

Started on	Sunday, 15 June 2025, 9:13 PM
State	Finished
Completed on	Sunday, 15 June 2025, 9:37 PM
Time taken	24 mins 36 secs
Marks	4.00/4.00
Grade	10.00 out of 10.00 (100%)



Viết chương trình đọc vào một **đơn đồ thị có hướng** và kiểm tra xem nó có chứa chu trình hay không.

Nhắc lại: *Chu trình là một đường đi có đỉnh đầu trùng với đỉnh cuối.*

Đầu vào (Input):

Dữ liệu đầu vào được nhập từ bàn phím với định dạng:

- Dòng đầu tiên chứa 2 số nguyên n và m, tương ứng là số đỉnh và số cung.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u, v mô tả cung (u, v).

Đầu ra (Output):

- In ra màn hình **CIRCLED** nếu đồ thị có chứa chu trình, ngược lại in ra **NO CIRCLE**
- Xem thêm ví dụ bên dưới.

For example:

Input	Result
3 3 1 3 3 2 2 1	CIRCLED
7 9 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7	NO CIRCLE

Answer: (penalty regime: 10, 20, ... %)

```
1 #include <stdio.h>
2 #define max 100
3 typedef int ElementType;
4
5 typedef struct
6 {
7     int n, m;
8     int A[max][max];
9 } Graph;
10
11 void init_graph(Graph *pG, int n)
12 {
13     pG->n = n;
14     pG->m = 0;
15     for (int u = 1; u <= n; u++)
16     {
17         for (int v = 1; v <= n; v++)
18         {
19             pG->A[u][v] = 0;
20         }
21     }
22 }
23
24 void add_edge(Graph *pG, int u, int v)
25 {
26     pG->A[u][v] = 1;
27     // pG->A[v][u] = 1;
28     pG->m++;
29 }
30
31 int adjacent(Graph *pG, int u, int v)
32 {
33     return pG->A[u][v] > 0;
34 }
35
36 /* kiểm tra đồ thị có chứa chu trình*/
37 #define WHITE 0
38 #define GRAY 1
39 #define BLACK 2
40
41 int color[max]; // lưu trạng thái của các đỉnh
42 int has_circle; // đồ thị chứa chu trình hay không
43
44 void DFS(Graph *pG, int u){
45     color[u] = GRAY;
46
47     for(int v = 1; v <= pG->n; v++)
48     {
49         if(adjacent(pG, u,v)){
50             if(color[v] == WHITE) // 2a. nếu v chưa duyệt
51                 DFS(pG,v); // gọi đệ quy duyệt nó
52             else if(color[v] == GRAY)// 2b. nếu v đang duyệt
53                 has_circle = 1; //chứa chu trình
54         }
55     }
56 }
```



```

55     color[u] = BLACK;
56 }
57 int main()
58 {
59     Graph G;
60     int n, m, u, v;
61     scanf("%d%d", &n, &m);
62     init_graph(&G, n);
63
64     for (int e = 0; e < m; e++)
65     {
66         scanf("%d%d", &u, &v);
67         add_edge(&G, u, v);
68     }
69     for(int u = 1; u <= G.n; u++)
70         color[u] = WHITE;
71
72     has_circle = 0;
73
74     for(int u = 1; u <= G.n ; u++)
75         if(color[u] == WHITE)
76             DFS(&G,u);
77     if(has_circle == 1)
78         printf("CIRCLED");
79     else
80         printf("NO CIRCLE");
81     return 0;
82 }
83

```

Debug: source code from all test runs

Run 1

```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    // pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u){
    color[u] = GRAY;

    for(int v = 1; v <= pG->n; v++)
        if(adjacent(pG, u,v)){
            if(color[v] == WHITE) // 2a. nếu v chưa duyệt
                DFS(pG,v); // gọi đệ quy duyệt nó
            else if(color[v] == GRAY) // 2b. nếu v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    color[u] = BLACK;
}

int main()
{
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int e = 0; e < m; e++)
    {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    for(int u = 1; u <= G.n; u++)
        color[u] = WHITE;

    has_circle = 0;

    for(int u = 1; u <= G.n ; u++)
        if(color[u] == WHITE)
            DFS(&G,u);
    if(has_circle == 1)
        printf("CIRCLED");
    else
        printf("NO CIRCLE");
    return 0;
}

