

Dracarys

Team Reference Library

May 11, 2016

Contents		FFT	21
多边形与圆面积交	2	NTT	22
二维几何	2	FWT	22
$n \log n$ 半平面交	4	Manacher/ 扩展 KMP	23
Delaunay 三角剖分	5	后缀数组 (倍增)	23
三维几何操作合并	7	后缀数组 (DC3)	24
三维凸包	8	后缀自动机	24
凸包上快速询问	9	后缀树 (With Walk)	24
圆的面积模板 ($n^2 \log n$)	10	后缀树 (With Pop Front)	27
三角形的心	11	字符串最小表示	29
最小覆盖球	12	轻重链剖分	29
经纬度求球面最短距离	12	KD Tree	30
长方体表面两点最短距离	12	Splay Tree	31
最大团	13	Link Cut Tree	31
极大团计数	13	Dominator Tree	32
KM	14	DancingLinks	33
最小树形图	15	弦图相关	34
无向图最小割	15	图同构 Hash	34
带花树	16	环状最长公共子序列	34
动态最小生成树	17	直线下有多少个格点	34
Hopcroft	17	费用流	35
素数判定	18	综合	35
启发式分解	18	积分表	36
二次剩余	19	Java	36
Pell 方程	19	Vimrc	37
日期公式	19		
Schreier-Sims	19		
Romberg	21		
线性规划	21		

多边形与圆面积交

```

1 double areaCT(Point pa, Point pb, double r) {
2     if (pa.len() < pb.len()) {
3         swap(pa, pb);
4     }
5     if (sign(pb.len()) == 0) {
6         return 0;
7     }
8     double a = pb.len(), b = pa.len(), c = (pb - pa).len();
9     double sinB = fabs(det(pb, pb - pa) / a / c),
10         cosB = dot(pb, pb - pa) / a / c,
11         sinC = fabs(det(pa, pb) / a / b),
12         cosC = dot(pa, pb) / a / b;
13     double B = atan2(sinB, cosB), C = atan2(sinC, cosC);
14     if (a > r) {
15         S = C / 2 * r * r;
16         h = a * b * sinC / c;
17         if (h < r && B < PI / 2) {
18             S -= (acos(h / r) * r * r - h * sqrt(r * r - h * h));
19         }
20     } else if (b > r) {
21         double theta = PI - B - asin(sinB / r * a);
22         S = a * r * sin(theta) / 2 + (C - theta) / 2 * r * r;
23     } else {
24         S = sinC * a * b / 2;
25     }
26     return S;
27 }

```

二维几何

```

1 #include <iostream>
2 #include <cmath>
3 #include <vector>
4
5 using namespace std;
6
7 const double PI = acos(-1.0);
8 const double EPS = 1e-8;
9
10 int sign(double x)
11 {
12     return x < -EPS ? -1 : x > EPS;
13 }
14
15 double newSqrt(double x)

```

```

16 {
17     return x < 0 ? 0 : sqrt(x);
18 }
19
20 struct Point {
21     double x, y;
22     Point(double x = 0, double y = 0) : x(x), y(y) {}
23     Point operator + (const Point &that) const {
24         return Point(x + that.x, y + that.y);
25     }
26     Point operator - (const Point &that) const {
27         return Point(x - that.x, y - that.y);
28     }
29     Point operator * (const double &that) const {
30         return Point(x * that, y * that);
31     }
32     Point operator / (const double &that) const {
33         return Point(x / that, y / that);
34     }
35     Point rotate(const double ang) { // 逆时针旋转 ang 弧度
36         return Point(cos(ang) * x - sin(ang) * y, cos(ang) * y + sin(ang) * x);
37     }
38     Point turn90() { // 逆时针旋转 90 度
39         return Point(-y, x);
40     }
41     double len2() const {
42         return x * x + y * y;
43     }
44     double len() const {
45         return sqrt(x * x + y * y);
46     }
47     Point unit() const {
48         return *this / len();
49     }
50     int operator < (const Point &that) const {
51         int d = sign(x - that.x); if (d) return d < 0;
52         return sign(y - that.y) < 0;
53     }
54 };
55 double det(Point a, Point b)
56 {
57     return a.x * b.y - b.x * a.y;
58 }
59 double dot(Point a, Point b)
60 {

```

```

61     return a.x * b.x + a.y * b.y;
62 }
63 double det(Point s, Point a, Point b)
64 {
65     return (a.x - s.x) * (b.y - s.y) - (b.x - s.x) * (a.y - s.y);
66 }
67
68 struct Line {
69     Point a, b;
70     Line(Point a, Point b) : a(a), b(b) {}
71 };
72
73 Point isLL(const Line &l1, const Line &l2) {
74     double s1 = det(l2.b - l2.a, l1.a - l2.a),
75            s2 = -det(l2.b - l2.a, l1.b - l2.a);
76     return (l1.a * s2 + l1.b * s1) / (s1 + s2);
77 }
78 bool onSeg(const Line &l, const Point &p) { // 点在线段上
79     return sign(det(p - l.a, l.b - l.a)) == 0 && sign(dot(p - l.a, p - l.b)) <= 0;
80 }
81 Point projection(const Line &l, const Point &p) { // 点到直线投影
82     return l.a + (l.b - l.a) * (dot(p - l.a, l.b - l.a) / (l.b - l.a).len2());
83 }
84 double disToLine(const Line &l, const Point &p) {
85     return abs(det(p - l.a, l.b - l.a) / (l.b - l.a).len());
86 }
87 double disToSeg(const Line &l, const Point &p) { // 点到线段距离
88     return sign(dot(p - l.a, l.b - l.a)) * sign(dot(p - l.b, l.a - l.b)) != 1 ?
89         disToLine(l, p) : min((p - l.a).len(), (p - l.b).len());
90 }
91 Point symmetryPoint(const Point a, const Point b) { // 点 b 关于点 a 的中心对称点
92     return a + a - b;
93 }
94 Point reflection(const Line &l, const Point &p) { // 点关于直线的对称点
95     return symmetryPoint(projection(l, p), p);
96 }
97 struct Circle {
98     Point o;
99     double r;
100     Circle(Point o = Point(0, 0), double r = 0) : o(o), r(r) {}
101 };
102
103 // 求圆与直线的交点
104 bool isCL(Circle a, Line l, Point &p1, Point &p2) {
105     if (sign(det(l.a - a.o, l.b - a.o) / (l.a - l.b).len()) > 0) return false;

```

```

106     Point o = isLL(Line(a.o, a.o + (l.b - l.a).turn90()), l);
107     Point delta = (l.b - l.a).unit() * newSqrt(a.r * a.r - (o - a.o).len2());
108     p1 = o + delta;
109     p2 = o - delta;
110     return true;
111 }
112
113 // 求圆与圆的交面积
114 double areaCC(const Circle &c1, const Circle &c2) {
115     double d = (c1.o - c2.o).len();
116     if (sign(d - (c1.r + c2.r)) >= 0) {
117         return 0;
118     }
119     if (sign(d - abs(c1.r - c2.r)) <= 0) {
120         double r = min(c1.r, c2.r);
121         return r * r * PI;
122     }
123     double x = (d * d + c1.r * c1.r - c2.r * c2.r) / (2 * d),
124            t1 = acos(x / c1.r), t2 = acos((d - x) / c2.r);
125     return c1.r * c1.r * t1 + c2.r * c2.r * t2 - d * c1.r * sin(t1);
126 }
127
128 // 求圆与圆的交点, 注意调用前要先判定重圆
129 bool isCC(Circle a, Circle b, Point &p1, Point &p2) {
130     double s1 = (a.o - b.o).len();
131     if (sign(s1 - a.r - b.r) > 0 || sign(s1 - abs(a.r - b.r)) < 0) return false;
132     double s2 = (a.r * a.r - b.r * b.r) / s1;
133     double aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
134     Point o = (b.o - a.o) * (aa / (aa + bb)) + a.o;
135     Point delta = (b.o - a.o).unit().turn90() * newSqrt(a.r * a.r - aa * aa);
136     p1 = o + delta, p2 = o - delta;
137     return true;
138 }
139
140 // 求点到圆的切点, 按关于点的左手方向返回两个点
141 bool tanCP(const Circle &c, const Point &p0, Point &p1, Point &p2)
142 {
143     double x = (p0 - c.o).len2(), d = x - c.r * c.r;
144     if (d < EPS) return false;
145     Point p = (p0 - c.o) * (c.r * c.r / x);
146     Point delta = ((p0 - c.o) * (-c.r * sqrt(d) / x)).turn90();
147     p1 = c.o + p + delta;
148     p2 = c.o + p - delta;
149     return true;
150 }

```

```

151
152 // 求圆到圆的外共切线, 按关于 c1.o 的左手方向返回两条线
153 vector<Line> extanCC(const Circle &c1, const Circle &c2)
154 {
155     vector<Line> ret;
156     if (sign(c1.r - c2.r) == 0) {
157         Point dir = c2.o - c1.o;
158         dir = (dir * (c1.r / dir.len())).turn90();
159         ret.push_back(Line(c1.o + dir, c2.o + dir));
160         ret.push_back(Line(c1.o - dir, c2.o - dir));
161     } else {
162         Point p = (c1.o * -c2.r + c2.o * c1.r) / (c1.r - c2.r);
163         Point p1, p2, q1, q2;
164         if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) {
165             if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
166             ret.push_back(Line(p1, q1));
167             ret.push_back(Line(p2, q2));
168         }
169     }
170     return ret;
171 }
172
173 // 求圆到圆的内共切线, 按关于 c1.o 的左手方向返回两条线
174 vector<Line> intanCC(const Circle &c1, const Circle &c2)
175 {
176     vector<Line> ret;
177     Point p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
178     Point p1, p2, q1, q2;
179     if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) {
180         ret.push_back(Line(p1, q1));
181         ret.push_back(Line(p2, q2));
182     }
183     return ret;
184 }
185
186 bool contain(vector<Point> polygon, Point p) { // 判断点 p
187     ↪ 是否被多边形包含, 包括落在边界上
188     int ret = 0, n = polygon.size();
189     for(int i = 0; i < n; ++i) {
190         Point u = polygon[i], v = polygon[(i + 1) % n];
191         if (onSeg(Line(u, v), p)) return true;
192         if (sign(u.y - v.y) <= 0) swap(u, v);
193         if (sign(p.y - u.y) > 0 || sign(p.y - v.y) <= 0) continue;
194         ret += sign(det(p, v, u)) > 0;
195     }

```

```

195     return ret & 1;
196 }
197
198 vector<Point> convexCut(const vector<Point>&ps, Line l) { // 用半平面 (q1,q2)
199     ↪ 的逆时针方向去切凸多边形
200     vector<Point> qs;
201     int n = ps.size();
202     for (int i = 0; i < n; ++i) {
203         Point p1 = ps[i], p2 = ps[(i + 1) % n];
204         int d1 = sign(det(l.a, l.b, p1)), d2 = sign(det(l.a, l.b, p2));
205         if (d1 >= 0) qs.push_back(p1);
206         if (d1 * d2 < 0) qs.push_back(isLL(Line(p1, p2), l));
207     }
208     return qs;
209 }
210 vector<Point> convexHull(vector<Point> ps) { // 求点集 ps 组成的凸包
211     int n = ps.size(); if (n <= 1) return ps;
212     sort(ps.begin(), ps.end());
213     vector<Point> qs;
214     for (int i = 0; i < n; qs.push_back(ps[i++]))
215         while (qs.size() > 1 && sign(det(qs[qs.size()-2], qs.back(), ps[i])) <= 0)
216             ↪ qs.pop_back();
217     for (int i = n - 2, t = qs.size(); i >= 0; qs.push_back(ps[i--]))
218         while ((int)qs.size() > t && sign(det(qs[(int)qs.size()-2], qs.back(), ps[i])) <=
219             ↪ 0) qs.pop_back();
220     qs.pop_back(); return qs;
221 }
222
223 int main()
224 {
225     Circle c1, c2;
226     c1.o = Point(0, 0); c1.r = 10;
227     c2.o = Point(10, 10); c1.r = 10;
228     Point p1, p2;
229     return 0;
230 }

```

$n \log n$ 半平面交

```

1 struct Point {
2     Point norm() const {
3         double l = len();
4         return Point(x / l, y / l);
5     }
6
7     int quad() const {

```

```

8     return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0);
9 }
10 };
11
12 struct Line {
13     bool include(const Point &p) const {
14         return sign(det(b - a, p - a)) > 0;
15     }
16
17     Line push() const{ // 将半平面向外推 eps
18         const double eps = 1e-6;
19         Point delta = (b - a).turn90().norm() * eps;
20         return Line(a - delta, b - delta);
21     }
22 };
23
24 bool sameDir(const Line &l0, const Line &l1) {
25     return parallel(l0, l1) && sign(dot(l0.b - l0.a, l1.b - l1.a)) == 1;
26 }
27
28 bool operator < (const Point &a, const Point &b) {
29     if (a.quad() != b.quad()) {
30         return a.quad() < b.quad();
31     } else {
32         return sign(det(a, b)) > 0;
33     }
34 }
35
36 bool operator < (const Line &l0, const Line &l1) {
37     if (sameDir(l0, l1)) {
38         return l1.include(l0.a);
39     } else {
40         return (l0.b - l0.a) < (l1.b - l1.a);
41     }
42 }
43
44 bool check(const Line &u, const Line &v, const Line &w) {
45     return w.include(intersect(u, v));
46 }
47
48 vector<Point> intersection(vector<Line> &l) {
49     sort(l.begin(), l.end());
50     deque<Line> q;
51     for (int i = 0; i < (int)l.size(); ++i) {
52         if (i && sameDir(l[i], l[i - 1])) {

```

```

53         continue;
54     }
55     while (q.size() > 1 && !check(q[q.size() - 2], q[q.size() - 1], l[i])) {
56         q.pop_back();
57     }
58     while (q.size() > 1 && !check(q[1], q[0], l[i])) {
59         q.pop_front();
60     }
61     q.push_back(l[i]);
62 }
63 while (q.size() > 2 && !check(q[q.size() - 2], q[q.size() - 1], q[0])) {
64     q.pop_back();
65 }
66 while (q.size() > 2 && !check(q[1], q[0], q[q.size() - 1])) {
67     q.pop_front();
68 }
69 vector<Point> ret;
70 for (int i = 0; i < (int)q.size(); ++i) {
71     ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
72 }
73 return ret;
74 }

```

Delaunay 三角剖分

```

1  /*
2  Delaunay Triangulation 随机增量算法 :
3  节点数至少为点数的 6 倍, 空间消耗较大注意计算内存使用
4  建图的过程在 build 中, 注意初始化内存池和初始三角形的坐标范围 (Triangulation::LOTS)
5  Triangulation::find 返回包含某点的三角形
6  Triangulation::add_point 将某点加入三角剖分
7  某个 Triangle 在三角剖分中当且仅当它的 has_children 为 0
8  如果要找到三角形 u 的邻域, 则枚举它的所有 u.edge[i].tri, 该条边的两个点为 u.p[(i+1)%3],
   ↪ u.p[(i+2)%3]
9  */
10 const int N = 100000 + 5;
11 const int MAX_TRIS = N * 6;
12 const double EPSILON = 1e-6;
13 const double PI = acos(-1.0);
14 using namespace std;
15
16 struct Point {
17     double x, y;
18     Point() : x(0), y(0) {}
19     Point(double x, double y) : x(x), y(y) {}
20     inline bool operator == (Point const& that) const {

