1 $O(m^2 \log n)$ 线性递推第 n 项

已知 $a_0, a_1, ..., a_{m-1}$
$a_n = c_0 * a_{n-m} + ... + c_{m-1} * a_{n-1}$
求 $a_n = v_0 * a_0 + v_1 * a_1 + ... + v_{m-1} * a_{m-1}$

```
void linear_recurrence(long long n, int m, int a[], int c[], int p) {
  long long v[M] = {1 % p}, u[M << 1], msk = !!n;
  for(long long i(n); i > 1; i >>= 1) msk <<= 1;
  for(long long x(0); msk; msk >>= 1, x <<= 1) {
    fill_n(u, m << 1, 0);
    int b(!!(n & msk));
    x |= b;
    if(x < m) u[x] = 1 % p;
    else {
      for(int i(0); i < m; i++)
        for(int j(0), t(i + b); j < m; j++, t++)
          u[t] = (u[t] + v[i] * v[j]) % p;
      for(int i((m << 1) - 1); i >= m; i--)
        for(int j(0), t(i - m); j < m; j++, t++)
          u[t] = (u[t] + c[j] * u[i]) % p;
    }
    copy(u, u + m, v);
  }
  //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
  for(int i(m); i < 2 * m; i++) {
    a[i] = 0;
    for(int j(0); j < m; j++)
      a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
  }
  for(int j(0); j < m; j++) {
    b[j] = 0;
    for(int i(0); i < m; i++)
      b[j] = (b[j] + v[i] * a[i + j]) % p;
  }
  for(int j(0); j < m; j++)
    a[j] = b[j];
}
```

2 NTT

```
const int modulo(786433);
const int G(10);//原根
int pw[999999];
void FFT(int 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]);
  }
  int unit_p0;
  for(int d(0); (1 << d) < n; d++) {
    int m(1 << d), m2(m * 2);
    unit_p0 = oper == 1?pw[(modulo - 1) / m2]:pw[modulo - 1 - (modulo - 1) / m2];
    for(int i = 0; i < n; i += m2) {
      int unit(1);
      for(int j(0); j < m; j++) {
        int &P1 = P[i + j + m], &P2 = P[i + j];
        int t = (long long)unit * P1 % modulo;
        P1 = (P2 - t + modulo) % modulo;
        P2 = (P2 + t) % modulo;
        unit = (long long)unit * unit_p0 % modulo;
      }
    }
  }
}

int nn;
int A[N], B[N], C[N];
//A * B = C;
//len = nn
void multiply() {
  FFT(A, nn, 1);
  FFT(B, nn, 1);
  for(int i(0); i < nn; i++) {
    C[i] = (long long)A[i] * B[i] % modulo;
  }
  FFT(C,  nn, -1);
}

int main() {
  pw[0] = 1;
  for(int i(1); i < modulo; i++) {
    pw[i] = (long long)pw[i - 1] * G % modulo;
  }
}
```

3 中国剩余定理

```
inline void euclid(const long long &a, const long long &b, long long &x, long long &y) {
  if (b == 0) x = 1, y = 0;
  else euclid(b, a % b, x, y), x -= a / b * y, swap(x, y);
}
inline bool crt(int n, long long r[], long long m[], long long &remainder, long long &
    modular) {
  remainder = modular = 1;
  for (int i = 0; i < n; ++i) {
    long long x, y; euclid(modular, m[i], x, y);
    long long divisor = gcd(modular, m[i]);
    if ((r[i] - remainder) % divisor) return false;
    x *= (r[i] - remainder) / divisor; ((x %= m[i]) += m[i]) %= m[i];
    remainder += modular * x; modular *= m[i] / divisor;
    ((remainder %= modular) += modular) %= modular;
  } return true;
}
```

4 Miller Rabin

```
int const n = 3; int const base[] = {2, 7, 61};
int const n = 9; int const base[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
inline long long power(const long long &x, const long long &k, const long long &modular)
    {
  long long ans = 1, num = x % modular;
  for (long long i = k; i > 0; i >>= 1) {
    if (i & 1) ans = multiply(ans, num, modular);
    num = multiply(num, num, modular);
  } return ans;
}
inline bool check(const long long &p, const long long &base) {
  long long n = p - 1; for (; !(n & 1); n >>= 1);
  long long m = power(base, n, p);
  for (; n != p - 1 && m != 1 && m != p - 1; )
    m = multiply(m, m, p), n <<= 1;
  return m == p - 1 || (n & 1) == 1;
}
inline bool prime(const long long &p) {
  for (int i = 0; i < n; ++i) if (base[i] == p) return true;
  if (p == 1 || !(p & 1)) return false;
  for (int i = 0; i < n; ++i) if (!check(p, base[i])) return false;
  return true;
}
```

5 Pollard Rho

```
inline long long pollard_rho(const long long &n, const long long &c) {
  long long x = rand() % (n - 1) + 1, y = x;
  for (int head = 1, tail = 2; true; ) {
    x = multiply(x, x, n);
    if ((x += c) >= n) x -= n;
    if (x == y) return n;
    long long d = __gcd(abs(x - y), n);
    if (d > 1 && d < n) return d;
    if ((++head) == tail) y = x, tail <<= 1;
  }
}
inline vector<long long> mergy(const vector<long long> &a, const vector<long long> &b) {
  vector<long long> vec;
  for (int i = 0; i < (int)a.size(); ++i) vec.push_back(a[i]);
  for (int i = 0; i < (int)b.size(); ++i) vec.push_back(b[i]);
  return vec;
}
inline vector<long long> factor(const long long &n) {
  if (n <= 1) return vector<long long>();
  if (miller_rabin::prime(n)) return vector<long long>(1, n);
  long long p = n; for (; p >= n; p = pollard_rho(n, rand() % (n - 1) + 1));
  return mergy(factor(n / p), factor(p));
}
```

6 直线下整点个数

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

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

7 FFT

```
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]);
  }
  Complex unit_p0;
  for (int d(0); (1 << d) < n; d++) {
    int m(1 << d), m2(m * 2);
    double p0(pi / m * oper);
    unit_p0.imag(sin(p0));
    unit_p0.real(cos(p0));
    for (int i(0); i < n; i += m2) {
      Complex unit = 1;
      for (int j = 0; j < m; j++) {
        Complex &P1 = P[i + j + m], &P2 = P[i + j];
        Complex t = unit * P1;
        P1 = P2 - t;
        P2 = P2 + t;
        unit = unit * unit_p0;
      }
    }
  }
}
void multiply() {
  FFT(a, n, 1);
  FFT(b, n, 1);
  for(int i(0); i < n; i++) {
    c[i] = a[i] * b[i];
  }
  FFT(c, n, -1);
  for(int i(0); i < n; i++) {
    ans[i] += (int)(c[i].real() / n + 0.5);
  }
}
```

8 解一元三次方程 + 求三阶二次型的标准型

```
double sqr(const double & x) {
    return x * x;
}
double eps(1e-8);
int main() {
    double A, B, C, D, E, F;
    for(;6 == scanf("%lf%lf%lf%lf%lf%lf", &A, &B, &C, &D, &E, &F);) {
      D /= 2; E /= 2; F /= 2;
        complex<double> a(1), b(-A - B - C), c(A * B + B * C + C * A - sqr(D) - sqr(E) -
            sqr(F)), d(-A * B * C - 2 * D * E * F + A * sqr(D) + B * sqr(E) + C * sqr(F)
            );
        complex<double> delta(pow(pow(b * c / 6. / a / a - b * b * b / 27. / a / a / a -
            d / 2. / a, 2) + pow(c / 3. / a - b * b / 9. / a / a, 3), 0.5));
        complex<double> p(pow(b * c / 6. / a / a - b * b * b / 27. / a / a / a - d / 2. /
             a + delta, 1. / 3));
        complex<double> q(pow(b * c / 6. / a / a - b * b * b / 27. / a / a / a - d / 2. /
             a - delta, 1. / 3));
        complex<double> omega1(-0.5, 0.5 * sqrt(3.)), omega2(-0.5, -0.5 * sqrt(3.));
        complex<double> x1(-b / 3. / a + p + q), x2(-b / 3. / a + omega1 * p + omega2 * q
            ), x3(-b / 3. / a + omega2 * p + omega1 * q);
        printf("%.10f\n", min(min(sqrt(1 / x1.real()), sqrt(1 / x2.real())), sqrt(1 / x3.
            real())));
    }
}
```

9 自适应辛普森

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

10 圆与多边形交

```
const double eps = 5e-7;
const int N = 2222;
const double pi = acos(-1.0);
int sign(double x) { return x < -eps ? -1 : x > eps; }
double sqr(double x) { return x * x; }
struct Point {
  double x, y;
  Point (double x = 0, double y = 0) : x(x), y(y) {}
  friend inline Point operator +(const Point &a, const Point &b) {
    return Point(a.x + b.x, a.y + b.y);
  }
  friend inline Point operator -(const Point &a, const Point &b) {
    return Point(a.x - b.x, a.y - b.y);
  }
  friend inline Point operator *(const Point &a, double k) {
    return Point(a.x * k, a.y * k);
  }
  friend inline Point operator /(const Point &a, double k) {
    return Point(a.x / k, a.y / k);
  }
  double dist() const { return hypot(x, y); }
  double dist2() const { return x * x + y * y; }
  double ang() const { return atan2(y, x); }
};
vector<Point> convex;
int n;
double radius;
Point points[N][2];
Point target;
double det(Point a, Point b, Point c) { return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) *
     (b.y - a.y); }
double dot(Point a, Point b, Point c) { return (b.x - a.x) * (c.x - a.x) + (b.y - a.y) *
     (c.y - a.y); }
double det(Point a, Point b) { return a.x * b.y - b.x * a.y; }
double dot(Point a, Point b) { return a.x * b.x + a.y * b.y; }
inline bool point_on_line(const Point &a, const Point &b, const Point &c) {
  return sign(det(Point(0, 0), a - b, c - b)) == 0 && dot(Point(0, 0), b - a, c - a) <
       eps;
}
double point_to_line(const Point &a, const Point &b, const Point &c) {
  return fabs(det(Point(0, 0), c - b, a - b)) / (b - c).dist();
}
Point project_to_line(const Point &p, const Point &a, const Point &b) {
  return a + (b - a) * dot(Point(0, 0), p - a, b - a) / sqr((b - a).dist());
}
Point intersect(Point a, Point b, Point c, Point d) {
  double s1 = det(a, b, c); double s2 = det(a, b, d); return (c * s2 - d * s1) / (s2 - s1
       );
}
inline Point line_to_circle(const Point &a, const Point &b) {
  double x = sqrt(sqr(radius) - sqr(point_to_line(Point(0, 0), a, b)));
  return project_to_line(Point(0, 0), a, b) - (b - a) / (b - a).dist() * x;
}
inline double area_tri(Point a, Point b) { return det(Point(0, 0), a, b) / 2; }
inline double area_cir(Point a, Point b, double radius) {
  if (sign(det(Point(0, 0), a, b)) == 0) return 0;
  a = a / a.dist() * radius; b = b / b.dist() * radius;
  double d = atan2(det(Point(0, 0), a, b), dot(Point(0, 0), a, b));
  return sqr(radius) * d / 2;
}
int intersect(const Point &a, const Point &b, Point &u, Point &v, double radius) {
  if (point_to_line(Point(0, 0), a, b) + eps > radius) return 0;
  u = line_to_circle(a, b); v = line_to_circle(b, a);
  return point_on_line(u, a, b) + point_on_line(v, a, b);
}
vector<Point> calc(vector<Point> vec, Point a, Point b) {
  vector<Point> result;
  for(int i = 0; i < (int)vec.size(); i++) {
    Point c = vec[i], d = vec[(i + 1) % (int)vec.size()];
    if (det(a, b, c) > -eps)
      result.push_back(c);
    if (sign(det(a, b, c)) * sign(det(a, b, d)) == -1)
      result.push_back(intersect(a, b, c, d));
  } return result;
}
double areaCT(double R, Point pa, Point pb) {
  if (pa.dist() < pb.dist()) swap(pa, pb);
  if (pb.dist() < eps) return 0;
  Point pc = pb - pa;
  double a = pb.dist(), b = pa.dist(), c = pc.dist();
  double cosB = dot(pb, pc) / a / c, B = acos(cosB);
  double cosC = dot(pa, pb) / a / b, C = acos(cosC);
  double S, h, theta;
  if (a > R) {
    S = C * 0.5 * R * R;
    h = a * b * sin(C) / c;
    if (h < R && B < pi * 0.5)
      S -= acos(h / R) * R * R - h * sqrt(max(0.0, R * R - h * h));
  } else if (b > R) {
    theta = pi - B - asin(sin(B) / R * a);
    S = 0.5 * a * R * sin(theta) + (C - theta) * 0.5 * R * R;
  } else S = 0.5 * sin(C) * a * b;
  return S;
}
void solve() {
  scanf("%lf%d", &radius, &n);
  convex.clear();
  convex.push_back(Point(-radius, -radius));
  convex.push_back(Point(radius, -radius));
  convex.push_back(Point(radius, radius));
  convex.push_back(Point(-radius, radius));
  for(int i = 1; i <= n; i++)
    scanf("%lf%lf%lf%lf", &points[i][0].x, &points[i][0].y, &points[i][1].x, &points[i
         ][1].y);
  scanf("%lf %lf", &target.x, &target.y);
  for(int i = 1; i <= n; i++) {
    if (det(points[i][0], points[i][1], target) < -eps)
      swap(points[i][0], points[i][1]);
    convex = calc(convex, points[i][0], points[i][1]);
  }
  double ans = 0;
  for(int i = 0; i < (int)convex.size(); i++)
    ans += areaCT(radius, convex[i], convex[(i + 1) % (int)convex.size()]) * sign(det(
         convex[i], convex[(i + 1) % (int)convex.size()]));
  printf("%.5f", max(0., fabs(ans) / (pi * radius * radius) * 100));
  puts("%");
}
```