```

Run 2

```
#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    // pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u){
    color[u] = GRAY;

    for(int v = 1; v <= pG->n; v++){
        if(adjacent(pG, u,v)){
            if(color[v] == WHITE) // 2a. nếu v chưa duyệt
                DFS(pG,v); // gọi đệ quy duyệt nó
            else if(color[v] == GRAY) // 2b. nếu v đang duyệt
                has_circle = 1; //chứa chu trình
        }
    }

    color[u] = BLACK;
}

int main()
{
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int e = 0; e < m; e++)
    {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }
    for(int u = 1; u <= G.n; u++)
        color[u] = WHITE;

    has_circle = 0;

    for(int u = 1; u <= G.n ; u++)
        if(color[u] == WHITE)
            DFS(&G,u);
    if(has_circle == 1)
        printf("CIRCLED");
    else
        printf("NO CIRCLE");
    return 0;
}
```



```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    // pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u){
    color[u] = GRAY;

    for(int v = 1; v <= pG->n; v++)
        if(adjacent(pG, u,v)){
            if(color[v] == WHITE) // 2a. nếu v chưa duyệt
                DFS(pG,v); // gọi đệ quy duyệt nó
            else if(color[v] == GRAY) // 2b. nếu v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    color[u] = BLACK;
}

int main()
{
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int e = 0; e < m; e++)
    {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    for(int u = 1; u <= G.n; u++)
        color[u] = WHITE;

    has_circle = 0;

    for(int u = 1; u <= G.n ; u++)
        if(color[u] == WHITE)
            DFS(&G,u);
    if(has_circle == 1)
        printf("CIRCLED");
    else
        printf("NO CIRCLE");
    return 0;
}

```

```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    // pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u){
    color[u] = GRAY;

    for(int v = 1; v <= pG->n; v++){
        if(adjacent(pG, u,v)){
            if(color[v] == WHITE) // 2a. nếu v chưa duyệt
                DFS(pG,v); // gọi đệ quy duyệt nó
            else if(color[v] == GRAY) // 2b. nếu v đang duyệt
                has_circle = 1; //chứa chu trình
        }
    }

    color[u] = BLACK;
}

int main()
{
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int e = 0; e < m; e++)
    {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }
    for(int u = 1; u <= G.n; u++)
        color[u] = WHITE;

    has_circle = 0;

    for(int u = 1; u <= G.n ; u++)
        if(color[u] == WHITE)
            DFS(&G,u);
    if(has_circle == 1)
        printf("CIRCLED");
    else
        printf("NO CIRCLE");
    return 0;
}

```



```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    // pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u){
    color[u] = GRAY;

    for(int v = 1; v <= pG->n; v++)
        if(adjacent(pG, u,v)){
            if(color[v] == WHITE) // 2a. nếu v chưa duyệt
                DFS(pG,v); // gọi đệ quy duyệt nó
            else if(color[v] == GRAY) // 2b. nếu v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    color[u] = BLACK;
}

int main()
{
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int e = 0; e < m; e++)
    {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    for(int u = 1; u <= G.n; u++)
        color[u] = WHITE;

    has_circle = 0;

    for(int u = 1; u <= G.n ; u++)
        if(color[u] == WHITE)
            DFS(&G,u);
    if(has_circle == 1)
        printf("CIRCLED");
    else
        printf("NO CIRCLE");
    return 0;
}

```

```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    // pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u){
    color[u] = GRAY;

    for(int v = 1; v <= pG->n; v++){
        if(adjacent(pG, u,v)){
            if(color[v] == WHITE) // 2a. nếu v chưa duyệt
                DFS(pG,v); // gọi đệ quy duyệt nó
            else if(color[v] == GRAY) // 2b. nếu v đang duyệt
                has_circle = 1; //chứa chu trình
        }
    }

    color[u] = BLACK;
}

int main()
{
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int e = 0; e < m; e++)
    {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }
    for(int u = 1; u <= G.n; u++)
        color[u] = WHITE;

    has_circle = 0;

    for(int u = 1; u <= G.n ; u++)
        if(color[u] == WHITE)
            DFS(&G,u);
    if(has_circle == 1)
        printf("CIRCLED");
    else
        printf("NO CIRCLE");
    return 0;
}

