Killer_47 模板

目录

图论

2-SAT

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <stack>
using namespace std;
const int INF = 0x7F7F7F7F;
const int MAXN = 1000 + 10;
const int MAXM = 1000000 + 10;

struct Edge{ int to, next; };
Edge es[MAXM];
int head[MAXN], low[MAXN], dfn[MAXN], belong[MAXN], a[MAXN], b[MAXN];
bool insta[MAXN];
int n, m, cnt, index;
stack<int> sta;

void add( int u, int v ) {
    es[cnt].to = v; es[cnt].next = head[u]; head[u] = cnt++;
    return ;
}

void tarjan( int u ) {
    int v;
    dfn[u] = low[u] = index++;
    sta.push( u );
    insta[u] = true;
    for( int i = head[u]; ~i; i = es[i].next ) {
        v = es[i].to;
        if( dfn[v] == -1 ) {
            tarjan( v );
            low[u] = min( low[u], low[v] );
        } else if( insta[v] ) {
            low[u] = min( low[u], dfn[v] );
        }
    }
    if( dfn[u] == low[u] ) {
        do {
            v = sta.top(); sta.pop();
            insta[v] = false;
```

```
            belong[v] = cnt;
        } while( u != v );
        ++cnt;
    }
    return ;
}

int main() {
    scanf( "%d%d", &n, &m );
    memset( head, -1, sizeof( head ) );
    memset( dfn, -1, sizeof( dfn ) );
    memset( low, -1, sizeof( low ) );
    memset( insta, false, sizeof( insta ) );
    memset( belong, -1, sizeof( belong ) );
    cnt = 0;
    for( int i = 0; i < m; ++i ) {
        scanf( "%d%d", a + i, b + i );
        if( a[i] > b[i] ) swap( a[i], b[i] );
    }
    for( int i = 0; i < m; ++i ) {
        for( int j = i + 1; j < m; ++j ) {
            if( ( a[i] < a[j] && b[i] < b[j] && a[j] < b[i] ) ||
               ( a[i] > a[j] && b[i] > b[j] && b[j] > a[i] ) ) {
                add( i * 2, j * 2 + 1 );
                add( j * 2, i * 2 + 1 );
                add( i * 2 + 1, j * 2 );
                add( j * 2 + 1, i * 2 );
            }
        }
    }
    index = 1; cnt = 1;
    for( int i = 0; i < m * 2; ++i ) {
        if( dfn[i] == -1 ) tarjan( i );
    }
    bool flag = true;
    for( int i = 0; i < m * 2; ++i ) {
        if( belong[i] == belong[i ^ 1] ) {
            flag = false;
            break;
        }
    }
    if( flag ) printf( "panda is telling the truth...\n" );
    else printf( "the evil panda is lying again\n" );
    return 0;
```

```
}
```

最短路次短路相关

AStar

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
const int INF = 0x7F7F7F7F;
const int MAXN = 10000 + 10;
const int MAXE = 1000000 + 10;

struct Edge { int to, cost, next; };
Edge es[MAXE];
struct Node { int u, f, g; };
int head[MAXN], h[MAXN];
bool vis[MAXN];
int n, m, k, cnt;
vector<int> path;

struct cmp {
    bool operator() ( const Node &a, const Node &b ) {
        return a.f > b.f;
    }
};

void add( int u, int v, int w ) {
    es[cnt].to = v; es[cnt].cost = w; es[cnt].next = head[u]; head[u] = cnt++;
    return ;
}

void spfa() {
    queue<int> que;
    memset( vis, false, sizeof( vis ) );
    memset( h, 0x7F, sizeof( h ) );
    que.push( n ); h[n] = 0; vis[n] = true;
    while( !que.empty() ) {
        int u = que.front(); que.pop();
        for( int i = head[u]; ~i; i = es[i].next ) {
            int v = es[i].to;
```

```
            if( h[v] > h[u] + es[i].cost ) {
                h[v] = h[u] + es[i].cost;
                if( !vis[v] ) {
                    vis[v] = true;
                    que.push( v );
                }
            }
        }
        vis[u] = false;
    }
    return ;
}

void astar() {
    priority_queue<Node, vector<Node>, cmp> que;
    Node tmp = { 1, h[1], 0 };
    que.push( tmp );
    for( int cur = 0; cur < k && !que.empty(); ) {
        tmp = que.top(), que.pop();
        int u = tmp.u, g = tmp.g;
        if( u == n ) { ++cur; path.push_back( g ); }
        for( int j = head[u]; ~j; j = es[j].next ) {
            tmp.u = es[j].to;
            tmp.g = g + es[j].cost;
            tmp.f = tmp.g + h[tmp.u];
            que.push( tmp );
        }
    }
    return ;
}

int main() {
    int a, b, c;
    k = 2;
    while( ~scanf( "%d%d", &n, &m ) ) {
        memset( head, -1, sizeof( head ) );
        cnt = 0;
        for( int i = 1; i <= m; ++i ) {
            scanf( "%d%d%d", &a, &b, &c );
            add( a, b, c );
            add( b, a, c );
        }
        spfa();
        astar();
```

```
        printf( "%d\n", path[k - 1] );
    }
    return 0;
}
```

Bellman-Ford(检查负环)

```
#include <iostream>
#include <cstdio>
using namespace std;
#define N 110
#define INF 0x3FFFFFFF

struct edge {
    int from, to, cost;
}es[N];
int dis[N];
int V, E, s; // s: start point
bool bellman( int s ) {
    for( int i = 0; i < V; ++i ) dis[i] = INF;
    bool flag;
    for( int i = 0; i < V - 1; ++i ) { // i < V;
        flag = false;
        for( int j = 0; j < E; ++j ) {
            edge tmp = es[j];
            if( dis[tmp.to] > dis[tmp.from] + tmp.cost ) {
                dis[tmp.to] = dis[tmp.from] + tmp.cost;
                flag = true;
            }
        }
    }
    for( int i = 0; i < E; ++i ) {
        edge &tmp = es[i];
        if( dis[tmp.to] > dis[tmp.from] + tmp.cost )
            return true;
    }
    return false;
}
int main() {
    scanf( "%d%d%d", &V, &E, &s );
    for( int i = 0; i < E; ++i )
        scanf( "%d%d%d", &es[i].from, &es[i].to, &es[i].cost );
    bellman( s );
    for( int i = 0; i < V; ++i )
        printf( "%d ", dis[i] );
```

```
    printf( "\n" );
    return 0;
}
```

Bellman-Ford

```
#include <iostream>
#include <cstdio>
using namespace std;
#define N 110
#define INF 0x7FFFFFFF

struct edge {
    int from, to, cost;
}es[N];

int dis[N];
int v, e, s; // s: start point

void bellman( int s ) {
    for( int i = 0; i < v; ++i ) dis[i] = INF;
    dis[s] = 0;
    while( true ) {
        bool flag = false;
        for( int i = 0; i < e; ++i ) {
            edge tmp = es[i];
            if( dis[tmp].from != INF && dis[tmp.to] > dis[tmp.from] + tmp.cost ) {
                dis[tmp.to] = dis[tmp.from] + tmp.cost;
                flag = true;
            }
        }
        if( !flag ) break;
    }
    return ;
}

int main() {
    scanf( "%d%d%d", &v, &e, &s );
    for( int i = 0; i < e; ++i )
        scanf( "%d%d%d", &es[i].from, &es[i].to, &es[i].cost );
    bellman( s );
    for( int i = 0; i < v; ++i )
        printf( "%d ", dis[i] );
    printf( "\n" );
    return 0;
```

```
}
```

Dijkstra

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
#define N 110
#define INF 0x7FFFFFFF
#define PII pair<int, int>

struct edge {
    int to, cost;
    edge( int t, int c ) { to = t; cost = c; }
};

vector<edge> G[N];
int dis[N];
int V;

void dijikstra( int s ) {
    priority_queue<PII, vector<PII>, greater<PII> > pq;
    for( int i = 0; i < V; ++i ) dis[i] = INF;
    dis[s] = 0;
    pq.push( PII( dis[s], s ) );
    while( !pq.empty() ) {
        PII p = pq.top(); pq.pop();
        int v = p.second;
        if( dis[v] < p.first ) continue;
        for( int i = 0; i < G[v].size(); ++i ) {
            edge &e = G[v][i];
            if( dis[e.to] > dis[v] + e.cost ) {
                dis[e.to] = dis[v] + e.cost;
                pq.push( PII( dis[e.to], e.to ) );
            }
        }
    }
    return ;
}

int main() {
    int m, n;
    while( ~scanf( "%d%d", &m, &n ) && ( m || n ) ) {
```

```
        for( int i = 0; i < m; ++i ) {
            int u, v, w;
            scanf( "%d%d%d", &u, &v, &w );
            G[u - 1].push_back( edge( v - 1, w ) );
            G[v - 1].push_back( edge( u - 1, w ) );
        }
        dijikstra( 0 );
        cout << dis[n - 1] << endl;
    }
    return 0;
}
```

Dijikstra(pq)

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
#define INF 0x7FFFFFFF
#define MAX_V 1000
#define MAX_E 1000000 + 10

struct edge { int to, cost; };
typedef pair<int, int> PII;

vector<edge> G[MAX_V];
int d[MAX_V];
int V;

void dijskra( int s ) {
    priority_queue<PII, vector<PII>, greater<PII> > pq;
    for( int i = 0; i < V; ++i ) d[i] = INF;
    d[s] = 0;
    pq.push( PII( 0, s ) );
    while( !pq.empty() ) {
        PII p = pq.top(); pq.pop();
        int v = p.second;
        if( d[v] < p.first ) continue;
        for( int i = 0; i < G[v].size(); ++i ) {
            edge e = G[v][i];
            if( d[e.to] > d[v] + e.cost ) {
                d[e.to] = d[v] + e.cost;
                pq.push( PII( d[e.to], e.to ) );
            }
```

```
        }
    }
    return ;
}

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

Dijikstra(次短路)

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
#define N 110
#define PII pair<int, int>
#define INF 0x7FFFFFFF

struct edge {
    int to, cost;
    edge( int t, int c ) { to = t; cost = c; }
};

vector<edge> G[N];
int V;
int dis[N], dis2[N];

void dijikstra( int s ) {
    priority_queue<PII, vector<PII>, greater<PII> > pq;
    for( int i = 0; i < V; ++i ) dis[i] = dis2[i] = INF;
    dis[s] = 0;
    pq.push( PII( dis[s], s ) );
    while( !pq.empty() ) {
        PII p = pq.top();
        pq.pop();
        int v = p.second, d = p.first;
        if( dis2[v] < d ) continue;
        for( int i = 0; i < G[v].size(); ++i ) {
            edge &e = G[v][i];
            int d2 = d + e.cost;
            if( d2 < dis[e.to] ) {
                swap( d2, dis[e.to] );
                pq.push( PII( dis[e.to], e.to ) );
```

```
            }
            if( d2 < dis2[e.to] && d2 > dis[e.to] ) {
                dis[e.to] = d2;
                pq.push( PII( dis2[e.to], e.to ) );
            }
        }
    }
    return ;
}
int main() {
    int m, n;
    while( ~scanf( "%d%d" ,&m, &n ) && ( m || n ) ) {
        for( int i = 0; i < n; ++i ) G[i].clear();
        V = n;
        int u, v, w;
        for( int i = 0; i < m; ++i ) {
            scanf( "%d%d%d", &u, &v, &w );
            G[u - 1].push_back( edge( v - 1, w ) );
            G[v - 1].push_back( edge( u - 1, w ) );
        }
        dijikstra( 0 );
        cout << dis[n - 1] << endl;
    }
    return 0;
}
```

Dijikstra(记录路径)

```
#include <iostream>
#include <cstdio>
using namespace std;
#define N 100010

int d[N], vis[N], pre[N];
int cost[N][N];
int V;

void dijkstra( int s ) {
    for( i = 1; i <= V; ++i ) {
        d[i] = INF;
        vis[i] = 0;
        pre[i] = -1;
    }
    d[s] = 0;
    while( true ) {
```

```
        int v = -1;
        for( int u = 1; u <= V; ++u ) {
            if( !vis[u] && ( v == -1 || d[u] < d[v] ) ) v = u;
        }

        if( v == -1 ) break;
        vis[v] = 1;

        for( int u = 1; u <= V; ++u ) {
            if( d[u] > d[v] + cost[v][u] ) {
                d[u] = d[v] + cost[v][u];
                pre[u] = v;
            }
        }
    }
    return ;
}
int main() {
    return 0;
}
```

Floyd

```
#include <iostream>
#include <cstdio>
using namespace std;
#define N 1010

int d[N][N];
int V;

int main() {
    int a, b, v;
    scanf( "%d", &V );
    for( int i = 0; i < V; ++i ) {
        scanf( "%d%d%d", &a, &b, &v );
        d[a][b] = d[b][a] = v;
    }
    for( int i = 0; i < V; ++i ) {
        for( int j = 0; j < V; ++j )
            for( int k = 0; k < V; ++k )
                d[i][j] = min( d[i][j], d[i][k] + d[k][j] );
    }
    return 0;
}
```

