          outCnt = 1;
89          out[0] = pt[i];
90          minball(i, pt);
91       }
92    }
93    return make_pair(res, sqrt(radius));
94  }
95  }
```

1.15 三维向量操作矩阵

- 绕单位向量 $u = (u_x, u_y, u_z)$ 右手方向旋转 θ 度的矩阵:

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

$$= \cos\theta I + \sin\theta \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} + (1 - \cos\theta) \begin{bmatrix} u_x^2 & u_xu_y & u_xu_z \\ u_yu_x & u_y^2 & u_yu_z \\ u_zu_x & u_zu_y & u_z^2 \end{bmatrix}$$

- 点 a 绕单位向量 $u = (u_x, u_y, u_z)$ 右手方向旋转 θ 度的对应点为 $a' = a\cos\theta + (u \times a)\sin\theta + (u \otimes u)a(1-\cos\theta)$

- 关于向量 v 作对称变换的矩阵 $H = I - 2\frac{vv^T}{v^Tv}$,

- 点 a 对称点: $a' = a - 2\frac{v^Ta}{v^Tv} \cdot v$

1.16 立体角

对于任意一个四面体 $OABC$, 从 O 点观察 ΔABC 的立体角 $\tan\frac{\Omega}{2} = \frac{\text{mix}(\vec{a}, \vec{b}, \vec{c})}{|a||b||c| + (\vec{a} \cdot \vec{b})|c| + (\vec{a} \cdot \vec{c})|b| + (\vec{b} \cdot \vec{c})|a|}$.

2 数据结构

2.1 动态凸包 (只支持插入)

```
1  #define x first    // upperHull ← (x, y)
2  #define y second   // lowerHull ← (x, -y)
3  typedef map<int, int> mii;
4  typedef map<int, int>::iterator mit;
5  struct point { point(const mit &p): x(p->first), y(p->second) {} };
6  inline bool checkInside(mii &a, const point &p) { // border inclusive
7     int x = p.x, y = p.y; mit p1 = a.lower_bound(x);
8     if (p1 == a.end()) return false; if (p1->x == x) return y <= p1->y;
9     if (p1 == a.begin()) return false; mit p2(p1--);
10    return sign(det(p - point(p1), point(p2) - p)) >= 0;
11 } inline void addPoint(mii &a, const point &p) { // no collinear points
12    int x = p.x, y = p.y; mit pnt = a.insert(make_pair(x, y)).first, p1, p2;
13    for (pnt->y = y; ; a.erase(p2)) {
14       p1 = pnt; if (++p1 == a.end()) break;
15       p2 = p1;  if (++p1 == a.end()) break;
16       if (det(point(p2) - p, point(p1) - p) < 0) break;
17    } for ( ; ; a.erase(p2)) {
18       if ((p1 = pnt) == a.begin()) break;
19       if (--p1 == a.begin()) break; p2 = p1--;
20       if (det(point(p2) - p, point(p1) - p) > 0) break;
21    }
22 }
```

2.2 Rope 用法

```
1  #include <ext/rope>
2  using __gnu_cxx::crope; using __gnu_cxx::rope;
3  a = b.substr(from, len);       // [from, from + len)
4  a = b.substr(from);            // [from, from)
5  b.c_str();                     // might lead to memory leaks
6  b.delete_c_str();              // delete the c_str that created before
7  a.insert(p, str);              // insert str before position p
8  a.erase(i, n);                 // erase [i, i + n)
```

2.3 可持久化 Treap

```
1  inline bool randomBySize(int a, int b) {
2     static long long seed = 1;
3     return (seed = seed * 48271 % 2147483647) * (a + b) < 2147483647LL * a;
4  }
5  tree merge(tree x, tree y) {
6     if (x == null) return y; if (y == null) return x;
7     tree t = NULL;
8     if (randomBySize(x->size, y->size)) t = newNode(x), t->r = merge(x->r, y);
9     else t = newNode(y), t->l = merge(x, y->l);
10    update(t); return t;
```

```
11  }
12  void splitByKey(tree t, int k, tree &l, tree &r) { // [-∞,k)[k,+infty)
13    if (t == null) l = r = null;
14    else if (t->key < k) l = newNode(t), splitByKey(t->r, k, l->r, r), update(l);
15    else                 r = newNode(t), splitByKey(t->l, k, l, r->l), update(r);
16  }
17  void splitBySize(tree t, int k, tree &l, tree &r) { // [1,k)[k,+∞)
18    static int s; if (t == null) l = r = null;
19    else if ((s = t->l->size + 1) < k) l = newNode(t), splitBySize(t->r, k - s, l->r, r), update(l);
20    else                               r = newNode(t), splitBySize(t->l, k, l, r->l), update(r);
21  }
```

2.4 左偏树

```
1   tree merge(tree a, tree b) {
2     if (a == null) return b;
3     if (b == null) return a;
4     if (a->key > b->key) swap(a, b);
5     a->rc = merge(a->rc, b);
6     a->rc->fa = a;
7     if (a->lc->dist < a->rc->dist) swap(a->lc, a->rc);
8     a->dist = a->rc->dist + 1;
9     return a;
10  }
11  void erase(tree t) {
12    tree x = t->fa, y = merge(t->lc, t->rc);
13    if (y != null) y->fa = x;
14    if (x == null) root = y;
15    else
16    for ((x->lc == t ? x->lc : x->rc) = y; x != null; y = x, x = x->fa) {
17      if (x->lc->dist < x->rc->dist) swap(x->lc, x->rc);
18      if (x->rc->dist + 1 == x->dist) return;
19      x->dist = x->rc->dist + 1;
20    }
21  }
```

2.5 Link-Cut Tree

```
1   struct node { int rev; node *pre, *ch[2]; } base[MAXN], nil, *null;
2   typedef node *tree;
3   #define isRoot(x) (x->pre->ch[0] != x && x->pre->ch[1] != x)
4   #define isRight(x) (x->pre->ch[1] == x)
5   inline void MakeRev(tree t) { if (t != null) { t->rev ^= 1; swap(t->ch[0], t->ch[1]); } }
6   inline void PushDown(tree t) { if (t->rev) { MakeRev(t->ch[0]); MakeRev(t->ch[1]); t->rev = 0; } }
7   inline void Rotate(tree x) {
8     tree y = x->pre; PushDown(y); PushDown(x);
9     int d = isRight(x);
10    if (!isRoot(y)) y->pre->ch[isRight(y)] = x; x->pre = y->pre;
11    if ((y->ch[d] = x->ch[!d]) != null) y->ch[d]->pre = y;
12    x->ch[!d] = y; y->pre = x; Update(y);
13  }
14  inline void Splay(tree x) {
15    PushDown(x); for (tree y; !isRoot(x); Rotate(x)) {
16      y = x->pre; if (!isRoot(y)) Rotate(isRight(x) != isRight(y) ? x : y);
17    } Update(x);
18  }
19  inline void Splay(tree x, tree to) {
20    PushDown(x); for (tree y; (y = x->pre) != to; Rotate(x)) if (y->pre != to)
21      Rotate(isRight(x) != isRight(y) ? x : y);
22    Update(x);
23  }
24  inline tree Access(tree t) {
25    tree last = null; for (; t != null; last = t, t = t->pre) Splay(t),t->ch[1] = last, Update(t);
26    return last;
27  }
28  inline void MakeRoot(tree t) { Access(t); Splay(t); MakeRev(t); }
29  inline tree FindRoot(tree t) { Access(t); Splay(t); tree last = null;
30    for ( ; t != null; last = t, t = t->ch[0]) PushDown(t); Splay(last); return last;
31  }
32  inline void Join(tree x, tree y) { MakeRoot(y); y->pre = x; }
33  inline void Cut(tree t) {Access(t); Splay(t); t->ch[0]->pre = null; t->ch[0] = null; Update(t);}
34  inline void Cut(tree x, tree y) {
35    tree upper = (Access(x), Access(y));
36    if (upper == x) { Splay(x); y->pre = null; x->ch[1] = null; Update(x); }
37    else if (upper == y) { Access(x); Splay(y); x->pre = null; y->ch[1] = null; Update(y); }
38    else assert(0); // impossible to happen
39  }
40  inline int Query(tree a, tree b) { // query the cost in path a <-> b, lca inclusive
```

```
41    Access(a); tree c = Access(b); // c is lca
42    int v1 = c->ch[1]->maxCost; Access(a);
43    int v2 = c->ch[1]->maxCost;
44    return max(max(v1, v2), c->cost);
45  }
46  void Init() {
47    null = &nil; null->ch[0] = null->ch[1] = null->pre = null; null->rev = 0;
48    Rep(i, 1, N) { node &n = base[i]; n.rev = 0; n.pre = n.ch[0] = n.ch[1] = null; }
49  }
```

2.6 K-D Tree Nearest

```
1   struct Point { int x, y; };
2   struct Rectangle {
3     int lx , rx , ly , ry;
4     void set(const Point &p) { lx = rx = p.x; ly = ry = p.y; }
5     void merge(const Point &o) {
6       lx = min(lx, o.x); rx = max(rx, o.x); ly = min(ly, o.y); ry = max(ry, o.y);
7     } void merge(const Rectangle &o) {
8       lx = min(lx , o.lx); rx = max(rx , o.rx); ly = min(ly , o.ly); ry = max(ry , o.ry);
9     } LL dist(const Point &p) {
10      LL res = 0;
11      if (p.x < lx) res += sqr(lx - p.x); else if (p.x > rx) res += sqr(p.x - rx);
12      if (p.y < ly) res += sqr(ly - p.y); else if (p.y > ry) res += sqr(p.y - ry);
13      return res;
14    }
15  };
16  struct Node { int child[2]; Point p; Rectangle rect; };
17  const int MAX_N = 1111111;
18  const LL INF = 100000000;
19  int n, m, tot, root; LL result;
20  Point a[MAX_N], p; Node tree[MAX_N];
21  int build(int s, int t, bool d) {
22    int k = ++tot, mid = (s + t) >> 1;
23    nth_element(a + s, a + mid , a + t, d ? cmpXY : cmpYX);
24    tree[k].p = a[mid]; tree[k].rect.set(a[mid]); tree[k].child[0] = tree[k].child[1] = 0;
25    if (s < mid)
26      tree[k].child[0] = build(s, mid , d ^ 1), tree[k].rect.merge(tree[tree[k].child[0]].rect);
27    if (mid + 1 < t)
28      tree[k].child[1] = build(mid + 1, t, d ^ 1), tree[k].rect.merge(tree[tree[k].child[1]].rect);
29    return k;
30  }
31  int insert(int root, bool d) {
32    if (root == 0) {
33      tree[++tot].p = p; tree[tot].rect.set(p); tree[tot].child[0] = tree[tot].child[1] = 0;
34      return tot;
35    } tree[root].rect.merge(p);
36    if ((d && cmpXY(p, tree[root].p)) || (!d && cmpYX(p, tree[root].p)))
37      tree[root].child[0] = insert(tree[root].child[0], d ^ 1);
38    else tree[root].child[1] = insert(tree[root].child[1], d ^ 1);
39    return root;
40  }
41  void query(int k, bool d) {
42    if (tree[k].rect.dist(p) >= result) return;
43    cMin(result, dist(tree[k].p, p));
44    if ((d && cmpXY(p, tree[k].p)) || (!d && cmpYX(p, tree[k].p))) {
45      if (tree[k].child[0]) query(tree[k].child[0], d ^ 1);
46      if (tree[k].child[1]) query(tree[k].child[1], d ^ 1);
47    } else {
48      if (tree[k].child[1]) query(tree[k].child[1], d ^ 1);
49      if (tree[k].child[0]) query(tree[k].child[0], d ^ 1);
50    }
51  }
52  void example(int n) {
53    root = tot = 0; scan(a); root = build(0, n, 0); // init, a[0...n-1]
54    scan(p); root = insert(root, 0); // insert
55    scan(p); result = INF; ans = query(root, 0); // query
56  }
```

2.7 K-D Tree Farthest

输入 n 个点, 对每个询问 px, py, k, 输出 k 远点的编号

```
1   struct Point { int x, y, id; };
2   struct Rectangle {
3     int lx, rx, ly, ry;
4     void set(const Point &p) { lx = rx = p.x; ly = ry = p.y; }
5     void merge(const Rectangle &o) {
6       lx = min(lx, o.lx); rx = max(rx, o.rx); ly = min(ly, o.ly); ry = max(ry, o.ry);
```