```

	Input	Expected	Got	
✓	3 3 1 3 3 2 2 1	CIRCLED	CIRCLED	✓
✓	7 9 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7	NO CIRCLE	NO CIRCLE	✓
✓	7 11 1 2 1 3 2 4 2 5 2 6 3 2 5 3 3 6 4 7 7 5 6 4	CIRCLED	CIRCLED	✓
✓	3 2 1 2 2 3	NO CIRCLE	NO CIRCLE	✓
✓	4 3 1 2 2 3 4 2	NO CIRCLE	NO CIRCLE	✓
✓	4 4 1 2 2 3 3 4 4 1	CIRCLED	CIRCLED	✓

Passed all tests! ✓

Question author's solution (C):

```

1  #include <stdio.h>
2
3  /* Khai báo CTDL Graph*/
4  #define MAX_N 100
5  typedef struct {
6      int n, m;
7      int A[MAX_N][MAX_N];
8  } Graph;
9
10 void init_graph(Graph *pG, int n) {
11     pG->n = n;
12     pG->m = 0;
13     for (int u = 1 ; u <= n; u++)
14         for (int v = 1 ; v <= n; v++)
15             pG->A[u][v] = 0;
16 }
17
18 void add_edge(Graph *pG, int u, int v) {
19     pG->A[u][v] += 1;
20     //if (u != v)
21     //    pG->A[v][u] += 1;
22
23     if (pG->A[u][v] > 1)
24         printf("đa cung (%d, %d)\n", u, v);
25     if (u == v)
26         printf("khuyen %d\n", u);
27
28     pG->m++;
29 }
30
31
32 int adjacent(Graph *pG, int u, int v) {
33     return pG->A[u][v] > 0;
34 }
35
36 #define WHITE 0
37 #define GRAY 1
38 #define BLACK 2
39
40 int color[MAX_N];    //Lưu trạng thái của các đỉnh

```

```

41 int has_circle; //Đồ thị chứa trình hay không
42
43 void DFS(Graph *pG, int u) {
44     //1. Tô màu đang duyệt cho u
45     color[u] = GRAY;
46
47     //2. Xét các đỉnh kề của u
48     for (int v = 1; v <= pG->n; v++)
49         if (adjacent(pG, u, v)) {
50             if (color[v] == WHITE) //2a. Nếu v chưa duyệt
51                 DFS(pG, v); //gọi đệ quy duyệt nó
52             else if (color[v] == GRAY) //2b. v đang duyệt
53                 has_circle = 1; //chứa chu trình
54         }
55
56     //3. Tô màu duyệt xong cho u
57     color[u] = BLACK;
58 }
59
60
61 int main() {
62     //1. Khai báo đồ thị G
63     Graph G;
64     //2. Đọc dữ liệu và dựng đồ thị
65     int n, m, u, v;
66     scanf("%d%d", &n, &m);
67     init_graph(&G, n);
68     for (int e = 0; e < m; e++) {
69         scanf("%d%d", &u, &v);
70         add_edge(&G, u, v);
71     }
72
73     for (int u = 1; u <= G.n; u++)
74         color[u] = WHITE;
75     //2. Khởi tạo biến has_circle
76     has_circle = 0;
77     //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
78     for (int u = 1; u <= G.n; u++)
79         if (color[u] == WHITE) //u chưa duyệt
80             DFS(&G, u); //gọi DFS(&G, u) để duyệt từ u
81     //4. Kiểm tra has_circle
82
83     if (has_circle)
84         printf("CIRCLED\n");
85     else
86         printf("NO CIRCLE\n");
87
88     return 0;
89 }
90
91

```

Correct

Marks for this submission: 1.00/1.00.

Viết chương trình đọc vào một **đơn đồ thị có hướng** và kiểm tra xem nó có chứa chu trình hay không. Nếu đồ thị có chu trình, in ra các đỉnh có trong chu trình.