```

```

21     return x == that.x && y == that.y;
22 }
23 };
24
25 inline double sqr(double x) { return x*x; }
26 double dist_sqr(Point const& a, Point const& b) {
27     return sqr(a.x-b.x) + sqr(a.y-b.y);
28 }
29 bool in_circumcircle(Point const& p1, Point const& p2, Point const& p3, Point const&
    ↪ p4) {
30     double u11 = p1.x - p4.x;
31     double u21 = p2.x - p4.x;
32     double u31 = p3.x - p4.x;
33     double u12 = p1.y - p4.y;
34     double u22 = p2.y - p4.y;
35     double u32 = p3.y - p4.y;
36     double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
37     double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
38     double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
39     double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33
    ↪ + u11*u22*u33;
40     return det > EPSILON;
41 }
42 double side(Point const& a, Point const& b, Point const& p) {
43     return (b.x-a.x)*(p.y-a.y) - (b.y-a.y)*(p.x-a.x);
44 }
45
46 typedef int SideRef;
47 struct Triangle;
48 typedef Triangle* TriangleRef;
49 struct Edge {
50     TriangleRef tri;
51     SideRef side;
52     Edge() : tri(0), side(0) {}
53     Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {}
54 };
55 struct Triangle {
56     Point p[3];
57     Edge edge[3];
58     TriangleRef children[3];
59     Triangle() {}
60     Triangle(Point const& p0, Point const& p1, Point const& p2) {
61         p[0] = p0; p[1] = p1; p[2] = p2;
62         children[0] = children[1] = children[2] = 0;
63     }

```

```

64 bool has_children() const {
65     return children[0] != 0;
66 }
67 int num_children() const {
68     return children[0] == 0 ? 0
69         : children[1] == 0 ? 1
70         : children[2] == 0 ? 2 : 3;
71 }
72 bool contains(Point const& q) const {
73     double a = side(p[0],p[1],q);
74     double b = side(p[1],p[2],q);
75     double c = side(p[2],p[0],q);
76     return a >= -EPSILON && b >= -EPSILON && c >= -EPSILON;
77 }
78 } triangle_pool[MAX_TRIS], *tot_triangles;
79 void set_edge(Edge a, Edge b) {
80     if (a.tri) a.tri->edge[a.side] = b;
81     if (b.tri) b.tri->edge[b.side] = a;
82     if (a.tri && b.tri) {
83         assert(a.tri->p[(a.side+1)%3] == b.tri->p[(b.side+2)%3]);
84         assert(a.tri->p[(a.side+2)%3] == b.tri->p[(b.side+1)%3]);
85     }
86 }
87 class Triangulation {
88 public:
89     Triangulation() {
90         const double LOTS = 1e6;
91         the_root = new(tot_triangles++)
    ↪ Triangle(Point(-LOTS,-LOTS),Point(+LOTS,-LOTS),Point(0,+LOTS));
92     }
93     ~Triangulation() {}
94     TriangleRef find(Point p) const {
95         return find(the_root,p);
96     }
97     void add_point(Point const& p) {
98         add_point(find(the_root,p),p);
99     }
100 private:
101     TriangleRef the_root;
102     static TriangleRef find(TriangleRef root, Point const& p) {
103         for( ; ; ) {
104             assert(root->contains(p));
105             if (!root->has_children()) {
106                 return root;
107             } else {

```

```

108     int flag = true;
109     for (int i = 0; i < 3 && root->children[i] ; ++i) {
110         if (root->children[i]->contains(p)) {
111             root = root->children[i];
112             break;
113         }
114     }
115     assert(flag&&"point not found");
116 }
117 }
118 }
119 void add_point(TriangleRef root, Point const& p) {
120     TriangleRef tab,tbc,tca;
121     /* split it into three triangles */
122     tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
123     tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
124     tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
125     set_edge(Edge(tab,0), Edge(tbc,1));
126     set_edge(Edge(tbc,0), Edge(tca,1));
127     set_edge(Edge(tca,0), Edge(tab,1));
128     set_edge(Edge(tab,2), root->edge[2]);
129     set_edge(Edge(tbc,2), root->edge[0]);
130     set_edge(Edge(tca,2), root->edge[1]);
131     root->children[0] = tab;
132     root->children[1] = tbc;
133     root->children[2] = tca;
134     flip(tab,2);
135     flip(tbc,2);
136     flip(tca,2);
137 }
138 void flip(TriangleRef tri, SideRef pi) {
139     TriangleRef trj = tri->edge[pi].tri;
140     int pj = tri->edge[pi].side;
141     if (!trj) return;
142     if (!in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
143     assert(tri->p[(pi+2)%3] == trj->p[(pj+1)%3]);
144     assert(tri->p[(pi+1)%3] == trj->p[(pj+2)%3]);
145     /* flip edge between tri,trj */
146     TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj],
147         ↪ tri->p[pi]);
148     TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi],
149         ↪ trj->p[pj]);
150     set_edge(Edge(trk,0), Edge(trl,0));
151     set_edge(Edge(trk,1), tri->edge[(pi+2)%3]);
152     set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);

```

```

151     set_edge(Edge(trl,1), trj->edge[(pj+2)%3]);
152     set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
153     tri->children[0] = trk; tri->children[1] = trl; tri->children[2] = 0;
154     trj->children[0] = trk; trj->children[1] = trl; trj->children[2] = 0;
155     flip(trk,1);
156     flip(trk,2);
157     flip(trl,1);
158     flip(trl,2);
159 }
160 };
161
162 int n;
163 Point ps[N];
164
165 void build()
166 {
167     tot_triangles = triange_pool;
168     cin >> n;
169     for(int i = 0; i < n; ++ i) {
170         int x, y;
171         scanf("%d%d", &x, &y);
172         ps[i].x = x; ps[i].y = y;
173     }
174     random_shuffle(ps, ps + n);
175     Triangulation tri;
176     for(int i = 0; i < n; ++ i) {
177         tri.add_point(ps[i]);
178     }
179 }
180
181 int main()
182 {
183     build();
184     return 0;
185 }

```

三维几何操作合并

```

1 struct Point3D {
2     double x, y, z;
3 };
4
5 Point3D det(const Point3D &a, const Point3D &b) {
6     return Point3D(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y *
7         ↪ b.x);
8 }

```



```

8 // 平面法向量 : 平面上两个向量叉积
9 // 点共平面 : 平面上一点与之的向量点积法向量为 0
10 // 点在线段 ( 直线 ) 上 : 共线且两边点积非正
11 // 点在三角形内 ( 不包含边界, 需再判断是与某条边共线 )
12 bool pointInTri(const Point3D &a, const Point3D &b, const Point3D &c, const Point3D
    ↪ &p) {
13     return sign(det(a - b, a - c).len() - det(p - a, p - b).len() - det(p - b, p -
    ↪ c).len() - det(p - c, p - a).len()) == 0;
14 }
15 // 共平面的两点是否在这平面上一条直线的同侧
16 bool sameSide(const Point3D &a, const Point3D &b, const Point3D &p0, const Point3D
    ↪ &p1) {
17     return sign(dot(det(a - b, p0 - b), det(a - b, p1 - b))) > 0;
18 }
19 // 两点在平面同侧 : 点积法向量符号相同
20 // 两直线平行 / 垂直 : 同二维
21 // 平面平行 / 垂直 : 判断法向量
22 // 线面垂直 : 法向量和直线平行
23 // 判断空间线段是否相交 : 四点共面两线段不平行相互在异侧
24 // 线段和三角形是否相交 : 线段在三角形平面不同侧
    ↪ 三角形任意两点在线段和第三点组成的平面的不同侧
25 // 求空间直线交点
26 Point3D intersection(const Point3D &a0, const Point3D &b0, const Point3D &a1, const
    ↪ Point3D &b1) {
27     double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) /
    ↪ ((a0.x - b0.x) * (a1.y - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));
28     return a0 + (b0 - a0) * t;
29 }
30 // 求平面和直线的交点
31 Point3D intersection(const Point3D &a, const Point3D &b, const Point3D &c, const
    ↪ Point3D &l0, const Point3D &l1) {
32     Point3D p = pVec(a, b, c); // 平面法向量
33     double t = (p.x * (a.x - l0.x) + p.y * (a.y - l0.y) + p.z * (a.z - l0.z)) / (p.x *
    ↪ (l1.x - l0.x) + p.y * (l1.y - l0.y) + p.z * (l1.z - l0.z));
34     return l0 + (l1 - l0) * t;
35 }
36 // 求平面交线 : 取不平行的一条直线的一个交点, 以及法向量叉积得到直线方向
37 // 点到直线距离 : 叉积得到三角形的面积除以底边
38 // 点到平面距离 : 点积法向量
39 // 直线间距离 : 平行时随便取一点求距离, 否则叉积方向向量得到方向点积计算长度
40 // 直线夹角 : 点积 平面夹角 : 法向量点积
41 // 三维向量旋转操作(绕向量 s 旋转 ang 角度), 对于右手系 s 指向观察者时逆时针
42 // 矩阵版
43 void rotate(const Point3D &s, double ang) {
44     double l = s.len(), x = s.x / l, y = s.y / l, z = s.z / l, sinA = sin(ang), cosA =
    ↪ cos(ang);

```

```

45     double p[4][4] = {CosA + (1 - CosA) * x * x, (1 - CosA) * x * y - SinA * z, (1 -
    ↪ CosA) * x * z + SinA * y, 0,
46     (1 - CosA) * y * x + SinA * z, CosA + (1 - CosA) * y * y, (1 - CosA) * y * z -
    ↪ SinA * x, 0,
47     (1 - CosA) * z * x - SinA * y, (1 - CosA) * z * y + SinA * x, CosA + (1 - CosA)
    ↪ * z * z, 0,
48     0, 0, 0, 1 };
49 }
50 // 计算版 : 把需要旋转的向量按照 s 分解, 做二维旋转, 再回到三维

```

三维凸包

```

1 #define SIZE(X) (int(X.size()))
2 #define PI 3.14159265358979323846264338327950288
3 struct Point {
4     Point cross(const Point &p) const
5     { return Point(y * p.z - z * p.y, z * p.x - x * p.z, x * p.y - y * p.x); }
6 } info[1005];
7 int mark[1005][1005], n, cnt;
8 double mix(const Point &a, const Point &b, const Point &c)
9 { return a.dot(b.cross(c)); }
10 double area(int a, int b, int c)
11 { return ((info[b] - info[a]).cross(info[c] - info[a])).length(); }
12 double volume(int a, int b, int c, int d)
13 { return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]); }
14 struct Face {
15     int a, b, c; Face() {}
16     Face(int a, int b, int c): a(a), b(b), c(c) {}
17     int &operator [](int k)
18     { if (k == 0) return a; if (k == 1) return b; return c; }
19 };
20 vector <Face> face;
21 inline void insert(int a, int b, int c) { face.push_back(Face(a, b, c)); }
22 void add(int v) {
23     vector <Face> tmp; int a, b, c; cnt++;
24     for (int i = 0; i < SIZE(face); i++) {
25         a = face[i][0]; b = face[i][1]; c = face[i][2];
26         if (Sign(volume(v, a, b, c)) < 0)
27             mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] =
    ↪ cnt;
28         else tmp.push_back(face[i]);
29     } face = tmp;
30     for (int i = 0; i < SIZE(tmp); i++) {
31         a = face[i][0]; b = face[i][1]; c = face[i][2];
32         if (mark[a][b] == cnt) insert(b, a, v);

```

```

33     if (mark[b][c] == cnt) insert(c, b, v);
34     if (mark[c][a] == cnt) insert(a, c, v);
35 }
36 int Find() {
37     for (int i = 2; i < n; i++) {
38         Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
39         if (ndir == Point()) continue; swap(info[i], info[2]);
40         for (int j = i + 1; j < n; j++) if (Sign(volume(0, 1, 2, j)) != 0) {
41             swap(info[j], info[3]); insert(0, 1, 2); insert(0, 2, 1); return 1;
42         } return 0; }
43 int main() {
44     for (; scanf("%d", &n) == 1; ) {
45         for (int i = 0; i < n; i++) info[i].Input();
46         sort(info, info + n); n = unique(info, info + n) - info;
47         face.clear(); random_shuffle(info, info + n);
48         if (Find()) { memset(mark, 0, sizeof(mark)); cnt = 0;
49             for (int i = 3; i < n; i++) add(i); vector<Point> Ndir;
50             for (int i = 0; i < SIZE(face); ++i) {
51                 Point p = (info[face[i][0]] - info[face[i][1]]).cross(
52                     info[face[i][2]] - info[face[i][1]]);
53                 p = p / p.length(); Ndir.push_back(p);
54             } sort(Ndir.begin(), Ndir.end());
55             int ans = unique(Ndir.begin(), Ndir.end()) - Ndir.begin();
56             printf("%d\n", ans);
57         } else printf("1\n");
58     }
59     // 求重心
60     double calcDist(const Point &p, int a, int b, int c)
61     { return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c)); }
62     //compute the minimal distance of center of any faces
63     double findDist() { //compute center of mass
64         double totalWeight = 0; Point center(.0, .0, .0);
65         Point first = info[face[0][0]];
66         for (int i = 0; i < SIZE(face); ++i) {
67             Point p = (info[face[i][0]]+info[face[i][1]]+info[face[i][2]]+first)*.25;
68             double weight = mix(info[face[i][0]] - first, info[face[i][1]]
69                 - first, info[face[i][2]] - first);
70             totalWeight += weight; center = center + p * weight;
71         } center = center / totalWeight;
72         double res = 1e100; //compute distance
73         for (int i = 0; i < SIZE(face); ++i)
74             res = min(res, calcDist(center, face[i][0], face[i][1], face[i][2]));
75         return res; }

```

凸包上快速询问

```

1  /*
2   给定凸包, log n 内完成各种询问, 具体操作有 :
3   1. 判定一个点是否在凸包内
4   2. 询问凸包外的点到凸包的两个切点
5   3. 询问一个向量关于凸包的切点
6   4. 询问一条直线和凸包的交点
7   INF 为坐标范围, 需要定义点类大于号
8   改成实数只需修改 sign 函数, 以及把 long long 改为 double 即可
9   构造函数时传入凸包要求无重点, 面积非空, 以及 pair(x,y) 的最小点放在第一个
10 */
11 #include <vector>
12 #include <functional>
13 using namespace std;
14
15 const int INF = 1000000000;
16
17 struct Convex
18 {
19     int n;
20     vector<Point> a;
21     vector<Point> upper, lower;
22     Convex(vector<Point> _a) : a(_a) {
23         n = a.size();
24         int ptr = 0;
25         for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
26         for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
27         for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
28         upper.push_back(a[0]);
29     }
30     int sign(long long x) {
31         return x < 0 ? -1 : x > 0;
32     }
33     pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
34         int l = 0, r = (int)convex.size() - 2;
35         for( ; l + 1 < r; ) {
36             int mid = (l + r) / 2;
37             if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
38             else l = mid;
39         }
40         return max(make_pair(vec.det(convex[r]), r), make_pair(vec.det(convex[0]), 0));
41     }
42     void update_tangent(const Point &p, int id, int &i0, int &i1) {
43         if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
44         if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;