```
7      }
8      LL dist(const Point &p) { LL res = 0;
9        res += max(sqr(rx - p.x), sqr(lx - p.x));
10       res += max(sqr(ry - p.y), sqr(ly - p.y));
11       return res;
12     }
13   }; struct Node { Point p; Rectangle rect; };
14   const int MAX_N = 111111;
15   const LL INF = 1LL << 60;
16   int n, m;
17   Point a[MAX_N], b[MAX_N ];
18   Node tree[MAX_N * 3];
19   Point p; // p is the query point
20   pair<LL, int> result[22];
21   void build(int k, int s, int t, bool d) {
22     int mid = (s + t) >> 1;
23     nth_element(a + s, a + mid , a + t, d ? cmpX : cmpY);
24     tree[k].p = a[mid];
25     tree[k].rect.set(a[mid]);
26     if (s < mid)
27       build(k << 1, s, mid , d ^ 1), tree[k].rect.merge(tree[k << 1]. rect);
28     if (mid + 1 < t)
29       build(k << 1 | 1, mid + 1, t, d ^ 1), tree[k].rect.merge(tree[k << 1 | 1]. rect);
30   }
31   void query(int k, int s, int t, bool d, int kth) {
32     if (tree[k].rect.dist(p) < result[kth].first) return;
33     pair<LL, int> tmp(dist(tree[k].p, p), -tree[k].p.id);
34     for (int i = 1; i <= kth; i++) if (tmp > result[i]) {
35       for (int j = kth + 1; j > i; j--) result[j] = result[j - 1]; result[i] = tmp;
36       break;
37     }
38     int mid = (s + t) >> 1;
39     if ((d && cmpX(p, tree[k].p)) || (!d && cmpY(p, tree[k].p))) {
40       if (mid + 1 < t) query(k << 1 | 1, mid + 1, t, d ^ 1, kth);
41       if (s < mid)     query(k << 1, s, mid , d ^ 1, kth);
42     } else {
43       if (s < mid)     query(k << 1, s, mid , d ^ 1, kth);
44       if (mid + 1 < t) query(k << 1 | 1, mid + 1, t, d ^ 1, kth);
45     }
46   }
47   void example(int n) {
48     scan(a); build(1, 0, n, 0); // init, a[0...n-1]
49     scan(p, k);   // query
50     Rep(j, 1, k) result[j].first = -1;
51     query(1, 0, n, 0, k); ans = -result[k].second + 1;
52   }
```

2.8 树链剖分

```
1   int N, fa[MAXN], dep[MAXN], que[MAXN], size[MAXN], own[MAXN];
2   int LCA(int x, int y) { if (x == y) return x;
3     for ( ; ; x = fa[own[x]]) {
4       if (dep[x] < dep[y]) swap(x, y); if (own[x] == own[y]) return y;
5       if (dep[own[x]] < dep[own[y]]) swap(x, y);
6     } return -1;
7   }
8   void Decomposion() {
9     static int path[MAXN]; int x, y, a, next, head = 0, tail = 0, cnt; // BFS omitted
10    for (int i = 1; i <= N; ++i) if (own[a = que[i]] == -1)
11      for (x = a, cnt = 0; ; x = next) { next = -1; own[x] = a; path[++cnt] = x;
12        for (edge e(fir[x]); e; e = e->next) if ( (y = e->to) != fa[x] )
13          if (next == -1 || size[y] > size[next]) next = y;
14        if (next == -1) { tree[a].init(cnt, path); break; }
15      }
16  }
```

3 字符串相关

3.1 Manacher

```
1   // len[i] : the max length of palindrome whose mid point is (i / 2)
2   void Manacher(int n, char cs[], int len[]) { // 0-based, len[] must be double sized
3     for (int i = 0; i < n + n; ++i) len[i] = 0;
4     for (int i = 0, j = 0, k; i < n * 2; i += k, j = max(j - k, 0)) {
5       while (i - j >= 0 && i + j + 1 < n * 2 && cs[(i - j) / 2] == cs[(i + j + 1) / 2]) j++;
6       len[i] = j; for (k = 1; i - k >= 0 && j - k >= 0 && len[i - k] != j - k; k++)
```

```
7       len[i + k] = min(len[i - k], j - k);
8     }
9   }
```

3.2 KMP

$$next[i] = \max\{len|A[0\ldots len-1] = A\text{的第 i 位向前或后的长度为 } len \text{ 的串}\}$$
$$ext[i] = \max\{len|A[0\ldots len-1] = B\text{的第 i 位向前或后的长度为 } len \text{ 的串}\}$$

```
1   void KMP(char *a, int la, char *b, int lb, int *next, int *ext) {
2     --a; --b; --next; --ext;
3     for (int i = 2, j = next[1] = 0; i <= la; i++) {
4       while (j && a[j + 1] != a[i]) j = next[j]; if (a[j + 1] == a[i]) ++j; next[i] = j;
5     } for (int i = 1, j = 0; i <= lb; ++i) {
6       while (j && a[j + 1] != b[i]) j = next[j]; if (a[j + 1] == b[i]) ++j; ext[i] = j;
7       if (j == la) j = next[j];
8     }
9   } void ExKMP(char *a, int la, char *b, int lb, int *next, int *ext) {
10    next[0] = la; for (int &j = next[1] = 0; j + 1 < la && a[j] == a[j + 1]; ++j);
11    for (int i = 2, k = 1; i < la; ++i) {
12      int p = k + next[k], l = next[i - k]; if (l < p - i) next[i] = l;
13      else for (int &j = next[k = i] = max(0, p - i); i + j < la && a[j] == a[i + j]; ++j);
14    } for (int &j = ext[0] = 0; j < la && j < lb && a[j] == b[j]; ++j);
15    for (int i = 1, k = 0; i < lb; ++i) {
16      int p = k + ext[k], l = next[i - k]; if (l < p - i) ext[i] = l;
17      else for (int &j = ext[k = i] = max(0, p - i); j < la && i + j < lb && a[j] == b[i + j]; ++j);
18    }
19  }
```

3.3 后缀自动机

```
1   struct node { int len; node *fa, *go[26]; } base[MAXNODE], *top = base, *root, *que[MAXNODE];
2   typedef node *tree;
3   inline tree newNode(int len) {
4     top->len = len; top->fa = NULL; memset(top->go, 0, sizeof(top->go)); return top++;
5   } inline tree newNode(int len, tree fa, tree *go) {
6     top->len = len; top->fa = fa; memcpy(top->go, go, sizeof(top->go)); return top++;
7   } void construct(char *A, int N) {
8     tree p = root = newNode(0), q, up, fa;
9     for (int i = 0; i < N; ++i) {
10      int w = A[i] - 'a'; up = p; p = newNode(i + 1);
11      for ( ; up && !up->go[w]; up = up->fa) up->go[w] = p;
12      if (!up) p->fa = root;
13      else { q = up->go[w];
14        if (up->len + 1 == q->len) p->fa = q;
15        else { fa = newNode(up->len + 1, q->fa, q->go);
16          for (p->fa = q->fa = fa; up && up->go[w] == q; up = up->fa) up->go[w] = fa;
17        }
18      }
19    } static int cnt[MAXLEN]; memset(cnt, 0, sizeof(int) * (N + 1));
20    for (tree i(base); i != top; ++i) ++cnt[i->len];
21    Rep(i, 1, N) cnt[i] += cnt[i - 1];
22    for (tree i(base); i != top; ++i) Q[ cnt[i->len]-- ] = i;
23  }
```

3.4 后缀数组

待排序的字符串放在 $r[0\ldots n-1]$ 中, 最大值小于 m.
$r[0\ldots n-2] > 0, r[n-1] = 0$.
结果放在 $sa[0\ldots n-1]$.

```
1   namespace SuffixArrayDoubling {
2     int wa[MAXN], wb[MAXN], wv[MAXN], ws[MAXN];
3     int cmp(int *r, int a, int b, int l) { return r[a] == r[b] && r[a + l] == r[b + l]; }
4     void da(int *r, int *sa, int n, int m) {
5       int i, j, p, *x = wa, *y = wb, *t;
6       for (i = 0; i < m; i++) ws[i] = 0;
7       for (i = 0; i < n; i++) ws[x[i] = r[i]]++;
8       for (i = 1; i < m; i++) ws[i] += ws[i - 1];
9       for (i = n - 1; i >= 0; i--) sa[--ws[x[i]]] = i;
10      for (j = 1, p = 1; p < n; j *= 2, m = p) {
11        for (p = 0, i = n - j; i < n; i++) y[p++] = i;
```

```
12        for (i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] - j;
13        for (i = 0; i < n; i++) wv[i] = x[y[i]];
14        for (i = 0; i < m; i++) ws[i] = 0;
15        for (i = 0; i < n; i++) ws[wv[i]]++;
16        for (i = 1; i < m; i++) ws[i] += ws[i - 1];
17        for (i = n - 1; i >= 0; i--) sa[--ws[wv[i]]] = y[i];
18        for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++)
19            x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
20 }}}
21 namespace CalcHeight {
22     int rank[MAXN], height[MAXN];
23     void calheight(int *r, int *sa, int n) {
24         int i, j, k = 0; for (i = 1; i <= n; i++) rank[sa[i]] = i;
25         for (i = 0; i < n; height[rank[i++]] = k)
26             for (k ? k-- : 0, j = sa[rank[i] - 1]; r[i + k] == r[j + k]; k++);
27 }}
```

3.5 环串最小表示

```
1 int minimalRepresentation(int N, char *s) { // s must be double-sized and 0-based
2     int i, j, k, l; for (i = 0; i < N; ++i) s[i + N] = s[i]; s[N + N] = 0;
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; if (s[i + k] < s[j + k]) j += k + 1;
6         else l = i + k, i = j, j = max(l, j) + 1;
7     } return i; // [i, i + N) is the minimal representation
8 }
```

4 图论

4.1 带花树

```
1 namespace Blossom {
2     int n, head, tail, S, T, lca;
3     int match[MAXN], Q[MAXN], pred[MAXN], label[MAXN], inq[MAXN], inb[MAXN];
4     vector<int> link[MAXN];
5     inline void push(int x) { Q[tail++] = x; inq[x] = true; }
6     int findCommonAncestor(int x, int y) {
7         static bool inPath[MAXN]; for (int i = 0; i < n; ++i) inPath[i] = 0;
8         for ( ; ; x = pred[ match[x] ]) { x = label[x]; inPath[x] = true; if (x == S) break; }
9         for ( ; ; y = pred[ match[y] ]) { y = label[y]; if (inPath[y]) break; } return y;
10    }
11    void resetTrace(int x, int lca) {
12        while (label[x] != lca) { int y = match[x]; inb[ label[x] ] = inb[ label[y] ] = true;
13        x = pred[y]; if (label[x] != lca) pred[x] = y; }}
14    void blossomContract(int x, int y) {
15        lca = findCommonAncestor(x, y);
16        Foru(i, 0, n) inb[i] = 0; resetTrace(x, lca); resetTrace(y, lca);
17        if (label[x] != lca) pred[x] = y; if (label[y] != lca) pred[y] = x;
18        Foru(i, 0, n) if (inb[ label[i] ]) { label[i] = lca; if (!inq[i]) push(i); }
19    }
20    bool findAugmentingPath() {
21        Foru(i, 0, n) pred[i] = -1, label[i] = i, inq[i] = 0;
22        int x, y, z; head = tail = 0;
23        for (push(S); head < tail; ) for (int i = (int)link[x = Q[head++]].size() - 1; i >= 0; --i) {
24            y = link[x][i]; if (label[x] == label[y] || x == match[y]) continue;
25            if (y == S || (match[y] >= 0 && pred[ match[y] ] >= 0)) blossomContract(x, y);
26            else if (pred[y] == -1) {
27                pred[y] = x; if (match[y] > 0) push(match[y]);
28                else {
29                    for (x = y; x >= 0; x = z) {
30                        y = pred[x], z = match[y]; match[x] = y, match[y] = x;
31                    } return true; }}} return false;
32    }
33    int findMaxMatching() {
34        int ans = 0; Foru(i, 0, n) match[i] = -1;
35        for (S = 0; S < n; ++S) if (match[S] == -1) if (findAugmentingPath()) ++ans;
36        return ans;
37    }
38 }
```

4.2 最大流