Spfa

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <vector>
#include <queue>
using namespace std;
#define N 110
#define INF 0x7FFFFFFF
#define PII pair<int, int>

vector<PII> G[N]; // first "cost", second "to"
int dis[N], vis[N];
int v, e, s;

void spfa( int s ) {
    for( int i = 0; i < v; ++i ) dis[i] = INF;
    memset( vis, 0, sizeof( vis ) );
    queue<int> q;
    q.push( s );
    vis[s] = 1; dis[s] = 0;
    while( !q.empty() ) {
        int tmp = q.front();
        q.pop();
        for( int i = 0; i < G[tmp].size(); ++i ) {
            if( dis[G[tmp][i].second] > dis[tmp] + G[tmp][i].first ) {
                dis[G[tmp][i].second] = dis[tmp] + G[tmp][i].first;
                if( !vis[G[tmp][i].second] ) {
                    vis[G[tmp][i].second] = 1;
                    q.push( G[tmp][i].second );
                }
            }
        }
        vis[tmp] = 0;
    }
    return ;
}

int main() {
    int t;
    scanf( "%d%d%d", &v, &e, &s );
    for( int i = 0; i < e; ++i )
        scanf( "%d%d%d", &t, &G[t][i].second, &G[t][i].first );
```

```
    spfa( s );
    for( int i = 0; i < v; ++i )
        printf( "%d ", dis[i] );
    printf( "\n" );
    return 0;
}
```

Spfa(pq-slf)

```
#include <iostream>
#include <cstdio>
#include <queue>
using namespace std;
#define INF 0x7FFFFFFF
#define MAX_V 1000
#define MAX_E 1000000 + 10

int cost[MAX_V][MAX_V];
int d[MAX_V];
bool used[MAX_V];
int num[MAX_V];
int V;
struct cmp {
    bool operator() ( int x, int y ) {
        return d[x] > d[y];
    }
};

bool spfa_slf_pq( int s ) {
    priority_queue<int, vector<int>, cmp > pq;
    for( int i = 0; i < V; ++i ) { d[i] = INF; used[i] = false; num[i] = 0; }
    d[s] = 0; used[s] = true; ++num[s];
    pq.push( s );
    while( !pq.empty() ) {
        int u = pq.top(); pq.pop();
        for( int i = 0; i < V; ++i ) {
            if( d[u] + cost[u][i] < d[i] ) {
                d[i] = d[u] + cost[u][i];
                if( !used[i] ) {
                    ++num[i];
                    if( num[i] > V ) return false;
                    pq.push( i );
                    used[i] = true;
                }
            }
```

```
        }
        used[s] = false;
    }
    return true;
}

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

网络流及相关

Ford-Fulkerson

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <vector>
using namespace std;
#define MAXV 10010
#define INF 0x7FFFFFFF

struct edge { int to, cap, rev; };

vector<edge> G[MAXV];
bool vis[MAXV];

void add_edge( int f, int t, int c ) {
    G[f].push_back( ( edge ){ t, c, G[t].size() } );
    G[t].push_back( ( edge ){ f, 0, G[f].size() - 1 } );
    return ;
}

int dfs( int v, int t, int f ) {
    if( v == t ) return f;
    vis[v] = true;
    for( int i = 0; i < G[v].size(); ++i ) {
        edge &e = G[v][i];
        if( !vis[e.to] && e.cap > 0 ) {
            int d = dfs( e.to, t, min( f, e.cap ) );
            if( d > 0 ) {
                e.cap -= d;
                G[e.to][e.rev] += d;
                return d;
```

```
            }
        }
    }
    return 0;
}

int max_flow( int s, int t ) {
    int flow = 0;
    while( true ) {
        memset( vis, 0, sizeof( vis ) );
        int f = dfs( s, t, INF );
        if( f == 0 ) return flow;
        flow += f;
    }
    return 0;
}

int main() {
    int n, m;
    int a, b, c;
    scanf( "%d%d", &n, &m );
    for( int i = 0; i < n; ++i ) {
        scanf( "%d%d%d", a, b, c );
        add( a, b, c );
    }
    cout << max_flow( 0, n - 1 ) << endl;
    return 0;
}
```

Dinic

```
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
typedef int MyType;
const int INF = 0x3F3F3F3F;
const int MAXN = 1000 + 10;
const int MAXE = 100000 + 10;

struct Edge { int to, next; MyType cap; };
Edge es[MAXE];
int head[MAXN], cur[MAXN], level[MAXN], que[MAXN];
int n, F, D, cnt, src, des;
```

```
void add( int u, int v, MyType c ) {
    es[cnt].to = v; es[cnt].cap = c; es[cnt].next = head[u]; head[u] = cnt++;
    es[cnt].to = u; es[cnt].cap = 0; es[cnt].next = head[v]; head[v] = cnt++;
    return ;
}

bool bfs() {
    int mf, me;
    memset( level, 0, sizeof( level ) );
    mf = me = 0;
    que[me++] = src;
    level[src] = 1;
    while( mf < me ) {
        int u = que[mf++];
        for( int i = head[u]; ~i; i = es[i].next ) {
            int v = es[i].to;
            if( level[v] == 0 && es[i].cap > 0 ) {
                level[v] = level[u] + 1;
                que[me++] = v;
            }
        }
    }
    return ( level[des] != 0 );
}

MyType dfs( int u, MyType f ) {
    if( u == des || f == 0 ) return f;
    MyType flow = 0;
    for( int &i = cur[u]; ~i; i = es[i].next ) {
        Edge &e = es[i];
        if( e.cap > 0 && level[e.to] == level[u] + 1 ) {
            MyType d = dfs( e.to, min( f, e.cap ) );
            if( d > 0 ) {
                e.cap -= d;
                es[i ^ 1].cap += d;
                flow += d;
                f -= d;
                if( f == 0 ) break;
            } else level[e.to] = -1;
        }
    }
    return flow;
}
```

```
MyType dinic() {
    MyType ret = 0;
    while( bfs() ) {
        for( int i = src; i <= des; ++i ) {
            cur[i] = head[i];
        }
        ret += dfs( src, INF );
    }
    return ret;
}

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

费用流

费用流 Spfa

```
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
typedef int MyType;
const MyType INF = 0x7F7F7F7F;
const int MAXN = 1000 + 10;
const int MAXE = 100000 + 10;

struct Edge { int to, next; MyType cap, cost; };
Edge es[MAXE];
int head[MAXN], que[MAXE], dis[MAXN], pre[MAXN];
bool vis[MAXN];
int n, m, cnt, src, des;

void add( int u, int v, MyType f, MyType c ) {
    es[cnt].to = v; es[cnt].cap = f; es[cnt].cost = c;
    es[cnt].next = head[u]; head[u] = cnt++;
    es[cnt].to = u; es[cnt].cap = 0; es[cnt].cost = -c;
    es[cnt].next = head[v]; head[v] = cnt++;
    return ;
}

bool spfa() {
    int mf, me;
```

```
    memset( vis, false, sizeof( vis ) );
    memset( dis, 0x7F, sizeof( dis ) );
    memset( pre, -1, sizeof( pre ) );
    mf = me = 0;
    que[me++] = src; dis[src] = 0; vis[src] = true;
    while( mf < me ) {
        int u = que[mf++];
        for( int i = head[u]; ~i; i = es[i].next ) {
            int v = es[i].to;
            if( es[i].cap > 0 && dis[v] > dis[u] + es[i].cost ) {
                dis[v] = dis[u] + es[i].cost;\
                pre[v] = i;
                if( !vis[v] ) {
                    vis[v] = true;
                    que[me++] = v;
                }
            }
        }
        vis[u] = false;
    }
    return dis[des] != INF;
}

MyType cflow() {
    MyType flow = INF;
    int u = des;
    while( ~pre[u] ) {
        u = pre[u];
        flow = min( flow, es[u].cap );
        u = es[u ^ 1].to;
    }
    u = des;
    while( ~pre[u] ) {
        u = pre[u];
        es[u].cap -= flow;
        es[u ^ 1].cap += flow;
        u = es[u ^ 1].to;
    }
    return flow;
}

MyType MCMF() {
    MyType mincost, maxflow;
    mincost = maxflow = 0;
```

```
    while( spfa() ) {
        MyType flow = cflow();
        maxflow += flow;
        mincost += flow * dis[des];
    }
    return mincost;
}

int main() {
    int a, b, c;
    while( ~scanf( "%d%d", &n, &m ) ) {
        memset( head, -1, sizeof( head ) );
        cnt = 0;
        src = 0; des = n + 1;
        add( src, 1, 2, 0 );
        for( int i = 0; i < m; ++i ) {
            scanf( "%d%d%d", &a, &b, &c );
            add( a, b, 1, c );
            add( b, a, 1, c );
        }
        add( n, des, 2, 0 );
        printf( "%d\n", MCMF() );
    }
    return 0;
}
```

生成树及相关

Prim

```
#include <iostream>
#include <cstdio>
using namespace std;
#define N 1010
#define INF 0x7FFFFFFF;
int V;
int dis[N][N];
int mincost[N];
bool vis[N];

int prim() {
    int res = 0;
    for( int i = 0; i < V; ++i ) {
        mincost[i] = INF;
```

```
        vis[i] = false;
    }
    mincost[0] = 0;
    while( true ) {
        int v = -1;
        for( int i = 0; i < V; ++i ) {
            if( !vis[i] && ( v != -1 || mincost[i] < mincost[v] ) )
                v = i;
        }
        if( v == -1 ) break;
        vis[v] = true;
        res += mincost[v];
        for( int i = 0; i < V; ++i ) {
            mincost[i] = min( mincost[i], dis[v][i] );
        }
    }
    return res;
}
```

Kruskal

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
#define N 100010
#define INF 0x7FFFFFFF

struct edge {
    int u, v, w;
}es[N];

int V, E;
int father[N], rankf[N];

bool cmp( const edge a, const edge b ) {
    return a.w < b.w;
}

int findf( int x ) {
    if( x != father[x] )
        father[x] = findf( father[x] );
    return father[x];
}
```

```
int kruskal() {
    int res = 0;
    sort( es, es + E, cmp );
    for( int i = 0; i < V; ++i ) { father[i] = i; rankf[i] = 0; }
    for( int i = 0; i < E; ++i ) {
        edge &e = es[i];
        int x = findf( e.u ), y = findf( e.v );
        if( x == y ) continue;
        if( rankf[x] > rankf[y] ) father[y] = x;
        else {
            if( rankf[x] == rankf[y] ) ++rankf[y];
            father[x] = y;
        }
        res += e.w;
    }
    return res;
}

int main() {
    int x, y, z;
    scanf( "%d%d", &V, &E );
    for( int i = 0; i < E; ++i ) {
        scanf( "%d%d%d", &x, &y, &z );
        es[i].u = x; es[i].v = y; es[i].w = z;
    }
    cout << kruskal() << endl;
    return 0;
}
```

Prim 次小生成树（无重边）

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
using namespace std;
typedef pair<int, int> PII;
const int INF = 0x7F7F7F7F;
const int MAXN = 1000 + 10;

PII poi[MAXN];
double dis[MAXN][MAXN], path[MAXN][MAXN], mincost[MAXN];
int ren[MAXN], pre[MAXN];
bool vis[MAXN], used[MAXN][MAXN];
int n;
```

```
double dist( const int i, const int j ) {
    double dx = poi[i].first - poi[j].first;
    double dy = poi[i].second - poi[j].second;
    return sqrt( dx * dx + dy * dy );
}

double prim() {
    double ret = 0;
    memset( used, false, sizeof( used ) );
    memset( vis, false, sizeof( vis ) );
    memset( path, 0, sizeof( path ) );
    for( int i = 0; i < n; ++i ) { mincost[i] = INF; pre[i] = 0; }
    mincost[0] = 0;
    while( true ) {
        int v = -1;
        for( int u = 0; u < n; ++u ) if( !vis[u] && ( v == -1 || mincost[u] <
mincost[v] ) ) v = u;
        if( v == -1 ) break;
        used[pre[v]][v] = used[v][pre[v]] = true;
        ret += mincost[v];
        vis[v] = true;
        for( int u = 0; u < n; ++u ) {
            if( vis[u] && v != u ) path[u][v] = path[v][u] = max( path[u][pre[v]],
mincost[v] );
            if( !vis[u] && mincost[u] > dis[u][v] ) {
                mincost[u] = dis[u][v];
                pre[u] = v;
            }
        }
    }
    return ret;
}

int main() {
    int t;
    scanf( "%d", &t );
    while( t-- ) {
        scanf( "%d", &n );
        for( int i = 0; i < n; ++i ) {
            scanf( "%d%d%d", &poi[i].first, &poi[i].second, ren + i );
            for( int j = 0; j < i; ++j ) dis[i][j] = dis[j][i] = dist( i, j );
            dis[i][i] = 0;
        }
```

```
        double tmp = prim();
        double ans = -1;
        for( int i = 0; i < n; ++i ) {
            for( int j = 0; j < n; ++j ) if( i != j ) {
                if( used[i][j] ) ans = max( ans, ( ren[i] + ren[j] ) / ( tmp -
dis[i][j] ) );
                else ans = max( ans, ( ren[i] + ren[j] ) / ( tmp - path[i][j] ) );
            }
        }
        printf( "%.2f\n", ans );
    }
    return 0;
}
```