```

```

45 }
46 void binary_search(int l, int r, Point p, int &i0, int &i1) {
47     if (l == r) return;
48     update_tangent(p, l % n, i0, i1);
49     int s1 = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
50     for( ; l + 1 < r; ) {
51         int mid = (l + r) / 2;
52         int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
53         if (smid == s1) l = mid;
54         else r = mid;
55     }
56     update_tangent(p, r % n, i0, i1);
57 }
58 int binary_search(Point u, Point v, int l, int r) {
59     int s1 = sign((v - u).det(a[l % n] - u));
60     for( ; l + 1 < r; ) {
61         int mid = (l + r) / 2;
62         int smid = sign((v - u).det(a[mid % n] - u));
63         if (smid == s1) l = mid;
64         else r = mid;
65     }
66     return l % n;
67 }
68 // 判定点是否在凸包内, 在边界返回 true
69 bool contain(Point p) {
70     if (p.x < lower[0].x || p.x > lower.back().x) return false;
71     int id = lower_bound(lower.begin(), lower.end(), Point(p.x, -INF)) -
        ↪ lower.begin();
72     if (lower[id].x == p.x) {
73         if (lower[id].y > p.y) return false;
74     } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
75     id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF), greater<Point>()) -
        ↪ - upper.begin();
76     if (upper[id].x == p.x) {
77         if (upper[id].y < p.y) return false;
78     } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
79     return true;
80 }
81 // 求点 p 关于凸包的两个切点, 如果在凸包外则有序返回编号, 多解返回任意一个 否则返回
    ↪ false
82 bool get_tangent(Point p, int &i0, int &i1) {
83     if (contain(p)) return false;
84     i0 = i1 = 0;
85     int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
86     binary_search(0, id, p, i0, i1);

```

```

87     binary_search(id, (int)lower.size(), p, i0, i1);
88     id = lower_bound(upper.begin(), upper.end(), p, greater<Point>()) -
        ↪ upper.begin();
89     binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
90     binary_search((int)lower.size() - 1 + id, (int)lower.size() - 1 +
        ↪ (int)upper.size(), p, i0, i1);
91     return true;
92 }
93 // 求凸包上和向量 vec 叉积最大的点, 返回编号, 有多个返回任意一个
94 int get_tangent(Point vec) {
95     pair<long long, int> ret = get_tangent(upper, vec);
96     ret.second = (ret.second + (int)lower.size() - 1) % n;
97     ret = max(ret, get_tangent(lower, vec));
98     return ret.second;
99 }
100 // 求凸包和直线 u,v 的交点, 如果无严格相交返回 false 。如果有则是和 (i,next(i))
    ↪ 的交点, 两个点无序, 交在点上不确定返回两条线段之一。
101 bool get_intersection(Point u, Point v, int &i0, int &i1) {
102     int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
103     if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
104         if (p0 > p1) swap(p0, p1);
105         i0 = binary_search(u, v, p0, p1);
106         i1 = binary_search(u, v, p1, p0 + n);
107         return true;
108     } else {
109         return false;
110     }
111 }
112 };

```

圆的面积模板 ($n^2 \log n$)

```

1 struct Event {
2     Point p;
3     double ang;
4     int delta;
5     Event (Point p = Point(0, 0), double ang = 0, double delta = 0) : p(p), ang(ang),
        ↪ delta(delta) {}
6 };
7
8 bool operator < (const Event &a, const Event &b) {
9     return a.ang < b.ang;
10 }
11
12 void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
13     double d2 = (a.o - b.o).len2(),

```

```

14     dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
15     pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 *
    ↪ 4));
16 Point d = b.o - a.o, p = d.rotate(PI / 2),
17     q0 = a.o + d * dRatio + p * pRatio,
18     q1 = a.o + d * dRatio - p * pRatio;
19 double ang0 = (q0 - a.o).ang(),
20     ang1 = (q1 - a.o).ang();
21 evt.push_back(Event(q1, ang1, 1));
22 evt.push_back(Event(q0, ang0, -1));
23 cnt += ang1 > ang0;
24 }
25
26 bool issame(const Circle &a, const Circle &b) {
27     return sign((a.o - b.o).len()) == 0 && sign(a.r - b.r) == 0;
28 }
29
30 bool overlap(const Circle &a, const Circle &b) {
31     return sign(a.r - b.r - (a.o - b.o).len()) >= 0;
32 }
33
34 bool intersect(const Circle &a, const Circle &b) {
35     return sign((a.o - b.o).len() - a.r - b.r) < 0;
36 }
37
38 int C;
39 Circle c[N];
40 double area[N];
41
42 void solve() {
43     memset(area, 0, sizeof(double) * (C + 1));
44     for (int i = 0; i < C; ++i) {
45         int cnt = 1;
46         vector<Event> evt;
47         for (int j = 0; j < i; ++j) {
48             if (issame(c[i], c[j])) {
49                 ++cnt;
50             }
51         }
52         for (int j = 0; j < C; ++j) {
53             if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) {
54                 ++cnt;
55             }
56         }
57         for (int j = 0; j < C; ++j) {

```

```

58             if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i],
    ↪ c[j])) {
59                 addEvent(c[i], c[j], evt, cnt);
60             }
61         }
62         if (evt.size() == 0) {
63             area[cnt] += PI * c[i].r * c[i].r;
64         } else {
65             sort(evt.begin(), evt.end());
66             evt.push_back(evt.front());
67             for (int j = 0; j + 1 < (int)evt.size(); ++j) {
68                 cnt += evt[j].delta;
69                 area[cnt] += det(evt[j].p, evt[j + 1].p) / 2;
70                 double ang = evt[j + 1].ang - evt[j].ang;
71                 if (ang < 0) {
72                     ang += PI * 2;
73                 }
74                 area[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2;
75             }
76         }
77     }
78 }

```

三角形的心

```

1 Point inCenter(const Point &A, const Point &B, const Point &C) { // 内心
2     double a = (B - C).len(), b = (C - A).len(), c = (A - B).len(),
3         s = fabs(det(B - A, C - A)),
4         r = s / p;
5     return (A * a + B * b + C * c) / (a + b + c);
6 }
7
8 Point circumCenter(const Point &a, const Point &b, const Point &c) { // 外心
9     Point bb = b - a, cc = c - a;
10    double db = bb.len2(), dc = cc.len2(), d = 2 * det(bb, cc);
11    return a - Point(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
12 }
13
14 Point othroCenter(const Point &a, const Point &b, const Point &c) { // 垂心
15     Point ba = b - a, ca = c - a, bc = b - c;
16     double Y = ba.y * ca.y * bc.y,
17         A = ca.x * ba.y - ba.x * ca.y,
18         x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
19         y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
20     return Point(x0, y0);
21 }

```

最小覆盖球

```

1 int npoint, nouter; Tpoint pt[200000], outer[4], res; double radius, tmp;
2 void ball() {
3     Tpoint q[3]; double m[3][3], sol[3], L[3], det;
4     int i, j; res.x = res.y = res.z = radius = 0;
5     switch ( nouter ) {
6     case 1: res=outer[0]; break;
7     case 2: res=(outer[0]+outer[1])/2; radius=dist2(res, outer[0]); break;
8     case 3:
9         for (i=0; i<2; ++i) q[i]=outer[i+1]-outer[0];
10        for (i=0; i<2; ++i) for(j=0; j<2; ++j) m[i][j]=dot(q[i], q[j])*2;
11        for (i=0; i<2; ++i) sol[i]=dot(q[i], q[i]);
12        if (fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0])<eps) return;
13        L[0]=(sol[0]*m[1][1]-sol[1]*m[0][1])/det;
14        L[1]=(sol[1]*m[0][0]-sol[0]*m[1][0])/det;
15        res=outer[0]+q[0]*L[0]+q[1]*L[1];
16        radius=dist2(res, outer[0]);
17        break;
18    case 4:
19        for (i=0; i<3; ++i) q[i]=outer[i+1]-outer[0], sol[i]=dot(q[i], q[i]);
20        for (i=0; i<3; ++i) for(j=0; j<3; ++j) m[i][j]=dot(q[i], q[j])*2;
21        det= m[0][0]*m[1][1]*m[2][2]
22            + m[0][1]*m[1][2]*m[2][0]
23            + m[0][2]*m[2][1]*m[1][0]
24            - m[0][2]*m[1][1]*m[2][0]
25            - m[0][1]*m[1][0]*m[2][2]
26            - m[0][0]*m[1][2]*m[2][1];
27        if ( fabs(det)<eps ) return;
28        for (j=0; j<3; ++j) {
29            for (i=0; i<3; ++i) m[i][j]=sol[i];
30            L[j]=( m[0][0]*m[1][1]*m[2][2]
31                + m[0][1]*m[1][2]*m[2][0]
32                + m[0][2]*m[2][1]*m[1][0]
33                - m[0][2]*m[1][1]*m[2][0]
34                - m[0][1]*m[1][0]*m[2][2]
35                - m[0][0]*m[1][2]*m[2][1]
36                ) / det;
37            for (i=0; i<3; ++i) m[i][j]=dot(q[i], q[j])*2;
38        } res=outer[0];
39        for (i=0; i<3; ++i) res = res + q[i] * L[i];
40        radius=dist2(res, outer[0]);
41    }
42    void minball(int n) { ball();
43        if ( nouter<4 ) for (int i=0; i<n; ++i)
44        if (dist2(res, pt[i])-radius>eps) {

```

```

45        outer[nouter++]=pt[i]; minball(i); --nouter;
46        if (i>0) { Tpoint Tt = pt[i];
47            memmove(&pt[1], &pt[0], sizeof(Tpoint)*i); pt[0]=Tt;
48        }
49    int main0(){
50        scanf("%d", &npoint);
51        for (int i=0; i<npoint; i++) scanf("%lf%lf%lf", &pt[i].x, &pt[i].y, &pt[i].z);
52        random_shuffle(pt, pt+npoint); radius=-1;
53        for (int i=0; i<npoint; i++) if (dist2(res, pt[i])-radius>eps)
54            nouter=1, outer[0]=pt[i], minball(i);
55        printf("%.5f\n", sqrt(radius));
56    }

```

经纬度求球面最短距离

```

1 //lati 为纬度 longi 为经度 R 为半径
2 double Dist(double lati1, double longi1, double lati2, double longi2, double R) {
3     double pi=acos(-1.0); lati1*=pi/180, longi1*=pi/180, lati2*=pi/180, longi2*=pi/180;
4     double x1=cos(lati1)*sin(longi1), y1=cos(lati1)*cos(longi1), z1=sin(lati1);
5     double x2=cos(lati2)*sin(longi2), y2=cos(lati2)*cos(longi2), z2=sin(lati2);
6     double theta=acos(x1*x2+y1*y2+z1*z2); return(R*theta);
7 }

```

长方体表面两点最短距离

```

1 int r;
2 void turn(int i, int j, int x, int y, int z, int x0, int y0, int L, int W, int H) {
3     if (z==0) { int R = x*x+y*y; if (R<r) r=R;
4     } else {
5         if(i>=0 && i<2) turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);
6         if(j>=0 && j<2) turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);
7         if(i<=0 && i>-2) turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
8         if(j<=0 && j>-2) turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
9     }
10    int main(){
11        int L, H, W, x1, y1, z1, x2, y2, z2;
12        cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
13        if (z1!=0 && z1!=H) if (y1==0 || y1==W)
14            swap(y1, z1), std::swap(y2, z2), std::swap(W, H);
15        else swap(x1, z1), std::swap(x2, z2), std::swap(L, H);
16        if (z1==H) z1=0, z2=H-z2;
17        r=0x3fffffff; turn(0, 0, x2-x1, y2-y1, z2, -x1, -y1, L, W, H);
18        cout<<r<<endl; return 0;
19    }

```

最大团

```

1 // Super Fast Maximum Clique
2 // To Build Graph: Maxclique(Edges, Number of Nodes)
3 // To Get Answer: mcqdyn(AnswerNodes Index Array, AnswerLength)
4 typedef bool BB[N];
5 struct Maxclique {
6     const BB* e; int pk, level; const float Tlimit;
7     struct Vertex{ int i, d; Vertex(int i):i(i),d(0){} };
8     typedef vector<Vertex> Vertices; typedef vector<int> ColorClass;
9     Vertices V; vector<ColorClass> C; ColorClass QMAX, Q;
10    static bool desc_degree(const Vertex &vi, const Vertex &vj){
11        return vi.d > vj.d;
12    }
13    void init_colors(Vertices &v){
14        const int max_degree = v[0].d;
15        for(int i = 0; i < (int)v.size(); i++) v[i].d = min(i, max_degree) + 1;
16    }
17    void set_degrees(Vertices &v){
18        for(int i = 0, j; i < (int)v.size(); i++)
19            for(v[i].d = j = 0; j < int(v.size()); j++)
20                v[i].d += e[v[i].i][v[j].i];
21    }
22    struct StepCount{ int i1, i2; StepCount():i1(0),i2(0){} };
23    vector<StepCount> S;
24    bool cut1(const int pi, const ColorClass &A){
25        for(int i = 0; i < (int)A.size(); i++) if (e[pi][A[i]]) return true;
26        return false;
27    }
28    void cut2(const Vertices &A, Vertices &B){
29        for(int i = 0; i < (int)A.size() - 1; i++)
30            if(e[A.back().i][A[i].i])
31                B.push_back(A[i].i);
32    }
33    void color_sort(Vertices &R){
34        int j = 0, maxno = 1, min_k = max((int)QMAX.size() - (int)Q.size() + 1, 1);
35        C[1].clear(), C[2].clear();
36        for(int i = 0; i < (int)R.size(); i++) {
37            int pi = R[i].i, k = 1;
38            while(cut1(pi, C[k])) k++;
39            if(k > maxno) maxno = k, C[maxno + 1].clear();
40            C[k].push_back(pi);
41            if(k < min_k) R[j++].i = pi;
42        }
43        if(j > 0) R[j - 1].d = 0;
44        for(int k = min_k; k <= maxno; k++)

```

```

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

```

极大团计数

```

1 //Bool g[][] 为图的邻接矩阵, 图点的标号由 1 至 n
2 void dfs(int size){
3     int i, j, k, t, cnt, best = 0;
4     if (ne[size]==ce[size]){ if (ce[size]==0) ++ans; return; }
5     for (t=0, i=1; i<=ne[size]; ++i) {

```

```

6   for (cnt=0, j=ne[size]+1; j<=ce[size]; ++j)
7   if (!g[list[size][i]][list[size][j]]) ++cnt;
8   if (t==0 || cnt<best) t=i, best=cnt;
9   } if (t && best<=0) return;
10  for (k=ne[size]+1; k<=ce[size]; ++k) {
11  if (t>0){ for (i=k; i<=ce[size]; ++i)
12  if (!g[list[size][t]][list[size][i]]) break;
13  swap(list[size][k], list[size][i]);
14  } i=list[size][k]; ne[size+1]=ce[size+1]=0;
15  for (j=1; j<k; ++j)if (g[i][list[size][j]])
16  list[size+1][++ne[size+1]]=list[size][j];
17  for (ce[size+1]=ne[size+1], j=k+1; j<=ce[size]; ++j)
18  if (g[i][list[size][j]]) list[size+1][++ce[size+1]]=list[size][j];
19  dfs(size+1); ++ne[size]; --best;
20  for (j=k+1, cnt=0; j<=ce[size]; ++j) if (!g[i][list[size][j]]) ++cnt;
21  if (t==0 || cnt<best) t=k, best=cnt;
22  if (t && best<=0) break;
23  }}
24  void work(){
25  ne[0]=0; ce[0]=0;
26  for (int i=1; i<=n; ++i) list[0][++ce[0]]=i;
27  ans=0; dfs(0);
28  }