11 动态凸包

```
#define x first
#define y second
typedef map<int, int> mii;
typedef map<int, int>::iterator mit;
struct point { // something omitted
  point(const mit &p): x(p->first), y(p->second) {}
};
inline bool checkInside(mii &a, const point &p) { // border inclusive
  int x = p.x, y = p.y;
  mit p1 = a.lower_bound(x);
  if (p1 == a.end()) return false;
  if (p1->x == x) return y <= p1->y;
  if (p1 == a.begin()) return false;
  mit p2(p1--);
  return sign(det(p - point(p1), point(p2) - p)) >= 0;
}
inline void addPoint(mii &a, const point &p) { // no collinear points
  int x = p.x, y = p.y;
  mit pnt = a.insert(make_pair(x, y)).first, p1, p2;
  for (pnt->y = y; ; a.erase(p2)) {
    p1 = pnt;
    if (++p1 == a.end())
      break;
    p2 = p1;
    if (++p1 == a.end())
```

```
      break;
    if (det(point(p2) - p, point(p1) - p) < 0)
      break;
  }
  for ( ; ; a.erase(p2)) {
    if ((p1 = pnt) == a.begin())
      break;
    if (--p1 == a.begin())
      break;
    p2 = p1--;
    if (det(point(p2) - p, point(p1) - p) > 0)
      break;
  }
}
```

‘upperHull $\leftarrow (x, y)$‘ ‘lowerHull $\leftarrow (x, -y)$‘

12 farmland

```

const int N = 11111, M = 111111 * 4;

struct eglist {
  int other[M], succ[M], last[M], sum;
  void clear() {
    memset(last, -1, sizeof(last));
    sum = 0;
  }
  void addEdge(int a, int b) {
    other[sum] = b, succ[sum] = last[a], last[a] = sum++;
    other[sum] = a, succ[sum] = last[b], last[b] = sum++;
  }
}e;

int n, m;
struct point {
  int x, y;
  point(int x, int y) : x(x), y(y) {}
  point() {}
  friend point operator -(point a, point b) {
    return point(a.x - b.x, a.y - b.y);
  }
  double arg() {
    return atan2(y, x);
  }
}points[N];

vector<pair<int, double> > vecs;
vector<int> ee[M];
vector<pair<double, pair<int, int> > > edges;
double length[M];
int tot, father[M], next[M], visit[M];

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

long long det(point a, point b) {
  return 1LL * a.x * b.y - 1LL * b.x * a.y;
}

double dist(point a, point b) {
  return sqrt(1.0 * (a.x - b.x) * (a.x - b.x) + 1.0 * (a.y - b.y) * (a.y - b.y));
}

int main() {
  scanf("%d␣%d", &n, &m);
  e.clear();
  for(int i = 1; i <= n; i++) {
    scanf("%d␣%d", &points[i].x, &points[i].y);
  }
  for(int i = 1; i <= m; i++) {
    int a, b;
    scanf("%d␣%d", &a, &b);
    e.addEdge(a, b);
  }
  for(int x = 1; x <= n; x++) {
    vector<pair<double, int> > pairs;
    for(int i = e.last[x]; ~i; i = e.succ[i]) {
      int y = e.other[i];
      pairs.push_back(make_pair((points[y] - points[x]).arg(), i));
    }
    sort(pairs.begin(), pairs.end());
    for(int i = 0; i < (int)pairs.size(); i++) {
      next[pairs[(i + 1) % (int)pairs.size()].second ^ 1] = pairs[i].second;
    }
  }
  memset(visit, 0, sizeof(visit));
  tot = 0;
  for(int start = 0; start < e.sum; start++) {
    if (visit[start])
      continue;
    long long total = 0;
    int now = start;
    vecs.clear();
    while(!visit[now]) {
      visit[now] = 1;
      total += det(points[e.other[now ^ 1]], points[e.other[now]]);
      vecs.push_back(make_pair(now / 2, dist(points[e.other[now ^ 1]], points[e.other[now
          ]])));
      now = next[now];
    }
    if (now == start && total > 0) {
      ++tot;
      for(int i = 0; i < (int)vecs.size(); i++) {
        ee[vecs[i].first].push_back(tot);
      }
    }
  }

  for(int i = 0; i < e.sum / 2; i++) {
    int a = 0, b = 0;
    if (ee[i].size() == 0)
      continue;
    else if (ee[i].size() == 1) {
      a = ee[i][0];
    } else if (ee[i].size() == 2) {
      a = ee[i][0], b = ee[i][1];
    }
    edges.push_back(make_pair(dist(points[e.other[i * 2]], points[e.other[i * 2 + 1]]),
        make_pair(a, b)));
  }
  sort(edges.begin(), edges.end());
  for(int i = 0; i <= tot; i++)
    father[i] = i;
  double ans = 0;
  for(int i = 0; i < (int)edges.size(); i++) {
    int a = edges[i].second.first, b = edges[i].second.second;
    double v = edges[i].first;
    if (find(a) != find(b)) {
      ans += v;
      father[father[a]] = father[b];
    }
  }
  printf("%.5f\n", ans);
}

scanf("%lf␣%lf␣%d", &W, &H, &n);
for (int i = 0; i < n; i++) {
  scanf("%lf␣%lf␣%lf␣%lf", &segments[i][0].x, &segments[i][0].y, &segments[i][1].x, &
      segments[i][1].y);
}
addSegment(Point(0, 0), Point(W, 0));
addSegment(Point(W, 0), Point(W, H));
addSegment(Point(W, H), Point(0, H));
addSegment(Point(0, H), Point(0, 0));

```

```
for (int i = 0; i < n; i++) {
  Points.push_back(segments[i][0]);
  Points.push_back(segments[i][1]);
  for (int j = 0; j < i; j++) {
    if (!parallel(segments[i][0], segments[i][1], segments[j][0], segments[j][1])) {
      Point p = intersect(segments[i][0], segments[i][1], segments[j][0], segments[j][1])
          ;
      if (p.on(segments[i][0], segments[i][1]) && p.on(segments[j][0], segments[j][1])) {
        Points.push_back(p);
      }
    }
  }
}
sort(Points.begin(), Points.end());
Points.erase(unique(Points.begin(), Points.end()), Points.end());

e.clear();
for (int i = 0; i < n; i++) {
  vector<pair<double, int> > pairs;
  for (int j = 0; j < Points.size(); j++) {
    if (Points[j].on(segments[i][0], segments[i][1]))
      pairs.push_back(make_pair((Points[j] - segments[i][0]).norm(), j));
  }
  sort(pairs.begin(), pairs.end());
  for (int i = 1; i < pairs.size(); i++) {
    e.addEdge(pairs[i - 1].second, pairs[i].second);
    e.addEdge(pairs[i].second, pairs[i - 1].second);
  }
}
```

13 半平面交

```
const double EPS = 1e-9;
int sign(const double x, const double eps = EPS) { return x < -eps ? -1 : x > eps; }
struct Point {
  double x, y;
  Point (const double &x = 0, const double &y = 0) : x(x), y(y) {}
  void in() { scanf("%lf%lf", &x, &y); }
  double len2() { return x * x + y * y; }
  double len() { return sqrt(x * x + y * y); }
  Point turn90() { return Point(-y, x); }
  Point norm() { double l = len(); return Point(x / l, y / l); }
  int quad() const { return sign(y) == 1 || sign(y) == 0 && sign(x) >= 0; }
};
Point operator + (const Point &a, const Point &b) { return Point(a.x + b.x, a.y + b.y); }
Point operator - (const Point &a, const Point &b) { return Point(a.x - b.x, a.y - b.y); }
Point operator * (const Point &a, const double &k) { return Point(a.x * k, a.y * k); }
Point operator / (const Point &a, const double &k) { return Point(a.x / k, a.y / k); }
double dot(const Point &a, const Point &b) { return a.x * b.x + a.y * b.y; }
double det(const Point &a, const Point &b) { return a.x * b.y - a.y * b.x; }
struct Line {
  Point a, b;
  void in() { a.in(), b.in(); }
  Line(const Point a = Point(0, 0), const Point b = Point(0, 0)) : a(a), b(b) {}
  bool include(const Point &p) const { return sign(det(b - a, p - a)) > 0; }
  Line push() {
    const double eps = 1e-6;
    Point delta = (b - a).turn90().norm() * eps;
    return Line(a - delta, b - delta);
  }
};
bool parallel(const Line &l0, const Line &l1) {
  return sign(det(l0.b - l0.a, l1.b - l1.a)) == 0;
}
bool sameDir(const Line &l0, const Line &l1) {
  return parallel(l0, l1) && sign(dot(l0.b - l0.a, l1.b - l1.a)) == 1;
}
Point intersect(const Line &l0, const Line &l1) {
  double s1 = det(l0.b - l0.a, l1.a - l0.a), s2 = det(l0.b - l0.a, l1.b - l0.a);
  return (l1.a * s2 - l1.b * s1) / (s2 - s1);
}
```

```
bool operator < (const Point &a, const Point &b) {
  if (a.quad() != b.quad()) return a.quad() < b.quad(); else return sign(det(a, b)) > 0;
}
bool operator < (const Line &l0, const Line &l1) {
  if (sameDir(l0, l1)) return l1.include(l0.a); else return (l0.b - l0.a) < (l1.b - l1.a)
      ;
}
bool check(const Line &u, const Line &v, const Line &w) { return w.include(intersect(u, v
    )); }
vector<Point> intersection(vector<Line> &l) {
  sort(l.begin(), l.end());
  deque<Line> q;
  for (int i = 0; i < (int)l.size(); ++i) {
    if (i && sameDir(l[i], l[i - 1])) continue;
    while (q.size() > 1 && !check(q[q.size() - 2], q[q.size() - 1], l[i])) q.pop_back();
    while (q.size() > 1 && !check(q[1], q[0], l[i])) q.pop_front();
    q.push_back(l[i]);
  }
  while (q.size() > 2 && !check(q[q.size() - 2], q[q.size() - 1], q[0])) q.pop_back();
  while (q.size() > 2 && !check(q[1], q[0], q[q.size() - 1])) q.pop_front();
  vector<Point> ret;
  for (int i = 0; i < (int)q.size(); ++i) {
    ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
  } return ret;
}
```

14 三维绕轴旋转

```
const double pi = acos(-1.0);
int n, m; char ch1; bool flag;
double a[4][4], s1, s2, x, y, z, w, b[4][4], c[4][4];
double sqr(double x)
{
  return x*x;
}
int main()
{
  scanf("%d\n", &n);
  memset(b, 0, sizeof(b));
  b[0][0] = b[1][1] = b[2][2] = b[3][3] = 1;//initial matrix
  for(int i = 1; i <= n; i++)
  {
    scanf("%c", &ch1);
    if(ch1 == 'T')
    {
      scanf("%lf␣%lf␣%lf\n", &x, &y, &z);//plus each coordinate by a number (x, y, z)
      memset(a, 0, sizeof(a));
      a[0][0] = 1; a[3][0] = x;
      a[1][1] = 1; a[3][1] = y;
      a[2][2] = 1; a[3][2] = z;
      a[3][3] = 1;
    }else if(ch1 == 'S')
    {
      scanf("%lf␣%lf␣%lf\n", &x, &y, &z);//multiply each coordinate by a number (x, y, z)
      memset(a, 0, sizeof(a));
      a[0][0] = x;
      a[1][1] = y;
      a[2][2] = z;
      a[3][3] = 1;
    }else
    {
      scanf("%lf␣%lf␣%lf␣%lf\n", &x, &y, &z, &w);
      //大拇指指向x轴正方向时，4指弯曲由y轴正方向指向z轴正方向
      //大拇指沿着原点到点(x, y, z)的向量，4指弯曲方向旋转w度
      w = w*pi/180;
      memset(a, 0, sizeof(a));
      s1 = x*x+y*y+z*z;
      a[3][3] = 1;
      a[0][0] = ((y*y+z*z)*cos(w)+x*x)/s1;      a[0][1] = x*y*(1-cos(w))/s1+z*sin(w)/sqrt
          (s1);  a[0][2] = x*z*(1-cos(w))/s1-y*sin(w)/sqrt(s1);
```

```
      a[1][0] = x*y*(1-cos(w))/s1-z*sin(w)/sqrt(s1);  a[1][1] = ((x*x+z*z)*cos(w)+y*y)/s1
           ;      a[1][2] = y*z*(1-cos(w))/s1+x*sin(w)/sqrt(s1);
      a[2][0] = x*z*(1-cos(w))/s1+y*sin(w)/sqrt(s1);  a[2][1] = y*z*(1-cos(w))/s1-x*sin(w
           )/sqrt(s1);  a[2][2] = ((x*x+y*y)*cos(w)+z*z)/s1;
    }
    memset(c, 0, sizeof(c));
    for(int i = 0; i < 4; i++)
      for(int j = 0; j < 4; j++)
        for(int k = 0; k < 4; k++)
          c[i][j] += b[i][k]*a[k][j];
    memcpy(b, c, sizeof(c));
  }
  scanf("%d", &m);
  for(int i = 1; i <= m; i++)
  {
    scanf("%lf%lf%lf", &x, &y, &z);//initial vector
    printf("%lf␣%lf␣%lf\n", x*b[0][0]+y*b[1][0]+z*b[2][0]+b[3][0], x*b[0][1]+y*b[1][1]+z*
         b[2][1]+b[3][1], x*b[0][2]+y*b[1][2]+z*b[2][2]+b[3][2]);
  }
  return 0;
}
```

15 $O(n \log k)$ 判断圆存在交集

传入 n 个圆，圆心存在 cir 中，半径存在 radius 中，nlogk 判断是否存在交集

```
int n;
double sx, sy, d;
vector<Point> cir;
vector<double> radius;

int isIntersectCircleToCircle(Point c1, double r1, Point c2, double r2)
{
  double dis = c1.distTo(c2);
  return sign(dis - (r1 + r2)) <= 0;
}

void getRange(double x, Point &c, double r, double &retl, double &retr)
{
  double tmp = sqrt(max(r * r - (c.x - x) * (c.x - x), 0.0));
  retl = c.y - tmp; retr = c.y + tmp;
}

int checkInLine(double x)
{
  double minR = INF, maxL = -INF;
  double tmpl, tmpr;
  for(int i = 0; i < n; ++ i) {
    if (sign(cir[i].x + radius[i] - x) < 0 || sign(cir[i].x - radius[i] - x) > 0)
      return false;
    getRange(x, cir[i], radius[i], tmpl, tmpr);
    maxL = max(tmpl, maxL);
    minR = min(tmpr, minR);
    if (maxL > minR) return false;
  }
  return true;
}

int shouldGoLeft(double x)
{
  if (checkInLine(x)) return 2;
  int onL = 0, onR = 0;
  for(int i = 0; i < n; ++ i) {
    if (sign(cir[i].x + radius[i] - x) < 0) onL = 1;
    if (sign(cir[i].x - radius[i] - x) > 0) onR = 1;
  }
  if (onL && onR) return -1;
  if (onL) return 1;
  if (onR) return 0;

  double minR = INF, maxL = -INF, tmpl, tmpr;
  int idMinR, idMaxL;

  for(int i = 0; i < n; ++ i) {
    getRange(x, cir[i], radius[i], tmpl, tmpr);
    if (tmpr < minR) {
      minR = tmpr;
      idMinR = i;
    }
    if (tmpl > maxL) {
      maxL = tmpl;
      idMaxL = i;
    }
  }
  if (! isIntersectCircleToCircle(cir[idMinR], radius[idMinR], cir[idMaxL], radius[idMaxL
       ]))
    return -1;
  Point p1, p2;
  intersectionCircleToCircle(cir[idMinR], radius[idMinR], cir[idMaxL], radius[idMaxL], p1
       , p2);
  return (p1.x < x);
}

int hasIntersectionCircles()
{
  double l = -INF, r = INF, mid;
  for(int i = 0; i < 100; ++ i) {
    mid = (l + r) * 0.5;
    int tmp = shouldGoLeft(mid);
    if (tmp < 0) return 0;
    if (tmp == 2) return 1;
    if (tmp) r = mid;
    else l = mid;
  }
  mid = (l + r) * 0.5;
  return checkInLine(mid);
}
```