Đầu vào (Input):

Dữ liệu đầu vào được nhập từ bàn phím với định dạng:

- Dòng đầu tiên chứa 2 số nguyên n và m, tương ứng là số đỉnh và số cung.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u, v mô tả cung (u, v).

Đầu ra (Output):

- Nếu đồ thị có chu trình, in ra các đỉnh có trong chu trình trên cùng 1 dòng, cách nhau 1 khoảng trắng. Đỉnh đầu và đỉnh cuối của chu trình giống nhau.
- Nếu đồ thị không chứa chu trình, in ra -1
- Nếu đồ thị có nhiều chu trình, chỉ cần in ra 1 chu trình bất kỳ.
- Xem thêm ví dụ bên dưới.

Trong ví dụ đầu tiên, ta tìm thấy chu trình 1 -> 3 -> 2 -> 1, vì thế chương trình phải in ra:

1 3 2 1

For example:

Input	Result
3 3 1 3 3 2 2 1	Chu trình hợp lệ.
7 9 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7	-1

Answer: (penalty regime: 10, 20, ... %)

```
1 #include <stdio.h>
2
3 /* Khai báo CTDL Graph*/
4 #define MAX_N 100
5 typedef struct {
6     int n, m;
7     int A[MAX_N][MAX_N];
8 } Graph;
9
10 void init_graph(Graph *pG, int n) {
11     pG->n = n;
12     pG->m = 0;
13     for (int u = 1 ; u <= n; u++)
14         for (int v = 1 ; v <= n; v++)
15             pG->A[u][v] = 0;
16 }
17
18 void add_edge(Graph *pG, int u, int v) {
19     pG->A[u][v] += 1;
20     //if (u != v)
21     //    pG->A[v][u] += 1;
22
23     if (pG->A[u][v] > 1)
24         printf("đa cung (%d, %d)\n", u, v);
25     if (u == v)
26         printf("khuyen %d\n", u);
27
28     pG->m++;
29 }
30
31
32 int adjacent(Graph *pG, int u, int v) {
33     return pG->A[u][v] > 0;
34 }
35
36 #define WHITE 0
37 #define GRAY 1
38 #define BLACK 2
39
40 int color[MAX_N]; //Lưu trạng thái của các đỉnh
41 int parent[MAX_N]; //Lưu trạng thái của các đỉnh
42 int has_circle; //Đồ thị chứa trình hay không
43 int start, end;
44
45 void DFS(Graph *pG, int u, int p) {
46     //1. Tô màu đang duyệt cho u
```



```

47     color[u] = GRAY;
48     parent[u] = p;
49
50     //2. Xét các đỉnh kề của u
51     for (int v = 1; v <= pG->n; v++)
52     {
53         if (adjacent(pG, u, v)) {
54             if (color[v] == WHITE) //2a. Nếu v chưa duyệt
55                 DFS(pG, v, u); //gọi đệ quy duyệt nó
56             else if (color[v] == GRAY) { //2b. v đang duyệt
57                 has_circle = 1; //chứa chu trình
58                 start = u;
59                 end = v;
60             }
61         }
62
63         //3. Tô màu duyệt xong cho u
64         color[u] = BLACK;
65     }
66
67     int main() {
68         //1. Khai báo đồ thị G
69         Graph G;
70         //2. Đọc dữ liệu và dựng đồ thị
71         int n, m, u, v;
72         scanf("%d%d", &n, &m);
73         init_graph(&G, n);
74         for (int e = 0; e < m; e++) {
75             scanf("%d%d", &u, &v);
76             add_edge(&G, u, v);
77         }
78
79         for (int u = 1; u <= G.n; u++)
80             color[u] = WHITE;
81         //2. Khởi tạo biến has_circle
82         has_circle = 0;
83         //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
84         for (int u = 1; u <= G.n; u++)
85             if (color[u] == WHITE) //u chưa duyệt
86                 DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
87         //4. Kiểm tra has_circle
88
89         if (has_circle) {
90             int A[100], i = 0;
91             A[0] = start;
92             int u = start;
93             do {
94                 u = parent[u];
95                 i++;
96                 A[i] = u;
97             } while (u != v);
98
99             for (int j = i; j >= 0; j--)
100                 printf("%d ", A[j]);
101
102             printf("%d\n", A[i]);
103         } else
104             printf("-1\n");
105
106         return 0;
107     }
108
109 }

