Contents

李沐阳

Aho-Corasick Automaton

```cpp
// ac 自动机
inline void build()
{
  int now = root; queue <int> team;
  fail[root] = root;
  for (int i = 0;i < 10;++i)
  {
    if (nxt[now][i] == -1) nxt[now][i] = root;
    else fail[nxt[now][i]] = root,team.push(nxt[now][i]);
  }
  while (!team.empty())
  {
    now = team.front(); team.pop();
    for (int i = 0;i < 10;++i)
    {
      if (nxt[now][i] == -1)
        nxt[now][i] = nxt[fail[now]][i];
      else
      {
        fail[nxt[now][i]] = nxt[fail[now]][i];
        team.push(nxt[now][i]);
      }
    }
  }
}
```

Bridges and Cut Vertices

```cpp
// 求割边和割点
const int maxn = 200010;
int N,M,cnt,Ts,dfn[maxn],low[maxn],nxt[maxn];
int toit[maxn],side[maxn];
bool bridge[maxn],cut[maxn];

inline void dfs(int now,int fa)
{
  dfn[now] = low[now] = ++Ts; int child = 0;
  for (int i = side[now];i;i = nxt[i])
  {
    if (toit[i] == fa) continue;
    if (!dfn[toit[i]])
    {
      dfs(toit[i],now); ++child;
      low[now] = min(low[now],low[toit[i]]);
      if (low[toit[i]] > dfn[now]) bridge[i] = true;
      if (low[toit[i]] >= dfn[now]) cut[now] = true;
    }
    else low[now] = min(low[now],dfn[toit[i]]);
  }
  if (!fa&&child == 1) cut[now] = false;
}
```

Circle Intersection

```
//modified
const double eps = 1e-7,pi = acos(-1.0);
int N,M; double area[maxn]; // area[k] -> area of intersections >= k.

struct Point
{
  double x,y;
  inline double angle() const { return atan2(y,x); }
};
struct Circle
{
  Point C; double r; int sgn;
  inline Circle() = default;
  inline Circle(const Point &_C,double _r,int _sgn):C(_C),r(_r),sgn(_sgn) {} // sgn 代表该圆的权值，默认 1
  friend inline bool operator==(const Circle &a,const Circle &b)
  {
    if (dcmp(a.r-b.r)) return false;
    if (dcmp(a.C.x-b.C.x)) return false;
    if (dcmp(a.C.y-b.C.y)) return false;
    if (a.sgn != b.sgn) return false;
    return true;
  }
  friend inline bool operator!=(const Circle &a,const Circle &b) { return !(a == b); }
}cir[maxn];

inline Point rotate(const Point &p,double cost,double sint)
{
  double x = p.x,y = p.y;
  return Point(x*cost-y*sint,x*sint+y*cost);
}
inline pair <Point,Point> crosspoint(const Point &ap,double ar,const Point &bp,double br)
{
  double d = (ap-bp).norm(),cost = (ar*ar+d*d-br*br)/(2*ar*d),sint = sqrt(1-cost*cost);
  Point v = ((bp-ap).unit())*ar;
  return make_pair(ap+rotate(v,cost,-sint),ap+rotate(v,cost,sint));
}
inline pair <Point,Point> crosspoint(const Circle &a,const Circle &b) { return crosspoint(a.C,a.r,b.C,b.r); }

inline bool overlap(const Circle &a,const Circle &b) { return dcmp(a.r-b.r-(a.C-b.C).norm()) >= 0; } // b 是不是在 a 里面
inline bool intersect(const Circle &a,const Circle &b)
{
  if (overlap(a,b)) return false;
  if (overlap(b,a)) return false;
  return dcmp((a.C-b.C).norm()-a.r-b.r) < 0;
}

struct Event
{
  Point p; double a; int d;
  inline Event() = default;
  inline Event(const Point &_p,double _a,double _d):p(_p),a(_a),d(_d) {}
  friend inline bool operator <(const Event &a,const Event &b) { return a.a < b.a; }
};

inline void solve()
{
  for (int i = 1;i <= M;++i) area[i] = 0;
  for (int i = 1;i <= M;++i)
  {
    int cnt = cir[i].sgn; if (cnt<0) cnt = 0; vector <Event> event;
    for (int j = 1;j < i;++j) if (cir[i] == cir[j]) cnt += cir[j].sgn;
    for (int j = 1;j <= M;++j)
      if (j != i&&cir[i] != cir[j]&&overlap(cir[j],cir[i])) cnt += cir[j].sgn;
    for (int j = 1;j <= M;++j)
      if (j != i&&intersect(cir[i],cir[j]))
      {
        pair <Point,Point> res = crosspoint(cir[i],cir[j]); swap(res.first,res.second);
        double alpha1 = (res.first-cir[i].C).angle(),alpha2 = (res.second-cir[i].C).angle();
        event.push_back(Event(res.second,alpha2,cir[j].sgn));
        event.push_back(Event(res.first,alpha1,-cir[j].sgn));
        cnt += (alpha2 > alpha1)*cir[j].sgn;
      }
    if (!event.size()) area[cnt] += pi*cir[i].r*cir[i].r*cir[i].sgn;
    else
    {
      sort(event.begin(),event.end());
      event.push_back(event.front());
      for (int j = 0;j+1 < (int)event.size();++j)
      {
        cnt += event[j].d;
        area[cnt] += event[j].p/event[j+1].p/2*cir[i].sgn;
        double alpha = event[j+1].a-event[j].a;
        if (alpha < 0) alpha += 2*pi;
        if (!dcmp(alpha)) continue;
        area[cnt] += alpha*cir[i].r*cir[i].r/2*cir[i].sgn;
        area[cnt] += -sin(alpha)*cir[i].r*cir[i].r/2*cir[i].sgn;
      }
    }
  }
}

// origin
struct Event {
  Point p;
  double ang;
  int delta;
  Event (Point p = Point(0, 0), double ang = 0, double delta = 0) : p(p), ang(ang), delta(delta) {}
};
bool operator < (const Event &a, const Event &b) {
  return a.ang < b.ang;
}
void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
  double d2 = (a.o - b.o).len2(),
       dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
       pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4));
  Point d = b.o - a.o, p = d.rotate(PI / 2),
      q0 = a.o + d * dRatio + p * pRatio,
      q1 = a.o + d * dRatio - p * pRatio;
  double ang0 = (q0 - a.o).ang(),
       ang1 = (q1 - a.o).ang();
  evt.push_back(Event(q1, ang1, 1));
  evt.push_back(Event(q0, ang0, -1));
  cnt += ang1 > ang0;
}
bool issame(const Circle &a, const Circle &b) { return sign((a.o - b.o).len()) == 0 && sign(a.r - b.r) == 0; }
bool overlap(const Circle &a, const Circle &b) { return sign(a.r - b.r - (a.o - b.o).len()) >= 0; }
bool intersect(const Circle &a, const Circle &b) { return sign((a.o - b.o).len() - a.r - b.r) < 0; }
Circle c[N];
double area[N];  // area[k] -> area of intersections >= k.
Point centroid[N];  //k 次圆的质心
bool keep[N];
void add(int cnt, DB a, Point c) {
  area[cnt] += a;
  centroid[cnt] = centroid[cnt] + c * a;
}
void solve(int C) {
  for (int i = 1; i <= C; ++ i) {
        area[i] = 0;
        centroid[i] = Point(0, 0);
    }
  for (int i = 0; i < C; ++i) {
    int cnt = 1;
    vector<Event> evt;
    for (int j = 0; j < i; ++j) if (issame(c[i], c[j])) ++cnt;
    for (int j = 0; j < C; ++j) {
      if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i])) {
        ++cnt;
      }
    }
    for (int j = 0; j < C; ++j) {
      if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i], c[j])) {
        addEvent(c[i], c[j], evt, cnt);
      }
    }
    if (evt.size() == 0u) {
      add(cnt, PI * c[i].r * c[i].r, c[i].o);
    } else {
      sort(evt.begin(), evt.end());
      evt.push_back(evt.front());
      for (int j = 0; j + 1 < (int)evt.size(); ++j) {
        cnt += evt[j].delta;
        add(cnt, det(evt[j].p, evt[j + 1].p) / 2, (evt[j].p + evt[j + 1].p) / 3);
        double ang = evt[j + 1].ang - evt[j].ang;
        if (ang < 0) {
          ang += PI * 2;
        }
                if (sign(ang) == 0) continue;
        double ang0 = evt[j].a,ang1 = evt[j+1].a;
                add(cnt, ang * c[i].r * c[i].r / 2, c[i].o +
                    Point(sin(ang1) - sin(ang0), -cos(ang1) + cos(ang0)) * (2 / (3 * ang) * c[i].r));
        add(cnt, -sin(ang) * c[i].r * c[i].r / 2, (c[i].o + evt[j].p + evt[j + 1].p) / 3);
      }
    }
  }
    for (int i = 1; i <= C; ++ i)
    if (sign(area[i])) {
      centroid[i] = centroid[i] / area[i];
    }
}
```

Counting Integral Points under Straight Line

```
// \sum_{i = 0}^{n-1} (a+bi)/m
inline ll count(ll n,ll a,ll b,ll m)
{
  if (!b) return n*(a/m);
  else if (a >= m) return n*(a/m)+count(n,a%m,b,m);
```

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

Cross Points of Circles

```
//圆圆求交，需先判定两圆有交
inline pair <Point,Point> CrossPoint(const Point &ap,double ar,const Point &bp,double br)
{
  double d = (ap-bp).norm();
  double cost = (ar*ar+d*d-br*br)/(2*ar*d),sint = sqrt(1-cost*cost);
  Point v = ((bp-ap)/(bp-ap).norm())*ar;
  return make_pair(ap+rotate(v,cost,-sint),ap+rotate(v,cost,sint));
}
```

Cross Points of Line and Circle

```
//a b 直线两点，o 圆心
//若 a b 为线段，则 0 <= t1,t2 <= 1
inline void CrossPoint(const Point &a,const Point &b,const Point &o,double r,Point *ret,int &num)
{
  double X0 = o.x,Y0 = o.y;
  double X1 = a.x,Y1 = a.y;
  double X2 = b.x,Y2 = b.y;
  double dx = X2-X1,dy = Y2-Y1;
  double A = dx*dx+dy*dy;
  double B = 2*dx*(X1-X0)+2*dy*(Y1-Y0);
  double C = (X1-X0)*(X1-X0)+(Y1-Y0)*(Y1-Y0)-r*r;
  double delta = B*B-4*A*C+eps;
  num = 0;
  if (delta >= 0)
  {
    double t1 = (-B-sqrt(delta))/(2*A);
    double t2 = (-B+sqrt(delta))/(2*A);
    ret[++num] = Point(X1+t1*dx,Y1+t1*dy);
    ret[++num] = Point(X1+t2*dx,Y1+t2*dy);
  }
}
```

Dominator Tree

```
// 例题：询问 i 号点到 N 号点所有必经点编号和
int N,M,Ts,cnt,side[maxn],nxt[maxn],toit[maxn],dfn[maxn],redfn[maxn],idom[maxn],best[maxn],semi[maxn];
int ans[maxn],anc[maxn],fa[maxn],child[maxn],size[maxn]; vector <int> prod[maxn],bucket[maxn],son[maxn];

inline void init()
{
  cnt = 1; memset(side,0,sizeof side); memset(ans,0,sizeof ans);
  for (int i = 0;i <= N;++i) prod[i].clear(),bucket[i].clear(),son[i].clear();
}

inline void add(int a,int b) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }

inline void dfs(int now)
{
  dfn[now] = ++Ts; redfn[Ts] = now;
  anc[Ts] = idom[Ts] = child[Ts] = size[Ts] = 0;
  semi[Ts] = best[Ts] = Ts;
  for (int i = side[now];i;i = nxt[i])
  {
    if (!dfn[toit[i]])
      dfs(toit[i]),fa[dfn[toit[i]]] = dfn[now];
    prod[dfn[toit[i]]].push_back(dfn[now]);
  }
}

inline void compress(int now)
{
  if (anc[anc[now]] != 0)
  {
    compress(anc[now]);
    if (semi[best[now]] > semi[best[anc[now]]])
      best[now] = best[anc[now]];
    anc[now] = anc[anc[now]];
  }
}

inline int eval(int now)
{
  if (!anc[now]) return now;
  else
  {
    compress(now);
    return semi[best[anc[now]]] >= semi[best[now]]?best[now]:best[anc[now]];
  }
}

inline void link(int v,int w)
{
  int s = w;
  while (semi[best[w]] < semi[best[child[w]]])
  {
    if (size[s]+size[child[child[s]]] >= 2*size[child[s]])
      anc[child[s]] = s,child[s] = child[child[s]];
    else size[child[s]] = size[s],s = anc[s] = child[s];
  }
  best[s] = best[w]; size[v] += size[w];
  if (size[v] < 2*size[w]) swap(s,child[v]);
  while (s) anc[s] = v,s = child[s];
}

inline void lengauer_tarjan()
{
  memset(dfn,0,sizeof dfn); memset(fa,-1,sizeof fa); Ts = 0;
  dfs(N); fa[1] = 0;
  for (int w = Ts;w > 1;--w)
  {
    for (auto x:prod[w])
    {
      int u = eval(x);
      if (semi[w] > semi[u]) semi[w] = semi[u];
    }
    bucket[semi[w]].push_back(w);
    link(fa[w],w); if (!fa[w]) continue;
    for (auto x:bucket[fa[w]])
    {
      int u = eval(x);
      if (semi[u] < fa[w]) idom[x] = u;
      else idom[x] = fa[w];
    }
    bucket[fa[w]].clear();
  }
  for (int w = 2;w <= Ts;++w)
    if (idom[w] != semi[w])
      idom[w] = idom[idom[w]];
  idom[1] = 0;
  for (int i = Ts;i > 1;--i)
  {
    if (fa[i] == -1) continue;
    son[idom[i]].push_back(i);
  }
}

inline void get_ans(int now)
{
  ans[redfn[now]] += redfn[now];
  for (auto x:son[now])
    ans[redfn[x]] += ans[redfn[now]],get_ans(x);
}

int main()
{
  while (scanf("%d %d",&N,&M) != EOF)
  {
    init();
    for (int i = 1,a,b;i <= M;++i)
      a = gi(),b = gi(),add(a,b);
    lengauer_tarjan(); get_ans(1);
    for (int i = 1;i <= N;++i)
      printf("%d%c",ans[i]," \n"[i == N]);
  }
  return 0;
}
```

Graham Scanning Algorithm

```
//凸包上最大四边形面积
int N,M; double ans;
struct Point { double x,y; }P[maxn],convex[maxn];
inline void Graham() {
  ConvexHull();
  for (int i = 1;i <= M;++i) convex[i+M] = convex[i];
  int p1,p2,p3,p4;
  for (p1 = 1;p1 <= M;++p1) {
    p2 = p1+1; p3 = p2+1; p4 = p3+1;
    for (;p3 < p1+M-1;++p3) {
      Point v = convex[p3]-convex[p1];
      while (p2 < p3&&fabs((convex[p2]-convex[p1])/v) < fabs((convex[p2+1]-convex[p1])/v)) ++p2;
      while (p4 < p1+M&&fabs((convex[p4]-convex[p1])/v) < fabs((convex[p4+1]-convex[p1])/v)) ++p4;
      ans = max(ans,fabs((convex[p2]-convex[p1])/v)+fabs((convex[p4]-convex[p1])/v));
    }
  }
  ans = ans/2;
}
```

Gray Code

```
//0-2^n-1 的格雷码，相邻两个二进制位中，恰好只有一位不同
inline vector <int> GrayCreat(int n)
{
```

```
  vector <int> res;
  for (int i = 0;i < (1<<n);++i) res.push_back(i^(i>>1));
  return res;
}
```

Half Plane Intersection

```
//半平面交，直线左侧半平面，注意最后是 tail-head <= 0 还是 tail-head <= 1
struct Point
{
  double x,y;
  inline double angle() const { return atan2(y,x); }
}P[maxn],pp[maxn],pol[maxn];

struct Line
{
  Point p,v;
  inline double slop() const { return v.angle(); }
  friend inline bool operator<(const Line &a,const Line &b) { return a.slop() < b.slop(); }
}line[maxn],qq[maxn];

inline bool onleft(const Line &L,const Point &p) { return dcmp(L.v/(p-L.p)) > 0; }

inline int half_plane_intersection()
{
    sort(lines+1,lines+tot+1);  //直线按斜率排序
    int head,tail;
    qq[head = tail = 1] = lines[1];
    for (int i = 2;i <= tot;++i)
    {
        while (head < tail&&!onleft(lines[i],pp[tail-1])) --tail;
        while (head < tail&&!onleft(lines[i],pp[head])) ++head;
        qq[++tail] = lines[i];
        if (parallel(qq[tail],qq[tail-1]))
        {
            tail--;
            if (onleft(qq[tail],lines[i].p)) qq[tail] = lines[i];
        }
        if (head < tail) pp[tail-1] = crosspoint(qq[tail],qq[tail-1]);
    }
    while (head < tail && !onleft(qq[head],pp[tail-1])) --tail;
    if (tail-head <= 0) return 0;
    pp[tail] = crosspoint(qq[tail],qq[head]);
    for (int i = head;i <= tail;++i) pol[++m] = pp[i];  //半平面交点
    pol[0] = pol[m];
    return m;
}
```

Hungary

```
inline int find(int x)
{
    for (int i = 1;i <= n;++i)
        if (w[x][i]&&!used[i])
        {
            used[i] = true;
            if (!from[i]||match(from[i])) { from[i] = x; return true; }
        }
    return false;
}

inline int hungry()
{
    int ret = 0; memset(from,-1,sizeof from);
    for (int i = 1;i <= n;++i)
    {
        memset(used,false,sizeof(used));
        if (match(i)) ret++;
    }
    return ret;
}
```

