Contents

李沐阳

AC 自动机

```
//ac 自动机
inline int newnode() { memset(nxt[L],-1,sizeof(nxt[L])); return ++L-1; }
inline void init() { L = 0; root = newnode(); }
inline void insert() {
  int len = strlen(buf),now = root;
  for (int i = 0;i < len;++i) {
    if (nxt[now][buf[i]-'0'] == -1) nxt[now][buf[i]-'0'] = newnode();
    now = nxt[now][buf[i]-'0'];
  }
  end[now] = true;
}
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]);
      }
    }
  }
}
```

Dinic Algorithm

```
// dinic
int source,sink,cnt = 1;
int d[maxv],side[maxv],cur[maxv],side[maxe],nxt[maxe],toit[maxe],cap[maxe]; bool in[maxv];
inline void add(int a,int b,int c) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; cap[cnt] = c; }
inline void ins(int a,int b,int c) { add(a,b,c); add(b,a,0); }
inline bool bfs() {
  queue <int> team; team.push(source); d[source] = 0;
  memset(in,false,tot+10); in[source] = true; team.push(source);
  while (!team.empty()) {
    int now = team.front(); team.pop(); cur[now] = side[now];
    for (int i = side[now];i;i = nxt[i]) {
      if (!cap[i]) continue;
      if (!in[toit[i]])
        in[toit[i]] = true,d[toit[i]] = d[now]+1,team.push(toit[i]);
    }
  }
  return in[sink];
}
inline int dfs(int now,int f) {
  if (now == sink||!f) return f;
  int used = 0,w;
  for (int &i = cur[now];i;i = nxt[i])
    if (cap[i]&&d[toit[i]] == d[now]+1) {
      w = dfs(toit[i],min(cap[i],f-used));
      used += w; cap[i] -= w; cap[i^1] += w;
      if (used == f) break;
    }
  return used;
}
inline int dinic(int S,int T) {
  source = S; sink = T; int ret = 0;
  while (bfs()) ret += dfs(source,inf);
```

```
  return ret;
}
```