```

	Input	Expected	Got	
✓	3 3 1 3 3 2 2 1	Chu trình hợp lệ.	Chu trình hợp lệ.	✓
✓	7 9 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7	-1	-1	✓

	Input	Expected	Got	
✓	7 11 1 2 1 3 2 4 2 5 2 6 3 2 5 3 3 6 4 7 7 5 6 4	Chu trình hợp lệ.	Chu trình hợp lệ.	✓
✓	3 2 1 2 2 3	-1	-1	✓
✓	4 3 1 2 2 3 4 2	-1	-1	✓
✓	4 4 1 2 2 3 3 4 4 1	Chu trình hợp lệ.	Chu trình hợp lệ.	✓

Passed all tests! ✓

Question author's solution (Python3):

```

1 #include <stdio.h>
2
3 /* Khai báo CTDL Graph*/
4 #define MAX_N 100
5 typedef struct {
6     int n, m;
7     int A[MAX_N][MAX_N];
8 } Graph;
9
10 void init_graph(Graph *pG, int n) {
11     pG->n = n;
12     pG->m = 0;
13     for (int u = 1 ; u <= n; u++)
14         for (int v = 1 ; v <= n; v++)
15             pG->A[u][v] = 0;
16 }
17
18 void add_edge(Graph *pG, int u, int v) {
19     pG->A[u][v] += 1;
20     //if (u != v)
21     //    pG->A[v][u] += 1;
22
23     if (pG->A[u][v] > 1)
24         printf("da cung (%d, %d)\n", u, v);
25     if (u == v)
26         printf("khuyen %d\n", u);
27
28     pG->m++;
29 }
30
31
32 int adjacent(Graph *pG, int u, int v) {
33     return pG->A[u][v] > 0;
34 }
35
36 #define WHITE 0
37 #define GRAY 1
38 #define BLACK 2
39
40 int color[MAX_N]; //Lưu trạng thái của các đỉnh
41 int parent[MAX_N]; //Lưu trạng thái của các đỉnh
42 int has_circle; //Đồ thị chứa trình hay không
43 int start, end;
44
45 void DFS(Graph *pG, int u, int p) {
46     //1. Tô màu đang duyệt cho u
47     color[u] = GRAY;
48     parent[u] = p;
49
50     //2. Xét các đỉnh kề của u
51     for (int v = 1; v <= pG->n; v++)
52         if (adjacent(pG, u, v)) {
53             if (color[v] == WHITE) //2a. Nếu v chưa duyệt
54                 DFS(pG, v, u); //gọi đệ quy duyệt nó
55             else if (color[v] == GRAY) { //2b. v đang duyệt
56                 has_circle = 1; //chứa chu trình
57                 start = u;

```



```

58         end = v;
59     }
60 }
61
62 //3. Tô màu duyệt xong cho u
63 color[u] = BLACK;
64 }
65
66
67 int main() {
68     //1. Khai báo đồ thị G
69     Graph G;
70     //2. Đọc dữ liệu và dựng đồ thị
71     int n, m, u, v;
72     scanf("%d%d", &n, &m);
73     init_graph(&G, n);
74     for (int e = 0; e < m; e++) {
75         scanf("%d%d", &u, &v);
76         add_edge(&G, u, v);
77     }
78
79     for (int u = 1; u <= G.n; u++)
80         color[u] = WHITE;
81     //2. Khởi tạo biến has_circle
82     has_circle = 0;
83     //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
84     for (int u = 1; u <= G.n; u++)
85         if (color[u] == WHITE) //u chưa duyệt
86             DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
87     //4. Kiểm tra has_circle
88
89     if (has_circle) {
90         int A[100], i = 0;
91         A[0] = start;
92         int u = start;
93         do {
94             u = parent[u];
95             i++;
96             A[i] = u;
97         } while (u != v);
98
99         for (int j = i; j >= 0; j--)
100             printf("%d ", A[j]);
101
102         printf("%d\n", A[i]);
103     } else
104         printf("-1\n");
105
106     return 0;
107 }
108
109

```

Correct

Marks for this submission: 1.00/1.00.