Intersecting Area of Circle and Polygon

```
const int maxn = 510;
const double eps = 1e-9;

struct Point { double x,y; }P[maxn],A,B;
int N; double K;

inline double getSectorArea(const Point &a,const Point &b,double r)
{
  double c = (2*r*r-((a-b)*(a-b)))/(2*r*r);
  double alpha = acos(c);
  return r*r*alpha/2.0;
}

inline pair <double,double> getSolution(double a,double b,double c)
{
  double delta = b*b-4*a*c;
  if (dcmp(delta) < 0) return make_pair(0,0);
  else return make_pair((-b-sqrt(delta))/(2*a),(-b+sqrt(delta))/(2*a));
}

inline pair <Point,Point> getIntersection(const Point &a,const Point &b,double r)
{
  Point d = b-a;
  double A = d*d,B = 2*(d*a),C = (a*a)-r*r;
  pair <double,double> s = getSolution(A,B,C);
  return make_pair(a+(d*s.first),a+(d*s.second));
}

inline double getPointDist(const Point &a,const Point &b)
{
  Point d = b-a;
  int sA = dcmp(a*d),sB = dcmp(b*d);
  if (sA*sB <= 0) return (a/b)/((a-b).norm());
  else return min(a.norm(),b.norm());
}

double getArea(const Point &a,const Point &b,double r)
{
  double dA = a*a,dB = b*b,dC = getPointDist(a,b),ans = 0;
  if (dcmp(dA-r*r) <= 0&&dcmp(dB-r*r) <= 0) return (a/b)/2;
  Point tA = a.unit()*r,tB = b.unit()*r;
  if (dcmp(dC-r) > 0) return getSectorArea(tA,tB,r);
  pair <Point,Point> ret = getIntersection(a,b,r);
  if (dcmp(dA-r*r) > 0&&dcmp(dB-r*r) > 0)
  {
    ans += getSectorArea(tA,ret.first,r);
    ans += (ret.first/ret.second)/2;
    ans += getSectorArea(ret.second,tB,r);
    return ans;
  }
  if (dcmp(dA-r*r) > 0) return (ret.first/b)/2+getSectorArea(tA,ret.first,r);
  else return (a/ret.second)/2.0+getSectorArea(ret.second,tB,r);
}

double getArea(int n,Point *p,const Point &c,double r)
{
  double ret = 0;
  for (int i = 0;i < n;++i)
  {
    int sgn = dcmp((p[i]-c)/(p[(i+1)%n]-c));
    if (sgn > 0) ret += getArea(p[i]-c,p[(i+1)%n]-c,r);
    else ret -= getArea(p[(i+1)%n]-c,p[i]-c,r);
  }
  return fabs(ret);
}
```

Intersection of Line and Convex Hull

```
//O(logN)
inline double getA(const Node &a)
{
  double ret = atan2(a.y,a.x);
  if (ret <= -pi/2) ret += 2*pi;
  return ret;
}

inline int find(double x)
{
  if (x <= w[1]||x >= w[m]) return 1;
  return upper_bound(w+1,w+m+1,x)-w;
}

inline bool intersect(const Node &a,const Node &b)
{
  int i = find(getA(b-a)),j = find(getA(a-b));
  if (dcmp((b-a)/(convex[i]-a))*dcmp((b-a)/(convex[j]-a)) > 0) return false;
  else return true;
}

inline int prework()
{
  ConvexConvex(); convex[m+1] = convex[1];
  for (int i = 1;i <= m;++i)
    w[i]= getA(convex[i+1]-convex[i]);
}
```

Kuhn-Munkres Algorithm

```
// Truly O(n^3), 最大权匹配
// 邻接矩阵，不能连的边设为-INF，求最小权匹配时边权取负，但不能连的还是 -INF，使用时先对 1 -> n 调用 hungary()，再 get_ans() 求值
struct KM
{
  int w[maxn][maxn],lx[maxn],ly[maxn],match[maxn],way[maxn],slack[maxn];
  bool used[maxn];

  inline void init()
```

```
  {
    for (int i = 1;i <= N;++i)
      match[i] = lx[i] = ly[i] = way[i] = 0;
  }

  inline void hungary(int x)
  {
    match[0] = x; int j0 = 0;
    for (int j = 0;j <= N;++j)
      slack[j] = inf,used[j] = false;
    do
    {
      used[j0] = true;
      int i0 = match[j0],delta = inf,j1 = 0;
      for (int j = 1;j <= N;++j)
        if (!used[j])
        {
          int cur = -w[i0][j]-lx[i0]-ly[j];
          if (cur < slack[j])
            slack[j] = cur,way[j] = j0;
          if (slack[j] < delta)
            delta = slack[j],j1 = j;
        }
      for (int j = 0;j <= N;++j)
      {
        if (used[j]) lx[match[j]] += delta,ly[j] -= delta;
        else slack[j] -= delta;
      }
      j0 = j1;
    }
    while (match[j0]);
    do
    {
      int j1 = way[j0];
      match[j0] = match[j1];
      j0 = j1;
    }
    while (j0);
  }

  inline void work() { for (int i = 1;i <= N;++i) hungary(i); }

  inline int get_ans()
  {
    int sum = 0;
    for (int i = 1;i <= N;++i)
    {
      // if (w[match[i]][i] == -inf) ; //无解
      if (match[i] > 0) sum += w[match[i]][i];
    }
    return sum;
  }
}km;
```

Link Cut Tree

```
inline bool isroot(int a) { return ch[fa[a]][0] != a&&ch[fa[a]][1] != a; }

inline void update(int x) { val[x] = (val[ch[x][0]]+val[ch[x][1]]).merge(x); }
inline void pushdown(int x)
{
  if (rev[x])
  {
    int &lc = ch[x][0],&rc = ch[x][1];
    swap(lc,rc);
    if (lc) rev[lc] ^= 1;
    if (rc) rev[rc] ^= 1;
    rev[x] = false;
  }
}

inline void rotate(int x)
{
  int y = fa[x],z = fa[y],l = ch[y][1] == x,r = l^1;
  if (!isroot(y)) ch[z][ch[z][1] == y] = x; fa[x] = z;
  if (ch[x][r]) fa[ch[x][r]] = y; ch[y][l] = ch[x][r];
  fa[y] = x; ch[x][r] = y; update(y); update(x);
}
inline void splay(int x)
{
  int top = 0,i;
  for (i = x;!isroot(i);i = fa[i]) stk[++top] = i; stk[++top] = i;
  while (top) pushdown(stk[top--]);
  while (!isroot(x))
  {
    int y = fa[x],z = fa[y];
    if (!isroot(y))
    {
      if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
      else rotate(y);
    }
    rotate(x);
  }
}

inline int access(int x)
{
  int t = 0;
  for (t = 0;x;t = x,x = fa[x])
    splay(x),ch[x][1] = t,update(x);
  return t;
}
inline int evert(int x) { int t; rev[t = access(x)] ^= 1; return t; }
inline int find(int x)
{
  x = access(x);
  while (pushdown(x),ch[x][0]) x = ch[x][0];
  return x;
}
inline void cut(int x,int y)
{
  evert(x); access(y); splay(y);
  if (ch[y][0] != x||ch[x][1] != 0) return;
  ch[y][0] = fa[x] = 0; update(x); update(y);
}
inline void link(int x,int y) { fa[evert(x)] = y; }
```

Maximal Matching in General Graphs

```
// 接口 int matching(), 返回最大匹配数,G 为邻接矩阵
inline void push(int x)
{
  team.push(x); check[x] = true;
  if (!treec[x]) tra[++cnt] = x,treec[x] = true;
}
inline int root(int x) { return f[x]?f[x] = root(f[x]):x; }

inline void clear()
{
  for (int i = 1,j;i <= cnt;++i)
  {
    j = tra[i]; father[j] = 0,f[j] = 0;
    check[j] = treec[j] = false;
  }
}

inline int lca(int u,int v)
{
  int len = 0;
  for (;u;u = father[match[u]])
    pathc[path[++len] = u = root(u)] = true;
  for (;;v = father[match[v]])
    if (pathc[v = root(v)]) break;
  for (int i = 1;i <= len;++i)
    pathc[path[i]] = false;
  return v;
}

inline void reset(int u,int p)
{
  for (int v;root(u) != p;)
  {
    if (!check[v = match[u]]) push(v);
    if (!f[u]) f[u] = p; if (!f[v]) f[v] = p;
    u = father[v]; if (root(u) != p) father[u] = v;
  }
}

inline void flower(int u,int v)
{
  int p = lca(u,v);
  if (root(u) != p) father[u] = v;
  if (root(v) != p) father[v] = u;
  reset(u,p); reset(v,p);
}

inline bool find(int x)
{
  while (!team.empty()) team.pop();
  cnt = 0; push(x);
  while (!team.empty())
  {
    int i = team.front(); team.pop();
    for (int j = 1;j <= N;++j)
      if (G[i][j]&&root(i) != root(j)&&match[j] != i)
      {
        if (match[j]&&father[match[j]]) flower(i,j);
        else if (!father[j])
        {
```

```cpp
          father[tra[++cnt] = j] = i; treec[j] = true;
          if (match[j]) push(match[j]);
          else
          {
            for (int k = i,l = j,p;k;l = p,k = father[l])
              p = match[k],match[k] = l,match[l] = k;
            return true;
          }
        }
      }
  }
  return false;
}

inline int matching()
{
  memset(father,0,sizeof father);  memset(f,0,sizeof f); memset(path,0,sizeof path);
  memset(tra,0,sizeof tra); memset(match,0,sizeof match); memset(check,false,sizeof check);
  memset(treec,false,sizeof treec); memset(pathc,false,sizeof pathc);
  int ret = cnt = 0;
  for (int i = 1;i <= N;++i)
  {
    if (match[i]) continue;
    if (find(i)) ++ret; clear();
  }
  return ret;
}
```

Merge Split Treap

```cpp
// jisuanke17123
// Warning: 给指针赋值时，不要赋 this，因为 this 是临时变量的地址
inline int rand(int n) { int x = rand(); if (x < 0) x = -x; return x%n+1; }

struct Node
{
  int size,key,val; Node *mn,*ch[2];
  inline Node *update()
  {
    mn = this; size = 1;
    if (ch[0])
    {
      size += ch[0]->size;
      if (ch[0]->mn->val < mn->val) mn = ch[0]->mn;
    }
    if (ch[1])
    {
      size += ch[1]->size;
      if (ch[1]->mn->val < mn->val) mn = ch[1]->mn;
    }
    return this;
  }
  inline Node() = default;
  inline Node(int v,Node *_mn):size(1),key(rand()),val(v),mn(_mn) { ch[0] = ch[1] = NULL; }
}pool[maxn*100/4],*root[maxn],*cur;
struct Status
{
  int l,r; ll val;
  inline Status() = default;
  inline Status(int _l,int _r,ll _val):l(_l),r(_r),val(_val) {}
  friend inline bool operator <(const Status &a,const Status &b) { return a.val > b.val; }
};

inline int sz(const Node *x) { if (x == NULL) return 0; else return x->size; }

inline Node *newnode(int v = 0) { *cur = Node(v,cur); return cur++; }

Node *insert(Node *p,Node *q)
{
  if (p == NULL&&q == NULL) return NULL;
  if (p == NULL||q == NULL) return p?p:q;
  Node *u = NULL;
  if (rand(sz(p)+sz(q)) < sz(p))
    u = p,u->ch[1] = insert(u->ch[1],q);
  else u = q,u->ch[0] = insert(p,u->ch[0]);
  return u->update();
}

Node *merge(Node *p,Node *q)
{
  if (p == NULL&&q == NULL) return NULL;
  if (p == NULL||q == NULL) return p?p:q;
  Node *u = newnode();
  if (rand(sz(p)+sz(q)) < sz(p))
    *u = *p,u->ch[1] = merge(u->ch[1],q);
  else *u = *q,u->ch[0] = merge(p,u->ch[0]);
  return u->update();
}

Node *split(Node *u,int l,int r)
{
  if (l > r||u == NULL) return 0;
  Node *x = NULL;
  if (l == 1&&r == sz(u))
  {
    x = newnode(); *x = *u;
    return x->update();
  }
  int lsz = sz(u->ch[0]);
  if (r <= lsz) return split(u->ch[0],l,r);
  if (l > lsz+1) return split(u->ch[1],l-lsz-1,r-lsz-1);
  x = newnode(); *x = *u;
  x->ch[0] = split(u->ch[0],l,lsz);
  x->ch[1] = split(u->ch[1],1,r-lsz-1);
  return x->update();
}

int get_pos(Node *rt,Node *mn)
{
  if (rt == mn) return sz(rt->ch[0]);
  else if (rt->ch[0]&&rt->ch[0]->mn == mn)
    return get_pos(rt->ch[0],mn);
  else return sz(rt->ch[0])+1+get_pos(rt->ch[1],mn);
}
inline pair <int,int> Qmin(Node *rt,int l,int r)
{
  if (l > r) return make_pair(-1,-1);
  Node *v = split(rt,l,r);
  auto ret = make_pair(v->mn->val,get_pos(v,v->mn)+1);
  return ret;
}
inline int get(Node *u,int x) { return split(u,x,x)->val; }

inline void init() { cur = pool; }

int main()
{
  struct timeb ttt; ftime(&ttt);
  srand(ttt.millitm+ttt.time*1000);
}
```

Modui Algorithm on Tree

```cpp
// 询问树上路径元素 mex，inc dec 复杂度不对，需要用线段树/set(带 log) 或者分块 (修改 O(1))
// 若包括 lca，每组询问需要把 lca 补 (inc) 上去。
const int Size = 337,maxn = 200010;
int N,Q,cnt,nxt[maxn],side[maxn],len[maxn],toit[maxn],f[maxn][20],key[maxn],timestamp;
int dep[maxn],L[maxn],R[maxn],dfn[maxn],ans[maxn],exist[maxn],show[maxn],res;
struct Node
{
  int a,b,c,id;
  Node() = default;
  Node(int _a,int _b,int _c = 0,int _id = 0):a(_a),b(_b),c(_c),id(_id) {}
  inline void read(int i)
  {
    id = i; scanf("%d %d",&a,&b); c = lca(a,b);
    if (c == a||c == b) { if (a != c) swap(a,b); a = L[c]+1; b = L[b]; }
    else { if (L[a] > L[b]) swap(a,b); a = R[a]; b = L[b]; }
  }
  friend inline bool operator <(const Node &x,const Node &y) { return x.b < y.b; }
}query[maxn];

inline bool cmp(const Node &x,const Node &y) { return x.a < y.a; }
inline void work()
{
  int l = 1,r = 0;
  for (int i = 1;i <= Q;++i) {
    while (r < query[i].b) {
      show[dfn[++r]]++;
      if (show[dfn[r]] == 2) dec(dfn[r]); else inc(dfn[r]); }
    while (l > query[i].a) {
      show[dfn[--l]]++;
      if (show[dfn[l]] == 2) dec(dfn[l]); else inc(dfn[l]); }
    while (r > query[i].b) {
      if (show[dfn[r]] == 1) dec(dfn[r]); else inc(dfn[r]);
      show[dfn[r--]]--; }
    while (l < query[i].a) {
      if (show[dfn[l]] == 1) dec(dfn[l]); else inc(dfn[l]);
      show[dfn[l++]]--; }
    ans[query[i].id] = res; }
}

int main() {
  dfs(1);
  for (int i = 1;i <= Q;++i) query[i].read(i);
  sort(query+1,query+Q+1,cmp);
  for (int i = 1,j;i <= Q;i = j) {
    for (j = i;j <= Q&&query[j].a-query[i].a <= Size;++j);
    sort(query+i+1,query+j);
```

```
  }
  work();
  for (int i = 1;i <= Q;++i) printf("%d\n",ans[i]);
}
```

Numerical Integration

```
//self-adapt simpson
inline long double simpson(long double l,long double r,long double mid,long double Cl,long double Cr,long double Cm)
{
    long double tCl = calc((l+mid)/2),tCr = calc((mid+r)/2);
  long double ans=(r-l)*(Cl+Cr+4*Cm)/6,lans=(mid-l)*(Cl+Cm+4*tCl)/6,rans=(r-mid)*(Cr+Cm+4*tCr)/6;
    if (r-l <= 1e-3&&fabs(lans+rans-ans)<eps) return ans;
  // if (dep > lim&&fabs(lans+rans-ans)<eps) return ans;
    else return simpson(l,mid,(l+mid)/2,Cl,Cm,tCl)+simpson(mid,r,(mid+r)/2,Cm,Cr,tCr);
}
```