K-Dimension Tree

```
struct Point { double x,y; int id; }P[maxn];
struct Rectangle
{
  double lx,rx,ly,ry;
  inline Rectangle() = default;
  inline Rectangle(double _lx,double _rx,double _ly,double _ry):lx(_lx),rx(_rx),ly(_ly),ry(_ry) {}
  inline void set(const Point &p) { lx = rx = p.x; ly = ry = p.y; }
  inline void merge(const Point &p)
  {
    lx = min(lx,p.x); rx = max(rx,p.x);
    ly = min(ly,p.y); ry = max(ry,p.y);
  }
  inline void merge(const Rectangle &r)
  {
    lx = min(lx,r.lx); rx = max(rx,r.rx);
    ly = min(ly,r.ly); ry = max(ry,r.ry);
  }
  // 最小距离，到 4 个角和 4 条边距离
  inline double dist(const Point &p)
  {
    if (p.x <= lx&&p.y <= ly) return (p-Point(lx,ly)).norm();
    else if (p.x <= rx&&p.y <= ly) return p.y-ly;
    else if (p.x >= rx&&p.y <= ly) return (p-Point(rx,ly)).norm();
    else if (p.x >= rx&&p.y <= ry) return p.x-rx;
    else if (p.x >= rx&&p.y >= ry) return (p-Point(rx,ry)).norm();
    else if (p.x >= lx&&p.y >= ry) return p.y-ry;
    else if (p.x <= lx&&p.y >= ry) return (p-Point(lx,ry)).norm();
    else if (p.x <= lx&&p.y >= ly) return p.x-lx;
    return 0;
  }
  // 最大距离，到 4 个角的距离
  inline double dist(const Point &p)
  {
    double ret = 0;
    ret += max((rx-p.x)*(rx-p.x),(lx-p.x)*(lx-p.x));
    ret += max((ry-p.y)*(ry-p.y),(ly-p.y)*(ly-p.y));
    return ret;
  }
};

struct Node {
  int child[2]; Point p; Rectangle r;
  inline Node() = default;
  inline Node(const Point &_p,const Rectangle &_r):p(_p),r(_r) { r.set(p); memset(child,0,8); }
  inline void set(const Point &_p) { p = _p; r.set(p); memset(child,0,8); }
}tree[maxn];

inline bool cmpx(const Point &a,const Point &b) {
  if (a.x != b.x) return a.x < b.x;
  else return a.y < b.y;
}
inline bool cmpy(const Point &a,const Point &b) {
  if (a.y != b.y) return a.y < b.y;
  else return a.x < b.x;
}
inline bool cmp(pair <double,int> a,pair <double,int> b) {
  int sgn = dcmp(a.first-b.first);
  if (sgn) return sgn < 0;
  else return a.second < b.second;
}
// 查询 k 大/小
inline void query(int now,const Point &p,int k,pair <double,int> ret[],bool dim = false) {
  if (dcmp(tree[now].r.dist(p)-ret[k].first) > 0) return;
  pair <double,int> val = make_pair((p-tree[now].p).norm(),tree[now].p.id);
  for (int i = 1;i <= k;++i) if (cmp(val,ret[i])) {
      for (int j = k+1;j > i;--j) ret[j] = ret[j-1];
      ret[i] = val; break;
    }
  if ((dim&&cmpx(p,tree[now].p))||(!dim&&cmpy(p,tree[now].p))) {
    if (tree[now].child[0]) query(tree[now].child[0],p,k,ret,dim^1);
    if (tree[now].child[1]) query(tree[now].child[1],p,k,ret,dim^1);
  } else {
    if (tree[now].child[1]) query(tree[now].child[1],p,k,ret,dim^1);
    if (tree[now].child[0]) query(tree[now].child[0],p,k,ret,dim^1);
  }
}
inline int build(int l,int r,bool dim) {
  int now = ++size,mid = (l+r)>>1;
  nth_element(vec.begin()+l-1,vec.begin()+mid-1,vec.begin()+r,dim?cmpx:cmpy);
  tree[now].set(vec[mid-1]);
  if (l < mid) {
    tree[now].child[0] = build(l,mid-1,dim^1);
    tree[now].r.merge(tree[tree[now].child[0]].r);
  }
  if (r > mid) {
    tree[now].child[1] = build(mid+1,r,dim^1);
    tree[now].r.merge(tree[tree[now].child[1]].r);
  }
  return now;
}
```

KM

```
// 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
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; }
```

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);
}
```

Splay

```
const int maxn = 500010,inf = 1<<29;
int N,M,root,cnt,arr[maxn],tag[maxn],key[maxn],fa[maxn],ch[maxn][2],lb[maxn],rb[maxn];
int wb[maxn],sum[maxn],size[maxn],stk[maxn]; bool rev[maxn]; char cmd[20]; queue <int> team;

inline int gi()
{
  char ch; int ret = 0,f = 1;
  do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
  if (ch == '-') f = -1,ch = getchar();
  do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
  return ret*f;
}

inline int newnode(int x = 0)
{
  int ret;
  if (!team.empty())
    ret = team.front(),team.pop();
  else ret = ++cnt;
  key[ret] = sum[ret] = lb[ret] = rb[ret] = wb[ret] = x;
  rev[ret] = false; tag[ret] = inf; size[ret] = 1;
  return ret;
}

inline void pushdown(int now) { }

inline void update(int now)
{
  // pushdown(now);
  int lc = ch[now][0],rc = ch[now][1];
  size[now] = size[lc]+size[rc]+1;
  sum[now] = sum[lc]+sum[rc]+key[now];
  if (lc&&rc) { }
  else if (lc) { }
  else if (rc) { }
  else { }
}

inline int build(int l,int r)
{
  int mid = (l+r) >> 1,ret = newnode(arr[mid]);
  if (l < mid) ch[ret][0] = build(l,mid-1),fa[ch[ret][0]] = ret;
  if (r > mid) ch[ret][1] = build(mid+1,r),fa[ch[ret][1]] = ret;
  update(ret); return ret;
}

inline void init()
{
  root = newnode(); ch[root][1] = newnode(); fa[2] = 1;
  for (int i = 1;i <= N;++i) arr[i] = gi();
  ch[2][0] = build(1,N); fa[ch[2][0]] = 2;
  update(2); update(1);
}

inline int find(int rk)
{
  for (int now = root;;)
  {
    pushdown(now);
    if (rk == size[ch[now][0]]+1) return now;
    else if (rk > size[ch[now][0]]+1)
      rk -= size[ch[now][0]]+1,now = ch[now][1];
    else now = ch[now][0];
  }
  return 0;
}

