Contents

1. Geometry

1.1 二维几何基础操作

```cpp
bool point_on_segment(const point &a,const line &b){
  return sgn (det (a - b.s, b.t - b.s)) == 0 && sgn (dot (b.s - a, b.t - a)) <= 0; }
bool two_side(const point &a,const point &b,const line &c) {
  return sgn (det (a - c.s, c.t - c.s)) * sgn (det (b - c.s, c.t - c.s)) < 0; }
bool intersect_judgment(const line &a,const line &b) {
  if (point_on_segment (b.s, a) || point_on_segment (b.t, a)) return true;
  if (point_on_segment (a.s, b) || point_on_segment (a.t, b)) return true;
  return two_side (a.s, a.t, b) && two_side (b.s, b.t, a); }
point line_intersect(const line &a, const line &b) {
  double s1 = det (a.t - a.s, b.s - a.s);
  double s2 = det (a.t - a.s, b.t - a.s);
  return (b.s * s2 - b.t * s1) / (s2 - s1); }
double point_to_line (const point &a, const line &b) {
```

```cpp
14      return fabs (det (b.t-b.s, a-b.s)) / dis (b.s, b.t); }
15  point project_to_line (const point &a, const line &b) {
16      return b.s + (b.t - b.s) * (dot (a - b.s, b.t - b.s) / (b.t - b.s).norm2 ()); }
17  double point_to_segment (const point &a, const line &b) {
18      if (sgn (dot (b.s - a, b.t - b.s)) * sgn (dot (b.t - a, b.t - b.s)) <= 0)
19          return fabs (det (b.t - b.s, a - b.s)) / dis (b.s, b.t);
20      return std::min (dis (a, b.s), dis (a, b.t)); }
21  bool in_polygon (const point &p, const std::vector <point> & po) {
22      int n = (int) po.size (); int counter = 0;
23      for (int i = 0; i < n; ++i) {
24          point a = po[i], b = po[(i + 1) % n];
25          if (point_on_segment (p, line (a, b))) return true;
26          int x = sgn (det (p - a, b - a)), y = sgn (a.y - p.y), z = sgn (b.y - p.y);
27          if (x > 0 && y <= 0 && z > 0) ++counter;
28          if (x < 0 && z <= 0 && y > 0) --counter; }
29      return counter != 0; }
30  std::vector <point> line_circle_intersect (const line &a, const circle &b) {
31      if (cmp (point_to_line (b.c, a), b.r) > 0) return std::vector <point> ();
32      double x = sqrt (sqr (b.r) - sqr (point_to_line (b.c, a)));
33      return std::vector <point> ({project_to_line (b.c, a) + (a.s - a.t).unit () * x, project_to_line (b.c, a) - (a.s -
        ↪ a.t).unit () * x}); }
34  double circle_intersect_area (const circle &a, const circle &b) {
35      double d = dis (a.c, b.c);
36      if (sgn (d - (a.r + b.r)) >= 0) return 0;
37      if (sgn (d - abs(a.r - b.r)) <= 0) {
38          double r = std::min (a.r, b.r);
39          return r * r * PI; }
40      double x = (d * d + a.r * a.r - b.r * b.r) / (2 * d),
41             t1 = acos (min (1., max (-1., x / a.r))),
42             t2 = acos (min (1., max (-1., (d - x) / b.r)));
43      return a.r * a.r * t1 + b.r * b.r * t2 - d * a.r * sin (t1); }
44  std::vector <point> circle_intersect (const circle &a, const circle &b) {
45      if (a.c == b.c || cmp (dis (a.c, b.c), a.r + b.r) > 0 || cmp (dis (a.c, b.c), std::abs (a.r - b.r)) < 0)
46          return std::vector <point> ();
47      point r = (b.c - a.c).unit ();
48      double d = dis (a.c, b.c);
49      double x = ((sqr (a.r) - sqr (b.r)) / d + d) / 2;
50      double h = sqrt (sqr (a.r) - sqr (x));
51      if (sgn (h) == 0) return std::vector <point> ({a.c + r * x});
52      return std::vector <point> ({a.c + r * x + r.rot90 () * h, a.c + r * x - r.rot90 () * h}); }
53  // 返回按照顺时针方向
54  std::vector <point> tangent (const point &a, const circle &b) {
55      circle p = make_circle (a, b.c);
56      return circle_intersect (p, b); }
57  std::vector <line> extangent (const circle &a, const circle &b) {
58      std::vector <line> ret;
59      if (cmp (dis (a.c, b.c), std::abs (a.r - b.r)) <= 0) return ret;
60      if (sgn (a.r - b.r) == 0) {
61          point dir = b.c - a.c;
62          dir = (dir * a.r / dir.norm ()).rot90 ();
63          ret.push_back (line (a.c + dir, b.c + dir));
64          ret.push_back (line (a.c - dir, b.c - dir));
65      } else {
66          point p = (b.c * a.r - a.c * b.r) / (a.r - b.r);
67          std::vector pp = tangent (p, a), qq = tangent (p, b);
68          if (pp.size () == 2 && qq.size () == 2) {
69              if (cmp (a.r, b.r) < 0) std::swap (pp[0], pp[1]), std::swap (qq[0], qq[1]);
70              ret.push_back(line (pp[0], qq[0]));
71              ret.push_back(line (pp[1], qq[1])); } }
72      return ret; }
73  std::vector <line> intangent (const circle &a, const circle &b) {
74      point p = (b.c * a.r + a.c * b.r) / (a.r + b.r);
75      std::vector pp = tangent (p, a), qq = tangent (p, b);
76      if (pp.size () == 2 && qq.size () == 2) {
77          ret.push_back (line (pp[0], qq[0]));
78          ret.push_back (line (pp[1], qq[1])); }
79      return ret; }
```

1.2 圆上整点

```cpp
1  void get(ll r, map<double, point>& m){//第一维极角☐第二维坐标
2      ll rr = 2*r;
3      ll sr = ll(sqrt(rr));
4      for (ll d = 1; d <= sr; d++)
```

```
5          if (rr % d == 0){
6              ll lim = sqrt(rr / (2*d));
7              for (ll a = 1; a <= lim; a++){
8                  ll b = sqrt(rr/d - a * a);
9                  if (a*a + b*b == rr/d && chk(a,b)) {
10                     ll Y = d * a * b, X = sqrt(r * r -  Y * Y);
11                     m.insert({atan2(Y,X), {X,Y}}),
12                     m.insert({atan2(-Y,X), {X,-Y}}),
13                     m.insert({atan2(Y,-X), {-X,Y}}),
14                     m.insert({atan2(-Y,-X), {-X,-Y}});
15                 }
16             }
17             if (d*d != rr){
18                 lim = sqrt(d/2);
19                 for (ll a = 1; a <= lim; a++){
20                     ll b = sqrtl(d - a*a);
21                     if (a*a + b*b == d && chk(a,b)) {
22                         ll Y = rr / d * a * b, X = sqrt(r * r -  Y * Y);
23                         m.insert({atan2(Y,X), {X,Y}}),
24                         m.insert({atan2(-Y,X), {X,-Y}}),
25                         m.insert({atan2(Y,-X), {-X,Y}}),
26                         m.insert({atan2(-Y,-X), {-X,-Y}});
27                     }
28                 }
29             }
30         }
31     m.insert({atan2(0,r), {r,0}});
32     m.insert({atan2(0,-r), {-r,0}});
33     m.insert({atan2(r,0), {0,r}});
34     m.insert({atan2(-r,0), {0,-r}});
35 }
```

1.3   直线半平面交

```
1  std::vector <point> cut (const std::vector<point> &c, line p) {
2      std::vector <point> ret;
3      if (c.empty ()) return ret;
4      for (int i = 0; i < (int) c.size (); ++i) {
5          int j = (i + 1) % (int) c.size ();
6          if (turn_left (p.s, p.t, c[i])) ret.push_back (c[i]);
7          if (two_side (c[i], c[j], p))
8              ret.push_back (line_intersect (p, line (c[i], c[j]))); }
9      return ret; }
10 bool turn_left (const line &l, const point &p) {
11     return turn_left (l.s, l.t, p); }
12 std::vector <point> half_plane_intersect (std::vector <line> h) {
13     typedef std::pair <double, line> polar;
14     std::vector <polar> g;
15     g.resize (h.size ());
16     for (int i = 0; i < (int) h.size (); ++i) {
17         point v = h[i].t - h[i].s;
18         g[i] = std::make_pair (atan2 (v.y, v.x), h[i]); }
19     sort (g.begin (), g.end (), [] (const polar &a, const polar &b) {
20         if (cmp (a.first, b.first) == 0)
21             return sgn (det (a.second.t - a.second.s, b.second.t - b.second.s)) < 0;
22         else
23             return cmp (a.first, b.first) < 0; });
24     h.resize (std::unique (g.begin (), g.end (), [] (const polar &a, const polar &b) { return cmp (a.first, b.first) == 0;
     ↪ }) - g.begin ());
25     for (int i = 0; i < (int) h.size (); ++i)
26         h[i] = g[i].second;
27     int fore = 0, rear = -1;
28     std::vector <line> ret;
29     for (int i = 0; i < (int) h.size (); ++i) {
30         while (fore < rear && !turn_left (h[i], line_intersect (ret[rear - 1], ret[rear]))) {
31             --rear; ret.pop_back (); }
32         while (fore < rear && !turn_left (h[i], line_intersect (ret[fore], ret[fore + 1])))
33             ++fore;
34         ++rear;
35         ret.push_back (h[i]); }
36     while (rear - fore > 1 && !turn_left (ret[fore], line_intersect (ret[rear - 1], ret[rear]))) {
37         --rear; ret.pop_back (); }
38     while (rear - fore > 1 && !turn_left (ret[rear], line_intersect (ret[fore], ret[fore + 1])))
39         ++fore;
```

```
40    if (rear - fore < 2) return std::vector <point> ();
41    std::vector <point> ans;
42    ans.resize (ret.size ());
43    for (int i = 0; i < (int) ret.size (); ++i)
44        ans[i] = line_intersect (ret[i], ret[(i + 1) % ret.size ()]);
45    return ans; }
```

1.4   凸包以及其他操作

```
1  vector<Point> cp;//存储凸包
2  Point O;//纵坐标最小点
3  bool cmp(Point a,Point b){//0~pi 极角排序
4      double A=(a-O)^(b-O);
5      return A>=eps || eq(A,0)&&dist(O,a)<dist(O,b);
6  }
7  void convex_hull(vector<Point>& poly)//凸包
8  {
9      int i,t=0,n=poly.size();
10     O.y=1e10;
11     for (i=0; i<n; i++)
12         if (poly[i].y<O.y || poly[i].y==O.y&&poly[i].x<O.x)
13             O=poly[i], t=i;
14     swap(poly[0],poly[t]);
15     sort(poly.begin()+1, poly.end(), cmp);
16
17     if (n<=2){
18         for (i=0; i<n; i++) cp.pb(poly[i]);
19         return;
20     }
21     cp.pb(poly[0]), cp.pb(poly[1]);
22     for (i=2; i<n; i++){
23         while (cp.size()>=2 && ((*(cp.end()-1) - *(cp.end()-2)) ^ (poly[i] - *(cp.end()-2)))<=0)
24             cp.pop_back();
25         cp.pb(poly[i]);
26     }
27  }
28
29  double area(vector<Point>& poly)//多边形求面积
30  {
31      int n=poly.size(), i;
32      double ret=0;
33      for (i=0; i<n; i++)
34          ret+=(poly[i] ^ poly[(i+1)%n])/2.0;
35      return ret;
36  }
37
38  bool operator<(Point a,Point b){return a.x<b.x||a.x==b.x&&a.y<b.y;}
39  bool operator==(Point a,Point b){return a.x==b.x&&a.y==b.y;}
40  bool cmpL(Line a,Line b){return a.k>b.k || a.k==b.k&&a.b>b.b || a.k==b.k&&a.b==b.b&&a.s<b.s ||
   ↪ a.k==b.k&&a.b==b.b&&a.s==b.s&&a.e<b.e;}
41  vector<Line> rd;
42  void combineCollinear(vector<Line>& l)//合并共线且相交的线段
43  {
44      int n=l.size(), i;
45      sort(l.begin(), l.end(), cmpL);
46      Line tmp;
47      tmp.k=tmp.b=-1e18;
48      for (i=0; i<n; i++){
49          if (l[i].k!=tmp.k || l[i].b!=tmp.b || tmp.e<l[i].s){
50              if (i) rd.pb(tmp);
51              tmp=l[i];
52          }
53          if (tmp.e<l[i].e) tmp.e=l[i].e;
54      }
55      rd.pb(tmp);
56  }
```

1.5   直线与凸包交点

```
1  // a 是顺时针凸包，i1 为 x 最小的点，j1 为 x 最大的点 需保证 j1 > i1
2  // n 是凸包上的点数，a 需复制多份或写循环数组类
3  int lowerBound(int le, int ri, const P & dir) {
4      while (le < ri) {
5          int mid((le + ri) / 2);
6          if (sign((a[mid + 1] - a[mid]) * dir) <= 0) {
```

```
7        le = mid + 1;
8    } else ri = mid; } }
9    return le; }
10 int boundLower(int le, int ri, const P & s, const P & t) {
11     while (le < ri) {
12         int mid((le + ri + 1) / 2);
13         if (sign((a[mid] - s) * (t - s)) <= 0) {
14             le = mid;
15         } else ri = mid - 1; }
16     return le; }
17 void calc(P s, P t) {
18     if(t < s) swap(t, s);
19     int i3(lowerBound(i1, j1, t - s)); // 和上凸包的切点
20     int j3(lowerBound(j1, i1 + n, s - t)); // 和下凸包的切点
21     int i4(boundLower(i3, j3, s, t)); // 如果有交则是右侧的交点，与 a[i4]~a[i4+1] 相交 要判断是否有交的话 就手动 check 一下
22     int j4(boundLower(j3, i3 + n, t, s)); // 如果有交左侧的交点，与 a[j4]~a[j4+1] 相交
23      // 返回的下标不一定在 [0 ~ n-1] 内
24 }
```

1.6 点到凸包切线

```
1 typedef vector<vector<P>> Convex;
2 #define sz(x) ((int) x.size())
3 int lb(P x, const vector<P> & v, int le, int ri, int sg) {
4     if (le > ri) le = ri;
5     int s(le), t(ri);
6     while (le != ri) {
7         int mid((le + ri) / 2);
8         if (sign(det(v[mid] - x, v[mid + 1] - v[mid])) == sg)
9             le = mid + 1; else ri = mid; }
10     return le; } // le 即为下标，按需返回
11 // v[0] 为顺时针上凸壳，v[1] 为顺时针下凸壳，均允许起始两个点横坐标相同
12 // 返回值为真代表严格在凸包外，顺时针旋转在 d1 方向先碰到凸包
13 bool getTan(P x, const Convex & v, int & d1, int & d2) {
14     if (x.x < v[0][0].x) {
15         d1 = lb(x, v[0], 0, sz(v[0]) - 1, 1);
16         d2 = lb(x, v[1], 0, sz(v[1]) - 1, -1) + (int) v[0].size() - 1;
17         return true;
18     } else if(x.x > v[0].back().x) {
19         d1 = lb(x, v[1], 0, sz(v[1]) - 1, 1) + (int) v[0].size() - 1;
20         d2 = lb(x, v[0], 0, sz(v[0]) - 1, -1);
21         return true;
22     } else {
23         for(int d(0); d < 2; d++) {
24             int id(lower_bound(v[d].begin(), v[d].end(), x, [&](const P & a, const P & b) { return d == 0 ? a < b : b < a; })
                  ↪ - v[d].begin());
25             if (id && (id == sz(v[d]) || det(v[d][id - 1] - x, v[d][id] - x) > 0)) {
26                 d1 = lb(x, v[d], id, sz(v[d]) - 1, 1);
27                 d2 = lb(x, v[d], 0, id, -1);
28                 if (d) {
29                     d1 += (int) v[0].size() - 1;
30                     d2 += (int) v[0].size() - 1; }
31                 return true; } } }
32     return false; }
```

1.7 闵可夫斯基和

```
1 // cv[0..1] 为两个顺时针凸包，其中起点等于终点，求出的闵可夫斯基和不一定是严格凸包
2 int i[2] = {0, 0}, len[2] = {(int)cv[0].size() - 1, (int)cv[1].size() - 1};
3 vector<point> mnk;
4 mnk.push_back(cv[0][0] + cv[1][0]);
5 do {
6     int d = (det(cv[0][i[0] + 1] - cv[0][i[0]], cv[1][i[1] + 1] - cv[1][i[1]]) >= 0);
7     mnk.push_back(cv[d][i[d] + 1] - cv[d][i[d]] + mnk.back());
8     i[d] = (i[d] + 1) % len[d];
9 } while(i[0] || i[1]);
```

1.8 Delaunay 三角剖分

```
1 /* Delaunay Triangulation 随机增量算法 :
2 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
3 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)
4 Triangulation::find 返回包含某点的三角形
5 Triangulation::add_point 将某点加入三角剖分
6 某个 Triangle 在三角剖分中当且仅当它的 has_child 为 0
```

```cpp
 7    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] */
 8    const int N = 100000 + 5, MAX_TRIS = N * 6;
 9    const double EPSILON = 1e-6, PI = acos(-1.0);
10    struct Point {
11      double x,y; Point():x(0),y(0){}
12        Point(double x, double y):x(x),y(y){}
13      bool operator ==(Point const& that)const {return x==that.x&&y==that.y;} };
14    inline double sqr(double x) { return x*x; }
15    double dist_sqr(Point const& a, Point const& b){return sqr(a.x-b.x)+sqr(a.y-b.y);}
16    bool in_circumcircle(Point const& p1, Point const& p2, Point const& p3, Point const& p4) {
17      double u11 = p1.x-p4.x, u21 = p2.x-p4.x, u31 = p3.x-p4.x;
18      double u12 = p1.y-p4.y, u22 = p2.y-p4.y, u32 = p3.y-p4.y;
19      double u13 = sqr(p1.x)-sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
20      double u23 = sqr(p2.x)-sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
21      double u33 = sqr(p3.x)-sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
22      double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33 + u11*u22*u33;
23      return det > EPSILON; }
24    double side(Point const& a, Point const& b, Point const& p) { return (b.x-a.x)*(p.y-a.y) - (b.y-a.y)*(p.x-a.x);}
25    typedef int SideRef; struct Triangle; typedef Triangle* TriangleRef;
26    struct Edge {
27      TriangleRef tri; SideRef side; Edge() : tri(0), side(0) {}
28      Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {} };
29    struct Triangle {
30      Point p[3];Edge edge[3];TriangleRef child[3]; Triangle(){}
31      Triangle(Point const& p0,Point const& p1,Point const& p2){
32        p[0] = p0; p[1] = p1; p[2] = p2;
33          child[0] = child[1] = child[2] = 0; }
34      bool has_child() const { return child[0] != 0; }
35      int num_child() const {
36        return child[0] == 0 ? 0
37          : child[1] == 0 ? 1
38          : child[2] == 0 ? 2 : 3; }
39      bool contains(Point const& q) const {
40        double a=side(p[0],p[1],q), b=side(p[1],p[2],q), c=side(p[2],p[0],q);
41        return a >= -EPSILON && b >= -EPSILON && c >= -EPSILON; }
42    } triange_pool[MAX_TRIS], *tot_triangles;
43    void set_edge(Edge a, Edge b) {
44      if (a.tri) a.tri->edge[a.side] = b;
45      if (b.tri) b.tri->edge[b.side] = a; }
46    class Triangulation {
47      public:
48        Triangulation() {
49          const double LOTS = 1e6;
50          the_root = new(tot_triangles++) Triangle (Point(-LOTS,-LOTS), Point(+LOTS,-LOTS), Point(0,+LOTS)); }
51        TriangleRef find(Point p) const { return find(the_root,p); }
52        void add_point(Point const& p) { add_point(find(the_root,p),p); }
53      private:
54        TriangleRef the_root;
55        static TriangleRef find(TriangleRef root,Point const& p){
56          for( ; ; ) {
57            if (!root->has_child()) return root;
58            else for (int i = 0; i < 3 && root->child[i] ; ++i)
59                if (root->child[i]->contains(p))
60                  {root = root->child[i]; break;} } }
61      void add_point(TriangleRef root, Point const& p) {
62        TriangleRef tab,tbc,tca;
63        tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
64        tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
65        tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
66        set_edge(Edge(tab,0),Edge(tbc,1)); set_edge(Edge(tbc,0),Edge(tca,1));
67        set_edge(Edge(tca,0),Edge(tab,1)); set_edge(Edge(tab,2),root->edge[2]);
68        set_edge(Edge(tbc,2),root->edge[0]); set_edge(Edge(tca,2),root->edge[1]);
69        root->child[0]=tab;root->child[1]=tbc;root->child[2]=tca;
70        flip(tab,2); flip(tbc,2); flip(tca,2); }
71      void flip(TriangleRef tri, SideRef pi) {
72        TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
73        if(!trj || !in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
74        TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj], tri->p[pi]);
75        TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi], trj->p[pj]);
76        set_edge(Edge(trk,0), Edge(trl,0));
77        set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
78        set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
79        tri->child[0]=trk; tri->child[1]=trl; tri->child[2]=0;
```

```
80      trj->child[0]=trk; trj->child[1]=trl; trj->child[2]=0;
81      flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2); } };
82 int n; Point ps[N];
83 void build(){
84   tot_triangles = triange_pool; cin >> n;
85   for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
86   random_shuffle(ps, ps + n); Triangulation tri;
87   for(int i = 0; i < n; ++ i) tri.add_point(ps[i]); }
88 //The Euclidean minimum spanning tree of a set of points is a subset of the Delaunay triangulation of the same points.
89 //Connecting the centers of the circumcircles produces the Voronoi diagram.
90 //No point in P is inside the circumcircle of any triangle.
91 //Maximize the minimum angle of all the angles of the triangles.
```

1.9   三角形 与 费马点

```
1 point incenter (const point &a, const point &b, const point &c) {
2     double p = dis (a, b) + dis (b, c) + dis (c, a);
3     return (a * dis (b, c) + b * dis (c, a) + c * dis (a, b)) / p; }
4 point circumcenter (const point &a, const point &b, const point &c) {
5     point p = b - a, q = c - a, s (dot (p, p) / 2, dot (q, q) / 2);
6     double d = det (p, q);
7     return a + point (det (s, point (p.y, q.y)), det (point (p.x, q.x), s)) / d; }
8 point orthocenter (const point &a, const point &b, const point &c) {
9     return a + b + c - circumcenter (a, b, c) * 2.0; }
10 point fermat_point (const point &a, const point &b, const point &c) {
11    if (a == b) return a; if (b == c) return b;
12    if (c == a) return c;
13    double ab = dis (a, b), bc = dis (b, c), ca = dis (c, a);
14    double cosa = dot (b - a, c - a) / ab / ca;
15    double cosb = dot (a - b, c - b) / ab / bc;
16    double cosc = dot (b - c, a - c) / ca / bc;
17    double sq3 = PI / 3.0; point mid;
18    if (sgn (cosa + 0.5) < 0) mid = a;
19    else if (sgn (cosb + 0.5) < 0) mid = b;
20    else if (sgn (cosc + 0.5) < 0) mid = c;
21    else if (sgn (det (b - a, c - a)) < 0)
22       mid = line_intersect (line (a, b + (c - b).rot (sq3)), line (b, c + (a - c).rot (sq3)));
23    else
24       mid = line_intersect (line (a, c + (b - c).rot (sq3)), line (c, b + (a - b).rot (sq3)));
25    return mid; }
```

1.10   圆并

```
1 int C; circle c[MAXN]; double area[MAXN];
2 struct event {
3    point p; double ang; int delta;
4    event (point p = point (), double ang = 0, int delta = 0) : p(p), ang(ang), delta(delta) {}
5    bool operator < (const event &a) { return ang < a.ang; } };
6 void addevent(const circle &a, const circle &b, std::vector<event> &evt, int &cnt) {
7    double d2 = (a.c - b.c).norm2(), dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
8       pRatio = sqrt (std::max (0., -(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4)));
9    point d = b.c - a.c, p = d.rot(PI / 2),
10      q0 = a.c + d * dRatio + p * pRatio,
11      q1 = a.c + d * dRatio - p * pRatio;
12    double ang0 = atan2 ((q0 - a.c).y, (q0 - a.c).x), ang1 = atan2 ((q1 - a.c).x, (q1 - a.c).y);
13    evt.emplace_back(q1,ang1,1); evt.emplace_back(q0,ang0,-1);
14    cnt += ang1 > ang0; }
15 bool issame(const circle &a, const circle &b) {
16    return sgn((a.c-b.c).norm()) == 0 && sgn(a.r-b.r) == 0; }
17 bool overlap(const circle &a, const circle &b) {
18    return sgn(a.r - b.r - (a.c - b.c).norm()) >= 0; }
19 bool intersect(const circle &a, const circle &b) {
20    return sgn((a.c - b.c).norm() - a.r - b.r) < 0; }
21 void solve() {
22    std::fill (area, area + C + 2, 0);
23    for (int i = 0; i < C; ++i) { int cnt = 1;
24       std::vector<event> evt;
25       for (int j=0; j<i; ++j) if (issame(c[i],c[j])) ++cnt;
26       for (int j = 0; j < C; ++j)
27          if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) ++cnt;
28       for (int j = 0; j < C; ++j)
29          if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i], c[j]))
30             addevent(c[i], c[j], evt, cnt);
31       if (evt.empty()) area[cnt] += PI * c[i].r * c[i].r;
32       else {
```

```
33          std::sort(evt.begin(), evt.end());
34          evt.push_back(evt.front());
35          for (int j = 0; j + 1 < (int)evt.size(); ++j) {
36              cnt += evt[j].delta;
37              area[cnt] += det(evt[j].p,evt[j + 1].p) / 2;
38              double ang = evt[j + 1].ang - evt[j].ang;
39              if (ang < 0) ang += PI * 2;
40              area[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2; } } } }
```

1.11   最小覆盖圆

```
1  circle minimum_circle (std::vector <point> p) { circle ret;
2      std::random_shuffle (p.begin (), p.end ());
3      for (int i = 0; i < (int) p.size (); ++i)
4          if (!in_circle (p[i], ret)) { ret = circle (p[i], 0);
5              for (int j = 0; j < i; ++j)
6                  if (!in_circle (p[j], ret)) {
7                      ret = make_circle (p[j], p[i]);
8                      for (int k = 0; k < j; ++k)
9                          if (!in_circle (p[k], ret))
10                             ret = make_circle(p[i],p[j],p[k]); } }
11     return ret; }
```

1.12   多边形与圆交

```
1  double sector_area (const point &a, const point &b, const double &r) {
2      double c = (2.0 * r * r - (a - b).norm2 ()) / (2.0 * r * r);
3      double al = acos (c);
4      return r * r * al / 2.0; }
5  double area(const point &a,const point &b,const double &r) {
6      double dA = dot (a, a), dB = dot (b, b), dC = point_to_segment (point (), line (a, b)), ans = 0.0;
7      if (sgn (dA - r * r) <= 0 && sgn (dB - r * r) <= 0) return det (a, b) / 2.0;
8      point tA = a.unit () * r;
9      point tB = b.unit () * r;
10     if (sgn (dC - r) > 0) return sector_area (tA, tB, r);
11     std::pair <point, point> ret = line_circle_intersect (line (a, b), circle (point (), r));
12     if (sgn (dA - r * r) > 0 && sgn (dB - r * r) > 0) {
13         ans += sector_area (tA, ret.first, r);
14         ans += det (ret.first, ret.second) / 2.0;
15         ans += sector_area (ret.second, tB, r);
16         return ans; }
17     if (sgn (dA - r * r) > 0)
18         return det (ret.first, b) / 2.0 + sector_area (tA, ret.first, r);
19     else
20         return det (a, ret.second) / 2.0 + sector_area (ret.second, tB, r); }
21 double solve(const std::vector<point> &p, const circle &c) {
22     double ret = 0.0;
23     for (int i = 0; i < (int) p.size (); ++i) {
24         int s = sgn (det (p[i] - c.c, p[ (i + 1) % p.size ()] - c.c));
25         if (s > 0)
26             ret += area (p[i] - c.c, p[ (i + 1) % p.size ()] - c.c, c.r);
27         else
28             ret -= area (p[ (i + 1) % p.size ()] - c.c, p[i] - c.c, c.r); }
29     return fabs (ret); }
```

1.13   阿波罗尼茨圆

```
1  硬币问题：易知两两相切的圆半径为 r1, r2, r3，求与他们都相切的圆的半径 r4
2  分母取负号，答案再取绝对值，为外切圆半径
3  分母取正号为内切圆半径
```
$$4 \quad // \ r_4^{\pm} = \frac{r_1 r_2 r_3}{r_1 r_2 + r_1 r_3 + r_2 r_3 \pm 2\sqrt{r_1 r_2 r_3 (r_1 + r_2 + r_3)}}$$

1.14   圆幂 圆反演 根轴

圆幂: 半径为 R 的圆 O, 任意一点 P 到 O 的幂为 $h = OP^2 - R^2$

圆幂定理: 过 P 的直线交圆在 A 和 B 两点, 则 $PA \cdot PB = |h|$

根轴: 到两圆等幂点的轨迹是一条垂直于连心线的直线

反演: 已知一圆 C, 圆心为 O, 半径为 r, 如果 P 与 P' 在过圆心 O 的直线上, 且 $OP \cdot OP' = r^2$, 则称 P 与 P' 关于 O 互为反演. 一般 C 取单位圆.

反演的性质:

不过反演中心的直线反形是过反演中心的圆, 反之亦然.

不过反演中心的圆, 它的反形是一个不过反演中心的圆.

两条直线在交点 A 的夹角, 等于它们的反形在相应点 A' 的夹角, 但方向相反.

两个相交圆周在交点 A 的夹角等于它们的反形在相应点 A' 的夹角, 但方向相反.

直线和圆周在交点 A 的夹角等于它们的反演图形在相应点 A' 的夹角, 但方向相反.

正交圆反形也正交. 相切圆反形也相切, 当切点为反演中心时, 反形为两条平行线.