有向图最小生成树 朱刘算法

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
using namespace std;
const double INF = 0x3F3F3F3F;
const int MAXN = 100 + 10;
const int MAXE = 100000 + 10;

struct edge{ int u, v; double cost; };
edge es[MAXE];
int ID[MAXN], vis[MAXN], pre[MAXN], x[MAXN], y[MAXN];
double IN[MAXN];
int n, m, cnt;

void add( int u, int v, double c ) {
    es[cnt].u = u; es[cnt].v = v; es[cnt].cost = c; ++cnt;
    return ;
}

double direct_MST( int root ) {
    double ans = 0;
    while( true ) {
        memset( ID, -1, sizeof( ID ) );
        memset( vis, -1, sizeof( vis ) );
        for( int i = 0; i < MAXN; ++i ) IN[i] = INF;
        for( int i = 0; i < m; ++i ) {
            int u = es[i].u;
            int v = es[i].v;
```

```
            if( es[i].cost < IN[v] && u != v ) {
                IN[v] = es[i].cost;
                pre[v] = u;
            }
        }
        for( int i = 0; i < n; ++i ) {
            if( i == root ) continue;
            if( IN[i] == INF ) return -1;
        }
        int tv = 0;
        IN[root] = 0;// pre[root] = root;
        for( int i = 0; i < n; ++i ) {
            ans += IN[i];
            int v = i;
            while( vis[v] != i && ID[v] == -1 && v != root ) {
                vis[v] = i;
                v = pre[v];
            }
            if( v != root && ID[v] == -1 ) {
                for( int u = pre[v]; u != v; u = pre[u] ) {
                    ID[u] = tv;
                }
                ID[v] = tv++;
            }
        }
        if( !tv ) break;
        for( int i = 0; i < n; ++i ) {
            if( ID[i] == -1 ) ID[i] = tv++;
        }
        for( int i = 0; i < m; ++i ) {
            int v = es[i].v;
            es[i].u = ID[es[i].u];
            es[i].v = ID[es[i].v];
            if( es[i].u != es[i].v )
                es[i].cost -= IN[v];
        }
        n = tv;
        root = ID[root];
    }
    return ans;
}

double dis( int i, int j ) {
    double dx = abs( x[i] - x[j] );
```

```
    double dy = abs( y[i] - y[j] );
    return sqrt( dx * dx + dy * dy );
}

int main() {
    int a, b;
    while( ~scanf( "%d%d", &n, &m ) ) {
        cnt = 0;
        for( int i = 0; i < n; ++i ) scanf( "%d%d", x + i, y + i );
        for( int i = 0; i < m; ++i ) {
            scanf( "%d%d", &a, &b );
            --a; --b;
            add( a, b, dis( a, b ) );
        }
        double ans = direct_MST( 0 );
        if( ans < 0 ) puts( "poor snoopy" );
        else printf( "%.2f\n", ans );
    }
    return 0;
}
```

生成树计数

```
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
typedef long long LL;
const int MAXN = 100 + 10;

LL c[MAXN][MAXN];
int n, m;

LL det( LL a[][MAXN], int n ) {
    LL ret = 1;
    for( int i = 1; i < n; ++i ) {
        for( int j = i + 1; j < n; ++j ) {
            while( a[j][i] ) {
                LL t = a[i][i] / a[j][i];
                for( int k = i; k < n; ++k ) a[i][k] = a[i][k] - a[j][k] * t;
                for( int k = i; k < n; ++k ) swap( a[i][k], a[j][k] );
                ret = -ret;
            }
        }
        if( a[i][i] == 0 ) return 0;
```

```
        ret = ret * a[i][i];
    }
    if( ret < 0 ) ret = -ret;
    return ret;
}

int main() {
    int t, a, b;
    scanf( "%d", &t );
    while( t-- ) {
        memset( c, 0, sizeof( c ) );
        scanf( "%d%d", &n, &m );
        for( int i = 0; i < m; ++i ) {
            scanf( "%d%d", &a, &b );
            --a; --b;
            c[a][b] = c[b][a] = -1;
            ++c[a][a]; ++c[b][b];
        }
        printf( "%lld\n", det( c, n ) );
    }
    return 0;
}
```

最小生成树计数

```
#include <algorithm>
#include <iostream>
#include <cstring>
#include <cstdio>
#include <vector>
using namespace std;
typedef long long LL;
const int MAXN = 1000 + 10;
const int MAXE = 100000 + 10;

struct Edge { int u, v, w; };
Edge es[MAXE];
int fa[MAXN], ka[MAXN];
LL g[MAXN][MAXN], c[MAXN][MAXN];
bool vis[MAXN];
int n, m, mod;
vector<int> vec[MAXN];

int mfind( int x, int *f ) {
    return x == f[x] ? x : f[x] = mfind( f[x], f );
```

```
}

LL det( LL a[][MAXN], int n ) {
    for( int i = 0; i < n; ++i ) {
        for( int j = 0; j < n; ++j )
            a[i][j] %= mod;
    }
    int ret = 1;
    for( int i = 1; i < n; ++i ) {
        for( int j = i + 1; j < n; ++j ) {
            while( a[j][i] ) {
                LL t = a[i][i] / a[j][i];
                for( int k = i; k < n; ++k )
                    a[i][k] = ( a[i][k] - a[j][k] * t ) % mod;
                for( int k = i; k < n; ++k )
                    swap( a[i][k], a[j][k] );
                ret = -ret;
            }
        }
        if( a[i][i] == 0 ) return 0;
        ret = ( ret * a[i][i] ) % mod;
    }
    return ( ret + mod ) % mod;
}

bool cmp( const Edge &a, const Edge &b ) {
    return a.w < b.w;
}

void gao() {
    sort( es, es + m, cmp );
    for( int i = 1; i <= n; ++i ) { fa[i] = i; vis[i] = false; }
    LL pre = -1, ans = 1;
    for( int k = 0; k <= m; ++k ) {
        if( es[k].w != pre || k == m ) {
            for( int i = 1; i <= n; ++i ) {
                if( vis[i] ) {
                    LL u = mfind( i, ka );
                    vec[u].push_back( i );
                    vis[i] = false;
                }
            }
            for( int i = 1; i <= n; ++i ) {
                if( vec[i].size() > 1 ) {
```

```
                memset( c, 0, sizeof( c ) );
                int len = vec[i].size();
                for( int j = 0; j < len; ++j ) {
                    for( int k = j + 1; k < len; ++k ) {
                        int a1 = vec[i][j], b1 = vec[i][k];
                        c[j][k] = ( c[k][j] -= g[a1][b1] );
                        c[j][j] += g[a1][b1]; c[k][k] += g[a1][b1];
                    }
                }
                LL ret = det( c, len );
                ans = ( ans * ret ) % mod;
                for( int j = 0; j < len; ++j ) fa[vec[i][j]] = i;
            }
        }
        for( int i = 1; i <= n; ++i ) {
            ka[i] = fa[i] = mfind( i, fa );
            vec[i].clear();
        }
        if( k == m ) break;
        pre = es[k].w;
    }
    int u = es[k].u, v = es[k].v;
    int a1 = mfind( u, fa ), b1 = mfind( v, fa );
    if( a1 == b1 ) continue;
    vis[a1] = vis[b1] = true;
    ka[mfind( a1, ka )] = mfind( b1, ka );
    ++g[a1][b1]; ++g[b1][a1];
}
bool flag = false;
for( int i = 2; i <= n && !flag; ++i ) {
    if( ka[i] != ka[i - 1] ) flag = true;
}
if( !m ) flag =  true;
printf( "%I64d\n", flag ? 0 : ans % mod );
return ;
}

int main() {
    while( ~scanf( "%d%d%d", &n, &m, &mod ) && n + m + mod ) {
        memset( g, 0, sizeof( g ) );
        for( int i = 1; i <= n; ++i ) vec[i].clear();
        for( int i = 0; i < m; ++i ) scanf( "%d%d%d", &es[i].u, &es[i].v, &es[i].w );
        gao();
    }
```

```
    return 0;
}
```

斯坦纳树

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <string>
#include <map>
using namespace std;
#define INF 0x3F3F3F3F

const int sta[]={ 0, 3, 12, 48, 192, 15, 51, 195, 60, 204, 240, 63, 207, 243, 252,
255 };
int dp[300][35], dis[35][35], info[10];
map<string, int> nmap;

int lowbit( int x ) {
    return ( x & ( -x ) );
}

int bit( int x ) {
    x = lowbit( x );
    int res;
    for( res = 0; x; x >>= 1, ++res );
    return res - 1;
}

int main() {
    int n, m, ans;
    int ta, tb, c;
    string s, t;
    while( ~scanf( "%d%d", &n, &m ) && ( n || m ) ) {
        memset( dis, 0x3F, sizeof( dis ) );
        memset( dp, 0x3F, sizeof( dp ) );
        nmap.clear();
        for( int i = 0; i < n; ++i ) {
            cin >> s;
            nmap[s] = i;
            dis[i][i] = 0;
        }
        for( int i = 0; i < m; ++i ) {
            cin >> s >> t >> c;
            ta = nmap[s]; tb = nmap[t];
```

```
        dis[ta][tb] = dis[tb][ta] = min( dis[ta][tb], c );
    }
    for( int k = 0; k < n; ++k ) {
        for( int i = 0; i < n; ++i )
            for( int j = 0; j < n; ++j )
                dis[i][j] = min( dis[i][j], dis[i][k] + dis[k][j] );
    }
    for( int i = 0; i < 8; ++i ) {
        cin >> s;
        info[i] = nmap[s];
        for( int j = 0; j < n; ++j ) {
            dp[1 << i][j] = dis[info[i]][j];
        }
    }
    for( int i = 0; i < 256; ++i ) {
        if( i & ( i - 1 ) == 0 ) continue;
        c = 0;
        for( int j = 0; j < n; ++j ) {
            for( int sub = i; sub; sub = ( sub - 1 ) & i ) {
                dp[i][j] = min( dp[i][j], dp[sub][j] + dp[i - sub][j] );
            }
            if( dp[i][j] < dp[i][c] ) c = j;
        }
        for( int j = 0; j < n; ++j ) {
            for( int k = 0; k < n; ++k ) {
                dp[i][k] = min( dp[i][k], dp[i][j] + dis[j][k] );
            }
        }
    }
    ans = INF;
    for( int p1 = 0; p1 < 16; ++p1 ) {
        for( int p2 = 0; p2 < 16; ++p2 ) {
            for( int p3 = 0; p3 < 16; ++p3 ) {
                for ( int p4 = 0; p4 < 16; ++p4 ) {
                    if( sta[p1] + sta[p2] + sta[p3] + sta[p4] == 255 ) {
                        for( int i = 0; i < n; ++i ) {
                            int tmp = 0;
                            if( sta[p1] ) tmp += dp[sta[p1]][info[bit( sta[p1] )]];
                            if( sta[p2] ) tmp += dp[sta[p2]][info[bit( sta[p2] )]];
                            if( sta[p3] ) tmp += dp[sta[p3]][info[bit( sta[p3] )]];
                            if( sta[p4] ) tmp += dp[sta[p4]][info[bit( sta[p4] )]];
                            ans = min( ans, tmp );
                        }
                    }
```

```
                }
            }
        }
    }
    printf( "%d\n", ans );
}
return 0;
}
```

欧拉路

Fleury

```
#include <cstdlib>
#include <cstring>
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;
/*
弗罗莱算法
*/

int stk[1005];
int top;
int N, M, ss, tt;
int mp[1005][1005];

void dfs(int x) {
    stk[top++] = x;
    for (int i = 1; i <= N; ++i) {
        if (mp[x][i]) {
            mp[x][i] = mp[i][x] = 0; // 删除此边
            dfs(i);
            break;
        }
    }
}
```

差分约束

SPFA 法

```
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
const int INF = 0x7F7F7F7F;
const int MAXN = 100 + 10;
const int MAXM = 100000 + 10;

struct Edge { int to, cost, next; };
Edge es[MAXM];
int head[MAXN], que[MAXM], dis[MAXN], ncnt[MAXN];
bool vis[MAXN];
int n, m, cnt, src;

void add( int u, int v, int w ) {
    es[cnt].to = v; es[cnt].cost = w; es[cnt].next = head[u]; head[u] = cnt++;
    return ;
}

bool spfa() {
    int mf, me;
    memset( vis, false, sizeof( vis ) );
    memset( dis, 0, sizeof( dis ) );
    memset( ncnt, 0, sizeof( ncnt ) );
    mf = me = 0;
    for( int i = 0; i <= n; ++i ) { que[me++] = i; vis[i] = true; }
    ++ncnt[0];
    while( mf != me ) {
        int u = que[mf++];
        if( mf >= MAXM ) mf -= MAXM;
        for( int i = head[u]; ~i; i = es[i].next ) {
            int v = es[i].to;
            if( dis[v] > dis[u] + es[i].cost ) {
                dis[v] = dis[u] + es[i].cost;
                if( !vis[v] ) {
                    vis[v] = true;
                    ++ncnt[v];
                    que[me++] = v;
                    if( ncnt[v] > n ) return false;
                    if( me >= MAXM ) me -= MAXM;
```

```
                }
            }
        }
        vis[u] = false;
    }
    return true;
}

int main() {
    int a, b, c;
    char ch[5];
    while( ~scanf( "%d", &n ) && n ) {
        memset( head, -1, sizeof( head ) );
        cnt = 0;
        scanf( "%d", &m );
        for( int i = 0; i < m; ++i ) {
            scanf( "%d%d%s%d", &a, &b, ch, &c );
            if( ch[0] == 'l' ) add( a - 1, a + b, c - 1 );
            else add( a + b, a - 1, -c - 1 );
        }
        if( spfa() ) puts( "lamentable kingdom" );
        else puts( "successful conspiracy" );
    }
    return 0;
}
```