Planar Graph

```
// 包括平面图转对偶图
inline int dcmp(double a)
{
  if (fabs(a) <= eps) return 0;
  else if (a > 0) return 1;
  else return -1;
}
struct Point
{
  double x,y;
  inline Point(double _x = 0,double _y = 0):x(_x),y(_y) {}
  inline void read() { x = gi(),y = gi(); }
  friend inline Point operator-(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
  inline double angle() { return atan2(y,x); }
}pp[maxn];
struct Segment
{
  int from,to,h,id,sur; // from 号点到 to 号点，h 为边权,suf 为这条有向边维出来的平面编号。
  inline Segment(int _from = 0,int _to = 0,int _h = 0,int _id = 0,int _sur = 0):from(_from),to(_to),h(_h),id(_id),sur(_sur) {}
  friend inline bool operator<(const Segment &a,const Segment &b) { return (pp[a.to]-pp[a.from]).angle() < (pp[b.to]-pp[b.from]).angle(); }
}edge[maxm*2];
vector <int> G[maxn];

inline void nadd(int u,int v,int h) { ++ncnt; G[u].push_back(ncnt); edge[ncnt] = Segment(u,v,h); }
inline void nins(int u,int v,int h) { nadd(u,v,h); nadd(v,u,h); }

inline bool cmp(int a,int b) { return edge[a] < edge[b]; }

inline void find_surface()
{
  for (int i = 1;i <= N;++i) sort(G[i].begin(),G[i].end(),cmp);
  for (int i = 1;i <= N;++i)
  {
    int nn = G[i].size();
    for (int j = 0;j < nn;++j)
      edge[G[i][j]].id = j;
  }
  for (int i = 2;i <= ncnt;++i)
    if (!edge[i].sur)
    {
      ++tot; int j = i,p,nn; vector <Point> vec;
      while (!edge[j].sur)
      {
        edge[j].sur = tot; vec.push_back(pp[edge[j].from]);
        p = edge[j].to; nn = G[p].size();
        j ^= 1; j = G[p][(edge[j].id+1)%nn];
      }
      double res = 0; nn = vec.size();
      for (j = 0;j < nn;++j)
        res += (vec[j]-vec[0])/(vec[(j+1)%nn]-vec[0]);
      res /= 2; space[tot] = res; // 第 tot 个平面的有向面积，外面的大平面面积为正，其余为负，大平面可能有多个（平面图不连通)
    }
  // 开始建边，以 mst 为例
  // for (int i = 2;i <= cnt;i += 2)
  // {
  //   if (space[edge[i].sur]<0&&space[edge[i^1].sur]<0)
  //     arr[++all] = (ARR) { edge[i].sur,edge[i^1].sur,edge[i].h };
  //   else arr[++all] = (ARR) { edge[i].sur,edge[i^1].sur,inf};
  // }
}

// 点定位
struct Scan
{
  double x,y; int bel,sign;
  inline Scan(double _x = 0,double _y = 0,int _bel = 0,int _sign = 0):x(_x),y(_y),bel(_bel),sign(_sign) {}
  friend inline bool operator < (const Scan &a,const Scan &b)
  {
    if (a.x != b.x) return a.x < b.x;
    else return a.sign > b.sign;
  }
}bac[maxn*4];

struct Splay
{
  int num,root,ch[maxn][2],fa[maxn],key[maxn]; queue <int> team;

  inline int newnode()
  {
    int ret;
    if (team.empty()) ret = ++num;
    else ret = team.front(),team.pop();
    fa[ret] = ch[ret][0] = ch[ret][1] = 0;
    return ret;
  }

  inline void init() { num = 0; root = newnode(); key[root] = cnt; }

  inline void rotate(int x)
  {
    int y = fa[x],z = fa[y],l = ch[y][1] == x,r = l^1;
    if (z != 0) ch[z][ch[z][1] == y] = x;
    fa[x] = z; fa[y] = x; fa[ch[x][r]] = y;
    ch[y][l] = ch[x][r]; ch[x][r] = y;
  }

  inline void splay(int x)
  {
    while (fa[x] != 0)
    {
      int y = fa[x],z = fa[y];
      if (fa[y] != 0)
      {
        if ((ch[y][0] == x)^(ch[z][0] == y)) rotate(x);
        else rotate(y);
      }
      rotate(x);
    }
    root = x;
  }

  inline int lower_bound(const Point &p)
  {
    int now = root,ret = 0;
    while (now)
    {
      int k = key[now];
      if ((p-pp[edge[k].from])/(pp[edge[k].to]-pp[edge[k].from]) >= 0)
        ret = k,now = ch[now][0];
      else now = ch[now][1];
    }
    return ret;
  }

  inline int find(int w)
  {
    int now = root;
    double x = pp[edge[w].to].x,y = pp[edge[w].to].y;
    double ang = (pp[edge[w].to] - pp[edge[w].from]).angle();
    while (now)
    {
      int k = key[now];
      if (k == w) return now;
      NODE p = pp[edge[k].to] - pp[edge[k].from],q = pp[edge[k].from];
      double xx = x - q.x,yy = q.y+xx/p.x*p.y;
      if (equal(yy,y))
      {
        double t = p.angle();
        now = ch[now][ang < t];
      }
      else now = ch[now][y > yy];
    }
  }

  inline void erase(int w)
  {
    int p = find(w);
    while (ch[p][0] || ch[p][1])
    {
      if (ch[p][0])
      {
        rotate(ch[p][0]);
        if (p == root) root = fa[p];
      }
      else
      {
        rotate(ch[p][1]);
        if (p == root) root = fa[p];
      }
```

```cpp
      }
      team.push(p);
      ch[fa[p]][ch[fa[p]][1] == p] = 0;
      fa[p] = 0;
    }

    inline void insert(int w)
    {
      int now = root,pre;
      double x = pp[edge[w].from].x,y = pp[edge[w].from].y;
      double ang = (pp[edge[w].to] - pp[edge[w].from]).angle();
      double xx,yy;
      while (true)
      {
        int k = key[now];
        NODE p = pp[edge[k].to] - pp[edge[k].from],q = pp[edge[k].from];
        xx = x - q.x,yy = q.y+xx/p.x*p.y;
        if (equal(yy,y))
        {
          double t = p.angle();
          pre = now,now = ch[now][ang > t];
          if (!now)
          {
            now = newnode();
            fa[now] = pre; ch[pre][ang > t] = now; key[now] = w;
            break;
          }
        }
        else
        {
          pre = now,now = ch[now][y > yy];
          if (!now)
          {
            now = newnode();
            fa[now] = pre; ch[pre][y>yy] = now; key[now] = w;
            break;
          }
        }
      }
      splay(now);
    }
}S;

inline void locate()
{
  int nn = 0;
  for (int i = 2;i <= cnt;i += 2)
  {
    if (!dcmp(pp[edge[i].from].x-pp[edge[i].to].x)) continue;
    bac[++nn] = Scan(pp[edge[i].from].x,pp[edge[i].from].y,i,2);
    bac[++nn] = Scan(pp[edge[i].to].x,pp[edge[i].to].y,i,3);
  }
  scanf("%d",&T); double x,y;
  // 查询 (x,y) 所在平面
  for (int i = 1;i <= T;++i)
  {
    scanf("%lf %lf",&x,&y);
    bac[++nn] = Scan(x,y,i,0);
    scanf("%lf %lf",&x,&y);
    bac[++nn] = Scan(x,y,i,1);
  }
  sort(bac+1,bac+nn+1);
  pp[++n] = Point(-oo,-oo); pp[++n] = (oo,-oo);
  edge[++cnt] = Edge(n-1,n);
  S.init(); int p;
  for (int i = 1;i <= nn;++i)
  {
    if (bac[i].sign == 2||bac[i].sign == 3)
    {
      if (bac[i].sign == 2) S.insert(bac[i].bel);
      else S.erase(bac[i].bel);
    }
    else
    {
      p = S.lower_bound(Point(bac[i].x,bac[i].y));
      query[bac[i].bel][bac[i].sign] = edge[p].sur;
    }
  }
}
```

Point Biconnected Component

```cpp
// Source: HackerRank - bonnie-and-clyde
const int maxn = 400010;
int N,M,Q,cnt = 1,side[maxn],toit[maxn],nxt[maxn],f[maxn][25],father[maxn],low[maxn];
int tot,dep[maxn],dfn[maxn],nside[maxn],ntoit[maxn],nnxt[maxn]; bool cut[maxn];
stack <int> S; vector <int> bel[maxn],bcc[maxn]; bool vis[maxn];

inline int find(int a) { if (father[a] != a) father[a] = find(father[a]); return father[a]; }

inline void add(int a,int b) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; }
inline void ins(int a,int b) { add(a,b); add(b,a); }
inline void nadd(int a,int b) { nnxt[++cnt] = nside[a]; nside[a] = cnt; ntoit[cnt] = b; }
inline void nins(int a,int b) { nadd(a,b); nadd(b,a); }

inline void tj(int now,int fa)
{
  dfn[now] = low[now] = ++cnt; int child = 0;
  for (int i = side[now];i;i = nxt[i])
  {
    if (toit[i] == fa) continue;
    if (!dfn[toit[i]])
    {
      S.push(i>>1); tj(toit[i],now); ++child;
      low[now] = min(low[now],low[toit[i]]);
      if (low[toit[i]] >= dfn[now])
      {
        cut[now] = true; ++tot;
        while (true)
        {
          int t = S.top(); S.pop();
          bel[toit[t<<1]].push_back(tot);  bel[toit[t<<1|1]].push_back(tot);
          bcc[tot].push_back(toit[t<<1]); bcc[tot].push_back(toit[t<<1|1]);
          if (t == (i>>1)) break;
        }
      }
    }
    else low[now] = min(low[now],dfn[toit[i]]);
  }
  if (!fa&&child == 1) cut[now] = false;
}

inline void build()
{
  vector <int> cuts; cnt = 1;
  for (int i = 1;i <= tot;++i)
  {
    sort(bcc[i].begin(),bcc[i].end());
    bcc[i].erase(unique(bcc[i].begin(),bcc[i].end()),bcc[i].end());
  }
  for (int i = 1;i <= N;++i) if (cut[i]) cuts.push_back(i);
  for (auto x:cuts)
  {
    sort(bel[x].begin(),bel[x].end());
    bel[x].erase(unique(bel[x].begin(),bel[x].end()),bel[x].end());
    ++tot; for (auto y:bel[x]) nins(tot,y);
    bel[x].clear(); bel[x].push_back(tot); bcc[tot].push_back(x);
  }
}

inline bool check(int u,int v,int w)
{
  if (find(u) != find(v)||find(v) != find(w)) return false;
  if (u == w||v == w) return true; if (u == v) return false;
  int uu = bel[u][0],vv = bel[v][0],ww = bel[w][0],su,sv;
  if (uu == ww||vv == ww) return true;
  if (lca(uu,ww) == ww) su = jump(uu,dep[uu]-dep[ww]-1); else su = f[ww][0];
  if (lca(vv,ww) == ww) sv = jump(vv,dep[vv]-dep[ww]-1); else sv = f[ww][0];
  if (su == sv)
  {
    if (!cut[w]) return false;
    else
    {
      if (su == uu||sv == vv) return true; int ssu,ssv;
      if (lca(su,uu) == su) ssu = jump(uu,dep[uu]-dep[su]-1); else ssu = f[su][0];
      if (lca(sv,vv) == sv) ssv = jump(vv,dep[vv]-dep[sv]-1); else ssv = f[sv][0];
      if (ssu == ssv) return false; else return true;
    }
  }
  else return true;
}

int main()
{
  N = gi(); M = gi(); Q = gi();
  for (int i = 1;i <= N;++i) father[i] = i;
  for (int i = 1,a,b;i <= M;++i)
  {
    ins(a = gi(),b = gi());
    a = find(a),b = find(b);
    if (a != b) father[a] = b;
  }
  cnt = 0; for (int i = 1;i <= N;++i) if (!dfn[i]) tj(i,0);
  build(); for (int i = 1;i <= N;++i) if (!vis[i]) dfs(i);
  while (Q--)
  {
    int u = gi(),v = gi(),w = gi();
    if (check(u,v,w)) puts("YES"); else puts("NO");
  }
```

```
  return 0;
}
```

Pollard Rho Algorithm

```
const int prime[] = {0,2,3,5,7,11,13,17,19,23,29,31};

inline ll mul(ll a,ll b,ll p) { return (a*b-((ll)((ld)a/p*b+1e-3)*p)+p)%p; }

inline bool check(ll m)
{
  if (m <= 2) return m == 2;
  ll tmp = m-1; int t = 0;
  while (!(tmp&1)) ++t,tmp >>= 1;
  for (int i = 1;i <= 10;++i)
  {
    int a = prime[i];
    if (a == m) return true;
    ll w = qsm(a,tmp,m);
    for (int it = 1;it <= t;++it)
    {
      ll pf = mul(w,w,m);
      if (pf == 1&&(w != 1&&w != m-1)) return false;
      w = pf;
    }
    if (w != 1) return false;
  }
  return true;
}
inline void rho(ll m)
{
  if (check(m)) { fac[++nn] = m; return; }
  while (true)
  {
    ll X = (ll)rand()*rand()%(m-1)+1,Y = X;
    ll c = (ll)rand()*rand()%(m-1)+1; int i,j;
    for (i = j = 2;;++i)
    {
      X = (mul(X,X,m)+c) % m;
      ll d = __gcd(abs(X-Y),m);
      if (1 < d&&d < m) { rho(d),rho(m/d); return; }
      if (X == Y) break; if (i == j) Y = X,j <<= 1;
    }
  }
}
inline void factor(ll m) { nn = 0; if (m > 1) rho(m); sort(fac+1,fac+nn+1); }
```

Polygon Class

```
inline bool PointOnSegment(const Point &t,const Point &a,const Point &b)
{
  if (dcmp((t-a)/(b-a))) return false;
  if (dcmp((t-a)*(t-b)) > 0) return false;
  return true;
}

inline bool in(const Point &a,const Point &b,const Point &c)
{
  double alpha = a.angle(),beta = b.angle(),gamma = c.angle(); // angle 返回 [0,2pi]
  if (alpha <= beta) return dcmp(gamma-alpha) > 0&&dcmp(beta-gamma) > 0;
  else return dcmp(gamma-alpha) > 0||dcmp(beta-gamma) > 0;
}

struct Polygon
{
  int n; Point a[maxn];
  inline Polygon() {}
  inline void read()
  {
    n = gi();
    for (int i = 0;i < n;++i) a[i].read();
    a[n] = a[0];
  }
  // 点是否在多边形内部，内部为 1，外部为 0，边界为 2，不管顺逆时针
  inline int Point_In(const Point &t) const
  {
    int num = 0;
    for (int i = 0;i < n;++i)
    {
      if (PointOnSegment(t,a[i],a[i+1])) return 2;
      int k = dcmp((a[i+1]-a[i])/(t-a[i]));
      int d1 = dcmp(a[i].y-t.y),d2 = dcmp(a[i+1].y-t.y);
      if (k > 0&&d1 <= 0&&d2 > 0) ++num;
      if (k < 0&&d2 <= 0&&d1 > 0) --num;
    }
    return num != 0;
  }
  // 判断多边形的方向，true 为逆时针，false 为顺时针，用叉积判断哪个多
  inline bool CalculateClockDirection()
  {
    int res = 0;
    for (int i = 0;i < n;++i)
    {
      int p = i-1,s = i+1,sgn;
      if (p < 0) p += n; if (s >= n) s -= n;
      sgn = dcmp((a[i]-a[p])/(a[s]-a[i]));
      if (sgn) { if (sgn > 0) ++res; else --res; }
    }
    return res > 0;
  }
  // 判断多边形方向，true 为逆时针，false 为顺时针，用 Green 公式
  inline bool CalculateClockDirection()
  {
    double res = 0;
    for (int i = 0;i < n;++i)
      res -= 0.5*(a[i+1].y+a[i].y)*(a[i+1].x-a[i].x);
    return res > 0;
  }

  // 线段 ab 是否有点严格在多边形内部，先判断线段是否与多边形边界有交，再判断 ab 是否与多边形有交，内部 false，外部 true
  inline bool can(int ia,int ib)
  {
    Point a = P[ia],b = P[ib],v = b-a;
    if (in(P[ia+1]-a,P[ia-1]-a,b-a)||in(P[ib+1]-b,P[ib-1]-b,a-b)) return false;
    for (register int i = 0;i < N;++i)
    {
      if (dcmp(v/(P[i]-a))*dcmp(v/(P[i+1]-a)) < 0&&dcmp(vec[i]/(a-P[i]))*dcmp(vec[i]/(b-P[i])) < 0)
        return false;
      if (PointOnSegment(a,P[i],P[i+1])||PointOnSegment(b,P[i],P[i+1])) return false;
      if (PointOnSegment(P[i],a,b)||PointOnSegment(P[i+1],a,b)) return false;
    }
    return true;
  }
}poly;
```

Simplex

```
// 有 n 个实数变量 x_1,x_2,...,x_n 和 m 条约束，其中第 i 条约束形如 \sum_{i=1}^{n} a_{i,j}x_j \le b_i。
// 此外这 n 个变量需要满足非负性限制，x_j \ge 0。
// 在满足上述所有条件的情况下，你需要指定每个变量 x_j 的取值，使得目标函数 F = \sum_{j=1}^{n} c_j x_j 的值最大。
// 第一行三个正整数 n,m,t。其中 t \in {0,1}。
// 第二行有 n 个整数 c_1,c_2,...,c_n，整数间均用一个空格分隔。
// 接下来 m 行，每行代表一条约束，其中第 i 行有 n+1 个整数 a_{i1},a_{i2},...,a_{in},b_i，整数间均用一个空格分隔。
// 如果不存在满足所有约束的解，仅输出一行``Infeasible''。
// 如果对于任意的 M，都存在一组解使得目标函数的值大于 M，仅输出一行``Unbounded''。
// 否则，第一行输出一个实数，表示目标函数的最大值 F。
// 如果 t = 1，那么你还需要输出第二行，用空格隔开的 n 个非负实数，表示此时 x_1,x_2,...,x_n 的取值，如有多组方案请任意输出其中一个。
// uoj 179
const int maxn = 30;

int N,M,op,tot,q[maxn],idx[maxn],idy[maxn]; double a[maxn][maxn],A[maxn];

inline void pivot(int x,int y)
{
  swap(idy[x],idx[y]);
  double tmp = a[x][y]; a[x][y] = 1/a[x][y];
  for (int i = 0;i <= N;++i) if (y != i) a[x][i] /= tmp;
  tot = 0; for (int i = 0;i <= N;++i) if (i != y&&(a[x][i] > eps||a[x][i] < -eps)) q[++tot] = i;
  for (int i = 0;i <= M;++i)
  {
    if ((x == i)||(a[i][y] < eps&&a[i][y] > -eps)) continue;
    for (int j = 1;j <= tot;++j) a[i][q[j]] -= a[x][q[j]]*a[i][y];
    a[i][y] = -a[i][y]/tmp;
  }
}

int main()
{
  scanf("%d %d %d",&N,&M,&op); srand(233);
  for (int i = 1;i <= N;++i) scanf("%lf",a[0]+i);
  for (int i = 1;i <= M;++i)
  {
    for (int j = 1;j <= N;++j) scanf("%lf",a[i]+j);
    scanf("%lf",a[i]);
  }
  for (int i = 1;i <= N;++i) idx[i] = i;
  for (int i = 1;i <= M;++i) idy[i] = i+N;
  while (true)
  {
    int x = 0,y = 0;
    for (int i = 1;i <= M;++i) if (a[i][0] < -eps&&((!x)||(rand()&1))) x = i; if (!x) break;
    for (int i = 1;i <= N;++i) if (a[x][i] < -eps&&((!y)||(rand()&1))) y = i; if (!y) return puts("Infeasible"),0;
    pivot(x,y);
  }
  while (true)
  {
    int x = 0,y = 0; double mn = 1e15;
    for (int i = 1;i <= N;++i) if (a[0][i] > eps) { y = i; break; } if (!y) break;
```

```
    for (int i = 1;i <= M;++i) if (a[i][y] > eps && a[i][0]/a[i][y] < mn) mn = a[i][0]/a[i][y],x = i; if (!x) return puts("Unbounded"),0;
    pivot(x,y);
  }
  printf("%.8lf\n",-a[0][0]); if (!op) return 0;
  for (int i = 1;i <= M;++i) if (idy[i] <= N) A[idy[i]] = a[i][0];
  for (int i = 1;i <= N;++i) printf("%.8lf ",A[i]);
  return 0;
}
```