1.15　三维绕轴旋转 三维基础操作

```
1  /* 大拇指指向x轴正方向时，4指弯曲由y轴正方向指向z轴正方向
2   大拇指沿着原点到点(x, y, z)的向量，4指弯曲方向旋转w度 */
3  /* (x, y, z) * A = (x_new, y_new, z_new), 行向量右乘转移矩阵 */
4  void calc(D x, D y, D z, D w) { // 三维绕轴旋转
5      w = w * pi / 180;
6      memset(a, 0, sizeof(a));
7      s1 = x * x + y * y + z * z;
8      a[0][0] = ((y*y+z*z)*cos(w)+x*x)/s1; a[0][1] = x*y*(1-cos(w))/s1+z*sin(w)/sqrt(s1); a[0][2] =
         ↪ x*z*(1-cos(w))/s1-y*sin(w)/sqrt(s1);
9      a[1][0] = x*y*(1-cos(w))/s1-z*sin(w)/sqrt(s1); a[1][1] = ((x*x+z*z)*cos(w)+y*y)/s1; a[1][2] =
         ↪ y*z*(1-cos(w))/s1+x*sin(w)/sqrt(s1);
10     a[2][0] = x*z*(1-cos(w))/s1+y*sin(w)/sqrt(s1); a[2][1] = y*z*(1-cos(w))/s1-x*sin(w)/sqrt(s1); a[2][2] =
         ↪ ((x*x+y*y)*cos(w)+z*z)/s1;
11 }
12 point3D cross (const point3D & a, const point3D & b) {
13     return point3D(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x); }
14 double mix(point3D a, point3D b, point3D c) {
15     return dot(cross(a, b), c); }
16 struct Line { point3D s, t; };
17 struct Plane { // nor 为单位法向量，离原点距离 m
18     point3D nor; double m;
19     Plane(point3D r, point3D a) : nor(r){
20         nor = 1 / r.len() * r;
21         m = dot(nor, a); } };
22 // 以下函数注意除以0的情况
23 // 点到平面投影
24 point3D project_to_plane(point3D a, Plane b) {
25 return a + (b.m - dot(a, b.nor)) * b.nor; }
26 // 点到直线投影
27 point3D project_to_line(point3D a, Line b) {
28 return b.s + dot(a - b.s, b.t - b.s) / dot(b.t - b.s, b.t - b.s) * (b.t - b.s); }
29 // 直线与直线最近点
30 pair<point3D, point3D> closest_two_lines(Line x, Line y) {
31 double a = dot(x.t - x.s, x.t - x.s);
32 double b = dot(x.t - x.s, y.t - y.s);
33 double e = dot(y.t - y.s, y.t - y.s);
34 double d = a*e - b*b; point3D r = x.s - y.s;
35 double c = dot(x.t - x.s, r), f = dot(y.t - y.s, r);
36 double s = (b*f - c*e) / d, t = (a*f - c*b) / d;
37 return {x.s + s*(x.t - x.s), y.s + t*(y.t - y.s)}; }
38 // 直线与平面交点
39 point3D intersect(Plane a, Line b) {
40 double t = dot(a.nor, a.m * a.nor - b.s) / dot(a.nor, b.t - b.s);
41 return b.s + t * (b.t - b.s); }
42 // 平面与平面求交线
43 Line intersect(Plane a, Plane b) {
44 point3D d=cross(a.nor,b.nor), d2=cross(b.nor,d);
45 double t = dot(d2, a.nor);
46 point3D s = 1 / t * (a.m - dot(b.m * b.nor, a.nor)) * d2 + b.m * b.nor;
47 return (Line) {s, s + d}; }
48 // 三个平面求交点
49 point3D intersect(Plane a, Plane b, Plane c) {
50 return intersect(a, intersect(b, c));
51 point3D c1 (a.nor.x, b.nor.x, c.nor.x);
52 point3D c2 (a.nor.y, b.nor.y, c.nor.y);
53 point3D c3 (a.nor.z, b.nor.z, c.nor.z);
54 point3D c4 (a.m, b.m, c.m);
55 return 1 / mix(c1, c2, c3) * point3D(mix(c4, c2, c3), mix(c1, c4, c3), mix(c1, c2, c4)); }
```

1.16　最小覆盖球

```
1  vector<point3D> vec;
2  Circle calc() {
3      if(vec.empty()) { return Circle(point3D(0, 0, 0), 0);
4      }else if(1 == (int)vec.size()) {return Circle(vec[0], 0);
5      }else if(2 == (int)vec.size()) {
6          return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
7      }else if(3 == (int)vec.size()) {
8          double r = (vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] - vec[0]).len() / 2 / fabs(cross(vec[0] -
             ↪ vec[2], vec[1] - vec[2]).len());
9          Plane ppp1 = Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0]));
10         return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])), Plane(vec[2] - vec[1], 0.5 * (vec[2] +
             ↪ vec[1]))), Plane(cross(vec[1] - vec[0], vec[2] - vec[0]), vec[0])), r);
```

```
11    }else {
12        point3D o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])), Plane(vec[2] - vec[0], 0.5 * (vec[2] +
          ↪ vec[0]))), Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0])))));
13        return Circle(o, (o - vec[0]).len()); } }
14 Circle miniBall(int n) {
15    Circle res(calc());
16    for(int i(0); i < n; i++) {
17        if(!in_circle(a[i], res)) { vec.push_back(a[i]);
18            res = miniBall(i); vec.pop_back();
19            if(i) { point3D tmp(a[i]);
20                memmove(a + 1, a, sizeof(point3D) * i);
21                a[0] = tmp; } } }
22    return res; }
23 int main() {
24    int n; scanf("%d", &n);
25    for(int i(0); i < n; i++) a[i].scan();
26    sort(a, a + n); n = unique(a, a + n) - a;
27    vec.clear(); random_shuffle(a, a + n);
28    printf("%.10f\n", miniBall(n).r); }
```

1.17   球面基础

球面距离: 连接球面两点的大圆劣弧 (所有曲线中最短)

球面角: 球面两个大圆弧所在半平面形成的二面角

球面凸多边形: 把一个球面多边形任意一边向两方无限延长成大圆, 其余边都在此大圆的同旁.

球面角盈E: 球面凸n边形的内角和与$(n-2)\pi$的差

离北极夹角θ, 距离h的球冠: $S = 2\pi Rh = 2\pi R^2(1 - \cos\theta)$, $V = \frac{\pi h^2}{3}(3R - h)$

球面凸n边形面积: $S = ER^2$

1.18   经纬度球面距离

```
1 // lontitude 经度范围: ±π, latitude 纬度范围: ±π/2
2 double sphereDis(double lon1, double lat1, double lon2, double lat2, double R) {
3     return R * acos(cos(lat1) * cos(lat2) * cos(lon1 - lon2) + sin(lat1) * sin(lat2)); }
```

1.19   最近点对

```
1 double solve(point *p, int l, int r) { // 左闭右开 返回距离平方
2     if(l + 1 >= r) return INF;
3     int m = (l + r) / 2; double mx = p[m].x; vector <point> v;
4     double ret = min(solve(p, l, m), solve(p, m, r));
5     for(int i = l; i < r; i ++)
6         if(sqr(p[i].x - mx) < ret) v.push_back(p[i]);
7     sort(v.begin(), v.end(), by_y);
8     for(int i = 0; i < v.size(); i ++)
9         for(int j = i + 1; j < v.size(); j ++) {
10            if(sqr(v[i].y - v[j].y) > ret) break;
11            ret = min(ret, (v[i] - v[j]).len2()); }
12    return ret; } // 需先对p[]按x进行排序
```

1.20   三维凸包

```
1 int mark[N][N], cnt;
2 D mix(const Point & a, const Point & b, const Point & c) { return a.dot(b.cross(c)); }
3 double volume(int a, int b, int c, int d) { return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]); }
4 typedef array<int, 3> Face; vector<Face> face;
5 inline void insert(int a, int b, int c) { face.push_back({a, b, c}); }
6 void add(int v) {
7     vector<Face> tmp; int a, b, c; cnt++;
8     for(auto f : face)
9         if(sign(volume(v, f[0], f[1], f[2])) < 0)
10            for(int i : f) for(int j : f) mark[i][j] = cnt;
11        else tmp.push_back(f);
12    face = tmp;
13    for(int i(0); i < (int)tmp.size(); i++) {
14        a = face[i][0]; b = face[i][1]; c = face[i][2];
15        if(mark[a][b] == cnt) insert(b, a, v);
16        if(mark[b][c] == cnt) insert(c, b, v);
17        if(mark[c][a] == cnt) insert(a, c, v); } }
18 int Find(int n) {
19    for(int i(2); i < n; i++) {
20        Point ndir=(info[0]-info[i]).cross(info[1]-info[i]);
21        if(ndir==Point(0,0,0))continue;swap(info[i],info[2]);
22        for(int j = i + 1; j < n; j++) if(sign(volume(0, 1, 2, j)) != 0) {
23            swap(info[j], info[3]); insert(0, 1, 2), insert(0, 2, 1); return 1; } } }
24 int main() {
25    int n; scanf("%d", &n);
```

```
26    for(int i(0); i < n; i++) info[i].scan();
27    random_shuffle(info, info + n);
28    Find(n);
29    for(int i = 3; i < n; i++) add(i); }
```

1.21   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;
18     turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
19     cout<<r<<endl; }
```

1.22   相关公式

1.22.1 Heron's Formula

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$
$$p = \frac{a+b+c}{2}$$

1.22.2   四面体内接球球心

假设 s_i 是第 i 个顶点相对面的面积 则有

$$\begin{cases} x = \dfrac{s_1 x_1 + s_2 x_2 + s_3 x_3 + s_4 x_4}{s_1 + s_2 + s_3 + s_4} \\ y = \dfrac{s_1 y_1 + s_2 y_2 + s_3 y_3 + s_4 y_4}{s_1 + s_2 + s_3 + s_4} \\ z = \dfrac{s_1 z_1 + s_2 z_2 + s_3 z_3 + s_4 z_4}{s_1 + s_2 + s_3 + s_4} \end{cases}$$

体积可以使用 $1/6$ 混合积求, 内接球半径为

$$r = \frac{3V}{s_1 + s_2 + s_3 + s_4}$$

1.22.3   三角形内心

$$\frac{a\vec{A} + b\vec{B} + c\vec{C}}{a+b+c}$$

1.22.4   三角形外心

$$\frac{\vec{A} + \vec{B} - \frac{\overrightarrow{BC} \cdot \overrightarrow{CA}}{\overrightarrow{AB} \times \overrightarrow{BC}} \overrightarrow{AB}^T}{2}$$

1.22.5   三角形垂心

$$\vec{H} = 3\vec{G} - 2\vec{O}$$

1.22.6   三角形偏心

$$\frac{-a\vec{A} + b\vec{B} + c\vec{C}}{-a+b+c}$$

剩余两点的同理.

1.22.7   三角形内接外接圆半径

$$r = \frac{2S}{a+b+c}, R = \frac{abc}{4S}$$

1.22.8 Pick's Theorem

$$S = I + \frac{B}{2} - 1$$

S is the area of lattice polygon, I is the number of lattice interior points, and B is the number of lattice boundary points.

1.22.9 Euler's Formula

For convex polyhedron: $V - E + F = 2$.

For planar graph: $|F| = |E| - |V| + n + 1$, n denotes the number of connected components.

1.23 三角公式

$$\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b$$
$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$
$$\tan(a \pm b) = \frac{\tan(a) \pm \tan(b)}{1 \mp \tan(a)\tan(b)}$$
$$\tan(a) \pm \tan(b) = \frac{\sin(a \pm b)}{\cos(a)\cos(b)}$$
$$\sin(a) + \sin(b) = 2\sin(\frac{a+b}{2})\cos(\frac{a-b}{2})$$
$$\sin(a) - \sin(b) = 2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$$
$$\cos(a) + \cos(b) = 2\cos(\frac{a+b}{2})\cos(\frac{a-b}{2})$$
$$\cos(a) - \cos(b) = -2\sin(\frac{a+b}{2})\sin(\frac{a-b}{2})$$

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

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

1.23.1 超球坐标系

$$
\begin{aligned}
x_1 &= r\cos(\phi_1) \\
x_2 &= r\sin(\phi_1)\cos(\phi_2) \\
&\cdots \\
x_{n-1} &= r\sin(\phi_1)\cdots\sin(\phi_{n-2})\cos(\phi_{n-1}) \\
x_n &= r\sin(\phi_1)\cdots\sin(\phi_{n-2})\sin(\phi_{n-1}) \\
\phi_{n-1} &\in [0, 2\pi] \\
\forall i = 1..n-1 \, \phi_i &\in [0, \pi]
\end{aligned}
$$

1.23.2 三维旋转公式

绕着 $(0,0,0) - (ux, uy, uz)$ 旋转 θ, (ux, uy, uz) 是单位向量

$$R = \begin{matrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{matrix}.$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

1.23.3 立体角公式

$$\phi : \text{二面角}$$
$$\Omega = (\phi_{ab} + \phi_{bc} + \phi_{ac}) \, \text{rad} - \pi \, \text{sr}$$

$$\tan\left(\frac{1}{2}\Omega/\text{rad}\right) = \frac{\left| \vec{a} \ \vec{b} \ \vec{c} \right|}{abc + \left(\vec{a}\cdot\vec{b}\right)c + (\vec{a}\cdot\vec{c})b + \left(\vec{b}\cdot\vec{c}\right)a}$$

$$\theta_s = \frac{\theta_a + \theta_b + \theta_c}{2}$$

1.23.4 常用体积公式

- Pyramid $V = \frac{1}{3}Sh$.
- Sphere $V = \frac{4}{3}\pi R^3$.
- Frustum $V = \frac{1}{3}h(S_1 + \sqrt{S_1 S_2} + S_2)$.
- Ellipsoid $V = \frac{4}{3}\pi abc$.

1.23.5 高维球体积

$$V_2 = \pi R^2, \ S_2 = 2\pi R$$
$$V_3 = \frac{4}{3}\pi R^3, \ S_3 = 4\pi R^2$$
$$V_4 = \frac{1}{2}\pi^2 R^4, \ S_4 = 2\pi^2 R^3$$
$$\text{Generally}, V_n = \frac{2\pi}{n}V_{n-2}, \ S_{n-1} = \frac{2\pi}{n-2}S_{n-3}$$
$$\text{Where}, S_0 = 2, \ V_1 = 2, \ S_1 = 2\pi, \ V_2 = \pi$$

2. Data Structure
2.1 KD 树

```
1  //P-节点类  d[2]-坐标 mi[2]-最小坐标 mx[2]-最大坐标 l,r-左右儿子 s-和  v-值 lz-懒惰标记 sz-子树大小
2  int rt,X0,X1,Y0,Y1,X,Y,O,an;
3  struct P{int d[2],mi[2],mx[2],l,r,s,v,lz,sz;}T[111111];
4  bool cmp(P a,P b) {
5      return a.d[O]<b.d[O]||a.d[O]==b.d[O]&&a.d[O^1]<b.d[O^1];}
6  void ADD(int k,int z) {
7      if(!k)return;T[k].v+=z;T[k].s+=T[k].sz*z;T[k].lz+=z;}
8  void pd(int k) { if(T[k].lz)
9      ADD(T[k].l,T[k].lz),ADD(T[k].r,T[k].lz),T[k].lz=0;}
10 int bd(int l,int r,int o){//将[l,r]建树，维度是o，返回[l,r]的根
11     O=o;int k=l+r>>1,i;nth_element(T+l+1,T+k+1,T+r+1,cmp);
12     for(i=0;i<2;i++)T[k].mi[i]=T[k].mx[i]=T[k].d[i];
13     T[k].lz=T[k].v=T[k].s=0;
14     if(l<k)for(O=T[k].l=bd(l,k-1,o^1),i=0;i<2;i++)
15        Min(T[k].mi[i],T[O].mi[i]), Max(T[k].mx[i],T[O].mx[i]);
16     if(k<r)for(O=T[k].r=bd(k+1,r,o^1),i=0;i<2;i++)
17        Min(T[k].mi[i],T[O].mi[i]), Max(T[k].mx[i],T[O].mx[i]);
18     T[k].sz=T[T[k].l].sz+T[T[k].r].sz+1;return k;
19 }void add(int k,int z){//范围+z，范围为[X0,X1][Y0,Y1]
20     if(X0>T[k].mx[0] || X1<T[k].mi[0] || Y0>T[k].mx[1] || Y1<T[k].mi[1]) return;
21     if(X0<=T[k].mi[0] && X1>=T[k].mx[0] && Y0<=T[k].mi[1] && Y1>=T[k].mx[1]) {ADD(k,z);return;}
22     if(X0<=T[k].d[0] && T[k].d[0]<=X1 && Y0<=T[k].d[1] && T[k].d[1]<=Y1) T[k].v+=z;
23     pd(k);add(T[k].l,z);add(T[k].r,z);
24     T[k].s=T[k].v+T[T[k].l].s+T[T[k].r].s;
25 }int qu(int k){//询问范围和
26     if(X0>T[k].mx[0] || X1<T[k].mi[0] || Y0>T[k].mx[1] || Y1<T[k].mi[1]) return 0;
27     if(X0<=T[k].mi[0] && X1>=T[k].mx[0] && Y0<=T[k].mi[1] && Y1>=T[k].mx[1]) return T[k].s;
28     int an=0;
29     if(X0<=T[k].d[0] && T[k].d[0]<=X1 && Y0<=T[k].d[1] && T[k].d[1]<=Y1) an=T[k].v;
30     pd(k); return an+qu(T[k].l)+qu(T[k].r);
31 }//外部将坐标存入T[i].d[2]后调用rt=bd(1,n,0)
32 int dis(P a){return max(a.mi[0]-X,0) + max(X-a.mx[0],0) + max(a.mi[1]-Y,0) + max(Y-a.mx[1],0);}
33 void fdmin(int k,int o){//找离当前点(X,Y)最近的点
34     int d=abs(T[k].d[0]-X)+abs(T[k].d[1]-Y),dl=inf,dr=inf;
35     an=min(an,d);
36     if(T[k].l)dl=dis(T[T[k].l]);
37     if(T[k].r)dr=dis(T[T[k].r]);
38     if(dl<dr){if(dl<an)fdmin(T[k].l,o^1); if(dr<an)fdmin(T[k].r,o^1);
39     }else{if(dr<an)fdmin(T[k].r,o^1); if(dl<an)fdmin(T[k].l,o^1);}}
```

2.2 点分树

```
1  #include<cstdio>
2  #include<cstring>
3  #include<iostream>
4  #include<algorithm>
5  #include<cmath>
6  #include<vector>
7  #define N 100500
8  #define int long long
9  using namespace std;
10 int val[N];
11 int e=1,head[N];
12 struct edge{
13     int v,w,next;
14 }ed[N<<1];
15 void add(int u,int v,int w){
16     ed[e].v=v;ed[e].w=w;
17     ed[e].next=head[u];
18     head[u]=e++;
19 }
20 int a[2*N],L[N],R[N],cnt,minn[2*N][20],lg[2*N];
21 void dfs(int x,int fa){
22     a[++cnt]=val[x];L[x]=cnt;
23     for(int i=head[x];i;i=ed[i].next){
24         int v=ed[i].v;
25         if(v==fa)continue;
26         val[v]=val[x]+ed[i].w;
27         dfs(v,x);a[++cnt]=val[x];
28     }
29     if(a[cnt]!=val[x])a[++cnt]=val[x];
```

```
30        R[x]=cnt;
31  }
32  int dis(int x,int y){
33        int l=R[x],r=L[y];
34        if(l>r)swap(l,r);
35        return val[x]+val[y]-2*min(minn[l][lg[r-l+1]],minn[r-(1<<lg[r-l+1])+1][lg[r-l+1]]);
36  }
37  int size[N],maxn[N];
38  bool vis[N];
39  int root,sum,rt,n,m,tot;
40  int f[N];
41  vector<pair<int,int> >son[N];
42  void getroot(int x,int fa){
43        size[x]=1;maxn[x]=0;
44        for(int i=head[x];i;i=ed[i].next){
45            int v=ed[i].v;
46            if(v==fa||vis[v])continue;
47            getroot(v,x);
48            size[x]+=size[v];
49            maxn[x]=max(maxn[x],size[v]);
50        }
51        maxn[x]=max(maxn[x],sum-size[x]);
52        if(maxn[x]<maxn[root])root=x;
53  }
54  void init(int x){
55        vis[x]=1;int all=sum;
56        for(int i=head[x];i;i=ed[i].next){
57            int v=ed[i].v;
58            if(vis[v])continue;
59            root=0;sum=size[v]<size[x]?size[v]:all-size[x];
60            getroot(v,0);
61            son[x].push_back(make_pair(v,root));
62            f[root]=x;
63            init(root);
64        }
65  }
66  int sum1[N],sum2[N],num[N];
67  void update(int x,int y){
68        tot+=y;
69        int now=x;
70        while(now){
71            num[now]+=y;
72            sum1[now]+=y*dis(x,now);
73            if(f[now])sum2[now]+=y*dis(x,f[now]);
74            now=f[now];
75        }
76  }
77  int query(int x){
78        int ans=sum1[x],now=x;
79        while(1){
80            ans+=(sum1[f[now]]-sum2[now])+(num[f[now]]-num[now])*dis(x,f[now]);
81            now=f[now];if(!f[now])break;
82        }
83        return ans;
84  }
85  int query(){
86        int now=rt,last,val=sum1[rt];
87        while(1){
88            last=now;
89            for(int i=0;i<son[now].size();i++){
90                int v=son[now][i].first,nxt=son[now][i].second;
91                int w=query(v);
92                if(w<val){now=nxt;val=query(now);break;}
93            }
94            if(now==last)return val;
95        }
96  }
97  void rmq(){
98        for(int i=1,j=1,tim=0;(i>>1)<=cnt;i<<=1,tim++)
99            while(j<=cnt&&j<(i<<1))lg[j++]=tim;
100       for(int i=1;i<=cnt;i++)minn[i][0]=a[i];
101       for(int i=1;(1<<i)<=cnt;i++)
102           for(int j=1;j+(1<<i)-1<=cnt;j++)
```

```
103              minn[j][i]=min(minn[j][i-1],minn[j+(1<<i-1)][i-1]);
104  }
105  signed main(){
106      scanf("%lld%lld",&n,&m);
107      for(int i=1,u,v,w;i<n;i++){
108          scanf("%lld%lld%lld",&u,&v,&w);
109          add(u,v,w);add(v,u,w);
110      }
111      dfs(1,1);rmq();
112      root=0;maxn[0]=N;sum=n;
113      getroot(1,0);
114      rt=root;
115      init(root);
116      int x,y;
117      while(m--){
118          scanf("%lld%lld",&x,&y);
119          update(x,y);
120          printf("%lld\n",query());
121      }
122      return 0;
123  }
```

2.3   LCT 动态树

```
1   bool ir(int x){return c[F[x]][0]!=x&&c[F[x]][1]!=x;}
2   void rev(int x){swap(c[x][0],c[x][1]);rv[x]^=1;}
3   void pd(int x){if(rv[x])rev(c[x][0]),rev(c[x][1]),rv[x]=0;}
4   void R(int x){
5       int y=F[x],k=c[y][0]==x;F[c[y][!k]=c[x][k]]=y;
6       F[x]=F[y];if(!ir(y))c[F[y]][c[F[y]][1]==y]=x;
7       F[c[x][k]=y]=x;ps(y);
8   }void dw(int x){if(!ir(x))dw(F[x]);pd(x);}
9   void sy(int x){for(dw(x);!ir(x);R(x))if(!ir(F[x]))
10  R(c[F[x]][0]==x^c[F[F[x]]][0]==F[x]?x:F[x]);ps(x);}
11  int fd(int x){for(acs(x),sy(x);c[x][0];x=c[x][0]);return x;}
12  void acs(int x){for(o=0;x;c[x][1]=o,ps(x),o=x,x=F[x])sy(x);}
13  void mrt(int x){acs(x);sy(x);rev(x);}
14  // 无向图 lk, ct, qu
15  void lk(int x,int y){mrt(x);F[x]=y;}
16  void ct(int x,int y){mrt(x);acs(y);sy(y);c[y][0]=F[x]=0;}
17  int qu(int x,int y){mrt(x);acs(y);sy(y);return V[y];}
18  // 有向图 lk, ct, qu
19  void lk(int x,int y){acs(x);sy(x);F[x]=y;}
20  void ct(int x){acs(x);sy(x);F[c[x][0]]=0;c[x][0]=0;ps(x);}
21  int qu(int x){acs(x);sy(x);return V[x];}
22  void add(int x,int y,int z){mrt(x);acs(y);sy(y);ADD(y,z);}
```

2.4   LCT(yzh)

```
1   struct Node {
2       Node *fa, *top, *c[2], *mi;
3       int val; bool rev, isrt;
4       Node () : mi(this), val(INF) {}
5       void setc(Node *ch, int p) {c[p] = ch; ch->fa = this;}
6       bool d() {return fa->c[1] == this;}
7       void upd() {
8           mi = this;
9           if (c[0]->mi->val < mi->val) mi = c[0]->mi;
10          if (c[1]->mi->val < mi->val) mi = c[1]->mi;
11      }
12      void revIt() {
13          rev ^= 1; swap(c[0], c[1]);
14      }
15      void relax() {
16          if (rev) {
17              c[0]->revIt(); c[1]->revIt(); rev = 0;
18          }
19      }
20      void setrt(Node *f);
21  } Tnull, *null = &Tnull, mem[MAXN + MAXM], *ptr = mem;
22
23  void Node::setrt(Node *f) {top = f; fa = null; isrt = 1;}
24
25  Node *newNode(int v = INF) {
26      ptr->fa = ptr->top = ptr->c[0] = ptr->c[1] = null;
```

```
27    ptr->mi = ptr; ptr->rev = 0; ptr->isrt = 1; ptr->val = v;
28    return ptr++;
29 }
30
31 void rot(Node *x) {
32    Node *fa = x->fa; bool d = x->d();
33    fa->fa->setc(x, fa->d());
34    fa->setc(x->c[!d], d);
35    x->setc(fa, !d);
36    fa->upd();
37    if (fa->isrt) {
38       fa->isrt = 0; x->isrt = 1; x->top = fa->top;
39    }
40 }
41
42 void pushto(Node *x) {
43    static Node *st[MAXN]; int top = 0;
44    while (x != null) st[top++] = x, x = x->fa;
45    while (top--) st[top]->relax();
46 }
47
48
49 void splay(Node *x) {
50    pushto(x);
51    while (x->fa != null) {
52       if (x->fa->fa == null) rot(x);
53       else x->d() == x->fa->d() ? (rot(x->fa), rot(x)) : (rot(x), rot(x));
54    }
55    x->upd();
56 }
57
58 void expose(Node *x) {
59    static Node *y;
60    for (y = null; x != null; y = x, x = x->top) {
61       splay(x);
62       x->c[1]->setrt(x); x->setc(y, 1);
63    }
64    //return y;
65 }
66
67 void mkrt(Node *x) {
68    expose(x); splay(x); x->revIt();
69 }
70
71 void Link(Node *u, Node *v) {
72    mkrt(u); u->top = v;
73 }
74
75 void Cut(Node *u, Node *v) {
76    mkrt(u); expose(v); splay(u);
77    u->setc(null, 1); u->upd();
78    v->setrt(null);
79 }
80
81 Node *query(Node *u, Node *v) {
82    mkrt(u); expose(v); splay(v);
83    return v->mi;
84 }
85
86 Node *getrt(Node *x) {
87    expose(x); splay(x);
88    while (x->c[0] != null) x = x->c[0];
89    splay(x);
90    return x;
91 }
92
93 struct Edge {
94    int u, v, w;
95    bool operator < (const Edge &a) const {
96       return w < a.w;
97    }
98 } e[MAXM];
99
```

```
100  int n, m, ans = INF;
101
102  Node *a[MAXN];
103  bool vis[MAXM];
104  int E[MAXN + MAXM];
105
106  void newE(int i) {
107      Node *mid = newNode(e[i].w);
108      Link(mid, a[e[i].u]); Link(mid, a[e[i].v]);
109      E[mid - mem] = i;
110      vis[i] = 1;
111  }
112
113  void delE(Node *mid) {
114      int i = E[mid - mem];
115      vis[i] = 0;
116      int u = e[i].u, v = e[i].v;
117      Cut(mid, a[u]); Cut(mid, a[v]);
118  }
119
120  int main() {
121      read(n); read(m);
122      for (int i = 1; i <= n; i++) a[i] = newNode();
123      for (int i = 1; i <= m; i++) read(e[i].u), read(e[i].v), read(e[i].w);
124      sort(e + 1, e + m + 1);
125      for (int i = 1, cnt = 0, j = 1; i <= m; i++) {
126          if (getrt(a[e[i].u]) != getrt(a[e[i].v])) {
127              cnt++;
128              newE(i);
129          }
130          else if (e[i].u != e[i].v){
131              Node *mi = query(a[e[i].u], a[e[i].v]);
132              delE(mi);
133              newE(i);
134          }
135          while (!vis[j]) j++;
136          if (cnt == n - 1) ans = min(ans, e[i].w - e[j].w);
137      }
138      printf("%d\n", ans == INF ? -1 : ans);
139      return 0;
140  }
```

2.5   可持久化平衡树

```
1   int Copy(int x){// 可持久化
2       id++;sz[id]=sz[x];L[id]=L[x];R[id]=R[x];
3       v[id]=v[x];return id;
4   }int merge(int x,int y){
5       // 合并 x 和 y 两颗子树，可持久化到 z 中
6       if(!x||!y)return x+y;int z;
7       int o=rand()%(sz[x]+sz[y]);// 注意 rand 上限
8       if(o<sz[x])z=Copy(y),L[z]=merge(x,L[y]);
9       else z=Copy(x),R[z]=merge(R[x],y);
10      ps(z);return z;
11  }void split(int x,int&y,int&z,int k){
12      // 将 x 分成 y 和 z 两颗子树，y 的大小为 k
13      y=z=0;if(!x)return;
14      if(sz[L[x]]>=k)z=Copy(x),split(L[x],y,L[z],k),ps(z);
15      else y=Copy(x),split(R[x],R[y],z,k-sz[L[x]]-1),ps(y);}
```

2.6   动态斜率优化

```
1   const LL is_query = - (1LL << 62);
2   struct Line { LL k, b; // kx + b
3   mutable function<const Line*()> succ;
4   bool operator < (const Line &rhs) const {
5       if (rhs.b != is_query) return k < rhs.k;
6       const Line* s = succ();
7       if (!s) return 0; LL x = rhs.k;
8       // 根据范围和有无 __int128 调整 是否使用 1.0L* 下同
9       return 1.0L * b - s->b < 1.0L * (s->k - k) * x; } };
10  struct HullDynamic : public multiset<Line> {
11  bool bad(iterator y) {     // upper hull for maximum
12      auto z = next(y); if (y == begin()) {
13          if (z == end()) return 0;
```

```
14        return y->k == z->k && y->b <= z->b; }
15     auto x = prev(y);
16     if (z == end()) return y->k == x->k && y->b <= x->b;
17     return 1.0L * (x->b - y->b) * (z->k - y->k) >= 1.0L * (y->b - z->b) * (y->k - x->k); }
18  void insert_line(LL k, LL b) {
19     auto y = insert({k, b});
20     y->succ = [=]{return next(y) == end() ? 0 : &*next(y);};
21     if (bad(y)) { erase(y); return; }
22     while (next(y) != end() && bad(next(y))) erase(next(y));
23     while (y != begin() && bad(prev(y))) erase(prev(y)); }
24  LL eval(LL x) { Line u = {x, is_query};
25     auto l = *lower_bound(u); return l.k * x + l.b; }};
```

2.7 treap

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   #define N 100500
4   #define tp pair<Treap *, Treap *>
5   int n, m, tot;
6   struct Treap{
7       Treap * ch[2];
8       int key, size, val, sum, rev;
9       void pushup(){
10          size = ch[0] -> size + ch[1] -> size + 1;
11          sum = ch[0] -> sum + ch[1] -> sum + val;
12      }
13  }*root[N], *null, mem[N << 5];
14  Treap * newTreap(int v){
15      Treap *rt = mem + (++tot);
16      rt -> ch[0] = rt -> ch[1] = null;
17      rt -> size = 1; rt -> val = rt -> sum = v;
18      rt -> key = rand(); rt -> rev = 0;
19      return rt;
20  }
21  Treap * copy(Treap *x){
22      Treap *rt = mem + (++tot);
23      *rt = *x; return rt;
24  }
25  void pushdown(Treap *x){
26      Treap * nxt;
27      if(x -> rev){
28          if(x -> ch[0] != null){
29              nxt = copy(x -> ch[0]);
30              nxt -> rev ^= 1;
31              x -> ch[0] = nxt;
32          }
33          if(x -> ch[1] != null){
34              nxt = copy(x -> ch[1]);
35              nxt -> rev ^= 1;
36              x -> ch[1] = nxt;
37          }
38          swap(x -> ch[0], x -> ch[1]);
39          x -> rev = 0;
40      }
41  }
42  Treap *merge(Treap *a, Treap *b){
43      if(a == null) return b;
44      if(b == null) return a;
45      Treap *nxt;
46      if(a -> key <= b -> key){
47          nxt = copy(a);
48          pushdown(nxt);
49          nxt -> ch[1] = merge(nxt -> ch[1], b);
50      }
51      else{
52          nxt = copy(b);
53          pushdown(nxt);
54          nxt -> ch[0] = merge(a, nxt -> ch[0]);
55      }
56      nxt -> pushup(); return nxt;
57  }
58  tp split(Treap *o, int k){
59      if(o == null) return tp(null, null);
```

```
60      Treap * rt = copy(o);
61      pushdown(rt);
62      tp x;
63      if(rt -> ch[0] -> size >= k){
64          x = split(rt -> ch[0], k);
65          rt -> ch[0] = x.second;
66          rt -> pushup(); x.second = rt;
67      }
68      else{
69          x = split(rt -> ch[1], k - rt -> ch[0] -> size - 1);
70          rt -> ch[1] = x.first;
71          rt -> pushup(); x.first = rt;
72      }
73      return x;
74  }
75  void reverse(Treap *rt, int t, int l, int r){
76      tp x = split(rt, l - 1);
77      tp y = split(x.second, r - l + 1);
78      Treap * nxt = copy(y.first);
79      nxt -> rev ^= 1;
80      root[t] = merge(x.first, merge(nxt, y.second));
81      return ;
82  }
83  void query(Treap *rt, int l, int r){
84      tp x = split(rt, l - 1);
85      tp y = split(x.second, r - l + 1);
86      printf("%d\n", y.first -> sum);
87      rt = merge(x.first, merge(y.first, y.second));
88      return ;
89  }
90  void travel(Treap *x){
91      if(x == null) return ;
92      pushdown(x);
93      travel(x -> ch[0]);
94      printf("%d ", x -> val);
95      travel(x -> ch[1]);
96      return;
97  }
98  int main(){
99      scanf("%d", &n);
100
101     null = mem;
102     null -> ch[0] = null -> ch[1] = null;
103     null -> size = null -> key = null -> val = null -> sum = 0;
104     root[0] = null;
105
106     for(int i = 1, x; i <= n; i++){
107         scanf("%d", &x);
108         root[0] = merge(root[0], newTreap(x));
109     }
110
111     scanf("%d", &m);
112     for(int i = 1, o, l, r; i <= m; i++){
113         root[i] = null;
114         scanf("%d", &o);
115         if(o == 1){
116             scanf("%d%d", &l, &r);
117             reverse(root[i - 1], i, l, r);
118         }
119         else if(o == 2){
120             root[i] = root[i - 1];
121             scanf("%d%d", &l, &r);
122             query(root[i], l, r);
123         }
124         else{
125             scanf("%d", &l);
126             root[i] = root[l];
127         }
128         //travel(root[i]);
129         //puts("");
130     }
131     return 0;
132 }
```