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

```
  }
  if (!aim) root = x;
}

inline void Delete(int &now)
{
  if (!now) return;
  Delete(ch[now][0]);
  Delete(ch[now][1]);
  team.push(now); now = 0;
}

inline void print()
{
  for (int i = 1;i <= cnt;++i)
    printf("%d:%d %d\n",i,ch[i][0],ch[i][1]);
  for (int i = 1;i <= cnt;++i)
    printf("%d:%d\n",i,fa[i]);

}

inline void laydown(int now)
{
  if (!now) return;
  pushdown(now);
  laydown(ch[now][0]);
  printf("%d ",key[now]);
  laydown(ch[now][1]);
  update(now);
}
```

几何基础

```
//计算几何常用公式
struct Point {  double x,y; inline Point unit() const {
    double len = norm(); if (!dcmp(len)) return Point(1,0); else return *this/len; }
  inline Point reflect(const Point &p) const {
    Point v = *this-p; double len = v.norm();
    v = v/len; return p+v*(1/len); }
  inline Point vertical() const { return Point(y,-x); }
  inline double angle() const {
    double ret = atan2(y,x); if (ret < 0) ret += 2*pi; return ret; }
};
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(); }
    friend inline bool operator <(const Line &l1,const Line &l2)
  { return l1.slop < l2.slop; }
  inline double dis(const Point &a) { fabs((a-p)/v)/(v.len()); } //点到直线距离
};

inline bool OnLine(const Line &l,const Point &p) { return !dcmp(l.v/(p-l.p)); } //点在直线上

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

inline bool parallel(const Line &a,const Line &b) { return !dcmp(a.v/b.v); } //直线平行
```

凸包

```
struct Point
{
    inline Point() = default;
  inline Point(double _x,double _y):x(_x),y(_y) {}
  inline Point unit() const
  {
    double len = norm();
    if (!dcmp(len)) return Point(1,0);
    else return *this/len;
  }
  inline double norm() const { return sqrt(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);
  }
  inline void read() { scanf("%lf %lf",&x,&y); }
  inline Point vertical() const { return Point(y,-x); }
  inline double angle() const
  {
    double ret = atan2(y,x);
    if (ret < 0) ret += 2*pi;
    return ret;
  }
  friend inline bool operator ==(const Point &a,const Point &b) { return
  ↪   !dcmp(a.x-b.x)&&!dcmp(a.y-b.y); }
    friend inline Point operator -(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend inline Point operator +(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
  friend inline Point operator /(const Point &a,double b) { return Point(a.x/b,a.y/b); }
  friend inline Point operator *(const Point &a,double b) { return Point(a.x*b,a.y*b); }
  friend inline Point operator *(double b,const Point &a) { return Point(a.x*b,a.y*b); }
  friend inline double operator /(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
  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--;
}
```

无向图最小割

```
int node[N], dist[N];
bool visit[N];
int solve(int n) {
  int answer = INT_MAX;
  for (int i = 0; i < n; ++i) node[i] = i;
  while (n > 1) {
    int max = 1;
    for (int i = 0; i < n; ++i) {
      dist[node[i]] = graph[node[0]][node[i]];
      if (dist[node[i]] > dist[node[max]]) max = i;
    }
    int prev = 0;
    memset(visit, 0, sizeof(visit));
    visit[node[0]] = true;
    for (int i = 1; i < n; ++i) {
      if (i == n - 1) {
        answer = std::min(answer, dist[node[max]]);
        for (int k = 0; k < n; ++k) {
          graph[node[k]][node[prev]] = (graph[node[prev]][node[k]] += graph[node[k]][node[max]]);
        }
        node[max] = node[--n];
      }
      visit[node[max]] = true;
      prev = max; max = -1;
      for (int j = 1; j < n; ++j) {
        if (!visit[node[j]]) {
          dist[node[j]] += graph[node[prev]][node[j]];
          if (max == -1 || dist[node[max]] < dist[node[j]]) {
            max = j;
          }
        }
      }
    }
  }
  return answer;
}
```