16 $O(n^2 \log n)$ 圆交 + 计算面积和重心

```
double pi = acos(-1.0), eps = 1e-12;
double sqr(const double & x) {
  return x * x;
}
double ans[2001];
int sign(const double & x) {
  return x < -eps?-1:x > eps;
}
struct Point {
  double x, y;
  Point(){}
  Point(const double & x, const double & y) : x(x), y(y) {}
  void scan() {scanf("%lf%lf", &x, &y);}
  double sqrlen() {return sqr(x) + sqr(y);}
  double len() {return sqrt(sqrlen());}
  Point rev() {return Point(y, -x);}
  void print() {printf("%f␣%f\n", x, y);}
  Point zoom(const double & d) {double lambda = d / len(); return Point(lambda * x,
       lambda * y);}
} dvd, a[2001];
Point centre[2001];
double atan2(const Point & x) {
  return atan2(x.y, x.x);
}
Point operator - (const Point & a, const Point & b) {
  return Point(a.x - b.x, a.y - b.y);
}
Point operator + (const Point & a, const Point & b) {
  return Point(a.x + b.x, a.y + b.y);
}
double operator * (const Point & a, const Point & b) {
  return a.x * b.y - a.y * b.x;
}
```

```
Point operator * (const double & a, const Point & b) {
  return Point(a * b.x, a * b.y);
}
double operator % (const Point & a, const Point & b) {
  return a.x * b.x + a.y * b.y;
}
struct circle {
  double r; Point o;
  circle() {}
  void scan() {
    o.scan();
    scanf("%lf", &r);
  }
} cir[2001];
struct arc {
  double theta;
  int delta;
  Point p;
  arc() {};
  arc(const double & theta, const Point & p, int d) : theta(theta), p(p), delta(d) {}
} vec[4444];
int nV;
inline bool operator < (const arc & a, const arc & b) {
  return a.theta + eps < b.theta;
}
int cnt;
inline void psh(const double t1, const Point p1, const double t2, const Point p2) {
  if(t2 + eps < t1)
    cnt++;
  vec[nV++] = arc(t1, p1, 1);
  vec[nV++] = arc(t2, p2, -1);
}
inline double cub(const double & x) {
  return x * x * x;
}
inline void combine(int d, const double & area, const Point & o) {
  if(sign(area) == 0) return;
  centre[d] = 1 / (ans[d] + area) * (ans[d] * centre[d] + area * o);
  ans[d] += area;
}
bool equal(const double & x, const double & y) {
  return x + eps>  y and y + eps > x;
}
bool equal(const Point & a, const Point & b) {
  return equal(a.x, b.x) and equal(a.y, b.y);
}
bool equal(const circle & a, const circle & b) {
  return equal(a.o, b.o) and equal(a.r, b.r);
}
bool f[2001];
int main() {
  //freopen("hdu4895.in", "r", stdin);
  int n, m, index;
  while(EOF != scanf("%d%d%d", &m, &n, &index)) {
    index--;
    for(int i(0); i < m; i++) {
      a[i].scan();
    }
    for(int i(0); i < n; i++) {
      cir[i].scan();//n个圆
    }
    for(int i(0); i < n; i++) {//这一段在去重圆 能加速 删掉不会错
      f[i] = true;
      for(int j(0); j < n; j++) if(i != j) {
        if(equal(cir[i], cir[j]) and i < j or !equal(cir[i], cir[j]) and cir[i].r < cir[j
            ].r + eps and (cir[i].o - cir[j].o).sqrlen() < sqr(cir[i].r - cir[j].r) +
            eps) {
          f[i] = false;
          break;
        }
      }
    }
    int n1(0);
    for(int i(0); i < n; i++)
      if(f[i])
        cir[n1++] = cir[i];
    n = n1;//去重圆结束
    fill(ans, ans + n + 1, 0);//ans[i]表示被圆覆盖至少i次的面积
    fill(centre, centre + n + 1, Point(0, 0));//centre[i]表示上面ans[i]部分的重心
    for(int i(0); i < m; i++)
      combine(0, a[i] * a[(i + 1) % m] * 0.5, 1. / 3 * (a[i] + a[(i + 1) % m]));
    for(int i(0); i < n; i++) {
      dvd = cir[i].o - Point(cir[i].r, 0);
      nV = 0;
      vec[nV++] = arc(-pi, dvd, 1);
      cnt = 0;
      for(int j(0); j < n; j++) if(j != i) {
        double d = (cir[j].o - cir[i].o).sqrlen();
        if(d < sqr(cir[j].r - cir[i].r) + eps) {
          if(cir[i].r + i * eps < cir[j].r + j * eps)
            psh(-pi, dvd, pi, dvd);
        }else if(d + eps < sqr(cir[j].r + cir[i].r)) {
          double lambda = 0.5 * (1 + (sqr(cir[i].r) - sqr(cir[j].r)) / d);
          Point cp(cir[i].o + lambda * (cir[j].o - cir[i].o));
          Point nor((cir[j].o - cir[i].o).rev().zoom(sqrt(sqr(cir[i].r) - (cp - cir[i].o)
              .sqrlen())));
          Point frm(cp + nor);
          Point to(cp - nor);
          psh(atan2(frm - cir[i].o), frm, atan2(to - cir[i].o), to);
        }
      }
      sort(vec + 1, vec + nV);
      vec[nV++] = arc(pi, dvd, -1);
      for(int j = 0; j + 1 < nV; j++) {
        cnt += vec[j].delta;
        //if(cnt == 1) {//如果只算ans[1]和centre[1], 可以加这个if加速.
          double theta(vec[j + 1].theta - vec[j].theta);
          double area(sqr(cir[i].r) * theta * 0.5);
          combine(cnt, area, cir[i].o + 1. / area / 3 * cub(cir[i].r) * Point(sin(vec[j +
              1].theta) - sin(vec[j].theta), cos(vec[j].theta) - cos(vec[j + 1].theta))
              );
          combine(cnt, -sqr(cir[i].r) * sin(theta) * 0.5, 1. / 3 * (cir[i].o + vec[j].p +
              vec[j + 1].p));
          combine(cnt, vec[j].p * vec[j + 1].p * 0.5, 1. / 3 * (vec[j].p + vec[j + 1].p))
              ;
        //}
      }
    }//板子部分结束 下面是题目
    combine(0, -ans[1], centre[1]);
    for(int i = 0; i < m; i++) {
      if(i != index)
        (a[index] - Point((a[i] - a[index]) * (centre[0] - a[index]), (a[i] - a[index]) %
            (centre[0] - a[index])).zoom((a[i] - a[index]).len())).print();
      else
        a[i].print();
    }

  }
  fclose(stdin);
  return 0;
}
```

17 三维凸包

```
const double eps = 1e-8;
int mark[1005][1005];
Point info[1005];
int n, cnt;
double mix(const Point &a, const Point &b, const Point &c) {
  return a.dot(b.cross(c));}
double area(int a, int b, int c) {
  return ((info[b] - info[a]).cross(info[c] - info[a])).length();}
double volume(int a, int b, int c, int d) {
  return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);}
struct Face {
  int a, b, c;
```

```
  Face() {}
  Face(int a, int b, int c): a(a), b(b), c(c) {}
  int &operator [](int k) { return k==0?a:k==1?b:c; }
};
vector <Face> face;
inline void insert(int a, int b, int c) { face.push_back(Face(a, b, c));}
void add(int v) {
  vector <Face> tmp;
  int a, b, c;
  cnt++;
  for (int i = 0; i < SIZE(face); i++) {
    a = face[i][0];  b = face[i][1];  c = face[i][2];
    if (Sign(volume(v, a, b, c)) < 0)
      mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] =
          mark[a][c] = cnt;
    else tmp.push_back(face[i]);
  }
  face = tmp;
  for (int i = 0; i < SIZE(tmp); i++) {
    a = face[i][0]; b = face[i][1]; c = face[i][2];
    if (mark[a][b] == cnt) insert(b, a, v);
    if (mark[b][c] == cnt) insert(c, b, v);
    if (mark[c][a] == cnt) insert(a, c, v);
  }
}
int Find() {
  for (int i = 2; i < n; i++) {
    Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
    if (ndir == Point()) continue;
    swap(info[i], info[2]);
    for (int j = i + 1; j < n; j++)
      if (Sign(volume(0, 1, 2, j)) != 0) {
        swap(info[j], info[3]);
        insert(0, 1, 2);  insert(0, 2, 1);
        return 1;
      }
  }
  return 0;
}
int main() {
  for (; scanf("%d", &n) == 1; ) {
    for (int i = 0; i < n; i++)
      info[i].Input();
    sort(info, info + n);
    n = unique(info, info + n) - info;
    face.clear();
    random_shuffle(info, info + n);
    if (Find()) {
      memset(mark, 0, sizeof(mark));
      cnt = 0;
      for (int i = 3; i < n; i++) add(i);
      vector<Point> Ndir;
      for (int i = 0; i < SIZE(face); ++i) {
        Point p = (info[face[i][0]] - info[face[i][1]]).cross
            (info[face[i][2]] - info[face[i][1]]);
        p = p / p.length();
        Ndir.push_back(p);
      }
      sort(Ndir.begin(), Ndir.end());
      int ans = unique(Ndir.begin(), Ndir.end()) - Ndir.begin();
      printf("%d\n", ans);
    } else {
      printf("1\n");
    }
  }
}
```

18 点在多边形内

```
bool in_polygon(const point &p, const vector<point> &poly) {
  int n = (int)poly.size();
  int counter = 0;
  for (int i = 0; i < n; ++i) {
    point a = poly[i], b = poly[(i + 1) % n];
    if (point_on_line(p, line(a, b)))
      return false; // bounded excluded
    int x = sign(det(p - a, b - a));
    int y = sign(a.y - p.y);
    int z = sign(b.y - p.y);
    if (x > 0 && y <= 0 && z > 0)
      counter++;
    if (x < 0 && z <= 0 && y > 0)
      counter--;
  }
  return counter != 0;
}
```

19 KD 树

```
曼哈顿距离版，欧几里得只需要把sqr改成x*x即可。

int const N = ;
struct Point { int x, y, id; };
inline long long sqr(const long long &x) { return abs(x); }
inline long long dist(const Point &a, const Point &b) { return sqr(a.x - b.x) + sqr(a.y -
      b.y); }
struct Rectangle {
  int lx, rx, ly, ry;
  inline void set(const Point &p) { lx = rx = p.x; ly = ry = p.y; }
  inline void mergy(const Point &p) {
    lx = min(lx, p.x); rx = max(rx, p.x);
    ly = min(ly, p.y); ry = max(ry, p.y);
  }
  inline void mergy(const Rectangle &r) {
    lx = min(lx, r.lx); rx = max(rx, r.rx);
    ly = min(ly, r.ly); ry = max(ry, r.ry);
  }
  /*  minimum distance  */
  inline long long dist(const Point &p) {
    if (p.x <= lx && p.y <= ly) return sqr(p.x - lx) + sqr(p.y - ly);
    if (p.x <= rx && p.y <= ly) return sqr(p.y - ly);
    if (p.x >= rx && p.y <= ly) return sqr(p.x - rx) + sqr(p.y - ly);
    if (p.x >= rx && p.y <= ry) return sqr(p.x - rx);
    if (p.x >= rx && p.y >= ry) return sqr(p.x - rx) + sqr(p.y - ry);
    if (p.x >= lx && p.y >= ry) return sqr(p.y - ry);
    if (p.x <= lx && p.y >= ry) return sqr(p.x - lx) + sqr(p.y - ry);
    if (p.x <= lx && p.y >= ly) return sqr(p.x - lx);
    return 0;
  }
  /*  maximum distance  */
  inline long long dist(const Point &p) {
    long long ret = 0;
    ret += max(sqr(rx - p.x), sqr(lx - p.x));
    ret += max(sqr(ry - p.y), sqr(ly - p.y));
    return ret;
  }
};
struct Node {
  int child[2]; Point p; Rectangle r;
  inline void set(const Point &_p) {
    p = _p; r.set(p); child[0] = child[1] = 0;
  }
};
int size;
Point a[N];
Node tree[N];
inline bool xcompare(const Point &a, const Point &b) {
  return a.x < b.x || a.x == b.x && a.y < b.y;
}
inline bool ycompare(const Point &a, const Point &b) {
  return a.y < b.y || a.y == b.y && a.x < b.x;
}
```

```
inline int build(int left, int right, bool dim = 0) {
  int x = ++size, mid = left + right >> 1;
  nth_element(a + left, a + mid, a + right, dim ? xcompare : ycompare);
  tree[x].set(a[mid]);
  if (left < mid) {
    tree[x].child[0] = build(left, mid, dim ^ 1);
    tree[x].r.mergy(tree[tree[x].child[0]].r);
  }
  if (mid + 1 < right) {
    tree[x].child[1] = build(mid + 1, right, dim ^ 1);
    tree[x].r.mergy(tree[tree[x].child[1]].r);
  } return x;
}
inline int insert(int x, const Point &p, bool dim = 0) {
  if (x == 0) { tree[++size].set(p); return size; }
  tree[x].r.mergy(p);
  if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p))
    tree[x].child[0] = insert(tree[x].child[0], p, dim ^ 1);
  else tree[x].child[1] = insert(tree[x].child[1], p, dim ^ 1);
  return x;
}
/*  query minimum */
inline void query(int x, const Point &p, long long &ret, bool dim = 0) {
  if (tree[x].r.dist(p) >= ret) return;
  ret = min(ret, dist(tree[x].p, p));
  if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p)) {
    if (tree[x].child[0]) query(tree[x].child[0], p, ret, dim ^ 1);
    if (tree[x].child[1]) query(tree[x].child[1], p, ret, dim ^ 1);
  } else {
    if (tree[x].child[1]) query(tree[x].child[1], p, ret, dim ^ 1);
    if (tree[x].child[0]) query(tree[x].child[0], p, ret, dim ^ 1);
  }
}
/*  query maximum */
inline void query(int x, const Point &p, long long &ret, bool dim = 0) {
  if (tree[x].r.dist(p) <= ret) { return; }
  ret = max(ret, dist(tree[x].p, p));
  if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p)) {
    if (tree[x].child[1]) query(tree[x].child[1], p, ret, dim ^ 1);
    if (tree[x].child[0]) query(tree[x].child[0], p, ret, dim ^ 1);
  } else {
    if (tree[x].child[0]) query(tree[x].child[0], p, ret, dim ^ 1);
    if (tree[x].child[1]) query(tree[x].child[1], p, ret, dim ^ 1);
  }
}
/*  query kth-minimum */
inline void query(int x, const Point &p, int k, pair<long long, int> ret[], bool dim = 0)
    {
  if (tree[x].r.dist(p) > ret[k].first) return;
  pair<long long, int> val = make_pair(dist(tree[x].p, p), tree[x].p.id);
  for (int i = 1; i <= k; ++i) if (val < ret[i]) {
    for (int j = k + 1; j > i; --j) ret[j] = ret[j - 1];
    ret[i] = val; break;
  }
  if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p)) {
    if (tree[x].child[0]) query(tree[x].child[0], p, k, ret, dim ^ 1);
    if (tree[x].child[1]) query(tree[x].child[1], p, k, ret, dim ^ 1);
  } else {
    if (tree[x].child[1]) query(tree[x].child[1], p, k, ret, dim ^ 1);
    if (tree[x].child[0]) query(tree[x].child[0], p, k, ret, dim ^ 1);
  }
}
/*  query kth-maximum */
inline void query(int x, const Point &p, int k, pair<long long, int> ret[], bool dim = 0)
    {
  if (tree[x].r.dist(p) < ret[k].first) return;
  pair<long long, int> val = make_pair(dist(tree[x].p, p), -tree[x].p.id);
  for (int i = 1; i <= k; ++i) if (val > ret[i]) {
    for (int j = k + 1; j > i; --j) ret[j] = ret[j - 1];
    ret[i] = val; break;
  }
  if (dim && xcompare(p, tree[x].p) || !dim && ycompare(p, tree[x].p)) {
    if (tree[x].child[1]) query(tree[x].child[1], p, k, ret, dim ^ 1);
    if (tree[x].child[0]) query(tree[x].child[0], p, k, ret, dim ^ 1);
  } else {
    if (tree[x].child[0]) query(tree[x].child[0], p, k, ret, dim ^ 1);
    if (tree[x].child[1]) query(tree[x].child[1], p, k, ret, dim ^ 1);
  }
}
inline void clear() { size = 0; }
```

