

```
// Kiem tra tinh lien thong manh va dem bo phan lien thong manh 1  
// do thi vo huong  
#include <stdio.h>
```

```
int num[100];  
int min_num[100];  
int on_stack[100];  
int k;  
int is_strong_connect;
```

```
int min(int x, int y) {  
    if (x <= y) {  
        return x;  
    }  
    return y;  
}
```

```
// List  
typedef struct {  
    int data[100];  
    int size;  
} List;
```

```
void make_null_list(List* L) {  
    L->size = 0;  
}
```

```
void push_back(List* L, int x) {  
    L->data[L->size] = x;  
    ++L->size;  
}
```

```
int element_at(List* L, int i) {  
    return L->data[i - 1];  
}
```

```
// Stack  
typedef struct {  
    int data[100];  
    int size;  
} Stack;
```

```
void make_null_stack(Stack* S) {
```

```

        S->size = 0;
    }

    void push(Stack* S, int x) {
        S->data[S->size] = x;
        ++S->size;
    }

    int top(Stack* S) {
        return S->data[S->size - 1];
    }

    void pop(Stack* S) {
        --S->size;
    }

    int empty(Stack* S) {
        return S->size == 0;
    }

    Stack S;

    // Graph
    typedef struct {
        int A[100][100];
        int n;
    } Graph;

    void init_graph(Graph* G, int n) {
        G->n = n;

        int i, j;
        for (i = 1; i <= n; ++i) {
            for (j = 1; j <= n; ++j) {
                G->A[i][j] = 0;
            }
        }
    }

    void add_egde(Graph* G, int x, int y) {
        G->A[x][y] = 1;
        //G->A[y][x] = 1;
    }

```

```

int adjacent(Graph* G, int x, int y) {
    return G->A[x][y];
}

List neighbors(Graph* G, int x) {
    int y;
    List list;

    make_null_list(&list);

    for (y = 1; y <= G->n; ++y) {
        if (adjacent(G, x, y)) {
            push_back(&list, y);
        }
    }

    return list;
}

void strong_connect(Graph* G, int x) {
    num[x] = min_num[x] = k; ++k; // k = 1;
    push(&S, x);
    on_stack[x] = 1;

    List list = neighbors(G, x);

    int i;
    for (i = 1; i <= list.size; ++i) {
        int y = element_at(&list, i);

        if (num[y] < 0) {
            strong_connect(G, y);
            min_num[x] = min(min_num[x], min_num[y]);
        } else {
            if (on_stack[y]) {
                min_num[x] = min(min_num[x], num[y]);
            }
        }
    }

    if (num[x] == min_num[x]) {
        int count = 0;
    }
}

```

```

    int w;

    do {
        ++count;
        w = top(&S);
        pop(&S);
        on_stack[w] = 0;
    } while (w != x);

    if (count == G->n) {
        is_strong_connect = 1;
    }
}

int main() {
    //freopen("dt.txt", "r", stdin);

    Graph G;
    int n, m, i, x, y;

    scanf("%d%d", &n, &m);

    init_graph(&G, n);

    for (i = 1; i <= m; ++i) {
        scanf("%d%d", &x, &y);

        add_egde(&G, x, y);
    }

    for (i = 1; i <= n; ++i) {
        num[i] = -1;
        on_stack[i] = 0;
    }

    k = 1;
    make_null_stack(&S);
    is_strong_connect = 0;

    strong_connect(&G, 1);

    if (is_strong_connect) {

```

```
        printf("strong connected");
    } else {
        printf("unconnected");
    }

    return 0;
}
```