```
1 namespace Maxflow {
2     int h[MAXNODE], vh[MAXNODE], S, T, Ncnt; edge cur[MAXNODE], pe[MAXNODE];
3     void init(int _S, int _T, int _Ncnt) { S = _S; T = _T; Ncnt = _Ncnt; }
4     int maxflow() {
5         static int Q[MAXNODE]; int x, y, augc, flow = 0, head = 0, tail = 0; edge e;
6         Rep(i, 0, Ncnt) cur[i] = fir[i]; Rep(i, 0, Ncnt) h[i] = INF; Rep(i, 0, Ncnt) vh[i] = 0;
7         for (Q[++tail] = T, h[T] = 0; head < tail; ) {
8             x = Q[++head]; ++vh[ h[x] ];
9             for (e = fir[x]; e; e = e->next) if (e->op->c)
10                if (h[y = e->to] >= INF) h[y] = h[x] + 1, Q[++tail] = y;
11        } for (x = S; h[S] < Ncnt; ) {
12            for (e = cur[x]; e; e = e->next) if (e->c)
13                if (h[y = e->to] + 1 == h[x]) { cur[x] = pe[y] = e; x = y; break; }
14            if (!e) {
15                if (--vh[ h[x] ] == 0) break; h[x] = Ncnt; cur[x] = NULL;
16                for (e = fir[x]; e; e = e->next) if (e->c)
17                    if ( cMin( h[x], h[e->to] + 1 ) ) cur[x] = e;
18                ++vh[ h[x] ];
19                if (x != S) x = pe[x]->op->to;
20            } else if (x == T) { augc = INF;
21                for (x = T; x != S; x = pe[x]->op->to) cMin(augc, pe[x]->c);
22                for (x = T; x != S; x = pe[x]->op->to) {
23                    pe[x]->c -= augc; pe[x]->op->c += augc;
24                } flow += augc;
25            }
26        } return flow;
27    }
28 }
```

4.3 KM

```
1 int N, Tcnt, w[MAXN][MAXN], slack[MAXN];
2 int lx[MAXN], linkx[MAXN], visy[MAXN], ly[MAXN], linky[MAXN], visx[MAXN]; // 初值全为 0
3 bool DFS(int x) { visx[x] = Tcnt;
4     Rep(y, 1, N) if(visy[y] != Tcnt) { int t = lx[x] + ly[y] - w[x][y];
5         if (t == 0) { visy[y] = Tcnt;
6             if (!linky[y] || DFS(linky[y])) { linkx[x] = y; linky[y] = x; return true; }
7         } else cMin(slack[y], t);
8     } return false;
9 } void KM() {
10    Tcnt = 0; Rep(x, 1, N) Rep(y, 1, N) cMax(lx[x], w[x][y]);
11    Rep(S, 1, N) { Rep(i, 1, N) slack[i] = INF;
12        for (++Tcnt; !DFS(S); ++Tcnt) { int d = INF;
13            Rep(y, 1, N) if(visy[y] != Tcnt) cMin(d, slack[y]);
14            Rep(x, 1, N) if(visx[x] == Tcnt) lx[x] -= d;
15            Rep(y, 1, N) if(visy[y] == Tcnt) ly[y] += d; else slack[y] -= d;
16        }
17    }
18 }
```

4.4 2-SAT 与 Kosaraju

注意 Kosaraju 需要建反图

```
1 namespace SCC {
2     int code[MAXN * 2], seq[MAXN * 2], sCnt;
3     void DFS_1(int x) { code[x] = 1;
4         for (edge e(fir[x]); e; e = e->next) if (code[e->to] == -1) DFS_1(e->to);
5         seq[++sCnt] = x;
6     } void DFS_2(int x) { code[x] = sCnt;
7         for (edge e(fir2[x]); e; e = e->next) if (code[e->to] == -1) DFS_2(e->to); }
8     void SCC(int N) {
9         sCnt = 0; for (int i = 1; i <= N; ++i) code[i] = -1;
10        for (int i = 1; i <= N; ++i) if (code[i] == -1) DFS_1(i);
11        sCnt = 0; for (int i = 1; i <= N; ++i) code[i] = -1;
12        for (int i = N; i >= 1; --i) if (code[seq[i]] == -1) {
13            ++sCnt; DFS_2(seq[i]); }
14    }
15 }// true     - 2i - 1
16 //  false    - 2i
17 bool TwoSat() { SCC::SCC(N + N);
18     // if code[2i - 1] = code[2i]: no solution
19     // if code[2i - 1] > code[2i]: i selected. else i not selected
20 }
```

4.5 全局最小割 Stoer-Wagner

```
int minCut(int N, int G[MAXN][MAXN]) { // 0-based
  static int weight[MAXN], used[MAXN]; int ans = INT_MAX;
  while (N > 1) {
    for (int i = 0; i < N; ++i) used[i] = false; used[0] = true;
    for (int i = 0; i < N; ++i) weight[i] = G[i][0];
    int S = -1, T = 0;
    for (int _r = 2; _r <= N; ++_r) { // N - 1 selections
      int x = -1;
      for (int i = 0; i < N; ++i) if (!used[i])
        if (x == -1 || weight[i] > weight[x]) x = i;
      for (int i = 0; i < N; ++i) weight[i] += G[x][i];
      S = T; T = x; used[x] = true;
    } ans = min(ans, weight[T]);
    for (int i = 0; i < N; ++i) G[i][S] += G[i][T], G[S][i] += G[i][T];
    G[S][S] = 0; --N;
    for (int i = 0; i <= N; ++i) swap(G[i][T], G[i][N]);
    for (int i = 0; i < N; ++i) swap(G[T][i], G[N][i]);
  } return ans;
}
```

4.6 欧拉路

```
vector<int> eulerianWalk(int N, int S) {
  static int res[MAXM], stack[MAXN]; static edge cur[MAXN];
  int rcnt = 0, top = 0, x; for (int i = 1; i <= N; ++i) cur[i] = fir[i];
  for (stack[top++] = S; top; ) {
    for (x = stack[--top]; ; ) {
      edge &e = cur[x]; if (e == NULL) break;
      stack[top++] = x; x = e->to; e = e->next;
      // 对于无向图需要删掉反向边
    } res[rcnt++] = x;
  } reverse(res, res + rcnt); return vector<int>(res, res + rcnt);
}
```

4.7 最大团搜索

```
namespace MaxClique { // 1-based
  int g[MAXN][MAXN], len[MAXN], list[MAXN][MAXN], mc[MAXN], ans, found;
  void DFS(int size) {
    if (len[size] == 0) { if (size > ans) ans = size, found = true; return; }
    for (int k = 0; k < len[size] && !found; ++k) {
      if (size + len[size] - k <= ans) break;
      int i = list[size][k]; if (size + mc[i] <= ans) break;
      for (int j = k + 1, len[size + 1] = 0; j < len[size]; ++j) if (g[i][list[size][j]])
        list[size + 1][len[size + 1]++] = list[size][j];
      DFS(size + 1);
    }
  }
  int work(int n) {
    mc[n] = ans = 1; for (int i = n - 1; i; --i) { found = false; len[1] = 0;
    for (int j = i + 1; j <= n; ++j) if (g[i][j]) list[1][len[1]++] = j;
    DFS(1); mc[i] = ans;
    } return ans;
  }
}
```

4.8 最小树形图

```
namespace EdmondsAlgorithm { // O(ElogE + V^2) !!! 0-based !!!
  struct enode { int from, c, key, delta, dep; enode *ch[2], *next;
  } ebase[maxn], *etop, *fir[maxn], nil, *null, *inEdge[maxn], *chs[maxn];
  typedef enode *edge; typedef enode *tree;
  int n, m, setFa[maxn], deg[maxn], que[maxn];
  inline void pushDown(tree x) { if (x->delta) {
    x->ch[0]->key += x->delta; x->ch[0]->delta += x->delta;
    x->ch[1]->key += x->delta; x->ch[1]->delta += x->delta; x->delta = 0;
  }}
  tree merge(tree x, tree y) {
    if (x == null) return y; if (y == null) return x;
```

```
    if (x->key > y->key) swap(x, y); pushDown(x); x->ch[1] = merge(x->ch[1], y);
    if (x->ch[0]->dep < x->ch[1]->dep) swap(x->ch[0], x->ch[1]);
    x->dep = x->ch[1]->dep + 1; return x;
  }
  void addEdge(int u, int v, int w) {
    etop->from = u; etop->c = etop->key = w; etop->delta = etop->dep = 0;
    etop->next = fir[v]; etop->ch[0] = etop->ch[1] = null;
    fir[v] = etop; inEdge[v] = merge(inEdge[v], etop++);
  }
  void deleteMin(tree &r) { pushDown(r); r = merge(r->ch[0], r->ch[1]); }
  int findSet(int x) { return setFa[x] == x ? x : setFa[x] = findSet(setFa[x]); }
  void clear(int V, int E) {
    null = &nil; null->ch[0] = null->ch[1] = null; null->dep = -1;
    n = V; m = E; etop = ebase; Foru(i, 0, V) fir[i] = NULL; Foru(i, 0, V) inEdge[i] = null;
  }
  int solve(int root) { int res = 0, head, tail;
    for (int i = 0; i < n; ++i) setFa[i] = i;
    for ( ; ; ) { memset(deg, 0, sizeof(int) * n); chs[root] = inEdge[root];
      for (int i = 0; i < n; ++i) if (i != root && setFa[i] == i) {
        while (findSet(inEdge[i]->from) == findSet(i)) deleteMin(inEdge[i]);
        ++deg[ findSet((chs[i] = inEdge[i])->from) ];
      }
      for (int i = head = tail = 0; i < n; ++i)
        if (i != root && setFa[i] == i && deg[i] == 0) que[tail++] = i;
      while (head < tail) {
        int x = findSet(chs[que[head++]]->from);
        if (--deg[x] == 0) que[tail++] = x;
      } bool found = false;
      for (int i = 0; i < n; ++i) if (i != root && setFa[i] == i && deg[i] > 0) {
        int j = i; tree temp = null; found = true;
        do {setFa[j = findSet(chs[j]->from)] = i;
          deleteMin(inEdge[j]); res += chs[j]->key;
          inEdge[j]->key -= chs[j]->key; inEdge[j]->delta -= chs[j]->key;
          temp = merge(temp, inEdge[j]);
        } while (j != i); inEdge[i] = temp;
      } if (!found) break;
    } for (int i = 0; i < n; ++ i) if (i != root && setFa[i] == i) res += chs[i]->key;
    return res;
  }
}
namespace ChuLiu { // O(V ^ 3) !!! 1-based !!!
  int n, used[maxn], pass[maxn], eg[maxn], more, que[maxn], g[maxn][maxn];
  void combine(int id, int &sum) { int tot = 0, from, i, j, k;
    for ( ; id != 0 && !pass[id]; id = eg[id]) que[tot++] = id, pass[id] = 1;
    for (from = 0; from < tot && que[from] != id; from++);
    if (from == tot) return; more = 1;
    for (i = from; i < tot; i++) {
      sum += g[eg[que[i]]][que[i]]; if (i == from) continue;
      for (j = used[que[i]] = 1; j <= n; j++) if (!used[j])
        if (g[que[i]][j] < g[id][j]) g[id][j] = g[que[i]][j];
    }
    for (i = 1; i <= n; i++) if (!used[i] && i != id)
      for (j = from; j < tot; j++) {
        k = que[j]; if (g[i][id] > g[i][k] - g[eg[k]][k])
        g[i][id] = g[i][k] - g[eg[k]][k];
      }
  }
  void clear(int V) { n = V; Rep(i, 1, V) Rep(j, 1, V) g[i][j] = inf; }
  int solve(int root) {
    int i, j, k, sum = 0; memset(used, 0, sizeof(int) * (n + 1));
    for (more = 1; more; ) {
      more = 0; memset(eg, 0, sizeof(int) * (n + 1));
      for (i = 1; i <= n; i++) if (!used[i] && i != root) {
        for (j = 1, k = 0; j <= n; j++) if (!used[j] && i != j)
          if (k == 0 || g[j][i] < g[k][i]) k = j;
        eg[i] = k;
      } memset(pass, 0, sizeof(int) * (n + 1));
      for (i = 1; i <= n; i++) if (!used[i] && !pass[i] && i != root)
        combine(i, sum);
    } for (i = 1; i <= n; i++) if (!used[i] && i != root) sum += g[eg[i]][i];
    return sum;
  }
}
```

4.9 离线动态最小生成树

$O(Qlog^2Q)$. $(qx[i], qy[i])$ 表示将编号为 $qx[i]$ 的边的权值改为 $qy[i]$, 删除一条边相当于将其权值改为 ∞, 加入一条边相当于将其权值从 ∞ 变成某个值.