```

KM

```

1  // 最小匹配, 自带初始化 n <= m 方案存在 p[] 中
2
3  const int N = 105;
4
5  const int INF = 1000000000; // 严格大于最大边权
6
7  int n, m, a[N][N];
8
9  int u[N], v[N], p[N], fa[N];
10
11 int minv[N];
12
13 bool used[N];
14
15 int km() {
16   for (int i = 0; i < n; ++i) {
17     u[i] = 0;
18   }
19   for (int i = 0; i <= m; ++i) {
20     v[i] = 0;

```

```

21   p[i] = n;
22   }
23   for (int i = 0; i < n; ++i) {
24     p[m] = i;
25     int j0 = m;
26     for (int j = 0; j <= m; ++j) {
27       minv[j] = INF;
28       used[j] = false;
29     }
30     do {
31       used[j0] = true;
32       int i0 = p[j0], delta = INF, j1;
33       for (int j = 0; j < m; ++j) {
34         if (!used[j]) {
35           int cur = a[i0][j] - u[i0] - v[j];
36           if (cur < minv[j]) {
37             minv[j] = cur;
38             fa[j] = j0;
39           }
40           if (minv[j] < delta) {
41             delta = minv[j];
42             j1 = j;
43           }
44         }
45       }
46       for (int j = 0; j <= m; ++j) {
47         if (used[j]) {
48           u[p[j]] += delta;
49           v[j] -= delta;
50         } else {
51           minv[j] -= delta;
52         }
53       }
54       j0 = j1;
55     } while (p[j0] != n);
56     do {
57       int j1 = fa[j0];
58       p[j0] = p[j1];
59       j0 = j1;
60     } while (j0 != m);
61   }
62   return -v[m];
63 }

```


最小树形图

```

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

```

```

45         }
46         fa[limit] = vis[limit] = limit;
47         for(int i = 0; i < int(C.size()); ++i){
48             int x = C[i];
49             fa[x] = limit;
50             for(int j = 1; j <= n; ++j){
51                 if(edge[j][x] != INF && edge[j][limit] > edge[j][x] - edge[sel[x]][x]){
52                     edge[j][limit] = edge[j][x] - edge[sel[x]][x];
53                     from[j][limit] = x;
54                 }
55             }
56         }
57         for(int j = 1; j <= n; ++j) if(getfa(j) == limit){
58             edge[j][limit] = INF;
59         }
60         sel[limit] = 1;
61         for(int j = 1; j <= n; ++j){
62             if(edge[sel[limit]][limit] > edge[j][limit]){
63                 sel[limit] = j;
64             }
65         }
66     }
67     if(prelimit == limit) break;
68 }
69 for(int i = limit; i > 1; --i){
70     sel[from[sel[i]][i]] = sel[i];
71 }
72 }
73 }

```

无向图最小割

```

1 int cost[maxn][maxn],seq[maxn],len[maxn],n,m,pop,ans;
2 bool used[maxn];
3 void Init(){
4     int i,j,a,b,c;
5     for(i=0;i<n;i++) for(j=0;j<n;j++) cost[i][j]=0;
6     for(i=0;i<m;i++){
7         scanf("%d %d %d",&a,&b,&c); cost[a][b]+=c; cost[b][a]+=c;
8     }
9     pop=n; for(i=0;i<n;i++) seq[i]=i;
10 }
11 void Work(){
12     ans=inf; int i,j,k,l,mm,sum,pk;
13     while(pop > 1){
14         for(i=1;i<pop;i++) used[seq[i]]=0; used[seq[0]]=1;

```



```

15   for(i=1;i<pop;i++) len[seq[i]]=cost[seq[0]][seq[i]];
16   pk=0; mm=-inf; k=-1;
17   for(i=1;i<pop;i++) if(len[seq[i]] > mm){ mm=len[seq[i]]; k=i; }
18   for(i=1;i<pop;i++){
19       used[seq[l=k]]=1;
20       if(i==pop-2) pk=k;
21       if(i==pop-1) break;
22       mm=-inf;
23       for(j=1;j<pop;j++) if(!used[seq[j]])
24           if((len[seq[j]]+cost[seq[l]][seq[j]]) > mm)
25               mm=len[seq[j]], k=j;
26   }
27   sum=0;
28   for(i=0;i<pop;i++) if(i != k) sum+=cost[seq[k]][seq[i]];
29   ans=min(ans,sum);
30   for(i=0;i<pop;i++)
31       cost[seq[k]][seq[i]]=cost[seq[i]][seq[k]]+cost[seq[pk]][seq[i]];
32   seq[pk]=seq[--pop];
33 }
34 printf("%d\n",ans);
35 }

```

带花树

```

1  vector<int> link[maxn];
2  int n,match[maxn],Queue[maxn],head,tail;
3  int pred[maxn],base[maxn],start,finish,newbase;
4  bool InQueue[maxn],InBlossom[maxn];
5  void push(int u){ Queue[tail++]=u;InQueue[u]=true; }
6  int pop(){ return Queue[head++]; }
7  int FindCommonAncestor(int u,int v){
8      bool InPath[maxn];
9      for(int i=0;i<n;i++) InPath[i]=0;
10     while(true){ u=base[u];InPath[u]=true;if(u==start) break;u=pred[match[u]]; }
11     while(true){ v=base[v];if(InPath[v]) break;v=pred[match[v]]; }
12     return v;
13 }
14 void ResetTrace(int u){
15     int v;
16     while(base[u]!=newbase){
17         v=match[u];
18         InBlossom[base[u]]=InBlossom[base[v]]=true;
19         u=pred[v];
20         if(base[u]!=newbase) pred[u]=v;
21     }
22 }

```

```

23 void BlossomContract(int u,int v){
24     newbase=FindCommonAncestor(u,v);
25     for (int i=0;i<n;i++)
26         InBlossom[i]=0;
27     ResetTrace(u);ResetTrace(v);
28     if(base[u]!=newbase) pred[u]=v;
29     if(base[v]!=newbase) pred[v]=u;
30     for(int i=0;i<n;i++){
31         if(InBlossom[base[i]]){
32             base[i]=newbase;
33             if(!InQueue[i]) push(i);
34         }
35     }
36 bool FindAugmentingPath(int u){
37     bool found=false;
38     for(int i=0;i<n;i++) pred[i]=-1,base[i]=i;
39     for (int i=0;i<n;i++) InQueue[i]=0;
40     start=u;finish=-1; head=tail=0; push(start);
41     while(head<tail){
42         int u=pop();
43         for(int i=link[u].size()-1;i>=0;i--){
44             int v=link[u][i];
45             if(base[u]!=base[v]&&match[u]!=v)
46                 if(v==start||(match[v]>=0&&pred[match[v]]>=0))
47                     BlossomContract(u,v);
48             else if(pred[v]==-1){
49                 pred[v]=u;
50                 if(match[v]>=0) push(match[v]);
51                 else{ finish=v; return true; }
52             }
53         }
54     }
55     return found;
56 }
57 void AugmentPath(){
58     int u=finish,v,w;
59     while(u>=0){ v=pred[u];w=match[v];match[v]=u;match[u]=v;u=w; }
60 }
61 void FindMaxMatching(){
62     for(int i=0;i<n;i++) match[i]=-1;
63     for(int i=0;i<n;i++) if(match[i]==-1) if(FindAugmentingPath(i)) AugmentPath();
64 }

```

动态最小生成树

```

1  /* 动态最小生成树  $Q(\log Q)^2$ 
2   (qx[i],qy[i]) 表示将编号为 qx[i] 的边的权值改为 qy[i]
3   删除一条边相当于将其权值改为  $\infty$ 
4   加入一条边相当于将其权值从  $\infty$  变成某个值 */
5  const int qsize=maxm+3*maxq;
6  int x[qsize],y[qsize],z[qsize],qx[maxq],qy[maxq],n,m,Q;
7  void init(){
8      scanf("%d%d",&n,&m);
9      for(int i=0;i<m;i++) scanf("%d%d%d",x+i,y+i,z+i);
10     scanf("%d",&Q);
11     for(int i=0;i<Q;i++){ scanf("%d%d",qx+i,qy+i); qx[i]--; }
12 }
13 int a[maxn],*tz;
14 int find(int x){
15     int root=x; while(a[root]) root=a[root];
16     int next; while(next=a[x]){ a[x]=root; x=next; }
17     return root;
18 }
19 inline bool cmp(const int &a,const int &b){ return tz[a]<tz[b]; }
20 int kx[maxn],ky[maxn],kt, vd[maxn],id[maxm], app[maxm];
21 bool extra[maxm];
22 void solve(int *qx,int *qy,int Q,int n,int *x,int *y,int *z,int m,long long ans){
23     if(Q==1){
24         for(int i=1;i<=n;i++) a[i]=0;
25         z[ qx[0] ]=qy[0];
26         for(int i=0;i<m;i++) id[i]=i;tz=z;
27         sort(id,id+m,cmp); int ri,rj;
28         for(int i=0;i<m;i++){
29             ri=find(x[id[i]]); rj=find(y[id[i]]);
30             if(ri!=rj){ ans+=z[id[i]]; a[ri]=rj; }
31         }
32         printf("%I64d\n",ans);
33         return;
34     }
35     int ri,rj;
36     //contract
37     kt=0;
38     for(int i=1;i<=n;i++) a[i]=0;
39     for(int i=0;i<Q;i++){
40         ri=find(x[qx[i]]); rj=find(y[qx[i]]); if(ri!=rj) a[ri]=rj;
41     }
42     int tm=0;
43     for(int i=0;i<m;i++) extra[i]=true;
44     for(int i=0;i<Q;i++) extra[ qx[i] ]=false;

```

```

45     for(int i=0;i<m;i++) if(extra[i]) id[tm++]=i;
46     tz=z; sort(id,id+tm,cmp);
47     for(int i=0;i<tm;i++){
48         ri=find(x[id[i]]); rj=find(y[id[i]]);
49         if(ri!=rj){
50             a[ri]=rj; ans += z[id[i]];
51             kx[kt]=x[id[i]]; ky[kt]=y[id[i]]; kt++;
52         }
53     }
54     for(int i=1;i<=n;i++) a[i]=0;
55     for(int i=0;i<kt;i++) a[ find(kx[i]) ]=find(ky[i]);
56     int n2=0;
57     for(int i=1;i<=n;i++) if(a[i]==0)
58         vd[i]=++n2;
59     for(int i=1;i<=n;i++) if(a[i])
60         vd[i]=vd[find(i)];
61     int m2=0, *Nx=x+m, *Ny=y+m, *Nz=z+m;
62     for(int i=0;i<m;i++) app[i]=-1;
63     for(int i=0;i<Q;i++) if(app[qx[i]]==-1){
64         Nx[m2]=vd[ x[ qx[i] ] ]; Ny[m2]=vd[ y[ qx[i] ] ]; Nz[m2]=z[ qx[i] ];
65         app[qx[i]]=m2; m2++;
66     }
67     for(int i=0;i<Q;i++){ z[ qx[i] ]=qy[i]; qx[i]=app[qx[i]]; }
68     for(int i=1;i<=n2;i++) a[i]=0;
69     for(int i=0;i<tm;i++){
70         ri=find(vd[ x[id[i]] ]); rj=find(vd[ y[id[i]] ]);
71         if(ri!=rj){
72             a[ri]=rj; Nx[m2]=vd[ x[id[i]] ];
73             Ny[m2]=vd[ y[id[i]] ]; Nz[m2]=z[id[i]]; m2++;
74         }
75     }
76     int mid=Q/2;
77     solve(qx,qy,mid,n2,Nx,Ny,Nz,m2,ans);
78     solve(qx+mid,qy+mid,Q-mid,n2,Nx,Ny,Nz,m2,ans);
79 }
80 void work(){ if(Q) solve(qx,qy,Q,n,x,y,z,m,0); }
81 int main(){init(); work(); return 0; }

```

Hopcroft

```

1  int from[1010],wh[1010],g[1010];
2  int num[100010],nxt[100010],tot;
3  int n,m,ans,h,t,q[1010],dx[1010],dy[1010];
4  bool bfs(){
5      bool ret=false;
6      h=0;t=0;

```

```

7   for(int i=0;i<n;i++) if(wh[i]==-1) t++, q[t]=i;
8   memset(dx,0,sizeof(dx)), memset(dy,0,sizeof(dy));
9   while(h++<t){
10      for(int i=g[q[h]];i!=0;i=nxt[i])
11         if(dy[num[i]]==0){
12            dy[num[i]]=dx[q[h]]+1;
13            if(from[num[i]]==-1) ret=true;
14            else{
15               dx[from[num[i]]]=dx[q[h]]+2;
16               q[++t]=from[num[i]];
17            }
18         }
19      }
20      return ret;
21 }
22 bool dfs(int x){
23     for(int i=g[x];i!=0;i=nxt[i]){
24         if(dy[num[i]]==dx[x]+1){
25             dy[num[i]]=0;
26             if(from[num[i]]==-1||dfs(from[num[i]])){
27                 wh[x]=num[i];from[num[i]]=x;return true;
28             }
29         }
30     }
31     return false;
32 }
33 void hopcroft(){
34     memset(from,-1,sizeof(from)), memset(wh,-1,sizeof(wh));
35     while(bfs()){
36         for(int i=0;i<n;i++)
37             if(wh[i]==-1&&dfs(i)) ans++;
38     }
39     void insert(int x,int y){ tot++;num[tot]=y;nxt[tot]=g[x];g[x]=tot; }
40     int main(){
41         while(scanf("%d %d",&n,&m)==2){
42             tot=0; memset(g,0,sizeof(g));
43             for(int i=0;i<n;i++){
44                 int x; scanf("%d",&x);
45                 for(int j=0;j<x;j++){
46                     int y; scanf("%d",&y);
47                     y--; insert(i,y);
48                 }
49             }
50             ans=0; hopcroft(); printf("%d\n",ans);
51         }

```

52 }

素数判定

```

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

```

启发式分解

```

1  int ansn; LL ans[1000];
2  LL func(LL x,LL n){ return(mod_mul(x,x,n)+1)%n; }
3  LL Pollard(LL n){
4      LL i,x,y,p;
5      if(Rabin_Miller(n)) return n;
6      if(!(n&1)) return 2;
7      for(i=1;i<20;i++){
8          x=i; y=func(x,n); p=gcd(y-x,n);
9          while(p==1) {x=func(x,n); y=func(func(y,n),n); p=gcd((y-x+n)%n,n)%n;}
10         if(p==0||p==n) continue;
11         return p;
12     }
13 }
14 void factor(LL n){
15     LL x;

```

```

16 x=Pollard(n);
17 if(x==n){ ans[ansn++]=x; return; }
18 factor(x), factor(n/x);
19 }

```

二次剩余

```

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

```

Pell 方程

```

1 ULL A,B,p[maxn],q[maxn],a[maxn],g[maxn],h[maxn];
2 int main() {
3     for (int test=1, n;scanf("%d",&n) && n;++test) {
4         printf("Case %d: ",test);
5         if (fabs(sqrt(n)-floor(sqrt(n)+1e-7))<=1e-7) {
6             int a=(int)(floor(sqrt(n)+1e-7)); printf("%d %d\n",a,1);
7         } else {
8             // 求  $x^2 - ny^2 = 1$  的最小正整数根, n 不是完全平方数
9             p[1]=q[0]=h[1]=1;p[0]=q[1]=g[1]=0;
10            a[2]=(int)(floor(sqrt(n)+1e-7));
11            for (int i=2;i;++i) {
12                g[i]=-g[i-1]+a[i]*h[i-1]; h[i]=(n-sqr(g[i]))/h[i-1];