20 树链剖分

```
int const N = ;
int n;
vector<int> adj[N];
int father[N], height[N], size[N], son[N], top[N], idx[N], num[N];
inline void prepare() {
  vector<int> queue; father[1] = height[1] = 0; queue.push_back(1);
  for (int head = 0; head < (int)queue.size(); ++head) {
    int x = queue[head];
    for (int i = 0; i < (int)adj[x].size(); ++i) {
      int y = adj[x][i];
      if (y != father[x])
        father[y] = x, height[y] = height[x] + 1, queue.push_back(y);
    }
  }
  for (int i = n - 1; i >= 0; --i) {
    int x = queue[i]; size[x] = 1; son[x] = -1;
    for (int j = 0; j < (int)adj[x].size(); ++j) {
      int y = adj[x][j];
      if (y != father[x]) {
        size[x] += size[y];
        if (son[x] == -1 || size[son[x]] < size[y]) son[x] = y;
      }
    }
  } int tot = 0; fill(top + 1, top + n + 1, 0);
  for (int i = 0; i < n; ++i) {
    int x = queue[i];
    if (top[x] == 0)
      for (int y = x; y != -1; y = son[y])
        top[y] = x, idx[y] = ++tot, num[tot] = //data[y];
  } build(1, 1, n);
}
inline void handle(int x, int y) {
  for (; true; ) {
    if (top[x] == top[y]) {
      if (x == y) handle(1, 1, n, idx[x], idx[x]);
      else {
        if (height[x] < height[y]) handle(1, 1, n, idx[x], idx[y]);
        else handle(1, 1, n, idx[y], idx[x]);
      } break;
    }
    if (height[top[x]] > height[top[y]])
      handle(1, 1, n, idx[top[x]], idx[x]), x = father[top[x]];
    else handle(1, 1, n, idx[top[y]], idx[y]), y = father[top[y]];
  }
}
```

21 可持久化左偏树

```
Node * persiMerge(Node * a, Node * b) {
  if(!a) return b;
  if(!b) return a;
  Node * res;
  if(a->v < b->v) {
    res = new Node(*a);
    res->s[1] = persiMerge(b, res->s[1]);
  }else {
    res = new Node(*b);
    res->s[1] = persiMerge(a, res->s[1]);
  }
```

```
  if(!res->s[0] or res->s[1] and res->s[0]->l < res->s[1]->l)
    swap(res->s[0], res->s[1]);
  res->l = res->s[1]?res->s[1]->l + 1:0;
  return res;
}
```

22 Treap

```
struct Node {
  Node *child[2]; int key; int size, count, aux;
  inline Node(int _aux) {
    child[0] = child[1] = 0; key = size = count = 0; aux = _aux;
  }
  inline void update() { size = count + child[0]->size + child[1]->size; }
};
Node *null;
inline void print(Node *&x) {
  if (x == null) return; print(x->child[0]); printf("%d␣", x->key);
  print(x->child[1]);
}
inline Node* create(int key)
  Node *x = new Node(rand() % INT_MAX); x->key = key; x->count = x->size = 1;
  x->child[0] = x->child[1] = null; return x;
}
inline void rotate(Node *&x, int dir) {
  Node *y = x->child[!dir]; x->child[!dir] = y->child[dir]; y->child[dir] = x;
  x->update(); y->update(); x = y;
}
inline void insert(Node *&x, int key) {
  if (x == null) { x = create(key); return; }
  if (x->key == key) x->count++;
  else if (x->key > key) {
    insert(x->child[0], key); if (x->child[0]->aux < x->aux) rotate(x, 1);
  } else {
    insert(x->child[1], key); if (x->child[1]->aux < x->aux) rotate(x, 0);
  } x->update();
}
inline void erase(Node *&x, int key) {
  if (x == null) return;
  if (x->key == key) {
    if (x->count > 1) x->count--;
    else if (x->child[0] == null && x->child[1] == null) {
      delete(x); x = null; return;
    } else if (x->child[0]->aux < x->child[1]->aux)
      rotate(x, 1), erase(x->child[1], key);
    else rotate(x, 0), erase(x->child[0], key);
  } else if (x->key > key) erase(x->child[0], key);
  else erase(x->child[1], key);
  x->update();
}
inline void prepare() { null = new Node(INT_MAX); }
```

23 坚固的 Treap

```
namespace functional_treap {
  struct node {
    int size;
    node *left, *right;
    inline node(node *_left, node *_right) {
      left = _left;
      right = _right;
    }
    inline node* update() {
      size = left->size + 1 + right->size;
      return this;
    }
    inline pair<node*, node*> split(int);
  };

  node* null;

  inline bool random(int x, int y) {
    return rand() % (x + y) < x;
  }

  inline node* mergy(node* x, node* y) {
    if (x == null) {
      return y;
    }
    if (y == null) {
      return x;
    }
    if (random(x->size, y->size)) {
      x->right = mergy(x->right, y);
      return x->update();
    }
    y->left = mergy(x, y->left);
    return y->update();
  }

  inline pair<node*, node*> node::split(int n) {
    if (this == null) {
      return make_pair(null, null);
    }
    if (n <= left->size) {
      pair<node*, node*> ret = left->split(n);
      left = null;
      return make_pair(ret.first, mergy(ret.second, this->update()));
    }
    pair<node*, node*> ret = right->split(n - left->size);
    right = null;
    return make_pair(mergy(this->update(), ret.first), ret.second);
  }

  inline void prepare() {
    null = new node(null, null);
    null->left = null->right = null;
  }
}
```

24 LCT

```
struct Node {
  Node *child[2], *father; bool head, rev; int val, sum, size;
  inline Node() { head = rev = val = sum = size = 0; }
  inline void set(Node *temp, int dir) {
    child[dir] = temp; temp->father = this;
  }
  inline int which() { return father->child[1] == this; }
  inline void update() {
    sum = val + child[0]->sum + child[1]->sum;
    size = 1 + child[0]->size + child[1]->size;
  }
  inline void release() {
    if (rev) child[0]->reverse(), child[1]->reverse(), rev = 0;
  }
  inline void reverse() {
    if (size == 0) return;
    rev ^= 1; swap(child[0], child[1]);
  }
};
Node *null, *tree[N];
inline Node* create(int val) {
  Node *temp = new Node();
  temp->val = temp->sum = val; temp->size = 1;
  temp->child[0] = temp->child[1] = temp->father = null;
  temp->head = true; return temp;
}
```

```
inline void rotate(Node *root) {
  Node *father = root->father; father->release(); root->release();
  int dir = root->which(); father->set(root->child[!dir], dir);
  if (father->head) {
    father->head = false; root->head = true;
    root->father = father->father;
  } else father->father->set(root, father->which());
  root->set(father, !dir); father->update();
}
inline void splay(Node *root) {
  for (root->release(); !root->head; )
    if (root->father->head) rotate(root);
    else root->which() == root->father->which() ? (rotate(root->father), rotate(root)) :
         (rotate(root), rotate(root));
  root->update();
}
inline void access(Node *root) {
  for (Node *temp = null; root != null; temp = root, root = root->father) {
    splay(root); root->child[1]->head = true;
    root->child[1] = temp; root->child[1]->head = false; root->update();
  }
}
inline void link(int son, int father) {
  access(tree[son]); splay(tree[son]); tree[son]->father = tree[father];
  tree[son]->reverse();
}
inline void cut(int x, int y) {
  access(tree[y]); splay(tree[x]);
  if (tree[x]->father == tree[y]) tree[x]->father = null;
  else {
    access(tree[x]); splay(tree[y]);
    if (tree[y]->father == tree[x]) tree[y]->father = null;
  }
}
inline void handle(int x, int y) {
  access(tree[x]); Node *root = tree[y];
  for (Node *temp = null; root != null; temp = root, root = root->father) {
    splay(root); if (root->father == null) {  }
    root->child[1]->head = true; root->child[1] = temp;
    root->child[1]->head = false; root->update();
  }
}
inline void init(int n, int val[]) {
  for (int i = 1; i <= n; ++i) tree[i] = create(val[i]);
}
inline void prepare() {
  null = new Node(); null->child[0] = null->child[1] = null->father = null;
}
```

25 Splay

```
namespace splay {
struct Node {
  Node *child[2], *father; int val, sum, size;
  inline Node() { val = sum = size = 0; }
  inline int which() { return father->child[1] == this; }
  inline void set(Node *temp, int dir) {
    child[dir] = temp; temp->father = this;
  }
  inline void update() {
    size = 1 + child[0]->size + child[1]->size;
    sum = val + child[0]->sum + child[1]->sum;
  }
  inline void release() {}
};
Node *null, *head;
inline void print(Node *root) {
  if (root == null) return; print(root->child[0]); printf("%d␣", root->val);
  print(root->child[1]);
}
inline Node* create(int val = 0) {
  Node *temp = new Node(); temp->val = val;
  temp->child[0] = temp->child[1] = temp->father = null; return temp;
}
inline void rotate(Node *root) {
  Node *father = root->father; int dir = root->which(); father->release();
  root->release; father->set(root->child[!dir], dir);
  father->father->set(root, father->which()); root->set(father, !dir);
  if (father == head) head = root;
  father->update();
}
inline void splay(Node *root, Node *target) {
  for (root->release(); root->father != target; )
    if (root->father->father == target) rotate(root);
    else root->which() == root->father->which() ? (rotate(root->father), rotate(root)) :
         (rotate(root), rotate(root));
  root->update();
}
inline int rank(Node *root) {
  splay(root, null); return root->child[0]->size + 1;
}
inline Node* find(int rank) {
  Node *now = head;
  for (; now->child[0]->size + 1 != rank; ) {
    now->release();
    if (now->child[0]->size + 1 > rank) now = now->child[0];
    else { rank -= now->child[0]->size + 1; now = now->child[1]; }
  } return now;
}
inline void splay(int left, int right) {
  splay(find(right), null); splay(find(left), head);
}
inline Node* insert(int pos, int val) {
  splay(pos, pos + 1); Node *now = head->child[0]; Node *cur = create(val);
  now->set(cur, 1); splay(cur, null); return head;
}
inline void insert(int pos, int n, int val[]) {
  splay(pos, pos + 1); Node *now = head->child[0];
  for (int i = 1; i <= n; ++i) {
    Node *cur = create(val[i]); now->set(cur, 1); now = cur;
  } splay(now, null);
}
inline void erase(Node *root) {
  int pos = rank(root); splay(pos - 1, pos + 1);
  head->child[0]->child[1] = null; head->child[0]->update(); head->update();
}
inline int query(int left, int right) {
  splay(left - 1, right + 1); return head->child[0]->child[1]->sum;
}
inline void prepare() {
  null = new Node(); head = create(); Node *tail = create();
  head->set(tail, 1); splay(tail, null);
}
```