```
const int maxn = 100000 + 5;
const int maxm = 1000000 + 5;
const int maxq = 1000000 + 5;
```

```
4   const int qsize = maxm + 3 * maxq;
5   int n, m, Q, x[qsize], y[qsize], z[qsize], qx[maxq], qy[maxq], a[maxn], *tz;
6   int kx[maxn], ky[maxn], kt, vd[maxn], id[maxm], app[maxm];
7   bool extra[maxm];
8   void init() {
9     scanf("%d%d", &n, &m); for (int i = 0; i < m; i++) scanf("%d%d%d", x + i, y + i, z + i);
10    scanf("%d", &Q); for (int i = 0; i < Q; i++) { scanf("%d%d", qx + i, qy + i); qx[i]--; }
11  }
12  int find(int x) {
13    int root = x, next; while (a[root]) root = a[root];
14    while ((next = a[x]) != 0) a[x] = root, x = next; return root;
15  }
16  inline bool cmp(const int &a, const int &b) { return tz[a] < tz[b]; }
17  void solve(int *qx, int *qy, int Q, int n, int *x, int *y, int *z, int m, long long ans) {
18    int ri, rj;
19    if (Q == 1) {
20      for (int i = 1; i <= n; i++) a[i] = 0; z[qx[0]] = qy[0];
21      for (int i = 0; i < m; i++) id[i] = i;
22      tz = z; sort(id, id + m, cmp);
23      for (int i = 0; i < m; i++) {
24        ri = find(x[id[i]]); rj = find(y[id[i]]);
25        if (ri != rj) ans += z[id[i]], a[ri] = rj;
26      } printf("%I64d\n", ans);
27      return;
28    } int tm = kt = 0, n2 = 0, m2 = 0;
29    for (int i = 1; i <= n; i++) a[i] = 0;
30    for (int i = 0; i < Q; i++) {
31      ri = find(x[qx[i]]); rj = find(y[qx[i]]); if (ri != rj) a[ri] = rj;
32    }
33    for (int i = 0; i < m; i++) extra[i] = true;
34    for (int i = 0; i < Q; i++) extra[qx[i]] = false;
35    for (int i = 0; i < m; i++) if (extra[i]) id[tm++] = i;
36    tz = z; sort(id, id + tm, cmp);
37    for (int i = 0; i < tm; i++) {
38      ri = find(x[id[i]]); rj = find(y[id[i]]);
39      if (ri != rj)
40        a[ri] = rj, ans += z[id[i]], kx[kt] = x[id[i]], ky[kt] = y[id[i]], kt++;
41    }
42    for (int i = 1; i <= n; i++) a[i] = 0;
43    for (int i = 0; i < kt; i++) a[find(kx[i])] = find(ky[i]);
44    for (int i = 1; i <= n; i++) if (a[i] == 0) vd[i] = ++n2;
45    for (int i = 1; i <= n; i++) if (a[i] != 0) vd[i] = vd[find(i)];
46    int *Nx = x + m, *Ny = y + m, *Nz = z + m;
47    for (int i = 0; i < m; i++) app[i] = -1;
48    for (int i = 0; i < Q; i++)
49      if (app[qx[i]] == -1)
50        Nx[m2] = vd[x[qx[i]]], Ny[m2] = vd[y[qx[i]]], Nz[m2] = z[qx[i]], app[qx[i]] = m2, m2++;
51    for (int i = 0; i < Q; i++) {
52      z[qx[i]] = qy[i];
53      qx[i] = app[qx[i]];
54    }
55    for (int i = 1; i <= n2; i++) a[i] = 0;
56    for (int i = 0; i < tm; i++) {
57      ri = find(vd[x[id[i]]]); rj = find(vd[y[id[i]]]);
58      if (ri != rj)
59        a[ri] = rj, Nx[m2] = vd[x[id[i]]], Ny[m2] = vd[y[id[i]]], Nz[m2] = z[id[i]], m2++;
60    }
61    int mid = Q / 2;
62    solve(qx, qy, mid, n2, Nx, Ny, Nz, m2, ans);
63    solve(qx + mid, qy + mid, Q - mid, n2, Nx, Ny, Nz, m2, ans);
64  }
65  void work() { if (Q) solve(qx, qy, Q, n, x, y, z, m, 0); }
66  int main() { init(); work(); return 0; }
```

4.10 弦图

- 团数 ≤ 色数

- 最大独立集数 ≤ 最小团覆盖数

- 任何一个弦图都至少有一个单纯点, 不是完全图的弦图至少有两个不相邻的单纯点.

- 设第 i 个点在弦图的完美消除序列第 $p(i)$ 个. 令 $N(v) = \{w | w 与 v 相邻且 p(w) > p(v)\}$ 弦图的极大团一定是 $v \cup N(v)$ 的形式.

- 弦图最多有 n 个极大团.

- 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, 满足 $Next(w) = v$ 且 $|N(v)| + 1 \leq |N(w)|$ 即可.

- 最小染色: 完美消除序列从后往前依次给每个点染色, 给每个点染上可以染的最小的颜色. (团数 = 色数)

- 最大独立集: 完美消除序列从前往后能选就选.

- 最小团覆盖: 设最大独立集为 $\{p_1, p_2, \ldots, p_t\}$, 则 $\{p_1 \cup N(p_1), \ldots, p_t \cup N(p_t)\}$ 为最小团覆盖. (最大独立集数 = 最小团覆盖数)

```
1   class Chordal { // 1-Based, G is the Graph, must be sorted before call Check_Chordal
2   public: // Construct will sort it automatically
3     int v[Maxn], id[Maxn];  bool inseq[Maxn]; priority_queue<pair<int, int> > pq;
4     vector<int> Construct_Perfect_Elimination_Sequence(vector<int> *G, int n) { // O(m + nlogn)
5       vector<int> seq(n + 1, 0);
6       for (int i = 0; i <= n; ++i) inseq[i] = false, sort(G[i].begin(), G[i].end()), v[i] = 0;
7       int cur = n; pair<int, int> Mx; while(!pq.empty()) pq.pop(); pq.push(make_pair(0, 1));
8       for (int i = n; i >= 1; --i) {
9         while (!pq.empty() && (Mx = pq.top(), inseq[Mx.second] || Mx.first != v[Mx.second])) pq.pop();
10        id[Mx.second] = cur;
11        int x = seq[cur--] = Mx.second, sz = (int)G[Mx.second].size(); inseq[x] = true;
12        for (int j = 0; j < sz; ++j) {
13          int y = G[x][j]; if(!inseq[y]) pq.push(make_pair(++v[y], y));
14        }
15      } return seq;
16    }
17    bool Check_Chordal(vector<int> *G, vector<int> &seq, int n) { // O(n + mlogn), plz gen seq first
18      bool isChordal = true;
19      for (int i = n - 1; i >= 1 && isChordal; --i) {
20        int x = seq[i], sz, y = -1;
21        if ((sz = (int)G[x].size()) == 0) continue;
22          for(int j = 0; j < sz; ++j) {
23          if (id[G[x][j]] < i) continue;
24          if (y == -1 || id[y] > id[G[x][j]]) y = G[x][j];
25        } if (y == -1) continue;
26        for (int j = 0; j < sz; ++j) {
27          int y1 = G[x][j]; if (id[y1] < i) continue;
28          if (y1 == y || binary_search(G[y].begin(), G[y].end(), y1)) continue;
29          isChordal = false; break;
30        }
31      } return isChordal;
32    }
33  };
```

4.11 小知识

- 平面图: 一定存在一个度小于等于 5 的点. $E \leq 3V - 6$. 欧拉公式: $V + F - E = 1 + 连通块数$

- 图连通度:

  1. $k-$ 连通 (k-connected): 对于任意一对结点都至少存在结点各不相同的 k 条路

  2. 点连通度 (vertex connectivity): 把图变成非连通图所需删除的最少点数

  3. Whitney 定理: 一个图是 $k-$ 连通的当且仅当它的点连通度至少为 k

- Lindstroem-Gessel-Viennot Lemma: 给定一个图的 n 个起点和 n 个终点, 令 $A_{ij} = $ 第 i 个起点到第 j 个终点的路径条数, 则从起点到终点的不相交路径条数为 $det(A)$

- 欧拉回路与树形图的联系: 对于出度等于入度的连通图 $s(G) = t_i(G) \prod_{j=1}^{n} (d^+(v_j) - 1)!$

- 密度子图: 给定无向图, 选取点集及其导出子图, 最大化 $W_e + P_v$ (点权可负).

  - $(S, u) = U, (u, T) = U - 2P_u - D_u, (u, v) = (v, u) = W_e$

  - $\text{ans} = \frac{Un - C[S,T]}{2}$, 解集为 $S - \{s\}$

- 最大权闭合图: 选 a 则 a 的后继必须被选

  - $P_u > 0, (S, u) = P_u, P_u < 0, (u, T) = -P_u$

  - $\text{ans} = \sum_{P_u > 0} P_u - C[S, T]$, 解集为 $S - \{s\}$

- 判定边是否属于最小割:
  - 可能属于最小割: (u,v) 不属于同一 SCC
  - 一定在所有最小割中: (u,v) 不属于同一 SCC, 且 S,u 在同一 SCC, u,T 在同一 SCC

5 数学

5.1 单纯形 Cpp

$$\max \{cx|Ax \le b, x \ge 0\}$$

```
const int MAXN = 11000, MAXM = 1100;
//  here MAXN is the MAX number of conditions, MAXM is the MAX number of vars

int avali[MAXM], avacnt;
double A[MAXN][MAXM];
double b[MAXN], c[MAXM];
double* simplex(int n, int m) {
//  here n is the number of conditions, m is the number of vars
  m++;
  int r = n, s = m - 1;
  static double D[MAXN + 2][MAXM + 1];
  static int ix[MAXN + MAXM];
  for (int i = 0; i < n + m; i++) ix[i] = i;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m - 1; j++) D[i][j] = -A[i][j];
    D[i][m - 1] = 1;
    D[i][m] = b[i];
    if (D[r][m] > D[i][m]) r = i;
  }
  for (int j = 0; j < m - 1; j++) D[n][j] = c[j];
  D[n + 1][m - 1] = -1;
  for (double d; ; ) {
    if (r < n) {
      int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t;
      D[r][s] = 1.0 / D[r][s];
      for (int j = 0; j <= m; j++) if (j != s) D[r][j] *= -D[r][s];
      avacnt = 0;
      for (int i = 0; i <= m; ++i)
        if(fabs(D[r][i]) > EPS)
          avali[avacnt++] = i;
      for (int i = 0; i <= n + 1; i++) if (i != r) {
        if(fabs(D[i][s]) < EPS) continue;
        double *cur1 = D[i], *cur2 = D[r], tmp = D[i][s];
        //for (int j = 0; j <= m; j++) if (j != s) cur1[j] += cur2[j] * tmp;
        for(int j = 0; j < avacnt; ++j) if(avali[j] != s) cur1[avali[j]] += cur2[avali[j]] * tmp;
        D[i][s] *= D[r][s];
      }
    }
    r = -1; s = -1;
    for (int j = 0; j < m; j++) if (s < 0 || ix[s] > ix[j]) {
      if (D[n + 1][j] > EPS || D[n + 1][j] > -EPS && D[n][j] > EPS) s = j;
    }
    if (s < 0) break;
    for (int i = 0; i < n; i++) if (D[i][s] < -EPS) {
      if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -EPS
          || d < EPS && ix[r + m] > ix[i + m])
        r = i;
    }
    if (r < 0) return null; // 非有界
  }
  if (D[n + 1][m] < -EPS) return null; // 无法执行
  static double x[MAXM - 1];
  for (int i = m; i < n + m; i++) if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
  return x; // 值为 D[n][m]
}
```

5.2 单纯形 Java

```
double[] simplex(double[][] A, double[] b, double[] c) {
  int n = A.length, m = A[0].length + 1, r = n, s = m - 1;
  double[][] D = new double[n + 2][m + 1];
  int[] ix = new int[n + m];
  for (int i = 0; i < n + m; i++) ix[i] = i;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m - 1; j++) D[i][j] = -A[i][j];
```

```
    D[i][m - 1] = 1; D[i][m] = b[i]; if (D[r][m] > D[i][m]) r = i;
  }
  for (int j = 0; j < m - 1; j++) D[n][j] = c[j];
  D[n + 1][m - 1] = -1;
  for (double d; ; ) {
    if (r < n) {
      int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t; D[r][s] = 1.0 / D[r][s];
      for (int j = 0; j <= m; j++) if (j != s) D[r][j] *= -D[r][s];
      for (int i = 0; i <= n + 1; i++) if (i != r) {
        for (int j = 0; j <= m; j++) if (j != s) D[i][j] += D[r][j] * D[i][s];
        D[i][s] *= D[r][s];
      }
    } r = -1; s = -1;
    for (int j = 0; j < m; j++) if (s < 0 || ix[s] > ix[j]) {
      if (D[n + 1][j] > EPS || D[n + 1][j] > -EPS && D[n][j] > EPS) s = j;
    }
    if (s < 0) break;
    for (int i = 0; i < n; i++) if (D[i][s] < -EPS) {
      if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -EPS
          || d < EPS && ix[r + m] > ix[i + m])
        r = i;
    }
    if (r < 0) return null; // 非有界
  } if (D[n + 1][m] < -EPS) return null; // 无法执行
  double[] x = new double[m - 1];
  for (int i = m; i < n + m; i++) if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
  return x; // 值为 D[n][m]
}
```

5.3 FFT