3. Tree & Graph

3.1 边双

```
1  /*
2   * Edge Bi-connected Component
3   * Check: hihoCoder 1184
4   */
5
6  int n, m;
7  int head[MAXN], nxt[MAXM << 1], to[MAXM << 1], ed;
8  // Opposite edge exists, set head[] to -1.
9
10 int dfn[MAXN], low[MAXN], bcc_id[MAXN], bcc_cnt, stamp;
11 bool isbridge[MAXM << 1], vis[MAXN];
12
13 std::vector<int> bcc[MAXN];
14
15 void Tarjan(int now, int fa) {
16     dfn[now] = low[now] = ++stamp;
17     for (int i = head[now]; ~i; i = nxt[i]) {
18         if (!dfn[to[i]]) {
19             Tarjan(to[i], now);
20             low[now] = std::min(low[now], low[to[i]]);
21             if (low[to[i]] > dfn[now])
22                 isbridge[i] = isbridge[i ^ 1] = 1;
23         }
24         else if (dfn[to[i]] < dfn[now] && to[i] != fa)
25             low[now] = std::min(low[now], dfn[to[i]]);
26     }
27 }
28
29 void DFS(int now) {
30     vis[now] = 1;
31     bcc_id[now] = bcc_cnt;
32     bcc[bcc_cnt].push_back(now);
33     for (int i = head[now]; ~i; i = nxt[i]) {
34         if (isbridge[i]) continue;
35         if (!vis[to[i]]) DFS(to[i]);
36     }
37 }
38
39 void EBCC() {
40     memset(dfn, 0, sizeof dfn);
41     memset(low, 0, sizeof low);
42     memset(isbridge, 0, sizeof isbridge);
43     memset(bcc_id, 0, sizeof bcc_id);
44     bcc_cnt = stamp = 0;
45
46     for (int i = 1; i <= n; ++i)
47         if (!dfn[i]) Tarjan(i, 0);
48
49     memset(vis, 0, sizeof vis);
50     for (int i = 1; i <= n; ++i)
51         if (!vis[i]) {
52             ++bcc_cnt;
53             DFS(i);
54         }
55 }
```

3.2 点双

```
1  /*
2   * Point Bi-connected Component
3   * Check: VALLA 5135
4   */
5
6  typedef std::pair<int, int> pii;
7  #define mkpair std::make_pair
8
9  int n, m;
10 std::vector<int> G[MAXN];
11
12 int dfn[MAXN], low[MAXN], bcc_id[MAXN], bcc_cnt, stamp;
13 bool iscut[MAXN];
```

```
14
15  std::vector<int> bcc[MAXN]; // Unnecessary
16
17  pii stk[MAXN]; int stk_top;
18  // Use a handwritten structure to get higher efficiency
19
20  void Tarjan(int now, int fa) {
21      int child = 0;
22      dfn[now] = low[now] = ++stamp;
23      for (int to: G[now]) {
24          if (!dfn[to]) {
25              stk[++stk_top] = mkpair(now, to); ++child;
26              Tarjan(to, now);
27              low[now] = std::min(low[now], low[to]);
28              if (low[to] >= dfn[now]) {
29                  iscut[now] = 1;
30                  bcc[++bcc_cnt].clear();
31                  while (1) {
32                      pii tmp = stk[stk_top--];
33                      if (bcc_id[tmp.first] != bcc_cnt) {
34                          bcc[bcc_cnt].push_back(tmp.first);
35                          bcc_id[tmp.first] = bcc_cnt;
36                      }
37                      if (bcc_id[tmp.second] != bcc_cnt) {
38                          bcc[bcc_cnt].push_back(tmp.second);
39                          bcc_id[tmp.second] = bcc_cnt;
40                      }
41                      if (tmp.first == now && tmp.second == to)
42                          break;
43                  }
44              }
45          }
46          else if (dfn[to] < dfn[now] && to != fa) {
47              stk[++stk_top] = mkpair(now, to);
48              low[now] = std::min(low[now], dfn[to]);
49          }
50      }
51      if (!fa && child == 1)
52          iscut[now] = 0;
53  }
54
55  void PBCC() {
56      memset(dfn, 0, sizeof dfn);
57      memset(low, 0, sizeof low);
58      memset(iscut, 0, sizeof iscut);
59      memset(bcc_id, 0, sizeof bcc_id);
60      stamp = bcc_cnt = stk_top = 0;
61
62      for (int i = 1; i <= n; ++i)
63          if (!dfn[i]) Tarjan(i, 0);
64  }
```

3.3   2-sat(含强联通分量)

```
1  //清点清边要两倍
2  int stamp, comps, top;
3  int dfn[N], low[N], comp[N], stack[N];
4  void add(int x, int a, int y, int b) {//取$X_a$则必须取$Y_b$,则$X_a$向$Y_b$连边,注意连边是对称的,即,此时实际上$X_b$也必须向$Y_a$连边.
5      edge[x << 1 | a].push_back(y << 1 | b);
6  }
7  void tarjan(int x) {
8      dfn[x] = low[x] = ++stamp;
9      stack[top++] = x;
10     for (int i = 0; i < (int)edge[x].size(); ++i) {
11         int y = edge[x][i];
12         if (!dfn[y]) {
13             tarjan(y);
14             low[x] = std::min(low[x], low[y]);
15         } else if (!comp[y]) {
16             low[x] = std::min(low[x], dfn[y]);
17         }
18     }
19     if (low[x] == dfn[x]) {
20         comps++;
```

```
21          do {
22              int y = stack[--top];
23              comp[y] = comps;
24          } while (stack[top] != x);
25      }
26  }
27  bool answer[N];
28  bool solve() {
29      int counter = n + n + 1;
30      stamp = top = comps = 0;
31      std::fill(dfn, dfn + counter, 0);
32      std::fill(comp, comp + counter, 0);
33      for (int i = 0; i < counter; ++i) {
34          if (!dfn[i]) {
35              tarjan(i);
36          }
37      }
38      for (int i = 0; i < n; ++i) {
39          if (comp[i << 1] == comp[i << 1 | 1]) {
40              return false;
41          }
42          answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
43      }
44      return true;
45  }
```

3.4   Stoer-Wagner 无向图最小割(树)

```
1   int d[N];bool v[N],g[N];
2   int get(int&s,int&t){
3       CL(d);CL(v);int i,j,k,an,mx;
4       fr(i,1,n){  k=mx=-1;
5          fr(j,1,n)if(!g[j]&&!v[j]&&d[j]>mx)k=j,mx=d[j];
6          if(k==-1)return an;
7          s=t;t=k;an=mx;v[k]=1;
8          fr(j,1,n)if(!g[j]&&!v[j])d[j]+=w[k][j];
9       }return an;}
10  int mincut(int n,int w[N][N]){
11  //n 为点数，w[i][j] 为 i 到 j 的流量，返回无向图所有点对最小割之和
12      int ans=0,i,j,s,t,x,y,z;
13      fr(i,1,n-1){
14          ans=min(ans,get(s,t));
15          g[t]=1;if(!ans)break;
16          fr(j,1,n)if(!g[j])w[s][j]=(w[j][s]+=w[j][t]);
17      }return ans;}
18  // 无向图最小割树
19  void fz(int l,int r){// 左闭右闭，分治建图
20      if(l==r)return;S=a[l];T=a[r];
21      reset();// 将所有边权复原
22      flow(S,T);// 做网络流
23      dfs(S);// 找割集，v[x]=1 属于 S 集，否则属于 T 集
24      ADD(S,T,fl);// 在最小割树中建边
25      L=l,R=r;fr(i,l,r) if(v[a[i]])q[L++]=a[i]; else q[R--]=a[i];
26      fr(i,l,r)a[i]=q[i];fz(l,L-1);fz(R+1,r);}
```

3.5   KM 最大权匹配

```
1   LL KM(int n,LL w[N][N]){ // n 为点，w 为边权  不存在的边权开 -n*(|maxv|+1),inf 设为 3n*(|maxv|+1)
2       static LL lx[N],ly[N],slk[N];
3       static int lk[N],pre[N];static bool vy[N];
4       LL ans=0;int x,py,d,p;
5       fr(i,1,n)fr(j,1,n)lx[i]=max(lx[i],w[i][j]);
6       fr(i,1,n){
7          fr(j,1,n)slk[j]=inf,vy[j]=0;
8          for(lk[py=0]=i;lk[py];py=p){
9              vy[py]=1;d=inf;x=lk[py];
10             fr(y,1,n)if(!vy[y]){
11                 if(lx[x]+ly[y]-w[x][y]<slk[y])
12                     slk[y]=lx[x]+ly[y]-w[x][y],pre[y]=py;
13                 if(slk[y]<d)d=slk[y],p=y;
14             }fr(y,0,n) if(vy[y])lx[lk[y]]-=d,ly[y]+=d; else slk[y]-=d;
15          }for(;py;py=pre[py])lk[py]=lk[pre[py]];}
16      fr(i,1,n)ans+=lx[i]+ly[i];
17      return ans;}//lk[] 为与右部点连的左部点
```

3.6   spfa 费用流

```
bool spfa(){
    memset(dis,0x3f,sizeof dis);
    memset(vis,0,sizeof vis);
    queue<int> q;
    q.push(S);dis[S]=0;
    while(!q.empty()){
        int x=q.front();q.pop();vis[x]=0;
        for(int i=head[x];i;i=ed[i].next){
            int v=ed[i].v;
            if(ed[i].f&&dis[v]>dis[x]+ed[i].w){
                dis[v]=dis[x]+ed[i].w;
                if(!vis[v]){vis[v]=1;q.push(v);}
            }
        }
    }
    return dis[T]!=dis[0];
}
int dfs(int x,int f){
    vis[x]=1;
    if(x==T){
        Flow+=f;
        Cost+=f*dis[T];
        return f;
    }
    int ans=0;
    for(int i=head[x];i;i=ed[i].next){
        int v=ed[i].v;
        if(ed[i].f&&dis[v]==dis[x]+ed[i].w&&!vis[v]){
            int nxt=dfs(v,min(f,ed[i].f));
            ans+=nxt;f-=nxt;ed[i].f-=nxt;ed[i^1].f+=nxt;
        }
        if(!f)break;
    }
    return ans;
}
void work(){
    while(spfa()){
        do{
            memset(vis,0,sizeof vis);
            dfs(S,inf);
        }while(vis[T]==1);
    }
    if(Flow==allflow)printf("%d\n",Cost/3);
    else puts("-1");
}
```

3.7   Hopcroft-Karp $O(\sqrt{N}M)$ 最大匹配

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

3.8   Blossom 带花树

```
struct blossom{//mat[i] 为 i 号点的匹配点
int n,m,h,t,W,tot,fir[N],la[M],ne[M],F[N];
```

```
3  int mat[N],pre[N],tp[N],q[N],vs[N];
4  void in(int x,int y){n=x;m=y;W=tot=0;CL(fir);}
5  void ins(x,y){}// 初始化 n 个点 m 条边的图
6  int fd(int x){return F[x]?F[x]=fd(F[x]):x;}
7  int lca(int u,int v){
8     for(++W;;u=pre[mat[u]],swap(u,v))
9        if(vs[u=fd(u)]==W)return u;else vs[u]=u?W:0;
10 }void aug(int u,int v){for (int w;u;v=pre[u=w]) w=mat[v],mat[mat[u]=v]=u;}
11 void blo(int u,int v,int f){
12    for(int w;fd(u)^f;u=pre[v=w])
13       pre[u]=v, F[u]?0:F[u]=f, F[w=mat[u]]?0:F[w]=f, tp[w]^1?0:tp[q[++t]=w]=-1;
14 }int bfs(int u){
15    int x,E,i;CL(tp);CL(F);
16    for(--tp[q[h=t=1]=u];h<=t;u=q[++h])
17    for(i=fir[u];i;i=ne[i])if(!tp[E=la[i]]){
18       if(!mat[E])return aug(E,u),1;
19       pre[E]=u,++tp[E],--tp[q[++t]=mat[E]];
20    }else if(tp[E]^1&&fd(u)^fd(E)) blo(u,E,x=lca(u,E)),blo(E,u,x);
21    return 0;}
22 int solve(){// 返回答案
23    int i,an=0;fr(i,1,n)mat[i]?0:an+=bfs(i);
24    return an;}
25 }G;
```

3.9   Cactus 仙人掌

```
1  // 求边仙人掌长度为len的简单路径条数ans[len]
2  const int N = 5005, MOD = 1e9 + 7; vector<int> G[N];
3  int f[N][N], dfn[N], tot, low[N], n, sz[N];
4  vector<int> getcycle(int x, int root) {
5     vector<int> cyc; // 获得root->x为第一条边的环
6     cyc.push_back(root); cyc.push_back(x);
7     while (true) { int nex = -1;
8        for (int u : G[cyc.back()]) {
9           if (low[u] != dfn[root] || cyc[cyc.size() - 2] == u || u == root) continue;
10          nex = u; break; }
11       if (nex == -1) break; cyc.push_back(nex); }
12    return cyc; }
13 int walk(int x, int root) { // 沿着root->x dp一圈 获得从root出发往子树走所有长度为len的简单路径条数f[root][len]
14    vector<int> cycle = getcycle(x, root);
15    int szm = 0; // 合并路径时只for到最长长度，保证复杂度
16    for (int i = 1; i < (int) cycle.size(); ++ i) {
17       int u = cycle[i], l1 = i, l2 = (int) cycle.size() - i;
18       for (int k = 0; k < sz[u]; ++ k) {
19          if (k + l1 < n) (f[root][k + l1] += f[u][k]) %= MOD;
20          if (k + l2 < n) (f[root][k + l2] += f[u][k]) %= MOD;
21          szm = max(szm, k + l1); szm = max(szm, k + l2); } }
22    return szm; }
23 int g[N], msz, ans[N], par[N];
24 void walk2(int x, int root) { // 统计最浅点在这个环上的路径
25    vector<int> cycle = getcycle(x, root);
26    for (int i = 1; i < (int) cycle.size(); ++ i) {
27       int u = cycle[i]; // 出发点为u往下走
28       int l1 = i, l2 = (int) cycle.size() - i;
29       for (int k = 0; k < sz[u]; ++ k) {
30          for (int l = 0; l < msz; ++ l) {
31             //g[l]为root之前子树伸出去长l路径条数 双向*2
32             if (l + k + l1 < n)
33                (ans[l+k+l1]+=2LL*g[l]*f[u][k]%MOD)%=MOD;
34             if (l + k + l2 < n)
35                (ans[l+k+l2]+=2LL*g[l]*f[u][k]%MOD)%=MOD;}}}
36    for (int i = 1; i < (int) cycle.size(); ++ i) {
37       int u = cycle[i], l1 = i, l2 = (int) cycle.size() - i;
38       for (int k = 0; k < sz[u]; ++ k) { // 更新g[l]
39          if (k + l1 < n) (g[k + l1] += f[u][k]) %= MOD;
40          if (k + l2 < n) (g[k + l2] += f[u][k]) %= MOD;
41          msz = max(msz, k+l1+1); msz = max(msz, k+l2+1); } }
42    for (int i = 1; i < (int) cycle.size(); ++ i) {
43       for (int j = 1; j < (int) cycle.size(); ++ j) {
44          if (i == j) continue;
45          // 环上u和v往下走出两条路径的合并
46          int u = cycle[i], v = cycle[j];
47          int l1 = abs(i-j), l2 = (int)cycle.size()-l1;
48          for (int k = 0; k < sz[u]; ++ k) {
```

```
49          for (int l = 0; l < sz[v]; ++ l) {
50              if (l+k+l1<n) (ans[l+k+l1] += 1LL * f[v][l]*f[u][k] % MOD) %= MOD;
51              if (l+k+l2<n) (ans[l+k+l2] += 1LL * f[v][l]*f[u][k] % MOD) %= MOD;}}}}
52    msz = min(msz, n); }
53 void dfs(int x, int p = 0) {
54    low[x] = dfn[x] = ++ tot; par[x] = p;
55    for (int u : G[x]) if (u != p) {
56       if (dfn[u]) low[x] = min(low[x], dfn[u]);
57          else { dfs(u, x); low[x] = min(low[x], low[u]); } }
58    f[x][0] = 1; sz[x] = 1;
59    for (int u : G[x]) if (u != p) {
60       if (low[u] == dfn[x] && par[u] != x) {
61          // 用环边f[x]
62          sz[x]=max(sz[x], walk(u,x)+1); sz[x]=min(sz[x],n); }
63       else if (low[u] > dfn[x]) {
64          // 树边更新f[x]
65          sz[x]=max(sz[x], sz[u] + 2); sz[x]=min(sz[x], n);
66          for (int k = 0; k < sz[u]; ++ k)
67             (f[x][k + 1] += f[u][k]) %= MOD; } }
68    // 子树背包
69    for (int i = 0; i < n; ++ i) g[i] = 0;
70    g[0] = 1; ans[0] ++; msz = 1;
71    for (int u : G[x]) { if (u == p) continue;
72       if (low[u]==dfn[x] && par[u]!=x) walk2(u, x); // 环边
73       else if (low[u] > dfn[x]) { // 树边
74          for (int k = 0; k < sz[u]; ++ k) {
75             for (int l = 0; l < msz; ++ l) {
76                if (k + l + 1 < n) (ans[k + l + 1] += 2LL * f[u][k] * g[l] % MOD) %= MOD; } }
77          for (int k = 0; k < sz[u]; ++ k)
78             (g[k + 1] += f[u][k]) %= MOD;
79          msz = max(msz, sz[u] + 1); msz = min(msz, n); } } }
```

3.10   Dominator Tree 支配树

```
1 struct Dominator_Tree{
2    //n为点数，s为起点，e[]中记录每条边
3    int n,s,cnt;int dfn[N],id[N],pa[N], semi[N],idom[N],p[N],mn[N];
4    vector<int>e[N],dom[N],be[N];
5    void ins(x,y){e[x].pb(y);}
6    void dfs(int x){//先得到DFS树
7       dfn[x]=++cnt;id[cnt]=x;
8       for(auto i:e[x]){
9          if(!dfn[i])dfs(i),pa[dfn[i]]=dfn[x];
10         be[dfn[i]].push_back(dfn[x]);
11      }}
12    int get(int x){//带权并查集
13       if(p[x]!=p[p[x]]){
14          if(semi[mn[x]]>semi[get(p[x])]) mn[x]=get(p[x]);
15          p[x]=p[p[x]];
16       }return mn[x];
17    }void LT(){//求出semi和idom得到支配树
18       for(int i=cnt;i>1;i--){
19          for(auto j:be[i]) semi[i]=min(semi[i],semi[get(j)]);
20          dom[semi[i]].push_back(i); int x=p[i]=pa[i];
21          for(auto j:dom[x]) idom[j]=(semi[get(j)]<x?get(j):x);
22          dom[x].clear();
23       }fr(i,2,cnt){
24          if(idom[i]!=semi[i])idom[i]=idom[idom[i]];
25          dom[id[idom[i]]].push_back(id[i]);
26       }
27    }void build(){//建立支配树
28       fr(i,1,n)dfn[i]=0,dom[i].clear(), be[i].clear(),p[i]=mn[i]=semi[i]=i;
29       cnt=0;dfs(s);LT();
30    }
31 }G;
```

3.11   朱刘算法 最小树形图 (含lazy tag可并堆)

```
1 using Val = long long;
2 #define nil mem
3 struct Node { Node *l,*r; int dist;int x,y;Val val,laz; }
4 mem[M] = {{nil, nil, -1}}; int sz = 0;
5 #define NEW(arg...) (new(mem + ++ sz)Node{nil,nil,0,arg})
6 void add(Node *x, Val o) {if(x!=nil){x->val+=o, x->laz+=o;}}
7 void down(Node *x){add(x->l,x->laz);add(x->r,x->laz);x->laz=0;}
```

```
8   Node *merge(Node *x, Node *y) {
9       if (x == nil) return y; if (y == nil) return x;
10      if (y->val < x->val) swap(x, y); //smalltop heap
11      down(x); x->r = merge(x->r, y);
12      if (x->l->dist < x->r->dist) swap(x->l, x->r);
13      x->dist = x->r->dist + 1; return x; }
14  Node *pop(Node *x){down(x); return merge(x->l, x->r);}
15  struct DSU { int f[N]; void clear(int n) {
16          for (int i=0; i<=n; ++i) f[i]=i; }
17      int fd(int x) { if (f[x]==x) return x;
18          return f[x]=fd(f[x]); }
19      int& operator[](int x) {return f[fd(x)];}};
20  DSU W, S; Node *H[N], *pe[N];
21  vector<pair<int, int>> G[N]; int dist[N], pa[N];
22  // addedge(x, y, w) : NEW(x, y, w, 0)
23  Val chuliu(int s, int n) { // O(ElogE)
24      for (int i = 1; i <= n; ++ i) G[i].clear();
25      Val re=0; W.clear(n); S.clear(n); int rid=0;
26      fill(H, H + n + 1, (Node*) nil);
27      for (auto i = mem + 1; i <= mem + sz; ++ i)
28          H[i->y] = merge(i, H[i->y]);
29      for (int i = 1; i <= n; ++ i) if (i != s)
30          for (;;) {
31              auto in = H[S[i]]; H[S[i]] = pop(H[S[i]]);
32              if (in == nil) return INF; // no solution
33              if (S[in -> x] == S[i]) continue;
34              re += in->val; pe[S[i]] = in;
35              // if (in->x == s) true root = in->y
36              add(H[S[i]], -in->val);
37              if (W[in->x]!=W[i]) {W[in->x]=W[i];break;}
38              G[in -> x].push_back({in->y,++rid});
39              for (int j=S[in->x]; j!=S[i]; j=S[pe[j]->x]) {
40                  G[pe[j]->x].push_back({pe[j]->y, rid});
41                  H[j] = merge(H[S[i]], H[j]); S[i]=S[j]; }}
42      ++ rid; for (int i=1; i<=n; ++ i) if(i!=s && S[i]==i)
43          G[pe[i]->x].push_back({pe[i]->y, rid}); return re;}
44  void makeSol(int s, int n) {
45      fill(dist, dist + n + 1, n + 1); pa[s] = 0;
46      for (multiset<pair<int, int>> h = {{0,s}}; !h.empty();){
47          int x=h.begin()->second;
48          h.erase(h.begin()); dist[x]=0;
49          for (auto i : G[x]) if (i.second < dist[i.first]) {
50              h.erase({dist[i.first], i.first});
51              h.insert({dist[i.first] = i.second, i.first});
52              pa[i.first] = x; }}}
```

3.12   最大团

```
1   // DN超级快最大团，建议 n <= 150
2   typedef bool BB[N]; struct Maxclique {
3   const BB *e; int pk, level; const float Tlimit;
4   struct Vertex { int i, d; Vertex(int i) : i(i), d(0) {}};
5   typedef vector<Vertex> Vertices; Vertices V;
6   typedef vector<int> ColorClass; ColorClass QMAX, Q;
7   vector<ColorClass> C;
8   static bool desc_degree(const Vertex &vi,const Vertex &vj)
9   { return vi.d > vj.d; }
10  void init_colors(Vertices &v) {
11      const int max_degree = v[0].d;
12      for (int i = 0; i < (int)v.size(); i++)
13          v[i].d = min(i, max_degree) + 1; }
14  void set_degrees(Vertices &v) {
15      for (int i = 0, j; i < (int)v.size(); i++)
16          for (v[i].d = j = 0; j < (int)v.size(); j++)
17              v[i].d += e[v[i].i][v[j].i]; }
18  struct StepCount{ int i1, i2; StepCount(): i1(0),i2(0){}};
19  vector<StepCount> S;
20  bool cut1(const int pi, const ColorClass &A) {
21      for (int i = 0; i < (int)A.size(); i++)
22          if (e[pi][A[i]]) return true; return false; }
23  void cut2(const Vertices &A, Vertices & B) {
24      for (int i = 0; i < (int)A.size() - 1; i++)
25          if (e[A.back().i][A[i].i]) B.push_back(A[i].i); }
26  void color_sort(Vertices & R) { int j=0, maxno=1;
```

```
27    int min_k=max((int)QMAX.size()-(int)Q.size()+1,1);
28    C[1].clear(), C[2].clear();
29    for (int i = 0; i < (int)R.size(); i++) {
30       int pi = R[i].i, k = 1; while (cut1(pi, C[k])) k++;
31       if (k > maxno) maxno = k, C[maxno + 1].clear();
32       C[k].push_back(pi); if (k < min_k) R[j++].i = pi; }
33    if (j > 0) R[j - 1].d = 0;
34    for (int k = min_k; k <= maxno; k++)
35       for (int i = 0; i < (int)C[k].size(); i++)
36          R[j].i = C[k][i], R[j++].d = k; }
37 void expand_dyn(Vertices &R) {
38    S[level].i1 = S[level].i1 + S[level-1].i1 - S[level].i2;
39    S[level].i2 = S[level - 1].i1;
40    while ((int)R.size()) {
41       if ((int)Q.size() + R.back().d > (int)QMAX.size()) {
42          Q.push_back(R.back().i); Vertices Rp; cut2(R, Rp);
43          if ((int)Rp.size()) {
44             if((float)S[level].i1/++pk<Tlimit)degree_sort(Rp);
45             color_sort(Rp); S[level].i1++, level++;
46             expand_dyn(Rp); level--;
47          } else if ((int)Q.size() > (int)QMAX.size()) QMAX=Q;
48          Q.pop_back(); } else return; R.pop_back(); }}
49 void mcqdyn(int *maxclique, int &sz) {
50    set_degrees(V); sort(V.begin(), V.end(), desc_degree);
51    init_colors(V);
52    for (int i=0; i<(int)V.size()+1; i++) S[i].i1=S[i].i2=0;
53    expand_dyn(V); sz = (int)QMAX.size();
54  for(int i=0;i<(int)QMAX.size();i++)maxclique[i]=QMAX[i];}
55 void degree_sort(Vertices & R) {
56    set_degrees(R); sort(R.begin(), R.end(), desc_degree); }
57 Maxclique(const BB *conn,const int sz,const float tt=.025)
58    : pk(0), level(1), Tlimit(tt){
59    for(int i = 0; i < sz; i++) V.push_back(Vertex(i));
60    e = conn, C.resize(sz + 1), S.resize(sz + 1); }};
61 BB e[N]; int ans, sol[N]; for (...) e[x][y]=e[y][x]=true;
62 Maxclique mc(e, n); mc.mcqdyn(sol, ans); // 全部0下标
63 for (int i = 0; i < ans; ++i) cout << sol[i] << endl;
```

3.13 极大团计数

```
1 // 0下标，需删除自环(即确保E_ii = false，补图要特别注意)
2 // 极大团计数，最坏情况O(3^(n/3))
3 ll ans; ull E[64]; #define bit(i) (1ULL << (i))
4 void dfs(ull P, ull X, ull R) { // 不需要方案时可去掉R相关语句
5    if (!P && !X) { ++ans; sol.pb(R); return; }
6    ull Q = P & ~E[__builtin_ctzll(P | X)];
7    for (int i; i = __builtin_ctzll(Q), Q; Q &= ~bit(i)) {
8       dfs(P & E[i], X & E[i], R | bit(i));
9       P &= ~bit(i), X |= bit(i); }}
10 ans = 0; dfs(n == 64 ? ~0ULL : bit(n) - 1, 0, 0);
```

3.14 有根树同构-Reshiram

```
1 const unsigned long long MAGIC = 4423;
2
3 unsigned long long magic[N];
4 std::pair<unsigned long long, int> hash[N];
5
6 void solve(int root) {
7    magic[0] = 1;
8    for (int i = 1; i <= n; ++i) {
9       magic[i] = magic[i - 1] * MAGIC;
10   }
11   std::vector<int> queue;
12   queue.push_back(root);
13   for (int head = 0; head < (int)queue.size(); ++head) {
14      int x = queue[head];
15      for (int i = 0; i < (int)son[x].size(); ++i) {
16         int y = son[x][i];
17         queue.push_back(y);
18      }
19   }
20   for (int index = n - 1; index >= 0; --index) {
21      int x = queue[index];
22      hash[x] = std::make_pair(0, 0);
```

```
23
24          std::vector<std::pair<unsigned long long, int> > value;
25          for (int i = 0; i < (int)son[x].size(); ++i) {
26              int y = son[x][i];
27              value.push_back(hash[y]);
28          }
29          std::sort(value.begin(), value.end());
30
31          hash[x].first = hash[x].first * magic[1] + 37;
32          hash[x].second++;
33          for (int i = 0; i < (int)value.size(); ++i) {
34              hash[x].first = hash[x].first * magic[value[i].second] + value[i].first;
35              hash[x].second += value[i].second;
36          }
37          hash[x].first = hash[x].first * magic[1] + 41;
38          hash[x].second++;
39      }
40 }
```

3.15   Dinic 最大流

```
1  template <int MAXN = 100000, int MAXM = 100000>
2  struct flow_edge_list { int size;
3  int begin[MAXN], dest[MAXM], next[MAXM],flow[MAXM],inv[MAXM];
4  void clear(int n) { size=0; std::fill(begin, begin+n, -1); }
5  flow_edge_list (int n = MAXN) { clear (n); }
6  void add_edge (int u, int v, int f) {
7     dest[size] = v; next[size] = begin[u]; flow[size] = f;
8     inv[size] = size + 1; begin[u] = size++;
9     dest[size] = u; next[size] = begin[v]; flow[size] = 0;
10    inv[size] = size - 1; begin[v] = size++; } };
11 template <int MAXN = 1000, int MAXM = 100000>
12 struct dinic { int n, s, t; int d[MAXN], w[MAXN], q[MAXN];
13 int bfs (flow_edge_list <MAXN, MAXM> &e) {
14    for (int i = 0; i < n; ++i) d[i] = -1; int l, r;
15    q[l = r = 0] = s, d[s] = 0;
16    for (; l <= r; l ++)
17       for (int k = e.begin[q[l]]; k > -1; k = e.next[k])
18          if (d[e.dest[k]] == -1 && e.flow[k] > 0)
19             d[e.dest[k]] = d[q[l]]+1, q[++r] = e.dest[k];
20    return d[t] > -1 ? 1 : 0; }
21 int dfs (flow_edge_list <MAXN, MAXM> &e, int u, int ext) {
22    if (u == t) return ext; int k = w[u], ret = 0;
23    for (; k>-1; k=e.next[k], w[u] = k) { if (ext==0) break;
24       if (d[e.dest[k]] == d[u] + 1 && e.flow[k] > 0) {
25          int flow=dfs(e, e.dest[k], std::min(e.flow[k],ext));
26          if (flow > 0) {
27             e.flow[k] -= flow, e.flow[e.inv[k]] += flow;
28             ret += flow, ext -= flow; } } }
29    if (k == -1) d[u] = -1; return ret; }
30 int solve(flow_edge_list<MAXN, MAXM> &e,int n,int s,int t) {
31    int ans = 0; dinic::n = n; dinic::s = s; dinic::t = t;
32    while (bfs (e)) {
33       for (int i = 0; i < n; ++i) w[i] = e.begin[i];
34       ans += dfs (e, s, INF); } return ans; } };
```