全局最小割

Stoer-Wagner

```
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
const int INF = 0x7F7F7F7F;
const int MAXN = 1000 + 10;

int mat[MAXN][MAXN], v[MAXN], dis[MAXN];
bool vis[MAXN];
int n, m;

int SW() {
    int ret = INF;
    for( int i = 0; i <= n; ++i ) v[i] = i;
```

```
    while( n > 1 ) {
        int pre = 0;
        memset( vis, false, sizeof( vis ) );
        memset( dis, 0, sizeof( dis ) );
        for( int i = 1; i < n; ++i ) {
            int k = -1;
            for( int j = 1; j < n; ++j ) {
                if( !vis[v[j]] ) {
                    dis[v[j]] += mat[v[pre]][v[j]];
                    if( k == -1 || dis[v[k]] < dis[v[j]] ) k = j;
                }
            }
            vis[v[k]] = true;
            if( i == n - 1 ) {
                ret = min( ret, dis[v[k]] );
                for( int j = 0; j < n; ++j ) {
                    mat[v[pre]][v[j]] = ( mat[v[j]][v[pre]] += mat[v[j]][v[k]] );
                }
                v[k] = v[--n];
            }
            pre = k;
        }
    }
    return ret;
}

int main() {
    int a, b, c;
    while( ~scanf( "%d%d", &n, &m ) ) {
        memset( mat, 0, sizeof( mat ) );
        while( m-- ) {
            scanf( "%d%d%d", &a, &b, &c );
            mat[a][b] = ( mat[b][a] += c );
        }
        printf( "%d\n", SW() );
    }
    return 0;
}
```

二分图及其相关

性质：

定义：

1.最大独立点集：在二分图中，求最少的点集，使得任意两个点之间没有直接边连接。

2.最小路径覆盖：

①在有向无环图中，求最少的路径，使他们覆盖所有的点，且每一个点只被一条路径覆盖。

②在有向无环图中，求最少的不相交的路径，使他们覆盖所有的点。

*最小路径覆盖数通俗点说就是给一个图，求最少要走几次才能把所有节点都遍历一次且仅一次。（并不要求把每条边都走完）

3.最小可重复路径覆盖：在有向无环图中，求最少的路径，使得他们覆盖所有的点，且每个点可以被多条路径覆盖。

4.最小边覆盖：在二分图中，求最少的边，使得他们覆盖所有的点，并且每一个点只被一条边覆盖。（实在不行可以把一个点看成一条边）

5.最小点覆盖：在二分图中，求最少的点集，使得每一条边至少都有端点在这个点集中。

6.Dilworth 定理：

偏序是在集合 X 上的二元关系≤（这只是个抽象符号，不是“小于或等于”），它满足自反性、反对称性和传递性。即，对于 X 中的任意元素 a,b 和 c，有：

自反性：a≤a;

反对称性：如果 a≤b 且 b≤a，则有 a=b;

传递性：如果 a≤b 且 b≤c，则 a≤c 。

①链：一个图中的一条路径，或：一个集合使得任意两个元素都可比较。

②反链：一个图中的一个点集使得任意两个点之间都没有路径相连，或：一个集合使得任意两个元素都不可比较

运算关系：

最大独立集 = 最小边覆盖 = N - 最大匹配 （条件：在二分图中）

DAG 最小路径覆盖 = N - 最大匹配 （条件:将 DAG 转化为二分图）

最小点覆盖 = 最大匹配 （条件：在二分图中） Konig 定理

反链 = 图的最小划分链数 = 最小可重复路径覆盖 （条件：在 DAG 中） Dilworth 定理

求 Dilworth 定理中的数：Floyd 传递闭包求出 DAG 中的每一对点之间的连接情况，然后求最小不可重复路径覆盖。

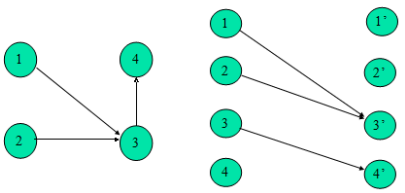
求最小路径覆盖：将每一个点拆成 Xi 和 Yi，如果有点 Ai->Bi，那么连边 XA->YB。求最大匹配。

DAG 的最小路径覆盖是指找最小数目的互相不相交的有向路径,满足 DAG 的所有顶点都被覆盖.

首先给出公式:DAG 的最小路径覆盖数=DAG 图中的节点数-相应二分图中的最大匹配数.

那么对应一个 DAG,如何构造相应的二分图?对于 DAG 中的一个顶点 p,二分图中有两个顶点 p 和 p',对应 DAG 中的一条有向边 p->q,二分图中有 p-q'的一条无向边.二分图中 p 属于 S 集合,p'属于 T 集合.

下面我们来解释上面公式为什么成立,思路参考 baihacker 神牛:



上图中,对应左边的 DAG 建立构造右边的二分图,可以找到二分图的一个最大匹配 M:1-3',3-4',那么 M 中的这两条匹配边怎样对应 DAG 中的路径的边?

使二分图中一条边对应 DAG 中的一条有向边,1-3'对应 DAG 图中的有向边 1->3,这样 DAG 中 1 就会有一个后继顶点(3 会是 1 的唯一后继,因为二分图中一个顶点至多关联一条边!),所以 1 不会成为 DAG 中一条路

径中的结尾顶点,同样,3-4'对应 DAG 中 3->4,3 也不会成为结尾顶点,那么原图中总共 4 个顶点,减去 2 个有后继的顶点,就剩下没有后继的顶点,即 DAG 路径的结尾顶点,而每个结尾顶点正好对应 DAG 中的一条路径,二分图中寻找最大匹配 M,就是找到了对应 DAG 中的非路径结尾顶点的最大数目,那么 DAG 中顶点数-|M|就是 DAG 中结尾顶点的最小数目,即 DAG 的最小路径覆盖数.

二分图匹配(Hopcroft-Karp)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <queue>
using namespace std;
#define MAXN 3000

const int INF = 1 << 28;
int g[MAXN][MAXN], Mx[MAXN], My[MAXN], Nx, Ny;
int dx[MAXN], dy[MAXN], dis;
bool vst[MAXN];

bool searchP() {
    queue<int> Q;
    dis = INF;
    memset( dx, -1, sizeof( dx ) );
    memset( dy, -1, sizeof( dy ) );
    for( int i = 0; i < Nx; ++i ) {
        if( Mx[i] == -1 ) {
            Q.push( i );
            dx[i] = 0;
        }
    }
    while( !Q.empty() ) {
        int u = Q.front();
        Q.pop();
        if( dx[u] > dis ) break;
        for( int v = 0; v < Ny; ++v ) {
            if( g[u][v] && dy[v] == -1 ) {
                dy[v] = dx[u] + 1;
                if( My[v] == -1 ) {
                    dis = dy[v];
                }
                else {
                    dx[My[v]] = dy[v] + 1;
                    Q.push( My[v] );
                }
            }
```

```
        }
    }
    return dis != INF;
}
bool DFS( int u ) {
    for( int v = 0; v < Ny; v++ ) {
        if( !vst[v] && g[u][v] && dy[v] == dx[u] + 1 ) {
            vst[v] = 1;
            if( My[v] != -1 && dy[v] == dis ) continue;
            if( My[v] == -1 || DFS( My[v] ) ) {
                My[v] = u;
                Mx[u] = v;
                return true;
            }
        }
    }
    return 0;
}
int MaxMatch() {
    int res = 0;
    memset( Mx, -1, sizeof( Mx ) );
    memset( My, -1, sizeof( My ) );
    while( searchP() ) {
        memset( vst, 0, sizeof( vst ) );
        for( int i = 0; i < Nx; i++ ) {
            if( Mx[i] == -1 && DFS( i ) )
                res++;
        }
    }
    return res;
}
int main() {
    return 0;
}
```

二分图匹配(匈牙利)

```
#include <iostream>
#include <cstdio>
using namespace std;
#define N 1000

int nmap[N][N];
int state[N], result[N];
int n, m, t, tmp;
```

```
int ans;

bool nfind( int x ) {
    int i;
    for( i = 0; i < m; ++i ) {
        if( nmap[x][i] = 1 && !state[i] ) {
            state[i] = 1;
            if( !result[i] || nfind( result[i] ) ) {
                result[i] = x;
                return true;
            }
        }
    }
    return false;
}
int main() {
    int i, j, k;
    ans = 0;
    scanf( "%d%d", &n, &m );
    for( i = 0; i < n; ++i ) {
        scanf( "%d", t );
        for( j = 0; j < t; ++j ) {
            scanf( "%d", &tmp );
            nmap[i][tmp - 1] = 1;
        }
    }
    for( i = 0; i < n; ++i ) {
        memset( state, 0, sizeof( state ) );
        if( nfind( i ) ) ++ans;
    }
    printf( "%d\n", &ans );
    return 0;
}
```

二分图最优匹配(KM)

```
#include <algorithm>
#include <iostream>
#include <cstring>
#include <cstdio>
#include <string>
using namespace std;
#define INF 99999999

int nmap[305][305];
```

```
int lx[305],ly[305];
bool x[305],y[305];
int link[305];
int n;

bool dfs( int u ) {
    int i;
    x[u] = true;
    for( i = 1; i <= n; ++i ) {
        if( lx[u] + ly[i] == nmap[u][i] && !y[i] ) {
            y[i] = true;
            if( link[i] == -1 || dfs( link[i] ) ) {
                link[i] = u;
                return true;
            }
        }
    }
    return false;
}

void KM() {
   int i, j, k;
   memset( x, 0, sizeof( x ) );
   memset( y, 0, sizeof( y ) );
   memset( link, -1, sizeof( link ) );
   for( i = 1; i <= n; ++i )
      lx[i] = INF;
   memset( ly, 0, sizeof( ly ) );
   for( k = 1; k <= n; ++k ) {
      while( true ) {
          memset( x, 0, sizeof( x ) );
          memset( y, 0, sizeof( y ) );
          if( dfs( k ) ) break;
          int d = INF;
          for( i = 1; i <= n; ++i ) {
              if( x[i] )
                  for( j = 1; j <= n; ++j )
                      if( !y[j] && lx[i] + ly[j] - nmap[i][j] < d )
                          d = lx[i] + ly[j] - nmap[i][j];
          }
          for( i = 1; i <= n; ++i ) {
              if( x[i] )
                  lx[i] = lx[i] - d;
          }
```

```
            for( i = 1; i <= n; ++i ) {
                if( y[i] )
                    ly[i] = ly[i] + d;
            }
        }
    }
    return ;
}

int main() {
    int i, j, k;
    while( ~scanf( "%d", &n ) ) {
        for( i = 1; i <= n; ++i )
            for( j = 1; j <= n; ++j )
                scanf( "%d", &nmap[i][j] );
       KM();
        int ans = 0;
        for( i = 1; i <= n; ++i )
            ans = ans + nmap[link[i]][i];
        printf( "%d\n", ans );
    }
    return 0;
}
```

数据结构

树状数组

```
#include <iostream>
using namespace std;
#define pain<int, int> PII
#define N 1000010

PII node[N];
int c[N];
int n;

int lowbit( int x )
{
    return x & ( -x );
}

void add( int i, int val )
{
```

```
    while( i <= n )
    {
        c[i] += val;
        i += lowbit( i );
    }
    return ;
}

int sum( int i )
{
    int ans = 0;
    while( i <= n )
    {
        ans += 1;
        i += lowbit( i );
    }
    return ans;
}

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

线段树

线段树单点更新单点值查询

```
#include <iostream>
#include <cstdio>
using namespace std;
#define MAXN 1000000
#define lson rt << 1, l, m
#define rson rt << 1 | 1, m + 1, r

struct Node {
    int l, r;
    int num;
}node[4 * MAXN];

void build( int rt, int l, int r ) {
    node[rt].l = l;
    node[rt].r = r;
    node[rt].num = 0;
```

```
    if( l == r ) return ;
    int m = ( node[rt].l + node[rt].r ) >> 1;
    build( lson );
    build( rson );
    return ;
}

void update( int rt, int pos, int value ) {
    if( node[rt].l == pos && node[rt].r == pos ) {
        node[rt].num = value;
        return ;
    }
    int m = ( node[rt].l + node[rt].r ) >> 1;
    if( pos <= m )
        update( rt << 1, pos, value );
    else
        update( rt << 1 | 1, pos, value );
    node[rt].num = node[rt << 1].num + node[rt << 1 | 1].num;
    return ;
}

int query( int rt, int pos ) {
    if( node[rt].l == pos && node[rt].r == pos )
        return node[rt].num;
    int m = ( node[rt].l + node[rt].r ) >> 1;
    if( pos <= m )
        return query( rt << 1, pos );
    else
        return query( rt << 1 | 1, pos );
}

int main() {
    int n = 5;
    build( 1, 1, n );
    update( 1, 4, 4 );
    update( 1, 2, 3 );
    cout << query( 1, 3 ) << endl;
    cout << query( 1, 2 ) << endl;
    return 0;
}
```