```
namespace FFT {
  #define mul(a, b) (Complex(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x))
  struct Complex {}; // something omitted
  void FFT(Complex P[], int n, int oper) {
    for (int i = 1, j = 0; i < n - 1; i++) {
      for (int s = n; j ^= s >>= 1, ~j & s; );
      if (i < j) swap(P[i], P[j]);
    }
    for (int d = 0; (1 << d) < n; d++) {
      int m = 1 << d, m2 = m * 2;
      double p0 = PI / m * oper;
      Complex unit_p0(cos(p0), sin(p0));
      for (int i = 0; i < n; i += m2) {
        Complex unit(1.0, 0.0);
        for (int j = 0; j < m; j++) {
          Complex &P1 = P[i + j + m], &P2 = P[i + j];
          Complex t = mul(unit, P1);
          P1 = Complex(P2.x - t.x, P2.y - t.y);
          P2 = Complex(P2.x + t.x, P2.y - t.y);
          unit = mul(unit, unit_p0);
  }}}}
  vector<int> doFFT(const vector<int> &a, const vector<int> &b) {
    vector<int> ret(max(0, (int) a.size() + (int) b.size() - 1), 0);
    static Complex A[MAXB], B[MAXB], C[MAXB];
    int len = 1; while (len < (int)ret.size()) len *= 2;
    for (int i = 0; i < len; i++) A[i] = i < (int)a.size() ? a[i] : 0;
    for (int i = 0; i < len; i++) B[i] = i < (int)b.size() ? b[i] : 0;
    FFT(A, len, 1); FFT(B, len, 1);
    for (int i = 0; i < len; i++) C[i] = mul(A[i], B[i]);
    FFT(C, len, -1);
    for (int i = 0; i < (int)ret.size(); i++)
      ret[i] = (int) (C[i].x / len + 0.5);
    return ret;
  }
}
```

5.4 整数 FFT

```
namespace FFT {
// 替代方案: 23068673(= 11 * 2^21 + 1), 原根为 3
  const int MOD = 786433, PRIMITIVE_ROOT = 10; // 3 * 2^18 + 1
  const int MAXB = 1 << 20;
  int getMod(int downLimit) { // 或者现场自己找一个 MOD
    for (int c = 3; ; ++c) { int t = (c << 21) | 1;
      if (t >= downLimit && isPrime(t)) return t;
  }}
```

```
9    int modInv(int a) { return a <= 1 ? a : (long long) (MOD - MOD / a) * modInv(MOD % a) % MOD; }
10   void NTT(int P[], int n, int oper) {
11     for (int i = 1, j = 0; i < n - 1; i++) {
12       for (int s = n; j ^= s >>= 1, ~j & s;);
13       if (i < j) swap(P[i], P[j]);
14     }
15     for (int d = 0; (1 << d) < n; d++) {
16       int m = 1 << d, m2 = m * 2;
17       long long unit_p0 = powMod(PRIMITIVE_ROOT, (MOD - 1) / m2);
18       if (oper < 0) unit_p0 = modInv(unit_p0);
19       for (int i = 0; i < n; i += m2) {
20         long long unit = 1;
21         for (int j = 0; j < m; j++) {
22           int &P1 = P[i + j + m], &P2 = P[i + j];
23           int t = unit * P1 % MOD;
24           P1 = (P2 - t + MOD) % MOD; P2 = (P2 + t) % MOD;
25           unit = unit * unit_p0 % MOD;
26   }}}}
27   vector<int> mul(const vector<int> &a, const vector<int> &b) {
28     vector<int> ret(max(0, (int) a.size() + (int) b.size() - 1), 0);
29     static int A[MAXB], B[MAXB], C[MAXB];
30     int len = 1; while (len < (int)ret.size()) len <<= 1;
31     for (int i = 0; i < len; i++) A[i] = i < (int)a.size() ? a[i] : 0;
32     for (int i = 0; i < len; i++) B[i] = i < (int)b.size() ? b[i] : 0;
33     NTT(A, len, 1); NTT(B, len, 1);
34     for (int i = 0; i < len; i++) C[i] = (long long) A[i] * B[i] % MOD;
35     NTT(C, len, -1); for (int i = 0, inv = modInv(len); i < (int)ret.size(); i++) ret[i] = (long long) C[
         i] * inv % MOD;
36     return ret;
37   }
38 }
```

5.5 扩展欧几里得

$$ax + by = g = gcd(x, y)$$

```
1  void exgcd(LL x, LL y, LL &a0, LL &b0, LL &g) {
2    LL a1 = b0 = 0, b1 = a0 = 1, t;
3    while (y != 0) {
4      t = a0 - x / y * a1, a0 = a1, a1 = t;
5      t = b0 - x / y * b1, b0 = b1, b1 = t;
6      t = x % y, x = y, y = t;
7    } if (x < 0) a0 = -a0, b0 = -b0, x = -x;
8    g = x;
9  }
```

5.6 线性同余方程

- 中国剩余定理: 设 $m_1, m_2, \cdots, m_k$ 两两互素, 则同余方程组 $x \equiv a_i \pmod{m_i}$ for $i = 1, 2, \cdots, k$ 在 $[0, M = m_1 m_2 \cdots m_k)$ 内有唯一解. 记 $M_i = M/m_i$, 找出 p_i 使得 $M_i p_i \equiv 1 \pmod{m_i}$, 记 $e_i = M_i p_i$, 则 $x \equiv e_1 a_1 + e_2 a_2 + \cdots + e_k a_k \pmod{M}$

- 多变元线性同余方程组: 方程的形式为 $a_1 x_1 + a_2 x_2 + \cdots + a_n x_n + b \equiv 0 \pmod{m}$, 令 $d = (a_1, a_2, \cdots, a_n, m)$, 有解的充要条件是 $d|b$, 解的个数为 $m^{n-1} d$

5.7 Miller-Rabin 素性测试

```
1  bool test(LL n, int base) {
2    LL m = n - 1, ret = 0; int s = 0;
3    for ( ; m % 2 == 0; ++s) m >>= 1; ret = pow_mod(base, m, n);
4    if (ret == 1 || ret == n - 1) return true;
5    for (--s; s >= 0; --s) {
6      ret = multiply_mod(ret, ret, n); if (ret == n - 1) return true;
7    } return false;
8  }
9  LL special[7] = {
10   1373653LL,        25326001LL,
11   3215031751LL,     25000000000LL,
12   2152302898747LL,  3474749660383LL, 341550071728321LL};
13 /*
14 * n < 2047                    test[] = {2}
15 * n < 1,373,653               test[] = {2, 3}
16 * n < 9,080,191               test[] = {31, 73}
```

```
17 * n < 25,326,001              test[] = {2, 3, 5}
18 * n < 4,759,123,141           test[] = {2, 7, 61}
19 * n < 1,122,004,669,633       test[] = {2, 13, 23, 1662803}
20 * n < 2,152,302,898,747       test[] = {2, 3, 5, 7, 11}
21 * n < 3,474,749,660,383       test[] = {2, 3, 5, 7, 11, 13}
22 * n < 341,550,071,728,321     test[] = {2, 3, 5, 7, 11, 13, 17}
23 * n < 3,825,123,056,546,413,051  test[] = {2, 3, 5, 7, 11, 13, 17, 19, 23}
24 */
25 bool is_prime(LL n) {
26   if (n < 2) return false;
27   if (n < 4) return true;
28   if (!test(n, 2) || !test(n, 3)) return false;
29   if (n < special[0]) return true;
30   if (!test(n, 5)) return false;
31   if (n < special[1]) return true;
32   if (!test(n, 7)) return false;
33   if (n == special[2]) return false;
34   if (n < special[3]) return true;
35   if (!test(n, 11)) return false;
36   if (n < special[4]) return true;
37   if (!test(n, 13)) return false;
38   if (n < special[5]) return true;
39   if (!test(n, 17)) return false;
40   if (n < special[6]) return true;
41   return test(n, 19) && test(n, 23) && test(n, 29) && test(n, 31) && test(n, 37);
42 }
```

5.8 PollardRho

```
1  LL pollardRho(LL n, LL seed) {
2    LL x, y, head = 1, tail = 2; x = y = random() % (n - 1) + 1;
3    for ( ; ; ) {
4      x = addMod(multiplyMod(x, x, n), seed, n);
5      if (x == y) return n; LL d = gcd(myAbs(x - y), n);
6      if (1 < d && d < n) return d;
7      if (++head == tail) y = x, tail <<= 1;
8  }} vector<LL> divisors;
9  void factorize(LL n) { // 需要保证 n > 1
10   if (isPrime(n)) divisors.push_back(n);
11   else { LL d = n;
12     while (d >= n) d = pollardRho(n, random() % (n - 1) + 1);
13     factorize(n / d); factorize(d);
14 }}
```

5.9 多项式求根

```
1  const double error = 1e-12;
2  const double infi = 1e+12;
3  int n; double a[10], x[10];
4  double f(double a[], int n, double x) {
5    double tmp = 1, sum = 0;
6    for (int i = 0; i <= n; i++) sum = sum + a[i] * tmp, tmp = tmp * x;
7    return sum;
8  }
9  double binary(double l, double r, double a[], int n) {
10   int sl = sign(f(a, n, l)), sr = sign(f(a, n, r));
11   if (sl == 0) return l; if (sr == 0) return r;
12   if (sl * sr > 0) return infi;
13   while (r - l > error) {
14     double mid = (l + r) / 2;
15     int ss = sign(f(a, n, mid));
16     if (ss == 0) return mid;
17     if (ss * sl > 0) l = mid; else r = mid;
18   } return l;
19 }
20 void solve(int n, double a[], double x[], int &nx) {
21   if (n == 1) { x[1] = -a[0] / a[1]; nx = 1; return; }
22   double da[10], dx[10]; int ndx;
23   for (int i = n; i >= 1; i--) da[i - 1] = a[i] * i;
24   solve(n - 1, da, dx, ndx); nx = 0;
25   if (ndx == 0) {
26     double tmp = binary(-infi, infi, a, n);
27     if (tmp < infi) x[++nx] = tmp; return;
28   } double tmp = binary(-infi, dx[1], a, n);
29   if (tmp < infi) x[++nx] = tmp;
30   for (int i = 1; i <= ndx - 1; i++) {
31     tmp = binary(dx[i], dx[i + 1], a, n);
32     if (tmp < infi) x[++nx] = tmp;
```

```
33    } tmp = binary(dx[ndx], infi, a, n);
34    if (tmp < infi) x[++nx] = tmp;
35  }
36  int main() {
37    scanf("%d", &n);
38    for (int i = n; i >= 0; i--) scanf("%lf", &a[i]);
39    int nx; solve(n, a, x, nx);
40    for (int i = 1; i <= nx; i++) printf("%0.6f\n", x[i]);
41    return 0;
42  }
```

5.10 线性递推

$$\text{for } a_{i+n} = \left(\sum_{i=0}^{n-1} k_j a_{i+j}\right) + d, \ a_m = \left(\sum_{i=0}^{n-1} c_i a_i\right) + c_n d$$

```
1   vector<int> recFormula(int n, int k[], int m) {
2     vector<int> c(n + 1, 0);
3     if (m < n) c[m] = 1;
4     else {
5       static int a[MAX_K * 2 + 1];
6       vector<int> b = recFormula(n, k, m >> 1);
7       for (int i = 0; i < n + n; ++i) a[i] = 0;
8       int s = m & 1;
9       for (int i = 0; i < n; i++) {
10        for (int j = 0; j < n; j++) a[i + j + s] += b[i] * b[j];
11        c[n] += b[i];
12      } c[n] = (c[n] + 1) * b[n];
13      for (int i = n * 2 - 1; i >= n; i--) {
14        int add = a[i]; if (add == 0) continue;
15        for (int j = 0; j < n; j++) a[i - n + j] += k[j] * add;
16        c[n] += add;
17      } for (int i = 0; i < n; ++i) c[i] = a[i];
18    } return c;
19  }
```

5.11 原根

原根 g: g 是模 n 简化剩余系构成的乘法群的生成元. 模 n 有原根的充要条件是 $n = 2, 4, p^n, 2p^n$, 其中 p 是奇质数, n 是正整数

```
1   vector<int> findPrimitiveRoot(int N) {
2     if (N <= 4) return vector<int>(1, max(1, N - 1));
3     static int factor[100];
4     int phi = N, totF = 0;
5     { // check no solution and calculate phi
6       int M = N, k = 0;
7       if (~M & 1) M >>= 1, phi >>= 1;
8       if (~M & 1) return vector<int>(0);
9       for (int d = 3; d * d <= M; ++d) if (M % d == 0) {
10        if (++k > 1) return vector<int>(0);
11        for (phi -= phi / d; M % d == 0; M /= d);
12      } if (M > 1) {
13        if (++k > 1) return vector<int>(0); phi -= phi / M;
14      }
15    } { // factorize phi
16      int M = phi;
17      for (int d = 2; d * d <= M; ++d) if (M % d == 0) {
18        for ( ; M % d == 0; M /= d); factor[++totF] = d;
19      } if (M > 1) factor[++totF] = M;
20    } vector<int> ans;
21    for (int g = 2; g <= N; ++g) if (Gcd(g, N) == 1) {
22      bool good = true;
23      for (int i = 1; i <= totF && good; ++i)
24        if (powMod(g, phi / factor[i], N) == 1) good = false;
25      if (!good) continue;
26      for (int i = 1, gp = g; i <= phi; ++i, gp = (LL)gp * g % N)
27        if (Gcd(i, phi) == 1) ans.push_back(gp);
28      break;
29    } sort(ans.begin(), ans.end());
30    return ans;
31  }
```

5.12 离散对数

$A^x \equiv B \pmod{()C}$, 对非质数 C 也适用.