匈牙利

```
//匈牙利算法
//Version1
inline bool find(int x)
{
```

```
  if (cor[x]) return false;
  for (int i = side[x];i;i = next[i]) if (!used[toit[i]])
  {
    used[toit[i]] = true;
    if (!cho[toit[i]]||find(cho[toit[i]]))
    {
      cho[toit[i]] = x; map[x] = toit[i];
      return true;
    }
  }
  return false;
}

inline void hungry()
{
  for (int i = 1;i <= p;++i)
    memset(used,false,sizeof(used)),find(i);
  for (int i = 1;i <= m;++i)
  {
    memset(used,false,sizeof(used)),cho[map[i]] = 0;
    find(i),cor[i] = true;
  }
}
//Version2
inline int find(int x)
{
    for (int i = 1;i <= n;++i)
        if (f[x][i]&&!used[i])
        {
            used[i] = true;
            if (!cho[i]||find(cho[i])) { cho[i] = x; return true; }
        }
    return false;
}

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

半平面交

```
//半平面交，直线左侧半平面，注意最后是 tail-head <= 0 还是 tail-head <= 1
inline int dcmp(double a)
{
  if (-eps <= a&&a <= eps) return 0;
  else if (a > 0) return 1; else return -1;
}

struct Point
{
  double x,y;
  inline Point() = default;
  inline Point(double _x,double _y):x(_x),y(_y) {}
  inline void read() { x = gi(),y = gi(); }
  inline Point vertical() const { return Point(-y,x); }
  inline Point unit() const
  {
    double len = norm();
    if (!dcmp(len)) return Point(1,0);
    else return *this/len;
  }
  inline double norm() const { return sqrt(x*x+y*y); }
  inline double angle() const { return atan2(y,x); }
  friend inline Point operator+(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
  friend inline Point operator-(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend inline Point operator*(const Point &a,double b) { return Point(a.x*b,a.y*b); }
  friend inline Point operator*(double b,const Point &a) { return Point(a.x*b,a.y*b); }
  friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.x; }
}P[maxn],pp[maxn],pol[maxn];

struct Line
{
  Point p,v;
  inline Line(const Point _p = Point(),const Point _v = Point()):p(_p),v(_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 bool parallel(const Line &a,const Line &b) { return !dcmp(a.v/b.v); }
inline Point crosspoint(const Line &a,const Line &b)
{
  Point u = a.p-b.p;
  double t = (b.v/u)/(a.v/b.v);
  return a.p+(a.v*t);
}

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;
}
```

后缀数组

```
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;
    }
}
```

圆与多边形面积交

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

inline int dcmp(double a)
{
  if (a > eps) return 1;
```

```
  else if (a < -eps) return -1;
  else return 0;
}

struct Point
{
  double x,y;
  Point() = default;
  Point(double _x,double _y):x(_x),y(_y) {}
  inline double norm() const { return sqrt(x*x+y*y); }
  inline Point unit() const { double len = norm(); return Point(x/len,y/len); }
  friend Point operator +(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
  friend Point operator -(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend Point operator *(const Point &a,double b) { return Point(a.x*b,a.y*b); }
  friend Point operator *(double b,const Point &a) { return Point(a.x*b,a.y*b); }
  friend Point operator /(const Point &a,double b) { return Point(a.x/b,a.y/b); }
  friend double operator /(const Point &a,const Point &b) { return a.x*b.y-b.x*a.y; }
  friend double operator *(const Point &a,const Point &b) { return a.x*b.x+a.y*b.y; }
  inline void read() { scanf("%lf %lf",&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);
}
```