Steiner Tree

```
/*
 *  Steiner Tree: 求，使得指定 K 个点连通的生成树的最小总权值
 *  st[i] 表示顶点 i 的标记值，如果 i 是指定集合内第 m(0<=m<K) 个点，则 st[i]=1<<m
 *  endSt=1<<K
 *  dptree[i][state] 表示以 i 为根，连通状态为 state 的生成树值
 */
inline void update(int &x,int y) { if (x == -1) x = y; else if (x > y) x = y; }
inline void spfa(int state)
{
  while (!team.empty())
  {
    int now = team.front(); team.pop();
    for (int i = side[now];i;i = nxt[i])
    {
      int v = toit[i];
      if (f[v][st[v]|state] == -1||f[v][st[v]|state] > f[now][state]+len[i])
      {
        f[v][st[v]|state] = f[now][state]+len[i];
        if ((st[v]|state) != state||vis[v][state]) continue;
        vis[v][state] = true; team.push(v);
      }
    }
    vis[now][state] = false;
  }
}
inline int work()
{
  endSt = 1<<(K<<1);
  memset(f,-1,sizeof(f)); memset(st,0,sizeof(st)); memset(dp,-1,sizeof(dp));
  memset(vis,false,sizeof(vis)); memset(side,0,sizeof(side));
  for (int i = 1;i <= K;++i) st[i] = 1<<(i-1);
  for (int i = 1;i <= K;++i) st[N-K+i] = 1<<(i+K-1);
  for (int i = 1;i <= N;++i) f[i][st[i]] = 0;
  for (int j = 1;j < endSt;++j)
  {
    for (int i = 1;i <= N;++i)
    {
      if (!st[i]||(st[i]&j))
        for (int sub = (j-1)&j;sub;sub = (sub-1)&j)
        {
          int x = sub|st[i],y = (j-sub)|st[i];
          if (f[i][x] != -1&&f[i][y] != -1)
            update(f[i][j],f[i][x]+f[i][y]);
        }
      if (f[i][j] != -1) team.push(i),vis[i][j] = true;
    }
    spfa(j);
  }
}
```

Strongly Connected Component

```
int dfn[maxn],low[maxn],timestamp;
stack <int> stk; vector <int> scc[maxn];
void tarjan(int now)
{
  dfn[now] = low[now] = ++timestamp;
  stk.push(now);
  for (int i = side[now];i;i = nxt[i])
  {
    if (!dfn[toit[i]])
      tarjan(toit[i]),low[now] = min(low[now],low[toit[i]]);
    else if (!bel[toit[i]]) low[now] = min(low[now],dfn[toit[i]]);
  }
  if (dfn[now] == low[now])
  {
    ++tot;
    while (stk.top() != now)
    {
      scc[tot].push_back(stk.top());
      bel[stk.top()] = tot; stk.pop();
    }
    scc[tot].push_back(stk.top());
    bel[stk.top()] = tot; stk.pop();
  }
}
```

Suffix Array

```
// 记得最后填一个字符集中没有的字符
inline void build(char *buf,int *Sa,int *Rank,int *Height,int n,int now,int m)
{
    int i,j,k,*x = t1,*y = t2;
    memset(c,0,4*m);
    for (i = 0;i < n;++i) c[x[i] = buf[i]-'A']++;
    for (i = 1;i < m;++i) c[i] += c[i-1];
    for (i = n-1;i >= 0;--i) Sa[--c[x[i]]] = i;
    for (k = 1;k < n;k <<= 1)
    {
        int p = 0;
        for (i = n-k;i < n;++i) y[p++] = i;
        for (i = 0;i < n;++i) if (Sa[i] >= k) y[p++] = Sa[i] - k;
        memset(c,0,4*m);
        for (i = 0;i < n;++i) c[x[y[i]]]++;
        for (i = 1;i < m;++i) c[i] += c[i-1];
        for (i = n-1;i >= 0;--i) Sa[--c[x[y[i]]]] = y[i];
        swap(x,y); p = 1; x[Sa[0]] = 0;
        for (i = 1;i < n;++i)
            x[Sa[i]] = y[Sa[i-1]] == y[Sa[i]]&&y[Sa[i-1]+k] == y[Sa[i]+k]?p-1:p++;
        if (p >= n) break; m = p;
    }
    for (i = 0;i < n;++i) Rank[Sa[i]] = i;
    for (i = k = 0;i < n;++i)
    {
        if (k) --k; if (!Rank[i]) continue;
        j = Sa[Rank[i]-1];
        while (i+k<n&&j+k<n&&buf[i+k]==buf[j+k]) ++k;
        Height[Rank[i]] = k;
    }
}
```

Zhu-Liu Algorithm

```
struct Directed_MT
{
  struct Edge
  {
    int u,v,w;
    inline Edge() = default;
    inline Edge(int _u,int _v,int _w):u(_u),v(_v),w(_w) {}
  };
  int n,m,vis[maxn],pre[maxn],id[maxn],in[maxn]; Edge edges[maxm];

  inline void init(int _n) { n = _n; m = 0; }
  inline void AddEdge(int u,int v,int w) { edges[m++] = Edge(u,v,w); }
  inline int work(int root)
  {
    int ret = 0;
    while (true)
    {
      // 初始化
      for (int i = 0;i < n;++i) in[i] = inf+1;
      for (int i = 0;i < m;++i)
      {
        int u = edges[i].u,v = edges[i].v;
        // 找寻最小入边，删除自环
        if (edges[i].w < in[v]&&u != v)
          in[v] = edges[i].w,pre[v] = u;
      }
      // 如果没有最小入边，表示该点不连通，则最小树形图形成失败
      for (int i = 0;i < n;++i)
      {
        if (i == root) continue;
        if (in[i] == inf+1) return inf;
      }
      int cnt = 0; // 记录缩点
      memset(id,-1,sizeof id); memset(vis,-1,sizeof vis);
      in[root] = 0;
      for (int i = 0;i < n;++i)
      {
        ret += in[i]; int v = i;
        // 找寻自环
        while (vis[v] != i&&id[v] == -1&&v != root)
          vis[v] = i,v = pre[v];
        if (v != root&&id[v] == -1)
        {
          // 这里不能从 i 开始找，因为 i 有可能不在自环内
          for (int u = pre[v];u != v;u = pre[u]) id[u] = cnt;
          id[v] = cnt++;
        }
      }
      // 如果没有自环了，表示最小树形图成功了
      if (!cnt) break;
      // 找到那些不是自环的，重新给那些点进行标记
      for (int i = 0;i < n;++i)
        if (id[i] == -1) id[i] = cnt++;
      for (int i = 0;i < m;++i)
      {
```

```
        int u = edges[i].u,v = edges[i].v;
        edges[i].v = id[v]; edges[i].u = id[u];
        if (id[u] != id[v]) edges[i].w -= in[v];
      }
      // 缩点完后，点的数量就变了
      n = cnt; root = id[root];
    }
    return ret;
  }
}MT;
```

Code

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

public class Main
{
  public static String reverse(String str) { return new StringBuffer(str).reverse().toString(); }

  public static void main(String args[])
  {
    Scanner cin = new Scanner(System.in);
    int T = cin.nextInt(); BigInteger zero = BigInteger.valueOf(0);
    while (T-- > 0)
    {
      int base1 = cin.nextInt(),base2 = cin.nextInt();
      String S = cin.next(); int len = S.length();
      System.out.println(base1+" "+S);
      BigInteger res = BigInteger.valueOf(0),b1 = BigInteger.valueOf(base1),b2 = BigInteger.valueOf(base2);
      for (int i = 0;i < len;++i)
      {
        res = res.multiply(b1);
        int rep = 0;
        if (S.charAt(i) >= '0'&&S.charAt(i) <= '9') rep = S.charAt(i)-'0';
        else if (S.charAt(i) >= 'A'&&S.charAt(i) <= 'Z') rep = 10+S.charAt(i)-'A';
        else rep = 36+S.charAt(i)-'a';
        res = res.add(BigInteger.valueOf(rep));
      }
      String ret = new String();
      // System.out.println(res);
      if (res.compareTo(zero) == 0) ret += '0';
      else
        while (res.compareTo(zero) > 0)
        {
          long val = res.remainder(b2).longValue();
          // System.out.println(val);
          if (val < 10) ret += (char)(val+'0');
          else if (val < 36) ret += (char)(val+'A'-10);
          else ret += (char)(val+'a'-36);
          res = res.divide(b2);
        }
      System.out.println(base2+" "+reverse(ret)+"\n");
    }
  }
  static void otherfuctions()
  {
    if (((BigInteger.valueOf(mid)).pow(mid)).compareTo(N) <= 0) l = mid+1;
    digit[i] = N.mod(_d).intValue();
    Arrays.fill(cho,false);
    for (int i = 1;i <= N;++i)
    {
      String S = cin.next();
      String[] str = S.split(":");
      BigDecimal a = new BigDecimal(str[0]),b = new BigDecimal(str[1]);
      ratio[i] = a.divide(a.add(b),30,BigDecimal.ROUND_HALF_EVEN);
    }
    Arrays.sort(ratio,1,N+1);
    d = temp.gcd(a[i]);
  }
}
```

Common Formulas 2D

```
//计算几何常用公式
struct Point
{
    double x,y;
  inline Point() = default;
  inline Point(double _x,double _y):x(_x),y(_y) {}
  inline Point reflect(const Point &p) const
  {
    Point v = *this-p; double len = v.norm();
    v = v/len; return p+v*(1/len);
  }
};
struct Line
{
    Point p,v; double slop;
  inline Line() = default;
  inline Line(const Point &_p,const Point &_v):p(_p),v(_v) {}
    inline void update() { slop = v.alpha(); }
    inline double dis(const Point &a) { fabs((a-p)/v)/(v.len()); } //点到直线距离
};

inline Point CrossPoint(const Line &a,const Line &b)  //直线交点，记得判断平行
{
    Point u = a.p - b.p;
    double t = (b.v/u)/(a.v/b.v);
    return a.p+a.v*t;
}

inline Point rotate(const Point &p,double cost,double sint)
{
  double x = p.x,y = p.y;
  return Point(x*cost-y*sint,x*sint+y*cost);
}

inline Point reflect(const Point &a,const Line &l) //点关于直线对称
{
  Point p = l.p,v = l.v; v = v.unit();
  return (2*v*(a-p))*v-(a-p)+p;
}

inline void TangentPoint(const Point &c1,double r2,const Point &c1,double r2) //两圆严格相离，求的是外切
{
  Point v = c1-c2; double len = v.norm(); v = v/len;
  double cost = (r2-r1)/len,sint = mysqrt(1-cost*cost);
  // 两对切点 {c1+r1*rotate(v,cost,sint),c2+r2*rotate(v,cost,sint)},{c1+r1*rotate(v,cost,-sint),c2+r2*rotate(v,cost,-sint)}
}
```

Common Formulas 3D

```
// --------------------Point--------------------
struct Point
{
  double x,y,z;
  friend inline Point operator /(const Point &a,const Point &b) { return Point(a.y*b.z-a.z*b.y,a.z*b.x-a.x*b.z,a.x*b.y-a.y*b.x); }
};
inline double mix(const Point &a,const Point &b,const Point &c) { return a*(b/c); } // The return-value divide 6 is the volume

// a 向量绕 b 向量逆时针选择 angle 弧度 (从 b 方向看)
inline Point rotate(const Point &a,const Point &b,double angle)
{
  Point e1,e2,e3 = b.unit();
  double len = a*e3;
  Point p = e3*len;
  e1 = a-p; if (dcmp(e1.norm())) e1 = e1.unit();
  e2 = e1/e3;
  double x1 = a*e1,y1 = a*e2;
  double x = x1*cos(angle)-y1*sin(angle);
  double y = x1*sin(angle)-y1*cos(angle);
  return e1*x+e2*y+p;
}
// --------------------Line--------------------
struct Line
{
  Point a,b; //直线上两点
  inline Line() = default;
  inline Line(const Point &_a,const Point &_b):a(_a),b(_b) {}
};

inline bool same_side(const Point &p1,const Point &p2,const Line &l) { return dcmp(((l.a-l.b)/(p1-l.b))*((l.a-l.b)/(p2-l.b))) > 0; }// 两点直线同侧
inline bool opposite_side(const Point &p1,const Point &p2,const Line &l) { return dcmp(((l.a-l.b)/(p1-l.b))*((l.a-l.b)/(p2-l.b))) < 0; }// 两点直线异侧

// 判断两条线段是否有交
inline bool intersect_in(const Line &u,const Line &v)
{
  if (!dots_onplane(u.a,u.b,v.a,v.b)) return false;
  if (!dots_inline(u.a,u.b,v.a)&&!dots_inline(u.a,u.b,v.b))
    return !same_side(u.a,u.b,v)&&!same_side(v.a,v.b,u);
  else return dot_online_in(u.a,v)||dot_online_in(u.b,v)||dot_online_in(v.a,u)||dot_online_in(v.b,u);
}
inline bool intersect_ex(const Line &u,const Line &v) { return dots_onplane(u.a,u.b,v.a,v.b)&&opposite_side(u.a,u.b,v)&&opposite_side(v.a,v.b,u); }
// 求两直线交点 (须保证共面不平行)
inline Point intersection(const Line &u,const Line &v)
{
  double t = ((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
    /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
  return u.a+(u.b-u.a)*t;
}
inline double ptoline(const Point &p,const Line &l) { ((p-l.a)/(l.b-l.a)).norm()/(l.a-l.b).norm(); } // 点到直线距离
// 直线到直线距离，平行时变成点到直线距离
inline double linetoline(const Line &u,const Line &v)
{
```

```
  Point n = (u.a-u.b)/(v.a-v.b);
  return fabs((u.a-v.a)*n)/n.norm();
}
// 两直线夹角 cos
inline double angle_cos(const Line &u,const Line &v) { return ((u.a-u.b)*(v.a-v.b))/(u.a-u.b).norm()/(v.a-v.b).norm(); }

// -------------------Plane--------------------
struct Plane
{
  Point a,b,c;
  inline Plane() = default;
  inline Plane(const Point &_a,const Point &_b,const Point &_c):a(_a),b(_b),c(_c) {}
  inline Point pvec() const { return (a-b)/(b-c); } // normal vector
};
// 四点共面
inline bool dots_onplane(const Point &a,const Point &b,const Point &c,const Point &d) { return dcmp(Plane(a,b,c).pvec()*(d-a)) == 0; }
// 点在三角形内
inline bool dot_inplane_in(const Point &p,const Plane &s) { return dcmp(((s.a-s.b)/(s.a-s.c)).norm()-((p-s.a)/(p-s.b)).norm()-((p-s.b)/(p-s.c)).norm()-((p-s.c)/(p-s.a)).norm()) }
inline bool dot_inplane_ex(const Point &p,const Plane &s)
{
  if (!dot_inplane_in(p,s.a,s.b,s.c)) return false;
  int s1 = dcmp((p-s.a)/(p-s.b)),s2 = dcmp((p-s.b)/(p-s.c)),s3 = dcmp((p-s.c)/(p-s.a));
  return ((s1 > 0&&s2 > 0&&s3 > 0)||(s1 < 0&&s2 < 0&&s3 < 0));
}
inline bool same_side(const Point &p1,const Point &p2,const Plane &s) // 同侧
{
  Point v = s.pvec();
  return dcmp((v/(p1-s.a))*(v/(p2-s.a))) > 0;
}
inline bool opposite_side(const Point &p1,const Point &p2,const Plane &s) // 异侧
{
  Point v = s.pvec();
  return dcmp((v/(p1-s.a))*(v/(p2-s.a))) < 0;
}
// 直线与平面是否有交
inline bool intersect_in(const Line &l,const Plane &s)
{
  return !same_side(l.a,l.b,s)&&
    !same_side(s.a,s.b,Plane(l.a,l.b,s.c))&&
    !same_side(s.b,s.c,Plane(l.a,l.b,s.a))&&
    !same_side(s.c,s.a,Plane(l.a,l.b,s.b));
}
inline bool intersect_ex(const Line &l,const Plane &s)
{
  return opposite_side(l.a,l.b,s)&&
    opposite_side(s.a,s.b,Plane(l.a,l.b,s.c))&&
    opposite_side(s.b,s.c,Plane(l.a,l.b,s.a))&&
    opposite_side(s.c,s.a,Plane(l.a,l.b,s.b));
}
// 线面交点
inline Point intersection(const Line &l,const Plane &s)
{
  Point n = s.pvec();
  double t = ((n*(s.a-l.a)))/(n*(l.b-l.a));
  return l.a+(l.b-l.a)*t;
}
// 求两平面交线
inline Line intersection(const Plane &pl1,const Plane &pl2)
{
  Line ret;
  ret.a = parallel(Line(pl2.a,pl2.b),pl1)?intersection(Line(pl2.b,pl2.c),pl1):intersection(Line(pl2.a,pl2.b),pl1);
  ret.b = ret.a+(pl1.pvec()/pl2.pvec());
  return ret;
}
// 点到直线距离
inline double ptoplane(const Point &p,const Plane &s) { double n = s.pvec(); return fabs(n*(p-s.a))/n.norm(); }
inline double angle_cos(const Plane &u,const Plane &v)
{
  Point n1 = u.pvec(),n2 = v.pvec();
  return (n1*n2)/n1.norm()/n2.norm();
}
inline double angle_sin(const Line &l,const Plane &s)
{
  Point n = s.pvec();
  return ((l.a-l.b)*n)/(l.a-l.b).norm()/n.norm();
}
```

Convex Hull 2D

```
struct Point
{
  double x,y;
    friend inline bool operator <(const Point &a,const Point &b)
  {
    if (a.x != b.x) return a.x < b.x;
    else return a.y < b.y;
  }
}P[maxn],convex[maxn];

inline void ConvexHull()
{
    sort(P+1,P+N+1);  //x 第一关键字, y 第二关键字从小到大排序
    for (int i = 1;i <= N;++i)
    {
        while (m > 1&&(convex[m]-convex[m-1])/(P[i]-convex[m-1]) <= 0) --m;
        convex[++m] = P[i];
    }
    int k = m;
    for (int i = N-1;i;--i)
    {
        while (m > k&&(convex[m]-convex[m-1])/(P[i]-convex[m - 1]) <= 0) --m;
        convex[++m] = P[i];
    }
    if (N > 1) m--;
}
```

Convex Hull 3D