```
1   int modLog(int A, int B, int C) {
2     static pii baby[MAX_SQRT_C + 11];
3     int d = 0; LL k = 1, D = 1; B %= C;
4     for (int i = 0; i < 100; ++i, k = k * A % C) // [0, log C]
5       if (k == B) return i;
6     for (int g; ; ++d) {
7       g = gcd(A, C); if (g == 1) break;
8       if (B % g != 0) return -1;
9       B /= g; C /= g; D = (A / g * D) % C;
10    } int m = (int) ceil(sqrt((double) C)); k = 1;
11    for (int i = 0; i <= m; ++i, k = k * A % C) baby[i] = pii(k, i);
12    sort(baby, baby + m + 1); // [0, m]
13    int n = unique(baby, baby + m + 1, equalFirst) - baby, am = powMod(A, m, C);
14    for (int i = 0; i <= m; ++i) {
15      LL e, x, y; exgcd(D, C, x, y, e); e = x * B % C;
16      if (e < 0) e += C;
17      if (e >= 0) {
18        int k = lower_bound(baby, baby + n, pii(e, -1)) - baby;
19        if (baby[k].first == e) return i * m + baby[k].second + d;
20      } D = D * am % C;
21    } return -1;
22  }
```

5.13 平方剩余

- Legrendre Symbol: 对奇质数 p, $\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{是平方剩余} \\ -1 & \text{是非平方剩余} = a^{\frac{p-1}{2}} \bmod p \\ 0 & a \equiv 0 \pmod{p} \end{cases}$

- 若 p 是奇质数, $\left(\frac{-1}{p}\right) = 1$ 当且仅当 $p \equiv 1 \pmod 4$

- 若 p 是奇质数, $\left(\frac{2}{p}\right) = 1$ 当且仅当 $p \equiv \pm 1 \pmod 8$

- 若 p, q 是奇素数且互质, $\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2} \times \frac{q-1}{2}}$

- Jacobi Symbol: 对奇数 $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$, $\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \left(\frac{a}{p_2}\right)^{\alpha_2} \cdots \left(\frac{a}{p_k}\right)^{\alpha_k}$

- Jacobi Symbol 为 -1 则一定不是平方剩余, 所有平方剩余的 Jacobi Symbol 都是 1, 但 1 不一定是平方剩余

  $ax^2 + bx + c \equiv 0 \pmod{p}$, 其中 $a \neq 0 \pmod{p}$, 且 p 是质数

```
1   inline int normalize(LL a, int P) { a %= P; return a < 0 ? a + P : a; }
2   vector<int> QuadraticResidue(LL a, LL b, LL c, int P) {
3     int h, t; LL r1, r2, delta, pb = 0;
4     a = normalize(a, P); b = normalize(b, P); c = normalize(c, P);
5     if (P == 2) { vector<int> res;
6       if (c % P == 0) res.push_back(0);
7       if ((a + b + c) % P == 0) res.push_back(1);
8       return res;
9     } delta = b * rev(a + a, P) % P;
10    a = normalize(-c * rev(a, P) + delta * delta, P);
11    if (powMod(a, P / 2, P) + 1 == P) return vector<int>(0);
12    for (t = 0, h = P / 2; h % 2 == 0; ++t, h /= 2);
13    r1 = powMod(a, h / 2, P);
14    if (t > 0) { do b = random() % (P - 2) + 2;
15      while (powMod(b, P / 2, P) + 1 != P); }
16    for (int i = 1; i <= t; ++i) {
17      LL d = r1 * r1 % P * a % P;
18      for (int j = 1; j <= t - i; ++j) d = d * d % P;
19      if (d + 1 == P) r1 = r1 * pb % P; pb = pb * pb % P;
20    } r1 = a * r1 % P; r2 = P - r1;
21    r1 = normalize(r1 - delta, P); r2 = normalize(r2 - delta, P);
22    if (r1 > r2) swap(r1, r2); vector<int> res(1, r1);
23    if (r1 != r2) res.push_back(r2);
24    return res;
25  }
```

5.14 N 次剩余

- 若 p 为奇质数, a 为 p 的 n 次剩余的充要条件是 $a^{\frac{p-1}{(a,p-1)}} \equiv 1 \pmod{p}$.

$x^N \equiv a \pmod{p}$, 其中 p 是质数

```
1  vector<int> solve(int p, int N, int a) {
2    if ((a %= p) == 0) return vector<int>(1, 0);
3    int g = findPrimitiveRoot(p), m = modLog(g, a, p); // g ^ m = a (mod p)
4    if (m == -1) return vector<int>(0);
5    LL B = p - 1, x, y, d; exgcd(N, B, x, y, d);
6    if (m % d != 0) return vector<int>(0);
7    vector<int> ret; x = (x * (m / d) % B + B) % B; // g ^ B mod p = g ^ (p - 1) mod p = 1
8    for (int i = 0, delta = B / d; i < d; ++i) {
9      x = (x + delta) % B; ret.push_back((int)powMod(g, x, p));
10   } sort(ret.begin(), ret.end());
11   ret.resize(unique(ret.begin(), ret.end()) - ret.begin());
12   return ret;
13 }
```

5.15 Romberg 积分