线段树单点更新区间求和查询

```
#include <iostream>
#include <cstdio>
```

```
using namespace std;
#define MAXN 1000000
#define lson rt << 1, l, m
#define rson rt << 1 | 1, m + 1, r

struct Node {
    int l, r;
    int num;
}node[4 * MAXN];

void build( int rt, int l, int r ) {
    node[rt].l = l;
    node[rt].r = r;
    node[rt].num = 0;
    if( l == r ) return ;
    int m = ( node[rt].l + node[rt].r ) >> 1;
    build( lson );
    build( rson );
    return ;
}

void update( int rt, int pos, int value ) {
    if( node[rt].l == pos && node[rt].r == pos ) {
        node[rt].num += value;
        return ;
    }
    int m = ( node[rt].l + node[rt].r ) >> 1;
    if( pos <= m )
        update( rt << 1, pos, value );
    else
        update( rt << 1 | 1, pos, value );
    node[rt].num = node[rt << 1].num + node[rt << 1 | 1].num;
    return ;
}

int query( int rt, int l, int r ) {
    if( node[rt].l == l && node[rt].r == r ) {
        return node[rt].num;
    }
    int sum = 0, m;
    m = ( node[rt].l + node[rt].r ) >> 1;
    if( r <= m )
        sum += query( rt << 1, l, r );
    else if( l > m )
```

```
        sum += query( rt << 1 | 1, l, r );
    else {
        sum += query( lson );
        sum += query( rson );
    }
    return sum;
}

int main() {
    int n = 5;
    build( 1, 1, n );
    update( 1, 3, 4 );
    update( 1, 2, 5 );
    cout << query( 1, 1, 2 ) << endl;
    return 0;
}
```

线段树区间更新单点值查询

```
#include <iostream>
#include <cstdio>
using namespace std;
#define MAXN 1000000
#define lson rt << 1, l, m
#define rson rt << 1 | 1, m + 1, r

struct Node {
    int l, r;
    int num;
}node[4 * MAXN];

void build( int rt, int l, int r ) {
    node[rt].l = l;
    node[rt].r = r;
    node[rt].num = 0;
    if( l == r ) return ;
    int m = ( node[rt].l + node[rt].r ) >> 1;
    build( lson );
    build( rson );
    return ;
}

void update( int rt, int l, int r, int value ) {
    if( node[rt].l == l && node[rt].r == r && l == r ) {
        node[rt].num = value;
```

```
        return ;
    }
    int m = ( node[rt].l + node[rt].r ) >> 1;
    if( r <= m )
        update( rt << 1, l, r, value );
    else if( l > m )
        update( rt << 1 | 1, l, r, value );
    else {
        update( lson, value );
        update( rson, value );
    }
    node[rt].num = max( node[rt << 1].num, node[rt << 1 | 1].num );
    return ;
}

int query( int rt, int pos ) {
    if( node[rt].l == pos && node[rt].r == pos )
        return node[rt].num;
    int m = ( node[rt].l + node[rt].r ) >> 1;
    if( pos <= m )
        return query( rt << 1, pos );
    else
        return query( rt << 1 | 1, pos );
}

int main() {
    int n = 5;
    build( 1, 1, n );
    update( 1, 2, 3, 3 );
    cout << query( 1, 2 ) << endl;
    return 0;
}
```

线段树区间更新区间求和查询（lazy）

```
#include <iostream>
#include <cstdio>
using namespace std;
#define MAXN 1000000
#define lson rt << 1, l, m
#define rson rt << 1 | 1, m + 1, r

struct Node {
    int l, r;
    int num, lazy;
```

```
}node[4 * MAXN];

void build( int rt, int l, int r ) {
    node[rt].l = l;
    node[rt].r = r;
    node[rt].num = 0;
    node[rt].lazy = 0;
    if( l == r ) return ;
    int m = ( node[rt].l + node[rt].r ) >> 1;
    build( lson );
    build( rson );
    return ;
}

void update( int rt, int l, int r, int value );
void dowork( int rt ) {
    if( node[rt].lazy ) {
        int son = rt << 1;
        update( son, node[son].l, node[son].r, node[son].lazy );
        son = rt << 1 | 1;
        update( son, node[son].l, node[son].r, node[son].lazy );
        node[rt].lazy = 0;
    }
    return ;
}

void update( int rt, int l, int r, int value ) {
    if( node[rt].l == l && node[rt].r == r ) {
        node[rt].num = value * ( r - l + 1 );
        if( l != r )
            node[rt].lazy += value;
        return ;
    }
    dowork( rt );
    int m = ( node[rt].l + node[rt].r ) >> 1;
    if( r <= m )
        update( rt << 1, l, r, value );
    else if( l > m )
        update( rt << 1 | 1, l, r, value );
    else {
        update( lson, value );
        update( rson, value );
    }
    node[rt].num = node[rt << 1].num + node[rt << 1 | 1].num;
```

```
    return ;
}

int query( int rt, int l, int r ) {
    if( node[rt].l == l && node[rt].r == r )
        return node[rt].num;
    dowork( rt );
    int m = ( node[rt].l + node[rt].r ) >> 1, sum = 0;
    if( r <= m )
        sum += query( rt << 1, l, r );
    else if( l > m )
        sum += query( rt << 1 | 1, l, r );
    else {
        sum += query( lson );
        sum += query( rson );
    }
    return sum;
}

int main() {
    int n = 5;
    build( 1, 1, n );
    cout << query( 1, 1, 5 ) << endl;
    update( 1, 3, 5, 4 );
    cout << query( 1, 1, 5 ) << endl;
    update( 1, 2, 3, 3 );
    cout << query( 1, 1, 5 ) << endl;
    return 0;
}
```

线段树区间更新区间求和查询（非 lazy）

```
#include <iostream>
#include <cstdio>
using namespace std;
#define MAXN 1000000
#define lson rt << 1, l, m
#define rson rt << 1 | 1, m + 1, r

struct Node {
    int l, r;
    int num;
}node[4 * MAXN];

void build( int rt, int l, int r ) {
```

```
    node[rt].l = l;
    node[rt].r = r;
    node[rt].num = 0;
    if( l == r ) return ;
    int m = ( node[rt].l + node[rt].r ) >> 1;
    build( lson );
    build( rson );
    return ;
}

void update( int rt, int l, int r, int value ) {
    if( node[rt].l == l && node[rt].r == r && l == r ) {
        node[rt].num = value;
        return ;
    }
    int m = ( node[rt].l + node[rt].r ) >> 1;
    if( r <= m )
        update( rt << 1, l, r, value );
    else if( l > m )
        update( rt << 1 | 1, l, r, value );
    else {
        update( lson, value );
        update( rson, value );
    }
    node[rt].num = node[rt << 1].num + node[rt << 1 | 1].num;
    return ;
}

int query( int rt, int l, int r ) {
    if( node[rt].l == l && node[rt].r == r )
        return node[rt].num;
    int m = ( node[rt].l + node[rt].r ) >> 1, sum = 0;
    if( r <= m )
        sum += query( rt << 1, l, r );
    else if( l > m )
        sum += query( rt << 1 | 1, l, r );
    else {
        sum += query( lson );
        sum += query( rson );
    }
    return sum;
}

int main() {
```

```
    int n = 5;
    build( 1, 1, n );
    update( 1, 3, 4, 5 );
    update( 1, 2, 3, 3 );
    cout << query( 1, 3, 4 ) << endl;
    return 0;
}
```

树链剖分模板

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int MAXN = 10000 + 10;
const int MAXM = 1000000 + 10;
#define MID(x, y) (((x) + (y)) >> 1)

int fa[MAXN], top[MAXN], w[MAXN], son[MAXN], dep[MAXN], sz[MAXN], r[MAXN];
int a[MAXN], b[MAXN];
LL c[MAXN];
int ind[MAXN];
int t[MAXM], nt[MAXM];
int cnt1, cnt2, cnt3;
int n, m;

struct node {
    int l, r;
    int a, b;
    LL sum;
}f[MAXM];
int rt;

void dfs1( int x, int d ) {
    dep[x] = d; son[x] = 0; sz[x] = 1;
    for( int k = ind[x]; ~k; k = nt[k] ) {
        if( t[k] != fa[x] ) {
            fa[t[k]] = x;
            dfs1( t[k], d + 1 );
            sz[x] += sz[t[k]];
            if( sz[t[k]] > sz[son[x]] ) son[x] = t[k];
        }
    }
    return ;
}
```

```
void dfs2( int x, int tt ) {
    w[x] = ++cnt2; top[x] = tt;
    if( son[x] ) dfs2( son[x], tt );
    for( int k = ind[x]; ~k; k = nt[k] ) {
        if( t[k] != fa[x] && t[k] != son[x] )
            dfs2( t[k], t[k] );
    }
    return ;
}

void add( int a, int b ) {
    t[cnt1] = b; nt[cnt1] = ind[a]; ind[a] = cnt1++;
    return ;
}

void update( int x ) {
    f[x].sum = f[f[x].l].sum + f[f[x].r].sum;
}

int bt( int a, int b ) {
    int x = cnt3++;
    f[x].a = a; f[x].b = b;
    if( a < b ) {
        int mid = MID( a, b );
        f[x].l = bt( a, mid );
        f[x].r = bt( mid + 1, b );
        f[x].sum = 0;
    } else f[x].sum = 0;
    return x;
}

// Query On ST, Do not Call Directly
LL query( int x, int a, int b ) {
    if( a <= f[x].a && f[x].b <= b ) return f[x].sum;
    int mid = MID( f[x].a, f[x].b );
    LL ans = 0;
    if( a <= mid ) ans += query( f[x].l, a, b );
    if( b > mid ) ans += query( f[x].r, a, b );
    return ans;
}

//Modify Point
void update( int x, int p, int cc ) {
    if( f[x].a == f[x].b ) { f[x].sum = cc; return; }
```

```
    int mid = MID( f[x].a, f[x].b );
    if( p <= mid ) update( f[x].l, p, cc );
    else update( f[x].r, p, cc );
    update( x );
    return ;
}

//Query Segment
LL query( int x, int y ) {
    int fx = top[x], fy = top[y];
    LL sum = 0;
    while( fx != fy ) {
        if( dep[fx] < dep[fy] ) {
            swap( x, y );
            swap( fx, fy );
        }
        sum += query( rt, w[fx], w[x] );
        x = fa[top[x]];
        fx = top[x];
    }
    if( dep[x] > dep[y] ) swap( x, y );
    if( x == y ) return sum;
    return sum + query( rt, w[son[x]], w[y] );
}

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

字符串

KMP

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

int main() {
    int i, j;
    int p[100010], len1, len2;
    char A[100010], B[100010];
    scanf( "%s%s", A, B );
    len1 = strlen( A );
```

```
    len2 = strlen( B );
    p[0] = -1;
    for( i = 1, j = -1; i < len2; ++i ) {
        while( ( j >= 0 ) && ( B[j + 1] != B[i] ) ) j = p[j];
        if( B[j + 1] == B[i] ) ++j;
        p[i] = j;
    }
    for( i = 0, j = -1; i < len1; ++i ) {
        while( ( j >= 0 ) && ( B[j + 1] != A[i] ) ) j = p[j];
        if( B[j + 1] == A[i] ) ++j;
        if( j == len2 - 1 ) {
            cout << i + 1 - len2 << endl;
//            j = p[j]; //multiple matching
            break;
        }
    }
    return 0;
}
```

AC 自动机（纯数组版）

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <queue>
using namespace std;
#define N 250001
#define root 0

const int kind = 26;
struct node {
    int next[kind], fail;
    int ncount;
    void init() {
        memset( next, -1, sizeof( next ) );
        fail = ncount = 0;
    }
}tree[N];
int n_size;
char str[4 * N];

void n_insert( char *s ) {
    int now, index;
    for( now = root; *s; s++ ) {
        index = *s - 'a';
```

```
        if( tree[now].next[index] == -1 ) {
            tree[++n_size].init();
            tree[now].next[index] = n_size;
        }
        now = tree[now].next[index];
    }
    tree[now].ncount++;
    return ;
}

void build_ac_auto() {
    queue<int> q;
    int p;
    q.push( root );
    tree[root].fail = -1;
    while( !q.empty() ) {
        int k = q.front();
        q.pop();
        for( int i = 0; i < kind; i++ ) {
            if( tree[k].next[i] != -1 ) {
                if( k == root ) tree[tree[k].next[i]].fail = root;
                else {
                    p = tree[k].fail;
                    while( p != -1 ) {
                        if( tree[p].next[i] != -1 ) {
                            tree[tree[k].next[i]].fail = tree[p].next[i];
                            break;
                        }
                        p = tree[p].fail;
                    }
                    if( p == -1 ) tree[tree[k].next[i]].fail = root;
                }
                q.push( tree[k].next[i] );
            }
            else {
                if( k == root ) tree[k].next[i] = root;
                else tree[k].next[i] = tree[tree[k].fail].next[i];
            }
        }
    }
    return ;
}

int query( char *s ) {
```

```
    int now, cnt = 0, index;
    for( now = root; *s; s++ ) {
        index = *s - 'a';
//        while( tree[now].next[index] == -1 && now ) now = tree[now].fail;
//        now = tree[now].next[index];
//        now = ( now == -1 ) ? root : now;
        int t = now;
        while( tree[t].ncount != -1 ) {
            cnt += tree[t].ncount;
            tree[t].ncount = -1;
            t = tree[t].fail;
        }
    }
    return cnt;
}

int main() {
    freopen( "out.txt", "r", stdin );
    int T, n;
    scanf( "%d", &T );
    while( T-- ) {
        char words[51];
        n_size = 0;
        tree[0].init();
        scanf( "%d", &n );
        for( int i = 0; i < n; i++ ) {
            scanf( "%s", words );
            n_insert( words );
        }
        build_ac_auto();
        scanf( "%s", str );
        printf( "%d\n", query( str ) );
    }
    return 0;
}
```