```
// 0-base
const double eps = 1e-9;
const int maxn = 1010;
int N,cnt,mark[maxn][maxn];

struct Point
{
  double x,y,z;
  friend inline bool operator <(const Point &a,const Point &b)
  {
    if (dcmp(a.x-b.x)) return a.x < b.x;
    else if (dcmp(a.y-b.y)) return a.y < b.y;
    else if (dcmp(a.z-b.z)) return a.z < b.z;
    else return false;
  }
  friend inline bool operator ==(const Point &a,const Point &b) { return !dcmp(a.x-b.x)&&!dcmp(a.y-b.y)&&!dcmp(a.z-b.z); }
}info[maxn];

inline double volume(int a,int b,int c,int d) { return mix(info[b]-info[a],info[c]-info[a],info[d]-info[a]); }

struct Face
{
  int a,b,c;
  inline Face() = default;
  inline Face(int _a,int _b,int _c):a(_a),b(_b),c(_c) {}
};
vector <Face> face;

inline void add(int v)
{
  vector <Face> tmp;
  int a,b,c; cnt++;
  for (int i = 0;i < (int)face.size();++i)
  {
    a = face[i].a; b = face[i].b; c = face[i].c;
    if (dcmp(volume(v,a,b,c)) < 0)
      mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
    else tmp.push_back(face[i]);
  }
  face = tmp;
  for (int i = 0;i < (int)tmp.size();++i)
  {
    a = face[i].a; b = face[i].b; c = face[i].c;
    if (mark[a][b] == cnt) face.push_back(Face(b,a,v));
    if (mark[b][c] == cnt) face.push_back(Face(c,b,v));
    if (mark[c][a] == cnt) face.push_back(Face(a,c,v));
  }
}

inline bool find()
{
  for (int i = 2;i < N;++i)
  {
    Point ndir = (info[0]-info[i])/(info[1]-info[i]);
    if (!dcmp(ndir.norm())) continue;
    swap(info[i],info[2]);
    for (int j = i+1;j < N;++j)
      if (dcmp(volume(0,1,2,j)) != 0)
      {
        swap(info[j],info[3]);
        face.push_back(Face(0,1,2));
        face.push_back(Face(0,2,1));
        return true;
      }
  }
  return false;
}

inline double ConvexHull()
{
```

```
  sort(info,info+N); N = unique(info,info+N)-info; face.clear();
  random_shuffle(info,info+N);
  if (find())
  {
    memset(mark,0,sizeof mark);  cnt = 0;
    for (int i = 3;i < N;++i) add(i);
    double ans = 0;
    for (int i = 0;i < (int)face.size();++i)
    {
      Point p = (info[face[i].a]-info[face[i].b])/(info[face[i].c]-info[face[i].b]);
      ans += p.norm();
    }
    return ans/2;
  }
  return -1;
}
```

Maximal Weighted Matching in General Graphs

```
// 0-base
const double eps = 1e-9;
const int maxn = 1010;
int N,cnt,mark[maxn][maxn];

struct Point
{
  double x,y,z;
  friend inline bool operator <(const Point &a,const Point &b)
  {
    if (dcmp(a.x-b.x)) return a.x < b.x;
    else if (dcmp(a.y-b.y)) return a.y < b.y;
    else if (dcmp(a.z-b.z)) return a.z < b.z;
    else return false;
  }
  friend inline bool operator ==(const Point &a,const Point &b) { return !dcmp(a.x-b.x)&&!dcmp(a.y-b.y)&&!dcmp(a.z-b.z); }
}info[maxn];

inline double volume(int a,int b,int c,int d) { return mix(info[b]-info[a],info[c]-info[a],info[d]-info[a]); }

struct Face
{
  int a,b,c;
  inline Face() = default;
  inline Face(int _a,int _b,int _c):a(_a),b(_b),c(_c) {}
};
vector <Face> face;

inline void add(int v)
{
  vector <Face> tmp;
  int a,b,c; cnt++;
  for (int i = 0;i < (int)face.size();++i)
  {
    a = face[i].a; b = face[i].b; c = face[i].c;
    if (dcmp(volume(v,a,b,c)) < 0)
      mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
    else tmp.push_back(face[i]);
  }
  face = tmp;
  for (int i = 0;i < (int)tmp.size();++i)
  {
    a = face[i].a; b = face[i].b; c = face[i].c;
    if (mark[a][b] == cnt) face.push_back(Face(b,a,v));
    if (mark[b][c] == cnt) face.push_back(Face(c,b,v));
    if (mark[c][a] == cnt) face.push_back(Face(a,c,v));
  }
}

inline bool find()
{
  for (int i = 2;i < N;++i)
  {
    Point ndir = (info[0]-info[i])/(info[1]-info[i]);
    if (!dcmp(ndir.norm())) continue;
    swap(info[i],info[2]);
    for (int j = i+1;j < N;++j)
      if (dcmp(volume(0,1,2,j)) != 0)
      {
        swap(info[j],info[3]);
        face.push_back(Face(0,1,2));
        face.push_back(Face(0,2,1));
        return true;
      }
  }
  return false;
}

inline double ConvexHull()
{
  sort(info,info+N); N = unique(info,info+N)-info; face.clear();
  random_shuffle(info,info+N);
  if (find())
  {
    memset(mark,0,sizeof mark);  cnt = 0;
    for (int i = 3;i < N;++i) add(i);
    double ans = 0;
    for (int i = 0;i < (int)face.size();++i)
    {
      Point p = (info[face[i].a]-info[face[i].b])/(info[face[i].c]-info[face[i].b]);
      ans += p.norm();
    }
    return ans/2;
  }
  return -1;
}
```

Minimal Ball Cover

```
#include <iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
using namespace std;
const double eps=1e-7;
struct point3D
{
    double x,y,z;
} data[35];
int n;
double dis(point3D a,point3D b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*(a.z-b.z));
}
double solve()
{
    double step=100,ans=1e30,mt;
    point3D z;
    z.x=z.y=z.z=0;
    int s=0;
    while(step>eps)
    {
        for(int i=0; i<n; i++)
            if(dis(z,data[s])<dis(z,data[i])) s=i;
        mt=dis(z,data[s]);
        ans=min(ans,mt);
        z.x+=(data[s].x-z.x)/mt*step;
        z.y+=(data[s].y-z.y)/mt*step;
        z.z+=(data[s].z-z.z)/mt*step;
        step*=0.98;
    }
    return ans;
}
int main()
{ // freopen("t.txt","r",stdin);
    double ans;
    while(~scanf("%d",&n),n)
    {
        for(int i=0; i<n; i++)
            scanf("%lf%lf%lf",&data[i].x,&data[i].y,&data[i].z);
        ans=solve();
        printf("%.5f\n",ans);
    }
    return 0;
}p
```

Minimal Circle Cover

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <vector>
using namespace std;
typedef long long ll;
const int N=2005;
const double INF=1e9;
const double eps=1e-8;
const double pi=acos(-1);
inline int read(){
    char c=getchar();int x=0,f=1;
    while(c<'0'||c>'9'){if(c=='-')f=-1; c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0'; c=getchar();}
    return x*f;
}

inline int sgn(double x){
    if(abs(x)<eps) return 0;
```

```
    else return x<0?-1:1;
}

struct Vector{
    double x,y;
    Vector(double a=0,double b=0):x(a),y(b){}
    bool operator <(const Vector &a)const{
        return sgn(x-a.x)<0||(sgn(x-a.x)==0&&sgn(y-a.y)<0);
    }
    void print(){printf("%lf %lf\n",x,y);}
};
typedef Vector Point;
Vector operator +(Vector a,Vector b){return Vector(a.x+b.x,a.y+b.y);}
Vector operator -(Vector a,Vector b){return Vector(a.x-b.x,a.y-b.y);}
Vector operator *(Vector a,double b){return Vector(a.x*b,a.y*b);}
Vector operator /(Vector a,double b){return Vector(a.x/b,a.y/b);}
bool operator ==(Vector a,Vector b){return sgn(a.x-b.x)==0&&sgn(a.y-b.y)==0;}
double Dot(Vector a,Vector b){return a.x*b.x+a.y*b.y;}
double Cross(Vector a,Vector b){return a.x*b.y-a.y*b.x;}
double Len(Vector a){return sqrt(Dot(a,a));}
Vector Normal(Vector a){
    return Vector(-a.y,a.x);//counterClockwise
};
struct Line{
    Point s,t;
    Line(){}
    Line(Point a,Point b):s(a),t(b){}
};
bool isLSI(Line l1,Line l2){
    Vector v=l1.t-l1.s,u=l2.s-l1.s,w=l2.t-l1.s;
    return sgn(Cross(v,u))!=sgn(Cross(v,w));
}
Point LI(Line a,Line b){
    Vector v=a.s-b.s,v1=a.t-a.s,v2=b.t-b.s;
    double t=Cross(v2,v)/Cross(v1,v2);
    return a.s+v1*t;
}
Point Circumcenter(Point a,Point b,Point c){
    Point p=(a+b)/2,q=(a+c)/2;
    Vector v=Normal(b-a),u=Normal(c-a);
    if(sgn(Cross(v,u))==0){
        if(sgn(Len(a-b)+Len(b-c)-Len(a-c))==0) return (a+c)/2;
        if(sgn(Len(a-b)+Len(a-c)-Len(b-c))==0) return (b+c)/2;
        if(sgn(Len(a-c)+Len(b-c)-Len(a-b))==0) return (a+b)/2;
    }
    return LI(Line(p,p+v),Line(q,q+u));
}

double minCircleCover(Point p[],int n,Point &c){
    random_shuffle(p+1,p+1+n);
    c=p[1];
    double r=0;
    for(int i=2;i<=n;i++)
        if(sgn(Len(c-p[i])-r)>0){
            c=p[i],r=0;
            for(int j=1;j<i;j++)
                if(sgn(Len(c-p[j])-r)>0){
                    c=(p[i]+p[j])/2,r=Len(c-p[i]);
                    for(int k=1;k<j;k++)
                        if(sgn(Len(c-p[k])-r)>0){
                            c=Circumcenter(p[i],p[j],p[k]);
                            r=Len(c-p[i]);
                        }
                }
        }
    return r;
}

int n;
Point p[N],c;
int main(int argc, const char * argv[]){
    while(true){
        n=read();if(n==0) break;
        for(int i=1;i<=n;i++) scanf("%lf%lf",&p[i].x,&p[i].y);
        double r=minCircleCover(p,n,c);
        printf("%.2f %.2f %.2f\n",c.x,c.y,r);
    }
}
```

Stoer Wagner Algorithm

```
int G[maxn][maxn],node[maxn],dis[maxn]; bool visit[maxn];

inline int solve(int n)
{
  if (n == 1) return inf;
  int answer = inf;
  for (int i = 0;i < n;++i) node[i] = i;
  while (n > 1)
  {
    int mx = 1;
    for (int i = 0;i < n;++i)
    {
      dis[node[i]] = G[node[0]][node[i]];
      if (dis[node[i]] > dis[node[mx]]) mx = i;
    }
    int prev = 0;
    memset(visit,false,sizeof visit);
    visit[node[0]] = true;
    for (int i = 1;i < n;++i)
    {
      if (i == n-1)
      {
        answer = min(answer,dis[node[mx]]);
        for (int k = 0;k < n;++k)
          G[node[k]][node[prev]] = (G[node[prev]][node[k]] += G[node[k]][node[mx]]);
        node[mx] = node[--n];
      }
      visit[node[mx]] = true; prev = mx; mx = -1;
      for (int j = 1;j < n;++j)
        if (!visit[node[j]])
        {
          dis[node[j]] += G[node[prev]][node[j]];
          if (mx == -1||dis[node[mx]] < dis[node[j]]) mx = j;
        }
    }
  }
  return answer;
}
```

Virtual Tree

```
int N,cnt,timestamp,dfn[maxn],f[maxn][25],side[maxn],H[maxn];
int dep[maxn],toit[maxn],nxt[maxn],last[maxn],cost[maxn],stk[maxn];
ll best[maxn],g[maxn];

inline void add(int a,int b,int c) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; cost[cnt] = c; }
inline void ins(int a,int b,int c) { add(a,b,c); add(b,a,c); }

inline void nadd(int a,int b,int idc)
{
  if (a == b) return;
  if (last[a] != idc) side[a] = 0,last[a] = idc;
  if (last[b] != idc) side[b] = 0,last[b] = idc;
  nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b;
}

inline bool cmp(int a,int b) { return dfn[a] < dfn[b]; }

inline void dfs(int now)
{
  dfn[now] = ++timestamp;
  for (int i = 1;(1<<i) <= dep[now];++i)
    f[now][i] = f[f[now][i-1]][i-1];
  for (int i = side[now];i;i = nxt[i])
    if (toit[i] != f[now][0])
    {
      best[toit[i]] = min(best[now],(ll)cost[i]);
      dep[toit[i]] = dep[now]+1;
      f[toit[i]][0] = now; dfs(toit[i]);
    }
}

inline int jump(int a,int step) { for (int i = 0;step;step >>= 1,++i) if (step&1) a = f[a][i]; return a; }
inline int lca(int a,int b)
{
  if (dep[a] < dep[b]) swap(a,b);
  a = jump(a,dep[a]-dep[b]);
  if (a == b) return a;
  for (int i = 0;i >= 0;)
  {
    if (f[a][i] != f[b][i])
      a = f[a][i],b = f[b][i],++i;
    else --i;
  }
  return f[a][0];
}

inline void work(int idc)
{
  cnt = 0; int K = gi(),tot,top;
  for (int i = 1;i <= K;++i) H[i] = gi();
  sort(H+1,H+K+1,cmp); H[tot = 1] = H[1];
  for (int i = 2;i <= K;++i) if (lca(H[tot],H[i]) != H[tot]) H[++tot] = H[i];
  stk[top = 1] = 1;
  for (int i = 1;i <= tot;++i)
  {
    int ans = lca(H[i],stk[top]);
    while (true)
```

```
    {
      if (dep[ans] >= dep[stk[top-1]]) { nadd(ans,stk[top--],idc); break; }
      nadd(stk[top-1],stk[top],idc); --top;
    }
    if (stk[top] != ans) stk[++top] = ans;
    if (stk[top] != H[i]) stk[++top] = H[i];
  }
  while (--top) nadd(stk[top],stk[top+1],idc);
  // dp(1); printf("%lld\n",g[1]);
}
```

Graph Isomorphism

哈希函数:

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

其中 K, A, B, C, D, P 为自选参数。两个图同构当且仅当 $F_K(i)$ 值一样。

Competition

```
;; Default Font: Courier 10 Pitch Bold    Size: 15
;; Remember to set CUA-mode and save your options.
(global-set-key (kbd "M-n") 'forward-paragraph)
(global-set-key (kbd "M-p") 'backward-paragraph)
(global-set-key (kbd "RET") 'newline-and-indent)
(global-set-key (kbd "<f8>") 'gud-gdb)
(defun compile-cpp ()
  (interactive)
  (compile (format "g++ -o %s %s -g -lm -Wall -std=c++11"  (file-name-sans-extension (buffer-name))(buffer-name))))
(global-set-key (kbd "<f9>") 'compile-cpp)
(global-linum-mode t)
(setq default-tab-width 4)
(setq c-basic-offset 4)
```

孙司宇

FFT

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
const double eps=1e-8;
const double PI=acos(-1.0);
struct Complex
{
    double real,image;
    Complex(double _real,double _image)
    {
        real=_real;
        image=_image;
    }
    Complex(){real=0;image=0;}
};

Complex operator + (const Complex &c1, const Complex &c2)
{
    return Complex(c1.real + c2.real, c1.image + c2.image);
}

Complex operator - (const Complex &c1, const Complex &c2)
{
    return Complex(c1.real - c2.real, c1.image - c2.image);
}