```
1  template <class T> double Romberg(const T&f, double a, double b, double eps = 1e-8) {
2    vector<double> t; double h = b - a, last, now; int k = 1, i = 1;
3    t.push_back(h * (f(a) + f(b)) / 2); // 梯形
4    do {
5      last = t.back(); now = 0; double x = a + h / 2;
6      for (int j = 0; j < k; ++j, x += h) now += f(x);
7      now = (t[0] + h * now) / 2; double k1 = 4.0 / 3.0, k2 = 1.0 / 3.0;
8      for (int j = 0; j < i; ++j, k1 = k2 + 1) {
9        double tmp = k1 * now - k2 * t[j];
10       t[j] = now; now = tmp; k2 /= 4 * k1 - k2; // 防止溢出
11     } t.push_back(now); k *= 2; h /= 2; ++i;
12   } while (fabs(last - now) > eps);
13   return t.back();
14 }
```

5.16 公式

5.16.1 级数与三角

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

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

- 错排: $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \cdots + \frac{(-1)^n}{n!}) = (n-1)(D_{n-2} - D_{n-1})$

- $\tan\alpha \pm \tan\beta = \frac{\sin(\alpha\pm\beta)}{\cos\alpha\cos\beta}$

- $\cos n\alpha = \binom{n}{0}\cos^n\alpha - \binom{n}{2}\cos^{n-2}\alpha\sin^2\alpha + \binom{n}{4}\cos^{n-4}\alpha\sin^4\alpha\cdots$

- $\sin n\alpha = \binom{n}{1}\cos^{n-1}\alpha\sin\alpha - \binom{n}{3}\cos^{n-3}\alpha\sin^3\alpha + \binom{n}{5}\cos^{n-5}\alpha\sin^5\alpha\cdots$

- $\sum_{n=1}^{N}\cos nx = \frac{\sin(N+\frac{1}{2})x - \sin\frac{x}{2}}{2\sin\frac{x}{2}}$

- $\sum_{n=1}^{N}\sin nx = \frac{-\cos(N+\frac{1}{2})x + \cos\frac{x}{2}}{2\sin\frac{x}{2}}$

- $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}\cdots$ for $x \in (-\infty, +\infty)$

- $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!}\cdots$ for $x \in (-\infty, +\infty)$

- $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!}\cdots$ for $x \in (-\infty, +\infty)$

- $\arcsin x = x + \sum_{n=1}^{\infty}\frac{(2n-1)!!}{(2n)!!}\frac{x^{2n+1}}{2n+1}$ for $x \in [-1, 1]$

- $\arccos x = \frac{\pi}{2} - \sum_{n=1}^{\infty}\frac{(2n-1)!!}{(2n)!!}\frac{x^{2n+1}}{2n+1}$ for $x \in [-1, 1]$

- $\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5}\cdots$ for $x \in [-1, 1]$

- $\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3}\cdots$ for $x \in (-1, 1]$

- $\int_0^{\frac{\pi}{2}}\sin^n x\mathrm{d}x = \begin{cases} \dfrac{(n-1)!!}{n!!} \times \dfrac{\pi}{2} & n是偶数 \\ \dfrac{(n-1)!!}{n!!} & n是奇数 \end{cases}$

- $\int_0^{+\infty}\frac{\sin x}{x}\mathrm{d}x = \frac{\pi}{2}$

- $\int_0^{+\infty} e^{-x^2}\mathrm{d}x = \frac{\sqrt{\pi}}{2}$

- 傅里叶级数: 设周期为 $2T$. 函数分段连续. 在不连续点的值为左右极限的平均数.

  - $a_n = \frac{1}{T}\int_{-T}^{T} f(x)\cos\frac{n\pi}{T}x\mathrm{d}x$

  - $b_n = \frac{1}{T}\int_{-T}^{T} f(x)\sin\frac{n\pi}{T}x\mathrm{d}x$

  - $f(x) = \frac{a_0}{2} + \sum_{n=1}^{+\infty}(a_n\cos\frac{n\pi}{T}x + b_n\sin\frac{n\pi}{T}x)$

- Beta 函数: $B(p, q) = \int_0^1 x^{p-1}(1-x)^{q-1}\mathrm{d}x$

  - 定义域 $(0, +\infty) \times (0, +\infty)$, 在定义域上连续

  - $B(p, q) = B(q, p) = \frac{q-1}{p+q-1}B(p, q-1) = 2\int_0^{\frac{\pi}{2}}\cos^{2p-1}\phi\sin^{2p-1}\phi\mathrm{d}\phi = \int_0^{+\infty}\frac{t^{q-1}}{(1+t)^{p+q}}\mathrm{d}t = \int_0^1\frac{t^{p-1}+t^{q-1}}{(1+t)^{(p+q)}}$

  - $B(\frac{1}{2}, \frac{1}{2}) = \pi$

- Gamma 函数: $\Gamma = \int_0^{+\infty} x^{s-1}e^{-x}\mathrm{d}x$

  - 定义域 $(0, +\infty)$, 在定义域上连续
  - $\Gamma(1) = 1$, $\Gamma(\frac{1}{2}) = \sqrt{\pi}$
  - $\Gamma(s) = (s-1)\Gamma(s-1)$
  - $B(p, q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}$
  - $\Gamma(s)\Gamma(1-s) = \frac{\pi}{\sin\pi s}$ for $s > 0$

– $\Gamma(s)\Gamma(s+\frac{1}{2}) = 2\sqrt{\pi}\frac{\Gamma(s)}{2^{2s-1}}$ for $0 < s < 1$

- 积分：平面图形面积、曲线弧长、旋转体体积、旋转曲面面积 $y = f(x)$, $\int_a^b f(x)\mathrm{d}x$, $\int_a^b \sqrt{1+f'^2(x)}\mathrm{d}x$,

$\pi\int_a^b f^2(x)\mathrm{d}x$, $2\pi\int_a^b |f(x)|\sqrt{1+f'^2(x)}\mathrm{d}x$

$x = x(t), y = y(t), t \in [T_1, T_2]$, $\int_{T_1}^{T_2}|y(t)x'(t)|\mathrm{d}t$, $\int_{T_1}^{T_2}\sqrt{x'^2(t)+y'^2(t)}\mathrm{d}t$, $\pi\int_{T_1}^{T_2}|x'(t)|y^2(t)\mathrm{d}t$,

$2\pi\int_{T_1}^{T_2}|y(t)|\sqrt{x'^2(t)+y'^2(t)}\mathrm{d}t$,

$r = r(\theta), \theta \in [\alpha, \beta]$, $\frac{1}{2}\int_\alpha^\beta r^2(\theta)\mathrm{d}\theta$, $\int_\alpha^\beta \sqrt{r^2(\theta)+r'^2(\theta)}\mathrm{d}\theta$, $\frac{2}{3}\pi\int_\alpha^\beta r^3(\theta)\sin\theta\mathrm{d}\theta$,

$2\pi\int_\alpha^\beta r(\theta)\sin\theta\sqrt{r^2(\theta)+r'^2(\theta)}\mathrm{d}\theta$

5.16.2 三次方程求根公式

对一元三次方程 $x^3 + px + q = 0$, 令

$$A = \sqrt[3]{-\frac{q}{2} + \sqrt{(\frac{q}{2})^2 + (\frac{p}{3})^3}}$$

$$B = \sqrt[3]{-\frac{q}{2} - \sqrt{(\frac{q}{2})^2 + (\frac{p}{3})^3}}$$

$$\omega = \frac{(-1+\mathrm{i}\sqrt{3})}{2}$$

则 $x_j = A\omega^j + B\omega^{2j}$ (j = 0, 1, 2).

当求解 $ax^3 + bx^2 + cx + d = 0$ 时, 令 $x = y - \frac{b}{3a}$, 再求解 y, 即转化为 $y^3 + py + q = 0$ 的形式. 其中,

$$p = \frac{b^2 - 3ac}{3a^2}$$

$$q = \frac{2b^3 - 9abc + 27a^2d}{27a^3}$$

卡尔丹判别法: 令 $\Delta = (\frac{q}{2})^2 + (\frac{p}{3})^3$. 当 $\Delta > 0$ 时, 有一个实根和一对个共轭虚根；当 $\Delta = 0$ 时, 有三个实根, 其中两个相等；当 $\Delta < 0$ 时, 有三个不相等的实根.

5.16.3 椭圆

- 椭圆 $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$, 其中离心率 $e = \frac{c}{a}, c = \sqrt{a^2 - b^2}$; 焦点参数 $p = \frac{b^2}{a}$

- 椭圆上 (x, y) 点处的曲率半径为 $R = a^2b^2(\frac{x^2}{a^4} + \frac{y^2}{b^4})^{\frac{3}{2}} = \frac{(r_1r_2)^{\frac{3}{2}}}{ab}$, 其中 r_1 和 r_2 分别为 (x, y) 与两焦点 F_1 和 F_2 的距离.

- 椭圆的周长 $L = 4a\int_0^{\frac{\pi}{2}}\sqrt{1 - e^2\sin^2 t}\mathrm{d}t = 4aE(e, \frac{\pi}{2})$, 其中

$$E(e, \frac{\pi}{2}) = \frac{\pi}{2}[1 - (\frac{1}{2})^2e^2 - (\frac{1\times 3}{2\times 4})^2\frac{e^4}{3} - (\frac{1\times 3\times 5}{2\times 4\times 6})^2\frac{e^6}{5} - \cdots]$$

- 设椭圆上点 $M(x, y), N(x, -y), x, y > 0, A(a, 0)$, 原点 $O(0,0)$, 扇形 OAM 的面积 $S_{OAM} = \frac{1}{2}ab\arccos\frac{x}{a}$, 弓形 MAN 的面积 $S_{MAN} = ab\arccos\frac{x}{a} - xy$.

- 设 θ 为 (x, y) 点关于椭圆中心的极角, r 为 (x, y) 到椭圆中心的距离, 椭圆极坐标方程:

$$x = r\cos\theta, y = r\sin\theta, r^2 = \frac{b^2a^2}{b^2\cos^2\theta + a^2\sin^2\theta}$$

5.16.4 抛物线

- 标准方程 $y^2 = 2px$, 曲率半径 $R = \frac{(p+2x)^{\frac{3}{2}}}{\sqrt{p}}$

- 弧长: 设 $M(x, y)$ 是抛物线上一点, 则 $L_{OM} = \frac{p}{2}[\sqrt{\frac{2x}{p}(1 + \frac{2x}{p})} + \ln(\sqrt{\frac{2x}{p}} + \sqrt{1 + \frac{2x}{p}})]$

- 弓形面积: 设 M, D 是抛物线上两点, 且分居一, 四象限. 做一条平行于 MD 且与抛物线相切的直线 L. 若 M 到 L 的距离为 h. 则有 $S_{MOD} = \frac{2}{3}MD \cdot h$.

5.16.5 重心

- 半径 r, 圆心角为 θ 的扇形的重心与圆心的距离为 $\frac{4r\sin\frac{\theta}{2}}{3\theta}$

- 半径 r, 圆心角为 θ 的圆弧的重心与圆心的距离为 $\frac{4r\sin^3\frac{\theta}{2}}{3(\theta - \sin\theta)}$

- 椭圆上半部分的重心与圆心的距离为 $\frac{4b}{3\pi}$

- 抛物线中弓形 MOD 的重心满足 $CQ = \frac{2}{5}PQ$, P 是直线 L 与抛物线的切点, Q 在 MD 上且 PQ 平行 x 轴, C 是重心

5.16.6 向量恒等式

- $\vec{a}\cdot(\vec{b}\times\vec{c}) = \vec{b}\cdot(\vec{c}\times\vec{a}) = \vec{c}\cdot(\vec{a}\times\vec{b})$

- $\vec{a}\times(\vec{b}\times\vec{c}) = (\vec{c}\times\vec{b})\times\vec{a} = \vec{b}(\vec{a}\cdot\vec{c}) - \vec{c}(\vec{a}\cdot\vec{b})$

5.16.7 常用几何公式

- 三角形的五心

  – 重心 $\vec{G} = \frac{\vec{A}+\vec{B}+\vec{C}}{3}$

  – 内心 $\vec{I} = \frac{a\vec{A}+b\vec{B}+c\vec{C}}{a+b+c}$, $R = \frac{2S}{a+b+c}$

  – 外心 $\vec{O} = \frac{\vec{A}+\vec{B}+\frac{\vec{AC}\cdot\vec{BC}}{\vec{AB}\times\vec{BC}}\vec{AB}^T}{2}$, $R = \frac{abc}{4S}$

  – 垂心 $\vec{H} = 3\vec{G} - 2\vec{O} = \vec{C} + \frac{\vec{BC}\cdot\vec{AC}}{\vec{BC}\times\vec{AC}}\vec{AB}^T$

  – 旁心 (三个) $\frac{-a\vec{A}+b\vec{B}+c\vec{C}}{-a+b+c}$

- 四边形: 设 D_1, D_2 为对角线, M 为对角线中点连线, A 为对角线夹角

– $a^2 + b^2 + c^2 + d^2 = D_1^2 + D_2^2 + 4M^2$

– $S = \frac{1}{2} D_1 D_2 \sin A$

– $ac + bd = D_1 D_2$ (内接四边形适用)

– Bretschneider 公式: $S = \sqrt{(p-a)(p-b)(p-c)(p-d) - abcd \cos^2(\frac{\theta}{2})}$, 其中 θ 为对角和

5.16.8 树的计数

- 有根数计数: 令 $S_{n,j} = \sum\limits_{1 \le i \le n/j} a_{n+1-ij} = S_{n-j,j} + a_{n+1-j}$

  于是, $n+1$ 个结点的有根数的总数为 $a_{n+1} = \dfrac{\sum\limits_{1 \le j \le n} j \cdot a_j \cdot S_{n,j}}{n}$

  附: $a_1 = 1, a_2 = 1, a_3 = 2, a_4 = 4, a_5 = 9, a_6 = 20, a_9 = 286, a_{11} = 1842$

- 无根树计数: 当 n 是奇数时, 则有 $a_n - \sum\limits_{1 \le i \le \frac{n}{2}} a_i a_{n-i}$ 种不同的无根树

  当 n 是偶数时, 则有 $a_n - \sum\limits_{1 \le i \le \frac{n}{2}} a_i a_{n-i} + \dfrac{1}{2} a_{\frac{n}{2}} (a_{\frac{n}{2}} + 1)$ 种不同的无根树

- Matrix-Tree 定理: 对任意图 G, 设 $\mathrm{mat}[i][i] = i$ 的度数, $\mathrm{mat}[i][j] = i$ 与 j 之间边数的相反数, 则 $\mathrm{mat}[i][j]$ 的任意余子式的行列式就是该图的生成树个数

5.17 小知识

- 勾股数: 设正整数 n 的质因数分解为 $n = \prod p_i^{a_i}$, 则 $x^2 + y^2 = n$ 有整数解的充要条件是 n 中不存在形如 $p_i \equiv 3 \pmod 4$ 且指数 a_i 为奇数的质因数 p_i

- 勾股数 2:

```
a[0] := 0;
s := 0;
for i := 1 to n - 2 do
  begin
    a[i] := a[i - 1] + 1;
    s := s + sqr(a[i]);
  end;
{======s + sqr(a[n-1]) + sqr(a[n]) = k^2======}
a[n - 1] := a[n - 2];
repeat
  a[n - 1] := a[n - 1] + 1;
until odd(s + sqr(a[n - 1])) and (a[n - 1] > 2);
a[n] := (s + sqr(a[n - 1]) - 1) shr 1;
```

  知道 s 和 a[n-1] 后, 直接求了 a[n]. 神奇了点.

  其实, 有当 n 为奇数: $n^2 + \lfloor \frac{n^2-1}{2} \rfloor^2 = \lfloor \frac{n^2+1}{2} \rfloor^2$

  若:
  $a = k \cdot (s^2 - t^2)$
  $b = 2 \cdot k \cdot s \cdot t$
  $c = k \cdot (s^2 + t^2)$
  则 $c^2 = a^2 + b^2$.

- Stirling 公式: $n! \approx \sqrt{2\pi n} (\frac{n}{e})^n$

- Pick 定理: 简单多边形, 不自交, 顶点如果全是整点. 则: 严格在多边形内部的整点数 $+ \frac{1}{2}$ 在边上的整点数 $- 1 =$ 面积

- Mersenne 素数: p 是素数且 $2^p - 1$ 的数是素数. (10000 以内的 p 有: 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941)

- Fermat 分解算法: 从 $t = \sqrt{n}$ 开始, 依次检查 $t^2 - n, (t+1)^2 - n, (t+2)^2 - n, \ldots$, 直到出现一个平方数 y, 由于 $t^2 - y^2 = n$, 因此分解得 $n = (t-y)(t+y)$. 显然, 当两个因数很接近时这个方法能很快找到结果, 但如果遇到一个素数, 则需要检查 $\frac{n+1}{2} - \sqrt{n}$ 个整数

- 牛顿迭代: $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

- 球与盒子的动人故事: (n 个球, m 个盒子, S 为第二类斯特林数)

  1. 球同, 盒同, 无空: dp
  2. 球同, 盒同, 可空: dp
  3. 球同, 盒不同, 无空: $\binom{n-1}{m-1}$
  4. 球同, 盒不同, 可空: $\binom{n+m-1}{n-1}$
  5. 球不同, 盒同, 无空: $S(n,m)$
  6. 球不同, 盒同, 可空: $\sum_{k=1}^{m} S(n,k)$
  7. 球不同, 盒不同, 无空: $m! S(n,m)$
  8. 球不同, 盒不同, 可空: m^n

- 组合数奇偶性: 若 $(n \& m) = m$, 则 $\binom{n}{m}$ 为奇数, 否则为偶数

- 格雷码 $G(x) = x \otimes (x >> 1)$

- Bell 数: B_n 代表将 n 个元素划分成若干个非空集合的方案数

  – $B_0 = B_1 = 1, B_n = \sum\limits_{k=0}^{n-1} \binom{n-1}{k} B_k$

  – $B_n = \sum\limits_{k=0}^{n} \left\{ {n \atop k} \right\}$

  – Bell 三角形: $a_{1,1} = 1, a_{n,1} = a_{n-1,n-1}, a_{n,m} = a_{n,m-1} + a_{n-1,m-1}, B_n = a_{n,1}$

  – 对质数 p, $B_{n+p} \equiv B_n + B_{n+1} \pmod p$

  – 对质数 p, $B_{n+p^m} \equiv m B_n + B_{n+1} \pmod p$

  – 对质数 p, 模的周期一定是 $\frac{p^p - 1}{p - 1}$ 的约数, $p \le 101$ 时就是这个值

  – 从 B_0 开始, 前几项是 $1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975 \cdots$

6 其他

6.1 Extended LIS