数论

Ploya 定理

```
#include <iostream>
#include <cstdio>
using namespace std;
#define LL long long
```

```
LL c, s;

LL gcd( LL a, LL b ) {
    return b == 0 ? a : gcd( b, a % b );
}

LL pow( LL a, LL b ) {
    LL ans = 1;
    while( b ) {
        if( b & 1 ) ans *= a;
        a *= a;
        b >>= 1;
    }
    return ans;
}

LL polya() {
    LL i, j;
    LL sum = 0;
    for( i = 1; i <= s; i++ )
        sum += pow( c, gcd( s, i ) );
    if( s & 1 )
        sum += s * pow( c, s / 2 + 1 );
    else
        sum += ( ( s / 2 ) * pow( c, s / 2 ) ) + ( ( s / 2 ) * pow( c, s / 2 + 1 ) );
    sum /= ( 2 * s );
    return sum;
}

int main() {
    while( ~scanf( "%I64d%I64d", &c, &s ) && ( c || s ) )
        printf( "%I64d\n", polya() );
    return 0;
}
```

组合数

```
#include <iostream>
#include <stdio.h>
#include <math.h>
using namespace std;

const int M = 1007;
const int MAXN = 1000;
```

```
long long C[MAXN+1][MAXN+1];
void Initial()
{
    int i,j;
    for(i=0; i<=MAXN; ++i)
    {
        C[0][i] = 0;
        C[i][0] = 1;
    }
    for(i=1; i<=MAXN; ++i)
    {
        for(j=1; j<=MAXN; ++j)
            C[i][j] = (C[i-1][j] + C[i-1][j-1]);
    }
}
long long Combination(int n, int m)
{
    return C[n][m];
}
int main()
{
    int T,i,m,n;
    Initial();
    while( ~scanf("%d%d",&n,&m) ){
        printf("C(%d£¬%d)=%I64d\n",n,m,Combination(n,m));
    }
    return 0;
}
```

快速线性筛素数

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
#define N 1000010
int prime[N];
bool vis[N];
int num;
void _prime() {
    int i, j;
    memset( vis, true, sizeof( vis ) );
    vis[0] = vis[1] = 0;
    for( i = 2, num = 0; i < N; ++i ) {
        if( vis[i] )
```

```
            prime[num++] = i;
        for( j = 0; j < num && i * prime[j] < N; ++j ) {
            vis[i * prime[j]] = 0;
            if( !( i % prime[j] ) )
               break;
        }
    }
    return ;
}
int main() {
    _prime();
    return 0;
}
```

动态规划

LIS(最长上升子序列)

```
#include<cstdio>
#include<algorithm>
using namespace std;

int n, a[20010];
int c[20010];
int len = 0;

int find( int x )
{
    int l = 1, r = len, mid;
    while( l <= r ) {
        mid = ( l + r ) >> 1;
        if( x > c[mid] ) l = mid + 1; //记忆方法:求上升序列，就表示 x 更大，那么就是大于
        else r = mid - 1;
    }
    return l;
}

int main() {
    scanf( "%d", &n );
    for( int i = 1; i <= n; i++ )
        scanf( "%d", &a[i] );
    for( int i = 1; i <= n; i++ ) {
        int k = find( a[i] );
        c[k] = a[i];
```

```
        len = max( len, k );
    }
    printf( "%d", len );
    return 0;
}
```

树上任意点到树上最远距离 O(n)

```
#include <bits/stdc++.h>
using namespace std;
#define INF 0x3f3f3f3f
const int MAXN = 50000 + 10;
const int MAXM = 100000 + 10;

struct Edge { int v, w, next; };
Edge es[MAXM];
int head[MAXN], mmax[MAXN][2], poi[MAXN][2], dis[MAXN], disf[MAXN][2], mlog[MAXN];
int d1[MAXN][17], d2[MAXN][17];
int n, m, cnt;

void add( int u, int v, int w ) {
    es[cnt].v = v; es[cnt].w = w; es[cnt].next = head[u]; head[u] = cnt++;
    return ;
}

void dfs( int u, int pre ) {
    bool flag = false;
    for( int i = head[u]; ~i; i = es[i].next ) {
        int v = es[i].v, w = es[i].w;
        if( v != pre ) {
            flag = true;
            dfs( v, u );
            if( mmax[u][0] < mmax[v][0] + w ) {
                mmax[u][1] = mmax[u][0];
                mmax[u][0] = mmax[v][0] + w;
                poi[u][1] = poi[u][0];
                poi[u][0] = v;
            } else if( mmax[u][1] < mmax[v][0] + w ) {
                mmax[u][1] = mmax[v][0] + w;
                poi[u][1] = v;
            }
        }
    }
    if( !flag ) {
        dis[u] = 0;
```

```
        mmax[u][0] = mmax[u][1] = 0;
        poi[u][0] = poi[u][1] = 0;
    }
    return ;
}

void dfs2( int u, int pre ) {
    for( int i = head[u]; ~i; i = es[i].next ) {
        int v = es[i].v, w = es[i].w;
        if( v != pre ) {
            if( v == poi[u][0] ) {
                dis[v] = max( mmax[v][0], w + disf[u][0] );
                disf[v][0] = max( disf[u][0] + w, mmax[v][1] );
                disf[v][1] = max( disf[u][0] + w, mmax[v][0] );
            } else {
                dis[v] = max( mmax[v][0], w + disf[u][1] );
                disf[v][0] = max( disf[u][1] + w, mmax[v][1] );
                disf[v][1] = max( disf[u][1] + w, mmax[v][0] );
            }
            dfs2( v, u );
        }
    }
    return ;
}

int main() {
    int a, b, c, q;
    while( ~scanf( "%d%d", &n, &m ) && n + m ) {
        memset( head, -1, sizeof( head ) );
        memset( mmax, 0, sizeof( mmax ) );
        memset( dis, 0, sizeof( dis ) );
        cnt = 0;
        add( 0, 1, 0 );
        for( int i = 0; i < n - 1; ++i ) {
            scanf( "%d%d%d", &a, &b, &c );
            add( a, b, c ); add( b, a, c );
        }
        dfs( 0, 0 );
        disf[0][0] = disf[0][1] = 0;
        poi[0][0] = 1; poi[0][1] = 0;
        dfs2( 0, 0 );
    }
    return 0;
}
```

计算几何

点与矩形最小距离

```
#include <bits/stdc++.h>
using namespace std;
const double eps = 1e-8;
const double PI = acos( -1.0 );

int sig( double x ) {
    if( fabs( x ) < eps ) return 0;
    return x > 0 ? 1 : -1;
}

struct Point {
    double x, y;
    Point() {}
    Point( const double xx, const double yy ) { x = xx; y = yy; }
    Point operator + ( const Point &tp ) const { return Point( x + tp.x, y + tp.y ); }
    Point operator - ( const Point &tp ) const { return Point( x - tp.x, y - tp.y ); }
    double operator * ( const Point &tp ) const { return x * tp.x + y * tp.y; }
    double operator ^ ( const Point &tp ) const { return x * tp.y - y * tp.x; }
    bool operator < ( const Point &tp ) const {
        if( sig( x - tp.x ) ) return sig( x - tp.x ) < 0;
        else return sig( y - tp.y ) < 0;
    }
};
Point poi[5], cir, src;
double r;

double dist( const Point &a, const Point &b ) {
    double dx = a.x - b.x;
    double dy = a.y - b.y;
    return sqrt( dx * dx + dy * dy );
}

double dptol( const Point &a, const Point &b, const Point &c ) {
    double ret = 0;
    if( sig( ( c - b ) * ( a - b ) ) > 0 && sig( ( b - c ) * ( a - c ) ) > 0 )
        ret = fabs( ( b - a ) ^ ( c - a ) ) / dist( b, c );
    else ret = min( dist( a, b ), dist( a, c ) );
    return ret;
}
```

```
double dptor( const Point &a ) {
    double d1 = min( dptol( a, poi[0], poi[1] ), dptol( a, poi[0], poi[2] ) );
    double d2 = min( dptol( a, poi[1], poi[3] ), dptol( a, poi[2], poi[3] ) );
    return min( d1, d2 );
}

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

点在多边形内

```
#include <bits/stdc++.h>
using namespace std;
const double eps = 1e-8;
const double PI = acos( -1.0 );

int sig( double x ) {
    if( fabs( x ) < eps ) return 0;
    return x > 0 ? 1 : -1;
}

struct Point {
    double x, y;
    Point() {}
    Point( const double xx, const double yy ) { x = xx; y = yy; }
    Point operator + ( const Point &tp ) const { return Point( x + tp.x, y + tp.y ); }
    Point operator - ( const Point &tp ) const { return Point( x - tp.x, y - tp.y ); }
    double operator * ( const Point &tp ) const { return x * tp.x + y * tp.y; }
    double operator ^ ( const Point &tp ) const { return x * tp.y - y * tp.x; }
    bool operator < ( const Point &tp ) const {
        if( sig( x - tp.x ) ) return sig( x - tp.x ) < 0;
        else return sig( y - tp.y ) < 0;
    }
};
typedef Point pVector;
typedef vector<Point> Polygon;

bool OnSegment( Point p, Point a, Point b ) {
    if( sig( ( p - a ) * ( p - b ) ) ) return 0;
    return sig( a.x - p.x ) * sig( b.x - p.x ) <= 0 && sig( a.y - p.y ) * sig( b.y -
p.y ) <= 0;
}
```

```
int isPointInPolygon( Point p, Polygon poly ) {
    int wn = 0;
    int n = poly.size();
    for( int i = 0; i < n; ++i ) {
        if( OnSegment( p, poly[i], poly[( i + 1 ) % n] ) ) return 0;
        int k = sig( ( poly[( i + 1 ) % n] - poly[i] ) ^ ( p - poly[i] ) );
        int d1 = sig( poly[i].y - p.y );
        int d2 = sig( poly[( i + 1 ) % n].y - p.y );
        if( k > 0 && d1 <= 0 && d2 > 0 ) ++wn;
        if( k < 0 && d2 <= 0 && d1 > 0 ) --wn;
    }
    return wn;
}

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

多边形与圆面积交

```
#include <algorithm>
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
using namespace std;
typedef long long LL;
const double INF = 1000000000000;
const double eps = 1e-12;
const double PI = acos( -1.0 );
const int MAXN = 100009;
const int MOD = 1000000007;

struct Point {
    double x,y;
    Point(){}
    Point( double xx, double yy ) { x = xx; y = yy; }
    Point operator - ( Point s ) { return Point( x - s.x, y - s.y ); }
    Point operator + ( Point s ) { return Point( x + s.x, y + s.y ); }
    double operator * ( Point s ) { return x * s.x + y * s.y; }
    double operator ^ ( Point s ) { return x * s.y - y * s.x; }
}poi[MAXN];

double mmax( double a, double b ) { return a > b ? a : b; }
double mmin( double a, double b ) { return a < b ? a : b; }
```

```
double len( Point a ) { return sqrt( a * a ); }
double dist( Point a, Point b ) { return len( b - a ); }

double cross( Point a, Point b, Point c ) { return ( b - a ) ^ ( c - a ); }
double dot( Point a, Point b, Point c ) { return ( b - a ) * ( c - a ); }

double area( Point b, Point c, double r ) {
    Point a( 0.0, 0.0 );
    if( dist( b, c ) < eps ) return 0.0;
    double h = fabs( cross( a, b, c ) ) / dist( b, c );
    //两个端点都在圆的外面则分为两种情况
    //两个端点都在圆内的情况
    //一个端点在圆上一个端点在圆内的情况
    if( dist( a, b ) > r - eps && dist( a, c ) > r - eps ) {
        double angle = acos( dot( a, b, c ) / dist( a, b ) / dist( a, c ) );
        if( h > r - eps ) return 0.5 * r * r * angle;
        else if( dot( b, a, c ) > 0 && dot( c, a, b ) > 0 ) {
            double angle1 = 2 * acos( h / r );
            return 0.5 * r * r * fabs( angle - angle1 ) + 0.5 * r * r * sin( angle1 );
        } else return 0.5 * r * r * angle;
    } else if( dist( a, b ) < r + eps && dist( a, c) < r + eps ) {
        return 0.5 * fabs( cross( a, b, c ) );
    } else {
        //默认 b 在圆内
        if( dist( a, b ) > dist( a, c ) ) swap(b,c);
        //ab 距离为 0 直接返回 0
        if( fabs( dist( a, b ) ) < eps ) return 0.0;
        if( dot( b, a, c ) < eps ) {
            double angle1 = acos( h / dist( a, b ) );
            double angle2 = acos( h / r ) - angle1;
            double angle3 = acos( h / dist( a, c ) ) - acos( h / r );
            return 0.5 * dist( a, b ) * r * sin( angle2 ) + 0.5 * r * r * angle3;
        } else {
            double angle1 = acos( h / dist( a, b ) );
            double angle2 = acos( h / r );
            double angle3 = acos( h / dist( a, c ) ) - angle2;
            return 0.5 * r * dist( a, b ) * sin( angle1 + angle2 ) + 0.5 * r * r *
angle3;
        }
    }
    return 0.0;
}
```