Complex operator * (const Complex &c1, const Complex &c2)
{
    return Complex(c1.real*c2.real - c1.image*c2.image, c1.real*c2.image + c1.image*c2.real);
}

int rev(int id,int len)
{
    int ret=0;
    for(int i=0;(1<<i)<len;i++)
    {
        ret<<=1;
        if(id&(1<<i))
      ret|=1;
    }
    return ret;
}
Complex* IterativeFFT(Complex* a,int len,int DFT)
{
    Complex* A=new Complex[len];
    for(int i=0;i<len;i++)
        A[rev(i,len)]=a[i];
    for(int s=1;(1<<s)<=len;s++)
    {
        int m=(1<<s);
        Complex wm=Complex(cos(DFT*2*PI/m),sin(DFT*2*PI/m));
        for(int k=0;k<len;k+=m)
        {
            Complex w=Complex(1,0);
            for(int j=0;j<(m>>1);j++)
            {
                Complex t=w*A[k+j+(m>>1)];
                Complex u=A[k+j];
                A[k+j]=u+t;
                A[k+j+(m>>1)]=u-t;
                w=w*wm;
            }
        }
    }
    if(DFT==-1)
  for(int i=0;i<len;i++)
  {
    A[i].real/=len;
    A[i].image/=len;
  }
    return A;
}
char s[101010],t[101010];
Complex a[202020],b[202020],c[202020];
int pr[202020];
int main()
{
  int len;
  scanf("%d",&len);
  scanf("%s",s);
  scanf("%s",t);
  for(int i=0;i<len;i++)
    a[i]=Complex(s[len-i-1]-'0',0);
  for(int i=0;i<len;i++)
    b[i]=Complex(t[len-i-1]-'0',0);
  int tmp=1;
  while(tmp<=len)
    tmp*=2;
  len=tmp*2;
  Complex* aa=IterativeFFT(a,len,1);
  Complex* bb=IterativeFFT(b,len,1);
  for(int i=0;i<len;i++)
    c[i]=aa[i]*bb[i];
  Complex* ans=IterativeFFT(c,len,-1);
  for(int i=0;i<len;i++)
    pr[i]=round(ans[i].real);
  for(int i=0;i<=len;i++)
  {
    pr[i+1]+=pr[i]/10;
    pr[i]%=10;
  }
  bool flag=0;
  for(int i=len-1;i>=0;i--)
  {
    if(pr[i]>0)
      flag=1;
    if(flag)
      printf("%d",pr[i]);
  }
  printf("\n");
    return 0;
}
```

NTT

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
const int N=(1<<18)+5, INF=1e9;
const double PI=acos(-1);
long long P=1004535809;
long long Pow(long long a, long long b,long long P)
{
    long long ans=1;
    for(; b; b>>=1, a=a*a%P)
        if(b&1) ans=ans*a%P;
    return ans;
}
struct NumberTheoreticTransform {
    int n, rev[N];
    long long g;
    void ini(int lim) {
```

```
        g=3;
        n=1; int k=0;
        while(n<lim) n<<=1, k++;
        for(int i=0; i<n; i++) rev[i] = (rev[i>>1]>>1) | ((i&1)<<(k-1));
    }
    void dft(long long *a, int flag) {
        for(int i=0; i<n; i++) if(i<rev[i]) swap(a[i], a[rev[i]]);
        for(int l=2; l<=n; l<<=1) {
            int m=l>>1;
            long long wn = Pow(g, flag==1 ? (P-1)/l : P-1-(P-1)/l, P);
            for(long long *p=a; p!=a+n; p+=l) {
                long long w=1;
                for(int k=0; k<m; k++) {
                    long long t = w * p[k+m]%P;
                    p[k+m]=(p[k]-t+P)%P;
                    p[k]=(p[k]+t)%P;
                    w=w*wn%P;
                }
            }
        }
        if(flag==-1) {
            long long inv=Pow(n, P-2, P);
            for(int i=0; i<n; i++) a[i]=a[i]*inv%P;
        }
    }
    void mul(long long *a, long long *b, int m) {
        ini(m);
        dft(a, 1); dft(b, 1);
        for(int i=0; i<n; i++) a[i]=a[i]*b[i];
        dft(a, -1);
    }
}f;

int n1, n2, m, c[N];
long long a[N], b[N];
char s1[N], s2[N];
int main()
{
  int n;
  scanf("%d",&n);
    scanf("%s%s",s1,s2);
    n1=strlen(s1); n2=strlen(s2);
    for(int i=0;i<n1;i++)
    a[i]=s1[n1-i-1]-'0';
    for(int i=0;i<n2;i++)
    b[i]=s2[n2-i-1]-'0';
    m=n1+n2-1;
    f.mul(a,b,m);
    for(int i=0;i<m;i++) c[i]=a[i];
    for(int i=0;i<m;i++) c[i+1]+=c[i]/10, c[i]%=10;
    if(c[m])
    m++;
    for(int i=m-1; i>=0; i--)
    printf("%d",c[i]);
  return 0;
}
```

SAM

```
#include<iostream>
#include<cstring>
using namespace std;
const int MaxPoint=1010101;
struct Suffix_AutoMachine{
  int son[MaxPoint][27],pre[MaxPoint],step[MaxPoint],right[MaxPoint],last,root,num;
  int NewNode(int stp)
  {
    num++;
    memset(son[num],0,sizeof(son[num]));
    pre[num]=0;
    step[num]=stp;
    return num;
  }
  Suffix_AutoMachine()
  {
    num=0;
    root=last=NewNode(0);
  }
  void push_back(int ch)
  {
    int np=NewNode(step[last]+1);
    right[np]=1;
    step[np]=step[last]+1;
    int p=last;
    while(p&&!son[p][ch])
    {
      son[p][ch]=np;
      p=pre[p];
    }
    if(!p)
      pre[np]=root;
    else
    {
      int q=son[p][ch];
      if(step[q]==step[p]+1)
        pre[np]=q;
      else
      {
        int nq=NewNode(step[p]+1);
        memcpy(son[nq],son[q],sizeof(son[q]));
        step[nq]=step[p]+1;
        pre[nq]=pre[q];
        pre[q]=pre[np]=nq;
        while(p&&son[p][ch]==q)
        {
          son[p][ch]=nq;
          p=pre[p];
        }
      }
    }
    last=np;
  }
};
/*

int arr[1010101];
bool Step_Cmp(int x,int y)
{
  return S.step[x]<S.step[y];
}
void Get_Right()
{
  for(int i=1;i<=S.num;i++)
    arr[i]=i;
  sort(arr+1,arr+S.num+1,Step_Cmp);
  for(int i=S.num;i>=2;i--)
    S.right[S.pre[arr[i]]]+=S.right[arr[i]];
}
*/
int main()
{

  return 0;
}
```

manacher

```
#include<iostream>
#include<cstring>
using namespace std;
char Mana[202020];
int cher[202020];
int Manacher(char *S)
{
  int len=strlen(S),id=0,mx=0,ret=0;
  Mana[0]='$';
  Mana[1]='#';
  for(int i=0;i<len;i++)
  {
    Mana[2*i+2]=S[i];
    Mana[2*i+3]='#';
  }
  Mana[2*len+2]=0;
  for(int i=1;i<=2*len+1;i++)
  {
    if(i<mx)
      cher[i]=min(cher[2*id-i],mx-i);
    else
      cher[i]=0;
    while(Mana[i+cher[i]+1]==Mana[i-cher[i]-1])
      cher[i]++;
    if(cher[i]+i>mx)
    {
      mx=cher[i]+i;
       id=i;
    }
    ret=max(ret,cher[i]);
  }
  return ret;
}
char S[101010];
int main()
{
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);
  cin>>S;
  cout<<Manacher(S)<<endl;
  return 0;
}
```

中国剩余定理

```cpp
// 51nod 1079
#include<iostream>
using namespace std;
int gcd(int x,int y)
{
  if(x==0)
    return y;
  if(y==0)
    return x;
  return gcd(y,x%y);
}
long long exgcd(long long a,long long b,long long &x,long long &y)
{
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    long long ans=exgcd(b,a%b,x,y);
    long long temp=x;
    x=y;
    y=temp-a/b*y;
    return ans;
}
void fix(long long &x,long long &y)
{
  x%=y;
  if(x<0)
    x+=y;
}
bool solve(int n, std::pair<long long, long long> input[],std::pair<long long, long long> &output)
{
  output = std::make_pair(1, 1);
  for(int i = 0; i < n; ++i)
  {
    long long number, useless;
    exgcd(output.second, input[i].second, number, useless);
    long long divisor = gcd(output.second, input[i].second);
    if((input[i].first - output.first) % divisor)
    {
      return false;
    }
    number *= (input[i].first - output.first) / divisor;
    fix(number,input[i].second);
    output.first += output.second * number;
    output.second *= input[i].second / divisor;
    fix(output.first, output.second);
  }
  return true;
}
pair<long long,long long> input[101010],output;
int main()
{
  int n;
  cin>>n;
  for(int i=0;i<n;i++)
    cin>>input[i].second>>input[i].first;
  solve(n,input,output);
  cout<<output.first<<endl;
  return 0;
}
```

回文自动机

```cpp
//Tsinsen A1280 最长双回文串
#include<iostream>
#include<cstring>
using namespace std;

const int maxn = 100005;// n(空间复杂度 o(n*ALP)), 实际开 n 即可
const int ALP = 26;

struct PAM{ // 每个节点代表一个回文串
    int next[maxn][ALP]; // next 指针, 参照 Trie 树
    int fail[maxn]; // fail 失配后缀链接
    int cnt[maxn]; // 此回文串出现个数
    int num[maxn];
    int len[maxn]; // 回文串长度
    int s[maxn]; // 存放添加的字符
    int last; //指向上一个字符所在的节点, 方便下一次 add
    int n; // 已添加字符个数
    int p; // 节点个数

    int newnode(int w)
  {// 初始化节点, w= 长度
        for(int i=0;i<ALP;i++)
            next[p][i] = 0;
        cnt[p] = 0;
        num[p] = 0;
        len[p] = w;
        return p++;
    }
    void init()
  {
        p = 0;
        newnode(0);
        newnode(-1);
        last = 0;
        n = 0;
        s[n] = -1; // 开头放一个字符集中没有的字符, 减少特判
        fail[0] = 1;
    }
    int get_fail(int x)
  { // 和 KMP 一样, 失配后找一个尽量最长的
        while(s[n-len[x]-1] != s[n]) x = fail[x];
        return x;
    }
    int add(int c)
  {
        c -= 'a';
        s[++n] = c;
        int cur = get_fail(last);
        if(!next[cur][c])
    {
            int now = newnode(len[cur]+2);
            fail[now] = next[get_fail(fail[cur])][c];
            next[cur][c] = now;
            num[now] = num[fail[now]] + 1;
        }
        last = next[cur][c];
        cnt[last]++;
        return len[last];
    }
    void count()
  {
        // 最后统计一遍每个节点出现个数
        // 父亲累加儿子的 cnt, 类似 SAM 中 parent 树
        // 满足 parent 拓扑关系
        for(int i=p-1;i>=0;i--)
            cnt[fail[i]] += cnt[i];
    }
}pam;
char S[101010];
int l[101010],r[101010];
int main()
{
  cin>>S;
  int len=strlen(S);
  pam.init();
  for(int i=0;i<len;i++)
    l[i]=pam.add(S[i]);
  pam.init();
  for(int i=len-1;i>=0;i--)
    r[i]=pam.add(S[i]);
  pam.init();
  int ans=0;
  for(int i=0;i<len-1;i++)
    ans=max(ans,l[i]+r[i+1]);
  cout<<ans<<endl;
  return 0;
}
```

多项式开方

```cpp
//    ²
//Nlog^2N
#include <cstdio>
#include <algorithm>
#define FOR(i,j,k) for(i=j;i<=k;++i)
#define rep(i,j,k) for(i=j;i<k;++i)
#define gmod(i) (((i)%mod+mod)%mod)
const int N = 262144, mod = 998244353, inv2 = 499122177;
using namespace std;
typedef long long ll;
ll qpow(ll x, int y) {
    ll z = 1;
    for (; y; x = x * x % mod, y /= 2)
        if (y & 1) z = z * x % mod;
    return z;
}
namespace NTT {
    int n, rev[N], inv_n, m = -1;
    void init(int c) {
        int k = -1, i;
        if (m == c) return; else m = c;
        for (n = 1; n <= m; n <<= 1) ++k;
```

```cpp
        inv_n = qpow(n, mod - 2);
        rep(i,0,n) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << k);
    }
    void ntt(int *a, int f) {
        int h, i, j;
        rep(i,0,n) if (i < rev[i]) swap(a[i], a[rev[i]]);
        for (h = 2; h <= n; h *= 2) {
            int wn = qpow(3, (mod - 1) / h);
            for (i = 0; i < n; i += h) {
                int w = 1;
                rep(j,0,h/2) {
                    int u = a[i + j], t = 1ll * a[i + j + h / 2] * w % mod;
                    a[i + j + h / 2] = (u - t + mod) % mod;
                    a[i + j] = (u + t) % mod;
                    w = 1ll * w * wn % mod;
                }
            }
        }
        if (f) {
            rep(i,1,n/2) swap(a[i], a[n - i]);
            rep(i,0,n) a[i] = 1ll * a[i] * inv_n % mod;
        }
    }
}
void inv(int *a, int *b, int n) {
    static int t[N];
    int i;
    if (n == 1) { b[0] = qpow(a[0], mod - 2); return; }
    inv(a, b, n / 2);
    rep(i,0,n) t[i] = a[i]; rep(i,n,2*n) t[i] = 0;
    NTT::init(n);
    NTT::ntt(t, 0); NTT::ntt(b, 0);
    rep(i,0,NTT::n) t[i] = (ll) b[i] * gmod(2ll - (ll) t[i] * b[i] % mod) % mod;
    NTT::ntt(t, 1);
    rep(i,0,n) b[i] = t[i]; rep(i,n,2*n) b[i] = 0;
}
void sqrt(int *a, int *b, int n) {
    static int t[N], b1[N];
    if (n == 1) { b[0] = 1; return; }
    int i;
    sqrt(a, b, n / 2);
    rep(i,0,n) b1[i] = 0;
    inv(b, b1, n);
    rep(i,0,n) t[i] = a[i]; rep(i,n,2*n) t[i] = 0;
    NTT::init(n);
    NTT::ntt(t, 0), NTT::ntt(b, 0), NTT::ntt(b1, 0);
    rep(i,0,NTT::n) t[i] = inv2 * ((b[i] + (ll) b1[i] * t[i] % mod) % mod) % mod;
    NTT::ntt(t, 1);
    rep(i,0,n) b[i] = t[i]; rep(i,n,2*n) b[i] = 0;
}
int main() {
    static int c[N], sc[N], ic[N];
    int i, x, n, m, l;
    scanf("%d%d", &n, &m);
    FOR(i,1,n) scanf("%d", &x), ++c[x];
    c[0] = gmod(1 - c[0]);
    FOR(i,1,m) c[i] = gmod(-4 * c[i]);
    for (l = 1; l <= m; l <<= 1);
    sqrt(c, sc, l);
    (++sc[0]) %= mod;
    inv(sc, ic, l);
    FOR(i,0,m) ic[i] = 2ll * ic[i] % mod;
    FOR(i,1,m) printf("%d\n", ic[i]);
    return 0;
}
```

多项式求逆

```cpp
//³ F bzoj3456
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
#define N 5000003
#define LL long long
#define p 1004535809
using namespace std;
int n,m;
int a[N],b[N],c[N],jc[N],inv_j[N],wn[N];
LL quickpow(LL num,LL x)
{
    LL base=num%p; LL ans=1;
    while (x) {
        if (x&1) ans=ans*base%p;
        x>>=1;
        base=base*base%p;
    }
    return ans;
}
void init()
{
    jc[0]=1; inv_j[0]=quickpow(jc[0],p-2);
    for (int i=1;i<=n;i++)
     jc[i]=(LL)jc[i-1]*i%p,inv_j[i]=quickpow(jc[i],p-2);
    for (int i=1;i<=n*8;i<<=1)
     wn[i]=quickpow(3,(p-1)/(i<<1));
}
void NTT(int n,int *a,int opt)
{
    for (int i=0,j=0;i<n;i++) {
        if (i>j) swap(a[i],a[j]);
        for (int l=n>>1;(j^=l)<l;l>>=1);
    }
    for (int i=1;i<n;i<<=1) {
        LL wn1=wn[i];
        for (int p1=i<<1,j=0;j<n;j+=p1) {
            LL w=1;
            for (int k=0;k<i;k++,w=(LL)w*wn1%p) {
                int x=a[j+k]; int y=(LL)a[j+k+i]*w%p;
                a[j+k]=(x+y)%p; a[j+k+i]=(x-y+p)%p;
            }
        }
    }
    if (opt==-1) reverse(a+1,a+n);
}
void inverse(int n,int *a,int *b,int *c)
{
    if (n==1) b[0]=quickpow(a[0],p-2);
    else {
        inverse((n+1)>>1,a,b,c);
        int k=0;
        for (k=1;k<=(n<<1);k<<=1);
        for (int i=0;i<n;i++) c[i]=a[i];
        for (int i=n;i<k;i++) c[i]=0;
        NTT(k,c,1);
        NTT(k,b,1);
        for (int i=0;i<k;i++) {
            b[i]=(LL)(2-(LL)c[i]*b[i]%p)*b[i]%p;
            if (b[i]<0) b[i]+=p;
        }
        NTT(k,b,-1);
        int inv=quickpow(k,p-2);
        for (int i=0;i<k;i++) b[i]=(LL)b[i]*inv%p;
        for (int i=n;i<k;i++) b[i]=0;
    }
}
int main()
{
    scanf("%d",&n); init();
    int n1=0;
    for (n1=1;n1<=n*2;n1<<=1);
    a[0]=1;
    for (int i=1;i<=n;i++) a[i]=(LL)quickpow(2,(LL)i*(i-1)/2)*inv_j[i]%p;
    inverse(n1,a,b,c);
    memset(c,0,sizeof(c));
    for (int i=1;i<=n;i++) c[i]=(LL)quickpow(2,(LL)i*(i-1)/2)*inv_j[i-1]%p;
    NTT(n1,b,1); NTT(n1,c,1);
    for (int i=0;i<=n1;i++) b[i]=(LL)b[i]*c[i]%p;
    NTT(n1,b,-1);
    LL inv=quickpow(n1,p-2);
    for (int i=0;i<=n1;i++) b[i]=(LL)b[i]*inv%p;
    printf("%d\n",(LL)b[n]*jc[n-1]%p);
    return 0;
}
```

广义 SAM

```cpp
#include<iostream>
#include<cstring>
using namespace std;
const int MaxPoint=1010101;
struct Suffix_AutoMachine{
  int son[MaxPoint][27],pre[MaxPoint],step[MaxPoint],right[MaxPoint],root,num;
  int NewNode(int stp)
  {
    num++;
    memset(son[num],0,sizeof(son[num]));
    pre[num]=0;
    step[num]=stp;
    return num;
  }
  Suffix_AutoMachine()
  {
    num=0;
    root=NewNode(0);
  }
  int push_back(int ch,int p)
```

```
  {
    int np=NewNode(step[p]+1);
    right[np]=1;
    step[np]=step[p]+1;
    while(p&&!son[p][ch])
    {
      son[p][ch]=np;
      p=pre[p];
    }
    if(!p)
      pre[np]=root;
    else
    {
      int q=son[p][ch];
      if(step[q]==step[p]+1)
        pre[np]=q;
      else
      {
        int nq=NewNode(step[p]+1);
        memcpy(son[nq],son[q],sizeof(son[q]));
        step[nq]=step[p]+1;
        pre[nq]=pre[q];
        pre[q]=pre[np]=nq;
        while(p&&son[p][ch]==q)
        {
          son[p][ch]=nq;
          p=pre[p];
        }
      }
    }
    return np;
  }
};
int main()
{

  return 0;
}
```

循环串最小表示

```
int getmin( char s[] )
{
    int i , j , k , m , t ;
    m = strlen( s ) ;
    i = 0 ; j = 1 ; k = 0 ;
    while( i < m && j < m && k < m )
    {
        t = s[( i + k ) % m] - s[( j + k ) % m] ;
        if( !t )
            ++ k ;
        else
        {
            if( t > 0 )
                i += k + 1 ;
            else
                j += k + 1 ;
            if( i == j )
                j ++ ;
            k = 0 ;
        }

    }
    return min(i,j);
}
```