```
1  int G[MAXN][MAXN];
2  void insertYoung(int v) {
3    for (int x = 1, y = INT_MAX; ; ++x) {
4      Down(y, *G[x]); while (y > 0 && G[x][y] >= v) --y;
5      if (++y > *G[x]) { ++*G[x]; G[x][y] = v; break; }
6      else swap(G[x][y], v);
7    }
8  }
9  int solve(int N, int seq[]) {
10   Rep(i, 1, N) *G[i] = 0;
11   Rep(i, 1, N) insertYoung(seq[i]);
12   printf("%d\n", *G[1] + *G[2]);
13   return 0;
14 }
```

6.2   生成 nCk

```
1  void nCk(int n, int k) {
2    for (int comb = (1 << k) - 1; comb < (1 << n); ) {
3      int x = comb & -comb, y = comb + x;
4      comb = (((comb & ~y) / x) >> 1) | y;
5    }
6  }
```

6.3   nextPermutation

```
1  boolean nextPermutation(int[] is) {
2    int n = is.length;
3    for (int i = n - 1; i > 0; i--) {
4      if (is[i - 1] < is[i]) {
5        int j = n; while (is[i - 1] >= is[--j]);
6        swap(is, i - 1, j); // swap is[i - 1], is[j]
7        rev(is, i, n); // reverse is[i, n)
8        return true;
9      }
10   } rev(is, 0, n);
11   return false;
12 }
```

6.4   Josephus 数与逆 Josephus 数

```
1  int josephus(int n, int m, int k) { int x = -1;
2    for (int i = n - k + 1; i <= n; i++) x = (x + m) % i; return x;
3  }
4  int invJosephus(int n, int m, int x) {
5    for (int i = n; ; i--) { if (x == i) return n - i; x = (x - m % i + i) % i; }
6  }
```

6.5   表达式求值

```
1  inline int getLevel(char ch) {
2    switch (ch) { case '+': case '-': return 0; case '*': return 1; } return -1;
3  }
4  int evaluate(char *&p, int level) {
5    int res;
6    if (level == 2) {
7      if (*p == '(') ++p, res = evaluate(p, 0);
8      else res = isdigit(*p) ? *p - '0' : value[*p - 'a'];
9      ++p; return res;
10   } res = evaluate(p, level + 1);
11   for (int next; *p && getLevel(*p) == level; ) {
12     char op = *p++; next = evaluate(p, level + 1);
13     switch (op) {
14       case '+': res += next; break;
15       case '-': res -= next; break;
16       case '*': res *= next; break;
17     }
18   } return res;
19 }
20 int makeEvaluation(char *str) { char *p = str; return evaluate(p, 0); }
```

6.6   直线下的整点个数

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

```
1  LL count(LL n, LL a, LL b, LL m) {
2    if (b == 0) return n * (a / m);
3    if (a >= m) return n * (a / m) + count(n, a % m, b, m);
4    if (b >= m) return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);
5    return count((a + b * n) / m, (a + b * n) % m, m, b);
6  }
```

6.7   Java 多项式

```
1  class Polynomial {
2    final static Polynomial ZERO = new Polynomial(new int[] { 0 });
3    final static Polynomial ONE = new Polynomial(new int[] { 1 });
4    final static Polynomial X = new Polynomial(new int[] { 0, 1 });
5    int[] coef;
6    static Polynomial valueOf(int val) { return new Polynomial(new int[] { val }); }
7    Polynomial(int[] coef) { this.coef = Arrays.copyOf(coef, coef.length); }
8    Polynomial add(Polynomial o, int mod); // omitted
9    Polynomial subtract(Polynomial o, int mod); // omitted
10   Polynomial multiply(Polynomial o, int mod); // omitted
11   Polynomial scale(int o, int mod); // omitted
12   public String toString() {
13     int n = coef.length; String ret = "";
14     for (int i = n - 1; i > 0; --i) if (coef[i] != 0)
15       ret += coef[i] + "x^" + i + "+";
16     return ret + coef[0];
17   }
18   static Polynomial lagrangeInterpolation(int[] x, int[] y, int mod) {
19     int n = x.length; Polynomial ret = Polynomial.ZERO;
20     for (int i = 0; i < n; ++i) {
21       Polynomial poly = Polynomial.valueOf(y[i]);
22       for (int j = 0; j < n; ++j) if (i != j) {
23         poly = poly.multiply(
24           Polynomial.X.subtract(Polynomial.valueOf(x[j]), mod), mod);
25         poly = poly.scale(powMod(x[i] - x[j] + mod, mod - 2, mod), mod);
26       } ret = ret.add(poly, mod);
27     } return ret;
28   }
29 }
```

6.8   long long 乘法取模

```
1  LL multiplyMod(LL a, LL b, LL P) { // 需要保证 a 和 b 非负
2    LL t = (a * b - LL((long double)a / P * b + 1e-3) * P) % P;
3    return t < 0 ? t + P : t;
4  }
```

6.9   重复覆盖

```
1  struct node { int x, y; node *l, *r, *u, *d; } base[MAX * MAX], *top, *head;
2  typedef node *link;
3  int row, col, nGE, ans, stamp, cntc[MAX], vis[MAX];
4  void removeExact(link c) { c->l->r = c->r; c->r->l = c->l;
5    for (link i = c->d; i != c; i = i->d)
6      for (link j = i->r; j != i; j = j->r) j->d->u = j->u, j->u->d = j->d, --cntc[j->y];
7  }
8  void resumeExact(link c) {
9    for (link i = c->u; i != c; i = i->u)
10     for (link j = i->l; j != i; j = j->l) j->d->u = j, j->u->d = j, ++cntc[j->y];
11   c->l->r = c; c->r->l = c;
12 }
13 void removeRepeat(link c) { for (link i = c->d; i != c; i = i->d) i->l->r = i->r, i->r->l = i->l; }
14 void resumeRepeat(link c) { for (link i = c->u; i != c; i = i->u) i->l->r = i; i->r->l = i; }
15 int calcH() { int y, res = 0; ++stamp;
16   for (link c = head->r; (y = c->y) <= row && c != head; c = c->r) if (vis[y] != stamp) {
17     vis[y] = stamp; ++res; for (link i = c->d; i != c; i = i->d)
18       for (link j = i->r; j != i; j = j->r) vis[j->y] = stamp;
19   } return res;
```

```
20  }
21  void DFS(int dep) { if (dep + calcH() >= ans) return;
22    if (head->r->y > nGE || head->r == head) { if (ans > dep) ans = dep; return; }
23    link c = NULL;
24    for (link i = head->r; i->y <= nGE && i != head; i = i->r)
25      if (!c || cntc[i->y] < cntc[c->y]) c = i;
26    for (link i = c->d; i != c; i = i->d) {
27      removeRepeat(i);
28      for (link j = i->r; j != i; j = j->r) if (j->y <= nGE) removeRepeat(j);
29      for (link j = i->r; j != i; j = j->r) if (j->y  > nGE) removeExact(base + j->y);
30      DFS(dep + 1);
31      for (link j = i->l; j != i; j = j->l) if (j->y  > nGE) resumeExact(base + j->y);
32      for (link j = i->l; j != i; j = j->l) if (j->y <= nGE) resumeRepeat(j);
33      resumeRepeat(i);
34    }
35  }
```

6.10 星期几判定

```
1  int getDay(int y, int m, int d) {
2    if (m <= 2) m += 12, y--;
3    if (y < 1752 || (y == 1752 && m < 9) || (y == 1752 && m == 9 && d < 3))
4      return (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 + 5) % 7 + 1;
5    return (d + 2 * m + 3 * (m + 1) / 5 + y + y / 4 - y / 100 + y / 400) % 7 + 1;
6  }
```

6.11 LCSequence Fast

```
1  ULL *a, *b, *s, c, d;
2  for (i = 0, a = appear[(int)B[k]], b = row[max(k - 1, 0)], s = X; i < bitSetLen; ++i)
3    *s++ = *a++ | *b++;                    // X = row[i - 1] or appear[ B[i] ]
4  for (i = 0, a = dp, c = d = 0; i < bitSetLen; ++a, c = d, ++i)
5    d = *a >> 63, *a = ~((*a << 1) + c);    // row[i] = -((row[i] << 1) + 1)
6  for (i = 0, a = dp, b = X, c = 0; i < bitSetLen; ++a, ++b, ++i)
7    d = *b + c, c = (*a >= -d), *a += d;    // row[i] += X
8  for (i = 0, a = dp, b = X; i < bitSetLen; ++a, ++b, ++i)
9    *a = (*a ^ *b) & *b;                    // row[i] = X and (row[i] xor X)
```

7 Templates

7.1 vim 配置

在.bashrc 中加入 export CXXFLAGS=”-Wall -Wconversion -Wextra -g3”

```
1  set nu ru nobk cindent si
2  set mouse=a sw=4 sts=4 ts=4
3  set hlsearch incsearch
4  set whichwrap=b,s,<,>,[,]
5  syntax on
6
7  nmap <C-A> ggVG
8  vmap <C-C> "+y
9
10 autocmd␣BufNewFile␣*.cpp␣0r␣~/Templates/cpp.cpp
11 map<F9>␣:!g++␣%␣-o␣%<␣-Wall␣-Wconversion␣-Wextra␣-g3␣<CR>
12 map<F5>␣:!./%<␣<CR>
13 map<F8>␣:!./%<␣%<.in␣<CR>
14
15 map<F3>␣:vnew␣%<.in␣<CR>
16 map<F4>␣:!(gedit␣%␣&)<CR>
```

7.2 C++

```
1  #pragma comment(linker, "/STACK:10240000")
2  #include <cstdio>
3  #include <cstdlib>
4  #include <cstring>
5  #include <iostream>
```

```
6  #include <algorithm>
7  #define Rep(i, a, b) for(int i = (a); i <= (b); ++i)
8  #define Foru(i, a, b) for(int i = (a); i < (b); ++i)
9  using namespace std;
10 typedef long long LL;
11 typedef pair<int, int> pii;
12 namespace BufferedReader {
13   char buff[MAX_BUFFER + 5], *ptr = buff, c; bool flag;
14   bool nextChar(char &c) {
15     if ( ( c = *ptr++) == 0 ) {
16       int tmp = fread(buff, 1, MAX_BUFFER, stdin);
17       buff[tmp] = 0; if (tmp == 0) return false;
18       ptr = buff; c = *ptr++;
19     } return true;
20   }
21   bool nextUnsignedInt(unsigned int &x) {
22     for (;;){if (!nextChar(c)) return false; if ('0'<=c && c<='9') break;}
23     for (x=c-'0'; nextChar(c); x = x * 10 + c - '0') if (c < '0' || c > '9') break;
24     return true;
25   }
26   bool nextInt(int &x) {
27     for (;;) { if (!nextChar(c)) return false; if (c=='-' || ('0'<=c && c<='9')) break; }
28     for ((c=='-') ? (x=0,flag=true) : (x=c-'0',flag=false); nextChar(c); x=x*10+c-'0')
29       if (c<'0' || c>'9') break;
30     if (flag) x=-x; return true;
31   }
32 };
33 #endif
```

7.3 Java

```
1  import java.io.*;
2  import java.util.*;
3  import java.math.*;
4
5  public class Main {
6    public void solve() {}
7    public void run() {
8      tokenizer = null; out = new PrintWriter(System.out);
9      in = new BufferedReader(new InputStreamReader(System.in));
10     solve();
11     out.close();
12   }
13   public static void main(String[] args) {
14     new Main().run();
15   }
16   public StringTokenizer tokenizer;
17   public BufferedReader in;
18   public PrintWriter out;
19   public String next() {
20     while (tokenizer == null || !tokenizer.hasMoreTokens()) {
21       try { tokenizer = new StringTokenizer(in.readLine()); }
22       catch (IOException e) { throw new RuntimeException(e); }
23     } return tokenizer.nextToken();
24   }
25 }
```

7.4 Eclipse 配置

Exec=env UBUNTU_MENUPROXY= /opt/eclipse/eclipse

preference general keys 把 word completion 设置成 alt+c, 把 content assistant 设置成 alt + /