26 Gabow 算法求点双联通分量 (非递归)

```
int color[222222], siz[222222], cnt[222222];
long long ans[222222];
vector<int> edges[222222];
vector<pair<int, int> > st0, st2;
vector<int> st1;
void psh(int v) {
  st0.push_back(make_pair(v, 0));
  color[v] = st1.size();
  st1.push_back(v);
}
int main() {
  freopen("travel.in", "r", stdin);
  freopen("travel.out", "w", stdout);
  int n, m;
  scanf("%d%d", &n, &m);
  for(int i(1); i <= m; i++) {
```

```
    int x, y;
    scanf("%d%d", &x, &y);
    edges[x].push_back(y);
    edges[y].push_back(x);
  }
  int c(n);
  fill(color + 1, color + 1 + n, 0);
  fill(ans + 1, ans + 1 + n, 0);
  fill(cnt + 1, cnt + 1 + n, 0);
  fill(siz + 1, siz + 1 + n, 0);
  for(int i(1); i <= n; i++) if(!color[i]) {
    psh(i);
    while(!st0.empty()) {

      int v(st0.back().first), p(st0.back().second++);
      if(p != (int)edges[v].size()) {
        int y(edges[v][p]);
        if(!color[y]) {
          psh(y);
          st2.push_back(make_pair(color[v], color[y]));
        }else
          while(!st2.empty() and st2.back().first > color[y])
            st2.pop_back();
      }else {
        st0.pop_back();
        siz[v]++;
        if(color[v] == 1)
          color[v] = c;
        else {
          int fa(st0.back().first);
          if(st2.back().second == color[v]) {
            st2.pop_back();
            color[v] = ++c;
            while(st1.back() != v) {
              color[st1.back()] = c;
              st1.pop_back();
            }
            st1.pop_back();
            ans[fa] += (long long)cnt[fa] * siz[v];
            cnt[fa] += siz[v];
          }
          siz[fa] += siz[v];
        }
        ans[v] += (long long)(n - cnt[v]) * cnt[v] + n - cnt[v] - 1;
      }
    }
  }
  for(int i(1); i <= n; i++) {
    cout << ans[i] << endl;//ans[i]:删去点i后，无法连通的{a, b}数，其中a, b
        为图中不同节点且无序.
  }
  fclose(stdin);
  fclose(stdout);
  return 0;
}
```

27 $O(EV^{0.5})$ HK 求二分图最大匹配

```
// hint :: 全部都是0base
// 用的时候，建好边，左边n个点，右边m个点，直接调用maxMatch即可

const int N = 3333;

vector<int> e[N];
int pairx[N], pairy[N], level[N];
int n, m;

bool dfs(int x) {
  for(int i = 0; i < (int)e[x].size(); i++) {
    int y = e[x][i];
    int w = pairy[y];
    if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
      pairx[x] = y;
      pairy[y] = x;
      return true;
    }
  }
  level[x] = -1;
  return false;
}

int maxMatch() {
  fill(pairx, pairx + n, -1);
  fill(pairy, pairy + m, -1);

  for(int answer = 0; ; ) {
    vector<int> queue;
    for(int i = 0; i < n; i++) {
      if (pairx[i] == -1) {
        level[i] = 0;
        queue.push_back(i);
      } else {
        level[i] = -1;
      }
    }

    for(int head = 0; head < (int)queue.size(); head++) {
      int x = queue[head];
      for(int i = 0; i < (int)e[x].size(); i++) {
        int y = e[x][i];
        int w = pairy[y];
        if (w != -1 && level[w] < 0) {
          level[w] = level[x] + 1;
          queue.push_back(w);
        }
      }
    }

    int delta = 0;
    for(int i = 0; i < n; i++) {
      if (pairx[i] == -1 && dfs(i)) {
        delta++;
      }
    }
    if (delta == 0) {
      return answer;
    } else {
      answer += delta;
    }
  }
}

int solve() {
  int timing;
  scanf("%d", &timing);

  static int x[N], y[N], s[N];
  scanf("%d", &n);
  for(int i = 0; i < n; i++) {
    scanf("%d␣%d␣%d", &x[i], &y[i], &s[i]);
    e[i].clear();
  }

  scanf("%d", &m);
  for(int i = 0; i < m; i++) {
    int xx, yy;
    scanf("%d␣%d", &xx, &yy);
    for(int j = 0; j < n; j++) {
      if (timing * timing * s[j] * s[j] >= (xx - x[j]) * (xx - x[j]) + (yy - y[j]) * (yy
          - y[j])) {
        e[j].push_back(i);
      }
    }
  }

  return maxMatch();
```

```
}

int main() {
  freopen("input.txt", "r", stdin);
  int test;
  scanf("%d", &test);
  while(test--) {
    static int testCount = 0;
    printf("Scenario #%d:\n", ++testCount);
    printf("%d\n", solve());
    puts("");
  }
  return 0;
}
```

28 $O(V^3)$ 最小树形图

```
const int maxn=1100;

int n,m , g[maxn][maxn] , used[maxn] , pass[maxn] , eg[maxn] , more , queue[maxn];

void combine (int id , int &sum ) {
  int tot = 0 , from , i , j , k ;
  for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
    queue[tot++]=id ; pass[id]=1;
  }
  for ( from=0; from<tot && queue[from]!=id ; from++);
  if  ( from==tot ) return ;
  more = 1 ;
  for ( i=from ; i<tot ; i++) {
    sum+=g[eg[queue[i]]][queue[i]] ;
    if ( i!=from ) {
      used[queue[i]]=1;
      for ( j = 1 ; j <= n ; j++) if ( !used[j] )
        if ( g[queue[i]][j]<g[id][j] ) g[id][j]=g[queue[i]][j] ;
    }
  }
  for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
    for ( j=from ; j<tot ; j++){
      k=queue[j];
      if ( g[i][id]>g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg[k]][k];
    }
  }
}

int mdst( int root ) { // return the total length of MDST
  int i , j , k , sum = 0 ;
  memset ( used , 0 , sizeof ( used ) ) ;
  for ( more =1; more ; ) {
    more = 0 ;
    memset (eg,0,sizeof(eg)) ;
    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(pass));
    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 ;
}


int main(){
   freopen("input.txt","r",stdin);
   freopen("output.txt","w",stdout);
   int i,j,k,test,cases;
   cases=0;
   scanf("%d",&test);
   while (test){
    test--;
    //if (n==0) break;
    scanf("%d%d",&n,&m);
//    memset(g,60,sizeof(g));
    foru(i,1,n)
      foru(j,1,n) g[i][j]=1000001;
    foru(i,1,m) {
      scanf("%d%d",&j,&k);
      j++;k++;
      scanf("%d",&g[j][k]);
    }
    cases++;
    printf("Case #%d: ",cases);
    k=mdst(1);
    if (k>1000000) printf("Possums!\n");  //===no
    else printf("%d\n",k);
  }

   return 0;
}
```

29 KM

```
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <vector>
#include <cstring>
#include <string>
#include <iostream>

#define foreach(e, x) for(__typeof(x.begin()) e = x.begin(); e != x.end(); ++e)

using namespace std;

const int N = 333;
const int INF = (1 << 30);

int mat[N][N], lx[N], ly[N], vx[N], vy[N], slack[N];
int n, match[N];

bool find(int x) {
  vx[x] = 1;
  for(int i = 1; i <= n; i++) {
    if (vy[i]) {
      continue;
    }
    int temp = lx[x] + ly[i] - mat[x][i];
    if (temp == 0) {
      vy[i] = 1;
      if (match[i] == -1 || find(match[i])) {
        match[i] = x;
        return true;
      }
    } else {
      slack[i] = min(slack[i], temp);
    }
  }
  return false;
}

int KM() {
  for(int i = 1; i <= n; i++) {
    lx[i] = -INF;
    ly[i] = 0;
    match[i] = -1;
    for(int j = 1; j <= n; j++) {
      lx[i] = max(lx[i], mat[i][j]);
    }
  }
  for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= n; j++) {
```

```
      slack[j] = INF;
    }
    for(; ;) {
      memset(vx, 0, sizeof(vx));
      memset(vy, 0, sizeof(vy));
      for(int j = 1; j <= n; j++) {
        slack[j] = INF;
      }
      if (find(i)) {
        break;
      }
      int delta = INF;
      for(int j = 1; j <= n; j++) {
        if (!vy[j]) {
          delta = min(delta, slack[j]);
        }
      }
      for(int j = 1; j <= n; j++) {
        if (vx[j]) {
          lx[j] -= delta;
        }
        if (vy[j]) {
          ly[j] += delta;
        } else {
          slack[j] -= delta;
        }
      }
    }
  }
  int answer = 0;
  for(int i = 1; i <= n; i++) {
    answer += mat[match[i]][i];
  }
  return answer;
}

int main() {
  while(scanf("%d", &n) != EOF) {
    for(int i = 1; i <= n; i++) {
      for(int j = 1; j <= n; j++) {
        scanf("%d", &mat[i][j]);
      }
    }
    printf("%d\n", KM());
  }
  return 0;
}
```

30 带花树

```
const int N = 300;
int n, Next[N], f[N], mark[N], visited[N], Link[N], Q[N], head, tail;
vector <int> E[N];
int getf(int x) { return f[x] == x ? x : f[x] = getf(f[x]); }
void merge(int x, int y) { x = getf(x); y = getf(y); if (x != y) f[x] = y; }
int LCA(int x, int y) {
  static int flag = 0;
  flag++;
  for (; ; swap(x, y)) if (x != -1) {
    x = getf(x);
    if (visited[x] == flag) return x;
    visited[x] = flag;
    if (Link[x] != -1) x = Next[Link[x]];
    else x = -1;
  }
}
void go(int a, int p) {
  while (a != p) {
    int b = Link[a], c = Next[b];
    if (getf(c) != p) Next[c] = b;
    if (mark[b] == 2) mark[Q[tail++] = b] = 1;
    if (mark[c] == 2) mark[Q[tail++] = c] = 1;
    merge(a, b); merge(b, c); a = c;
  }
}
void find(int s) {
  for (int i = 0; i < n; i++) {
    Next[i] = -1; f[i] = i;
    mark[i] = 0; visited[i] = -1;
  }
  head = tail = 0; Q[tail++] = s; mark[s] = 1;
  for (; head < tail && Link[s] == -1; ) {
    for (int i = 0, x = Q[head++]; i < (int)E[x].size(); i++) {
      if (Link[x] != E[x][i] && getf(x) != getf(E[x][i]) && mark[E[x][i]] != 2) {
        int y = E[x][i];
        if (mark[y] == 1) {
          int p = LCA(x, y);
          if (getf(x) != p) Next[x] = y;
          if (getf(y) != p) Next[y] = x;
          go(x, p);
          go(y, p);
        }
        else if (Link[y] == -1) {
          Next[y] = x;
          for (int j = y; j != -1; ) {
            int k = Next[j];
            int tmp = Link[k];
            Link[j] = k;
            Link[k] = j;
            j = tmp;
          }
          break;
        }
        else {
          Next[y] = x;
          mark[Q[tail++] = Link[y]] = 1;
          mark[y] = 2;
        }
      }
    }
  }
}
int main() {
  for (int i = 0; i < n; i++) Link[i] = -1;
  for (int i = 0; i < n; i++) if (Link[i] == -1) {
    find(i);
  }
  int ans = 0;
  for (int i = 0; i < n; i++) ans += Link[i] != -1;
  return ans;
}
```

31 无向图最小割

```
const int V = 100;
#define typec int
const typec inf = 0x3f3f3f; // max of res
const typec maxw = 1000; // maximum edge weight
typec g[V][V], w[V]; //g[i][j] = g[j][i]
int a[V], v[V], na[V];
typec mincut(int n) {
  int i, j, pv, zj;
  typec best = maxw * n * n;
  for (i = 0; i < n; i++) v[i] = i; // vertex: 0 ~ n-1
  while (n > 1) {
    for (a[v[0]] = 1, i = 1; i < n; i++) {
      a[v[i]] = 0; na[i - 1] = i;
      w[i] = g[v[0]][v[i]];
    }
    for (pv = v[0], i = 1; i < n; i++) {
      for (zj = -1, j = 1; j < n; j++)
        if (!a[v[j]] && (zj < 0 || w[j] > w[zj]))
```

```
          zj = j;
      a[v[zj]] = 1;
      if (i == n - 1) {
        if (best > w[zj]) best = w[zj];
        for (i = 0; i < n; i++)
          g[v[i]][pv] = g[pv][v[i]] +=
            g[v[zj]][v[i]];
        v[zj] = v[--n];
        break;
      }
      pv = v[zj];
      for (j = 1; j < n; j++)
        if(!a[v[j]])
          w[j] += g[v[zj]][v[j]];
    }
  }
  return best;
}
```

32 哈密尔顿回路

```
bool graph[N][N];
int n, l[N], r[N], next[N], last[N], s, t;
char buf[10010];
void cover(int x) { l[r[x]] = l[x]; r[l[x]] = r[x]; }
int adjacent(int x) {
  for (int i = r[0]; i <= n; i = r[i]) if (graph[x][i]) return i;
  return 0;
}
int main() {
  scanf("%d\n", &n);
  for (int i = 1; i <= n; ++i) {
    gets(buf);
    string str = buf;
    istringstream sin(str);
    int x;
    while (sin >> x) {
      graph[i][x] = true;
    }
    l[i] = i - 1;
    r[i] = i + 1;
  }
  for (int i = 2; i <= n; ++i)
    if (graph[1][i]) {
      s = 1;
      t = i;
      cover(s);
      cover(t);
      next[s] = t;
      break;
    }
  while (true) {
    int x;
    while (x = adjacent(s)) {
      next[x] = s;
      s = x;
      cover(s);
    }
    while (x = adjacent(t)) {
      next[t] = x;
      t = x;
      cover(t);
    }
    if (!graph[s][t]) {
      for (int i = s, j; i != t; i = next[i])
        if (graph[s][next[i]] && graph[t][i]) {
          for (j = s; j != i; j = next[j])
            last[next[j]] = j;
          j = next[s];
          next[s] = next[i];
          next[t] = i;
          t = j;
          for (j = i; j != s; j = last[j])
            next[j] = last[j];
          break;
        }
    }
    next[t] = s;
    if (r[0] > n)
      break;
    for (int i = s; i != t; i = next[i])
      if (adjacent(i)) {
        s = next[i];
        t = i;
        next[t] = 0;
        break;
      }
  }
  for (int i = s; ; i = next[i]) {
    if (i == 1) {
      printf("%d", i);
      for (int j = next[i]; j != i; j = next[j])
        printf("␣%d", j);
      printf("␣%d\n", i);
      break;
    }
    if (i == t)
      break;
  }
}
```