最大团搜索

```
#include<iostream>
using namespace std;
int ans;
int num[1010];
int path[1010];
int a[1010][1010],n;
bool dfs(int *adj,int total,int cnt)
{
    int i,j,k;
    int t[1010];
    if(total==0)
  {
        if(ans<cnt)
    {
            ans=cnt;
      return 1;
        }
        return 0;
    }
    for(i=0;i<total;i++)
  {
        if(cnt+(total-i)<=ans)
      return 0;
        if(cnt+num[adj[i]]<=ans)
      return 0;
        for(k=0,j=i+1;j<total;j++)
    if(a[adj[i]][adj[j]])
            t[k++]=adj[j];
        if(dfs(t,k,cnt+1))
      return 1;
    }
  return 0;
}
int MaxClique()
{
    int i,j,k;
    int adj[1010];
    if(n<=0)
    return 0;
    ans=1;
    for(i=n-1;i>=0;i--)
  {
        for(k=0,j=i+1;j<n;j++)
        if(a[i][j])
      adj[k++]=j;
        dfs(adj,k,1);
        num[i]=ans;
    }
    return ans;
}
int main()
{
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  while(cin>>n)
  {
    if(n==0)
      break;
    for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
      cin>>a[i][j];
    cout<<MaxClique()<<endl;
  }
  return 0;
}
```

求原根

```
//51Nod - 1135
#include <iostream>
#include <string.h>
#include <algorithm>
#include <stdio.h>
#include <math.h>
#include <bitset>

using namespace std;
typedef long long LL;

const int N = 1000010;

bitset<N> prime;
int p[N],pri[N];
int k,cnt;

void isprime()
{
    prime.set();
    for(int i=2; i<N; i++)
    {
        if(prime[i])
        {
            p[k++] = i;
            for(int j=i+i; j<N; j+=i)
                prime[j] = false;
        }
    }
}

void Divide(int n)
{
    cnt = 0;
    int t = (int)sqrt(1.0*n);
    for(int i=0; p[i]<=t; i++)
    {
        if(n%p[i]==0)
        {
            pri[cnt++] = p[i];
            while(n%p[i]==0) n /= p[i];
        }
```

```
    }
    if(n > 1)
        pri[cnt++] = n;
}

LL quick_mod(LL a,LL b,LL m)
{
    LL ans = 1;
    a %= m;
    while(b)
    {
        if(b&1)
        {
            ans = ans * a % m;
            b--;
        }
        b >>= 1;
        a = a * a % m;
    }
    return ans;
}

int main()
{
    int P;
    isprime();
    while(cin>>P)
    {
        Divide(P-1);
        for(int g=2; g<P; g++)
        {
            bool flag = true;
            for(int i=0; i<cnt; i++)
            {
                int t = (P - 1) / pri[i];
                if(quick_mod(g,t,P) == 1)
                {
                    flag = false;
                    break;
                }
            }
            if(flag)
            {
                int root = g;
                cout<<root<<endl;
        break;
            }
        }
    }
    return 0;
}
```

线性递推多项式

```
//³μ¸
void linear_recurrence(long long n, int m, int a[], int c[], int p)
{
  long long v[M] = {1 % p},u[M << 1], msk = !!n;
  for(long long i(n); i > 1; i >>= 1)
  {
    msk <<= 1;
  }
  for(long long x(0); msk; msk >>= 1, x <<= 1)
  {
    fill_n(u, m << 1, 0);
    int b(!!(n & msk));
    x |= b;
    if(x < m)
    {
      u[x] = 1 % p;
    }
    else
    {
      for(int i(0); i < m; i++)
      {
        for(int j(0), t(i + b); j < m; j++, t++)
        {
          u[t] = (u[t] + v[i] * v[j]) % p;
        }
      }
      for(int i((m << 1) - 1); i >= m; i--)
      {
        for(int j(0), t(i - m); j < m; j++, t++)
        {
          u[t] = (u[t] + c[j] * u[i]) % p;
        }
      }
    }
    copy(u, u + m, v);
  }
  for(int i(m); i < 2 * m; i++)
  {
    a[i] = 0;
    for(int j(0); j < m; j++)
    {
      a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
    }
  }
  for(int j(0); j < m; j++)
  {
    b[j] = 0;
    for(int i(0); i < m; i++)
    {
      b[j] = (b[j] + v[i] * a[i + j]) % p;
    }
  }
  for(int j(0); j < m; j++)
  {
    a[j] = b[j];
  }
}
```

经纬度球面距离

```
double sphereDis(double lon1, double lat1, double lon2, double lat2, double R) {
  return R*acos(cos(lat1)*cos(lat2)*cos(lon1-lon2)+sin(lat1)*sin(lat2));
}
```

日期公式

```
int zeller(int y, int m, int d) { // y 年 m 月 d 日是星期几
  if (m <= 2) y--, m += 12; int c = y / 100; y %= 100;
  int w = ((c >> 2) - (c << 1) + y + (y >> 2) + (13 * (m + 1) / 5) + d - 1) % 7;
  if (w < 0) w += 7; return w;
}
int getId(int y, int m, int d) { // y 年 m 月 d 日的日期编号
  if (m < 3) {y--; m += 12;}
  return 365 * y + y / 4 - y / 100 + y / 400 + (153 * m + 2) / 5 + d;
}
```

Manacher

注意事项：1-based 算法，请注意下标。

```
int manacher(char *text, int length, int *palindrome) {
  static char buffer[MAXN];
  for (int i = 1; i <= length; i++) {
    buffer[2 * i - 1] = text[i];
    if (i != 0) buffer[2 * i] = '#';
  }
  palindrome[1] = 1;
  for (int i = 2, j = 0; i <= 2 * length - 1; ++i) {
    if (j + palindrome[j] <= i) palindrome[i] = 0;
    else palindrome[i] = std::min(palindrome[(j << 1) - i], j + palindrome[j] - i);
    while (i - palindrome[i] >= 1 && i + palindrome[i] <= 2 * length - 1 && buffer[i - palindrome[i]] == buffer[i + palindrome[i]]) {
      palindrome[i]++;
    }
    if (i + palindrome[i] > j + palindrome[j]) j = i;
  }
  int answer = 0;
  for (int i = 1; i < 2 * length; i++) {
    if (i & 1) answer = std::max(answer, 2 * (palindrome[i] - 1 >> 1) + 1);
    else answer = std::max(answer, 2 * (palindrome[i] >> 1));
  }
  return answer;
}
```

丁尧尧

kth.shortest.path

```
#include <cstdio>
#include <cstring>
#include <queue>
using namespace std;

const int N = 1010;
const int M = 100010;
const int oo = 0x3f3f3f3f;

struct Elist {
  int _head[N], _dest[M], _dist[M], _last[M], etot;
  inline void adde( int u, int v, int d ) {
    etot++;
    _dest[etot] = v;
    _dist[etot] = d;
    _last[etot] = _head[u];
    _head[u] = etot;
  }
```

```cpp
  inline int head( int u ) { return _head[u]; }
  inline int dest( int t ) { return _dest[t]; }
  inline int dist( int t ) { return _dist[t]; }
  inline int last( int t ) { return _last[t]; }
};
struct Stat {
  int u, d;
  Stat(){}
  Stat( int u, int d ):u(u),d(d){}
};

int n, m, K;
Elist e, re;
int src, dst;
int rdis[N];
bool done[N];

bool operator<( const Stat &r, const Stat &s ) {
  return r.d + rdis[r.u] > s.d + rdis[s.u];
}
void dijkstra() {
  memset( done, false, sizeof(done) );
  memset( rdis, 0x3f, sizeof(rdis) );
  priority_queue<pair<int,int> > Q;
  Q.push( make_pair(0,dst) );
  rdis[dst] = 0;
  while( !Q.empty() ) {
    int u = Q.top().second;
    Q.pop();
    if( done[u] ) continue;
    done[u] = true;
    for( int t = re.head(u); t; t = re.last(t) ) {
      int v = re.dest(t), d = re.dist(t);
      if( done[v] ) continue;
      if( rdis[v] > rdis[u] + d ) {
        rdis[v] = rdis[u] + d;
        Q.push( make_pair( -rdis[v], v ) );
      }
    }
  }
}
int astar() {
  int pcnt = 0;
  priority_queue<Stat> Q;

  if( rdis[src] == oo ) return -1;
  if( src == dst ) K++;
  Q.push( Stat( src, 0 ) );
  while( !Q.empty() ) {
    Stat s = Q.top();
    Q.pop();
    if( s.u == dst ) {
      pcnt++;
      if( pcnt == K )
        return s.d;
    }
    for( int t = e.head(s.u); t; t = e.last(t) ) {
      int v = e.dest(t), d = e.dist(t);
      if( rdis[v] == oo ) continue;
      Q.push( Stat( v, s.d + d ) );
    }
  }
  return -1;
}
int main() {
  scanf( "%d%d", &n, &m );
  for( int i = 1; i <= m; i++ ) {
    int u, v, d;
    scanf( "%d%d%d", &u, &v, &d );
    e.adde( u, v, d );
    re.adde( v, u, d );
  }
  scanf( "%d%d%d", &src, &dst, &K );
  dijkstra();
  printf( "%d\n", astar() );
}
```

弦图

```cpp
/*
一些定义：
弦图是一种特殊图：它的所有极小环都只有 3 个顶点。
单纯点：该顶点与其邻接点在原图中的导出子图是一个完全图。
图 G 的完美消去序列：一个顶点序列 a1a2a3...an，使得对于每个元素 ai，ai 在 ai、ai+1、ai+2...an 的导出子图中是一个单纯点。
弦图有一个性质：任何一个弦图都至少存在一个单纯点（该点和其邻接点组成一个完全图）
弦图另一个性质：一个图是弦图当且仅当其存在完美消去序列。(归纳证明)
最大势算法 (msc)：若原图是弦图，则该算法计算出的序列是完美消去序列。
算法大致思想：从后往前计算序列，每次选择点 v 作为序列中的元素，v 是还未选的点中与已经选了的点连边最多的点。
然后检查该序列是否是完美消去序列。
*/
#include <cstdio>
#include <cstring>
#define N 1010
#define M N*N*2
int n, m;
bool c[N][N];
int qu[N], inq[N], dgr[N];
int stk[N], top;
void msc() {
    dgr[0] = -1;
    for( int i=n; i>=1; i-- ) {
        int s = 0;
        for( int u=1; u<=n; u++ )
            if( !inq[u] && dgr[u]>dgr[s] ) s=u;
        qu[i] = s;
        inq[s] = true;
        for( int u=1; u<=n; u++ )
            if( !inq[u] && c[s][u] ) dgr[u]++;
    }
}
bool check() {
    for( int i=n; i>=1; i-- ) {
        int s=qu[i];
        top = 0;
        for( int j=i+1; j<=n; j++ )
            if( c[s][qu[j]] ) stk[++top] = qu[j];
        if( top==0 ) continue;
        for( int j=2; j<=top; j++ )
            if( !c[stk[1]][stk[j]] ) return false;
    }
    return true;
}
int main() {
    scanf( "%d%d", &n, &m );
    for( int i=1,u,v; i<=m; i++ ) {
        scanf( "%d%d", &u, &v );
        c[u][v] = c[v][u] = 1;
    }
    msc();
    printf( "%s\n", check() ? "Perfect" : "Imperfect" );
}

/*
给定一个弦图，问最少染色数。
对于弦图的一个完美消去序列，从后往前染色，每次染可以染的最小编号的颜色，
由完美消去序列的定义， 序列任一后缀的点的导出子图中，由该后缀第一个元素
及其邻接点导出的子图一定是完全图，所以，序列中 某一元素染的颜色编号是该
完全图的大小。所以最小染色数小于等于最大团的点数，而显然前者又大于等于后者，
故弦图的最小染色数等于最大团的大小。
*/
#include <cstdio>
#include <vector>
#define maxn 10010
using namespace std;
int n, m;
vector<int> g[maxn];
bool done[maxn];
int label[maxn], pos[maxn];
int msc() {
    int rt = 0;
    for( int i=n; i>=1; i-- ) {
        int mu = 0;
        for( int u=1; u<=n; u++ ) {
            if( !done[u] ) {
                if( !mu || label[u]>label[mu] )
                    mu = u;
            }
        }
        done[mu] = true;
        pos[mu] = i;
        int cnt = 0;
        for( int t=0; t<g[mu].size(); t++ ) {
            int v = g[mu][t];
            if( done[v] ) {
                cnt++;
            } else {
                label[v]++;
            }
        }
        rt = max( rt, cnt+1 );
    }
    return rt;
}
int main() {
    scanf( "%d%d", &n, &m );
    for( int i=1,u,v; i<=m; i++ ) {
        scanf( "%d%d", &u, &v );
        g[u].push_back(v);
```

```
        g[v].push_back(u);
    }
    printf( "%d\n", msc() );
}
```

集合幂级数

```
#include <cstdio>
const int N = 10;
int n, U;
int a[1<<N], b[1<<N], c[1<<N];
void trans( int a[], int flag ) {
    for( int b=0; b<n; b++ ) {
        int u = U ^ (1<<b);
        for( int s=u,t=1<<(n-1); t; s=(s-1)&u,t-- ) {
            int l=a[s], r=a[s|(1<<b)];
            /* NOT AND
            if( flag==1 ) { a[s] = l+r; a[s|(1<<b)] = r;
            } else { a[s] = r; a[s|(1<<b)] = l-r; } */
            /* NOT XOR
            if( flag==1 ) { a[s] = l+r; a[s|(1<<b)] = l-r;
            } else { a[s] = (l-r)/2; a[s|(1<<b)] = (l+r)/2; } */
            /* NOT OR
            if( flag==1 ) { a[s] = l; a[s|(1<<b)] = l+r;
            } else { a[s] = r-l; a[s|(1<<b)] = l; } */
            /* OR
            if( flag==1 ) { a[s] = l; a[s|(1<<b)] = l+r;
            } else { a[s] = l; a[s|(1<<b)] = r-l; } */
            /* AND
            if( flag==1 ) { a[s] = l+r; a[s|(1<<b)] = r;
            } else { a[s] = l-r; a[s|(1<<b)] = r; } */
            /* XOR
            if( flag==1 ) { a[s] = l+r; a[s|(1<<b)] = l-r;
            } else { a[s] = (l+r)/2; a[s|(1<<b)] = (l-r)/2; } */
        }
    }
}
int main() {
    scanf( "%d", &n );
    U = (1<<n)-1;
    for( int i=0; i<=U; i++ ) scanf( "%d", a+i );
    for( int i=0; i<=U; i++ ) scanf( "%d", b+i );
    trans(a,1); trans(b,1);
    for( int s=0; s<=U; s++ ) c[s] = a[s]*b[s];
    trans(c,-1);
    for( int s=0; s<=U; s++ ) printf( "%d ", c[s] );
    printf( "\n" );
}
```

其他

Java Hints

```
import java.util.*;
import java.math.*;
import java.io.*;
public class Main{
  static class Task{
    void solve(int testId, InputReader cin, PrintWriter cout) {
      // Write down the code you want
    }
  };
  public static void main(String args[]) {
    InputStream inputStream = System.in;
    OutputStream outputStream = System.out;
    InputReader in = new InputReader(inputStream);
    PrintWriter out = new PrintWriter(outputStream);
    TaskA solver = new TaskA();
    solver.solve(1, in, out);
    out.close();
  }
  static class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;
    public InputReader(InputStream stream) {
      reader = new BufferedReader(new InputStreamReader(stream), 32768);
      tokenizer = null;
    }
    public String next() {
      while (tokenizer == null || !tokenizer.hasMoreTokens()) {
        try {
          tokenizer = new StringTokenizer(reader.readLine());
        } catch (IOException e) {
          throw new RuntimeException(e);
        }
      }
      return tokenizer.nextToken();
    }
    public int nextInt() {
      return Integer.parseInt(next());
    }
  }
};
// Arrays
int a[];
.fill(a[,int fromIndex,int toIndex],val);|.sort(a[, int fromIndex, int toIndex])
// String
String s;
.charAt(int i);|compareTo(String)|compareToIgnoreCase ()|contains(String)|
length ()|substring(int l, int len)
// BigInteger
.abs()|.add()|bitLength()|subtract()|divide()|remainder()|divideAndRemainder()|
modPow(b, c)|pow(int) | multiply () | compareTo () |
gcd() | intValue () | longValue () | isProbablePrime(int c) (1 - 1/2^c) |
nextProbablePrime () | shiftLeft(int) | valueOf ()
// BigDecimal
.ROUND_CEILING | ROUND_DOWN_FLOOR | ROUND_HALF_DOWN | ROUND_HALF_EVEN | ROUND_HALF_UP | ROUND_UP
.divide(BigDecimal b, int scale , int round_mode) | doubleValue () | movePointLeft(int) | pow(int) |
setScale(int scale , int round_mode) | stripTrailingZeros ()
// StringBuilder
StringBuilder sb = new StringBuilder ();
sb.append(elem) | out.println(sb)
// TODO Java STL 的使用方法以及上面这些方法的检验
```

常用结论

上下界网络流

$B(u,v)$ 表示边 (u,v) 流量的下界，$C(u,v)$ 表示边 (u,v) 流量的上界，$F(u,v)$ 表示边 (u,v) 的流量。设 $G(u,v)=F(u,v)-B(u,v)$，显然有：$0\le G(u,v)\le C(u,v)-B(u,v)$

无源汇的上下界可行流

建立超级源点 S^* 和超级汇点 T^*，对于原图每条边 (u,v) 在新网络中连如下三条边：$S^*\to v$，容量为 $B(u,v)$；$u\to T^*$，容量为 $B(u,v)$；$u\to v$，容量为 $C(u,v)-B(u,v)$。最后求新网络的最大流，判断从超级源点 S^* 出发的边是否都满流即可，边 (u,v) 的最终解中的实际流量为 $G(u,v)+B(u,v)$。

有源汇的上下界可行流

从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边。按照**无源汇的上下界可行流**一样做即可，流量即为 $T\to S$ 边上的流量。

有源汇的上下界最大流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 ∞，下届为 x 的边。x 满足二分性质，找到最大的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最大流。
2. 从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边，变成无源汇的网络。按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 和超级汇点 T^*，求一遍 $S^*\to T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S\to T$ 的最大流即可。