圆的 K 次交

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

inline int dcmp(double a)
{
  if (-eps <= a&&a <= eps) return 0;
  else if (a > 0) return 1; else return -1;
}

struct Point
{
  double x,y;
  inline Point() = default;
  inline Point(double _x,double _y):x(_x),y(_y) {}
  inline void read() { x = gi(),y = gi(); }
  inline double norm() const { return sqrt(x*x+y*y); }
  inline double angle() const { return atan2(y,x); }
  inline Point unit() const { double len = norm(); return Point(x/len,y/len); }
  friend inline Point operator-(const Point &a,const Point &b) { return Point(a.x-b.x,a.y-b.y); }
  friend inline Point operator+(const Point &a,const Point &b) { return Point(a.x+b.x,a.y+b.y); }
  friend inline Point operator*(const Point &a,double b) { return Point(a.x*b,a.y*b); }
  friend inline Point operator*(double b,const Point &a) { return Point(a.x*b,a.y*b); }
  friend inline Point operator/(const Point &a,double b) { return Point(a.x/b,a.y/b); }
  friend inline double operator/(const Point &a,const Point &b) { return a.x*b.y-a.y*b.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];
    }
}
```

最小覆盖圆

```
circle minimum_circle(vector<point> p) {
  circle ret;
  random_shuffle(p.begin(), p.end());
  for (int i = 0; i < (int)p.size(); ++i)
    if (!in_circle(p[i], ret)) {
      ret = circle(p[i], 0);
      for (int j = 0; j < i; ++j)
        if (!in_circle(p[j], ret)) {
          ret = make_circle(p[j], p[i]);
          for (int k = 0; k < j; ++k)
            if (!in_circle(p[k], ret)) ret = make_circle(p[i], p[j], p[k]);
        }
    }
  return ret;
}
```

三维凸包

```
struct Triangle{ // Construction function removed.
  TPoint a, b, c;
  double getArea() {
    TPoint ret = det(b - a, c - a);
    return dist(ret) / 2.0;
  }
};
namespace Convex_Hull {
  struct Face{ // Construction function removed.
    int a, b, c;
    bool isOnConvex;
  };
  int nFace, left, right, whe[MAXN][MAXN];
  Face queue[MAXF], tmp[MAXF];
  bool isVisible(const std::vector<TPoint> &p, const Face &f, const TPoint &a) {
    return dcmp(detdot(p[f.a], p[f.b], p[f.c], a)) > 0;
  }
  bool init(std::vector<TPoint> &p) {
    bool check = false;
    for (int i = 1; i < (int)p.size(); i++) {
      if (dcmp(sqrdist(p[0], p[i]))) {
        std::swap(p[1], p[i]);
        check = true;
        break;
      }
    }
    if (!check) return false;
    check = false;
    for (int i = 2; i < (int)p.size(); i++) {
      if (dcmp(sqrdist(det(p[i] - p[0], p[1] - p[0])))) {
```