33 弦图判定

```
int n, m, first[1001], l, next[2000001], where[2000001],f[1001], a[1001], c[1001], L
     [1001], R[1001],
v[1001], idx[1001], pos[1001];
bool b[1001][1001];

int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}

inline void makelist(int x, int y){
    where[++l] = y;
    next[l] = first[x];
    first[x] = l;
}

bool cmp(const int &x, const int &y){
    return(idx[x] < idx[y]);
}

int main(){
   //freopen("1015.in", "r", stdin);
   // freopen("1015.out", "w", stdout);
    for (;;)
    {
        n = read(); m = read();
        if (!n && !m) return 0;
        memset(first, 0, sizeof(first)); l = 0;
        memset(b, false, sizeof(b));
        for (int i = 1; i <= m; i++)
        {
            int x = read(), y = read();
            if (x != y && !b[x][y])
            {
               b[x][y] = true; b[y][x] = true;
               makelist(x, y); makelist(y, x);
```

```
            }
        }
        memset(f, 0, sizeof(f));
        memset(L, 0, sizeof(L));
        memset(R, 255, sizeof(R));
        L[0] = 1; R[0] = n;
        for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
        memset(idx, 0, sizeof(idx));
        memset(v, 0, sizeof(v));
        for (int i = n; i; --i)
        {
            int now = c[i];
            R[f[now]]--;
            if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
            idx[now] = i; v[i] = now;
            for (int x = first[now]; x; x = next[x])
                if (!idx[where[x]])
                {
                    swap(c[pos[where[x]]], c[R[f[where[x]]]]);
                    pos[c[pos[where[x]]]] = pos[where[x]];
                    pos[where[x]] = R[f[where[x]]];
                    L[f[where[x]] + 1] = R[f[where[x]]]--;
                    if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
                    if (R[f[where[x]] + 1] == -1)
                        R[f[where[x]] + 1] = L[f[where[x]] + 1];
                    ++f[where[x]];
                }
        }
        bool ok = true;
        //v是完美消除序列.
        for (int i = 1; i <= n && ok; i++)
        {
            int cnt = 0;
            for (int x = first[v[i]]; x; x = next[x])
                if (idx[where[x]] > i) c[++cnt] = where[x];
            sort(c + 1, c + cnt + 1, cmp);
            bool can = true;
            for (int j = 2; j <= cnt; j++)
                if (!b[c[1]][c[j]])
                {
                    ok = false;
                    break;
                }
        }
        if (ok) printf("Perfect\n");
        else printf("Imperfect\n");
        printf("\n");
    }
}
```

34 弦图求团数

```
int n, m, first[100001], next[2000001], where[2000001], l, L[100001], R[100001], c
     [100001], f[100001],
pos[100001], idx[100001], v[100001], ans;

inline void makelist(int x, int y){
    where[++l] = y;
    next[l] = first[x];
    first[x] = l;
}

int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}

int main(){
    freopen("1006.in", "r", stdin);
    freopen("1006.out", "w", stdout);
    memset(first, 0, sizeof(first)); l = 0;
    n = read(); m = read();
    for (int i = 1; i <= m; i++)
    {
        int x, y;
        x = read(); y = read();
        makelist(x, y); makelist(y, x);
    }
    memset(L, 0, sizeof(L));
    memset(R, 255, sizeof(R));
    memset(f, 0, sizeof(f));
    memset(idx, 0, sizeof(idx));
    for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
    L[0] = 1; R[0] = n; ans = 0;
    for (int i = n; i; --i)
    {
        int now = c[i], cnt = 1;
        idx[now] = i; v[i] = now;
        if (--R[f[now]] < L[f[now]]) R[f[now]] = -1;
        for (int x = first[now]; x; x = next[x])
            if (!idx[where[x]])
            {
                swap(c[pos[where[x]]], c[R[f[where[x]]]]);
                pos[c[pos[where[x]]]] = pos[where[x]];
                pos[where[x]] = R[f[where[x]]];
                L[f[where[x]] + 1] = R[f[where[x]]]--;
                if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
                if (R[f[where[x]] + 1] == -1) R[f[where[x]] + 1] = L[f[where[x]] + 1];
                ++f[where[x]];
            }
            else ++cnt;
        ans = max(ans, cnt);
    }
    printf("%d\n", ans);
}
```

35 ZKW 费用流

```
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <cstring>
#include <cmath>
using namespace std;

const int N = 105 << 2, M = 205 * 205 * 2;
const int inf = 1000000000;

struct eglist {
  int other[M], succ[M], last[N], cap[M], cost[M], sum;
  void clear() {
    memset(last, -1, sizeof(last));
    sum = 0;
  }
  void _addEdge(int a, int b, int c, int d) {
    other[sum] = b, succ[sum] = last[a], last[a] = sum, cost[sum] = d, cap[sum++] = c;
  }
  void addEdge(int a, int b, int c, int d) {
    _addEdge(a, b, c, d);
    _addEdge(b, a, 0, -d);
  }
}e;

int n, m, S, T, tot, totFlow, totCost;
int dis[N], slack[N], visit[N], cur[N];

int modlable() {
  int delta = inf;
  for(int i = 1; i <= T; i++) {
```

```
    if (!visit[i] && slack[i] < delta)
      delta = slack[i];
    slack[i] = inf;
    cur[i] = e.last[i];
  }
  if (delta == inf)
    return 1;
  for(int i = 1; i <= T; i++)
    if (visit[i])
      dis[i] += delta;
  return 0;
}

int dfs(int x, int flow) {
  if (x == T) {
    totFlow += flow;
    totCost += flow * (dis[S] - dis[T]);
    return flow;
  }
  visit[x] = 1;
  int left = flow;
  for(int &i = cur[x]; ~i; i = e.succ[i])
    if (e.cap[i] > 0 && !visit[e.other[i]]) {
      int y = e.other[i];
      if (dis[y] + e.cost[i] == dis[x]) {
        int delta = dfs(y, min(left, e.cap[i]));
        e.cap[i] -= delta;
        e.cap[i ^ 1] += delta;
        left -= delta;
        if (!left)
          return flow;
      } else {
        slack[y] = min(slack[y], dis[y] + e.cost[i] - dis[x]);
      }
    }
  return flow - left;
}

pair<int, int> minCost() {
  totFlow = 0, totCost = 0;
  fill(dis + 1, dis + T + 1, 0);
  for(int i = 1; i <= T; i++)
    cur[i] = e.last[i];
  do {
    do {
      fill(visit + 1, visit + T + 1, 0);
    } while(dfs(S, inf));
  } while(!modlable());
  return make_pair(totFlow, totCost);
}

void run() {
  scanf("%d␣%d", &m, &n);
  e.clear();
  S = m + n + 1, T = m + n + 2;
  tot = 0;
  for(int i = 1; i <= m; i++) {
    int times;
    scanf("%d", &times);
    e.addEdge(S, i, times, 0);
  }
  for(int i = 1; i <= n; i++) {
    int times;
    scanf("%d", &times);
    e.addEdge(i + m, T, times, 0);
  }
  for(int i = 1; i <= m; i++)
    for(int j = 1; j <= n; j++) {
      int cost;
      scanf("%d", &cost);
      e.addEdge(i, j + m, inf, cost);
    }
  pair<int, int> tmp = minCost();
  printf("%d\n", tmp.second);
}

int main() {
  int Test;
  scanf("%d", &Test);
  for(; Test--; run());
  return 0;
}
```

36 扩展 KMP

传入字符串 s 和长度 N，next[i]=LCP(s, s[i..N-1])

```
void z(char *s, int *next, int N)
{
  int j = 0, k = 1;
  while (j + 1 < N && s[j] == s[j + 1]) ++ j;
  next[0] = N - 1; next[1] = j;
  for(int i = 2; i < N; ++ i) {
    int far = k + next[k] - 1, L = next[i - k];
    if (L < far - i + 1) next[i] = L;
    else {
      j = max(0, far - i + 1);
      while (i + j < N && s[j] == s[i + j]) ++ j;
      next[i] = j; k = i;
    }
  }
}
```

37 后缀数组

字符串后面会自动加上一个最小字符 \0.

```
const int N = 4 * int(1e5) + 10;

int n, m;
int sa[N], ta[N], tb[N], *rank = ta, *tmp = tb;
int height[N], myLog[N], f[N][20];
int str[N];

bool cmp(int i, int j, int l) {
  return tmp[i] == tmp[j] && tmp[i + l] == tmp[j + l];
}

void radixSort() {
  static int w[N];
  fill(w, w + m, 0);
  for (int i = 0; i < n; i++) {
    w[rank[i]]++;
  }
  for (int i = 1; i < m; i++) {
    w[i] += w[i - 1];
  }
  for (int i = n - 1; i >= 0; i--) {
    sa[--w[rank[tmp[i]]]] = tmp[i];
  }
}

void suffixArray() {
  for (int i = 0; i < n; i++) {
    rank[i] = str[i];
    tmp[i] = i;
  }
  radixSort();
  for (int j = 1, i, p; j < n; j <<= 1, m = p) {
    for (i = n - j, p = 0; i < n; i++) {
      tmp[p++] = i;
    }
    for (i = 0; i < n; i++) {
```

```
      if (sa[i] >= j) {
        tmp[p++] = sa[i] - j;
      }
    }
    radixSort();
    for (swap(tmp, rank), rank[sa[0]] = 0, i = p = 1; i < n; i++) {
      rank[sa[i]] = cmp(sa[i - 1], sa[i], j) ? p - 1 : p++;
    }
  }
  for (int i = 0, j, k = 0; i < n; ++i, k = max(k - 1, 0)) {
    if (rank[i]) {
      j = sa[rank[i] - 1];
      for (; str[i + k] == str[j + k]; k++);
      height[rank[i]] = k;
    }
  }
  for (int i = 2; i <= n; i++) {
    myLog[i] = myLog[i >> 1] + 1;
  }
  for (int i = 1; i < n; i++) {
    f[i][0] = height[i];
  }
  for (int j = 1; 1 << j <= n; j++) {
    for (int i = 1; i + (1 << j) <= n; i++) {
      f[i][j] = min(f[i][j - 1], f[i + (1 << j - 1)][j - 1]);
    }
  }
}

int lcp(int l, int r) {
  if (l > r) {
    return 0;
  }
  int len = myLog[r - l + 1];
  return min(f[l][len], f[r - (1 << len) + 1][len]);
}

int nBase, mBase;
int cnt[N];
char buf[N];

int pos(int x) {
  return x / (mBase << 1 | 1 );
}

int main() {
  n = 0;
  m = 256;
  scanf("%d%d", &nBase, &mBase);
  for (int i = 0; i < nBase; i++) {
    scanf("%s", buf);
    for (int j = 0; j < mBase; j++) {
      str[n++] = buf[j];
    }
    for (int j = 0; j < mBase; j++) {
      str[n++] = buf[j];
    }
    str[n++] = i < nBase - 1 ? m++ : 0;
  }
  suffixArray();
  int result = 0, total = 0;
  for (int i = 0, j = 0; i < n; i++) {
    for (; j < n && total < nBase; j++) {
      int p = pos(sa[j]);
      total += cnt[p]++ == 0;
    }
    if (total == nBase) {
      result = max(result, lcp(i + 1, j - 1));
    }
    int p = pos(sa[i]);
    total -= --cnt[p] == 0;
  }
  result = min(result, mBase);
  printf("%d\n", result);
  vector <int> ans(n);
  total = 0;
  memset(cnt, 0, sizeof(cnt));
  for (int i = 0, j = 0; i < n; i++) {
    for (; j < n && total < nBase; j++) {
      int p = pos(sa[j]);
      total += cnt[p]++ == 0;
    }
    if (total == nBase && lcp(i + 1, j - 1) >= result) {
      for (int k = i; k < j; k++) {
        int p = pos(sa[k]);
        ans[p] = sa[k] % (mBase << 1 | 1);
      }
      break;
    }
    int p = pos(sa[i]);
    total -= --cnt[p] == 0;
  }
  for (int i = 0; i < nBase; i++) {
    printf("%d\n", ans[i] % mBase + 1);
  }
}
```

38 DC3

```
//DC3 待排序的字符串放在 r 数组中，从 r[0] 到 r[n-1]，长度为 n，且最大值小于 m.
//约定除 r[n-1] 外所有的 r[i] 都大于 0，r[n-1]=0。
//函数结束后，结果放在 sa 数组中，从 sa[0] 到 sa[n-1]。
//r 必须开长度乘 3
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)

int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b)
{
  return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
}
int c12(int k,int *r,int a,int b)
{
  if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
  else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m)
{
  int i;
  for(i=0;i<n;i++) wv[i]=r[a[i]];
  for(i=0;i<m;i++) wss[i]=0;
  for(i=0;i<n;i++) wss[wv[i]]++;
  for(i=1;i<m;i++) wss[i]+=wss[i-1];
  for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m)
{
  int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
  r[n]=r[n+1]=0;
  for(i=0;i<n;i++)
    if(i%3!=0) wa[tbc++]=i;
  sort(r+2,wa,wb,tbc,m);
  sort(r+1,wb,wa,tbc,m);
  sort(r,wa,wb,tbc,m);
  for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
    rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
  if (p<tbc) dc3(rn,san,tbc,p);
  else for (i=0;i<tbc;i++) san[rn[i]]=i;
  for (i=0;i<tbc;i++)
    if(san[i]<tb) wb[ta++]=san[i]*3;
  if(n%3==1) wb[ta++]=n-1;
  sort(r,wb,wa,ta,m);
  for(i=0;i<tbc;i++)
    wv[wb[i]=G(san[i])]=i;
```

```
  for(i=0,j=0,p=0;i<ta && j<tbc;p++)
    sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
  for(;i<ta;p++) sa[p]=wa[i++];
  for(;j<tbc;p++) sa[p]=wb[j++];
}

int main(){
  int n,m=0;
  scanf("%d",&n);
  for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
  printf("%d\n",m);
  s[n++]=0;
  dc3(s,sa,n,m);
  for (int i=0;i<n;i++) printf("%d␣",sa[i]);printf("\n");
}
```

39 AC 自动机

```
int const N = ;
struct Node {
  Node *next[N], *fail; int count;
  inline Node() { memset(next, 0, sizeof(next)); fail = 0; count = 0; }
};
Node *root;
inline int idx(char x) { return x - 'a'; }
inline void insert(Node *x, char *str) {
  int len = (int)strlen(str);
  for (int i = 0; i < len; ++i) {
    int c = idx(str[i]);
    if (!x->next[c]) x->next[c] = new Node();
    x = x->next[c];
  } x->count++;
}
inline void build() {
  vector<Node*> queue; queue.push_back(root->fail = root);
  for (int head = 0; head < (int)queue.size(); ++head) {
    Node* x = queue[head];
    for (int i = 0; i < N; ++i)
      if (x->next[i]) {
        x->next[i]->fail = (x == root) ? root : x->fail->next[i];
        x->next[i]->count += x->next[i]->fail->count;
        queue.push_back(x->next[i]);
      } else x->next[i] = (x == root) ? root : x->fail->next[i];
  }
}
inline void prepare() { root = new Node(); }
```