```

```

13         a[i+1]=(g[i]+a[2])/h[i]; p[i]=a[i]*p[i-1]+p[i-2];
14         q[i]=a[i]*q[i-1]+q[i-2];
15         if (sqr((ULL)(p[i]))-n*sqr((ULL)(q[i]))==1){
16             A=p[i];B=q[i];break; }
17     } cout << A << ' ' << B <<endl;
18 }}}}

```

日期公式

```

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

```

Schreier-Sims

```

1 namespace Schreier_Sims_Algorithm{
2     struct Permutation{
3         vector<int> P;
4         Permutation(){}
5         Permutation(int n){
6             P.resize(n);
7         }
8         Permutation inv()const{
9             Permutation ret(P.size());
10            for(int i = 0; i < int(P.size()); ++i) ret.P[P[i]] = i;
11            return ret;
12        }
13        int &operator [] (const int &dn){
14            return P[dn];
15        }
16        void resize(const size_t &sz){
17            P.resize(sz);
18        }
19        size_t size()const{
20            return P.size();
21        }
22        const int &operator [] (const int &dn)const{
23            return P[dn];
24        }
25    };

```

```

26 Permutation operator *(const Permutation &a, const Permutation &b){
27     Permutation ret(a.size());
28     for(int i = 0; i < (int)a.size(); ++i){
29         ret[i] = b[a[i]];
30     }
31     return ret;
32 }
33
34 typedef vector<Permutation> Bucket;
35 typedef vector<int> Table;
36 typedef pair<int,int> pii;
37 int n, m;
38 vector<Bucket> buckets, bucketsInv;
39 vector<Table> lookupTable;
40
41 int fastFilter(const Permutation &g, bool addToGroup = true){
42     int n = buckets.size();
43     Permutation p;
44     for(int i = 0; i < n; ++i){
45         int res = lookupTable[i][p[i]];
46         if(res == -1){
47             if(addToGroup){
48                 buckets[i].push_back(p);
49                 bucketsInv[i].push_back(p.inv());
50                 lookupTable[i][p[i]] = (int)buckets[i].size() - 1;
51             }
52             return i;
53         }
54         p = p * bucketsInv[i][res];
55         swap(i1,i2);
56     }
57     return -1;
58 }
59
60 long long calcTotalSize(){
61     long long ret = 1;
62     for(int i = 0; i < n; ++i){
63         ret *= buckets[i].size();
64     }
65     return ret;
66 }
67
68 bool inGroup(const Permutation &g){
69     return fastFilter(g, false) == -1;
70 }

```

```

71
72 void solve(const Bucket &gen,int _n){// m perm[0..n - 1]s
73     n = _n, m = gen.size();
74     //clear all
75     vector<Bucket> _buckets(n);
76     swap(buckets, _buckets);
77     vector<Bucket> _bucketsInv(n);
78     swap(bucketsInv, _bucketsInv);
79     vector<Table> _lookupTable(n);
80     swap(lookupTable, _lookupTable);
81 }
82
83 for(int i = 0; i < n; ++i){
84     lookupTable[i].resize(n);
85     fill(lookupTable[i].begin(), lookupTable[i].end(), -1);
86 }
87
88 Permutation id(n);
89 for(int i = 0; i < n; ++i){
90     id[i] = i;
91 }
92
93 for(int i = 0; i < n; ++i){
94     buckets[i].push_back(id);
95     bucketsInv[i].push_back(id);
96     lookupTable[i][i] = 0;
97 }
98
99 for(int i = 0; i < m; ++i){
100     fastFilter(gen[i]);
101 }
102
103 queue<pair<point,point> > toUpdate;
104 for(int i = 0; i < n; ++i){
105     for(int j = i; j < n; ++j){
106         for(int k = 0; k < (int)buckets[i].size(); ++k){
107             for(int l = 0; l < (int)buckets[j].size(); ++l){
108                 toUpdate.push(make_pair(pii(i,k), pii(j,l)));
109             }
110         }
111     }
112 }
113
114 while(!toUpdate.empty()){
115     pii a = toUpdate.front().first;
116     pii b = toUpdate.front().second;
117     toUpdate.pop();
118     int res = fastFilter(buckets[a.first][a.second] * buckets[b.first][b.second]);
119     if(res == -1) continue;
120     pii newPair(res, (int)buckets[res].size() - 1);
121     for(int i = 0; i < n; ++i){

```

```

116     for(int j = 0; j < (int)buckets[i].size(); ++j){
117         if(i <= res){
118             toUpdate.push(make_pair(pii(i , j), newPair));
119         }
120         if(res<=i){
121             toUpdate.push(make_pair(newPair, pii(i, j)));
122         }
123     }
124 }
125 }
126 }
127 }

```

Romberg

```

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

```

线性规划

```

1  // 求max{cx | Ax ≤ b, x ≥ 0}的解
2  typedef vector<double> VD;
3  VD simplex(vector<VD> A, VD b, VD c) {
4      int n = A.size(), m = A[0].size() + 1, r = n, s = m - 1;
5      vector<VD> D(n + 2, VD(m + 1, 0)); vector<int> ix(n + m);
6      for (int i = 0; i < n + m; ++ i) ix[i] = i;
7      for (int i = 0; i < n; ++ i) {
8          for (int j = 0; j < m - 1; ++ j) D[i][j] = -A[i][j];
9          D[i][m - 1] = 1; D[i][m] = b[i];
10         if (D[r][m] > D[i][m]) r = i;
11     }
12     for (int j = 0; j < m - 1; ++ j) D[n][j] = c[j];
13     D[n + 1][m - 1] = -1;
14     for (double d; ; ) {
15         if (r < n) {
16             int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t;

```

```

17     D[r][s] = 1.0 / D[r][s]; vector<int> speedUp;
18     for (int j = 0; j <= m; ++ j) if (j != s) {
19         D[r][j] *= -D[r][s];
20         if(D[r][j]) speedUp.push_back(j);
21     }
22     for (int i = 0; i <= n + 1; ++ i) if (i != r) {
23         for(int j = 0; j < speedUp.size(); ++ j)
24             D[i][speedUp[j]] += D[r][speedUp[j]] * D[i][s];
25         D[i][s] *= D[r][s];
26     } r = -1; s = -1;
27     for (int j = 0; j < m; ++ j) if (s < 0 || ix[s] > ix[j])
28         if (D[n + 1][j] > EPS || (D[n + 1][j] > -EPS && D[n][j] > EPS)) s = j;
29     if (s < 0) break;
30     for (int i = 0; i < n; ++ i) if (D[i][s] < -EPS)
31         if (r < 0 || (d = D[r][m] / D[r][s] - D[i][m] / D[i][s]) < -EPS
32             || (d < EPS && ix[r + m] > ix[i + m])) r = i;
33     if (r < 0) return VD(); // 无边界
34 }
35 if (D[n + 1][m] < -EPS) return VD(); // 无解
36 VD x(m - 1);
37 for (int i = m; i < n + m; ++ i) if (ix[i] < m - 1) x[ix[i]] = D[i - m][m];
38 return x; // 最优值在 D[n][m]
39 }

```

FFT

```

1  // double 精度对109 + 7 取模最多可以做到220
2
3  const int MOD = 1000003;
4
5  const double PI = acos(-1);
6
7  typedef complex<double> Complex;
8
9  const int N = 65536, L = 15, MASK = (1 << L) - 1;
10
11 Complex w[N];
12
13 void FFTInit() {
14     for (int i = 0; i < N; ++i) {
15         w[i] = Complex(cos(2 * i * PI / N), sin(2 * i * PI / N));
16     }
17 }
18
19 void FFT(Complex p[], int n) {

```

```

20 for (int i = 1, j = 0; i < n - 1; ++i) {
21     for (int s = n; j ^= s >>= 1, ~j & s;);
22     if (i < j) {
23         swap(p[i], p[j]);
24     }
25 }
26 for (int d = 0; (1 << d) < n; ++d) {
27     int m = 1 << d, m2 = m * 2, rm = n >> (d + 1);
28     for (int i = 0; i < n; i += m2) {
29         for (int j = 0; j < m; ++j) {
30             Complex &p1 = p[i + j + m], &p2 = p[i + j];
31             Complex t = w[rm * j] * p1;
32             p1 = p2 - t;
33             p2 = p2 + t;
34         }
35     }
36 }
37 }
38
39 Complex A[N], B[N], C[N], D[N];
40
41 void mul(int a[N], int b[N]) {
42     for (int i = 0; i < N; ++i) {
43         A[i] = Complex(a[i] >> L, a[i] & MASK);
44         B[i] = Complex(b[i] >> L, b[i] & MASK);
45     }
46     FFT(A, N);
47     FFT(B, N);
48     for (int i = 0; i < N; ++i) {
49         int j = (N - i) % N;
50         Complex da = (A[i] - conj(A[j])) * Complex(0, -0.5),
51             db = (A[i] + conj(A[j])) * Complex(0.5, 0),
52             dc = (B[i] - conj(B[j])) * Complex(0, -0.5),
53             dd = (B[i] + conj(B[j])) * Complex(0.5, 0);
54         C[j] = da * dd + da * dc * Complex(0, 1);
55         D[j] = db * dd + db * dc * Complex(0, 1);
56     }
57     FFT(C, N);
58     FFT(D, N);
59     for (int i = 0; i < N; ++i) {
60         long long da = (long long)(C[i].imag() / N + 0.5) % MOD,
61             db = (long long)(C[i].real() / N + 0.5) % MOD,
62             dc = (long long)(D[i].imag() / N + 0.5) % MOD,
63             dd = (long long)(D[i].real() / N + 0.5) % MOD;
64         a[i] = ((dd << (L * 2)) + ((db + dc) << L) + da) % MOD;

```

```

65     }
66 }

```

NTT

```

1 //R 是 2^n*q+1 形质数 p 的原根
2 void NFT(int P[], int n, int oper) {
3     for (int i = 1, j = 0; i < n - 1; ++i) {
4         for (int s = n; j ^= s >>= 1, ~j & s;);
5         if (i < j) {
6             swap(P[i], P[j]);
7         }
8     }
9     for (int d = 0; (1 << d) < n; ++d) {
10         int m = 1 << d, m2 = m * 2;
11         int unit_p0 = powmod(R, (MOD - 1) / m2);
12         if (oper < 0) {
13             unit_p0 = inverse(unit_p0);
14         }
15         for (int i = 0; i < n; i += m2) {
16             int unit = 1;
17             for (int j = 0; j < m; ++j) {
18                 int &P1 = P[i + j + m],
19                     &P2 = P[i + j];
20                 int t = (long long)unit * P1 % MOD;
21                 P1 = (P2 - t + MOD) % MOD;
22                 P2 = (P2 + t) % MOD;
23                 unit = (long long)unit * unit_p0 % MOD;
24             }
25         }
26     }
27 }

```

FWT

```

1 void FWT(int a[N], int N) {
2     for (int d = 1; d < N; d <= 1) {
3         int d2 = d << 1;
4         for (int i = 0; i < N; i += d2) {
5             int *x = a + i, *y = a + i + d;
6             for (int j = 0; j < d; ++j, ++x, ++y) {
7                 if ((*x += *y) >= MOD) {
8                     *x -= MOD;
9                 }
10                if ((*y = *x - (*y << 1)) < 0) {
11                    if ((*y += MOD) < 0) {
12                        *y += MOD;

```

```

13     }
14     }
15     }
16     }
17 }
18 }
19
20 void xorPow(int a[N], int n, int b[N]) {
21     memset(b, 0, sizeof(int) * N);
22     b[0] = 1;
23     FWT(a, N);
24     FWT(b, N);
25     while(n) {
26         if (n & 1) {
27             dot(b, a, N);
28         }
29         dot(a, a, N);
30         n >>= 1;
31     }
32     FWT(b, N);
33     norm(b, N);
34 }

```

```

20 k = 1;
21 for (i = 2; i < M; i++) {
22     int Len = k + Next[k], L = Next[i - k];
23     if (L < Len - i) {
24         Next[i] = L;
25     } else {
26         for (j = max(0, Len - i); i + j < M && a[j] == a[i + j]; j++);
27         Next[i] = j;
28         k = i;
29     }
30 }
31 for (j = 0; j < N && j < M && a[j] == b[j]; j++);
32 ret[0] = j;
33 k = 0;
34 for (i = 1; i < N; i++) {
35     int Len = k + ret[k], L = Next[i - k];
36     if (L < Len - i) {
37         ret[i] = L;
38     } else {
39         for (j = max(0, Len - i); j < M && i + j < N && a[j] == b[i + j]; j++);
40         ret[i] = j;
41         k = i;
42     }
43 }
44 }

```

Manacher/ 扩展 KMP

```

1 void Manacher(char text[], int n, int palindrome[]) {
2     palindrome[0] = 1;
3     for (int i = 1, j = 0, i < (n << 1) - 1; ++ i) {
4         int p = i >> 1;
5         int q = i - p;
6         int r = (j + 1 >> 1) + palindrome[j] - 1;
7         palindrome[i] = r < q ? 0 : min(r - q + 1, palindrome[(j << 1) - i]);
8         while (0 <= p - palindrome[i] && q + palindrome[i] < n && text[p -
9             ↪ palindrome[i]] == text[q + palindrome[i]]) {
10             palindrome[i] ++;
11         }
12         if (q + palindrome[i] - 1 > r) {
13             j = i;
14         }
15     }
16 }
17 void ExtendedKMP(char *a, char *b, int M, int N, int *Next, int *ret) { // a ->
18     ↪ 模式串 b -> 匹配串
19     int i, j, k;
20     for (j = 0; 1 + j < M && a[j] == a[1 + j]; j++); Next[1] = j;

```

后缀数组 (倍增)

```

1 int rank[MAX_N], height[MAX_N];
2 int cmp(int *x, int a, int b, int d) {
3     return x[a] == x[b] && x[a + d] == x[b + d];
4 }
5 void doubling(int *a, int N, int M) {
6     static int sRank[MAX_N], tmpA[MAX_N], tmpB[MAX_N];
7     int *x = tmpA, *y = tmpB;
8     for (int i = 0; i < M; ++ i) sRank[i] = 0;
9     for (int i = 0; i < N; ++ i) ++ sRank[x[i] = a[i]];
10    for (int i = 1; i < M; ++ i) sRank[i] += sRank[i - 1];
11    for (int i = N - 1; i >= 0; -- i) sa[-- sRank[x[i]]] = i;
12    for (int d = 1, p = 0; p < N; M = p, d <= 1) {
13        p = 0; for (int i = N - d; i < N; ++ i) y[p++] = i;
14        for (int i = 0; i < N; ++ i) if (sa[i] >= d) y[p++] = sa[i] - d;
15        for (int i = 0; i < M; ++ i) sRank[i] = 0;
16        for (int i = 0; i < N; ++ i) ++ sRank[x[i]];
17        for (int i = 1; i < M; ++ i) sRank[i] += sRank[i - 1];
18        for (int i = N - 1; i >= 0; -- i) sa[-- sRank[x[y[i]]]] = y[i];