有源汇的上下界最小流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 x，下界为 0 的边。x 满足二分性质，找到最小的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最小流。
2. 按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^*\to T^*$ 的最大流，但是注意这一次不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0，所以 S^*，T^* 无影响，再直接求一次 $S^*\to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T\to S$ 边上的流量即为原图的最小流，否则无解。

上下界费用流

来源：BZOJ 3876 设汇 t，源 s，超级源 S，超级汇 T，本质是每条边的下界为 1，上界为 MAX，跑一遍有源汇的上下界最小费用最小流。（因为上界无穷大，所以只要满足所有下界的最小费用最小流）

1. 对每个点 x：从 x 到 t 连一条费用为 0，流量为 MAX 的边，表示可以任意停止当前的剧情（接下来的剧情从更优的路径去走，画个样例就知道了）
2. 对于每一条边权为 z 的边 $x\to y$：
   - 从 S 到 y 连一条流量为 1，费用为 z 的边，代表这条边至少要被走一次。
   - 从 x 到 y 连一条流量为 MAX，费用为 z 的边，代表这条边除了至少走的一次之外还可以随便走。
   - 从 x 到 T 连一条流量为 1，费用为 0 的边。（注意是每一条 $x->y$ 的边都连，或者你可以记下 x 的出边数 K_x，连一次流量为 K_x，费用为 0 的边）。

建完图后从 S 到 T 跑一遍费用流，即可。（当前跑出来的就是满足上下界的最小费用最小流了）

弦图相关

1. 团数 $\le$ 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 $|N(v)| + 1 \le |N(w)|$ 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

Bernoulli 数

1. 初始化: $B_0(n) = 1$
2. 递推公式: $B_m(n) = n^m - \displaystyle\sum_{k=0}^{m-1} \binom{m}{k} \frac{B_k(n)}{m-k+1}$
3. 应用: $\displaystyle\sum_{k=1}^{n} k^m = \frac{1}{m+1} \sum_{k=0}^{m} \binom{m+1}{k} n^{m+1-k}$

常见错误

1. 数组或者变量类型开错, 例如将 double 开成 int;
2. 函数忘记返回返回值;
3. 初始化数组没有初始化完全;
4. 对空间限制判断不足导致 MLE

博弈游戏

巴什博奕

1. 只有一堆 n 个物品，两个人轮流从这堆物品中取物，规定每次至少取一个，最多取 m 个。最后取光者得胜。
2. 显然，如果 $n = m + 1$，那么由于一次最多只能取 m 个，所以，无论先取者拿走多少个，后取者都能够一次拿走剩余的物品，后者取胜。因此我们发现了如何取胜的法则：如果 $n = m + 1\, r + s$，(r 为任意自然数，$s \le m$), 那么先取者要拿走 s 个物品，如果后取者拿走 $k (k \le m)$ 个，那么先取者再拿走 $m + 1 - k$ 个，结果剩下 $(m + 1)(r - 1)$ 个，以后保持这样的取法，那么先取者肯定获胜。总之，要保持给对手留下 $(m + 1)$ 的倍数，就能最后获胜。

威佐夫博弈

1. 有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。
2. 判断一个局势 (a, b) 为奇异局势（必败态）的方法：$a_k = [k(1 + \sqrt{5})/2]$ $b_k = a_k + k$

阶梯博奕

1. 博弈在一列阶梯上进行，每个阶梯上放着自然数个点，两个人进行阶梯博弈，每一步则是将一个阶梯上的若干个点（至少一个）移到前面去，最后没有点可以移动的人输。
2. 解决方法：把所有奇数阶梯看成 N 堆石子，做 NIM。(把石子从奇数堆移动到偶数堆可以理解为拿走石子，就相当于几个奇数堆的石子在做 Nim)

图上删边游戏

链的删边游戏

1. 游戏规则：对于一条链，其中一个端点是根，两人轮流删边，脱离根的部分也算被删去，最后没边可删的人输。
2. 做法：$sg[i] = n - dist(i) - 1$（其中 n 表示总点数，$dist(i)$ 表示离根的距离）

树的删边游戏

1. 游戏规则：对于一棵有根树，两人轮流删边，脱离根的部分也算被删去，没边可删的人输。
2. 做法：叶子结点的 $sg = 0$，其他节点的 sg 等于儿子结点的 $sg + 1$ 的异或和。

局部连通图的删边游戏

1. 游戏规则：在一个局部连通图上，两人轮流删边，脱离根的部分也算被删去，没边可删的人输。局部连通图的构图规则是，在一棵基础树上加边得到，所有形成的环保证不共用边，且只与基础树有一个公共点。
2. 做法：去掉所有的偶环，将所有的奇环变为长度为 1 的链，然后做树的删边游戏。

常用数学公式

求和公式

1. $\sum_{k=1}^{n} (2k-1)^2 = \frac{n(4n^2-1)}{3}$
2. $\sum_{k=1}^{n} k^3 = [\frac{n(n+1)}{2}]^2$
3. $\sum_{k=1}^{n} (2k-1)^3 = n^2(2n^2-1)$
4. $\sum_{k=1}^{n} k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$
5. $\sum_{k=1}^{n} k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$
6. $\sum_{k=1}^{n} k(k+1) = \frac{n(n+1)(n+2)}{3}$
7. $\sum_{k=1}^{n} k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$
8. $\sum_{k=1}^{n} k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$
9. $\displaystyle\frac{1}{(1-x)^{n+1}} = \sum_{i=0}^{n} \binom{i+n}{i} x^i$
10. $\displaystyle\frac{1}{\sqrt{1-4x}} = \sum_{i=0}^{n} \binom{2i}{i} x^i$

斐波那契数列

1. $fib_0 = 0, fib_1 = 1, fib_n = fib_{n-1} + fib_{n-2}$
2. $fib_{n+2} \cdot fib_n - fib_{n+1}^2 = (-1)^{n+1}$
3. $fib_{-n} = (-1)^{n-1} fib_n$
4. $fib_{n+k} = fib_k \cdot fib_{n+1} + fib_{k-1} \cdot fib_n$
5. $gcd(fib_m, fib_n) = fib_{gcd(m,n)}$
6. $fib_m | fib_n^2 \Leftrightarrow nfib_n | m$

错排公式

$$D_n = (n-1)(D_{n-2} - D_{n-1}) == n! \cdot (1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + \frac{(-1)^n}{n!})$$

莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & 若n = 1 \\ (-1)^k & 若n无平方数因子，且n = p_1 p_2 \ldots p_k \\ 0 & 若n有大于1的平方数因数 \end{cases}$$

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & 若n = 1 \\ 0 & 其他情况 \end{cases}$$

$$g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$$

$$g(x) = \sum_{n=1}^{[x]} f(\frac{x}{n}) \Leftrightarrow f(x) = \sum_{n=1}^{[x]} \mu(n) g(\frac{x}{n})$$

Burnside 引理

设 G 是一个有限群，作用在集合 X 上。对每个 g 属于 G，令 X^g 表示 X 中在 g 作用下的不动元素，轨道数（记作 $|X/G|$）为 $|X/G| = \dfrac{1}{|G|} \displaystyle\sum_{g \in G} |X^g|$.

五边形数定理

设 $p(n)$ 是 n 的拆分数，有 $p(n) = \displaystyle\sum_{k \in \mathbb{Z} \setminus \{0\}} (-1)^{k-1} p\left(n - \frac{k(3k-1)}{2}\right)$

树的计数

1. 有根树计数：n+1 个结点的有根树的个数为 $a_{n+1}=\frac{\sum_{j=1}^{n} j\cdot a_j\cdot S_{n,j}}{n}$，其中，$S_{n,j}=\sum_{i=1}^{n/j} a_{n+1-ij}=S_{n-j,j}+a_{n+1-j}$
2. 无根树计数：当 n 为奇数时，n 个结点的无根树的个数为 $a_n-\sum_{i=1}^{n/2} a_i a_{n-i}$，当 n 为偶数时，n 个结点的无根树的个数为 $a_n-\sum_{i=1}^{n/2} a_i a_{n-i}+\frac{1}{2}a_{\frac{n}{2}}(a_{\frac{n}{2}}+1)$
3. n 个结点的完全图的生成树个数为：n^{n-2}
4. 矩阵 - 树定理：图 G 由 n 个结点构成，设 $\boldsymbol{A}[G]$ 为图 G 的邻接矩阵、$\boldsymbol{D}[G]$ 为图 G 的度数矩阵，则图 G 的不同生成树的个数为 $\boldsymbol{C}[G]=\boldsymbol{D}[G]-\boldsymbol{A}[G]$ 的任意一个 $n-1$ 阶主子式的行列式值。

欧拉公式

平面图的顶点个数、边数和面的个数有如下关系：$V-E+F=C+1$

其中，V 是顶点的数目，E 是边的数目，F 是面的数目，C 是组成图形的连通部分的数目。当图是单连通图的时候，公式简化为：$V-E+F=2$

皮克定理

给定顶点坐标均是整点（或正方形格点）的简单多边形，其面积 A 和内部格点数目 i、边上格点数目 b 的关系：$A=i+\frac{b}{2}-1$

牛顿恒等式

设

$$\prod_{i=1}^{n}(x-x_i)=a_n+a_{n-1}x+\cdots+a_1x^{n-1}+a_0x^n$$

$$p_k=\sum_{i=1}^{n}x_i^k$$

则

$$a_0p_k+a_1p_{k-1}+\cdots+a_{k-1}p_1+ka_k=0$$

特别地，对于

$$|\boldsymbol{A}-\lambda\boldsymbol{E}|=(-1)^n(a_n+a_{n-1}\lambda+\cdots+a_1\lambda^{n-1}+a_0\lambda^n)$$

有

$$p_k=\mathrm{Tr}(\boldsymbol{A}^k)$$

平面几何公式

三角形

1. 面积：$S=\frac{a\cdot H_a}{2}=\frac{ab\cdot sinC}{2}=\sqrt{p(p-a)(p-b)(p-c)}\left(\frac{a+b+c}{2}\right)$
2. 中线：$M_a=\frac{\sqrt{2(b^2+c^2)-a^2}}{2}=\frac{\sqrt{b^2+c^2+2bc\cdot cosA}}{2}$
3. 角平分线：$T_a=\frac{\sqrt{bc\cdot[(b+c)^2-a^2]}}{b+c}=\frac{2bc}{b+c}cos\frac{A}{2}$
4. 高线：$H_a=bsinC=csinB=\sqrt{b^2-(\frac{a^2+b^2-c^2}{2a})^2}$
5. 内切圆半径
$$r=\frac{S}{p}=\frac{arcsin\frac{B}{2}\cdot sin\frac{C}{2}}{sin\frac{B+C}{2}}=4R\cdot sin\frac{A}{2}sin\frac{B}{2}sin\frac{C}{2}$$
$$=\sqrt{\frac{(p-a)(p-b)(p-c)}{p}}=p\cdot tan\frac{A}{2}tan\frac{B}{2}tan\frac{C}{2}$$
6. 外接圆半径：$R=\frac{abc}{4S}=\frac{a}{2sinA}=\frac{b}{2sinB}=\frac{c}{2sinC}$

四边形

D_1,D_2 为对角线，M 对角线中点连线，A 为对角线夹角，p 为半周长

1. $a^2+b^2+c^2+d^2=D_1^2+D_2^2+4M^2$
2. $S=\frac{1}{2}D_1D_2sinA$
3. 对于圆内接四边形：$ac+bd=D_1D_2$
4. 对于圆内接四边形：$S=\sqrt{(p-a)(p-b)(p-c)(p-d)}$

正 n 边形

R 为外接圆半径，r 为内切圆半径

1. 中心角：$A=\frac{2\pi}{n}$
2. 内角：$C=\frac{n-2}{n}\pi$
3. 边长：$a=2\sqrt{R^2-r^2}=2R\cdot sin\frac{A}{2}=2r\cdot tan\frac{A}{2}$
4. 面积：$S=\frac{nar}{2}=nr^2\cdot tan\frac{A}{2}=\frac{nR^2}{2}\cdot sinA=\frac{na^2}{4\cdot tan\frac{A}{2}}$

圆

1. 弧长：$l=rA$
2. 弦长：$a=2\sqrt{2hr-h^2}=2r\cdot sin\frac{A}{2}$
3. 弓形高：$h=r-\sqrt{r^2-\frac{a^2}{4}}=r(1-cos\frac{A}{2})=\frac{1}{2}\cdot arctan\frac{A}{4}$
4. 扇形面积：$S_1=\frac{rl}{2}=\frac{r^2A}{2}$
5. 弓形面积：$S_2=\frac{rl-a(r-h)}{2}=\frac{r^2}{2}(A-sinA)$

棱柱

1. 体积（A 为底面积，h 为高）：$V=Ah$
2. 侧面积（l 为棱长，p 为直截面周长）：$S=lp$
3. 全面积：$T=S+2A$

棱锥

1. 体积（A 为底面积，h 为高）：$V=Ah$
2. 正棱锥侧面积（l 为棱长，p 为直截面周长）：$S=lp$
3. 正棱锥全面积：$T=S+2A$

棱台

1. 体积（A_1,A_2 为上下底面积，h 为高）：$V=(A_1+A_2+\sqrt{A_1A_2})\cdot\frac{h}{3}$
2. 正棱台侧面积（p_1,p_2 为上下底面周长，l 为斜高）：$S=\frac{p_1+p_2}{2}l$
3. 正棱台全面积：$T=S+A_1+A_2$

圆柱

1. 侧面积：$S=2\pi rh$
2. 全面积：$T=2\pi r(h+r)$
3. 体积：$V=\pi r^2h$

圆锥

1. 母线：$l=\sqrt{h^2+r^2}$
2. 侧面积：$S=\pi rl$
3. 全面积：$T=\pi r(l+r)$
4. 体积：$V=\frac{\pi}{3}r^2h$

圆台

1. 母线：$l=\sqrt{h^2+(r_1-r_2)^2}$
2. 侧面积：$S=\pi(r_1+r_2)l$
3. 全面积：$T=\pi r_1(l+r_1)+\pi r_2(l+r_2)$
4. 体积：$V=\frac{\pi}{3}(r_1^2+r_2^2+r_1r_2)h$

球

1. 全面积：$T=4\pi r^2$
2. 体积：$V=\frac{4}{3}\pi r^3$

球台

1. 侧面积：$S=2\pi rh$
2. 全面积：$T=\pi(2rh+r_1^2+r_2^2)$
3. 体积：$V=\frac{\pi h[3(r_1^2+r_2^2)+h^2]}{6}$

球扇形

1. 全面积（h 为球冠高，r_0 为球冠底面半径）：$T=\pi r(2h+r_0)$
2. 体积：$V=\frac{2}{3}\pi r^2 h$

立体几何公式

球面三角公式

设 a,b,c 是边长，A,B,C 是所对的二面角，有余弦定理

$$cosa=cosb\cdot cosc+sinb\cdot sinc\cdot cosA$$

正弦定理

$$\frac{sinA}{sina}=\frac{sinB}{sinb}=\frac{sinC}{sinc}$$

三角形面积是 $A+B+C-\pi$

四面体体积公式

U,V,W,u,v,w 是四面体的 6 条棱，U,V,W 构成三角形，$(U,u),(V,v),(W,w)$ 互为对棱，则

$$V=\frac{\sqrt{(s-2a)(s-2b)(s-2c)(s-2d)}}{192uvw}$$

其中

$$\left\{\begin{array}{lcl}
a & = & \sqrt{xYZ},\\
b & = & \sqrt{yZX},\\
c & = & \sqrt{zXY},\\
d & = & \sqrt{xyz},\\
s & = & a+b+c+d,\\
X & = & (w-U+v)(U+v+w),\\
x & = & (U-v+w)(v-w+U),\\
Y & = & (u-V+w)(V+w+u),\\
y & = & (V-w+u)(w-u+V),\\
Z & = & (v-W+u)(W+u+v),\\
z & = & (W-u+v)(u-v+W)
\end{array}\right.$$

附录

NTT 素数及原根列表

Id	Primes	PRT	Id	Primes	PRT	Id	Primes	PRT
1	7340033	3	38	311427073	7	75	786432001	7
2	13631489	15	39	330301441	22	76	799014913	13
3	23068673	3	40	347078657	3	77	800063489	3
4	26214401	3	41	359661569	3	78	802160641	11
5	28311553	5	42	361758721	29	79	818937857	5
6	69206017	5	43	377487361	7	80	824180737	5
7	70254593	3	44	383778817	5	81	833617921	13
8	81788929	7	45	387973121	6	82	850395137	3
9	101711873	3	46	399507457	5	83	862978049	3
10	104857601	3	47	409993217	3	84	880803841	26
11	111149057	3	48	415236097	5	85	883949569	7
12	113246209	7	49	447741953	3	86	897581057	3
13	120586241	6	50	459276289	11	87	899678209	7
14	132120577	5	51	463470593	3	88	907018241	3
15	136314881	3	52	468713473	5	89	913309697	3
16	138412033	5	53	469762049	3	90	918552577	5
17	141557761	26	54	493879297	10	91	919601153	3
18	147849217	5	55	531628033	5	92	924844033	5
19	155189249	6	56	576716801	6	93	925892609	3
20	158334977	3	57	581959681	11	94	935329793	3
21	163577857	23	58	595591169	3	95	938475521	3
22	167772161	3	59	597688321	11	96	940572673	7
23	169869313	5	60	605028353	3	97	943718401	7
24	185597953	5	61	635437057	11	98	950009857	7
25	186646529	3	62	639631361	6	99	957349889	6
26	199229441	3	63	645922817	3	100	962592769	7
27	204472321	19	64	648019969	17	101	972029953	10
28	211812353	3	65	655360001	3	102	975175681	17
29	221249537	3	66	666894337	5	103	976224257	3
30	230686721	6	67	683671553	3	104	985661441	3
31	246415361	3	68	710934529	17	105	998244353	3
32	249561089	3	69	715128833	3	106	1004535809	3
33	257949697	5	70	718274561	3	107	1007681537	3
34	270532609	22	71	740294657	3	108	1012924417	5
35	274726913	3	72	745537537	5	109	1045430273	3
36	290455553	3	73	754974721	11	110	1051721729	6
37	305135617	5	74	770703361	11	111	1053818881	7