3.16   Dijkstra 费用流

```
1  const LL INF = 1e18; struct Edge { LL f, c; int to, r; };
2  vector<Edge> G[N]; LL d[N]; bool fst = true;
3  int S, T, prv[N], prp[N], cur[N], vst[N];
4  bool SPFA(int S) {
5     if(fst) { fst = 0;
6        // 此处为第一次求最短路，可 Dij 就和下面一样，不可就 SPFA 或根据图性质 DP
7        return d[T] != INF; }
8     fill(d + 1, d + 1 + T, INF); // 此处为 Dij
9     priority_queue<pair<LL,int>> pq; pq.push({0,S}); d[S]=0;
10    while(1) {
11       while(!pq.empty() && -pq.top().first != d[pq.top().second]) pq.pop();
12       if(pq.empty()) break;
13       int v(pq.top().second); pq.pop(); int cnt(0);
14       for (Edge e : G[v]) {
15          if (e.f && d[e.to] > d[v] + e.c) {
16             d[e.to] = d[v] + e.c; prv[e.to] = v;
17             prp[e.to]=cnt; pq.push({-d[e.to], e.to}); }
```

```
18        cnt++; } }
19     return d[T] != INF; }
20 LL aug(int v, LL flow) {
21     if(v == T) return flow; vst[v] = 1; LL flow1(flow);
22     for(int & i(cur[v]); i < (int)G[v].size(); i++) {
23         Edge & e = G[v][i];
24         if(e.f && d[v] + e.c == d[e.to] && !vst[e.to]) {
25             LL flow1(aug(e.to, min(flow, e.f)));
26             flow-=flow1; e.f-=flow1; G[e.to][e.r].f += flow1; }
27         if(flow == 0) { vst[v] = 0; return flow1 - flow; } }
28     return flow1 - flow; }
29 LL mcmf() { LL ans = 0, sT = 0;
30     while (SPFA(S)) { sT += d[T]; // 多路增广
31         for(int i(1); i <= T; i++) cur[i] = 0, vst[i] = 0;
32         ans += sT * aug(S, INF);
33         for(int i(1); i <= T; i++)
34             for(auto & e : G[i]) e.c += d[i] - d[e.to];}
35     return ans; }
36 void add(int u, int v, int f, int c) {
37     G[u].push_back({f, c, v, (int) G[v].size()});
38     G[v].push_back({0, -c, u, (int) G[u].size() - 1}); }
39 int main() { // 初始化 S, T, T 编号最大, 1base
40     // add(x, y, cap, cost)
41     LL ans = mcmf(); }
```

3.17 弦图说明

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 $\{p_1, p_2, \ldots, p_t\}$, 则 $\{p_1 \cup N(p_1), \ldots, p_t \cup N(p_t)\}$ 为最小团覆盖

```
1 //id[i]为点i的标号，seq[i]为标号为i的点，G[]存图
2 int q[N],label[N],id[N],vis[N],seq[N],c[N]; vector<int>G[N];
3 struct P{int lab,u;bool operator<(const P&a) const {return lab<a.lab;}};
4 void mcs(){//MCS算法求标号序列,优先队列做到O(mlgn)
5     int i,j,u,v;CL(id);CL(label);
6     CL(seq);priority_queue<P>Q;
7     fr(i,1,n)Q.push(P{0,i}); //label_i表示第i个点与多少个已标号的点相邻
8     dr(i,n,1){
9         for(;id[Q.top().u];)Q.pop(); //每次选label_i最大的未标号的点标号
10        u=Q.top().u;Q.pop();id[u]=i;
11        for(j=0;j<G[u].size();j++)if(v=G[u][j],!id[v]) label[v]++,Q.push(P{label[v],v});
12    }fr(i,1,n)seq[id[i]]=i;
13 }bool ok(){//O(m)判断是否是弦图
14    int i,j,t,u,v,w;CL(vis);
15    dr(i,n,1){
16        u=seq[i];t=0;//标号从小到大找点
17        for(j=0;j<G[u].size();j++)
18            if(v=G[u][j],id[v]>id[u])q[++t]=v;
19        if(!t)continue;w=q[1];//找标号大于它的点中最小的
20        fr(j,1,t)if(id[q[j]]<id[w])w=q[j];
21        for(j=0;j<G[w].size();j++)vis[G[w][j]]=i;
22        fr(j,1,t)if(q[j]!=w)if(vis[q[j]]!=i)return 0;
23    }return 1;
24 int setcolor(){//弦图最小染色 团数=染色数
25    int an=0,i,j,u,v;CL(vis);CL(c);
26    for(i=n;i;i--){
27        u=seq[i];
28        for(j=0;j<G[u].size();j++)vis[c[G[u][j]]]=i;
29        for(j=1;vis[j]==i;j++);//找最小的没出现的颜色
30        c[u]=j;an=max(an,j);
31    }return an;
32 }mcs();puts(ok()?"YES":"NO");printf("%d\n",setcolor());
```

3.18 欧拉回路

```
1 int x,y,t,tot,fir[N],d1[N],d2[N],q[M],la[M],ne[M],va[M];bool v[M];
2 void ins(int x,int y,int z){/*ADDEDGE*/d1[x]++;d2[y]++;}
3 void dfs(int x){
```

```
4      for(int i=fir[x];i;i=fir[x]){
5          for(;i&&v[abs(va[i])];i=ne[i]);fir[x]=i; // 将已经走过的边删去
6          if(i)v[abs(va[i])]=1,dfs(la[i]),q[++t]=va[i];}// 走第一个未走过的边
7  }void Eular(p,n,m){
8  //p=1 是无向图, p=2 是有向图   n 是点数, m 是边数
9      fr(i,1,m)if(rd(x,y),ins(x,y,i),p==1)ins(y,x,-i);
10     fr(i,1,n)if(p==1&&d1[i]%2||p==2&&d1[i]^d2[i])
11         return puts("NO"),0;// 不合法情况
12     dfs(la[1]); // 从一条边开始找
13     if(t<m)return puts("NO"),0; // 没有欧拉回路
14     for(puts("YES");t;printf("%d ",q[t--]));} // 输出方案, 正数是正向边, 负数是反向边
```

3.19   K短路

```
1  int F[N],FF[N];namespace Left_Tree{//可持久化左偏树
2      struct P{int l,r,h,v,x,y;}Tr[N*40];int RT[N],num;
3      //l和r是左右儿子,h是高度,v是数值,x和y是在图中的两点
4      int New(P o){Tr[++num]=o;return num;}
5      void start(){num=0;Tr[0].l=Tr[0].r=Tr[0].h=0;Tr[0].v=inf;}
6      int mg(int x,int y){
7          if(!x)return y;
8          if(Tr[x].v>Tr[y].v)swap(x,y);
9          int o=New(Tr[x]);Tr[o].r=mg(Tr[o].r,y);
10         if(Tr[Tr[o].l].h<Tr[Tr[o].r].h)swap(Tr[o].l,Tr[o].r);
11         Tr[o].h=Tr[Tr[o].r].h+1;return o;}
12     void add(int&k,int v,int x,int y){
13         int o=New(Tr[0]);
14         Tr[o].v=v;Tr[o].x=x;Tr[o].y=y;
15         k=mg(k,o);    }}
16 using namespace Left_Tree;
17 struct SPFA{//SPFA, 这里要记录路径
18     void in(){tot=0;CL(fir);}void ins(x,y,z){}
19     void work(int S,int n){//F[]求最短路从哪个点来▯FF[]记最短路从哪条边来
20 }}A;
21 struct Kshort{
22 int tot,n,m,S,T,k,fir[N],va[M],la[M],ne[M];bool v[N];
23 struct P{
24     int x,y,z;P(){}P(int x,int y,int z):x(x),y(y),z(z){}
25     bool operator<(P a)const{return a.z<z;}};
26 priority_queue<P>Q;void in(){tot=0;CL(fir);}
27 void ins(x,y,z){}
28 void init(){//将图读入
29     int i,x,y,z;in();A.in();start();rd(n,m)
30     fr(i,1,m)rd(x,y,z),A.ins(y,x,z),ins(x,y,z);
31     rd(S,T,k);if(S==T)k++;//注意起点终点相同的情况
32     A.work(T,n);}//A是反向边
33 void dfs(int x){
34     if(v[x])return;v[x]=1;if(F[x])RT[x]=RT[F[x]];
35     for(int i=fir[x],y;i;i=ne[i])if(y=la[i],A.d[y]!=inf&&FF[x]!=i)
36         add(RT[x],A.d[y]-A.d[x]+va[i],x,y);
37     for(int i=A.fir[x];i;i=A.ne[i])if(F[A.la[i]]==x)dfs(A.la[i]);}
38 int work(){//返回答案, 没有返回-1
39     int i,x;dfs(T);
40     if(!--k)return A.d[S]==inf?-1:A.d[S];
41     P u,w;if(RT[S])Q.push(P(S,RT[S],A.d[S]+Tr[RT[S]].v));
42     for(;k--;){
43         if(Q.empty())return -1;u=Q.top();Q.pop();
44         if(x=mg(Tr[u.y].l,Tr[u.y].r))
45             Q.push(P(u.x,x,Tr[x].v-Tr[u.y].v+u.z));
46         if(RT[x=Tr[u.y].y])Q.push(P(x,RT[x],u.z+Tr[RT[x]].v));}
47     return u.z;}}G;
```

3.20   动态 MST

```
1  int to[N],v[N],q[N];LL Area;
2  struct P{int x,y;double o;}p[N],e[N];
3  struct cmp{bool operator()(int a,int b){return e[a].o<e[b].o;}};
4  LL operator*(P a,P b){return 1ll*a.x*b.y-1ll*a.y*b.x;}
5  set<int,cmp>S[N];set<int,cmp>::iterator it;
6  void ADD(int x,int y){}//在新图中加边 不能有重复点 可以有多余边
7  void Planet(n,m){ //n为点数, m为边数, p[i].x/y为n个点坐标
8      int j,x,y,cnt=0;
9      fr(i,0,m-1){
10         rd(x);rd(y); //第i条边连接第x个点和第y个点
11         e[i<<1]=P{x,y, atan2(p[y].x-p[x].x,p[y].y-p[x].y)}; //加双向边, 支持双向边权
```

```
12      e[i<<1|1]=P{y,x, atan2(p[x].x-p[y].x,p[x].y-p[y].y)};
13    }m*=2;
14    fr(i,0,m-1)S[e[i].x].insert(i);
15    fr(i,0,m-1)if(!v[i]){
16        for(q[t=1]=j=i;;q[++t]=j=*it){ //按极角排序选最少转动角度找出下一条边
17            it=S[e[j].y].upper_bound(j^1);
18            if(it==S[e[j].y].end())it=S[e[j].y].begin();
19            if(*it==i)break;
20        }Area=0;fr(j,1,t) Area+=p[e[q[j]].x]*p[e[q[j]].y],v[q[j]]=1;
21        if(Area>0){cnt++;fr(j,1,t)to[q[j]]=cnt;} //面积大于0是有限区域
22    }fr(i,0,m-1)ADD(to[i],to[i^1]);} //若这条边有边权，在e[i].z中记录 // 如果to[i]为0不一定是外平面，要判在一个最小的多边形内
```

3.21 树同构

```
1  // O(1) 求逆 时间复杂度 O(n) MOD 需要是质数
2  #define fors(i) for (auto i : e[x]) if (i != p)
3  int ra[N]; void prepare() {
4      for (int i = 0; i < N; ++ i) ra[i] = rand() % MOD;}
5  struct Sub {
6      vector<int> s; int d1, d2, H1, H2;
7      Sub() {d1 = d2 = 0; s.clear();}
8      void add(int d, int v) { s.push_back(v);
9          if (d>d1) d2=d1, d1=d; else if (d>d2) d2=d; }
10      int hash() { H1 = H2 = 1; for (int i : s) {
11          H1 = (ll) H1 * (ra[d1]+i) % MOD;
12          H2 = (ll) H2 * (ra[d2]+i) % MOD; } return H1;}
13      pii del(int d, int v) { if (d==d1)
14          return {d2+1, (ll)H2*reverse(ra[d2]+v) % MOD};
15          return {d1+1, (ll)H1*reverse(ra[d1]+v) % MOD};}};
16  pii U[N]; int A[N]; Sub tree[N]; //A[x]为以x为根的哈希值
17  void dfsD(int x, int p) { tree[x] = Sub();
18      fors(i) { dfsD(i, x);
19          tree[x].add(tree[i].d1 + 1, tree[i].H1); }
20      tree[x].hash(); }
21  void dfsU(int x, int p) {
22      if (p) tree[x].add(U[x].first, U[x].second);
23      A[x] = tree[x].hash();
24      fors(i){U[i]=tree[x].del(tree[i].d1+1,tree[i].H1);
25          dfsU(i, x); } }
```

3.22 虚树

```
1  rd(m);v[1]=a[1]=1,t=++m; //选定m个关键点，1号点直接加入虚树
2  fr(i,2,m)rd(x),v[a[i]]=V[a[i]]=1; //v[]表示是选的点或LCA后的点，V[]表示选的点
3  for(sort(a+1,a+m+1,cmp),i=1;i<m;i++) //新加必要关键点
4      if(!v[x=lca(a[i],a[i+1])])v[a[++t]=x]=1;
5  for(m=t,sort(a+1,a+m+1,cmp),ed=0,q[t=1]=1,i=2; i<=m; ins(q[t],a[i]),q[++t]=a[i++])
6      for(;st[a[i]]<st[q[t]]||en[a[i]]>en[q[t]];t--); //再排一遍序，单调栈建虚数
7  fr(i,1,m)v[a[i]]=V[a[i]]=fir[a[i]]=0; //求解答案后还原
```

3.23 动态最小生成树

```
1  int n, m, q; // O((m + q) log q)
2  struct EdgeInfo {
3  int u, v, w, l, r; EdgeInfo(int u,int v,int w,int l,int r) :
4      u(u),v(v),w(w),l(l),r(r){} EdgeInfo() {} };
5  long long ans[N]; int find(int f[], int u) {
6      return f[u] == u ? u : f[u] = find(f, f[u]); }
7  bool join(int f[], int u, int v){ u=find(f,u), v=find(f, v);
8      if (u == v) return false; return f[u] = v, true; }
9  void dfs(int l, int r, int n, const vector<EdgeInfo> &list, long long base) {
10      if (list.empty())
11          { for (int i=l; i<=r; i++) ans[l]=base; return ; }
12      static vector<EdgeInfo> all,part;all.clear();part.clear();
13      for (auto &e : list) if (e.l <= l && r <= e.r) {
14          all.push_back(e);
15          } else if (l <= e.r && e.l <= r) {part.push_back(e);}
16      static int f[N], color[N], id[N]; // Contraction
17      for (int i = 0; i < n; i++) f[i] = color[i] = i;
18      for (auto &e : part) join(f, e.u, e.v);
19      for (auto &e : all) if (join(f, e.u, e.v)) {
20          join(color, e.u, e.v); base += e.w; }
21      if (l == r) { ans[l] = base; return ; }
22      for (int i = 0; i < n; i++) id[i] = -1;
23      int tot = 0; for (int u = 0; u < n; u++) {
```

```
24        int v = find(color, u); if (id[v] == -1) id[v] = tot++;
25        id[u] = id[v]; } int m = 0; // Reduction
26     for (int i = 0; i < tot; i++) f[i] = i;
27     for (auto &e : part) { e.u = id[find(color, e.u)];
28        e.v = id[find(color, e.v)]; }
29     for (auto &e : all) {
30        e.u = id[find(color, e.u)], e.v = id[find(color, e.v)];
31        if (e.u == e.v) continue; assert(e.u<tot && e.v<tot);
32        if (join(f, e.u, e.v)) all[m++] = e; }
33     all.resize(m); vector<EdgeInfo> new_list;
34     for (int i=0, j=0; i < part.size() || j < all.size(); ) {
35        if(i<part.size()&&(j==all.size()||all[j].w>part[i].w)){
36           new_list.push_back(part[i++]);
37        } else { new_list.push_back(all[j++]); } }
38     int mid = (l + r) / 2; dfs(l, mid, tot, new_list, base);
39     dfs(mid + 1, r, tot, new_list, base); }
40 int main() { scanf("%d %d %d", &n, &m, &q);
41     vector<pair<int, int> > memo; static int u[N], v[N], w[N];
42     for (int i = 0; i < m; i++) {
43        scanf("%d %d %d", &u[i], &v[i], &w[i]);
44        --u[i], --v[i]; memo.push_back({0, w[i]}); }
45     vector<EdgeInfo> info; // 把第 k 条边权值改为 d
46     for (int i = 0; i < q; i++) {
47        int k, d; scanf("%d %d", &k, &d); --k;
48        if (memo[k].first < i) {
49           info.push_back({u[k], v[k], memo[k].second, memo[k].first, i - 1});
50        } memo[k] = {i, d}; }
51     for (int i = 0; i < m; i++) {
52        info.push_back({u[i], v[i], memo[i].second, memo[i].first, q - 1}); }
53     sort(info.begin(), info.end(), [&](const EdgeInfo &a, const EdgeInfo &b) { return a.w < b.w; });
54     dfs(0, q - 1, n, info, 0);
55     for (int i = 0; i < q; i++) printf("%lld\n", ans[i]); }
```

3.24   树链的交

假设两条链$(a_1, b_1), (a_2, b_2)$的lca分别为c_1, c_2.再求出$(a_1, a_2), (a_1, b_2), (b_1, a_2), (b_1, b_2)$的lca,记为$d_1, d_2, d_3, d_4$.将$(d_1, d_2, d_3, d_4)$按照深度从小到大排序,$(c_1, c_2)$也从小到大排序.两条链有交当且仅当$dep[c_1] \leq dep[d_1]$且$dep[c_2] \leq dep[d_3]$,则$(d_3, d_4)$是链交的两个端点.

3.25   最小乘积生成树与匹配

假设边权有两个值a, b,现在要求一个生成树或者匹配使得$sum_a \times sum_b$最小.这两类问题本质是相同的.思路如下:

假设一个方案中a的和记为S_a,b的和记为S_b.则要使乘积最小,显然可能的(S_a, S_b)必然处于下凸壳上.现在,先求出使得S_a最小的方案$P_1(a_1, b_1)$以及使得S_b最小的方案$P_2(a_2, b_2)$,可以直接Kruscal或者KM.然后,要找到距离线段P_1P_2最远的一个点$C(a, b)$,分治下去.找到C的方法如下:

令向量AB与AC叉积最大,即最大化$(a - a_1)(b_2 - b_1) - (b - b_1)(a_2 - a_1) = a(b_2 - b_1) + b(a_1 - a_2) - a_1(b_2 - b_1) + b_1(a_2 - a_1)$.

后两项是常数,不管,所以我们要最大化的目标是$a(b_2 - b_1) + b(a_1 - a_2)$.这样每条边的边权就变成了$k_1a + k_2b$的形式,可以Kruscal或者KM求出.

3.26   图论知识
3.26.1   完全动态MST

一个N个点M条边的无向图, 每次可以修改任意一条边的权值, 在每个修改操作后输出当前最小生成树的边权和. $N, M, Q \leq 50000$.

我们假设, 有一个图$\{S\}$, 有k条边在之后会被修改. 在k个MST中, 有些边永远出现在这些MST 中, 而有些边永远不会出现在这些MST中. 我们可以尝试求出这些边, 从而缩小图的规模.

找出无用边   将需要修改的边标记为∞, 然后跑MST, 这时不在MST上的且值不为∞边必为无用边, 删除这些边, 减少边数（注意还原）.

找出必须边   将需要修改的边标记为$-\infty$, 然后跑MST, 这时在MST上且不为$-\infty$的边为必须边, 将这些边连接的点合并, 缩小点集（注意还原）.

假设当前区间内需要修改的边数为k, 进行删去无用边和找出必须边操作后, 图中最多剩下$k + 1$个点和$2k$条边. 如果每次都暴力求MST, 那么时间复杂度为$O(nlg^2n)$; 如果利用排好序的边求MST, 并使用路径压缩+按秩合并的并查集, 那么时间复杂度为$O(nlgn\alpha(n))$.

3.26.2   LCT常见应用

动态维护边双   可以通过LCT来解决一类动态边双连通分量问题. 即静态的询问可以用边双连通分量来解决, 而树有加边等操作的问题.

把一个边双连通分量缩到LCT的一个点中, 然后在LCT上求出答案. 缩点的方法为加边时判断两点的连通性, 如果已经联通则把两点在目前LCT路径上的点都缩成一个点.

动态维护基环森林   通过LCT可以动态维护基环森林, 即每个点有且仅有一个出度的图. 有修改操作, 即改变某个点的出边. 对于每颗基环森林记录一个点为根, 并把环上额外的一条边单独记出, 剩下的边用LCT维护. 一般使用有向LCT 维护.

修改时分以下几种情况讨论: 1, 修改的点是根, 如果改的父亲在同一个联通块中, 直接改额外边, 否则删去额外边, 在LCT上加边. 2, 修改的点不是根, 那么把这个点和其父亲的联系切除. 如果该点和根在一个环上, 那么把多的那条边加到LCT上. 最后如果改的那个父亲和修改的点在一个联通块中, 记录额外边, 否则LCT上加边. 具体见例题.

3.26.3   LCT解决子树询问

通过记录轻边信息可以快速地维护出整颗LCT的一些值. 如子树和, 子树最大值等. 在Access时要进行虚实边切换, 这时减去实边的贡献, 并加上新加虚边的贡献即可. 有时需要套用数据结构, 如Set来维护最值等问题.

模板: 1, $x \to y$链$+z$; 2, $x \to y$链变为z; 3, 在以x为根的树对y子树的点权求和; 4, $x \to y$ 链取max; 5, $x \to y$链求和; 6, 连接x y; 7, 断开x y. 保证操作合法.

V 单点值; sz 平衡树的 SZ; mv 链上最大; S 链上和; sm 区间相同标记; lz 区间加标记; B 虚边之和; ST 子树信息和; SM 子树+链上信息和. 更新时, $S[x] = S[c[x][0]] + S[c[x][1]] + V[x]$, $ST[x] = B[x] + ST[c[x][0]] + ST[c[x][1]]$, $SM[x] = S[x] + ST[x]$.

3.26.4 差分约束

若要使得所有变量两两的值最接近, 则将如果将源点到各点的距离初始化为0. 若要使得某一变量与其余变量的差最大, 则将源点到各点的距离初始化为∞, 其中之一为0. 若求最小方案则跑最长路, 否则跑最短路.

3.26.5 斯坦纳树

在一个无向带权图 $G = (V, E)$ 中, 将指定的 k 个点连通的一颗树称为斯坦纳树, 边权总和最小的斯坦纳树称为最小斯坦纳树.

我们可以用DP+SPFA的方法求解斯坦纳树. 用 $F_{i,state}$ 表示以 i 为根, 指定集合中的点的联通状态为 $state$ 的生成树的最小总权值, 有两种转移方程.

第一种, 通过两个子集合并进行转移, 即 $F_{i,state} = min(F_{i,subset1} + F_{i,subset2})$, 这一部分使用DP完成.

第二种, 在当前的联通状态下, 对该联通状态进行松弛操作, 即 $F_{i,state} = min(F_{i,state}, F_{j,state} + w(i, j))$, 这一部分使用SPFA完成.

时间复杂度 $O(V * 3^k + cE * 2^k)$, c 为SPFA复杂度中的常数.

3.26.6 李超线段树

李超线段树可以动态在添加若干条线段或直线 $(a_i, b_i) \to (a_j, b_j)$, 每次求 $[l, r]$ 上最上面的那条线段的值. 思想是让线段树中一个节点只对应一条直线, 如果在这个区间加入一条直线, 那么分类讨论. 如果新加的这条直线在左右两端都比原来的更优, 则替换原来的直线, 将原来的直线扔掉. 如果左右两端都比原来的劣, 将这条直线扔掉. 如果一段比原来的优, 一段比原来的劣, 那么判断一下两条线的交点, 判断哪条直线可以完全覆盖一段一半的区间, 把它保留, 另一条直线下传到另一半区间. 时间复杂度 $O(nlgn)$.

3.26.7 吉如一线段树

吉如一线段树能解决一类区间和某个数取最大或最小, 区间求和的问题. 以区间取最小值为例, 在线段树的每一个节点额外维护区间中的最大值 ma, 严格次大值 se 以及最大值个数 t. 现在假设我们要让区间 $[L, R]$ 对 x 取min, 先在线段树中定若干个节点, 对于每个节点分三种情况讨论: 1, 当 $ma \le x$ 时, 显然这一次修改不会对这个节点产生影响, 直接退出; 2, 当 $se < x < ma$ 时, 显然这一次修改只会影响到所有最大值, 所以把 num 加上 $t * (x - ma)$, 把 ma 更新为 x, 打上标记退出; 3, 当 $se \ge x$ 时, 无法直接更新着一个节点的信息, 对当前节点的左儿子和右儿子递归处理. 单次操作均摊复杂度 $O(lg^2 n)$.

3.26.8 二分图最大匹配

最大独立集 最小覆盖点集 最小路径覆盖 最大独立集指求一个二分图中最大的一个点集, 使得该点集内的点互不相连. 最大独立集=总顶点数-最大匹配数. 最小覆盖点集指用最少的点, 使所有的边至少和一个点有关联. 最小覆盖点集=最大匹配数. 最小路径覆盖指一个DAG图 G 中用最少的路径使得所有点都被经过. 最小路径覆盖=总点数-最大匹配数 (拆点构图). 最大独立集 S 与最小覆盖集 T 互补. 构造方法: 1.做最大匹配, 没有匹配的空闲点 $u \in S$ 2.如果 $u \in S$ 那么 u 的邻点必然属于 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.

二分图最大匹配关键点 关键点指的是一定在最大匹配中的点. 由于二分图左右两侧是对称的, 我们只考虑找左侧的关键点. 先求任意一个最大匹配, 然后给二分图定向: 匹配边从右到左, 非匹配边从左到右, 从左侧每个不在最大匹配中的点出发DFS, 给到达的那些点打上标记, 最终左侧每个没有标记的匹配点即将爱为关键点. 时间复杂度 $O(n + m)$.

Hall定理 二分图 $G = (X, Y, E)$ 有完备匹配的充要条件是: 对于 X 的任意一个子集 S 都满足 $|S|leq|A(S)|$, $A(S)$ 是 Y 的子集, 是 S 的邻集. 邻集的定义是与 S 有边的点集.

3.26.9 稳定婚姻问题

有 n 位男士和 n 位女士, 每个人都对每个异性有一个不同的喜欢程度, 现在使得每人恰好有一个异性配偶. 如果男士 u 和女士 v 不是配偶但喜欢对方的程度都大于喜欢各自当前的配偶, 则称他们为一个不稳定对. 稳定婚姻问题是为了找出一个不含不稳定对的方案.

稳定婚姻问题的经典算法为求婚-拒绝算法, 即男士按自己喜欢程度从高到底依次向每位女士求婚, 直到有一个接受他. 女士遇到比当前配偶更差的男士时拒绝他, 遇到更喜欢的男士时就接受他, 并抛弃以前的配偶. 被抛弃的男士继续按照列表向剩下的女士依次求婚, 直到所有人都有配偶. 如果算法一定能得到一个匹配, 而且这个匹配一定是稳定的. 时间复杂度 $O(n^2)$.

3.26.10 最大流和最小割

常见建模方法 拆点; 黑白染色; 流量正无穷表示冲突; 缩点; 数据结构优化建图; 最小割 每个变元拉一条 S 到 T 的链, 割在哪里表示取值, 相互连边表示依赖关系; 先把收益拿下, 在考虑冲突与代价的影响.

判断一条边是否可能/一定在最小割中 令 G' 为残量网络 G 在强联通分量缩点之后的图. 那么一定在最小割中的边 (u, v): (u, v) 满流, 且在 G' 中 $u = S, v = T$; 可能在最小割方案中的边 (u, v): (u, v) 满流, 或 (u, v) 满流, 且在 G' 中 $u \ne v$.

混合图欧拉回路 把无向边随便定向, 计算每个点的入度和出度, 如果有某个点出入度之差 $deg_i = in_i - out_i$ 为奇数, 肯定不存在欧拉回路. 对于 $deg_i > 0$ 的点, 连接边 $(i, T, deg_i/2)$; 对于 $deg_i < 0$ 的点, 连接边 $(S, i, -deg_i/2)$. 最后检查是否满流即可.

3.26.11 一些网络流建图

无源汇有上下界可行流

每条边 (u, v) 有一个上界容量 $C_{u,v}$ 和下界容量 $B_{u,v}$, 我们让下界变为0,上界变为 $C_{u,v} - B_{u,v}$, 但这样做流量不守恒. 建立超级源点 SS 和超级汇点 TT, 用 du_i 来记录每个节点的流量情况, $du_i = \sum B_{j,i} - \sum B_{i,j}$, 添加一些附加弧. 当 $du_i > 0$ 时, 连边 (SS, i, du_i); 当 $du_i < 0$ 时, 连边 $(i, TT, -du_i)$. 最后对 (SS, TT) 求一次最大流即可, 当所有附加边全部满流时 (即 $maxflow == du_i > 0$) 时有可行解.

有源汇有上下界最大可行流

建立超级源点 SS 和超级汇点 TT, 首先判断是否存在可行流, 用无源汇有上下界可行流的方法判断. 增设一条从 T 到 S 没有下界容量为无穷的边, 那么原图就变成了一个无源汇有上下界可行流问题. 同样地建图后, 对 (SS, TT) 进行一次最大流, 判断是否有可行解.

如果有可行解, 删除超级源点 SS 和超级汇点 TT, 并删去 T 到 S 的这条边, 再对 (S, T) 进行一次最大流, 此时得到的 $maxflow$ 即为有源汇有上下界最大可行流.

有源汇有上下界最小可行流

建立超级源点 SS 和超级汇点 TT, 和无源汇有上下界可行流一样新增一些边, 然后从SS到TT跑最大流. 接着加上边 (T, S, ∞), 再从 SS 到 TT 跑一遍最大流.

如果所有新增边都是满的, 则存在可行流, 此时 T 到 S 这条边的流量即为最小可行流.

有上下界费用流

如果求无源汇有上下界最小费用可行流或有源汇有上下界最小费用最大可行流, 用1.6.3.1/1.6.3.2 的构图方法, 给边加上费用即可.

求有源汇有上下界最小费用最小可行流, 要先用1.6.3.3的方法建图, 先求出一个保证必要边满流情况下的最小费用. 如果费用全部非负, 那么这时的费用就是答案. 如果费用有负数, 那么流多了可能更好, 继续做从 S 到 T 的流量任意的最小费用流, 加上原来的费用就是答案.

费用流消负环 新建超级源SS汇TT, 对于所有流量非空的负权边e, 先流满 (ans+=e.f*e.c, e.rev.f+=e.f, e.f=0), 再连边SS→e.to, e.from→TT, 流量均为e.f(>0), 费用均为0. 再连边T→S流量∞费用0. 此时没有负环了. 做一遍SS到TT的最小费用最大流, 将费用累加ans, 拆掉T→S的那条边 (此边的流量为残量网络中S→T的流量). 此时负环已消, 再继续跑最小费用最大流.

二物流　水源S1, 水汇T1, 油源S2, 油汇T2, 每根管道流量共用. 求流量和最大.

建超级源SS1汇TT1, 连边SS1→S1,SS1→S2,T1→TT1,T2→TT1, 设最大流为x1.

建超级源SS2汇TT2, 连边SS2→S1,SS2→T2,T1→TT2,S2→TT2, 设最大流为x2.

则最大流中水流量 $\frac{x1+x2}{2}$, 油流量 $\frac{x1-x2}{2}$.

3.26.12　割点于割边

割点 割边　一个点 u 是割点, 当且仅当: 1. u 为非树根且有树边 (u,v) 满足 $dfn_u \leq low_v$; 2. u 为树根且有多于一个的子树. 一条无向边 (u,v) 是桥, 当且仅当 (u,v) 是树边, 且满足 $dfn_u < low_v$.