```



```

19 swap(x,y); x[sa[0]]=0; p=1;
20 for(int i=1;i<N;++i) x[sa[i]]=cmp(y,sa[i],sa[i-1],d)?p-1:p++;
21 }
22 }
23 void calcHeight(){
24     for(int i=0;i<N;++i) rank[sa[i]]=i;
25     int cur=0; for(int i=0;i<N;++i)
26     if(rank[i]){
27         if(cur) cur--;
28         for(;a[i+cur]==a[sa[rank[i]-1]+cur];++cur);
29         height[rank[i]]=cur;
30     }
31 }

```

后缀数组 (DC3)

```

1 // 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
2 // 约定除 r[n-1] 外所有的 r[i] 都大于 0, r[n-1]=0
3 // 函数结束后, 结果放在 sa 数组中, 从 sa[0] 到 sa[n-1]
4 #define maxn 10000
5 #define F(x) ((x)/3+((x)%3==1?0:tb))
6 #define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
7 int wa[maxn],wb[maxn],wv[maxn],wss[maxn]; // 必须这么大
8 int s[maxn*3],sa[maxn*3];
9 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];}
10 int c12(int k,int *r,int a,int b){
11     if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
12     else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
13 }
14 void sort(int *r,int *a,int *b,int n,int m){
15     int i; for(i=0;i<n;i++) wv[i]=r[a[i]];
16     for(i=0;i<m;i++) wss[i]=0; for(i=0;i<n;i++) wss[wv[i]]++;
17     for(i=1;i<m;i++) wss[i]+=wss[i-1];
18     for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
19 }
20 void dc3(int *r,int *sa,int n,int m){
21     int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
22     r[n]=r[n+1]=0;
23     for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;
24     sort(r+2,wa,tbc,m); sort(r+1,wb,wa,tbc,m); sort(r,wa,wb,tbc,m);
25     for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
26         rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
27     if(p<tbc) dc3(rn,san,tbc,p);
28     else for(i=0;i<tbc;i++) san[rn[i]]=i;
29     for(i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]*3;
30     if(n%3==1) wb[ta++]=n-1;

```

```

31 sort(r,wb,wa,ta,m); for(i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;
32 for(i=0,j=0,p=0;i<ta && j<tbc;p++)
33     sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
34 for(;i<ta;p++) sa[p]=wa[i++]; for(;j<tbc;p++) sa[p]=wb[j++];}
35 int main(){
36     int n,m=0; scanf("%d",&n);
37     for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
38     printf("%d\n",m); s[n++]=0; dc3(s,sa,n,m);
39     for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
40 }

```

后缀自动机

```

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

```

后缀树 (With Walk)

```

1 int pos;
2
3 struct Node {

```

```
4  int l, r;
5
6  Node *suf, *ch[C];
7
8  Node (int l = -1, int r = INF) : l(l), r(r) {
9      suf = 0;
10     memset(ch, 0, sizeof(ch));
11 }
12
13 int len() {
14     return min(r, pos + 1) - 1;
15 }
16 };
17
18 int top;
19
20 Node pool[N << 1];
21
22 Node *root, *nxtSuf, *cur;
23
24 int text[N];
25
26 int remCnt, curP, curLen;
27
28 void init() {
29     top = 0, pos = -1, remCnt = 0, curP = 0, curLen = 0;
30     nxtSuf = NULL;
31     root = cur = new(pool + (top++)) Node(-1, -1);
32 }
33
34 void link(Node *u) {
35     if (nxtSuf) {
36         nxtSuf->suf = u;
37     }
38     nxtSuf = u;
39 }
40
41 bool walk(Node *u) {
42     int len = u->len();
43     if (curLen >= len) {
44         curP += len;
45         curLen -= len;
46         cur = u;
47         return true;
48     }
```

```
49     return false;
50 }
51
52 void extend(int c) {
53     text[++pos] = c;
54     nxtSuf = NULL;
55     ++remCnt;
56     while (remCnt) {
57         curP = curLen ? curP : pos;
58         int curE = text[curP];
59         if (!cur->ch[curE]) {
60             cur->ch[curE] = new(pool + (top++)) Node(pos);
61             link(cur);
62         } else {
63             Node *nxt = cur->ch[curE];
64             if (walk(nxt)) {
65                 continue;
66             }
67             if (text[nxt->l + curLen] == c) {
68                 ++curLen;
69                 link(cur);
70                 break;
71             }
72             Node *split = new(pool + (top++)) Node(nxt->l, nxt->l + curLen);
73             cur->ch[curE] = split;
74             split->ch[c] = new(pool + (top++)) Node(pos);
75             nxt->l += curLen;
76             split->ch[text[nxt->l]] = nxt;
77             link(split);
78         }
79         --remCnt;
80         if (cur == root && curLen > 0) {
81             --curLen;
82             curP = pos - remCnt + 1;
83         } else {
84             cur = cur->suf ? cur->suf : root;
85         }
86     }
87 }
88
89 void finish() {
90     nxtSuf = NULL;
91     for (int i = 0; i < top; ++i) {
92         if (pool[i].r == INF) {
93             link(pool + i);
```

```

94     }
95 }
96 while (remCnt > 0) {
97     if (curLen) {
98         int curE = text[curP];
99         Node *nxt = cur->ch[curE];
100         if (walk(nxt)) {
101             continue;
102         }
103         Node *split = new(pool + (top++)) Node(nxt->l, nxt->l + curLen);
104         cur->ch[curE] = split;
105         nxt->l += curLen;
106         split->ch[text[nxt->l]] = nxt;
107         link(split);
108     } else {
109         link(cur);
110     }
111     --remCnt;
112     if (cur == root && curLen > 0) {
113         --curLen;
114         curP = pos - remCnt + 1;
115     } else {
116         cur = cur->suf ? cur->suf : root;
117     }
118 }
119 if (nxtSuf != root) {
120     link(root);
121 }
122 }
123
124 int buf[N];
125
126 struct Walker {
127     Node *cur, *root;
128     int curP, curLen, pos, totLen;
129
130     Walker(Node* root) : root(root) {
131         cur = root;
132         curP = -1;
133         curLen = 0;
134         totLen = 0;
135         pos = -1;
136     }
137
138     void descend() {

```

```

139     while (curLen > 0) {
140         Node *nxt = cur->ch[buf[curP]];
141         int len = nxt->len();
142         if (curLen >= len) {
143             curP += len;
144             curLen -= len;
145             cur = nxt;
146         } else {
147             break;
148         }
149     }
150 }
151
152 void walk(int c) {
153     buf[++pos] = c;
154     while (curLen) {
155         Node *nxt = cur->ch[buf[curP]];
156         if (nxt->l + curLen <= ::pos && text[nxt->l + curLen] == c) {
157             ++totLen;
158             ++curLen;
159             break;
160         } else {
161             --totLen;
162             if (cur == root && curLen > 0) {
163                 --curLen;
164                 curP = pos - curLen;
165             } else {
166                 cur = cur->suf;
167             }
168             descend();
169         }
170     }
171     if (curLen == 0) {
172         curP = pos;
173         int curE = buf[curP];
174         while (cur && !cur->ch[curE]) {
175             --totLen;
176             cur = cur->suf;
177         }
178         if (cur) {
179             ++totLen;
180             curLen = 1;
181         } else {
182             cur = root;
183             totLen = 0;

```

```

184     }
185     }
186     descend();
187 }
188 };

```

后缀树 (With Pop Front)

```

1  int pos;
2
3  int text[N];
4
5  struct Node {
6      int l, r;
7
8      Node *suf, *ch[C];
9
10     int dgr;
11
12     Node *fa;
13
14     Node (int l = -1, int r = INF) : l(l), r(r) {
15         suf = fa = NULL;
16         memset(ch, 0, sizeof(ch));
17         dgr = 0;
18     }
19
20     Node* addEdge(Node *t) {
21         int c = text[t->l];
22         dgr += !ch[c];
23         ch[c] = t;
24         t->fa = this;
25         return t;
26     }
27
28     int len() {
29         return min(r, pos + 1) - l;
30     }
31 };
32
33 int top;
34
35 Node pool[N << 1];
36
37 Node *root, *nxtSuf, *cur;
38

```

```

39 int remCnt, curP, curLen;
40
41 long long size;
42
43 queue<Node*> leaves;
44
45 void init() {
46     top = 0, pos = -1;
47     remCnt = 0, curP = 0, curLen = 0;
48     nxtSuf = NULL;
49     root = cur = new(pool + (top++)) Node(-1, -1);
50     size = 0;
51     while (leaves.size()) {
52         leaves.pop();
53     }
54 }
55
56 void link(Node *u) {
57     if (nxtSuf) {
58         nxtSuf->suf = u;
59     }
60     nxtSuf = u;
61 }
62
63 bool walk(Node *u) {
64     int len = u->len();
65     if (curLen >= len) {
66         curP += len;
67         curLen -= len;
68         cur = u;
69         return true;
70     }
71     return false;
72 }
73
74 void extend(int c) {
75     text[++pos] = c;
76     nxtSuf = NULL;
77     ++remCnt;
78     while (remCnt) {
79         curP = curLen ? curP : pos;
80         int curE = text[curP];
81         if (!cur->ch[curE]) {
82             leaves.push(cur->addEdge(new(pool + (top++)) Node(pos)));
83             link(cur);

```

```

84     } else {
85         Node *nxt = cur->ch[curE];
86         if (walk(nxt)) {
87             continue;
88         }
89         if (text[nxt->l + curLen] == c) {
90             ++curLen;
91             link(cur);
92             break;
93         }
94         Node *split = new(pool + (top++)) Node(nxt->l, nxt->l + curLen);
95         cur->addEdge(split);
96         leaves.push(split->addEdge(new(pool + (top++)) Node(pos)));
97         nxt->l += curLen;
98         split->addEdge(nxt);
99         link(split);
100     }
101     --remCnt;
102     if (cur == root && curLen > 0) {
103         curP = pos - (--curLen);
104     } else {
105         cur = cur->suf ? cur->suf : root;
106     }
107 }
108 size += leaves.size();
109 }
110
111 void finish() {
112     nxtSuf = NULL;
113     for (int i = 0; i < top; ++i) {
114         if (pool[i].r == INF) {
115             link(pool + i);
116         }
117     }
118     while (remCnt > 0) {
119         if (curLen) {
120             int curE = text[curP];
121             Node *nxt = cur->ch[curE];
122             if (walk(nxt)) {
123                 continue;
124             }
125             Node *split = new(pool + (top++)) Node(nxt->l, nxt->l + curLen);
126             leaves.push(cur->addEdge(split));
127             nxt->l += curLen;
128             split->addEdge(nxt);

```

```

129         link(split);
130     } else {
131         leaves.push(cur);
132         link(cur);
133     }
134     --remCnt;
135     if (cur == root && curLen > 0) {
136         --curLen;
137         curP = pos - remCnt + 1;
138     } else {
139         cur = cur->suf ? cur->suf : root;
140     }
141 }
142 if (nxtSuf != root) {
143     link(root);
144 }
145 }
146
147 void eraseUp(Node *u) {
148     size -= u->len();
149     int ch = text[u->l];
150     u = u->fa;
151     u->ch[ch] = NULL;
152     --(u->dgr);
153 }
154
155 void erase() {
156     Node *u = leaves.front();
157     leaves.pop();
158     while (u->dgr == 0 && u != cur) {
159         eraseUp(u);
160     }
161     if (u == cur) {
162         if (cur->dgr == 0 && curLen == 0) {
163             int len = u->len();
164             curLen = len;
165             curP = pos - len + 1;
166             cur = cur->fa;
167             eraseUp(u);
168         }
169         if (curLen) {
170             int curE = text[curP];
171             if (!cur->ch[curE]) {
172                 Node *leaf = new(pool + (top++)) Node(pos - curLen + 1);
173                 leaves.push(cur->addEdge(leaf));

```

```

174     size += leaf->len();
175     --remCnt;
176     if (cur == root && curLen > 0) {
177         curP = pos - (--curLen) + 1;
178     } else {
179         cur = cur->suf ? cur->suf : root;
180     }
181     while (curLen && walk(cur->ch[text[curP]])) {
182         continue;
183     }
184 }
185 }
186 }
187 }
188
189 int n;
190
191 char s[N], buf[N];
192
193 int ord[N], stop, sord[N << 1];
194
195 void dfs(Node *u) {
196     sord[u - pool] = stop++;
197     for (int i = 0; i < C; ++i) {
198         if (u->ch[i]) {
199             dfs(u->ch[i]);
200         }
201     }
202 }
203
204 void getOrd() {
205     init();
206     for (int i = 0; i < n; ++i) {
207         extend(s[i] - 'a');
208     }
209     finish();
210     stop = 0;
211     dfs(root);
212     int i = 0;
213     while (leaves.size()) {
214         ord[i++] = sord[leaves.front() - pool];
215         leaves.pop();
216     }
217 }

```

字符串最小表示

```

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

```

轻重链剖分

```

1 struct Tree(){}*root[N];
2 int father[N],size[N],depth[N];
3 int bfsOrd[N],pathId[N],ordInPath[N],sqn[N];
4 void doBfs(int s){
5     int qh=0,qt=0,*que=bfsOrd; father[s]=-1; depth[s]=0;
6     for(que[qt++]=s;qh<qt;){
7         int u=que[qh++];
8         foreach(iter,adj[u]){
9             int v=*iter; if(v==father[u]) continue;
10            father[v]=u; depth[v]=depth[u]+1; que[qt++]=v;
11        }
12    }
13 }
14 void doSplit(){
15     for(int i=N-1;i>=0;--i){
16         int u=bfsOrd[i]; size[u]=1;
17         foreach(iter,adj[u]){
18             int v=*iter; if(v==father[u]) continue; size[u]+=size[v];
19         }
20     }
21     memset(pathId,-1,sizeof pathId);
22     for(int i=0;i<N;++i){
23         int top=bfsOrd[i],cnt=0;
24         if(pathId[top]!=-1) continue;
25         for(int next,u=top;u!=-1;u=next){
26             sqn[cnt]=val[u]; ordInPath[u]=cnt; pathId[u]=top; ++cnt;
27             next=-1;
28             foreach(iter,adj[u]){
29                 int v=*iter; if(v==father[u]) continue;
30                 if(next<0||size[next]<size[v]) next=v;
31             }
32         }

```

```

33     root[top]=new Tree(0,cnt,sqn);
34 }
35 }
36 void prepare(){ doBfs(0); doSplit(); }

```

KD Tree

```

1  #include <cstdio>
2  #include <vector>
3  #include <iostream>
4  #include <algorithm>
5
6  using namespace std;
7  // 带插入版本，没有写内存回收，空间复杂度  $n \log n$ ，如果不需要插入可以大大简化
8  // N 为最大点数，D 为每个点的最大维度，d 为实际维度
9  // 以查找最近点为例 ret 为当前最近点的距离的平方，用来剪枝，查询 k 近或 k 远的方法类似
10 // 使用时注意先 initNull
11 const long long INF = (int)1e9 + 10;
12 const int N = 200000 + 10;
13 const int D = 5;
14 const double SCALE = 0.75;
15 struct Point { int x[D]; } buf[N];
16 int d;
17 struct Node {
18     int depth, size;
19     Node *ch[2], *p;
20     Point val, maxv, minv;
21     void set(Node *t, int d) { ch[d] = t; t->p = this; }
22     bool dir() { return this == p->ch[1]; }
23     bool balanced() {
24         return (double)max(ch[0]->size, ch[1]->size) <= (double)size * SCALE;
25     }
26     void update() {
27         size = ch[0]->size + ch[1]->size + 1;
28         for(int i = 0; i < d; ++i) {
29             maxv.x[i] = max(val.x[i], max(ch[0]->maxv.x[i], ch[1]->maxv.x[i]));
30             minv.x[i] = min(val.x[i], min(ch[0]->minv.x[i], ch[1]->minv.x[i]));
31         }
32     }
33 } nodePool[N], *totNode, *null;
34 Node* newNode(Point p, int depth) {
35     Node *t = totNode++;
36     t->ch[0] = t->ch[1] = t->p = null;
37     t->depth = depth;
38     t->val = t->maxv = t->minv = p;
39     t->size = 1;