```
        std::swap(p[2], p[i]);
        check = true;
        break;
      }
    }
    if (!check) return false;
    check = false;
    for (int i = 3; i < (int)p.size(); i++) {
      if (dcmp(detdot(p[0], p[1], p[2], p[i])) {
        std::swap(p[3], p[i]);
        check = true;
        break;
      }
    }
    if (!check) return false;
    for (int i = 0; i < (int)p.size(); i++)
      for (int j = 0; j < (int)p.size(); j++) {
        whe[i][j] = -1;
      }
    return true;
  }
  void pushface(const int &a, const int &b, const int &c) {
    nFace++;
    tmp[nFace] = Face(a, b, c);
    tmp[nFace].isOnConvex = true;
    whe[a][b] = nFace;
    whe[b][c] = nFace;
    whe[c][a] = nFace;
  }
  bool deal(const std::vector<TPoint> &p, const std::pair<int, int> &now, const TPoint &base) {
    int id = whe[now.second][now.first];
    if (!tmp[id].isOnConvex) return true;
    if (isVisible(p, tmp[id], base)) {
      queue[++right] = tmp[id];
      tmp[id].isOnConvex = false;
      return true;
    }
    return false;
  }
  std::vector<Triangle> getConvex(std::vector<TPoint> &p) {
    static std::vector<Triangle> ret;
    ret.clear();
    if (!init(p)) return ret;
    if (!isVisible(p, Face(0,1,2),p[3])) pushface(0,1,2); else pushface(0,2,1);
    if (!isVisible(p, Face(0,1,3),p[2])) pushface(0,1,3); else pushface(0,3,1);
    if (!isVisible(p, Face(0,2,3),p[1])) pushface(0,2,3); else pushface(0,3,2);
    if (!isVisible(p, Face(1,2,3),p[0])) pushface(1,2,3); else pushface(1,3,2);
    for (int a = 4; a < (int)p.size(); a++) {
      TPoint base = p[a];
      for (int i = 1; i <= nFace; i++) {
        if (tmp[i].isOnConvex && isVisible(p, tmp[i], base)) {
          left = 0, right = 0;
          queue[++right] = tmp[i];
          tmp[i].isOnConvex = false;
          while (left < right) {
            Face now = queue[++left];
            if (!deal(p,std::make_pair(now.a,now.b),base)) pushface(now.a,now.b,a);
            if (!deal(p,std::make_pair(now.b,now.c),base)) pushface(now.b,now.c,a);
            if (!deal(p,std::make_pair(now.c,now.a),base)) pushface(now.c,now.a,a);
          }
          break;
        }
      }
    }
    for (int i = 1; i <= nFace; i++) {
      Face now = tmp[i];
      if (now.isOnConvex)ret.push_back(Triangle(p[now.a],p[now.b],p[now.c]));
    }
    return ret;
  }
};
// Usage
std::vector<TPoint> p;
std::vector<Triangle> answer;
answer = Convex_Hull::getConvex(p);
```

三维绕轴旋转

注意事项：以右手拇指为向量方向，逆时针绕轴（剩下四根手指方向）旋转 θ 角的右乘矩阵。

```
Matrix getTrans(const double &a, const double &b, const double &c) {
    Matrix ret;
    ret.a[0][0] = 1; ret.a[0][1] = 0; ret.a[0][2] = 0; ret.a[0][3] = 0;
    ret.a[1][0] = 0; ret.a[1][1] = 1; ret.a[1][2] = 0; ret.a[1][3] = 0;
    ret.a[2][0] = 0; ret.a[2][1] = 0; ret.a[2][2] = 1; ret.a[2][3] = 0;
    ret.a[3][0] = a; ret.a[3][1] = b; ret.a[3][2] = c; ret.a[3][3] = 1;
    return ret;
}
Matrix getRotate(const double &a, const double &b, const double &c, const double &theta) {
    Matrix ret;
    ret.a[0][0] = a * a * (1 - cos(theta)) + cos(theta);
    ret.a[0][1] = a * b * (1 - cos(theta)) + c * sin(theta);
    ret.a[0][2] = a * c * (1 - cos(theta)) - b * sin(theta);
    ret.a[0][3] = 0;
    ret.a[1][0] = b * a * (1 - cos(theta)) - c * sin(theta);
    ret.a[1][1] = b * b * (1 - cos(theta)) + cos(theta);
    ret.a[1][2] = b * c * (1 - cos(theta)) + a * sin(theta);
    ret.a[1][3] = 0;
    ret.a[2][0] = c * a * (1 - cos(theta)) + b * sin(theta);
    ret.a[2][1] = c * b * (1 - cos(theta)) - a * sin(theta);
    ret.a[2][2] = c * c * (1 - cos(theta)) + cos(theta);
    ret.a[2][3] = 0;
    ret.a[3][0] = 0; ret.a[3][1] = 0; ret.a[3][2] = 0; ret.a[3][3] = 1;
    return ret;
}
Matrix getRotate(const double &ax, const double &ay, const double &az, const double &bx, const double
 ↪   &by, const double &bz, const double &theta) {
    double l = dist(Point(0, 0, 0), Point(bx, by, bz));
    Matrix ret = getTrans(-ax, -ay, -az);
    ret = ret * getRotate(bx / l, by / l, bz / l, theta);
    ret = ret * getTrans(ax, ay, az);
    return ret;
}
```

平面图

```
// 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];
      }
    }
    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;
    }
  }
}
```

弦图染色最大势

```
#include<algorithm>
#include<queue>
#include<cstdio>
#include<cstdlib>
#include<set>
using namespace std;