```
int main() {
    int n;
    double x, y, v, ang, t, g, r;
    while( ~scanf( "%lf%lf%lf%lf%lf%lf%lf", &x, &y, &v, &ang, &t, &g, &r ) &&
           x + y + v + ang + t + g + r ) {
        scanf( "%d", &n );
        for( int i = 0; i < n; ++i ) {
            scanf( "%lf%lf", &poi[i].x, &poi[i].y );
        }
        poi[n] = poi[0];
        Point O( x, y );
        double tmp = sin( ang / 180 * PI );
        O.x += v * t * cos( ang / 180 * PI );
        if( t * g <= v ) O.y += ( v * tmp + ( v * tmp - g * t ) ) / 2 * t;
        else {
            double tt = v * tmp / g;
            O.y += ( v * tmp / 2 ) * tt;
            tt = t - tt;
            O.y -= ( g * tt * tt ) / 2;
        }
        for( int i = 0; i <= n; ++i ) poi[i] = poi[i] - O;
        O = Point( 0, 0 );
        double sum = 0;
        for( int i = 0; i < n; ++i ) {
            int j = i + 1;
            double s = area( poi[i], poi[j], r );
            if( cross( O, poi[i], poi[j] ) > 0 ) sum += s;
            else sum -= s;
        }
        printf( "%.2lf\n", fabs( sum ) );
    }
    return 0;
}
```

矩形面积并

```
//¾ØÐÎÃæ»ý²¢
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <string>
#include <functional>
using namespace std;
```

```
const double eps = 1e-10;
int n;
pair<double, int> c[10000];
struct point
{
    double x, y;
} p[600][5];
int dblcmake_pair( double x )
{
    if( fabs(x) < eps )  return 0;
    return x > 0 ? 1 : -1;
}
double cross( point& p1, point& p2, point& p3 )
{
    return (p2.x-p1.x)*(p3.y-p1.y) - (p2.y-p1.y)*(p3.x-p1.x);
}
double dot( point aa, point bb )
{
    return aa.x*bb.x + aa.y*bb.y;
}
double segP( point p1, point p2, point p3 )
{
    if( dblcmake_pair(p2.x-p3.x) )
        return (p1.x-p2.x)/(p3.x-p2.x);
    else
        return (p1.y-p2.y)/(p3.y-p2.y);
}
double polyUnion()
{
    int i, j, ii, jj, ta, tb, r, d;
    double z, w, s, sum, tc, td;
    point tmake_pair1, tmake_pair2;
    sum = 0;
    for( i = 0; i < n; ++i )  for( ii = 0; ii < 4; ++ii )
    {
        r = 0;
        c[r++] = make_pair(0., 0);
        c[r++] = make_pair(1., 0);
        for( j = 0; j < n; ++j )  if( i-j )
            for( jj = 0; jj < 4; ++jj )
            {
                ta = dblcmake_pair( cross(p[i][ii], p[i][ii+1], p[j][jj]) );
                tb = dblcmake_pair( cross(p[i][ii], p[i][ii+1], p[j][jj+1]) );
```

```
                if( !ta && !tb )
                {
                    tmake_pair1.x = p[j][jj+1].x-p[j][jj].x;
                    tmake_pair1.y = p[j][jj+1].y-p[j][jj].y;
                    tmake_pair2.x = p[i][ii+1].x-p[i][ii].x;
                    tmake_pair2.y = p[i][ii+1].y-p[i][ii].y;
                    if( dblcmake_pair( dot(tmake_pair1, tmake_pair2) ) > 0 &&j < i )
                    {
                       c[r++] = make_pair( segP(p[j][jj],p[i][ii],p[i][ii+1]), 1 );
                       c[r++]=make_pair(segP(p[j][jj+1],p[i][ii],p[i][ii+1]),-1 );
                    }
                }
                else if( ta >= 0 && tb < 0 )
                {
                    tc = cross(p[j][jj], p[j][jj+1], p[i][ii]);
                    td = cross(p[j][jj], p[j][jj+1], p[i][ii+1]);
                    c[r++] = make_pair(tc/(tc-td), 1);
                }
                else if( ta < 0 && tb >= 0 )
                {
                    tc = cross(p[j][jj], p[j][jj+1], p[i][ii]);
                    td = cross(p[j][jj], p[j][jj+1], p[i][ii+1]);
                    c[r++] = make_pair(tc/(tc-td), -1);
                }
            }
        sort(c, c+r);
        z = min(max(c[0].first, 0.), 1.);
        d = c[0].second;
        s = 0;
        for( j = 1; j < r; ++j )
        {
            w = min(max(c[j].first, 0.), 1.);
            if( !d ) s += w-z;
            d += c[j].second;
            z = w;
        }
        tmake_pair1.x = tmake_pair1.y = 0;
        sum += cross(tmake_pair1, p[i][ii], p[i][ii+1])*s;
    }
    return 0.5*sum;
}
int main()
{
    int i, j;
```

```
    double area, tmake_pair;
    while( scanf("%d", &n) != EOF )
    {
        area = 0;
        for( i = 0; i < n; ++i )
        {
            for( j = 0; j < 4; ++j )
                scanf("%lf %lf", &p[i][j].x, &p[i][j].y);
            p[i][4] = p[i][0];
            tmake_pair = 0;
            for( j = 1; j <= 4; ++j )
                tmake_pair += p[i][j-1].x*p[i][j].y - p[i][j-1].y*p[i][j].x;
            area += fabs(tmake_pair);
            if( dblcmake_pair(tmake_pair) < 0 )   swap(p[i][1], p[i][3]);
        }
        printf("%.10lf\n", 0.5*area/polyUnion() );
    }
    return 0;
}
```

凸多边形面积并

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

const int maxn = 505;
const int maxm = 5005;
const double eps = 1e-8;
const double PI = acos(-1.0);

inline int dcmp(double a)
{
    return a < -eps ? -1 : a > eps;
}

struct Point
{
    double x, y;
    Point(){}
    Point(double a, double b): x(a), y(b){}
    bool operator < (const Point p) const
    {
```

```
        return y+eps < p.y || (y < p.y+eps && x+eps < p.x);
    }
    bool operator == (const Point p) const
    {
        return !dcmp(x-p.x) && !dcmp(y-p.y);
    }
    Point operator + (const Point p) const
    {
        return Point(x+p.x, y+p.y);
    }
    Point operator - (const Point p) const
    {
        return Point(x-p.x, y-p.y);
    }
    Point operator * (const double p) const
    {
        return Point(x*p, y*p);
    }
    Point operator / (const double p) const
    {
        return Point(x/p, y/p);
    }
    double operator * (const Point p) const
    {
        return x*p.y - y*p.x;
    }
    double operator / (const Point p) const
    {
        return x*p.x + y*p.y;
    }
    void input()
    {
        scanf("%lf %lf", &x, &y);
    }
};

struct Polygon
{
    int n;
    Point p[maxn];
    Point& operator [] (const int i)
    {
        return p[i];
    }
```

```
    void init()
    {
        double x1, x2, y1, y2;
        n = 4;
        for(int i = 0; i < 4; i++)
            p[i].input();
    }
    double Area()
    {
        double sum = 0;
        for(int i = 1; i < n-1; i++)
            sum += (p[i]-p[0]) * (p[i+1]-p[0]);
        return sum / 2.0;
    }
};

struct Polygons
{
    int n;
    Polygon py[maxn];
    std::pair <double, int> c[maxm];
    void init()
    {
        n = 0;
    }
    void push(Polygon p)
    {
        p[p.n] = p[0];
        py[n++] = p;
    }
    double seg(Point p, Point p1, Point p2)
    {
        if(!dcmp(p1.x-p2.x))
            return (p.y-p1.y) / (p2.y-p1.y);
        return (p.x-p1.x) / (p2.x-p1.x);
    }
    double PolyUnion()
    {
        int d, r, ta, tb;
        double s, w, z, sum, tc, td;
        sum = 0;
        for(int i = 0; i < n; i++)
            for(int ii = 0; ii < py[i].n; ii++)
            {
```

```
                r = 0;
                c[r++] = make_pair(0.0, 0);
                c[r++] = make_pair(1.0, 0);
                for(int j = 0; j < n; j++)
                {
                    if(i == j)
                        continue;
                    for(int jj = 0; jj < py[j].n; jj++)
                    {
                        ta = dcmp((py[i][ii+1]-py[i][ii])*(py[j][jj]-py[i][ii]));
                       tb = dcmp((py[i][ii+1]-py[i][ii])*(py[j][jj+1]-py[i][ii]));
                        if(!ta && !tb)
                        {
                            if((py[j][jj+1]-py[j][jj])/(py[i][ii+1]-py[i][ii]) > 0
&& j < i)
                            {
                                c[r++]   =   make_pair(seg(py[j][jj],   py[i][ii],
py[i][ii+1]), 1);
                                c[r++]  =  make_pair(seg(py[j][jj+1],  py[i][ii],
py[i][ii+1]), -1);
                            }
                        }
                        else if(ta >= 0 && tb < 0)
                        {
                            tc = (py[j][jj+1]-py[j][jj]) * (py[i][ii]-py[j][jj]);
                            td = (py[j][jj+1]-py[j][jj]) * (py[i][ii+1]-py[j][jj]);
                            c[r++] = make_pair(tc/(tc-td), 1);
                        }
                        else if(ta < 0 && tb >= 0)
                        {
                            tc = (py[j][jj+1]-py[j][jj]) * (py[i][ii]-py[j][jj]);
                            td = (py[j][jj+1]-py[j][jj]) * (py[i][ii+1]-py[j][jj]);
                            c[r++] = make_pair(tc/(tc-td), -1);
                        }
                    }
                }
                std::sort(c, c+r);
                z = std::min(std::max(c[0].first, 0.0), 1.0);
                d = c[0].second;
                s = 0;
                for(int j = 1; j < r; j++)
                {
                    w = std::min(std::max(c[j].first, 0.0), 1.0);
                    if(!d)
```

```
                        s += w - z;
                    d += c[j].second;
                    z = w;
                }
                sum += (py[i][ii]*py[i][ii+1]) * s;
            }
        return sum / 2.0;
    }
};

Polygons P;
Polygon pp;

int main()
{
    int n;
    double area, sum = 0;

    scanf("%d",&n);
    P.init();
    for(int i = 0; i < n; i++)
    {
        pp.init();
        area = pp.Area();
        if(area < 0)
        {
            for(int j = 0,k = pp.n-1; j < k; j++, k--)
                std::swap(pp[j], pp[k]);
            area = -area;
        }
        sum += area;
        P.push(pp);
    }
    printf("%.10f\n", sum / P.PolyUnion());
    return 0;
}
```

旋转卡壳模板

```
//计算凸包直径，输入凸包 ch，顶点个数为 n，按逆时针排列，输出直径的平方
int rotating_calipers(Point *ch,int n)
{
   int q=1,ans=0;
   ch[n]=ch[0];
   for(int p=0;p<n;p++)
```

```
    {
        while(cross(ch[p+1],ch[q+1],ch[p])>cross(ch[p+1],ch[q],ch[p]))
            q=(q+1)%n;
        ans=max(ans,max(dist2(ch[p],ch[q]),dist2(ch[p+1],ch[q+1])));
    }
    return ans;
}
```

圆的面积并

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<algorithm>
#include<cstring>
#define ld double
#define eps 1e-13
using namespace std;
int n,top,st,ed;
ld xl[1001],xr[1001];
ld ans;
bool del[1001];
struct data{ld x,y,r;}t[1001],sk[1001];
struct line{ld l,r;}p[1001];
ld dis(data a,data b)
{return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));}
bool cmp1(data a,data b){return a.r<b.r;}
bool cmp2(data a,data b){return a.x-a.r<b.x-b.r;}
bool cmp3(line a,line b){return a.l<b.l;}
void ini()
{
     scanf("%d",&n);
     for(int i=1;i<=n;i++)
        {scanf("%lf%lf%lf",&t[i].x,&t[i].y,&t[i].r);}
   sort(t+1,t+n+1,cmp1);
   for(int i=1;i<=n;i++)
      for(int j=i+1;j<=n;j++)
         if(dis(t[i],t[j])<=t[j].r-t[i].r)
                {del[i]=1;break;}
   for(int i=1;i<=n;i++)if(!del[i])sk[++top]=t[i];n=top;
   sort(sk+1,sk+n+1,cmp2);
}
ld getf(ld x)
{
    int sz=0,i,j;ld r,len=0,dis;
```