Viết chương trình đọc vào một **đơn đồ thị vô hướng** và kiểm tra xem nó có chứa chu trình hay không.

Nhắc lại: *Chu trình là một đường đi có đỉnh đầu trùng với đỉnh cuối.*

Đầu vào (Input):

Dữ liệu đầu vào được nhập từ bàn phím với định dạng:

- Dòng đầu tiên chứa 2 số nguyên n và m, tương ứng là số đỉnh và số cung.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u, v mô tả cung (u, v).

Đầu ra (Output):

- In ra màn hình **CIRCLED** nếu đồ thị có chứa chu trình, ngược lại in ra **NO CIRCLE**
- Xem thêm ví dụ bên dưới.

For example:

Input	Result
3 2 1 3 3 2	NO CIRCLE
7 9 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7	CIRCLED
7 11 1 2 1 3 2 4 2 5 2 6 3 2 3 5 3 6 4 7 5 7 6 4	CIRCLED
3 2 1 2 2 3	NO CIRCLE
4 3 1 2 2 3 4 2	NO CIRCLE

Answer: (penalty regime: 10, 20, ... %)

```
1 #include <stdio.h>
2 #define max 100
3 typedef int ElementType;
4
5 typedef struct
6 {
7     int n, m;
8     int A[max][max];
9 } Graph;
10
11 void init_graph(Graph *pG, int n)
12 {
13     pG->n = n;
14     pG->m = 0;
15     for (int u = 1; u <= n; u++)
16     {
17         for (int v = 1; v <= n; v++)
18         {
19             pG->A[u][v] = 0;
20         }
21     }
22 }
23
24 void add_edge(Graph *pG, int u, int v)
25 {
26     pG->A[u][v] = 1;
27     pG->A[v][u] = 1;
28     pG->m++;
29 }
30
31 int adjacent(Graph *pG, int u, int v)
32 {
```



```

33     return pG->A[u][v] > 0;
34 }
35
36 /* kiểm tra đồ thị có chứa chu trình*/
37 #define WHITE 0
38 #define GRAY 1
39 #define BLACK 2
40
41 int color[max]; // lưu trạng thái của các đỉnh
42 int has_circle; // đồ thị chứa chu trình hay không
43
44
45 void DFS(Graph *pG, int u, int p) {
46     //1. Tô màu đang duyệt cho u
47     color[u] = GRAY;
48
49     //2. Xét các đỉnh kề của u
50     for (int v = 1; v <= pG->n; v++)
51         if (adjacent(pG, u, v)) {
52             if (v == p) //2a. Nếu v == p
53                 continue; //bỏ qua
54
55             if (color[v] == WHITE) //2a. Nếu v chưa duyệt
56                 DFS(pG, v, u); //gọi đệ quy duyệt nó
57             else if (color[v] == GRAY) //2b. v đang duyệt
58                 has_circle = 1; //chứa chu trình
59         }
60
61     //3. Tô màu duyệt xong cho u
62     color[u] = BLACK;
63 }
64
65
66 int main() {
67     Graph G;
68     int n, m, u, v;
69     scanf("%d%d", &n, &m);
70     init_graph(&G, n);
71     for (int e = 0; e < m; e++) {
72         scanf("%d%d", &u, &v);
73         add_edge(&G, u, v);
74     }
75
76     for (int u = 1; u <= G.n; u++)
77         color[u] = WHITE;
78     //2. Khởi tạo biến has_circle
79     has_circle = 0;
80     //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
81     for (int u = 1; u <= G.n; u++)
82         if (color[u] == WHITE) //u chưa duyệt
83             DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
84     //4. Kiểm tra has_circle
85
86     if (has_circle)
87         printf("CIRCLED\n");
88     else
89         printf("NO CIRCLE\n");
90
91     return 0;
92 }
93
94