40 极长回文子串

```
//CF17 - E
typedef long long int64;
const int N = 4 * int(1e6) + 111;
const int mod = 51123987;
int n;
int input[N];
int start[N], finish[N];
int f[N];
int64 ans;
void prepare() {
  int k = 0;
  for (int i = 0; i < n; ++i) {
    if (k + f[k] < i) {
      int &l = f[i] = 0;
      for (; i - l - 1 >= 0 && i + l + 1 < n && input[i - l - 1] ==
           input[i + l + 1]; l++);
      k = i;
    } else {
      int &l = f[i] = f[k - (i - k)];
      if (i + l >= k + f[k]) {
        l = min(l, k + f[k] - i);
        for (; i - l - 1 >= 0 && i + l + 1 < n && input[i - l - 1] ==
             input[i + l + 1]; l++);
        k = i;
      }
    }
    int l = i - f[i], r = i + f[i];
    l += l & 1;
    r -= r & 1;
    if (l <= r) {
      l /= 2;
      r /= 2;
      int mid1 = l + r >> 1;
      int mid2 = mid1 + ((l + r) & 1);
      start[l]++;
      start[mid1 + 1]--;
      finish[mid2]++;
      finish[r + 1]--;
      ans = (ans + (r - l) / 2 + 1) % mod;
    }
  }
}
int main() {
  scanf("%d␣", &n);
  for (int i = 0; i < n; ++i) {
    input[i << 1] = getchar();
    if (i < n - 1)
      input[i << 1 | 1] = '*';
  }
  n = n * 2 - 1;
  prepare();
  ans = ans * (ans - 1) / 2 % mod;
  n = (n + 1) / 2;
  int sum = 0;
  for (int i = 0; i < n; ++i) {
    if (i) {
      start[i] = (start[i] + start[i - 1]) % mod;
      finish[i] = (finish[i] + finish[i - 1]) % mod;
    }
    ans = (ans - (int64)start[i] * sum % mod) % mod;
    sum = (sum + finish[i]) % mod;
  }
  cout << (ans + mod) % mod << endl;
}
```

41 后缀自动机多次询问串在母串中出现的次数

```

const int N = 255555;
const int C = 36;

struct Node {
  Node *next[C], *fail;
  int count, len;
  void clear() {
    for(int i = 0; i < C; i++)
      next[i] = NULL;
    len = count = 0;
    fail = NULL;
  }
};

Node *tail, *q[N * 2], pool[N * 2], *head;
int used = 0;
char bufer[N * 2];
int buc[N * 2], f[N * 2];

Node *newNode() {
  pool[used++].clear();
```

```
  return &pool[used - 1];
}

void add(int x) {
  Node *np = newNode(), *p = tail;
  tail = np;
  np->len = p->len + 1;
  for(; p && !p->next[x]; p = p->fail)
    p->next[x] = np;
  if (!p)
    np->fail = head;
  else if (p->len + 1 == p->next[x]->len)
    np->fail = p->next[x];
  else {
    Node *q = p->next[x], *nq = newNode();
    *nq = *q;
    nq->len = p->len + 1;
    q->fail = np->fail = nq;
    for(; p && p->next[x] == q; p = p->fail)
      p->next[x] = nq;
  }
}

int main() {
  scanf("%s\n", bufer);
  int length = strlen(bufer);
  head = tail = newNode();
  for(int i = 0; i < length; i++)
    add(bufer[i] - 'a');
  for(int i = 0; i < used; ++i)
    ++buc[pool[i].len];
  for(int i = 1; i <= length; i++)
    buc[i] += buc[i - 1];
  for(int i = used - 1; i >= 0; i--)
    q[--buc[pool[i].len]] = &pool[i];
  Node *iter = head;
  for(int i = 0; i < length; ++i)
    (iter = iter->next[bufer[i] - 'a'])->count++;
  for(int i = used - 1; i > 0; --i) {
    f[q[i]->len] = max(f[q[i]->len], q[i]->count);
    q[i]->fail->count += q[i]->count;
  }
  for(int i = length - 1; i > 0; --i) {
    f[i] = max(f[i + 1], f[i]);
  }
  for(int i = 1; i <= length; i++)
    printf("%d\n", f[i]);
  return 0;
}
```

42 循环串的最小表示

```
struct cyc_string
{
  int n, offset;
  char str[max_length];
  char & operator [] (int x)
  {return str[((offset + x) % n)];}
  cyc_string(){offset = 0;}
};
void minimum_circular_representation(cyc_string & a)
{
  int i = 0, j = 1, dlt = 0, n = a.n;
  while(i < n and j < n and dlt < n)
  {
    if(a[i + dlt] == a[j + dlt]) dlt++;
    else
    {
      if(a[i + dlt] > a[j + dlt]) i += dlt + 1; else j += dlt + 1;
      dlt = 0;
    }
  }
  a.offset = min(i, j);
}
int main()
{return 0;}
```

43 快速求逆

```
int inverse(int x, int modulo) {
  if(x == 1)
    return 1;
  return (long long)(modulo - modulo / x) * inverse(modulo % x, modulo) % modulo;
}
```

44 求某年某月某日是星期几

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

45 LL*LL%LL

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

46 next_nCk

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

47 单纯形

test on uva 12567

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

```
  vector<vector<double> > D(n + 2, vector<double>(m + 1));
  vector<int> ix(n + m);
  for(int i = 0; i < n + m; i++) ix[i] = i;
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < m - 1; j++)
      D[i][j] = -A[i][j];
    D[i][m - 1] = 1; D[i][m] = b[i];
    if (D[r][m] > D[i][m])
      r = i;
  }
  for(int j = 0; j < m - 1; j++) D[n][j] = c[j];
  D[n + 1][m - 1] = -1;
  for(double d; ;) {
    if (r < n) {
      swap(ix[s], ix[r + m]);
      D[r][s] = 1. / D[r][s];
      for(int j = 0; j <= m; j++)
        if (j != s) D[r][j] *= -D[r][s];
      for(int i = 0; i <= n + 1; i++)
        if (i != r) {
          for(int j = 0; j <= m; j++)
            if (j != s) D[i][j] += D[r][j] * D[i][s];
          D[i][s] *= D[r][s];
        }
    }
    r = -1, s = -1;
    for(int j = 0; j < m; j++)
      if (s < 0 || ix[s] > ix[j])
        if (D[n + 1][j] > eps || D[n + 1][j] > -eps && D[n][j] > eps)
          s = j;
    if (s < 0) break;
    for(int i = 0; i < n; i++)
      if (D[i][s] < -eps)
        if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -eps || d < eps && ix[
            r + m] > ix[i + m])
          r = i;
    if (r < 0) return vector<double> ();
  }
  if (D[n + 1][m] < -eps) return vector<double> ();
  vector<double> x(m - 1);
  for(int i = m; i < n + m; i++)
    if (ix[i] < m - 1)
      x[ix[i]] = D[i - m][m];
  return x;
}
```

48 环状最长公共子序列

```
const int N = 2222;

int a[N], b[N];
int n, dp[N][N], from[N][N];

int run() {
  scanf("%d", &n);
  for(int i = 1; i <= n; i++) {
    scanf("%d", &a[i]);
    a[i + n] = a[i];
    b[n - i + 1] = a[i];
  }
  memset(from, 0, sizeof(from));
  int ans = 0;
  for(int i = 1; i <= 2 * n; i++) {
    from[i][0] = 2;
    int upleft = 0, up = 0, left = 0;
    for(int j = 1; j <= n; j++) {
      upleft = up;
      if (a[i] == b[j]) {
        upleft++;
      } else {
        upleft = INT_MIN;
      }
      if (from[i - 1][j])
        up++;
      int mm = max(up, max(left, upleft));
      if (mm == left) {
        from[i][j] = 0;
      } else if (mm == upleft)
        from[i][j] = 1;
      else
        from[i][j] = 2;
      left = mm;
    }
    if (i >= n) {
      int count = 0;
      for(int x = i, y = n; y; ) {
        if (from[x][y] == 1) {
          x--; y--;
          count++;
        } else if (from[x][y] == 0)
          y--;
        else
          x--;
      }
      ans = max(ans, count);
      int x = i - n + 1;
      from[x][0] = 0;
      int y = 0;
      for(; y <= n && from[x][y] == 0; y++);
      for(; x <= i; x++) {
        from[x][y] = 0;
        if (x == i) {
          break;
        }
        for(; y <= n; ++y) {
          if (from[x + 1][y] == 2) {
            break;
          }
          if (y + 1 <= n && from[x + 1][y + 1] == 1) {
            y++;
            break;
          }
        }
      }
    }
  }
  if (n)
    printf("%d\n", ans);
  return n;
}

int main() {
  for(; run(); );
  return 0;
}
```

49 长方体表面两点最近距离

```
int r;
void turn(int i, int j, int x, int y, int z,int x0, int y0, int L, int W, int H) {
  if (z==0) {
    int R = x*x+y*y;
    if (R<r) r=R;
  }
  else{
    if(i>=0 && i< 2)
      turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);
    if(j>=0 && j< 2)
      turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);
    if(i<=0 && i>-2)
      turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
    if(j<=0 && j>-2)
```

```
      turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
  }
}
int main(){
  int L, H, W, x1, y1, z1, x2, y2, z2;
  cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
  if (z1!=0 && z1!=H)
  if (y1==0 || y1==W)
    swap(y1,z1), std::swap(y2,z2), std::swap(W,H);
  else
    swap(x1,z1), std::swap(x2,z2), std::swap(L,H);
  if (z1==H) z1=0, z2=H-z2;
  r=0x3fffffff; turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
  cout<<r<<endl;
  return 0;
}
```

50 插头 DP

```
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <vector>
#include <iostream>
using namespace std;

typedef long long int64;
typedef pair<int, long long> State;
const int MAXN = 8;

char map[MAXN + 10][MAXN + 10];
int n, m, lastx, lasty;
int64 ans;
vector<State> vec[2];


void mergy(int cur) {
  sort(vec[cur].begin(), vec[cur].end());
  int size = 0;
  for(int i = 0, j = 0; i < vec[cur].size(); i = j) {
    vec[cur][size] = vec[cur][i];
    j = i + 1;
    while(j < vec[cur].size() && vec[cur][j].first == vec[cur][size].first)
      vec[cur][size].second += vec[cur][j].second, j++;
    size++;
  }
  vec[cur].resize(size);
}

void next_line(int cur) {
  int size = 0;
  for(int i = 0; i < vec[cur].size(); i++) {
    int sta = vec[cur][i].first;
    if ((sta >> (m << 1)) == 0) {
      vec[cur][size] = vec[cur][i];
      vec[cur][size].first <<= 2;
      size++;
    }
  }
  vec[cur].resize(size);
}

inline int replace(int sta, int pos, int v) {
  return (sta & (~(3 << (pos << 1)))) | (v << (pos << 1));
}

inline int replace(int &sta, int pos, int v1, int v2) {
  int res = replace(sta, pos, v1);
  res = replace(res, pos + 1, v2);
  return res;
}

int Trans(int sta, int pos) {
  int cnt = 1, v = (sta >> (pos << 1) & 3);
  if (v == 1) {
    sta = replace(sta, pos, 0, 0);
    for(int i = pos + 2; ; i++) {
      if ((sta >> (i << 1) & 3) == 1)
        cnt++;
      else if ((sta >> (i << 1) & 3) == 2)
        cnt--;
      if (cnt == 0)
        return replace(sta, i, 1);
    }
  } else {
    sta = replace(sta, pos, 0, 0);
    for(int i = pos - 1; ; i--) {
      if ((sta >> (i << 1) & 3) == 1)
        cnt--;
      else if ((sta >> (i << 1) & 3) == 2)
        cnt++;
      if (cnt == 0)
        return replace(sta, i, 2);
    }
  }
}

void dp_block(int i, int j, int cur) {
  for(int s = 0; s < vec[cur].size(); s++) {
    int sta = vec[cur][s].first;
    int64 val = vec[cur][s].second;
    int left = (sta >> (j << 1)) & 3, up = (sta >> ((j + 1) << 1)) & 3;
    if (left == 0 && up == 0) {
      vec[cur ^ 1].push_back(State(sta, val));
    }
  }
}

void dp_blank(int i, int j, int cur) {
  for(int s = 0; s < vec[cur].size(); s++) {
    int sta = vec[cur][s].first;
    int64 val = vec[cur][s].second;
    int left = (sta >> (j << 1)) & 3, up = (sta >> ((j + 1) << 1)) & 3, ns = 0;
    if (left && up) {
      if (left == 2 && up == 1) {
        vec[cur ^ 1].push_back(State(replace(sta, j, 0, 0), val));
      } else if (left == 1 && up == 2) {
        if (replace(sta, j, 0, 0) == 0 && i == lastx && j == lasty)
          ans += val;
      } else if (left == 1 && up == 1) {
        vec[cur ^ 1].push_back(State(Trans(sta, j), val));
      } else if (left == 2 && up == 2) {
        vec[cur ^ 1].push_back(State(Trans(sta, j), val));
      }
    } else if (left || up) {
      vec[cur ^ 1].push_back(State(sta, val));
      vec[cur ^ 1].push_back(State(replace(sta, j, up, left), val));
    } else {
      vec[cur ^ 1].push_back(State(replace(sta, j, 1, 2), val));
    }
  }
}

void show(int cur) {
  for(int i = 0; i < vec[cur].size(); i++)
    printf("%d␣%I64d\n", vec[cur][i].first, vec[cur][i].second);
  printf("step\n");
}

int main() {
  freopen("input.txt", "r", stdin);
  while(scanf("%d␣%d", &n, &m) == 2) {
    ans = 0;
    lastx = lasty = -1;
    gets(map[0]);
    for(int i = 0; i < n; i++) {
      scanf("%s", map[i]);
```

```
      for(int j = 0; j < m; j++) {
        if (map[i][j] == '.') {
          lastx = i, lasty = j;
        }
      }
    }
    if (lastx == -1) {
      printf("0\n");
      continue;
    }
    int cur = 0;
    vec[cur].clear();
    vec[cur].push_back(State(0, 1));
    for(int i = 0; i < n; i++) {
      for(int j = 0; j < m; j++) {
        vec[cur ^ 1].clear();
        if (map[i][j] == '.')
          dp_blank(i, j, cur);
        else
          dp_block(i, j, cur);
        cur ^= 1;
        mergy(cur);
        //show(cur);
      }
      next_line(cur);
    }
    cout << ans << endl;
  }
  return 0;
}
```