3.26.13　2-SAT

如果选 A 就必须选 B 就从 A 向 B 连一条边, 如果两个只能选一个的条件在同一个强连通分量中就不合法. 输出可行方案可以比较 X 和 X' 的 bl 的大小, 大的选 X'. 建图优化一般考虑前后缀的合并.

3.26.14　弦图

定义　我们称连接环中不相邻的两个点的边为弦. 一个无向图称为弦图, 当图中任意长度都大于3 的环都至少有一个弦. 弦图的每一个诱导子图一定是弦图.

单纯点　设 $N(v)$ 表示与点 v 相邻的点集. 一个点称为单纯点当 $v + N(v)$ 的诱导子图为一个团. 引理: 任何一个弦图都至少有一个单纯点, 不是完全图的弦图至少有两个不相邻的单纯点.

完美消除序列　一个序列 $v_1, v_2, ..., v_n$ 满足 v_i 在 $v_i, v_{i+1}, ..., v_n$ 的诱导子图中为一个单纯点. 一个无向图是弦图当且仅当它有一个完美消除序列.

最大势算法　最大势算法能判断一个图是否是弦图. 从 n 到1的顺序依次给点标号 标号为 i 的点出现在完美消除序列的第 i 个. 设 $label_i$ 表示第 i 个点与多少个已标号的点相邻, 每次选择 $label_i$ 最大的未标号的点进行标号.

然后判断这个序列是否为完美序列. 如果依次判断 $v_{i+1}, ..., .v_n$ 中所有与 v_i 相邻的点是否构成一个团, 时间复杂度为 $O(nm)$. 考虑优化, 设 $v_{i+1}, ..., v_n$ 中所有与 v_i 相邻的点依次为 $v_{j1}, ..., v_{jk}$. 只需判断 v_{j1} 是否与 $v_{j2}, ..., v_{jk}$ 相邻即可. 时间复杂度 $O(n + m)$.

弦图的染色　按照完美消除序列中的点倒着给图中的点贪心染尽可能最小的颜色, 这样一定能用最少的颜色数给图中所有点染色. 弦图的团数＝染色数.

最大独立集　完美消除序列从前往后能选就选. 最大独立集＝最小团覆盖.

3.26.15　三元环

有一种简单的写法, 对于每条无向边 (u,v), 如果 $deg_u < deg_v$, 那么连有向边 (u,v), 否则连有向边 (v,u)(注意度数相等以点标号为第二关键字判断). 然后枚举每个点 x, 假设 x 是三元环中度数最小的点, 然后暴力往后面枚两条边找到点 y, 判断是否有边 (x,y) 即可. 可以证明, 这样的时间复杂度也是为 $O(m\sqrt{m})$ 的.

3.26.16　图同构

令 $F_t(i) = (F_{t-1}(i) * A + \sum_{i \to j} F_{t-1}(j) * B + \sum_{j \to i} F_{t-1}(j) * C + D * (i - a)) \bmod P$, 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.

3.26.17　竞赛图存在 Landau's Theorem

n 个点竞赛图点按出度按升序排序, 前 i 个点的出度之和不小于 $\frac{i(i-1)}{2}$, 度数总和等于 $\frac{n(n-1)}{2}$. 否则可以用优先队列构造出方案.

3.26.18　Ramsey Theorem

6个人中至少存在3人相互认识或者相互不认识. $R(3,3) = 6, R(4,4) = 18$

3.26.19　树的计数 Prufer序列

树和其prufer编码一一对应, 一颗 n 个点的树, 其prufer编码长度为 $n - 2$, 且度数为 d_i 的点在prufer 编码中出现 $d_i - 1$ 次.

由树得到序列: 总共需要 $n - 2$ 步, 第 i 步在当前的树中寻找具有最小标号的叶子节点, 将与其相连的点的标号设为Prufer序列的第 i 个元素 p_i, 并将此叶子节点从树中删除, 直到最后得到一个长度为 $n - 2$ 的Prufer 序列和一个只有两个节点的树.

由序列得到树: 先将所有点的度赋初值为1, 然后加上它的编号在Prufer序列中出现的次数, 得到每个点的度; 执行 $n - 2$ 步, 第 i 步选取具有最小标号的度为1的点 u 与 $v = p_i$ 相连, 得到树中的一条边, 并将 u 和 v 的度减一. 最后再把剩下的两个度为1的点连边, 加入到树中.

相关结论: n 个点完全图, 每个点度数依次为 $d_1, d_2, ..., d_n$, 这样生成树的棵树为: $\frac{(n-2)!}{(d_1-1)!(d_2-1)!...(d_n-1)!}$.

左边有 n_1 个点, 右边有 n_2 个点的完全二分图的生成树棵树为 $n_1^{n_2-1} + n_2^{n_1-1}$.

m 个连通块, 每个连通块有 c_i 个点, 把他们全部连通的生成树方案数: $(\sum c_i)^{m-2} \prod c_i$

3.26.20　有根树的计数

首先, 令 $S_{n,j} = \sum_{1 \leq j \leq n/j}$; 于是 $n + 1$ 个结点的有根树的总数为 $a_{n+1} = \frac{\sum_{j=1}^{n} ja_j S_{n-j}}{n}$. 注: $a_1 = 1, a_2 = 1, a_3 = 2, a_4 = 4, a_5 = 9, a_6 = 20, a_9 = 286, a_11 = 1842$.

3.26.21　无根树的计数

当 n 是奇数时, 有 $a_n - \sum_i^{n/2} a_i a_{n-i}$ 种不同的无根树.

当 n 时偶数时, 有 $a_n - \sum_i^{n/2} a_i a_{n-i} + \frac{1}{2} a_{n/2}(a_{n/2} + 1)$ 种不同的无根树.

3.26.22　生成树计数 Kirchhoff's Matrix-Tree Thoerem

Kirchhoff Matrix $T = Deg - A$, Deg 是度数对角阵, A 是邻接矩阵. 无向图度数矩阵是每个点度数; 有向图度数矩阵是每个点入度.

邻接矩阵 $A[u][v]$ 表示 $u->v$ 边个数, 重边按照边数计算, 自环不计入度数.

无向图生成树计数: $c = |K$ 的任意1个 $n-1$ 阶主子式 $|$

有向图外向树计数: $c = |$ 去掉根所在的那阶得到的主子式 $|$

3.26.23　有向图欧拉回路计数 BEST Thoerem

$$ec(G) = t_w(G) \prod_{v \in V} (\deg(v) - 1)!$$

其中 deg 为入度(欧拉图中等于出度), $t_w(G)$ 为以 w 为根的外向树的个数. 相关计算参考生成树计数.

欧拉连通图中任意两点外向树个数相同: $t_v(G) = t_w(G)$.

3.26.24　Edmonds Matrix

Edmonds matrix A of a balanced ($|U| = |V|$) bipartite graph $G = (U, V, E)$:

$$A_{ij} = \begin{cases} x_{ij} & (u_i, v_j) \in E \\ 0 & (u_i, v_j) \notin E \end{cases}$$

where the x_{ij} are indeterminates. G 有完美匹配当且仅当关于 x_{ij} 的多项式 $det(A_{ij})$ 不恒为0. 完美匹配的个数等于多项式中单项式的个数.

3.26.25　Count Acyclic Orientations

The chromatic polynomial is a function $P_G(q)$ that counts the number of q-colorings of G.

Triangle K_3 : $t(t-1)(t-2)$

Complete graph K_n : $t(t-1)(t-2)\cdots(t-(n-1))$

Tree with n vertices : $t(t-1)^{n-1}$

Cycle C_n : $(t-1)^n + (-1)^n(t-1)$

The number of acyclic orientations of an n-vertex graph G is $(-1)^n P_G(-1)$, where $P_G(q)$ is the chromatic polynomial of the graph G.

3.26.26　Boruvka最小生成树算法

初始时每个点作为一个单独连通块,每轮对于每个连通块找出其与其他连通块之间边权最小的连边,并将这些边连起来,更新连通块.容易证明最多进行log轮.本算法适用于完全图且边权由两个点权计算得来(例如异或等)的模型.

3.26.27　最大乘积生成树与最大乘积匹配

4. String

4.1　KMP 扩展 KMP 循环最小表示

```
void kmp(string A,int*p){//A 为模式串, p 为失配数组
    int n=A.length(),i=1,j=0;
    for(CL(p);i<n;i++){
        for(;j&&A[j]^A[i];j=p[j-1]);
        if(A[i]==A[j])j++;
        p[i]=j;}
}int gans(string A,string B,int*p){
//B 为标准串, A 为待匹配串, p 为失配数组
    int n=B.length(),m=A.length(),j=0;
    fr(i,0,m-1){
        for(;j&&B[j]^A[i];j=p[j-1]);
        if(B[j]==A[i])j++;
        if(j==n)an++,j=p[j-1];
    } return an; }
void exkmp(char *s, int *a, int n) {
// a[i]表示第i个后缀与整个串的最长公共前缀
// 如果想求一个字符串相对另外一个字符串的最长公共前缀, 可以把他们拼接起来从而求得
    a[0] = n; int p = 0, r = 0;
    for (int i = 1; i < n; ++i) {
        a[i] = (r > i) ? min(r - i, a[i - p]) : 0;
        while (i + a[i] < n && s[i + a[i]] == s[a[i]]) ++a[i];
        if (r < i + a[i]) r = i + a[i], p = i;
}}string mini(string A){// 最小表示法, 返回最小表示
    string AN;AN.clear();
    int i=0,j=1,k=0,x,n=A.length();
    for(;j<n&&k<n;){
        x=A[(i+k)%n]-A[(j+k)%n];
        x?x>0?i=max(j,i+k+1),j=i+1 :j=j+k+1,k=0:k++;
    }for(j=0;j<n;j++)AN=AN+A[(i+j)%n];
    return AN;}
```

4.2　Manacher

```
// 这段代码仅仅处理奇回文▢使用时请往字符串中间加入 # 来使用
for(int i = 1, j = 0; i != (n << 1) - 1; ++i){
    int p=i>>1, q = i - p, r = ((j + 1) >> 1) + l[j] - 1;
    l[i] = r < q ? 0 : min(r - q + 1, l[(j << 1) - i]);
    while (p - l[i] != -1 && q + l[i] != n
        && s[p - l[i]] == s[q + l[i]]) l[i]++;
    if(q + l[i] - 1 > r) j=i;
    a += l[i];
}
```

4.3　Aho-Corasick Automation AC 自动机

```
#define root 0
int tr[maxv][alphabet],cnt;//AC Automation
int en[maxv][N], fail[maxv];//en[u][i]指的是匹配到第i位,在自动机的节点u处可以成功匹配多少个模式串
string st[maxv];//st存储对应节点的字符串
void ins(int u,int d,const string &s,const int suflen)//suflen在这个程序中指的是这一个模式串后面通配符长度
{
    if (d==s.size()) {
        for (int i=suflen; i<=n; i++) en[u][i]++;
        return;
    }
    if (!tr[u][s[d]-'0']) tr[u][s[d]-'0']=++cnt,st[cnt]=st[u]+s[d];
    ins(tr[u][s[d]-'0'],d+1,s,suflen);
}
void buildAC()
{
```

```
16      queue<int> q;
17      for (i=0; i<alphabet; i++)
18          if (tr[root][i]) q.push(tr[root][i]);
19      while (!q.empty())
20      {
21          int u=q.front();
22          q.pop();
23          for (i=0; i<alphabet; i++)
24              if (tr[u][i]){
25                  q.push(tr[u][i]);
26                  fail[tr[u][i]]=tr[fail[u]][i];
27              }
28              else tr[u][i]=tr[fail[u]][i];
29      }
30  }
31  bool vis[maxv];
32  void update(int u)//这一段一般来说必不可少，因为走到"abcd"的节点上实际上也匹配了"cd"模式串
33  {
34      if (vis[u] || u==root) return;
35      vis[u]=1, update(fail[u]);
36      for (int i=0; i<=n; i++) en[u][i]+=en[fail[u]][i];
37  }
38  void work(){//构建一个普通AC自动机
39      doInsert();
40      buildAC();
41      for (int i=1; i<=cnt; i++) update(i);
42  }
43  /*以下是一段AC自动机上求字典序最小的dp实例
44      f[i][v]:待匹配串长度为i,走到v节点总共匹配模式串次数的最大值
45      pre[i][v], match[i][v]:记录转移路径,便于输出答案
46      struct rec{pair<int,int> p; int pos;}是一个更新状态转移优先顺序的记录
47
48      int seq[maxv];//按照seq的顺序更新状态,排在前面的点字典序小
49      memset(f,128,sizeof(f)), memset(pre,-1,sizeof(pre));
50      f[0][root]=en[root][n];
51      for (i=0; i<=cnt; i++) seq[i]=i;//因为一开始只有root处有dp值,也就是说其他点根本没有用,因此"初始字典序"可以是任意的
52      rec p[maxv];
53      for (i=1; i<=n; i++)
54      {
55          for (k=0; k<=cnt; k++)//因为保证了seq的字典序顺序,所以转移的时候只需要考虑>的情况
56          {
57              int u=seq[k];
58              for (j=0; j<alphabet; j++)
59              {
60                  int v=tr[u][j];
61                  if (f[i-1][u]+en[v][n-i]>f[i][v])
62                      f[i][v]=f[i-1][u]+en[v][n-i], pre[i][v]=u, alpha[i][v]=j, p[v].p=make_pair(k,j);
63          //k是当前字典序,j是新加进来字符的字典序
64              }
65          }
66          for (k=0; k<=cnt; k++) p[k].pos=k;
67          sort(p,p+cnt+1,cmp);
68          for (k=0; k<=cnt; k++) seq[k]=p[k].pos;//更新字典序顺序
69      }
70  */
```

4.4 Lydon Word Decomposition

```
1   //满足s的最小后缀等于s本身的串s称为Lyndon串.
2   //等价于: s是它自己的所有循环移位中唯一最小的一个.
3   //任意字符串s可以分解为 s = s_1 s_2 ... s_k,其中 s_i 是Lyndon串, s_i ≥ s_{i+1}.且这种分解方法是唯一的.
4   void mnsuf(char *s, int *mn, int n) { // 每个前缀的最小后缀
5       for (int i = 0; i < n; ) {
6           int j = i, k = i + 1; mn[i] = i;
7           for (; k < n && s[j] <= s[k]; ++ k)
8               if (s[j] == s[k]) mn[k] = mn[j] + k - j, ++j;
9                   else mn[k] = j = i;
10          for (; i <= j; i += k - j) {} } } // lyn+=s[i..i+k-j-1]
11  void mxsuf(char *s, int *mx, int n) { // 每个前缀的最大后缀
12      fill(mx, mx + n, -1);
13      for (int i = 0; i < n; ) {
14          int j = i, k = i + 1; if (mx[i] == -1) mx[i] = i;
15          for (; k < n && s[j] >= s[k]; ++k) {
16              j = s[j] == s[k] ? j + 1 : i;
```

```
17        if (mx[k] == -1) mx[k] = i; }
18       for (; i <= j; i += k - j) {} } }
```

4.5   Suffix Array 后缀数组

```
1  #include <bits/stdc++.h>
2  #define N 100500
3  using namespace std;
4  char s[N];
5  int n, rk[N], sa[N], ht[N], buc[N], wa[N], wb[N];
6  bool cmp(int *d, int a, int b, int c){
7      return d[a] == d[b] && d[a + c] == d[b + c];
8  }
9  void getheight(int n){
10     int i, j, k = 0;
11     for(i = 0; i < n; i++) rk[sa[i]] = i;
12     for(i = 0; i < n - 1; ht[rk[i++]] = k)
13         for(k ? k-- : 0, j = sa[rk[i] - 1]; s[i + k] == s[j + k]; k++);
14     return ;
15 }
16 void da(int n, int m = 130){
17     int i, j, p, *x =wa, *y = wb;
18     for(i = 0; i < m; i++) buc[i] = 0;
19     for(i = 0; i < n; i++) buc[x[i] = s[i]]++;
20     for(i = 1; i < m; i++) buc[i] += buc[i - 1];
21     for(i = n - 1; ~i; i--) sa[--buc[x[i]]] = i;
22     for(j = 1, p = 1; p < n; j <<= 1, m = p){
23         for(p = 0, i = n - j; i < n; i++) y[p++] = i;
24         for(i = 0; i < n; i++) if(sa[i] >= j) y[p++] = sa[i] - j;
25         for(i = 0; i < m; i++) buc[i] = 0;
26         for(i = 0; i < n; i++) buc[x[y[i]]]++;
27         for(i = 1; i < m; i++) buc[i] += buc[i - 1];
28         for(i = n - 1; ~i; i--) sa[--buc[x[y[i]]]] = y[i];
29         for(swap(x, y), x[sa[0]] = 0, p = 1, i = 1; i < n; i++)
30             x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
31     }
32     getheight(n);
33 }
34 int main(){
35     scanf("%s", s);
36     n = strlen(s);
37     da(n + 1);
38     return 0;
39 }
```

4.6   Suffix Automation 后缀自动机

```
1  #define root 0
2  struct SAM{
3      int e[N*4][26];
4      int lst, cur, link[N*4], len[N*4], cnt;
5      char lstc[N*4];//lstc仅仅用于后缀树▢表示这个后缀树节点的开头字母(也就是反序自动机上的结尾)▢用于统计后缀树上的一些信息
6      bool isleaf[N*4];
7
8      void init(){
9          memset(isleaf, 0, sizeof isleaf),
10         memset(len, 0, sizeof len),
11         mem(e, 0, sizeof e);
12         cnt = 0;
13         link[root] = -1, cur = root;
14     }
15     void extend(int c){
16         c-='a';
17         lst = cur, cur = ++cnt;
18         len[cur] = len[lst]+1, isleaf[cur] = 1, lstc[cur] = c+'a';
19
20         int u = lst, v;
21         while(u!=-1 && !e[u][c])
22             e[u][c] = cur, u = link[u];
23         if (u == -1) {link[cur] = root; continue;}
24
25         v=e[u][c];
26         if (len[v] == len[u]+1) {link[cur] = v; continue;}
27
28         int clone = ++cnt;
```

```
29          len[clone] = len[u]+1, link[clone] = link[v],
30          memcpy(e[clone], e[v], sizeof e[v]), lstc[clone] = lstc[v];
31          link[v] = link[cur] = clone;
32          for (; u!=-1 && e[u][c]==v; u=link[u])
33              e[u][c] = clone;
34      }
35  };
```

4.7   Suffix Balanced Tree 后缀平衡树

```
1  // 后缀平衡树每次在字符串开头添加或删除字符，考虑在当前字符串 S 前插入一个字符 c，那么相当于在后缀平衡树中插入一个新的后缀 cS，简单
   ↪ 的话可以使用预处理哈希二分 LCP 判断两个后缀的大小作 cmp，直接写 set，时间复杂度 O(nlg^2n)．为了方便可以把字符反过来做
2  // 例题：加一个字符或删一个字符，同时询问不同子串个数
3  struct cmp{
4      bool operator()(int a,int b){
5          int p=lcp(a,b);//注意这里是后面加 lcp是反过来的
6          if(a==p)return 0;if(b==p)return 1;
7          return s[a-p]<s[b-p];}
8  };set<int,cmp>S;set<int,cmp>::iterator il,ir;
9  void del(){S.erase(L--);}//在后面删字符
10 void add(char ch){//在后面加字符
11     s[++L]=ch;mx=0;il=ir=S.lower_bound(L);
12     if(il!=S.begin())mx=max(mx,lcp(L,*--il));
13     if(ir!=S.end())mx=max(mx,lcp(L,*ir));
14     an[L]=an[L-1]+L-mx;S.insert(L);
15 }
16 LL getan(){printf("%lld\n",an[L]);}//询问不同子串个数
```

4.8   Total LCS 子串对目标求 LCS

```
1  const int N = 2005; int H[N][N], V[N][N]; char s[N], t[N];
2  int main() { gets(s + 1); gets(t + 1);
3      int n = (int) strlen(s + 1), m = (int) strlen(t + 1);
4      for (int i = 1; i <= m; ++ i) H[0][i] = i;
5      for (int i = 1; i <= n; ++ i) {
6          for (int j = 1; j <= m; ++ j) {
7              if (s[i] == t[j]) {
8                  H[i][j] = V[i][j - 1]; V[i][j] = H[i - 1][j];
9              } else {
10                 H[i][j] = max(H[i-1][j], V[i][j-1]);
11                 V[i][j] = min(H[i-1][j], V[i][j-1]); } } }
12     for (int i = 1; i <= m; ++ i) { int ans = 0;
13         for (int j = i; j <= m; ++ j) { ans += H[n][j] < i;
14             printf("%d%c", ans, " \n"[j == m]); } } }
```

4.9   Palindrome Automation 回文自动机

```
1  struct PAM{
2      int next[N][26], fail[N], cnt[N], num[N], len[N];
3      //fail:失配后跳到fail指向的节点
4      //cnt:统计该节点表示的字符串在原串中出现了几次
5      //num:统计这个节点可以表示多少个本质不同的回文子串(其实就是fail树的深度)
6      //len:这个串的长度
7      int last, n, p, s[N];
8      int newnode(int l){
9          memset(next[p], 0, sizeof next[p]);
10         num[p] = cnt[p] = 0;
11         len[p] = l;
12         return p++;
13     }
14     void init(){
15         p = 0;
16         newnode(0), newnode(-1);
17         last = 0;
18         n = 0, s[n] = -1;
19         fail[0] = 1;
20     }
21     int get_fail(int x){
22         while (s[n - len[x] - 1] != s[n])
23             x = fail[x];
24         return x;
25     }
26
27     void add(int c){
28         c -= 'a';
```

```
29        s[++n] = c;
30        int cur = get_fail(last);
31        if (!next[cur][c]){
32            int now = newnode(len[cur] + 2);
33            fail[now] = next[get_fail(fail[cur])][c];
34            next[cur][c] = now;
35            num[now] = num[fail[now]] + 1;
36        }
37        last = next[cur][c];
38        cnt[last]++;
39    }
40    void count(){//注意在把所有字符扔进PAM以后要更新一下cnt数组
41        for (int i=p-1; i>=0; i--)
42            cnt[fail[i]] += cnt[i];
43    }
44 };
```

4.10   Tips

4.10.1   双回文串

如果$s = x_1x_2 = y_1y_2 = z_1z_2, |x_1| < |y_1| < |z_1|, x_2, y_1, y_2, z_1$是回文串, 则$x_1$和$z_2$也是回文串.

4.10.2   Border 的结构

字符串s的所有不小于$|s|/2$的border长度组成一个等差数列.

字符串s的所有 border 按长度排序后可分成$O(\log|s|)$段, 每段是一个等差数列. 回文串的回文后缀同时也是它的border.

4.10.3   子串最小后缀

设$s[p..n]$是$s[i..n], (l \le i \le r)$中最小者, 则$\operatorname{minsuf}(l, r)$等于$s[p..r]$的最短非空 border. $\operatorname{minsuf}(l, r) = \min\{s[p..r], \operatorname{minsuf}(r - 2^k + 1, r)\}, (2^k < r-l+1 \le 2^{k+1})$.

4.10.4   子串最大后缀

从左往右扫, 用set维护后缀的字典序递减的单调队列, 并在对应时刻添加"小于事件"点以便在之后修改队列; 查询直接在set里lower_bound.

5. Math 数学

5.1   CRT 中国剩余定理

```
1 bool crt_merge(LL a1, LL m1, LL a2, LL m2, LL &A, LL &M) {
2 LL c = a2 - a1, d = __gcd(m1, m2); //合并两个模方程
3 if(c % d) return 0; // gcd(m1, m2) | (a2 - a1)时才有解
4 c = (c % m2 + m2) % m2; c /= d; m1 /= d; m2 /= d;
5 c = c * inv(m1 % m2, m2) % m2; //0逆元可任意值
6 M = m1*m2*d; A = (c *m1 %M *d %M +a1) % M; return 1;}//有解
```

5.2   扩展卢卡斯

```
1 int l,a[33],p[33],P[33];
2 U fac(int k,LL n){// 求 n! mod pk^tk, 返回值 U{ 不包含 pk 的值 ,pk 出现的次数 }
3     if (!n)return U{1,0};LL x=n/p[k],y=n/P[k],ans=1;int i;
4     if(y){// 求出循环节的答案
5         for(i=2;i<P[k];i++)if(i%p[k])ans=ans*i%P[k];
6         ans=Pw(ans,y,P[k]);
7     }for(i=y*P[k];i<=n;i++) if(i%p[k])ans=ans*i%M;// 求零散部分
8     U z=fac(k,x);return U{ans*z.x%M,x+z.z};
9 }LL get(int k,LL n,LL m){// 求 C(n,m) mod pk^tk
10    U a=fac(k,n),b=fac(k,m),c=fac(k,n-m);// 分三部分求解
11    return Pw(p[k],a.z-b.z-c.z,P[k])*a.x%P[k]* inv(b.x,P[k])%P[k]*inv(c.x,P[k])%P[k];
12 }LL CRT(){// CRT 合并答案
13    LL d,w,y,x,ans=0;
14    fr(i,1,l)w=M/P[i],exgcd(w,P[i],x,y), ans=(ans+w*x%M*a[i])%M;
15    return (ans+M)%M;
16 }LL C(LL n,LL m){// 求 C(n,m)
17    fr(i,1,l)a[i]=get(i,n,m);
18    return CRT();
19 }LL exLucas(LL n,LL m,int M){
20    int jj=M,i; // 求 C(n,m)mod M,M=prod(pi^ki), 时间 O(pi^kilg^2n)
21    for(i=2;i*i<=jj;i++)if(jj%i==0) for(p[++l]=i,P[l]=1;jj%i==0;P[l]*=p[l])jj/=i;
22    if(jj>1)l++,p[l]=P[l]=jj;
23    return C(n,m);}
```

5.3   Factorial Mod 阶乘取模

```
1 // n! mod p^q Time : O(pq^2 (log^2 n)/(log p))
2 // Output : {a, b} means a*p^b
3 using Val=unsigned long long; //Val 需要 mod p^q 意义下 + *
4 typedef vector<Val> poly;
5 poly polymul(const poly &a,const poly &b){
6     int n = (int) a.size(); poly c (n, Val(0));
7     for (int i = 0; i < n; ++ i) {
8         for (int j = 0; i + j < n; ++ j) {
```

```
9          c[i + j] = c[i + j] + a[i] * b[j]; } }
10     return c; } Val choo[70][70];
11 poly polyshift(const poly &a, Val delta) {
12     int n = (int) a.size(); poly res (n, Val(0));
13     for (int i = 0; i < n; ++ i) { Val d = 1;
14         for (int j = 0; j <= i; ++ j) {
15             res[i - j] = res[i - j]+a[i]*choo[i][j]*d;
16             d = d * delta; } } return res; }
17 void prepare(int q) {
18     for (int i = 0; i < q; ++ i) { choo[i][0] = Val(1);
19         for (int j = 1; j <= i; ++ j)
20             choo[i][j]=choo[i-1][j-1]+choo[i-1][j]; } }
21 pair<Val, LL> fact(LL n, LL p, LL q) { Val ans = 1;
22     for (int r = 1; r < p; ++ r) {
23         poly x (q, Val(0)), res (q, Val(0));
24         res[0] = 1; LL _res = 0; x[0] = r; LL _x = 0;
25         if (q > 1) x[1] = p, _x = 1; LL m = (n - r + p) / p;
26         while (m) { if (m & 1) {
27                 res=polymul(res,polyshift(x,_res)); _res+=_x; }
28             m >>= 1; x = polymul(x, polyshift(x, _x)); _x+=_x; }
29         ans = ans * res[0]; }
30     LL cnt = n / p; if (n >= p) { auto tmp=fact(n / p, p, q);
31         ans = ans * tmp.first; cnt += tmp.second; }
32     return {ans, cnt}; }
```

5.4　平方剩余

```
1  // x^2=a (mod p),0 <=a<p,返回 true or false 代表是否存在解
2  // p必须是质数，若是多个单次质数的乘积 可以分别求解再用CRT合并
3  // 复杂度为 O(log n)
4  void multiply(ll &c, ll &d, ll a, ll b, ll w) {
5      int cc = (a * c + b * d % MOD * w) % MOD;
6      int dd = (a * d + b * c) % MOD; c = cc, d = dd; }
7  bool solve(int n, int &x) {
8      if (n==0) return x=0,true; if (MOD==2) return x=1,true;
9      if (power(n, MOD / 2, MOD) == MOD - 1) return false;
10     ll c = 1, d = 0, b = 1, a, w;
11     // finding a such that a^2 - n is not a square
12     do { a = rand() % MOD; w = (a * a - n + MOD) % MOD;
13         if (w == 0) return x = a, true;
14     } while (power(w, MOD / 2, MOD) != MOD - 1);
15     for (int times = (MOD + 1) / 2; times; times >>= 1) {
16         if (times & 1) multiply(c, d, a, b, w);
17         multiply(a, b, a, b, w); }
18     // x = (a + sqrt(w)) ^ ((p + 1) / 2)
19     return x = c, true; }
```

5.5　**Baby-step Giant-step BSGS** 离散对数

```
1  LL inv(LL a,LL n){LL x,y;exgcd(a,n,x,y);return(x+n)%n;}
2  LL bsgs(LL a,LL b,LL n){// 在 (a,n)=1 时求最小的 x 使得 a^x mod n=b
3      LL m=sqrt(n+0.5),e=1,i;map<LL,LL>mp;mp[1]=0;
4      for(i=1;i<m;i++)if(!mp.count(e=e*a%n))mp[e]=i;
5      e=e*a%n;e=inv(e,n);// e=a^m, 求出其逆元后放到等式右边
6      for(i=0;i<m;b=b*e%n,i++)if(mp.count(b))return i*m+mp[b];
7      return -1;// 无解
8  }LL exbsgs(LL a,LL b,LL n){// 求最小的 x 使 a^x mod n=b
9      LL V,k=0,d,e=1;
10     for(;(d=gcd(a,n))!=1;){
11         if(b%d)return b==1?0:-1;// 如果 (a,n)=1, 要么 x=0&b=1, 要么无解
12         k++;n=n/d;b=b/d;e=e*a/d%n;
13         if(e==b)return k; }// 特判
14     V=bsgs(a,b*inv(e,n)%n,n);return ~V?V+k:V;}// 有解返回 V+k
```

5.6　线性同余不等式

```
1  // Find the minimal non-negtive solutions for l ≤ d · x mod m ≤ r
2  // 0 ≤ d,l,r < m;l ≤ r,O(log n)
3  LL cal(LL m, LL d, LL l, LL r) {
4      if (l==0) return 0; if (d==0) return MXL; // 无解
5      if (d * 2 > m) return cal(m, m - d, m - r, m - l);
6      if ((l - 1) / d < r / d) return (l - 1) / d + 1;
7      LL k = cal(d, (-m % d + d) % d, l % d, r % d);
8      return k==MXL ? MXL : (k*m + l - 1)/d+1;}// 无解 2
9  // return all x satisfying l1<=x<=r1 and l2<=(x*mul+add)%LIM<=r2
```

```
10  // here LIM = 2^32 so we use UI instead of "%".
11  // O(log p + #solutions)
12  struct Jump { UI val, step;
13      Jump(UI val, UI step) : val(val), step(step) { }
14      Jump operator + (const Jump & b) const {
15          return Jump(val + b.val, step + b.step); }
16      Jump operator - (const Jump & b) const {
17          return Jump(val - b.val, step + b.step); }};
18  inline Jump operator * (UI x, const Jump & a) {
19      return Jump(x * a.val, x * a.step); }
20  vector<UI> solve(UI l1, UI r1, UI l2, UI r2, pair<UI,UI> muladd) {
21      UI mul = muladd.first, add = muladd.second, w = r2 - l2;
22      Jump up(mul, 1), dn(-mul, 1); UI s(l1 * mul + add);
23      Jump lo(r2 - s, 0), hi(s - l2, 0);
24      function<void(Jump&, Jump&)> sub=[&](Jump& a, Jump& b){
25          if (a.val > w) {
26              UI t(((LL)a.val-max(0LL, w+1LL-b.val)) / b.val);
27              a = a - t * b; } };
28      sub(lo, up), sub(hi, dn);
29      while (up.val > w || dn.val > w) {
30          sub(up, dn); sub(lo, up);
31          sub(dn, up); sub(hi, dn); }
32      assert(up.val + dn.val > w); vector<UI> res;
33      Jump bg(s + mul * min(lo.step, hi.step), min(lo.step, hi.step));
34      while (bg.step <= r1 - l1) {
35          if (l2 <= bg.val && bg.val <= r2)
36              res.push_back(bg.step + l1);
37          if (l2 <= bg.val-dn.val && bg.val-dn.val <= r2) {
38              bg = bg - dn;
39          } else bg = bg + up; }
40      return res; }
```