```

```

40     return t;
41 }
42 long long ret;
43 int ctr;
44 int cmp(const Point &a, const Point &b) { return a.x[ctr] < b.x[ctr]; }
45 struct KDTree {
46     Node *root;
47     KDTree() { root = null; }
48     KDTree(Point *a, int n) {
49         root = build(a, 0, n - 1, 0);
50     }
51     Node *build(Point *a, int l, int r, int depth) {
52         if (l > r) return null;
53         ctr = depth;
54         sort(a + l, a + r + 1, cmp);
55         int mid = (l + r) >> 1;
56         Node *t = newNode(a[mid], depth);
57         t->set(build(a, l, mid - 1, (depth + 1) % d), 0);
58         t->set(build(a, mid + 1, r, (depth + 1) % d), 1);
59         t->update();
60         return t;
61     }
62     void tranverse(Node *t, Point *vec, int &tot) {
63         if (t == null) return;
64         vec[tot++] = t->val;
65         tranverse(t->ch[0], vec, tot);
66         tranverse(t->ch[1], vec, tot);
67     }
68     void rebuild(Node *t) {
69         Node *p = t->p;
70         int tot = 0;
71         tranverse(t, buf, tot);
72         Node *u = build(buf, 0, tot - 1, t->depth);
73         p->set(u, t->dir());
74         for( ; p != null; p = p->p) p->update();
75         if (t == root) root = u;
76     }
77     void insert(Point p) {
78         if (root == null) { root = newNode(p, 0); return; }
79         Node *cur = root, *last = null;
80         int dir = 0;
81         for( ; cur != null; ) {
82             last = cur;
83             dir = (p.x[cur->depth] > cur->val.x[cur->depth]);
84             cur = cur->ch[dir];

```

```

85     }
86     Node *t = newNode(p, (last->depth + 1) % d), *bad = null;
87     last->set(t, dir);
88     for( ; t != null; t = t->p) {
89         t->update();
90         if (!t->balanced()) bad = t;
91     }
92     if (bad != null) rebuild(bad);
93 }
94 long long calcEval(Point u, Node *t, int d) {
95     long long l = t->minv.x[d], r = t->maxv.x[d], x = u.x[d];
96     if (x >= l && x <= r) return 0LL;
97     long long ret = min(abs(x - l), abs(x - r));
98     return ret * ret;
99 }
100 void updateAns(Point u, Point p) {
101     // 在这里更新答案
102 }
103 void query(Node *t, Point p) {
104     if (t == null) return;
105     updateAns(t->val, p);
106     long long evalLeft = calcEval(p, t->ch[0], t->depth);
107     long long evalRight = calcEval(p, t->ch[1], t->depth);
108     if (evalLeft <= evalRight) {
109         query(t->ch[0], p);
110         if (ret > evalRight) query(t->ch[1], p);
111     } else {
112         query(t->ch[1], p);
113         if (ret > evalLeft) query(t->ch[0], p);
114     }
115 }
116 void query(Point p) {
117     query(root, p);
118 }
119 };
120 void initNull() {
121     totNode = nodePool;
122     null = totNode++;
123     null->size = 0;
124     for(int i = 0; i < d; ++i) {
125         null->maxv.x[i] = -INF;
126         null->minv.x[i] = INF;
127     }
128 }

```

Splay Tree

```

1 // 注意初始化内存池和 null 节点
2 struct Node{
3     int rev,size; Node *ch[2],*p;
4     void set(Node*,int); int dir(); void update(); void relax(); void appRev();
5 } nodePool[MAX_NODE],*curNode,*null;
6 Node *newNode(){
7     Node *t=curNode++; t->rev=0, t->size=1;
8     t->ch[0]=t->ch[1]=t->p=null; return t;
9 }
10 struct Splay{
11     Node *root;
12     Splay(){ root=newNode(); root->set(newNode(),0); root->update(); }
13     void rot(Node *t){
14         Node *p=t->p; int d=t->dir();
15         p->relax(); t->relax();
16         if(p==root) root=t;
17         p->set(t->ch[!d],d); p->p->set(t,p->dir()); t->set(p,!d);
18         p->update();
19     }
20     void splay(Node *t,Node *f=null){
21         for(t->relax();t->p!=f;)
22             if(t->p->p==f) rot(t);
23             else t->dir()==t->p->dir()?(rot(t->p),rot(t)):(rot(t),rot(t));
24         t->update();
25     }
26 };
27 void initNull(){ curNode=nodePool;null=curNode++;null->size=0; }
28 void Node::set(Node *t,int _d){ ch[_d]=t; t->p=this; }
29 int Node::dir(){ return this==p->ch[1]; }
30 void Node::update(){ size=ch[0]->size+ch[1]->size+1;}
31 void Node::relax(){ if(rev) ch[0]->appRev(), ch[1]->appRev(), rev=false; }
32 void Node::appRev(){ if(this==null) return; rev^=true; swap(ch[0],ch[1]); }

```

Link Cut Tree

```

1 // 注意初始化 null 节点, 单点的 is_root 初始为 true
2 struct Node{
3     Node *ch[2], *p;
4     int is_root, rev;
5     bool dir();
6     void set(Node*, bool);
7     void update();
8     void relax();
9     void app_rev();
10 } *null;

```



```

11 void rot(Node *t){
12     Node *p=t->p; bool d=t->dir();
13     p->relax(); t->relax(); p->set(t->ch[!d],d);
14     if(p->is_root) t->p=p->p, swap(p->is_root,t->is_root);
15     else p->p->set(t,p->dir());
16     t->set(p,!d); p->update();
17 }
18 void splay(Node *t){
19     for(t->relax();!t->is_root;)
20         if(t->p->is_root) rot(t);
21         else t->dir()==t->p->dir() ?(rot(t->p),rot(t)) :(rot(t),rot(t));
22     t->update();
23 }
24 void access(Node *t){
25     for(Node *s=null; t!=null; s=t,t=t->p){
26         splay(t);
27         if (t->p == null) { /*TODO*/ }
28         t->ch[1]->is_root=true; s->is_root=false;
29         t->ch[1]=s; t->update();
30     }
31 }
32 bool Node::dir(){ return this==p->ch[1]; }
33 void Node::set(Node *t,bool _d){ ch[_d]=t; t->p=this; }
34 void Node::update(){ }
35 void Node::app_rev(){ if (this == null) return; rev ^= true; swap(ch[0], ch[1]); }
36 void Node::relax() { if(this==null) return; if (rev) { ch[0]->app_rev();
    ↪ ch[1]->app_rev(); rev = false; } }
37 void make_root(Node *u) { access(u); splay(u); u->app_rev(); }

```

Dominator Tree

```

1 vector<int> prec[N], succ[N];
2
3 vector<int> ord;
4
5 int stamp, vis[N];
6
7 int num[N];
8
9 int fa[N];
10
11 void dfs(int u) {
12     vis[u] = stamp;
13     num[u] = ord.size();
14     ord.push_back(u);
15     for (int i = 0; i < (int)succ[u].size(); ++i) {

```

```

16         int v = succ[u][i];
17         if (vis[v] != stamp) {
18             fa[v] = u;
19             dfs(v);
20         }
21     }
22 }
23
24 int fs[N], mins[N];
25
26 int dom[N], sem[N];
27
28 int find(int u) {
29     if (u != fs[u]) {
30         int v = fs[u];
31         fs[u] = find(fs[u]);
32         if (mins[v] != -1 && num[sem[mins[v]]] < num[sem[mins[u]]]) {
33             mins[u] = mins[v];
34         }
35     }
36     return fs[u];
37 }
38
39 void merge(int u, int v) {
40     fs[u] = v;
41 }
42
43 vector<int> buf[N];
44
45 int buf2[N];
46
47 void mark(int source) {
48     ord.clear();
49     ++stamp;
50     dfs(source);
51     for (int i = 0; i < (int)ord.size(); ++i) {
52         int u = ord[i];
53         fs[u] = u;
54         mins[u] = -1;
55         buf2[u] = -1;
56     }
57     for (int i = (int)ord.size() - 1; i > 0; --i) {
58         int u = ord[i], p = fa[u];
59         sem[u] = p;
60         for (int j = 0; j < (int)prec[u].size(); ++j) {

```

```

61     int v = prec[u][j];
62     if (use[v] != stamp) {
63         continue;
64     }
65     if (num[v] > num[u]) {
66         find(v);
67         v = sem[mins[v]];
68     }
69     if (num[v] < num[sem[u]]) {
70         sem[u] = v;
71     }
72 }
73 buf[sem[u]].push_back(u);
74 mins[u] = u;
75 merge(u, p);
76 while (buf[p].size()) {
77     int v = buf[p].back();
78     buf[p].pop_back();
79     find(v);
80     if (sem[v] == sem[mins[v]]) {
81         dom[v] = sem[v];
82     } else {
83         buf2[v] = mins[v];
84     }
85 }
86 }
87 dom[ord[0]] = ord[0];
88 for (int i = 0; i < (int)ord.size(); ++i) {
89     int u = ord[i];
90     if (~buf2[u]) {
91         dom[u] = dom[buf2[u]];
92     }
93 }
94 }

```

DancingLinks

```

1 struct node{
2     node *left,*right,*up,*down,*col; int row,cnt;
3 }*head,*col[MAXC],Node[MAXNODE],*ans[MAXNODE];
4 int totNode;
5 void insert(const std::vector<int> &V,int rownum){
6     std::vector<node*> N;
7     for(int i=0;i<(int)V.size();++i){
8         node* now=Node+(totNode++); now->row=rownum;
9         now->col=now->up=col[V[i]], now->down=col[V[i]]->down;

```

```

10     now->up->down=now, now->down->up=now;
11     now->col->cnt++; N.push_back(now);
12 }
13 for(int i=0;i<(int)V.size();++i)
14     N[i]->right=N[(i+1)%V.size()], N[i]->left=N[(i-1+V.size())%V.size()];
15 }
16 void Remove(node *x){
17     x->left->right=x->right, x->right->left=x->left;
18     for(node *i=x->down;i!=x;i=i->down)
19         for(node *j=i->right;j!=i;j=j->right)
20             j->up->down=j->down, j->down->up=j->up, --(j->col->cnt);
21 }
22 void Resume(node *x){
23     for(node *i=x->up;i!=x;i=i->up)
24         for(node *j=i->left;j!=i;j=j->left)
25             j->up->down=j->down->up=j, ++(j->col->cnt);
26     x->left->right=x, x->right->left=x;
27 }
28 bool search(int tot){
29     if(head->right==head) return true;
30     node *choose=NULL;
31     for(node *i=head->right;i!=head;i=i->right){
32         if(choose==NULL||choose->cnt>i->cnt) choose=i;
33         if(choose->cnt<2) break;
34     }
35     Remove(choose);
36     for(node *i=choose->down;i!=choose;i=i->down){
37         for(node *j=i->right;j!=i;j=j->right) Remove(j->col);
38         ans[tot]=i;
39         if(search(tot+1)) return true;
40         ans[tot]=NULL;
41         for(node *j=i->left;j!=i;j=j->left) Resume(j->col);
42     }
43     Resume(choose);
44     return false;
45 }
46 void prepare(int totC){
47     head=Node+totC;
48     for(int i=0;i<totC;++i) col[i]=Node+i;
49     totNode=totC+1;
50     for(int i=0;i<=totC;++i){
51         (Node+i)->right=Node+(i+1)%totC;
52         (Node+i)->left=Node+(i+totC)%totC;
53         (Node+i)->up=(Node+i)->down=Node+i;
54     }

```

55 }

弦图相关

1. 团数 \leq 色数, 弦图团数 = 色数
2. 设 $next(v)$ 表示 $N(v)$ 中最前的点. 令 w^* 表示所有满足 $A \in B$ 的 w 中最后的一个点, 判断 $v \cup N(v)$ 是否为极大团, 只需判断是否存在一个 w , 满足 $Next(w) = v$ 且 $|N(v)| + 1 \leq |N(w)|$ 即可.
3. 最小染色: 完美消除序列从后往前依次给每个点染色, 给每个点染上可以染的最小的颜色
4. 最大独立集: 完美消除序列从前往后能选就选
5. 弦图最大独立集数 = 最小团覆盖数, 最小团覆盖: 设最大独立集为 $\{p_1, p_2, \dots, p_t\}$, 则 $\{p_1 \cup N(p_1), \dots, p_t \cup N(p_t)\}$ 为最小团覆盖

图同构 Hash

$$F_t(i) = (F_{t-1}(i) \times A + \sum_{i \rightarrow j} F_{t-1}(j) \times B + \sum_{j \rightarrow i} F_{t-1}(j) \times C + D \times (i = a)) \bmod P$$

枚举点 a 迭代 K 次后求得的就是 a 点所对应的 hash 值
其中 K, A, B, C, D, P 为 hash 参数, 可自选

环状最长公共子序列

```

1 int n, a[N << 1], b[N << 1];
2
3 bool has(int i, int j) {
4     return a[(i - 1) % n] == b[(j - 1) % n];
5 }
6
7 const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1, 0}};
8
9 int from[N][N];
10
11 int solve() {
12     memset(from, 0, sizeof(from));
13     int ret = 0;
14     for (int i = 1; i <= 2 * n; ++i) {
15         from[i][0] = 2;
16         int left = 0, up = 0;
17         for (int j = 1; j <= n; ++j) {
18             int upleft = up + 1 + !has(i - 1, j);
19             if (!has(i, j)) {
20                 upleft = INT_MIN;
21             }
22             int max = std::max(left, std::max(upleft, up));
23             if (left == max) {
24                 from[i][j] = 0;

```

```

25     } else if (upleft == max) {
26         from[i][j] = 1;
27     } else {
28         from[i][j] = 2;
29     }
30     left = max;
31 }
32 if (i >= n) {
33     int count = 0;
34     for (int x = i, y = n; y;) {
35         int t = from[x][y];
36         count += t == 1;
37         x += DELTA[t][0];
38         y += DELTA[t][1];
39     }
40     ret = std::max(ret, count);
41     int x = i - n + 1;
42     from[x][0] = 0;
43     int y = 0;
44     while (y <= n && from[x][y] == 0) {
45         y++;
46     }
47     for (; x <= i; ++x) {
48         from[x][y] = 0;
49         if (x == i) {
50             break;
51         }
52         for (; y <= n; ++y) {
53             if (from[x + 1][y] == 2) {
54                 break;
55             }
56             if (y + 1 <= n && from[x + 1][y + 1] == 1) {
57                 y++;
58                 break;
59             }
60         }
61     }
62 }
63 }
64 return ret;
65 }

```

直线下有多少个格点

```

1 LL solve(LL n, LL a, LL b, LL m) {
2     // 计算 for (int i=0; i<n; ++i) s+=floor((a+b*i)/m)