#define maxn 10010
#define maxc 510
#define maxm 1000010
int tot,n,m,cnt,color[maxn][maxc],label[maxn],all;
int side[maxn],next[maxm*2],toit[maxm*2],per[maxn];
bool in[maxn];
struct node
{
  int key,ord;
  friend bool operator < (node a,node b) {return a.key > b.key; }
};
multiset <node> S;

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

inline void ins(int a,int b){add(a,b); add(b,a);}

inline void mcs()
{
  int i,u;
  for (i = 1;i <= n;++i) S.insert((node){0,i});
  while (all < n)
  {
    u = (*S.begin()).ord; S.erase(S.begin()); if (in[u]) continue;
    in[u] = true; per[++all] = u;
    for (i = side[u];i;i = next[i])
      if (!in[toit[i]])
      {
        label[toit[i]]++;
        S.insert((node){label[toit[i]],toit[i]});
      }
  }
}

inline void paint()
{
  int p,i,j,t;
  for (p = 1;p <= n;++p)
  {
    i = per[p];
    for (j = 1;j <= tot;++j)
      if (!color[i][j]) {t = j; break; }
    if (j == tot + 1) t = ++tot;
    for (j = side[i];j;j = next[j])
      color[toit[j]][t] = true;
  }
}

int main()
{
  freopen("1006.in","r",stdin);
  freopen("1006.out","w",stdout);
  scanf("%d %d",&n,&m);
  for (int i = 1;i <= m;++i)
  { int a,b; scanf("%d %d",&a,&b); ins(a,b); }
  mcs();
  paint();
  printf("%d",tot);
  fclose(stdin); fclose(stdout);
  return 0;
}
```

强连通分量

```
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();
  }
}
```

支配树

```
//建出来的树点的编号 i 在原图中是 redfn[i]
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 int gi()
{
  char ch; int ret = 0,f = 1;
  do ch = getchar(); while (!(ch >= '0'&&ch <= '9')&&ch != '-');
  if (ch == '-') f = -1,ch = getchar();
  do ret = ret*10+ch-'0',ch = getchar(); while (ch >= '0'&&ch <= '9');
  return ret*f;
}

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);
  }
}
```

点双联通分量

```
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);
  }
}
```

线性规划

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

#define maxn (30)
#define eps (1e-8)

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()
{
```

```
  freopen("179.in","r",stdin);
  freopen("179.out","w",stdout);
  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]);
  fclose(stdin); fclose(stdout);
  return 0;
}
```

费用流

```
int side[maxv],nxt[maxe],toit[maxe],cost[maxe],pre[maxv];
int cap[maxv],arr[maxv],dis[maxv]; bool in[maxv];
int source,sink;

inline void add(int a,int b,int c,int d) { nxt[++cnt] = side[a]; side[a] = cnt; toit[cnt] = b; cap[cnt]
 ↪  = c; cost[cnt] = d; }
inline void ins(int a,int b,int c,int d) { add(a,b,c,d); add(b,a,0,-d); }

inline bool spfa(int &Flow,int &Cost)
{
    queue <int> team; team.push(source);
    memset(dis,0x7,4*(sink+5));
    dis[source] = 0; in[source] = true;
  arr[source] = inf; arr[sink] = 0;
    while (!team.empty())
    {
        int now = team.front(); team.pop();
        for (int i = side[now];i;i = nxt[i])
        {
            if (!cap[i]) continue;
            if (dis[toit[i]] > dis[now]+cost[i])
            {
                arr[toit[i]] = min(cap[i],arr[now]); pre[toit[i]] = i;
                dis[toit[i]] = dis[now]+cost[i];
                if (!in[toit[i]]) in[toit[i]] = true,team.push(toit[i]);
            }
        }
        in[now] = false;
    }
    if (!arr[sink]) return false;
    Flow += arr[sink];
    for (int now = sink,i;now != source;now = toit[i^1])
    {
        i = pre[now]; Cost += cost[pre[now]]*arr[sink];
        cap[i] -= arr[sink]; cap[i^1] += arr[sink];
    }
    return true;
}
```