5.7 Miller Rabin And Pollard Rho 启发式分解

```
1   // Miller Rabin
2   typedef long long LL;
3   bool millerrabin(LL p){
4       if (p==2) return 1;
5       if (p<2 || p%2==0 || p == 46856248255981ll) return 0;
6       LL t = p-1, s = 0;
7       while (t%2==0)
8           s++ , t >>= 1;
9       int prime[]={2,3,7,61,24251};
10      for (int i=0; i<5 && prime[i]<p; i++){
11          LL a = prime[i], b = powmod(a , t , p);// [!] this powmod may calculate 1e16 * 1e16 !!!!
12          for (int j=1; j<=s; j++){
13              LL k = b * b % p;
14              if (k==1 && b!=1 && b!=p-1) return 0;
15              b = k;
16          }
17          if (b!=1) return 0;
18      }
19      return 1;
20  }
21  //Pollard Rho
22  const LL threshold = 13E9;
23  LL factorize (LL number, LL seed) {
24      LL x = rand() % (number - 1) + 1, y = x;
25      for (int head = 1, tail = 2; ; ) {
26          x = mul_mod (x, x, number); x = (x + seed) % number;
27          if (x == y) return number;
28          LL answer = gcd (abs (x - y), number);
29          if (answer > 1 && answer < number) return answer;
30          if (++head == tail) { y = x; tail <<= 1; } } }
31  void search (LL number, std::vector<LL> &divisor) {
32      if (number <= 1) return;
33      if (is_prime.solve (number)) divisor.push_back (number);
34      else {
35          LL factor = number;
36          for (; factor >= number; factor = factorize (number, rand () % (number - 1) + 1));
37          search (number / factor, divisor);
38          search (factor, divisor); } }
```

5.8 杜教筛

```cpp
LL P(LL n){// 求欧拉函数前缀和
    if(n<M)return phi[n];//M=n^(2/3),phi[n] 为预处理前缀和
    LL x=1ll*n*(n+1)/2,i=2,la;
    for(;i<=n;i=la+1)la=n/(n/i),x-=P(n/i)*(la-i+1);
    return x;}
LL U(LL n){// 求莫比乌斯函数前缀和
    if(n<M)return u[n];// 预处理的前缀和
    LL x=1,i=2,la;
    for(;i<=n;i=la+1)la=n/(n/i),x-=U(n/i)*(la-i+1);
    return x;}
```

5.9 Extended Eratosthenes Sieve 扩展埃氏筛

```cpp
ll num[N],c;
ll sum1[N],sum2[N],g1[N],g2[N],g[N];
int id1[N],id2[N];
int H(ll x){return x<N?id1[x]:id2[n/x];}

ll f(ll p,int e){return p^e;}
ll S(int x,int j){
    ll ret=(g[x] - (sum2[j-1]-sum1[j-1]+(j>1?2:0)) +M)%M;
    for (int k=j; 1ll*pr[k]*pr[k]<=num[x]&&k<=cnt; k++){
        ll pke=pr[k];//pke: Pk*e
        for (int e=1; pke*pr[k]<=num[x]; e++,pke*=pr[k])
            (ret+=S(H(num[x]/pke),k+1)*f(pr[k],e)%M + f(pr[k],e+1)) %=M;
    }
    return ret;
}
int min_25(){
    ll i,j;
    for (i=1; i<=n; i=n/(n/i)+1)
        num[++c]=n/i, n/i<N?id1[n/i]=c:id2[n/(n/i)]=c;//unordered_map太慢,改用两个数组id1,id2用于哈希
    for (i=1; i<=cnt; i++)
        sum1[i]=sum1[i-1]+1, sum2[i]=(sum2[i-1]+pr[i])%M;
    for (i=1; i<=c; i++)//g1(x)=1, g2(x)=x
        g1[i]=num[i]-1,
        g2[i]=(num[i]&1)? (num[i]-1)/2%M*((num[i]+2)%M)%M : (num[i]+2)/2%M*((num[i]-1)%M)%M;
    for (j=1; j<=cnt; j++)
        for (i=1; 1ll*pr[j]*pr[j]<=num[i]&&i<=c; i++)
            g1[i]=(g1[i] - g1[H(num[i]/pr[j])] + sum1[j-1] +M)%M,
            g2[i]=(g2[i] - pr[j]*g2[H(num[i]/pr[j])]%M + pr[j]*sum2[j-1]%M +M)%M;
    for (i=1; i<=c; i++)
        g[i]=(g2[i]-g1[i]+(num[i]>=2?2:0) +M)%M;//f(p)=p-1, exception:f(2)=3
    return S(1,1)+1; //f(1)=1
}
```

5.10 阶和原根

```cpp
bool ok(LL g,LL p){// 验证 g 是否是 p 的原根
    for(int i=1;i<=t;i++)if(Pw(g,(p-1)/q[i],p)==1)return 0;
    return 1;
}LL primitive_root(LL p){// 求 p 的原根
    LL i,n=p-1,g=1;t=0;
    for(i=2;i*i<=n;i++)if(n%i==0)for(q[++t]=i;n%i==0;n/=i);
    if(n!=1)q[++t]=n;
    for(;;g++)if(ok(g,p))return g;}
```
//当$gcd(a,m)=1$时，使$a^x \equiv 1(mod m)$成立最小正整数x称为a对于模m的阶，记$ord_m(a)$.
//阶的性质：$a^n \equiv 1(mod\ m)$充要条件是$ord_m(a)|n$，可推出$ord_m(a)|\psi(m)$.
//当$ord_m(g)=\psi(m)$时，则称g是模m的一个原根，则$g^0,g^1,...,g^{\psi(m)-1}$覆盖了$m$以内所有与$m$互质的数.
//并不是所有数都存在原根，模意义下存在原根充要条件：$m=2,4,p^k,2p^k$. 其中为p奇素数，k为正整数.

//求阶的做法:对令t = p-1,对t的所有质因子尝试整除,直到a^t模p不等于1为止. O(log^2 n).

5.11 生成函数

在组合数学中 生成函数是一种很重要求解工具

5.11.1 普通型生成函数

求组合计数问题(集合划分问题)时,应该使用普通型生成函数.组合计数问题:把若干个"相同的球"放入若干个"相同盒子"里类型的问题,应该使用普通型生成函数.例如:

有一元硬币,二元硬币,三元硬币无穷个,求面值和等于n的组合方案数.

解:$f(x)=(1+x+x^2+\ldots)(1+x^2+x^4+\ldots)(1+x^3+x^6+\ldots)$, 最后$x^n$的系数就是答案.

一般地,这类问题的生成函数是一个等比数列,所以求和的结果是一个分式.例如$f(x)=1+x+x^2+\cdots=\frac{1}{1-x}$.

5.11.2　指数型生成函数

求多重排列问题时,应该使用指数型生成函数.多重排列问题:有m种物品,每种物品有无穷多个,要求用这些物品组成长度为n的排列,求方案数. 分析:在不考虑同种物品的情况下,总方案数显然是$n!$个.而假设某种物品分配方案中第i种物品有a_i个,则总方案数要除以$a_i!$.所以构造函数:

$$f(x) = (1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots)^m,$$
$$ans = c_n * n!.$$

例2:把若t个奇数和s个偶数组成长度为n的排列,要求每个偶数个数必须是偶数,奇数个数必须是奇数,求方案数.

$$f(x) = (\frac{1}{1!}x + \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + \dots)^t(1 + \frac{1}{2!}x^2 + \frac{1}{4!}x^4 + \frac{1}{6!}x^6 + \dots)^s$$

一般地,这种函数都是一个指数函数的泰勒展开式,所以求和结果是一个指数函数.例如$f(x) = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \dots = e^x$.

拓展:如果物品个数不同的情况下贡献(权值)是不同的,例如1个一元硬币的权值是w_1 2个是w_2,....那么可以进一步修改为$f(x) = w_0 + w_1x + w_2x^2 + \dots$.

5.12　反演

5.12.1　二项式反演

形式1:若$G(n) = \sum_{i=0}^{n} C_n^i F(i)$,则有$F(n) = \sum_{i=0}^{n} (-1)^{n-i} C_n^i G(i)$.
形式2:若$G(k) = \sum_{i=k}^{n} C_i^k F(i)$,则有$F(k) = \sum_{i=k}^{n} (-1)^{i-k} C_i^k G(i)$.
形式3:若$G(n) = \sum_{i=0}^{n} (-1)^i C_n^i F(i)$,则有$F(n) = \sum_{i=0}^{n} (-1)^i C_n^i G(i)$.

5.12.2　单位根反演

求$\sum_{i=0}^{\lfloor \frac{n}{k} \rfloor} C_n^{ik}$.
引理:$\frac{1}{k} \sum_{i=0}^{k-1} \omega_k^{in} = [k \mid n]$.
反演:$Ans = \sum_{i=0}^{n} C_n^i [k \mid i]$
$= \sum_{i=0}^{n} C_n^i (\frac{1}{k} \sum_{j=0}^{k-1} \omega_k^{ij})$
$= \frac{1}{k} \sum_{i=0}^{n} C_n^i \sum_{j=0}^{k-1} \omega_k^{ij}$
$= \frac{1}{k} \sum_{j=0}^{k-1} (\sum_{i=0}^{n} C_n^i (\omega_k^j)^i)$
$= \frac{1}{k} \sum_{j=0}^{k-1} (1 + \omega_k^j)^n$.
另,如果要求的是$[n\%k = t]$,其实就是$[k \mid (n - t)]$.同理推式子即可.

5.13　CDQ+NTT

```cpp
#include <bits/stdc++.h>
#define mod 998244353
#define N 133333
using namespace std;
int n, f[N], tmp1[N], tmp2[N], len, inv_len, rev[N], a[N], b[N];
int qp(int a, int b){
    int c = 1;
    for(; b; b >>= 1, a = 1ll * a * a % mod)
        if(b & 1) c = 1ll * c * a % mod;
    return c;
}
void UPD(int &a, int b){a = (a + b) % mod;}
void ntt(int *a, int o){
    register int i, j, k, dan, now, t;
    for(i = 0; i < len; i++)if(i < rev[i]) swap(a[i], a[rev[i]]);
    for(k = 2; k <= len; k <<= 1){
        dan = qp(3, o == 1 ? (mod - 1) / k : (mod - 1 - (mod - 1) / k));
        for(i = 0; i < len; i += k){
            now = 1;
            for(j = 0; j < (k >> 1); j++, now = 1ll * now * dan % mod){
                t = 1ll * a[i + j + (k >> 1)] * now % mod;
                a[i + j + (k >> 1)] = (a[i + j] - t + mod) % mod;
                a[i + j] = (a[i + j] + t) % mod;
            }
        }
    }
    if(o == -1){
        for(i = 0; i < len; i++)
            a[i] = 1ll * a[i] * inv_len % mod;
    }
}
void mul(int *a, int *b, int l1, int l2){
    for(len = 1; len <= l1 + l2; len <<= 1);
    inv_len = qp(len, mod - 2);
    for(int i = 0; i < len; i++){
        if(i & 1) rev[i] = (rev[i >> 1] >> 1) | (len >> 1);
        else rev[i] = rev[i >> 1] >> 1;
    }
    for(int i = l1; i < len; i++) a[i] = 0;
    for(int i = l2; i < len; i++) b[i] = 0;
    ntt(a, 1); ntt(b, 1);
    for(int i = 0; i < len; i++) a[i] = 1ll * a[i] * b[i] % mod;
    ntt(a, -1);
```

```
44  }
45  void cdq(int l, int r){
46      if(l == r) return ;
47      printf("%d %d\n", l, r);
48      int mid = (l + r) >> 1;
49      cdq(l, mid);
50      for(int i = l; i <= mid; i++){
51          tmp1[i - l] = f[i];
52          tmp2[i - l] = f[i];
53      }
54      mul(tmp1, tmp2, mid - l + 1, mid - l + 1);
55      for(int i = max(l << 1, mid + 1); i <= r; i++)
56          UPD(f[i], tmp1[i - (l << 1)]);
57
58      for(int i = l; i <= mid; i++) tmp1[i - l] = f[i];
59      for(int i = 0; i <= min(l - 1, r - l); i++)
60          tmp2[i] = f[i];
61      mul(tmp1, tmp2, mid - l + 1, min(l - 1, r - l) + 1);
62      for(int i = mid + 1; i - l < len && i <= r; i++)
63          UPD(f[i], 2ll * tmp1[i - l] % mod);
64      //printf("l = %d  r = %d   %d\n", l, r, f[5]);
65      cdq(mid + 1, r);
66  }
67  int main(){
68      f[1] = 1;
69      scanf("%d", &n);
70      cdq(1, n + 1);
71      for(int i = 1; i <= n + 1; i++)
72          printf("%d ", f[i]);
73      puts("");
74      return 0;
75  }
```

5.14   FFT NTT FWT 多项式操作

```
1   // double 精度对10^9 + 7 取模最多可以做到2^20
2   // FFT(Reverse(FFT(a,N)),N)=Na, Reverse是a_0不变,1到N-1反过来
3   const int MOD = 1e9 + 7; const double PI = acos(-1);
4   typedef complex<double> Complex;
5   const int MAXN = 262144, L = 15, MASK = (1<<L) - 1;
6   Complex w[MAXN];
7   void FFTInit(int N) {
8       for (int i = 0; i < N; ++i)
9           w[i] = Complex(cos(2*i*PI/N), sin(2*i*PI/N));}
10  void FFT(Complex p[], int n) { FFTInit(N);
11      for (int i = 1, j = 0; i < n - 1; ++i) {
12          for (int s = n; j ^= s >>= 1, ~j & s;);
13          if (i < j) swap(p[i], p[j]); }
14      for (int d = 0; (1 << d) < n; ++d) {
15          int m = 1 << d, m2 = m * 2, rm = n >> (d+1);
16          for (int i = 0; i < n; i += m2) {
17              for (int j = 0; j < m; ++j) {
18                  Complex &p1 = p[i+j+m], &p2 = p[i+j];
19                  Complex t = w[rm * j] * p1;
20                  p1 = p2 - t, p2 = p2 + t; } } } }
21  void FFT_inv(Complex p[],int n) {
22      FFT(p,n); reverse(p + 1, p + n);
23      for (int i = 0; i < n; ++ i) p[i] /= n; }
24  Complex A[MAXN], B[MAXN], C[MAXN], D[MAXN];
25  void mul(int *a, int *b, int N) {
26      for (int i = 0; i < N; ++i) {
27          A[i] = Complex(a[i] >> L, a[i] & MASK);
28          B[i] = Complex(b[i] >> L, b[i] & MASK); }
29      FFT(A, N), FFT(B, N);
30      for (int i = 0; i < N; ++i) { int j = (N - i) % N;
31          Complex da = (A[i]-conj(A[j]))*Complex(0, -0.5),
32                  db = (A[i]+conj(A[j]))*Complex(0.5, 0),
33                  dc = (B[i]-conj(B[j]))*Complex(0, -0.5),
34                  dd = (B[i]+conj(B[j]))*Complex(0.5, 0);
35          C[j] = da * dd + da * dc * Complex(0, 1);
36          D[j] = db * dd + db * dc * Complex(0, 1); }
37      FFT(C, N), FFT(D, N);
38      for (int i = 0; i < N; ++i) {
39          LL  da = (LL)(C[i].imag() / N + 0.5) % MOD,
```

```
40          db = (LL)(C[i].real() / N + 0.5) % MOD,
41          dc = (LL)(D[i].imag() / N + 0.5) % MOD,
42          dd = (LL)(D[i].real() / N + 0.5) % MOD;
43      a[i]=((dd << (L*2)) + ((db+dc) << L) + da) %MOD;}}
44 // 4179340454199820289LL (4e18) 原根=3 两倍不会爆 LL
45 // 2013265921 原根=31 两倍平方不会爆 LL
46 // 998244353 原根=3 // 1004535809 原根=3 // 469762049 原根=3
47 void NTT(int *a,int n,int f=1){
48      int i,j,k,m,u,v,w,wm;
49      for(i=n>>1,j=1,k;j<n-1;j++){
50          if(i>j)swap(a[i],a[j]);
51          for(k=n>>1;k<=i;k>>=1)i^=k;i^=k;
52      }for(m=2;m<=n;m<<=1)
53          for(i=0,wm=Pw(G,f==1?(M-1)/m:(M-1)/m*(m-1),M);i<n;i+=m)
54              for(j=i,w=1;j<i+(m>>1);j++){
55                  u=a[j],v=1ll*w*a[j+(m>>1)]%M;
56                  if((a[j]=u+v)>=M)a[j]-=M;
57                  if((a[j+(m>>1)]=u-v)<0) a[j+(m>>1)]+=M;
58                  w=1ll*w*wm%M;
59              }
60      if(f==-1)for(w=Pw(n,M-2,M),i=0;i<n;i++)a[i]=1ll*a[i]*w%M;
61 }
62 void FWT(int w){//w=1为正运算, w=0为逆运算
63      int i,j,k,x,y;
64      for(i=1;i<N;i*=2)for(j=0;j<N;j+=i*2){
65          for(k=0;k<i;k++) {
66          x=a[j+k],y=a[i+j+k];
67          if(w){
68              //xor a[j+k]=x+y a[i+j+k]=x-y
69              //and a[j+k]=x+y
70              //or  a[i+j+k]=x+y
71          }else{
72              //xor a[j+k]=(x+y)/2 a[i+j+k]=(x-y)/2
73              //and a[j+k]=x-y
74              //or  a[i+j+k]=y-x
75      } } } }
```

5.14.1 多项式牛顿法

已知函数$G(x)$, 求多项式$F(x) \mod x^n$满足方程$G(F(x)) \equiv 0 \mod x^n$.

当$n = 1$时, 有$G(F(x)) \equiv 0 \mod x$, 根据实际情况 (逆元, 二次剩余) 求解. 假设已经求出了$G(F_0(x)) \equiv 0 \mod x^n$, 考虑扩展到$\mod x^{2n}$下: 将$G(F(x))$在$F_0(x)$点泰勒展开, 有

$$G(F(x)) = G(F_0(x)) + \frac{G'(F_0(x))}{1!}(F(x) - F_0(x)) + \cdots$$

因为$F(x)$和$F_0(x)$的最后n项相同, 所以$(F(x) - F_0(x))^2$的最低的非0项次数大于$2n$, 经过推导可以得到

$$F(x) \equiv F_0(x) - \frac{G(F_0(x))}{G'(F_0(x))} \mod x^{2n}$$

应用 (以下复杂度均为$O(n \log n)$):

多项式求逆 (给定$A(x)$, 求$A(x)B(x) \equiv 1 \mod x^n$): 构造方程$A(x)B(x) - 1 \equiv 0 \mod x^n$, 初始解$G_{invA}(B(x)) \equiv A[0]^{-1} \mod x$, 递推式$F(x) \equiv 2F_0(x) - A(x)F_0^2(x) \mod x^{2n}$

多项式开方 (给定$A(x)$, 求$B^2(x) \equiv A(x) \mod x^n$): 初始解$G_{sqrtA}(B(x)) \equiv \sqrt{A[0]} \mod x$, 递推式$F(x) \equiv \frac{F_0^2(x)+A(x)}{2F_0(x)} \mod x^{2n}$

多项式对数 (给定常数项为1的$A(x)$, $B(x) \equiv \ln A(x)$): 对x求导得$(\ln A(x))' = \frac{A'(x)}{A(x)}$, 使用多项式求逆, 再积分回去$\ln A(x) \equiv \int \frac{A'(x)}{A(x)}$

多项式指数 (给定常数项为0 的$A(x)$, 求$B(x) \equiv e^{A(X)}$): 初始解$G_{expA}(B(x)) \equiv 1$, 递推式$F(x) \equiv F_0(x)(1 - \ln F_0(x) + A(x))$

多项式任意幂次 (给定$A(x)$, 求$B(x) \equiv A^k(x), k \in Q$): $A^k(x) \equiv e^{k \ln A(x)}$

```
1 void Inv(int*A,int*B,int n){ //注意数组大小2n
2 //多项式求逆, B = A⁻¹,n需为2的幂次
3     static int C[N];B[0]=Pw(A[0],M-2,M);B[1]=0; //n=1时B[0] = A[0]⁻¹
4     for(int m=2,i;m<=n;m<<=1){//递归转递推
5         for(i=0;i<m;i++)C[i]=A[i];
6         for(i=m;i<2*m;i++)C[i]=B[i]=0; //在模xᵐ意义下超过m次均为0
7         NTT(C,m*2);NTT(B,m*2);
```

```
 8          //g(x) = g_0(x)(2 - f(x)g_0(x))(  mod x^n)
 9      for(i=0;i<m*2;i++)
10          B[i]=1ll*B[i]*(2-1ll*B[i]*C[i]%M+M)%M;
11      NTT(B,m*2,-1);for(i=m;i<m*2;i++)B[i]=0;}
12 }
13 void Sqrt(int*A,int*B,int n){//多项式开根，B=sqrt(A)，n为2的幂次
14     static int D[N],IB[N];
15     B[0]=1;B[1]=0;//n=1时根据题意或二次剩余求解
16     int I2=Pw(2,M-2,M),m,i;
17     for(m=2;m<=n;m<<=1){//递归转递推
18         for(i=0;i<m;i++)D[i]=A[i];
19         for(i=m;i<2*m;i++)D[i]=B[i]=0;
20         NTT(D,m*2);Inv(B,IB,m);NTT(IB,m*2);NTT(B,m*2);
21         for(i=0;i<m*2;i++)
22             B[i]=(1ll*B[i]*I2+1ll*I2*D[i]%M*IB[i])%M;
23         NTT(B,m*2,-1);for(i=m;i<m*2;i++)B[i]=0;
24     }
25 }
26 // 多项式除法：给定n次多项式A(x)和m ≤ n次多项式B(x)，求出D(x)，R(x)满足A(x) = D(x)B(x) + R(x)，并且 degD ≤ n − m，
   ↪ degR < m，复杂度O(n log n)，常用于线性递推将2k项系数拍回k项时的优化：本质是将2k项的多项式除以k项零化多项式得到的余数
27 void Div(int *a, int n, int *b, int m, int *d, int *r) {
28     // 注意这里n和m为多项式长度，注意需要4倍空间
29     static int A[MAXN], B[MAXN]; while (!b[m - 1]) m --;
30     int p = 1, t = n - m + 1; while (p < t << 1) p <<= 1;
31     fill(A, A+p, 0); reverse_copy(b, b+m, A); Inv(A, B, p);
32     fill(B+t, B+p, 0); NTT(B, p); reverse_copy(a, a+n, A);
33     fill(A + t, A + p, 0); NTT(A, p);
34     for (int i = 0; i < p; ++i) A[i] = 1LL*A[i]*B[i] % M;
35     NTT(A, p, -1); reverse(A,A+t); copy(A,A+t,d); //lenD<=t
36     for (p = 1; p < n; p <<= 1);
37     fill(A + t, A + p, 0); NTT(A, p); copy(b, b + m, B);
38     fill(B + m, B + p, 0); NTT(B, p);
39     for (int i = 0; i < p; ++i) A[i] = 1LL*A[i]*B[i] % M;
40     NTT(A, p, -1);
41     for (int i = 0; i < m; ++i) r[i] = (a[i]-A[i]+M) % M;
42     fill(r+m, r+p, 0); assert(r[m-1] == 0); } //lenR < m
```

5.14.2 多点求值 与 快速插值

多点求值: 给出$F(x)$和$x_1, \cdots, x_n$，求$F(x_1), \cdots, F(x_n)$.

考虑分治，设$L(x) = \prod_{i=1}^{\lfloor n/2 \rfloor}(x - x_i)$，$R(x) = \prod_{i=\lfloor n/2 \rfloor+1}^{n}(x - x_i)$，那么对于$1 \le i \le \lfloor n/2 \rfloor$有$F(x_i) = (F \mod L)(x_i)$，对于$\lfloor n/2 \rfloor < i \le n$有$F(x_i) = (F \mod R)(x_i)$. 复杂度$O(n \log^2 n)$.

快速插值: 给出n个x_i与y_i，求一个$n - 1$次多项式满足$F(x_i) = y_i$.

考虑拉格朗日插值: $F(x) = \sum_{i=1}^{n} \frac{\prod_{i \ne j}(x - x_j)}{\prod_{i \ne j}(x_i - x_j)} y_i$.

对每个i先求出$\prod_{i \ne j}(x_i - x_j)$. 设$M(x) = \prod_{i=1}^{n}(x - x_i)$，那么想要的是$\frac{M(x)}{x - x_i}$. 取$x = x_i$时，上下都为$0$，使用洛必达法则，则原式化为$M'(x)$. 使用分治算出$M(x)$，使用多点求值算出每个$\prod_{i \ne j}(x_i - x_j) = M'(x_i)$.

设$\frac{y_i}{\prod_{i \ne j}(x_i - x_j)} = v_i$，现在要求出$\sum_{i=1}^{n} v_i \prod_{i \ne j}(x - x_j)$. 使用分治: 设$L(x) = \prod_{i=1}^{\lfloor n/2 \rfloor}(x - x_i)$，$R(x) = \prod_{i=\lfloor n/2 \rfloor+1}^{n}(x - x_i)$，则原式化为:
$\left(\sum_{i=1}^{\lfloor n/2 \rfloor} v_i \prod_{i \ne j, j \le \lfloor n/2 \rfloor}(x - x_j) \right) R(x) + \left(\sum_{i=\lfloor n/2 \rfloor+1}^{n} v_i \prod_{i \ne j, j > \lfloor n/2 \rfloor}(x - x_j) \right) L(x)$，递归计算. 复杂度$O(n \log^2 n)$.

5.15 多项式插值

```
 1 // 拉格朗日插值 n 为点数，x 为要求的 f(x)，已知 f(X[i])=Y[i]
 2 D lagrange(int n,D x,D X[],D Y[]){
 3     int i,j;D ans,v;fr(i,n){
 4         v=1;fr(j,n)if(i!=j)v*=(x-X[j])/(X[i]-X[j]);
 5         ans+=v*Y[i];}
 6     return ans;}
 7 // 牛顿插值，给出 f(X[i])=Y[i]),i=0...n-1
 8 void pre(){//O(n^2) 预处理
 9     fr(i,0,n-1)f[i][0]=Y[i];
10     fr(i,1,n-1)fr(j,1,i) f[i][j]=(f[i][j-1]-f[i-1][j-1])/(X[i]-X[i-j]); }
11 D getfx(D x){//O(n) 询问单点值
12     D an=f[0][0],v=1;
13     fr(i,1,n-1)v*=(x-X[i-1]),an+=f[i][i]*v;
14     return an;}
```