```

```

3 //n,m,a,b>0
4 if(b==0) return n*(a/m);
5 if(a>=m) return n*(a/m)+solve(n,a%m,b,m);
6 if(b>=m) return (n-1)*n/2*(b/m)+solve(n,a,b%m,m);
7 return solve((a+b*n)/m,(a+b*n)%m,m,b);
8 }

```

费用流

```

1 // Q is a priority_queue<PII, vector<PII>, greater<PII> >
2 // for an edge(s, t): u is the capacity, v is the cost, nxt is the next edge,
3 // op is the opposite edge
4 // this code can not deal with negative cycles
5 typedef pair<int,int> PII;
6 struct edge{ int t,u,v; edge *nxt,*op; }E[MAXE],*V[MAXV];
7 int D[MAXN], dist[MAXN], maxflow, mincost; bool in[MAXN];
8 bool modlabel(){
9     while(!Q.empty()) Q.pop();
10    for(int i=S;i<=T;++i){
11        if(in[i]) D[i]=0,Q.push(PII(0,i));
12        else D[i]=inf;
13    }
14    while(!Q.empty()){
15        int x=Q.top().first,y=Q.top().second;
16        Q.pop();
17        if(y==T) break;
18        if(D[y]<x) continue;
19        for(edge *ii=V[y];ii;ii=ii->nxt) if(ii->u){
20            if(x+(ii->v+dist[ii->t]-dist[y])<D[ii->t]){
21                D[ii->t]=x+(ii->v+dist[ii->t]-dist[y]);
22                Q.push(PII(D[ii->t],ii->t));
23            }
24        }
25    }
26    if(D[T]==inf) return false;
27    for(int i=S;i<=T;++i) if(D[i]>D[T]) dist[i]+=D[T]-D[i];
28    return true;
29 }
30 int aug(int p,int limit){
31     if(p==T) return maxflow+=limit,mincost+=limit*dist[S],limit;
32     in[p]=1; int kk,ll=limit;
33     for(edge *ii=V[p];ii;ii=ii->nxt) if(ii->u){
34         if(!in[ii->t]&&dist[ii->t]+ii->v==dist[p]){
35             kk=aug(ii->t,min(ii->u,ll));
36             ll-=kk,ii->u-=kk,ii->op->u+=kk;
37             if(!ll) return in[p]=0,limit;

```

```

38     }
39 }
40 return limit-ll;
41 }
42 PII mincostFlow(){
43     for(int i=S;i<=T;++i) dist[i]=i==T?inf:0;
44     while(!Q.empty()) Q.pop();
45     Q.push(PII(0,T));
46     while(!Q.empty()){
47         int x=Q.top().first,y=Q.top().second;
48         Q.pop();
49         if(dist[y]<x) continue;
50         for(edge *ii=V[y];ii;ii=ii->nxt) if(ii->op->u&&ii->v+x<dist[ii->t){
51             dist[ii->t]=ii->v+x;
52             Q.push(PII(dist[ii->t],ii->t));
53         }
54     }
55     maxflow=mincost=0;
56     do{
57         do{
58             memset(in,0,sizeof(in));
59         }while(aug(S,maxflow));
60     }while(modlabel());
61     return PII(maxflow,mincost);
62 }

```

综合

定理 1: 最小覆盖数 = 最大匹配数

定理 2: 最大独立集 S 与 最小覆盖集 T 互补

算法:

1. 做最大匹配，没有匹配的空闲点 $\in S$
2. 如果 $u \in S$ 那么 u 的临点必然属于 T
3. 如果一对匹配的点中有一个属于 T 那么另外一个属于 S
4. 还不能确定的，把左子图的放入 S ，右子图放入 T

算法结束

上下界无源汇可行流：不用添 $T \rightarrow S$ ，判断是否流量平衡

上下界有源汇可行流：添 $T \rightarrow S$ (下界 0, 上界 ∞)，判断是否流量平衡

上下界最小流：不添 $T \rightarrow S$ 先流一遍，再添 $T \rightarrow S$ (下界 0, 上界 ∞) 在残图上流一遍，答案为 $S \rightarrow T$ 的流量值

上下界最大流：添 $T \rightarrow S$ (下界 0, 上界 ∞) 流一遍，再在残图上流一遍 S 到 T 的最大流，答案为前者的 $S \rightarrow T$ 的值 + 残图中 $S \rightarrow T$ 的最大流

Stirling 公式 $n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$

Stirling 数

第一类 :n 个元素的项目分作 k 个环排列的方法数目

积分表

Integrals of Rational Functions	$\int \frac{1}{1+x^2} dx = \tan^{-1} x$	$\int \frac{1}{a^2+x^2} dx = \frac{1}{a} \tan^{-1} \frac{x}{a}$	$\int \frac{x}{a^2+x^2} dx = \frac{1}{2} \ln a^2 + x^2 $	$\int \frac{x^2}{a^2+x^2} dx = x - a \tan^{-1} \frac{x}{a}$	$\int \frac{x^3}{a^2+x^2} dx = \frac{1}{2} x^2 - \frac{1}{2} a^2 \ln a^2 + x^2 $
$\int \frac{1}{ax^2+bx+c} dx = \frac{2}{\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}}$	$\int \frac{1}{(x+a)(x+b)} dx = \frac{1}{b-a} \ln \frac{a+x}{b+x}, a \neq b$	$\int \frac{x}{(x+a)^2} dx = \frac{a}{a+x} + \ln a+x $	$\int \frac{x}{ax^2+bx+c} dx = \frac{1}{2a} \ln ax^2+bx+c - \frac{b}{a\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}}$		
Integrals with Roots	$\int \frac{x}{\sqrt{x \pm a}} dx = \frac{2}{3} (x \mp 2a) \sqrt{x \pm a}$	$\int \sqrt{\frac{x}{a-x}} dx = -\sqrt{x(a-x)} - a \tan^{-1} \frac{\sqrt{x(a-x)}}{x-a}$	$\int \sqrt{\frac{x}{a+x}} dx = \sqrt{x(a+x)} - a \ln [\sqrt{x} + \sqrt{x+a}]$	$\int x \sqrt{x^2 \pm a^2} dx = \frac{1}{3} (x^2 \pm a^2)^{3/2}$	
$\int x \sqrt{ax+b} dx = \frac{2}{15a^2} (-2b^2 + abx + 3a^2 x^2) \sqrt{ax+b}$	$\int \sqrt{x(ax+b)} dx = \frac{1}{4a^{3/2}} \left[(2ax+b) \sqrt{ax(ax+b)} - b^2 \ln a\sqrt{x} + \sqrt{a(ax+b)} \right]$	$\int \sqrt{x^2 \pm a^2} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \pm \frac{1}{2} a^2 \ln x + \sqrt{x^2 \pm a^2} $			
$\int \sqrt{x^3(ax+b)} dx = \left[\frac{b}{12a} - \frac{b^2}{8a^2 x} + \frac{x}{3} \right] \sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}} \ln a\sqrt{x} + \sqrt{a(ax+b)} $	$\int \sqrt{a^2 - x^2} dx = \frac{1}{2} x \sqrt{a^2 - x^2} + \frac{1}{2} a^2 \tan^{-1} \frac{x}{\sqrt{a^2 - x^2}}$	$\int \frac{x^2}{\sqrt{x^2 \pm a^2}} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \mp \frac{1}{2} a^2 \ln x + \sqrt{x^2 \pm a^2} $			
$\int \frac{1}{\sqrt{x^2 \pm a^2}} dx = \ln x + \sqrt{x^2 \pm a^2} $	$\int \frac{1}{\sqrt{a^2 - x^2}} dx = \sin^{-1} \frac{x}{a}$	$\int \frac{x}{\sqrt{x^2 \pm a^2}} dx = \sqrt{x^2 \pm a^2}$	$\int \frac{x}{\sqrt{a^2 - x^2}} dx = -\sqrt{a^2 - x^2}$	$\int \sqrt{ax^2+bx+c} dx = \frac{b+2ax}{4a} \sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}} \ln 2ax+b+2\sqrt{a(ax^2+bx+c)} $	$\int \frac{1}{\sqrt{ax^2+bx+c}} dx = \frac{1}{\sqrt{a}} \ln 2ax+b+2\sqrt{a(ax^2+bx+c)} $
$\int x \sqrt{ax^2+bx+c} dx = \frac{1}{48a^{5/2}} \left(2\sqrt{a} \sqrt{ax^2+bx+c} \times (-3b^2+2abx+8a(c+ax^2)) + 3(b^3-4abc) \ln b+2ax+2\sqrt{a} \sqrt{ax^2+bx+c} \right)$				Integrals with Logarithms	$\int \ln(ax+b) dx = \left(x + \frac{b}{a}\right) \ln(ax+b) - x, a \neq 0$
$\int \frac{x}{\sqrt{ax^2+bx+c}} dx = \frac{1}{a} \sqrt{ax^2+bx+c} - \frac{b}{2a^{3/2}} \ln 2ax+b+2\sqrt{a(ax^2+bx+c)} $	$\int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2 \sqrt{a^2+x^2}}$				
$\int \frac{\ln ax}{x} dx = \frac{1}{2} (\ln ax)^2$	$\int \ln(x^2+a^2) dx = x \ln(x^2+a^2) + 2a \tan^{-1} \frac{x}{a} - 2x$	$\int \ln(x^2-a^2) dx = x \ln(x^2-a^2) + a \ln \frac{x+a}{x-a} - 2x$	$\int x \ln(ax+b) dx = \frac{bx}{2a} - \frac{1}{4} x^2 + \frac{1}{2} \left(x^2 - \frac{b^2}{a^2}\right) \ln(ax+b)$		
$\int \ln(ax^2+bx+c) dx = \frac{1}{a} \sqrt{4ac-b^2} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} - 2x + \left(\frac{b}{2a} + x\right) \ln(ax^2+bx+c)$	$\int x \ln(a^2-b^2x^2) dx = -\frac{1}{2} x^2 + \frac{1}{2} \left(x^2 - \frac{a^2}{b^2}\right) \ln(a^2-b^2x^2)$			Integrals with Exponentials	
$\int x^n e^{ax} dx = \frac{x^n e^{ax}}{a} - \frac{n}{a} \int x^{n-1} e^{ax} dx$	$\int x e^{-ax^2} dx = -\frac{1}{2a} e^{-ax^2}$	Integrals with Trigonometric Functions	$\int \sin^3 ax dx = -\frac{3 \cos ax}{4a} + \frac{\cos 3ax}{12a}$	$\int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a}$	
$\int \cos^3 ax dx = \frac{3 \sin ax}{4a} + \frac{\sin 3ax}{12a}$	$\int \cos ax \sin bxdx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b$	$\int \sin^2 ax \cos bxdx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)}$	$\int \sin^2 x \cos x dx = \frac{1}{3} \sin^3 x$		
$\int \cos^2 ax \sin bxdx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)}$	$\int \cos^2 ax \sin ax dx = -\frac{1}{3a} \cos^3 ax$	$\int \sin^2 ax \cos^2 bxdx = \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)}$	$\int \sin^2 ax \cos^2 ax dx = \frac{x}{8} - \frac{\sin 4ax}{32a}$		
$\int \tan ax dx = -\frac{1}{a} \ln \cos ax$	$\int \tan^2 ax dx = -x + \frac{1}{a} \tan ax$	$\int \tan^3 ax dx = \frac{1}{a} \ln \cos ax + \frac{1}{2a} \sec^2 ax$	$\int \sec x dx = \ln \sec x + \tan x = 2 \tanh^{-1} \left(\tan \frac{x}{2}\right)$	$\int \sec^2 ax dx = \frac{1}{a} \tan ax$	
$\int \sec^3 x dx = \frac{1}{2} \sec x \tan x + \frac{1}{2} \ln \sec x + \tan x $	$\int \sec x \tan x dx = \sec x$	$\int \sec^2 x \tan x dx = \frac{1}{2} \sec^2 x$	$\int \sec^n x \tan x dx = \frac{1}{n} \sec^n x, n \neq 0$	$\int \csc x dx = \ln \left \tan \frac{x}{2} \right = \ln \csc x - \cot x + C$	
$\int \csc^2 ax dx = -\frac{1}{a} \cot ax$	$\int \csc^3 x dx = -\frac{1}{2} \cot x \csc x + \frac{1}{2} \ln \csc x - \cot x $	$\int \csc^n x \cot x dx = -\frac{1}{n} \csc^n x, n \neq 0$	$\int \sec x \csc x dx = \ln \tan x $	Products of Trigonometric Functions and Monomials	
$\int x \cos x dx = \cos x + x \sin x$	$\int x \cos ax dx = \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax$	$\int x^2 \cos x dx = 2x \cos x + (x^2 - 2) \sin x$	$\int x^2 \cos ax dx = \frac{2x \cos ax}{a^2} + \frac{a^2 x^2 - 2}{a^3} \sin ax$	$\int x \sin x dx = -x \cos x + \sin x$	
$\int x \sin ax dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2}$	$\int x^2 \sin x dx = (2 - x^2) \cos x + 2x \sin x$	$\int x^2 \sin ax dx = \frac{2-a^2 x^2}{a^3} \cos ax + \frac{2x \sin ax}{a^2}$	Products of Trigonometric Functions and Exponentials		
$\int e^x \sin x dx = \frac{1}{2} e^x (\sin x - \cos x)$	$\int e^{bx} \sin ax dx = \frac{1}{a^2+b^2} e^{bx} (b \sin ax - a \cos ax)$	$\int e^{bx} \cos ax dx = \frac{1}{a^2+b^2} e^{bx} (a \sin ax + b \cos ax)$	$\int x e^x \sin x dx = \frac{1}{2} e^x (\cos x - x \cos x + x \sin x)$		
$\int x e^x \cos x dx = \frac{1}{2} e^x (x \cos x - \sin x + x \sin x)$	$\int e^x \cos x dx = \frac{1}{2} e^x (\sin x + \cos x)$				

Java

```

1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4
5 public class Main{
6     BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
7     PrintWriter writer = new PrintWriter(System.out);

```

8 StringTokenizer tokenizer = null;

9

10 void solve() throws Exception {

11 }

12 void run()throws Exception{

13 try{

14 while (true) {

15 solve();

```
16     }
17     }
18     catch(Exception e){
19     }
20     finally{
21         reader.close();
22         writer.close();
23     }
24 }
25 String next()throws Exception{
26     for(;tokenizer == null || !tokenizer.hasMoreTokens();){
27         tokenizer = new StringTokenizer(reader.readLine());
28     }
29     return tokenizer.nextToken();
30 }
31 int nextInt()throws Exception{
32     return Integer.parseInt(next());
33 }
34 double nextDouble()throws Exception{
35     return Double.parseDouble(next());
36 }
37 BigInteger nextBigInteger()throws Exception{
38     return new BigInteger(next());
39 }
40 public static void main(String args[])throws Exception{
```

```
41     (new Main()).run();
42     }
43 }
```

Vimrc

```
1 \begin{lstlisting}
2 set nu ai ci si mouse=a ts=4 sts=4 sw=4
3
4 nmap<C-A> ggVG
5 vmap<C-C> "+y
6
7 nmap<F3> : vs %<.in <CR>
8 nmap<F5> : !./%< <CR>
9 nmap<F8> : !./%< < %<.in <CR>
10 nmap<F9> : !g++ % -o %< -Wall <CR>
11
12 "nmap<F4> : !gedit % <CR>
13 "autocmd BufNewFile *.cpp 0r ~/temp.cpp
14 "set hlsearch incsearch
15
16 "syntax on
17 "filetype plugin indent on
18 \end{lstlisting}
```