```
    for(i=st;i<=ed;i++)
   {
      if(x<=xl[i]||x>=xr[i])continue;
      dis=sqrt(sk[i].r-(x-sk[i].x)*(x-sk[i].x));
      p[++sz].l=sk[i].y-dis;p[sz].r=sk[i].y+dis;
   }
    sort(p+1,p+sz+1,cmp3);
    for(i=1;i<=sz;i++)
    {
        r=p[i].r;
        for(j=i+1;j<=sz;j++)
        {
            if(p[j].l>r)break;
            if(r<p[j].r)r=p[j].r;
        }
        len+=r-p[i].l;i=j-1;
    }
    return len;
}
ld cal(ld l,ld fl,ld fmid,ld fr)
{return (fl+fmid*4+fr)*l/6;}
ld simpson(ld l,ld mid,ld r,ld fl,ld fmid,ld fr,ld s)
{
    ld m1=(l+mid)/2,m2=(r+mid)/2;
    ld f1=getf(m1),f2=getf(m2);
    ld g1=cal(mid-l,fl,f1,fmid),g2=cal(r-mid,fmid,f2,fr);
    if(fabs(g1+g2-s)<eps)return g1+g2;
   return simpson(l,m1,mid,fl,f1,fmid,g1)+simpson(mid,m2,r,fmid,f2,fr,g2);
}
void work()
{
    int i,j;ld l,r,mid,fl,fr,fmid;
   for(i=1;i<=n;i++){xl[i]=sk[i].x-sk[i].r;xr[i]=sk[i].x+sk[i].r;sk[i].r*=sk[i].r;}
    for(i=1;i<=n;i++)
    {
        l=xl[i];r=xr[i];
        for(j=i+1;j<=n;j++)
        {
            if(xl[j]>r)break;
        if(xr[j]>r)r=xr[j];
        }
        st=i;ed=j-1;i=j-1;
     mid=(l+r)/2;
     fl=getf(l);fr=getf(r);fmid=getf(mid);
```

```
      ans+=simpson(l,mid,r,fl,fmid,fr,cal(r-l,fl,fmid,fr));
    }
}
int main()
{
    ini();
    work();
    printf("%.3lf",ans);
    return 0;
}
```

其他

归并排序(求逆序对数)

```
#include <iostream>
#include <cstdio>
using namespace std;
#define MAXN 500010
#define INF 0x3FFFFFFF

int L[MAXN], R[MAXN], a[MAXN];
long long cnt;

void _merge( int l, int m, int r ) {
    int i, j, k;
    int n1, n2;
    n1 = m - l + 1;
    n2 = r - m;
    for( i = 0; i < n1; ++i )
        L[i] = a[l + i];
    for( i = 0; i < n2; ++i )
        R[i] = a[m + 1 + i];
    L[n1] = INF;
    R[n2] = INF;
    i = j = 0;
    for( k = l; k <= r; ++k ) {
        if( L[i] <= R[j] )
            a[k] = L[i++];
        else {
            a[k] = R[j++];
            cnt += n1 - i;
        }
    }
```

```
    return ;
}

void _merge_sort( int l, int r ) {
    if( l < r ) {
        int m = ( l + r ) / 2;
        _merge_sort( l, m );
        _merge_sort( m + 1, r );
        _merge( l, m, r );
    }
    return ;
}

int main() {
    int i;
    int n;
    while( ~scanf( "%d", &n ) && n ) {
        cnt = 0;
        for( i = 0; i < n; ++i )
            scanf( "%d", a + i );
        _merge_sort( 0, n - 1 );
        printf( "%lld\n", cnt );
    }
    return 0;
}
```

大数模板

```
#include<iostream>
#include<string>
#include<cstring>
#include<iomanip>
#include<algorithm>
using namespace std;

#define MAXN 9999
#define MAXSIZE 10
#define DLEN 4

class BigNum
{
private:
    int a[500];    //可以控制大数的位数
    int len;       //大数长度
public:
```

```
    BigNum(){ len = 1;memset(a,0,sizeof(a)); }   //构造函数
    BigNum(const int);       //将一个 int 类型的变量转化为大数
    BigNum(const char*);     //将一个字符串类型的变量转化为大数
    BigNum(const BigNum &);  //拷贝构造函数
    BigNum &operator=(const BigNum &);   //重载赋值运算符，大数之间进行赋值运算

    friend istream& operator>>(istream&,  BigNum&);   //重载输入运算符
    friend ostream& operator<<(ostream&,  BigNum&);   //重载输出运算符

    BigNum operator+(const BigNum &) const; //重载加法运算符，两个大数之间的相加运算
    BigNum operator-(const BigNum &) const; //重载减法运算符，两个大数之间的相减运算
    BigNum operator*(const BigNum &) const; //重载乘法运算符，两个大数之间的相乘运算
    BigNum operator/(const int  &) const;//重载除法运算符，大数对一个整数进行相除运算

    BigNum operator^(const int  &) const;    //大数的 n 次方运算
    int    operator%(const int  &) const;    //大数对一个 int 类型的变量进行取模运算
    bool   operator>(const BigNum & T)const;   //大数和另一个大数的大小比较
    bool   operator>(const int & t)const;      //大数和一个 int 类型的变量的大小比较
    bool   operator<(const BigNum & t)const;

    void print();       //输出大数
};
BigNum::BigNum(const int b)     //将一个 int 类型的变量转化为大数
{
    int c,d = b;
    len = 0;
    memset(a,0,sizeof(a));
    while(d > MAXN)
    {
        c = d - (d / (MAXN + 1)) * (MAXN + 1);
        d = d / (MAXN + 1);
        a[len++] = c;
    }
    a[len++] = d;
}
BigNum::BigNum(const char*s)     //将一个字符串类型的变量转化为大数
{
    int t,k,index,l,i;
    memset(a,0,sizeof(a));
    l=strlen(s);
    len=l/DLEN;
    if(l%DLEN)
        len++;
    index=0;
```

```
    for(i=l-1;i>=0;i-=DLEN)
    {
        t=0;
        k=i-DLEN+1;
        if(k<0)
            k=0;
        for(int j=k;j<=i;j++)
            t=t*10+s[j]-'0';
        a[index++]=t;
    }
}
BigNum::BigNum(const BigNum & T) : len(T.len)  //拷贝构造函数
{
    int i;
    memset(a,0,sizeof(a));
    for(i = 0 ; i < len ; i++)
        a[i] = T.a[i];
}
BigNum & BigNum::operator=(const BigNum & n)   //重载赋值运算符，大数之间进行赋值运算
{
    int i;
    len = n.len;
    memset(a,0,sizeof(a));
    for(i = 0 ; i < len ; i++)
        a[i] = n.a[i];
    return *this;
}
istream& operator>>(istream & in,  BigNum & b)   //重载输入运算符
{
    char ch[MAXSIZE*4];
    int i = -1;
    in>>ch;
    int l=strlen(ch);
    int count=0,sum=0;
    for(i=l-1;i>=0;)
    {
        sum = 0;
        int t=1;
        for(int j=0;j<4&&i>=0;j++,i--,t*=10)
        {
            sum+=(ch[i]-'0')*t;
        }
        b.a[count]=sum;
        count++;
```

```
    }
    b.len =count++;
    return in;

}
ostream& operator<<(ostream& out,  BigNum& b)   //重载输出运算符
{
    int i;
    cout << b.a[b.len - 1];
    for(i = b.len - 2 ; i >= 0 ; i--)
    {
        cout.width(DLEN);
        cout.fill('0');
        cout << b.a[i];
    }
    return out;
}

BigNum BigNum::operator+(const BigNum & T) const   //两个大数之间的相加运算
{
    BigNum t(*this);
    int i,big;      //位数
    big = T.len > len ? T.len : len;
    for(i = 0 ; i < big ; i++)
    {
        t.a[i] +=T.a[i];
        if(t.a[i] > MAXN)
        {
            t.a[i + 1]++;
            t.a[i] -=MAXN+1;
        }
    }
    if(t.a[big] != 0)
        t.len = big + 1;
    else
        t.len = big;
    return t;
}
BigNum BigNum::operator-(const BigNum & T) const   //两个大数之间的相减运算
{
    int i,j,big;
    bool flag;
    BigNum t1,t2;
    if(*this>T)
```

```
    {
        t1=*this;
        t2=T;
        flag=0;
    }
    else
    {
        t1=T;
        t2=*this;
        flag=1;
    }
    big=t1.len;
    for(i = 0 ; i < big ; i++)
    {
        if(t1.a[i] < t2.a[i])
        {
            j = i + 1;
            while(t1.a[j] == 0)
                j++;
            t1.a[j--]--;
            while(j > i)
                t1.a[j--] += MAXN;
            t1.a[i] += MAXN + 1 - t2.a[i];
        }
        else
            t1.a[i] -= t2.a[i];
    }
    t1.len = big;
    while(t1.a[len - 1] == 0 && t1.len > 1)
    {
        t1.len--;
        big--;
    }
    if(flag)
        t1.a[big-1]=0-t1.a[big-1];
    return t1;
}

BigNum BigNum::operator*(const BigNum & T) const   //两个大数之间的相乘运算
{
    BigNum ret;
    int i,j,up;
    int temp,temp1;
    for(i = 0 ; i < len ; i++)
```

```
    {
        up = 0;
        for(j = 0 ; j < T.len ; j++)
        {
            temp = a[i] * T.a[j] + ret.a[i + j] + up;
            if(temp > MAXN)
            {
                temp1 = temp - temp / (MAXN + 1) * (MAXN + 1);
                up = temp / (MAXN + 1);
                ret.a[i + j] = temp1;
            }
            else
            {
                up = 0;
                ret.a[i + j] = temp;
            }
        }
        if(up != 0)
            ret.a[i + j] = up;
    }
    ret.len = i + j;
    while(ret.a[ret.len - 1] == 0 && ret.len > 1)
        ret.len--;
    return ret;
}
BigNum BigNum::operator/(const int & b) const   //大数对一个整数进行相除运算
{
    BigNum ret;
    int i,down = 0;
    for(i = len - 1 ; i >= 0 ; i--)
    {
        ret.a[i] = (a[i] + down * (MAXN + 1)) / b;
        down = a[i] + down * (MAXN + 1) - ret.a[i] * b;
    }
    ret.len = len;
    while(ret.a[ret.len - 1] == 0 && ret.len > 1)
        ret.len--;
    return ret;
}
int BigNum::operator %(const int & b) const    //大数对一个 int 类型的变量进行取模运算
{
    int i,d=0;
    for (i = len-1; i>=0; i--)
    {
```

```
        d = ((d * (MAXN+1))% b + a[i])% b;
    }
    return d;
}
BigNum BigNum::operator^(const int & n) const    //大数的 n 次方运算
{
    BigNum t,ret(1);
    int i;
    if(n<0)
        exit(-1);
    if(n==0)
        return 1;
    if(n==1)
        return *this;
    int m=n;
    while(m>1)
    {
        t=*this;
        for( i=1;i<<1<=m;i<<=1)
        {
            t=t*t;
        }
        m-=i;
        ret=ret*t;
        if(m==1)
            ret=ret*(*this);
    }
    return ret;
}
bool BigNum::operator>(const BigNum & T) const   //大数和另一个大数的大小比较
{
    int ln;
    if(len > T.len)
        return true;
    else if(len == T.len)
    {
        ln = len - 1;
        while(a[ln] == T.a[ln] && ln >= 0)
            ln--;
        if(ln >= 0 && a[ln] > T.a[ln])
            return true;
        else
            return false;
    }
```

```
    else
        return false;
}
bool BigNum::operator >(const int & t) const    //大数和一个 int 类型的变量的大小比较
{
    BigNum b(t);
    return *this>b;
}

bool BigNum::operator <(const BigNum & t)const
{
   return t > *this;
}

void BigNum::print()    //输出大数
{
    int i;
    cout << a[len - 1];
    for(i = len - 2 ; i >= 0 ; i--)
    {
        cout.width(DLEN);
        cout.fill('0');
        cout << a[i];
    }
    cout << endl;
}
BigNum pow( BigNum a, int b ) {
   BigNum res( 1 );
   while( b ) {
       if( b % 2 == 1 ) res = res * a;
       a = a * a;
       b = b / 2;
   }
   return res;
}

int main() {
//    long long sum;
   for( int n = 1; n <= 10; ++n ) {
       BigNum ten( 10 );
       BigNum a( pow( ten, n - 1 ) ), b( pow( ten, n ) );
       for( BigNum i = a; i < b; i = i + 1 ) {
           BigNum tmp = i;
           BigNum sum( 0 );
```

```
            while( tmp > 0 ) {
                sum = sum + pow( tmp % 10, n );
                tmp = tmp / 10;
            }
            if( !( sum > i ) && !( sum < i ) ) i.print();//cout << i << endl;
        }
    }
    return 0;
}
```