5.16 快速插值

```
1 /*
2  * 多项式相关算法模板集合
3  */
4 #include <bits/stdc++.h>
```

```cpp
using namespace std;
typedef long long ll;
// 2281701377是一个原根为3的指数,平方刚好不会爆long long
// 另外一个常用的998244353 = 119ll * (1 << 23) + 1
// 1004535809=479⊠221+1 加起来刚好不会爆 int 也不错
const ll mod_v = 17ll * (1 << 27) + 1;
const int MAXL = 18, N = 1 << MAXL, M = 1 << MAXL;  // MAXL最大取mov_v的2次幂
struct polynomial {
    ll coef[N]; // size设为多项式系数个数的两倍
    ll a[N], b[N], d[N], r[N];
    ll xcoor[N], ycoor[N], v[N], mcoef[N];   // size设为点个数的两倍
    vector<vector<ll> > poly_divisor;

    static ll mod_pow(ll x,ll n,ll m) {
        ll ans = 1;
        while(n > 0){
            if(n & 1)
                ans = ans*x%m;
            x = x*x%m;
            n >>= 1;
        }
        return ans;
    }

    struct FastNumberTheoreticTransform  {
        ll omega[N], omegaInverse[N];
        int range;

        // 初始化频率
        void init (const int& n) {
            range = n;
            ll base = mod_pow(3, (mod_v - 1) / n, mod_v);
            ll inv_base = mod_pow(base, mod_v - 2, mod_v);
            omega[0] = omegaInverse[0] = 1;
            for (int i = 1; i < n; ++i) {
                omega[i] = omega[i - 1] * base % mod_v;
                omegaInverse[i] = omegaInverse[i - 1] * inv_base % mod_v;
            }
        }

        // Cooley-Tukey算法:O(n*logn)
        void transform (ll *a, const ll *omega, const int &n) {
            for (int i = 0, j = 0; i < n; ++i) {
                if (i > j) std::swap (a[i], a[j]);
                for(int l = n >> 1; ( j ^= l ) < l; l >>= 1);
             }

            for (int l = 2; l <= n; l <<= 1) {
                int m = l / 2;
                for (ll *p = a; p != a + n; p += l) {
                    for (int i = 0; i < m; ++i) {
                        ll t = omega[range / l * i] * p[m + i] % mod_v;
                        p[m + i] = (p[i] - t + mod_v) % mod_v;
                        p[i] = (p[i] + t) % mod_v;
                    }
                }
            }
        }


        // 时域转频域
        void dft (ll *a, const int& n) {
            transform(a, omega, n);
        }

        // 频域转时域
        void idft (ll *a, const int& n) {
            transform(a, omegaInverse, n);
            for (int i = 0; i < n; ++i) a[i] = a[i] * mod_pow(n, mod_v - 2, mod_v) % mod_v;
        }
    } fnt ;

    // 与模比较转换值
```

```
78     ll mod_trans(ll v) {
79         return abs(v) <= mod_v / 2 ? v : (v < 0 ? v + mod_v : v - mod_v);
80     }
81
82     // 分治NTT
83     void binary_subpoly(int l, int r, int idx) {
84         if (l == r - 1) {
85             poly_divisor[idx].push_back(-xcoor[l]);
86             poly_divisor[idx].push_back(1);
87         } else {
88             int lidx = (idx << 1) + 1, ridx = lidx + 1;
89             binary_subpoly(l, (l + r) / 2, lidx);
90             binary_subpoly((l + r) / 2, r, ridx);
91             int t = poly_divisor[lidx].size() + poly_divisor[ridx].size() - 1;
92             int p = 1;
93             while(p < t) p <<= 1;
94             copy(poly_divisor[lidx].begin(), poly_divisor[lidx].end(), a);
95             fill(a + poly_divisor[lidx].size(), a + p, 0);
96             copy(poly_divisor[ridx].begin(), poly_divisor[ridx].end(), b);
97             fill(b + poly_divisor[ridx].size(), b + p, 0);
98             fnt.dft(a, p);
99             fnt.dft(b, p);
100            for (int i = 0; i < p; i++)  a[i] *= b[i];
101            fnt.idft(a, p);
102            for (int i = 0; i < t; i++) poly_divisor[idx].push_back(mod_trans(a[i]));
103        }
104    }
105
106    // 模x^deg,a为要求逆元的多项式系数,结果存放在b[0~deg]中
107    // T(deg) = T(deg/2) + deg*log(deg),复杂度O(deg*log(deg))
108    void polynomial_inverse(int deg, ll* a, ll* b, ll* tmp) {
109        if(deg == 1) {
110            b[0] = mod_pow(a[0], mod_v - 2, mod_v);
111        } else {
112            polynomial_inverse((deg + 1) >> 1, a, b, tmp);
113            int p = 1;
114            while(p < (deg << 1) - 1) p <<= 1;
115            copy(a, a + deg, tmp);
116            fill(tmp + deg, tmp + p, 0);
117            fill(b + ((deg + 1) >> 1), b + p, 0);
118            //fnt.init(p);
119            fnt.dft(tmp, p);
120            fnt.dft(b, p);
121            for(int i = 0; i != p; ++i) {
122                b[i] = (2 - tmp[i] * b[i] % mod_v) * b[i] % mod_v;
123                if(b[i] < 0) b[i] += mod_v;
124            }
125            fnt.idft(b, p);
126            fill(b + deg, b + p, 0);
127        }
128    }
129
130    // A = D*B + R,A为n项n-1次幂,B为m项m-1次幂,D为n-m+1项n-m次幂,R为m-1项m-2次幂
131    // 要求a,b中系数以低次到多次顺序排列
132    // n >= m,复杂度O(n*logn)，n < m,复杂度O(n)
133    int polynomial_division(int n, int m, ll *A, ll *B, ll *D, ll *R) {
134        if (n < m) {
135            copy(A, A + n, R);
136            return n;
137        } else {
138            static ll A0[N], B0[N], tmp[N]; //数组太大会爆栈,添加到全局区
139
140            int p = 1, t = n - m + 1;
141            while(p < (t << 1) - 1) p <<= 1;
142
143            fill(A0, A0 + p, 0);
144            reverse_copy(B, B + m, A0);
145            polynomial_inverse(t, A0, B0, tmp);
146            fill(B0 + t, B0 + p, 0);
147            fnt.dft(B0, p);
148
149            reverse_copy(A, A + n, A0);
150            fill(A0 + t, A0 + p, 0);
```

```cpp
            fnt.dft(A0, p);

            for(int i = 0; i != p; ++i)
                A0[i] = A0[i] * B0[i] % mod_v;
            fnt.idft(A0, p);
            reverse(A0, A0 + t);
            copy(A0, A0 + t, D);

            for(p = 1; p < n; p <<= 1);
            fill(A0 + t, A0 + p, 0);
            fnt.dft(A0, p);
            copy(B, B + m, B0);
            fill(B0 + m, B0 + p, 0);
            fnt.dft(B0, p);
            for(int i = 0; i != p; ++i)
                A0[i] = A0[i] * B0[i] % mod_v;
            fnt.idft(A0, p);
            for(int i = 0; i != m - 1; ++i)
                R[i] = ((A[i] - A0[i]) % mod_v + mod_v) % mod_v;
            //fill(R + m - 1, R + p, 0);
            return m - 1;
        }
    }

    // 多项式的点值计算
    // l和r为存储要求的点数组的左右边界(左闭右开)，idx为除数多项式索引(初始0)
    // polycoef为用于计算点的多项式系数,num为其系数个数
    // 设多项式项数x=num,点数y=r-1,n=max(x,y),复杂度O(n(logn)^2)
    // 结果存储于v[]数组中
    void polynomial_calculator(int l, int r, int idx, int num, ll *polycoef) {
        int mid = (l + r) / 2;
        int lidx = (idx << 1) + 1, ridx = lidx + 1;
        int lsize = poly_divisor[lidx].size(), rsize = poly_divisor[ridx].size();
        ll *lmod_poly = new ll[lsize - 1], *rmod_poly = new ll[rsize - 1];
        copy(poly_divisor[lidx].begin(), poly_divisor[lidx].end(), a);
        copy(poly_divisor[ridx].begin(), poly_divisor[ridx].end(), b);
        int lplen = polynomial_division(num, lsize, polycoef, a, d, lmod_poly);
        int rplen = polynomial_division(num, rsize, polycoef, b, d, rmod_poly);
        if (l == mid - 1) {
            v[l] = mod_trans(lmod_poly[0]);
        } else {
            polynomial_calculator(l, (l + r) / 2, lidx, lplen, lmod_poly);
        }
        if (r == mid + 1) {
            v[(l + r) / 2] = mod_trans(rmod_poly[0]);
        } else {
            polynomial_calculator((l + r) / 2, r, ridx, rplen, rmod_poly);
        }
        delete []lmod_poly;
        delete []rmod_poly;
    }

    // 拉格朗日插值▢二分▢快速数论变换
    // l和r为存储要求的点数组的左右边界(左闭右开)，idx为由点二分构造出多项式的索引(初始0)
    // polycoef为插值得到的多项式结果
    // 设点个数为n,复杂度O(n*(logn)^2),结果polycoef为n-1次多项式
    void polynomial_interpolate(int l, int r, int idx, ll *polycoef) {
        if (l == r - 1) {
            polycoef[0] = ycoor[l] * mod_pow(v[l], mod_v - 2, mod_v) % mod_v;
        } else {
            int mid = (l + r) >> 1;
            int lidx = (idx << 1) + 1, ridx = lidx + 1;
            int sz = poly_divisor[idx].size() - 1;
            int lsize = poly_divisor[lidx].size(), rsize = poly_divisor[ridx].size();
            int p = 1;
            while (p < sz) p <<= 1;
            ll *leftpoly = new ll[p], *rightpoly = new ll[p];
            polynomial_interpolate(l, mid, lidx, leftpoly);
            polynomial_interpolate(mid, r, ridx, rightpoly);
            copy(poly_divisor[lidx].begin(), poly_divisor[lidx].end(), a);
            copy(poly_divisor[ridx].begin(), poly_divisor[ridx].end(), b);
            fill(leftpoly + lsize - 1, leftpoly + p, 0);
            fill(rightpoly + rsize - 1, rightpoly + p, 0);
```

```
224            fill(a + lsize, a + p, 0);
225            fill(b + rsize, b + p, 0);
226            fnt.dft(leftpoly, p);
227            fnt.dft(b, p);
228            for (int i = 0; i < p; i++) leftpoly[i] = leftpoly[i] * b[i] % mod_v;
229            fnt.idft(leftpoly, p);
230            fnt.dft(rightpoly, p);
231            fnt.dft(a, p);
232            for (int i = 0; i < p; i++) rightpoly[i] = rightpoly[i] * a[i] % mod_v;
233            fnt.idft(rightpoly, p);
234            for (int i = 0; i < sz; i++) polycoef[i] = mod_trans((leftpoly[i] + rightpoly[i]) % mod_v);
235            delete []leftpoly;
236            delete []rightpoly;
237        }
238    }
239
240    // 初始化点个数到二分子多项式个数
241    // 调用polynomial_calculator和polynomial_interpolate前调用
242    void init(int vnum) {
243        int vnum2 = 1;
244        while (vnum2 < vnum) vnum2 <<= 1;
245        for (int i = 0; i < 2 * vnum2 - 1; i++)  poly_divisor.push_back(vector<ll>());
246    }
247 } poly;
```

5.17 Simplex 单纯形

```
1  double an[N*2];int q[N*2];
2  void pivot(int l,int e){//转轴操作
3     int i,j;a[l][e]=1/a[l][e];b[l]*=a[l][e];
4     fr(i,1,n)if(i!=e)a[l][i]*=a[l][e];
5     fr(i,1,m)if(i!=l&&fabs(a[i][e])>eps){
6         b[i]-=a[i][e]*b[l];
7         fr(j,1,n)if(j!=e)a[i][j]-=a[i][e]*a[l][j];
8         a[i][e]*=-a[l][e];
9     }v+=c[e]*b[l];
10    fr(i,1,n)if(i!=e)c[i]-=c[e]*a[l][i];
11    c[e]*=-a[l][e];swap(q[e],q[l+n]);//记录方案
12 }void pre(){//特殊处理b_i<0的情况，没有这种情况可不写
13    for(;;){
14        dr(i,m,1)if(b[i]<-eps)break;
15        if(!i)break;
16        dr(j,n,1)if(a[i][j]<-eps)break;
17        if(!j)return puts("Infeasible"),0;//无解
18        pivot(i,j);}
19 }//n为变量数，m为约束数，令目标函数$F = \sum c_j x_j$最大，第i条约束为$\sum a_{ij} x_j <= b_i$
20 int main(int n,int m,double c[N],double b[M],double a[M][N]){
21    //输出答案，并输出此时$x_1, x_2, ..., x_n$的值
22    double v,p;int l,e;fr(i,1,n)q[i]=i;
23    for(pre();;pivot(l,e)){//寻找更优解
24        dr(i,n,1)if(c[i]>eps)break;//选择系数大于0的
25        if(!i)break;//找到最优解
26        p=inf;e=i;fr(i,1,m)if(a[i][e]>eps&&p>b[i]/a[i][e])
27            p=b[i]/a[i][e],l=i;
28        if(p==inf)return puts("Unbounded"),0; //找不到可行的基本变量，最优解无穷大
29    }
30    printf("%.8f",v);//输出答案
31    fr(i,n+1,n+m)an[q[i]]=b[i-n];
32    fr(i,1,n)printf("%.8lf ",an[i]);}//输出方案
33 // 标准型: maximize $\mathbf{c^T x}$, subject to $\mathbf{Ax} \le \mathbf{b}$ and $\mathbf{x} \ge \mathbf{0}$
34 // 对偶型: minimize $\mathbf{b^T y}$, subject to $\mathbf{A^T x} \ge \mathbf{c}$ and $\mathbf{y} \ge \mathbf{0}$
```

5.18 线性递推

```
1  // Complexity: init O(n^2log) query O(n^2logk)
2  // Requirement: const LOG const MOD
3  // Example: In: {1, 3} {2, 1} an = 2an-1 + an-2
4  //          Out: calc(3) = 7
5  typedef vector<int> poly;
6  struct LinearRec {
7     int n; poly first, trans; vector<poly> bin;
8  poly add(poly &a, poly &b) {
9     poly res(n * 2 + 1, 0);
```

```
10    // 不要每次新开 vector，可以使用矩阵乘法优化
11    for (int i = 0; i <= n; ++i) {
12       for (int j = 0; j <= n; ++j) {
13          (res[i+j]+=(LL)a[i] * b[j] % MOD) %= MOD;
14    for (int i = 2 * n; i > n; --i) {
15       for (int j = 0; j < n; ++j) {
16          (res[i-1-j]+=(LL)res[i]*trans[j]%MOD) %=MOD;}
17       res[i] = 0; }
18    res.erase(res.begin() + n + 1, res.end());
19    return res; }
20 LinearRec(poly &first, poly &trans): first(first), trans(trans) {
21    n = first.size(); poly a(n + 1, 0); a[1] = 1;
22    bin.push_back(a); for (int i = 1; i < LOG; ++i)
23       bin.push_back(add(bin[i - 1], bin[i - 1])); }
24 int calc(int k) { poly a(n + 1, 0); a[0] = 1;
25    for (int i = 0; i < LOG; ++i)
26       if (k >> i & 1) a = add(a, bin[i]);
27    int ret = 0; for (int i = 0; i < n; ++i)
28       if ((ret += (LL)a[i + 1] * first[i] % MOD) >= MOD)
29          ret -= MOD;
30    return ret; }};
```

5.19   Berlekamp-Massey 最小多项式

```
1  // Complexity: O(n^2) Requirement: const MOD, inverse(int)
2  // Input: the first elements of the sequence
3  // Output: the recursive equation of the given sequence
4  // Example In: {1, 1, 2, 3}
5  // Example Out: {1, 1000000006, 1000000006} (MOD = 1e9+7)
6  struct Poly { vector<int> a; Poly() { a.clear(); }
7     Poly(vector<int> &a): a(a) {}
8     int length() const { return a.size(); }
9     Poly move(int d) { vector<int> na(d, 0);
10       na.insert(na.end(), a.begin(), a.end());
11       return Poly(na); }
12    int calc(vector<int> &d, int pos) { int ret = 0;
13       for (int i = 0; i < (int)a.size(); ++i) {
14          if ((ret+=(LL)d[pos - i]*a[i]%MOD) >= MOD) {
15             ret -= MOD; }}
16       return ret; }
17    Poly operator - (const Poly &b) {
18       vector<int> na(max(this->length(), b.length()));
19       for (int i = 0; i < (int)na.size(); ++i) {
20          int aa = i < this->length() ? this->a[i] : 0,
21             bb = i < b.length() ? b.a[i] : 0;
22          na[i] = (aa + MOD - bb) % MOD; }
23       return Poly(na); } };
24 Poly operator * (const int &c, const Poly &p) {
25    vector<int> na(p.length());
26    for (int i = 0; i < (int)na.size(); ++i) {
27       na[i] = (LL)c * p.a[i] % MOD; }
28    return na; }
29 vector<int> solve(vector<int> a) {
30    int n = a.size(); Poly s, b;
31    s.a.push_back(1), b.a.push_back(1);
32    for (int i = 0, j = -1, ld = 1; i < n; ++i) {
33       int d = s.calc(a, i); if (d) {
34          if ((s.length() - 1) * 2 <= i) {
35             Poly ob = b; b = s;
36             s=s-(LL)d*inverse(ld)%MOD*ob.move(i - j);
37             j = i; ld = d;
38          } else {
39             s=s-(LL)d*inverse(ld)%MOD*b.move(i-j);}}}
40    //Caution: s.a might be shorter than expected
41    return s.a; }
42 /* 求行列式 -> 求特征多项式 : det(A)=(-1)^nPA(0)
43 求矩阵或向量列最小多项式 : 随机投影成数列
44 如果最小多项式里面有 x 的因子，那么行列式为 0，否则
45 随机乘上对角阵 D, det(A)=det(AD)/det(D)*/
```

5.20   Pell 方程

```
1  // x^2 - n * y^2 = 1 最小正整数根，n 为完全平方数时无解
2  // x_{k+1} = x_0 x_k + n y_0 y_k
```

```
3    // y_{k+1} = x_0 y_k + y_0 x_k
4    pair<LL, LL> peLL(LL n) {
5        static LL p[N], q[N], g[N], h[N], a[N];
6        p[1] = q[0] = h[1] = 1; p[0] = q[1] = g[1] = 0;
7        a[2] = (LL)(floor(sqrtl(n) + 1e-7L));
8        for(int i = 2; ; i ++) {
9            g[i] = -g[i - 1] + a[i] * h[i - 1];
10           h[i] = (n - g[i] * g[i]) / h[i - 1];
11           a[i + 1] = (g[i] + a[2]) / h[i];
12           p[i] = a[i] * p[i - 1] + p[i - 2];
13           q[i] = a[i] * q[i - 1] + q[i - 2];
14           if(p[i] * p[i] - n * q[i] * q[i] == 1)
15               return {p[i], q[i]}; }}
```

5.21 解一元三次方程

```
1    double a(p[3]), b(p[2]), c(p[1]), d(p[0]);
2    double k(b / a), m(c / a), n(d / a);
3    double p(-k * k / 3. + m);
4    double q(2. * k * k * k / 27 - k * m / 3. + n);
5    Complex omega[3] = {Complex(1, 0), Complex(-0.5, 0.5 * sqrt(3)), Complex(-0.5, -0.5 * sqrt(3))};
6    Complex r1, r2; double delta(q * q / 4 + p * p * p / 27);
7    if (delta > 0) {
8        r1 = cubrt(-q / 2. + sqrt(delta));
9        r2 = cubrt(-q / 2. - sqrt(delta));
10   } else {
11       r1 = pow(-q / 2. + pow(Complex(delta), 0.5), 1. / 3);
12       r2 = pow(-q / 2. - pow(Complex(delta), 0.5), 1. / 3); }
13   for(int _(0); _ < 3; _++) {
14       Complex x = -k/3. + r1*omega[_] + r2*omega[_* 2 % 3]; }
```

5.22 自适应 Simpson

```
1    // Adaptive Simpson's method : double simpson::solve (double (*f) (double), double l, double r, double eps) : integrates f
     ↪ over (l, r) with error eps.
2    struct simpson {
3    double area (double (*f) (double), double l, double r) {
4        double m = l + (r - l) / 2;
5        return (f (l) + 4 * f (m) + f (r)) * (r - l) / 6;
6    }
7    double solve (double (*f) (double), double l, double r, double eps, double a) {
8        double m = l + (r - l) / 2;
9        double left = area (f, l, m), right = area (f, m, r);
10       if (fabs (left + right - a) <= 15 * eps) return left + right + (left + right - a) / 15.0;
11       return solve (f, l, m, eps / 2, left) + solve (f, m, r, eps / 2, right);
12   }
13   double solve (double (*f) (double), double l, double r, double eps) {
14       return solve (f, l, r, eps, area (f, l, r));
15   }};
```

5.23 Schreier-Sims 求排列群的阶

```
1    struct Perm{
2        vector<int> P; Perm() {} Perm(int n) { P.resize(n); }
3        Perm inv() const{ Perm ret(P.size());
4            for(int i = 0; i < int(P.size()); ++i) ret.P[P[i]]=i;
5            return ret; }
6        int &operator [](const int &dn){ return P[dn]; }
7        void resize(const size_t &sz){ P.resize(sz); }
8        size_t size()const{ return P.size(); }
9        const int &operator[](const int &dn)const{return P[dn];}};
10   Perm operator *(const Perm &a, const Perm &b){
11       Perm ret(a.size());
12       for(int i = 0; i < (int)a.size(); ++i) ret[i] = b[a[i]];
13       return ret; }
14   typedef vector<Perm> Bucket; typedef vector<int> Table;
15   typedef pair<int,int> PII; int n, m;
16   vector<Bucket> buckets, bucketsInv;vector<Table> lookupTable;
17   int fastFilter(const Perm &g, bool addToGroup = true) {
18       int n = buckets.size(); Perm p(g);
19       for(int i = 0; i < n; ++i){
20           int res = lookupTable[i][p[i]];
21           if(res == -1) {
```

```
22          if(addToGroup) { buckets[i].push_back(p);
23              bucketsInv[i].push_back(p.inv());
24              lookupTable[i][p[i]]=(int)buckets[i].size()-1; }
25          return i; }
26      p = p * bucketsInv[i][res]; }
27      return -1; }
28  long long calcTotalSize() { long long ret = 1;
29      for(int i = 0; i < n; ++i) ret *= buckets[i].size();
30      return ret; }
31  bool inGroup(const Perm &g){return fastFilter(g,false)==-1;}
32  void solve(const Bucket &gen,int _n) {// m perm[0..n - 1]s
33      n = _n, m = gen.size(); { //clear all
34          vector<Bucket> _b(n); swap(buckets, _b);
35          vector<Bucket> _I(n); swap(bucketsInv, _I);
36          vector<Table> _T(n); swap(lookupTable, _T); }
37      for(int i = 0; i < n; ++i){
38          lookupTable[i].resize(n);
39          fill(lookupTable[i].begin(),lookupTable[i].end(),-1);}
40      Perm id(n); for(int i = 0; i < n; ++i) id[i] = i;
41      for(int i = 0; i < n; ++i){
42          buckets[i].push_back(id);bucketsInv[i].push_back(id);
43          lookupTable[i][i] = 0; }
44      for(int i = 0; i < m; ++i) fastFilter(gen[i]);
45      queue<pair<PII,PII> > toUpdate;
46      for(int i = 0; i < n; ++i)
47          for(int j = i; j < n; ++j)
48              for(int k = 0; k < (int) buckets[i].size(); ++k)
49                  for(int l = 0; l < (int) buckets[j].size(); ++l)
50                      toUpdate.push(make_pair(PII(i,k), PII(j,l)));
51      while(!toUpdate.empty()){
52          PII a=toUpdate.front().first;
53          PII b=toUpdate.front().second; toUpdate.pop();
54          int res = fastFilter(buckets[a.first][a.second] * buckets[b.first][b.second]);
55          if(res==-1) continue;
56          PII newPair(res, (int)buckets[res].size() - 1);
57          for(int i = 0; i < n; ++i)
58              for(int j = 0; j < (int)buckets[i].size(); ++j){
59                  if(i <= res)
60                      toUpdate.push(make_pair(PII(i, j), newPair));
61                  if(res <= i)
62                      toUpdate.push(make_pair(newPair,PII(i,j)));}}}
```

5.24   类欧几里得 直线下格点统计

```
// f(n,a,b,c) = ∑_{i=0}^{n} ⌊(a+bi)/c⌋,  n,a,b,c > 0
// g(n,a,b,c) = ∑_{i=0}^{n} i⌊(a+bi)/c⌋
// h(n,a,b,c) = ∑_{i=0}^{n} ⌊(a+bi)/c⌋²
struct node
{
    ll f,g,h;
    node(){f=0,g=0,h=0;}
    node(ll a,ll b,ll c){f=a,g=b,h=c;}
    void print(){printf("%lld %lld %lld\n",f,h,g);}
};
node solve(ll n,ll a,ll b,ll c)
{
    if(!a)return node((b/c)*(n+1)%mod,(n*(n+1)%mod*inv2)%mod*(b/c)%mod,(b/c)*(b/c)%mod*(n+1)%mod);
    node ret=node();
    if(a>=c||b>=c)
    {
        ll t1=a/c,t2=b/c,s1=n*(n+1)%mod*inv2%mod,s2=n*(n+1)%mod*(2*n+1)%mod*inv6%mod;
        node tmp=solve(n,a%c,b%c,c);
        ret.f=(tmp.f+(n+1)*t2%mod+s1*t1%mod)%mod;
        ret.g=(tmp.g+s1*t2%mod+s2*t1%mod)%mod;
        ret.h=(t1*t1%mod*s2%mod+(n+1)*t2%mod*t2%mod+2*t1*t2%mod*s1%mod+2*t1*tmp.g%mod+2*t2*tmp.f%mod+tmp.h)%mod;
        return ret;
    }
    ll m=(a*n+b)/c;node tmp=solve(m-1,c,c-b-1,a);
    ret.f=(n*m%mod-tmp.f+mod)%mod;
    ret.g=((n*(n+1)%mod*m%mod-tmp.f-tmp.h)%mod+mod)*inv2%mod;
    ret.h=((n*m%mod*m%mod-tmp.g*2%mod-tmp.f)%mod+mod)%mod;
    return ret;
}
```

6. 卡常

6.1 手写哈希

```
1  template <class T,int P = 314159/*,451411,1141109,2119969*/>
2  struct hashmap {
3  ULL id[P]; T val[P];
4  int rec[P]; // del: no many clears
5  hashmap() {memset(id, -1, sizeof id);}
6  int get(const ULL &x) const {
7      for (int i = int(x % P), j = 1; ~id[i]; i = i + j >= P ? i + j - P : i + j, j = j + 2 >= P ? j + 2 - P : j + 2) {
8          if (id[i] == x) return val[i];
9      }
10     return 0;
11 }
12 int& operator [] (const ULL &x) {
13     for (int i = int(x % P), j = 1;        ; i = i + j >= P ? i + j - P : i + j, j = j + 2 >= P ? j + 2 - P : j + 2) {
14         if (id[i] == x) return val[i];
15         else if (id[i] == -1llu) {
16             id[i] = x;
17             rec[++rec[0]] = i;  // del: no many clears
18             return val[i];
19         }
20     }
21 }
22 void clear() { // del
23     while(rec[0]) id[rec[rec[0]]] = -1, val[rec[rec[0]]] = 0, --rec[0];
24 }
25 void fullclear() {
26     memset(id, -1, sizeof id);
27     rec[0] = 0;
28 }
29 };
```

6.2 基数排序

```
1  void SORT(int a[], int c[], int n, int l) {
2      for (int i = 0; i < SZ; i++) b[i] = 0;
3      for (int i = 1; i <= n; i++) b[(a[i] >> l) & (SZ - 1)]++;
4      for (int i = 1; i < SZ; i++) b[i] += b[i - 1];
5      for (int i = n; i; i--) c[b[(a[i] >> l) & (SZ - 1)]--] = a[i];
6  }
7
8  void Sort(int *a, int b){
9      if (b < 23333) sort(a + 1, a + b + 1);
10     else {
11         SORT(a, c, b, 0);
12         SORT(c, a, b, 14);
13     }
14 }
```

6.3 子集求和(特殊的高维前缀和)

```
1  for (i = 0; i < n; i++)
2      for (j = 0; j < (1 << n); j++)
3          if (j >> i & 1) x[j] += x[j ^ (1 << i)]
```

6.4 快速乘

```
1  inline int mul_mod(const int a, const int b, const int mo) {
2      int ret;
3      __asm__ __volatile__ ("\tmull %%ebx\n\tdivl %%ecx\n" : "=d"(ret):"a"(a),"b"(b),"c"(mo));
4      return ret;
5  }
6
7  typedef long long ll;
8  inline ll mul_mod(const ll a, const ll b, const ll mo) {
9      ll ret;
10     __asm__ __volatile__ ("\tmulq %%rbx\n\tdivq %%rcx\n" : "=d"(ret):"a"(a),"b"(b),"c"(mo));
11     return ret;
12 }
```

7. Miscellany

7.1 Zeller 日期公式

```
1  // weekday=(id+1)%7;{Sun=0,Mon=1,...}   getId(1, 1, 1) = 0
2  int getId(int y, int m, int d) {
3      if (m < 3) { y --; m += 12; }
4      return 365 * y + y / 4 - y / 100 + y / 400 + (153 * (m - 3) + 2) / 5 + d - 307; }
5  //  当y小于0时，将除法改为向下取整后即可保证正确性，或统一加400的倍数年
6  auto date(int id) {
7      int x=id+1789995, n, i, j, y, m, d;
8      n = 4 * x / 146097; x -= (146097 * n + 3) / 4;
9      i = (4000 * (x + 1)) / 1461001; x -= 1461 * i / 4 - 31;
10     j = 80 * x / 2447; d = x - 2447 * j / 80; x = j / 11;
11     m = j + 2 - 12 * x; y = 100 * (n - 49) + i + x;
12     return make_tuple(y, m, d); }
```

7.2 Dancing Links 精确覆盖搜索

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

7.3 Java Template

```
1  import java.io.*; import java.util.*; import java.math.*;
2  public class Main {
3  static class solver { public void run(Scanner cin, PrintStream out) {} }
4  public static void main(String[] args) {
5  // Fast Reader & Big Numbers
```

```java
 6  InputReader in = new InputReader(System.in);
 7  PrintWriter out = new PrintWriter(System.out);
 8  BigInteger c = in.nextInt();
 9  out.println(c.toString(8)); out.close(); // as Oct
10  BigDecimal d = new BigDecimal(10.0);
11  // d=d.divide(num, length, BigDecimal.ROUND_HALF_UP)
12  d.setScale(2, BigDecimal.ROUND_FLOOR); // 用于输出
13  System.out.printf("%.6f\n", 1.23); // C 格式
14  BigInteger num = BigInteger.valueOf(6);
15  num.isProbablePrime(10); // 1 - 1 / 2 ^ certainty
16  BigInteger rev = num.modPow(BigInteger.valueOf(-1), BigInteger.valueOf(25));   // rev=6^-1mod25 要互质
17  num = num.nextProbablePrime(); num.intValue();
18  Scanner cin=new Scanner(System.in);//SimpleReader
19  int n = cin.nextInt();
20  int [] a = new int [n]; // 初值未定义
21  // Random rand.nextInt(N) [0,N)
22  Arrays.sort(a, 5, 10, cmp); // sort(a+5, a+10)
23  ArrayList<Long> list = new ArrayList(); // vector
24  // .add(val) .add(pos, val) .remove(pos)
25  Comparator cmp=new Comparator<Long>(){ // 自定义逆序
26    @Override public int compare(Long o1, Long o2) {
27    /* o1 < o2 ? 1 :( o1 > o2  ? -1 : 0) */ } };
28  // Collections. shuffle(list) sort(list, cmp)
29  Long [] tmp = list.toArray(new Long [0]);
30  // list.get(pos) list.size()
31  Map<Integer,String> m = new HashMap<Integer,String>();
32  //m.put(key,val) get(key) containsKey(key) remove(key)
33  for (Map.Entry<Integer,String> entry:m.entrySet()); //entry.getKey() getValue()
34  Set<String> s = new HashSet<String>(); // TreeSet
35  //s.add(val)contains(val)remove(val);for(var : s)
36  solver Task=new solver();Task.run(cin,System.out);
37  PriorityQueue<Integer> Q=new PriorityQueue<Integer>();
38  // Q. offer(val) poll() peek() size()
39  // Read / Write a file : FileWriter FileReader PrintStream
40  } static class InputReader { // Fast Reader
41  public BufferedReader reader;
42  public StringTokenizer tokenizer;
43  public InputReader(InputStream stream) {
44    reader = new BufferedReader(new InputStreamReader(stream), 32768);
45    tokenizer = null; }
46  public String next() {
47    while (tokenizer==null||!tokenizer.hasMoreTokens()) {
48      try { String line = reader.readLine();
49        /*line == null ? end of file*/
50        tokenizer = new StringTokenizer(line);
51      } catch (IOException e) {
52        throw new RuntimeException(e); }
53    } return tokenizer.nextToken(); }
54  public BigInteger nextInt() {
55    //return Long.parseLong(next()); Double Integer
56    return new BigInteger(next(), 16); // as Hex
57  } } }
58  //-----------------------------------------------------------------
59  import java.math.*;
60  import java.util.Scanner;
61
62  public class Main
63  {
64    public static void main (String[] args)
65    {
66      BigInteger dp[][][]=new BigInteger [65][65][65];
67      dp[0][0][0]=BigInteger.ONE;
68      for (int i=1;i<=60;i++)
69        for (int j=0;j<=i;j++)
70          for (int k=0;k<=j;k++)
71          {
72            dp[i][j][k]=BigInteger.valueOf(0);
73            if (i>j) dp[i][j][k]=dp[i][j][k].add(dp[i-1][j][k]);
74            if (j>k) dp[i][j][k]=dp[i][j][k].add(dp[i][j-1][k]);
75            if (k>0) dp[i][j][k]=dp[i][j][k].add(dp[i][j][k-1]);
76          }
77      Scanner cin=new Scanner (System.in);
78      while (cin.hasNext())
```

```
79      {
80        int x=cin.nextInt();
81        System.out.println(dp[x][x][x]);
82        System.out.println();
83      }
84    }
85 }
```

7.4 Vimrc

```
1 source $VIMRUNTIME/vimrc_example.vim
2 source $VIMRUNTIME/mswin.vim
3 behave mswin
4 set hlsearch ci ai si nu ts=4 sw=4
5 color slate
6 map <F9> : ! g++ % -o %< -g -Wall -Wextra -Wconversion -Wshadow -std=c++11 <CR>
7 map <F5> : ! time ./%< <CR>
8 map <F8> : ! time ./%< < %<.in <CR>
9 map <F12> :r std.cpp <CR> ggdd : 15 <CR>
```

7.5 读入优化相关

```
1 #define __ __attribute__ ((optimize ("-O3")))
2 #define _ __ __inline __attribute__ ((__gnu_inline__, __always_inline__, __artificial__))
3 __inline int next_uint () {
4   const int SIZE = 110000; static char buf[SIZE + 1]; static int p = SIZE;
5   register int ans = 0, f = 1, sgn = 1;
6   while ((p < SIZE || (p = 0, buf[fread (buf, 1, SIZE, stdin)] = 0, buf[0])) && (isdigit (buf[p]) && (ans = ans * 10 +
     ↪ buf[p] - '0', f = 0, 1) || f /* && (buf[p] == '-' && (sgn = 0), 1) */ )) ++p;
7   return sgn ? ans : -ans; }
8 bitset._Find_first();bitset._Find_next(idx);
9 struct HashFunc{size_t operator()(const KEY &key)const{}};
```

8. Appendix

8.1 Formulas 公式表
8.1.1 Mobius Inversion

$$F(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) F(\frac{n}{d})$$

$$F(n) = \sum_{n|d} f(d) \Rightarrow f(n) = \sum_{n|d} \mu(\frac{d}{n}) F(d)$$

$$[x = 1] = \sum_{d|x} \mu(d), \quad x = \sum_{d|x} \mu(d)$$

8.1.2 Gcd Inversion

$$\sum_{a=1}^{n} \sum_{b=1}^{n} gcd^2(a,b) = \sum_{d=1}^{n} d^2 \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} [gcd(i,j) = 1]$$

$$= \sum_{d=1}^{n} d^2 \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} \sum_{t|gcd(i,j)} \mu(t)$$

$$= \sum_{d=1}^{n} d^2 \sum_{t=1}^{\lfloor \frac{n}{d} \rfloor} \mu(t) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} [t|i] \sum_{j=1}^{\lfloor \frac{n}{d} \rfloor} [t|j]$$

$$= \sum_{d=1}^{n} d^2 \sum_{t=1}^{\lfloor \frac{n}{d} \rfloor} \mu(t) \lfloor \frac{n}{dt} \rfloor^2$$

The formula can be calculated in $O(nlogn)$ complexity. Moreover, let $l = dt$, then

$$\sum_{d=1}^{n} d^2 \sum_{t=1}^{\lfloor \frac{n}{d} \rfloor} \mu(t) \lfloor \frac{n}{dt} \rfloor^2 = \sum_{l=1}^{n} \lfloor \frac{n}{l} \rfloor^2 \sum_{d|l} d^2 \mu(\frac{l}{d})$$