```

Debug: source code from all test runs

Run 1

```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u, int p) {
    //1. Tô màu đang duyệt cho u
    color[u] = GRAY;

    //2. Xét các đỉnh kề của u
    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v)) {
            if (v == p) //2a. Nếu v == p
                continue; //bỏ qua

            if (color[v] == WHITE) //2a. Nếu v chưa duyệt
                DFS(pG, v, u); //gọi đệ quy duyệt nó
            else if (color[v] == GRAY) //2b. v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    //3. Tô màu duyệt xong cho u
    color[u] = BLACK;
}

int main() {
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);
    for (int e = 0; e < m; e++) {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    for (int u = 1; u <= G.n; u++)
        color[u] = WHITE;
    //2. Khởi tạo biến has_circle
    has_circle = 0;
    //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
    for (int u = 1; u <= G.n; u++)
        if (color[u] == WHITE) //u chưa duyệt
            DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
    //4. Kiểm tra has_circle

```

```
    if (has_circle)
        printf("CIRCLED\n");
    else
        printf("NO CIRCLE\n");

    return 0;
}
```

Run 2



```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u, int p) {
    //1. Tô màu đang duyệt cho u
    color[u] = GRAY;

    //2. Xét các đỉnh kề của u
    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v)) {
            if (v == p) //2a. Nếu v == p
                continue; //bỏ qua

            if (color[v] == WHITE) //2a. Nếu v chưa duyệt
                DFS(pG, v, u); //gọi đệ quy duyệt nó
            else if (color[v] == GRAY) //2b. v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    //3. Tô màu duyệt xong cho u
    color[u] = BLACK;
}

int main() {
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);
    for (int e = 0; e < m; e++) {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    for (int u = 1; u <= G.n; u++)
        color[u] = WHITE;
    //2. Khởi tạo biến has_circle
    has_circle = 0;
    //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
    for (int u = 1; u <= G.n; u++)
        if (color[u] == WHITE) //u chưa duyệt
            DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
    //4. Kiểm tra has_circle

```

```
    if (has_circle)
        printf("CIRCLED\n");
    else
        printf("NO CIRCLE\n");

    return 0;
}
```

Run 3



```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u, int p) {
    //1. Tô màu đang duyệt cho u
    color[u] = GRAY;

    //2. Xét các đỉnh kề của u
    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v)) {
            if (v == p) //2a. Nếu v == p
                continue; //bỏ qua

            if (color[v] == WHITE) //2a. Nếu v chưa duyệt
                DFS(pG, v, u); //gọi đệ quy duyệt nó
            else if (color[v] == GRAY) //2b. v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    //3. Tô màu duyệt xong cho u
    color[u] = BLACK;
}

int main() {
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);
    for (int e = 0; e < m; e++) {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    for (int u = 1; u <= G.n; u++)
        color[u] = WHITE;
    //2. Khởi tạo biến has_circle
    has_circle = 0;
    //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
    for (int u = 1; u <= G.n; u++)
        if (color[u] == WHITE) //u chưa duyệt
            DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
    //4. Kiểm tra has_circle

```



```
    if (has_circle)
        printf("CIRCLED\n");
    else
        printf("NO CIRCLE\n");

    return 0;
}
```

Run 4

```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u, int p) {
    //1. Tô màu đang duyệt cho u
    color[u] = GRAY;

    //2. Xét các đỉnh kề của u
    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v)) {
            if (v == p) //2a. Nếu v == p
                continue; //bỏ qua

            if (color[v] == WHITE) //2a. Nếu v chưa duyệt
                DFS(pG, v, u); //gọi đệ quy duyệt nó
            else if (color[v] == GRAY) //2b. v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    //3. Tô màu duyệt xong cho u
    color[u] = BLACK;
}

int main() {
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);
    for (int e = 0; e < m; e++) {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    for (int u = 1; u <= G.n; u++)
        color[u] = WHITE;
    //2. Khởi tạo biến has_circle
    has_circle = 0;
    //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
    for (int u = 1; u <= G.n; u++)
        if (color[u] == WHITE) //u chưa duyệt
            DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
    //4. Kiểm tra has_circle

```

```
    if (has_circle)
        printf("CIRCLED\n");
    else
        printf("NO CIRCLE\n");

    return 0;
}
```