直线下整点个数

注意事项： 返回结果为：$\sum_{0 \le i < n} \lfloor \frac{a + b \cdot i}{m} \rfloor$ 即直线下整点个数。

```
long long solve(const long long &n, const long long &a,
                const long long &b, const long long &m) {
  if (b == 0) return n * (a / m);
  if (a >= m) return n * (a / m) + solve(n, a % m, b, m);
  if (b >= m) return (n - 1) * n / 2 * (b / m) + solve(n, a, b % m, m);
  return solve((a + b * n) / m, (a + b * n) % m, m, b);
}
```

闪电素数判定

```
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool check(const long long &prime, const long long &base) {
  long long number = prime - 1;
  for (; ~number & 1; number >>= 1);
  long long result = power_mod(base, number, prime);
  for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1) {
    result = multiply_mod(result, result, prime);
  }
  return result == prime - 1 || (number & 1) == 1;
}
bool miller_rabin(const long long &number) {
  if (number < 2) return false;
  if (number < 4) return true;
  if (~number & 1) return false;
  for (int i = 0; i < 12 && BASE[i] < number; ++i) {
    if (!check(number, BASE[i])) {
      return false;
    }
  }
  return true;
}
```

闪电质因数分解

```
long long pollard_rho(const long long &number, const long long &seed) {
  long long x = rand() % (number - 1) + 1, y = x;
  for (int head = 1, tail = 2; ; ) {
    x = multiply_mod(x, x, number);
    x = add_mod(x, seed, number);
    if (x == y) return number;
    long long answer = std::__gcd(abs(x - y), number);
    if (answer > 1 && answer < number) return answer;
    if (++head == tail) {
      y = x;
      tail <<= 1;
    }
  }
}
void factorize(const long long &number, std::vector<long long> &divisor) {
  if (number > 1) {
    if (miller_rabin(number)) divisor.push_back(number);
    else {
      long long factor = number;
      for (; factor >= number; factor = pollard_rho(number, rand() % (number - 1) + 1));
      factorize(number / factor, divisor);
      factorize(factor, divisor);
    }
  }
}
```

自适应辛普森

```
double area(const double &left, const double &right) {
  double mid = (left + right) / 2;
  return (right - left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
}
double simpson(const double &left, const double &right,
               const double &eps, const double &area_sum) {
  double mid = (left + right) / 2;
  double area_left = area(left, mid), area_right = area(mid, right);
  double area_total = area_left + area_right;
  if (std::abs(area_total - area_sum) < 15 * eps) {
    return area_total + (area_total - area_sum) / 15;
  }
  return simpson(left, mid, eps / 2, area_left) + simpson(mid, right, eps / 2, area_right);
}
double simpson(const double &left, const double &right, const double &eps) {
  return simpson(left, right, eps, area(left, right));
}
```

孙司宇

FFT

```cpp
#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

```cpp
#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

```
// 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

```
//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

```
//    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;
        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

```
//³  🄵 bzoj3456
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#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

```
#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>
```

```cpp
#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

```cpp
//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

```cpp
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

```cpp
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;
  }
  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

```
/*
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
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 $\{p_1,p_2,\dots,p_t\}$，则 $\{p_1\cup N(p_1),\dots,p_t\cup N(p_t)\}$ 为最小团覆盖

Bernoulli 数

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

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. $\frac{1}{(1-x)^{n+1}}=\sum_{i=0}^{n}\binom{i+n}{i}x^i$
10. $\frac{1}{\sqrt{1-4x}}=\sum_{i=0}^{n}\binom{2i}{i}x^i$

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

$$\mu(n)=\begin{cases}1 & 若n=1\\(-1)^k & 若n无平方数因子，且n=p_1p_2\ldots p_k\\0 & 若n有大于1的平方数因数\end{cases}$$

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

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

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

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 $|X/G|=\frac{1}{|G|}\sum_{g\in G}|X^g|$.

五边形数定理

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

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 $a_n-\sum_{i=1}^{n/2}a_ia_{n-i}$，当 n 为偶数时，n 个结点的无根树的个数为 $a_n-\sum_{i=1}^{n/2}a_ia_{n-i}+\frac{1}{2}a_{\frac{n}{2}}(a_{\frac{n}{2}}+1)$
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：$V = \frac{2}{3}\pi r^2h$

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

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

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