Let $f(l) = \sum_{d|l} d^2 \mu(\frac{l}{d})$. It can be proven that $f(l)$ is multiplicative. Besides, $f(p^k) = p^{2k} - p^{2k-2}$. Therefore, with linear sieve the formula can be solved in $O(n)$ complexity.
8.1.3 Arithmetic Function

$$(p-1)! \equiv -1 (mod\ p)$$

$$a > 1, m, n > 0, \text{ then } gcd(a^m - 1, a^n - 1) = a^{\gcd(n,m)} - 1$$

$$\mu^2(n) = \sum_{d^2|n} \mu(d)$$

$$a > b, \gcd(a, b) = 1, \text{ then } \gcd(a^m - b^m, a^n - b^n) = a^{\gcd(m,n)} - b^{\gcd(m,n)}$$

$$\prod_{k=1, gcd(k,m)=1}^{m} k \equiv \begin{cases} -1 & \mod m, \, m = 4, p^q, 2p^q \\ 1 & \mod m, \text{ otherwise} \end{cases}$$

$$\sigma_k(n) = \sum_{d|n} d^k = \prod_{i=1}^{\omega(n)} \frac{p_i^{(a_i+1)k} - 1}{p_i^k - 1}$$

$$J_k(n) = n^k \prod_{p|n}(1 - \frac{1}{p^k})$$

$J_k(n)$ is the number of k-tuples of positive integers all less than or equal to n that form a coprime $(k+1)$-tuple together with n.

$$\sum_{\delta|n} J_k(\delta) = n^k$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} n[gcd(i,j) = 1]ij = \sum_{i=1}^{n} i^2 \varphi(i)$$

$$\sum_{\delta|n} \delta^s J_r(\delta) J_s(\frac{n}{\delta}) = J_{r+s}(n)$$

$$\sum_{\delta|n} \varphi(\delta) d(\frac{n}{\delta}) = \sigma(n), \quad \sum_{\delta|n} |\mu(\delta)| = 2^{\omega(n)}$$

$$\sum_{\delta|n} 2^{\omega(\delta)} = d(n^2), \quad \sum_{\delta|n} d(\delta^2) = d^2(n)$$

$$\sum_{\delta|n} d(\frac{n}{\delta}) 2^{\omega(\delta)} = d^2(n), \quad \sum_{\delta|n} \frac{\mu(\delta)}{\delta} = \frac{\varphi(n)}{n}$$

$$\sum_{\delta|n} \frac{\mu(\delta)}{\varphi(\delta)} = d(n), \quad \sum_{\delta|n} \frac{\mu^2(\delta)}{\varphi(\delta)} = \frac{n}{\varphi(n)}$$

$$n|\varphi(a^n - 1)$$

$$\sum_{\substack{1 \le k \le n \\ gcd(k,n)=1}} f(\gcd(k-1,n)) = \varphi(n) \sum_{d|n} \frac{(\mu * f)(d)}{\varphi(d)}$$

$$\varphi(\text{lcm}(m,n))\varphi(\gcd(m,n)) = \varphi(m)\varphi(n)$$

$$\sum_{\delta|n} d^3(\delta) = (\sum_{\delta|n} d(\delta))^2$$

$$d(uv) = \sum_{\delta| \gcd(u,v)} \mu(\delta) d(\frac{u}{\delta}) d(\frac{v}{\delta})$$

$$\sigma_k(u)\sigma_k(v) = \sum_{\delta| \gcd(u,v)} \delta^k \sigma_k(\frac{uv}{\delta^2})$$

$$\mu(n) = \sum_{k=1}^{n} [\gcd(k,n) = 1] \cos 2\pi \frac{k}{n}$$

$$\varphi(n) = \sum_{k=1}^{n} [\gcd(k,n) = 1] = \sum_{k=1}^{n} \gcd(k,n) \cos 2\pi \frac{k}{n}$$

$$\begin{cases} S(n) = \sum_{k=1}^{n} (f * g)(k) \\ \sum_{k=1}^{n} S(\lfloor \frac{n}{k} \rfloor) = \sum_{i=1}^{n} f(i) \sum_{j=1}^{\lfloor \frac{n}{i} \rfloor} (g * 1)(j) \end{cases}$$

$$\begin{cases} S(n) = \sum_{k=1}^{n} (f \cdot g)(k), g \text{ completely multiplicative} \\ \sum_{k=1}^{n} S(\lfloor \frac{n}{k} \rfloor) g(k) = \sum_{k=1}^{n} (f * 1)(k)g(k) \end{cases}$$

8.1.4 Binomial Coefficients

$$\binom{n}{k} = (-1)^k \binom{k-n-1}{k}, \quad \sum_{k \le n} \binom{r+k}{k} = \binom{r+n+1}{n}$$

$$\sum_{k=0}^{n} \binom{k}{m} = \binom{n+1}{m+1}$$

$$\sqrt{1+z} = 1 + \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k \times 2^{2k-1}} \binom{2k-2}{k-1} z^k$$

$$\sum_{k=0}^{r} \binom{r-k}{m} \binom{s+k}{n} = \binom{r+s+1}{m+n+1}$$

$$C_{n,m} = \binom{n+m}{m} - \binom{n+m}{m-1}, n \geq m$$

$$\binom{n}{k} \equiv [n\&k = k] \pmod 2$$

$$\binom{n_1 + \cdots + n_p}{m} = \sum_{k_1 + \cdots + k_p = m} \binom{n_1}{k_1} \cdots \binom{n_p}{k_p}$$

8.1.5   Fibonacci Numbers

$$F(z) = \frac{z}{1 - z - z^2}$$

$$f_n = \frac{\phi^n - \hat{\phi}^n}{\sqrt 5}, \phi = \frac{1 + \sqrt 5}{2}, \hat{\phi} = \frac{1 - \sqrt 5}{2}$$

$$\sum_{k=1}^n f_k = f_{n+2} - 1, \quad \sum_{k=1}^n f_k^2 = f_n f_{n+1}$$

$$\sum_{k=0}^n f_k f_{n-k} = \frac{1}{5}(n-1)f_n + \frac{2}{5}n f_{n-1}$$

$$\frac{f_{2n}}{f_n} = f_{n-1} + f_{n+1}$$

$$f_1 + 2f_2 + 3f_3 + \cdots + nf_n = nf_{n+2} - f_{n+3} + 2]$$

$$\gcd(f_m, f_n) = f_{\gcd(m,n)}$$

$$f_n^2 + (-1)^n = f_{n+1} f_{n-1}$$

$$f_{n+k} = f_n f_{k+1} + f_{n-1} f_k$$

$$f_{2n+1} = f_n^2 + f_{n+1}^2$$

$$(-1)^k f_{n-k} = f_n f_{k-1} - f_{n-1} f_k$$

$$\text{Modulo } f_n, f_{mn+r} \equiv \begin{cases} f_r, & m \bmod 4 = 0; \\ (-1)^{r+1} f_{n-r}, & m \bmod 4 = 1; \\ (-1)^n f_r, & m \bmod 4 = 2; \\ (-1)^{r+1+n} f_{n-r}, & m \bmod 4 = 3. \end{cases}$$

8.1.6   Lucas Numbers

$$L_0 = 2, L_1 = 1, L_n = L_{n-1} + L_{n-2} = (\frac{1 + \sqrt 5}{2})^n + (\frac{1 - \sqrt 5}{2})^n$$

$$L(x) = \frac{2 - x}{1 - x - x^2}$$

8.1.7   Catlan Numbers

$$c_1 = 1, c_n = \sum_{i=0}^{n-1} c_i c_{n-1-i} = c_{n-1} \frac{4n-2}{n+1} = \frac{\binom{2n}{n}}{n+1} = \binom{2n}{n} - \binom{2n}{n-1}$$

$$c(x) = \frac{1 - \sqrt{1 - 4x}}{2x}$$

Usage: n对括号序列; n个点满二叉树; $n*n$的方格左下到右上不过对角线方案数; 凸$n+2$边形三角形分割数; n个数的出栈方案数; $2n$个顶点连接, 线段两两不交的方案数.

8.1.8   Stirling Cycle Numbers
把n个元素集合分作k个非空环方案数.

$$s(n,0) = 0, s(n,n) = 1, s(n+1,k) = s(n,k-1) - nS(n,k)$$

$$s(n,k) = (-1)^{n-k} \begin{bmatrix} n \\ k \end{bmatrix}$$

$$\begin{bmatrix} n+1 \\ k \end{bmatrix} = n \begin{bmatrix} n \\ k \end{bmatrix} + \begin{bmatrix} n \\ k-1 \end{bmatrix}, \quad \begin{bmatrix} n+1 \\ 2 \end{bmatrix} = n! H_n$$

$$x^n = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} (-1)^{n-k} x^k, \quad x^{\overline{n}} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} x^k$$

8.1.9   Stirling Subset Numbers
把n个元素集合分作k个非空子集方案数.

$$\begin{Bmatrix} n+1 \\ k \end{Bmatrix} = k \begin{Bmatrix} n \\ k \end{Bmatrix} + \begin{Bmatrix} n \\ k-1 \end{Bmatrix}$$

$$x^n = \sum_k \begin{Bmatrix} n \\ k \end{Bmatrix} x^{\underline{k}} = \sum_k \begin{Bmatrix} n \\ k \end{Bmatrix} (-1)^{n-k} x^{\overline{k}}$$

$$m! \begin{Bmatrix} n \\ m \end{Bmatrix} = \sum_k \binom{m}{k} k^n (-1)^{m-k}$$

For fixed k, generating functions :

$$\sum_{n=0}^\infty \begin{Bmatrix} n \\ k \end{Bmatrix} x^{n-k} = \prod_{r=1}^k \frac{1}{1 - rx}$$

8.1.10 Motzkin Numbers

圆上n点间画不相交弦的方案数. 选n个数$k_1, k_2, ..., k_n \in \{-1, 0, 1\}$ 保证$\sum_i^a k_i(1 \le a \le n)$非负且所有数总和为0的方案数.

$$M_{n+1} = M_n + \sum_i^{n-1} M_i M_{n-1-i} = \frac{(2n+3)M_n + 3nM_{n-1}}{n+3}$$

$$M_n = \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2k} Catlan(k)$$

$$M(X) = \frac{1 - x - \sqrt{1 - 2x - 3x^2}}{2x^2}$$

8.1.11 Eulerian Numbers

$$\left\langle {n \atop k} \right\rangle = (k+1)\left\langle {n-1 \atop k} \right\rangle + (n-k)\left\langle {n-1 \atop k-1} \right\rangle$$

$$x^n = \sum_k \left\langle {n \atop k} \right\rangle \binom{x+k}{n}$$

$$\left\langle {n \atop m} \right\rangle = \sum_{k=0}^{m} \binom{n+1}{k}(m+1-k)^n(-1)^k$$

8.1.12 Harmonic Numbers

$$\sum_{k=1}^{n} H_k = (n+1)H_n - n$$

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

$$\sum_{k=1}^{n} \binom{k}{m} H_k = \binom{n+1}{m+1}(H_{n+1} - \frac{1}{m+1})$$

8.1.13 Pentagonal Number Theorem

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

$$p(n) = p(n-1) + p(n-2) - p(n-5) - p(n-7) + \cdots$$

$$f(n,k) = p(n) - p(n-k) - p(n-2k) + p(n-5k) + p(n-7k) - \cdots$$

8.1.14 Bell Numbers

n个元素集合划分的方案数.

$$B_n = \sum_{k=1}^{n} \left\{ {n \atop k} \right\}, \quad B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

$$B_{p^m+n} \equiv mB_n + B_{n+1} \pmod{p}$$

$$B(x) = \sum_{n=0}^{\infty} \frac{B_n}{n!} x^n = e^{e^x - 1}$$

8.1.15 Bernoulli Numbers

$$B_n = 1 - \sum_{k=0}^{n-1} \binom{n}{k} \frac{B_k}{n-k+1}$$

$$G(x) = \sum_{k=0}^{\infty} \frac{B_k}{k!} x^k = \frac{1}{\sum_{k=0}^{\infty} \frac{x^k}{(k+1)!}}$$

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

8.1.16 Sum of Powers

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = (\frac{n(n+1)}{2})^2$$

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

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

8.1.17 Sum of Squares

$r_k(n)$表示用k个平方数组成n的方案数. 假设:

$$n = 2^{a_0} p_1^{2a_1} \cdots p_r^{2a_r} q_1 b_1 \cdots q_s b_s$$

其中$p_i \equiv 3 \mod 4$, $q_i \equiv 1 \mod 4$, 那么

$$r_2(n) = \begin{cases} 0 & \text{if any } a_i \text{ is a half-integer} \\ 4\prod_1^r (b_i+1) & \text{if all } a_i \text{ are integers} \end{cases}$$

$r_3(n) > 0$当且仅当n不满足$4^a(8b+7)$的形式 $(a, b$为整数$)$.

8.1.18 Pythagorean Triple

枚举 $x^2 + y^2 = z^2$ 的三元组: 可令 $x = m^2 - n^2, y = 2mn, z = m^2 + n^2$, 枚举 m 和 n 即可 $O(n)$ 枚举勾股数. 判断素勾股数方法: m, n 至少一个为偶数并且 m, n 互质, 那么 x, y, z 就是素勾股数.

8.1.19 Tetrahedron Volume

If U, V, W, u, v, w are lengths of edges of the tetrahedron (first three form a triangle; u opposite to U and so on)

$$V = \frac{\sqrt{4u^2v^2w^2 - \sum_{cyc} u^2(v^2 + w^2 - U^2)^2 + \prod_{cyc} (v^2 + w^2 - U^2)}}{12}$$

8.1.20 杨氏矩阵 与 钩子公式

满足: 格子 (i, j) 没有元素, 则它右边和上边相邻格子也没有元素; 格子 (i, j) 有元素 $a[i][j]$, 则它右边和上边相邻格子要么没有元素, 要么有元素且比 $a[i][j]$ 大.

计数: $F_1 = 1, F_2 = 2, F_n = F_{n-1} + (n-1)F_{n-2}, F(x) = e^{x + \frac{x^2}{2}}$

钩子公式: 对于给定形状 λ, 不同杨氏矩阵的个数为:

$$d_\lambda = \frac{n!}{\prod h_\lambda(i, j)}$$

$h_\lambda(i, j)$ 表示该格子右边和上边的格子数量加 1.

8.1.21 重心

半径为 r, 圆心角为 θ 的扇形重心与圆心的距离为 $\frac{4r\sin(\theta/2)}{3\theta}$

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

8.1.22 常见游戏

Anti-Nim 游戏 n 堆石子轮流拿 谁走最后一步输 结论 先手胜当且仅当 1. 所有堆石子数都为 1 且游戏的 SG 值为 0 (即有偶数个孤单堆-每堆只有 1 个石子数) 2. 存在某堆石子数大于 1 且游戏的 SG 值不为 0.

斐波那契博弈   有一堆物品, 两人轮流取物品, 先手最少取一个, 至多无上限, 但不能把物品取完, 之后每次取的物品数不能超过上一次取的物品数的二倍且至少为一件, 取走最后一件物品的人获胜. 结论: 先手胜当且仅当物品数 n 不是斐波那契数.

威佐夫博弈   有两堆石子, 博弈双方每次可以取一堆石子中的任意个, 不能不取, 或者取两堆石子中的相同个. 先取完者赢. 结论: 求出两堆石子 A 和 B 的差值 C, 如果 $\left\lfloor C * \frac{\sqrt{5}+1}{2} \right\rfloor = min(A, B)$ 那么后手赢, 否则先手赢.

约瑟夫环   令 n 个人标号为 $0, 1, 2, ..., n-1$, 令 $f_{i,m}$ 表示 i 个人报 m 胜利者的编号, 则 $f_{1,m} = 0, f_{i,m} = (f_{i-1,m} + m) mod \ i$.

8.1.23 错排公式

$$D_1 = 0, D_2 = 1, D_n = n!(\frac{1}{0!} - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + ... + \frac{(-1)^n}{n!})$$
$$D_n = (n-1)(D_{n-1} + D_{n-2})$$

8.1.24 概率相关

对于随机变量 X, 期望用 $E(X)$ 表示, 方差用 $D(X)$ 表示, 则 $D(X) = E(X - E(X))^2 = E(X^2) - (E(X))^2, D(X + Y) = D(X) + D(Y)D(aX) = a^2D(X)$

$$E[x] = \sum_{i=1}^{\infty} P(X \geq i)$$

8.1.25 常用泰勒展开

$$f(x) = \frac{f(x_0)}{0!} + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots$$
$$\frac{1}{(1-x)^n} = 1 + \binom{n}{1}x + \binom{n+1}{2}x^2 + \binom{n+2}{3}x^3 + \cdots$$
$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^i}{i!}$$

8.1.26 类卡特兰数

从 $(1,1)$ 出发走到 (n,m), 只能向右或者向上走, 不能越过 y=x 这条线 (即保证 $x \geq y$), 合法方案数是 $C_{n+m-2}^{n-1} - C_{n+m-2}^{n}$.

8.1.27 邻接矩阵行列式的意义

在无向图中取若干个环, 一种取法权值就是边权的乘积, 对行列式的贡献是 $(-1)^{even}$, 其中 $even$ 是偶环的个数.

8.1.28 Others (某些近似数值公式在这里)

$$S_j = \sum_{k=1}^{n} x_k^j$$
$$h_m = \sum_{1 \leq j_1 < \cdots < j_m \leq n} x_{j_1} \cdots x_{j_m}, \quad H_m = \sum_{1 \leq j_1 \leq \cdots \leq j_m \leq n} x_{j_1} \cdots x_{j_m}$$
$$h_n = \frac{1}{n} \sum_{k=1}^{n} (-1)^{k+1} S_k h_{n-k}$$
$$H_n = \frac{1}{n} \sum_{k=1}^{n} S_k H_{n-k}$$
$$\sum_{k=0}^{n} kc^k = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}$$
$$\sum_{i=1}^{n} = ln(n) + \Gamma, (\Gamma \approx 0.5772156649015328606065 1209)$$
$$n! = \sqrt{2\pi n}(\frac{n}{e})^n(1 + \frac{1}{12n} + \frac{1}{288n^2} + O(\frac{1}{n^3}))$$
$$\max\{x_a - x_b, y_a - y_b, z_a - z_b\} - \min\{x_a - x_b, y_a - y_b, z_a - z_b\}$$
$$= \frac{1}{2} \sum_{cyc} |(x_a - y_a) - (x_b - y_b)|$$

$$(a+b)(b+c)(c+a) = \frac{(a+b+c)^3 - a^3 - b^3 - c^3}{3}$$

$$a^3 + b^3 = (a+b)(a^2 - ab + b^2), a^3 - b^3 = (a-b)(a^2 + ab + b^2)$$

$$a^n + b^n = (a+b)(a^{n-1} - a^{n-2}b + a^{n-3}b^2 - ... - ab^{n-2} + b^{n-1})(n \bmod 2 = 1)$$

划分问题: n个$k-1$维向量最多把k维空间分为$\sum_{i=0}^{k} C_n^i$ 份.

8.2 Calculus Table 导数表

$(\frac{u}{v})' = \frac{u'v - uv'}{v^2}$

$(a^x)' = (\ln a)a^x$

$(\tan x)' = \sec^2 x$

$(\cot x)' = \csc^2 x$

$(\sec x)' = \tan x \sec x$

$(\csc x)' = -\cot x \csc x$

$(\arcsin x)' = \frac{1}{\sqrt{1-x^2}}$

$(\arccos x)' = -\frac{1}{\sqrt{1-x^2}}$

$(\arctan x)' = \frac{1}{1+x^2}$

$(\text{arccot } x)' = -\frac{1}{1+x^2}$

$(\text{arccsc } x)' = -\frac{1}{x\sqrt{1-x^2}}$

$(\text{arcsec } x)' = \frac{1}{x\sqrt{1-x^2}}$

$(\tanh x)' = \text{sech}^2 x$

$(\coth x)' = -\text{csch}^2 x$

$(\text{sech } x)' = -\text{sech } x \tanh x$

$(\text{csch } x)' = -\text{csch } x \coth x$

$(\text{arcsinh } x)' = \frac{1}{\sqrt{1+x^2}}$

$(\text{arccosh } x)' = \frac{1}{\sqrt{x^2-1}}$

$(\text{arctanh } x)' = \frac{1}{1-x^2}$

$(\text{arccoth } x)' = \frac{1}{x^2-1}$

$(\text{arccsch } x)' = -\frac{1}{|x|\sqrt{1+x^2}}$

$(\text{arcsech } x)' = -\frac{1}{x\sqrt{1-x^2}}$

8.3 Integration Table 积分表

8.3.1 $ax^2 + bx + c (a > 0)$

1. $\int \frac{dx}{ax^2+bx+c} = \begin{cases} \frac{2}{\sqrt{4ac-b^2}} \arctan \frac{2ax+b}{\sqrt{4ac-b^2}} + C & (b^2 < 4ac) \\ \frac{1}{\sqrt{b^2-4ac}} \ln \left| \frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}} \right| + C & (b^2 > 4ac) \end{cases}$

2. $\int \frac{x}{ax^2+bx+c}dx = \frac{1}{2a} \ln|ax^2+bx+c| - \frac{b}{2a} \int \frac{dx}{ax^2+bx+c}$

8.3.2 $\sqrt{\pm ax^2 + bx + c}(a > 0)$

1. $\int \frac{dx}{\sqrt{ax^2+bx+c}} = \frac{1}{\sqrt{a}} \ln|2ax + b + 2\sqrt{a}\sqrt{ax^2+bx+c}| + C$

2. $\int \sqrt{ax^2+bx+c}dx = \frac{2ax+b}{4a}\sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8\sqrt{a^3}} \ln|2ax + b + 2\sqrt{a}\sqrt{ax^2+bx+c}| + C$

3. $\int \frac{x}{\sqrt{ax^2+bx+c}}dx = \frac{1}{a}\sqrt{ax^2+bx+c} - \frac{b}{2\sqrt{a^3}} \ln|2ax + b + 2\sqrt{a}\sqrt{ax^2+bx+c}| + C$

4. $\int \frac{dx}{\sqrt{c+bx-ax^2}} = -\frac{1}{\sqrt{a}} \arcsin \frac{2ax-b}{\sqrt{b^2+4ac}} + C$

5. $\int \sqrt{c+bx-ax^2}dx = \frac{2ax-b}{4a}\sqrt{c+bx-ax^2} + \frac{b^2+4ac}{8\sqrt{a^3}} \arcsin \frac{2ax-b}{\sqrt{b^2+4ac}} + C$

6. $\int \frac{x}{\sqrt{c+bx-ax^2}}dx = -\frac{1}{a}\sqrt{c+bx-ax^2} + \frac{b}{2\sqrt{a^3}} \arcsin \frac{2ax-b}{\sqrt{b^2+4ac}} + C$

8.3.3 $\sqrt{\pm\frac{x-a}{x-b}}$或$\sqrt{(x-a)(x-b)}$

1. $\int \frac{dx}{\sqrt{(x-a)(b-x)}} = 2\arcsin \sqrt{\frac{x-a}{b-x}} + C \ (a < b)$

2. $\int \sqrt{(x-a)(b-x)}dx = \frac{2x-a-b}{4}\sqrt{(x-a)(b-x)} + \frac{(b-a)^2}{4} \arcsin \sqrt{\frac{x-a}{b-x}} + C, (a < b)$

8.3.4 三角函数的积分

1. $\int \tan x\, dx = -\ln|\cos x| + C$

2. $\int \cot x\, dx = \ln|\sin x| + C$

3. $\int \sec x\, dx = \ln \left| \tan\left(\frac{\pi}{4} + \frac{x}{2}\right) \right| + C = \ln|\sec x + \tan x| + C$

4. $\int \csc x\, dx = \ln \left| \tan \frac{x}{2} \right| + C = \ln|\csc x - \cot x| + C$

5. $\int \sec^2 x\, dx = \tan x + C$

6. $\int \csc^2 x\, dx = -\cot x + C$

7. $\int \sec x \tan x\, dx = \sec x + C$

8. $\int \csc x \cot x\, dx = -\csc x + C$

9. $\int \sin^2 x\, dx = \frac{x}{2} - \frac{1}{4} \sin 2x + C$

10. $\int \cos^2 x\, dx = \frac{x}{2} + \frac{1}{4} \sin 2x + C$

11. $\int \sin^n x\, dx = -\frac{1}{n} \sin^{n-1} x \cos x + \frac{n-1}{n} \int \sin^{n-2} x\, dx$

12. $\int \cos^n x\, dx = \frac{1}{n} \cos^{n-1} x \sin x + \frac{n-1}{n} \int \cos^{n-2} x\, dx$

13. $\int \frac{dx}{\sin^n x} = -\frac{1}{n-1} \frac{\cos x}{\sin^{n-1} x} + \frac{n-2}{n-1} \int \frac{dx}{\sin^{n-2} x}$

14. $\int \frac{dx}{\cos^n x} = \frac{1}{n-1} \frac{\sin x}{\cos^{n-1} x} + \frac{n-2}{n-1} \int \frac{dx}{\cos^{n-2} x}$

15.

$$\int \cos^m x \sin^n x \mathrm{d}x$$

$$= \frac{1}{m+n} \cos^{m-1} x \sin^{n+1} x + \frac{m-1}{m+n} \int \cos^{m-2} x \sin^n x \mathrm{d}x$$

$$= -\frac{1}{m+n} \cos^{m+1} x \sin^{n-1} x + \frac{n-1}{m+1} \int \cos^m x \sin^{n-2} x \mathrm{d}x$$

16. $\int \frac{\mathrm{d}x}{a+b\sin x} = \begin{cases} \frac{2}{\sqrt{a^2-b^2}} \arctan \frac{a\tan\frac{x}{2}+b}{\sqrt{a^2-b^2}} + C & (a^2 > b^2) \\ \frac{1}{\sqrt{b^2-a^2}} \ln \left| \frac{a\tan\frac{x}{2}+b-\sqrt{b^2-a^2}}{a\tan\frac{x}{2}+b+\sqrt{b^2-a^2}} \right| + C & (a^2 < b^2) \end{cases}$

17. $\int \frac{\mathrm{d}x}{a+b\cos x} = \begin{cases} \frac{2}{a+b} \sqrt{\frac{a+b}{a-b}} \arctan \left(\sqrt{\frac{a-b}{a+b}} \tan \frac{x}{2} \right) + C & (a^2 > b^2) \\ \frac{1}{a+b} \sqrt{\frac{a+b}{a-b}} \ln \left| \frac{\tan\frac{x}{2}+\sqrt{\frac{a+b}{b-a}}}{\tan\frac{x}{2}-\sqrt{\frac{a+b}{b-a}}} \right| + C & (a^2 < b^2) \end{cases}$

18. $\int \frac{\mathrm{d}x}{a^2\cos^2 x + b^2\sin^2 x} = \frac{1}{ab} \arctan \left(\frac{b}{a}\tan x \right) + C$

19. $\int \frac{\mathrm{d}x}{a^2\cos^2 x - b^2\sin^2 x} = \frac{1}{2ab} \ln \left| \frac{b\tan x+a}{b\tan x-a} \right| + C$

20. $\int x\sin ax\mathrm{d}x = \frac{1}{a^2}\sin ax - \frac{1}{a}x\cos ax + C$

21. $\int x^2\sin ax\mathrm{d}x = -\frac{1}{a}x^2\cos ax + \frac{2}{a^2}x\sin ax + \frac{2}{a^3}\cos ax + C$

22. $\int x\cos ax\mathrm{d}x = \frac{1}{a^2}\cos ax + \frac{1}{a}x\sin ax + C$

23. $\int x^2\cos ax\mathrm{d}x = \frac{1}{a}x^2\sin ax + \frac{2}{a^2}x\cos ax - \frac{2}{a^3}\sin ax + C$

8.3.5 反三角函数的积分（其中 $a > 0$ ）

1. $\int \arcsin\frac{x}{a}\mathrm{d}x = x\arcsin\frac{x}{a} + \sqrt{a^2-x^2} + C$

2. $\int x\arcsin\frac{x}{a}\mathrm{d}x = \left(\frac{x^2}{2} - \frac{a^2}{4}\right)\arcsin\frac{x}{a} + \frac{x}{4}\sqrt{x^2-x^2} + C$

3. $\int x^2\arcsin\frac{x}{a}\mathrm{d}x = \frac{x^3}{3}\arcsin\frac{x}{a} + \frac{1}{9}(x^2+2a^2)\sqrt{a^2-x^2} + C$

4. $\int \arccos\frac{x}{a}\mathrm{d}x = x\,arccos\frac{x}{a} - \sqrt{a^2-x^2} + C$

5. $\int x\arccos\frac{x}{a}\mathrm{d}x = \left(\frac{x^2}{2} - \frac{a^2}{4}\right)\arccos\frac{x}{a} - \frac{x}{4}\sqrt{a^2-x^2} + C$

6. $\int x^2\arccos\frac{x}{a}\mathrm{d}x = \frac{x^3}{3}\arccos\frac{x}{a} - \frac{1}{9}(x^2+2a^2)\sqrt{a^2-x^2} + C$

7. $\int \arctan\frac{x}{a}\mathrm{d}x = x\arctan\frac{x}{a} - \frac{a}{2}\ln(a^2+x^2) + C$

8. $\int x\arctan\frac{x}{a}\mathrm{d}x = \frac{1}{2}(a^2+x^2)\arctan\frac{x}{a} - \frac{a}{2}x + C$

9. $\int x^2\arctan\frac{x}{a}\mathrm{d}x = \frac{x^3}{3}\arctan\frac{x}{a} - \frac{a}{6}x^2 + \frac{a^3}{6}\ln(a^2+x^2) + C$

8.3.6 指数函数的积分

1. $\int a^x\mathrm{d}x = \frac{1}{\ln a}a^x + C$

2. $\int \mathrm{e}^{ax}\mathrm{d}x = \frac{1}{a}a^{ax} + C$

3. $\int x\mathrm{e}^{ax}\mathrm{d}x = \frac{1}{a^2}(ax-1)a^{ax} + C$

4. $\int x^n\mathrm{e}^{ax}\mathrm{d}x = \frac{1}{a}x^n\mathrm{e}^{ax} - \frac{n}{a}\int x^{n-1}\mathrm{e}^{ax}\mathrm{d}x$

5. $\int xa^x\mathrm{d}x = \frac{x}{\ln a}a^x - \frac{1}{(\ln a)^2}a^x + C$

6. $\int x^na^x\mathrm{d}x = \frac{1}{\ln a}x^na^x - \frac{n}{\ln a}\int x^{n-1}a^x\mathrm{d}x$

7. $\int \mathrm{e}^{ax}\sin bx\mathrm{d}x = \frac{1}{a^2+b^2}\mathrm{e}^{ax}(a\sin bx - b\cos bx) + C$

8. $\int \mathrm{e}^{ax}\cos bx\mathrm{d}x = \frac{1}{a^2+b^2}\mathrm{e}^{ax}(b\sin bx + a\cos bx) + C$

9. $\int \mathrm{e}^{ax}\sin^n bx\mathrm{d}x = \frac{1}{a^2+b^2n^2}\mathrm{e}^{ax}\sin^{n-1} bx(a\sin bx - nb\cos bx) + \frac{n(n-1)b^2}{a^2+b^2n^2}\int \mathrm{e}^{ax}\sin^{n-2} bx\mathrm{d}x$

10. $\int \mathrm{e}^{ax}\cos^n bx\mathrm{d}x = \frac{1}{a^2+b^2n^2}\mathrm{e}^{ax}\cos^{n-1} bx(a\cos bx + nb\sin bx) + \frac{n(n-1)b^2}{a^2+b^2n^2}\int \mathrm{e}^{ax}\cos^{n-2} bx\mathrm{d}x$

8.3.7 对数函数的积分

1. $\int \ln x\mathrm{d}x = x\ln x - x + C$

2. $\int \frac{\mathrm{d}x}{x\ln x} = \ln\left|\ln x\right| + C$

3. $\int x^n\ln x\mathrm{d}x = \frac{1}{n+1}x^{n+1}\left(\ln x - \frac{1}{n+1}\right) + C$

4. $\int(\ln x)^n\mathrm{d}x = x(\ln x)^n - n\int(\ln x)^{n-1}\mathrm{d}x$

5. $\int x^m(\ln x)^n\mathrm{d}x = \frac{1}{m+1}x^{m+1}(\ln x)^n - \frac{n}{m+1}\int x^m(\ln x)^{n-1}\mathrm{d}x$