51 极大团搜索

Int g[][] 为图的邻接矩阵。MC(V) 表示点集 V 的最大团令 Si=vi, vi+1, ..., vn, mc[i] 表示 MC(Si) 倒着算 mc[i], 那么显然 MC(V)=mc[1] 此外有 mc[i]=mc[i+1] or mc[i]=mc[i+1]+1

```
void init(){
  int i, j;
  for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%d", &g[i][j]);
}
void dfs(int size){
  int i, j, k;
  if (len[size]==0) {
    if (size>ans) {
      ans=size; found=true;
    }
    return;
  }
  for (k=0; k<len[size] && !found; ++k) {
    if (size+len[size]-k<=ans) break;
    i=list[size][k];
    if (size+mc[i]<=ans) break;
    for (j=k+1, len[size+1]=0; j<len[size]; ++j)
    if (g[i][list[size][j]]) list[size+1][len[size+1]++]=list[size][j];
    dfs(size+1);
  }
}
void work(){
  int i, j;
  mc[n]=ans=1;
  for (i=n-1; i; --i) {
    found=false;
    len[1]=0;
    for (j=i+1; j<=n; ++j) if (g[i][j]) list[1][len[1]++]=j;
    dfs(1);
    mc[i]=ans;
  }
}
void print(){
  printf("%d\n", ans);
}
```

52 Dancing Links X

```
int const N = , M = , G = ;
struct node {
  int col, row, left, right, up, down;
  inline void clear() { col = row = left = right = up = down = 0; }
} grid[G];
int n, m, tot;
int cnt[M], head[N], tail[N];
inline void prepare() {
  tot = m + 1;
  for (int i = 1; i <= n; ++i) head[i] = tail[i] = 0;
  for (int i = 1; i <= m + 1; ++i) {
    grid[i].col = i; grid[i].left = i - 1; grid[i].right = i + 1;
    grid[i].up = i; grid[i].down = i; cnt[i] = 0;
  }
  grid[1].left = m + 1; grid[m + 1].right = 1;
}
inline void remove(int x) {
  grid[grid[x].right].left = grid[x].left;
  grid[grid[x].left].right = grid[x].right;
  for (int y = grid[x].down; y != x; y = grid[y].down)
    for (int z = grid[y].right; z != y; z = grid[z].right) {
      cnt[grid[z].col]--;
      grid[grid[z].down].up = grid[z].up;
      grid[grid[z].up].down = grid[z].down;
    }
}
inline void resume(int x) {
  for (int y = grid[x].up; y != x; y = grid[y].up)
    for (int z = grid[y].left; z != y; z = grid[z].left) {
      cnt[grid[z].col]++;
      grid[grid[z].up].down = z;
      grid[grid[z].down].up = z;
    }
  grid[grid[x].right].left = x; grid[grid[x].left].right = x;
}
inline void add(int x, int y) {
  tot++; cnt[y]++;
  if (!head[x]) head[x] = tot;
  if (!tail[x]) tail[x] = tot;
  grid[tot].row = x; grid[tot].col = y;
  grid[tot].up = grid[y].up; grid[grid[y].up].down = tot;
  grid[tot].down = y; grid[y].up = tot;
  grid[tot].left = tail[x]; grid[tail[x]].right = tot;
  grid[tot].right = head[x]; grid[head[x]].left = tot;
  tail[x] = tot;
}
inline bool dfs(int dep) {
  if (grid[m + 1].right == m + 1) return true;
  int x = grid[m + 1].right;
  for (int i = x; i != m + 1; i = grid[i].right) if (cnt[i] < cnt[x]) x = i;
  if (!cnt[x]) return false;
  remove(x);
  for (int i = grid[x].down; i != x; i = grid[i].down) {
    for (int j = grid[i].right; j != i; j = grid[j].right) remove(grid[j].col);
    if (dfs(dep + 1)) return true;
    for (int j = grid[i].left; j != i; j = grid[j].left) resume(grid[j].col);
  }
  resume(x); return false;
}
inline void clear() { for (int i = 1; i <= tot; ++i) grid[i].clear(); }
```

53 积分表

$$\arcsin x \to \frac{1}{\sqrt{1-x^2}}$$

$$\arccos x \to -\frac{1}{\sqrt{1-x^2}}$$

$$\arctan x \to \frac{1}{1+x^2}$$

$$a^x \to \frac{a^x}{\ln a}$$

$$\sin x \to -\cos x$$

$$\cos x \to \sin x$$

$$\tan x \to -\ln \cos x$$

$$\sec x \to \ln \tan(\frac{x}{2}+\frac{\pi}{4})$$

$$\tan^2 x \to \tan x - x$$

$$\csc x \to \ln \tan \frac{x}{2}$$

$$\sin^2 x \to \frac{x}{2} - \frac{1}{2}\sin x \cos x$$

$$\cos^2 x \to \frac{x}{2} + \frac{1}{2}\sin x \cos x$$

$$\sec^2 x \to \tan x$$

$$\frac{1}{\sqrt{a^2-x^2}} \to \arcsin \frac{x}{a}$$

$$\csc^2 x \to -\cot x$$

$$\frac{1}{a^2-x^2}(|x|<|a|) \to \frac{1}{2a}\ln\frac{a+x}{a-x}$$

$$\frac{1}{x^2-a^2}(|x|>|a|) \to \frac{1}{2a}\ln\frac{x-a}{x+a}$$

$$\sqrt{a^2-x^2} \to \frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a}$$

$$\frac{1}{\sqrt{x^2+a^2}} \to \ln(x+\sqrt{a^2+x^2})$$

$$\sqrt{a^2+x^2} \to \frac{x}{2}\sqrt{a^2+x^2} + \frac{a^2}{2}\ln(x+\sqrt{a^2+x^2})$$

$$\frac{1}{\sqrt{x^2-a^2}} \to \ln(x+\sqrt{x^2-a^2})$$

$$\sqrt{x^2-a^2} \to \frac{x}{2}\sqrt{x^2-a^2} - \frac{a^2}{2}\ln(x+\sqrt{x^2-a^2})$$

$$\frac{1}{x\sqrt{a^2-x^2}} \to -\frac{1}{a}\ln\frac{a+\sqrt{a^2-x^2}}{x}$$

$$\frac{1}{x\sqrt{x^2-a^2}} \to \frac{1}{a}\arccos\frac{a}{x}$$

$$\frac{1}{x\sqrt{a^2+x^2}} \to -\frac{1}{a}\ln\frac{a+\sqrt{a^2+x^2}}{x}$$

$$\frac{1}{\sqrt{2ax-x^2}} \to \arccos(1-\frac{x}{a})$$

$$\frac{x}{ax+b} \to \frac{x}{a} - \frac{b}{a^2}\ln(ax+b)$$

$$\sqrt{2ax-x^2} \to \frac{x-a}{2}\sqrt{2ax-x^2} + \frac{a^2}{2}\arcsin(\frac{x}{a}-1)$$

$$\frac{1}{x\sqrt{ax+b}}(b<0) \to \frac{2}{\sqrt{-b}}\arctan\sqrt{\frac{ax+b}{-b}}$$

$$x\sqrt{ax+b} \to \frac{2(3ax-2b)}{15a^2}(ax+b)^{\frac{3}{2}}$$

$$\frac{1}{x\sqrt{ax+b}}(b>0) \to \frac{1}{\sqrt{b}}\ln\frac{\sqrt{ax+b}-\sqrt{b}}{\sqrt{ax+b}+\sqrt{b}}$$

$$\frac{x}{\sqrt{ax+b}} \to \frac{2(ax-2b)}{3a^2}\sqrt{ax+b}$$

$$\frac{1}{x^2\sqrt{ax+b}} \to -\frac{\sqrt{ax+b}}{bx} - \frac{a}{2b}\int\frac{\mathrm{d}x}{x\sqrt{ax+b}}$$

$$\frac{\sqrt{ax+b}}{x} \to 2\sqrt{ax+b} + b\int\frac{\mathrm{d}x}{x\sqrt{ax+b}}$$

$$\frac{1}{\sqrt{(ax+b)^n}}(n>2) \to \frac{-2}{a(n-2)}\cdot\frac{1}{\sqrt{(ax+b)^{n-2}}}$$

$$\frac{1}{ax^2+c}(a>0,c>0) \to \frac{1}{\sqrt{ac}}\arctan\left(x\sqrt{\frac{a}{c}}\right)$$

$$\frac{x}{ax^2+c} \to \frac{1}{2a}\ln(ax^2+c)$$

$$\frac{1}{ax^2+c}(a+,c-) \to \frac{1}{2\sqrt{-ac}}\ln\frac{x\sqrt{a}-\sqrt{-c}}{x\sqrt{a}+\sqrt{-c}}$$

$$\frac{1}{x(ax^2+c)} \to \frac{1}{2c}\ln\frac{x^2}{ax^2+c}$$

$$\frac{1}{ax^2+c}(a-,c+) \to \frac{1}{2\sqrt{-ac}}\ln\frac{\sqrt{c}+x\sqrt{-a}}{\sqrt{c}-x\sqrt{-a}}$$

$$x\sqrt{ax^2+c} \to \frac{1}{3a}\sqrt{(ax^2+c)^3}$$

$$\frac{1}{(ax^2+c)^n}(n>1) \to \frac{x}{2c(n-1)(ax^2+c)^{n-1}} + \frac{2n-3}{2c(n-1)}\int\frac{\mathrm{d}x}{(ax^2+c)^{n-1}}$$

$$\frac{x^n}{ax^2+c}(n\neq 1) \to \frac{x^{n-1}}{a(n-1)} - \frac{c}{a}\int\frac{x^{n-2}}{ax^2+c}\mathrm{d}x$$

$$\frac{1}{x^2(ax^2+c)} \to \frac{-1}{cx} - \frac{a}{c}\int\frac{\mathrm{d}x}{ax^2+c}$$

$$\frac{1}{x^2(ax^2+c)^n}(n\geq 2) \to \frac{1}{c}\int\frac{\mathrm{d}x}{x^2(ax^2+c)^{n-1}} - \frac{a}{c}\int\frac{\mathrm{d}x}{(ax^2+c)^n}$$

$$\sqrt{ax^2+c}(a>0) \to \frac{x}{2}\sqrt{ax^2+c} + \frac{c}{2\sqrt{a}}\ln(x\sqrt{a}+\sqrt{ax^2+c})$$

$$\sqrt{ax^2+c}(a<0) \to \frac{x}{2}\sqrt{ax^2+c} + \frac{c}{2\sqrt{-a}}\arcsin(x\sqrt{\frac{-a}{c}})$$

$$\frac{1}{\sqrt{ax^2+c}}(a>0)\to\frac{1}{\sqrt{a}}\ln(x\sqrt{a}+\sqrt{ax^2+c})$$

$$\frac{1}{\sqrt{ax^2+c}}(a<0)\to\frac{1}{\sqrt{-a}}\arcsin(x\sqrt{-\frac{a}{c}})$$

$$\sin^2 ax\to\frac{x}{2}-\frac{1}{4a}\sin 2ax$$

$$\cos^2 ax\to\frac{x}{2}+\frac{1}{4a}\sin 2ax$$

$$\frac{1}{\sin ax}\to\frac{1}{a}\ln\tan\frac{ax}{2}$$

$$\frac{1}{\cos^2 ax}\to\frac{1}{a}\tan ax$$

$$\frac{1}{\cos ax}\to\frac{1}{a}\ln\tan(\frac{\pi}{4}+\frac{ax}{2})$$

$$\ln(ax)\to x\ln(ax)-x$$

$$\sin^3 ax\to\frac{-1}{a}\cos ax+\frac{1}{3a}\cos^3 ax$$

$$\cos^3 ax\to\frac{1}{a}\sin ax-\frac{1}{3a}\sin^3 ax$$

$$\frac{1}{\sin^2 ax}\to-\frac{1}{a}\cot ax$$

$$x\ln(ax)\to\frac{x^2}{2}\ln(ax)-\frac{x^2}{4}$$

$$\cos ax\to\frac{1}{a}\sin ax$$

$$x^2e^{ax}\to\frac{e^{ax}}{a^3}(a^2x^2-2ax+2)$$

$$(\ln(ax))^2\to x(\ln(ax))^2-2x\ln(ax)+2x$$

$$x^2\ln(ax)\to\frac{x^3}{3}\ln(ax)-\frac{x^3}{9}$$

$$x^n\ln(ax)\to\frac{x^{n+1}}{n+1}\ln(ax)-\frac{x^{n+1}}{(n+1)^2}$$

$$\sin(\ln ax)\to\frac{x}{2}[\sin(\ln ax)-\cos(\ln ax)]$$

$$\cos(\ln ax)\to\frac{x}{2}[\sin(\ln ax)+\cos(\ln ax)]$$

54 网络流

下界:(u, v) 下界为 c: 超级源到 t 建流量为 c, s 到超级汇建流量为 c, (原来的汇到原来的源建无穷, 如果有), 流一遍超级源出边满了就存在可行流.
下界最大流 (有源汇): 上面的搞完从原来的源到原来的汇流一遍
下界最小流 (有源汇): 上面的搞完从原来的汇到原来的源流一遍

55 2-SAT

每对点都选择强连通时 color 较小的

56 二分图

二分图最小覆盖集: 从右边的所有没有匹配过的点出发走增广路, 右边所有没有打上记号的点, 加上左边已经有记号的点.
最小覆盖数 = 最大匹配数.

57 Java

```
import java.io.*;
import java.util.*;
import java.math.*;
public class Main {
  public static void main(String[] args) {
    InputStream inputStream = System.in;
    OutputStream outputStream = System.out;
    InputReader in = new InputReader(inputStream);
    PrintWriter out = new PrintWriter(outputStream);
    Task solver = new Task();
    solver.solve(1, in, out);
    out.close();
  }
}
class Task {
  public void solve(int testNumber, InputReader in, PrintWriter out) {
  }
}
class InputReader {
  public BufferedReader reader;
  public StringTokenizer tokenizer;
  public InputReader(InputStream stream) {
    reader = new BufferedReader(new InputStreamReader(stream), 32768);
    tokenizer = null;
  }
  public String next() {
    while (tokenizer == null || !tokenizer.hasMoreTokens()) {
      try {
        tokenizer = new StringTokenizer(reader.readLine());
      } catch (IOException e) {
        throw new RuntimeException(e);
      }
    }
    return tokenizer.nextToken();
  }
  public int nextInt() {
    return Integer.parseInt(next());
  }
  public long nextLong() {
    return Long.parseLong(next());
  }
}
```

58 Rope

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