Run 5

```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u, int p) {
    //1. Tô màu đang duyệt cho u
    color[u] = GRAY;

    //2. Xét các đỉnh kề của u
    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v)) {
            if (v == p) //2a. Nếu v == p
                continue; //bỏ qua

            if (color[v] == WHITE) //2a. Nếu v chưa duyệt
                DFS(pG, v, u); //gọi đệ quy duyệt nó
            else if (color[v] == GRAY) //2b. v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    //3. Tô màu duyệt xong cho u
    color[u] = BLACK;
}

int main() {
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);
    for (int e = 0; e < m; e++) {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    for (int u = 1; u <= G.n; u++)
        color[u] = WHITE;
    //2. Khởi tạo biến has_circle
    has_circle = 0;
    //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
    for (int u = 1; u <= G.n; u++)
        if (color[u] == WHITE) //u chưa duyệt
            DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
    //4. Kiểm tra has_circle

```

```
    if (has_circle)
        printf("CIRCLED\n");
    else
        printf("NO CIRCLE\n");

    return 0;
}
```

Run 6

```

#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} Graph;

void init_graph(Graph *pG, int n)
{
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
    {
        for (int v = 1; v <= n; v++)
        {
            pG->A[u][v] = 0;
        }
    }
}

void add_edge(Graph *pG, int u, int v)
{
    pG->A[u][v] = 1;
    pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max]; // lưu trạng thái của các đỉnh
int has_circle; // đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u, int p) {
    //1. Tô màu đang duyệt cho u
    color[u] = GRAY;

    //2. Xét các đỉnh kề của u
    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v)) {
            if (v == p) //2a. Nếu v == p
                continue; //bỏ qua

            if (color[v] == WHITE) //2a. Nếu v chưa duyệt
                DFS(pG, v, u); //gọi đệ quy duyệt nó
            else if (color[v] == GRAY) //2b. v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    //3. Tô màu duyệt xong cho u
    color[u] = BLACK;
}

int main() {
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);
    for (int e = 0; e < m; e++) {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }

    for (int u = 1; u <= G.n; u++)
        color[u] = WHITE;
    //2. Khởi tạo biến has_circle
    has_circle = 0;
    //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
    for (int u = 1; u <= G.n; u++)
        if (color[u] == WHITE) //u chưa duyệt
            DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
    //4. Kiểm tra has_circle

```

```

    if (has_circle)
        printf("CIRCLED\n");
    else
        printf("NO CIRCLE\n");

    return 0;
}

```

	Input	Expected	Got	
✓	3 2 1 3 3 2	NO CIRCLE	NO CIRCLE	✓
✓	7 9 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7	CIRCLED	CIRCLED	✓
✓	7 11 1 2 1 3 2 4 2 5 2 6 3 2 3 5 3 6 4 7 5 7 6 4	CIRCLED	CIRCLED	✓
✓	3 2 1 2 2 3	NO CIRCLE	NO CIRCLE	✓
✓	4 3 1 2 2 3 4 2	NO CIRCLE	NO CIRCLE	✓
✓	4 4 1 2 2 3 4 3 4 1	CIRCLED	CIRCLED	✓

Passed all tests! ✓

Question author's solution (C):

```

1  #include <stdio.h>
2
3  /* Khai báo CTDL Graph*/
4  #define MAX_N 100
5  typedef struct {
6      int n, m;
7      int A[MAX_N][MAX_N];
8  } Graph;
9
10 void init_graph(Graph *pG, int n) {
11     pG->n = n;
12     pG->m = 0;
13     for (int u = 1 ; u <= n; u++)
14         for (int v = 1 ; v <= n; v++)
15             pG->A[u][v] = 0;
16 }
17
18 void add_edge(Graph *pG, int u, int v) {
19     pG->A[u][v] += 1;
20     if (u != v)
21         pG->A[v][u] += 1;
22
23     if (pG->A[u][v] > 1)
24         printf("da cung (%d, %d)\n", u, v);
25     if (u == v)
26         printf("khuyen %d\n", u);
27
28     pG->m++;
29 }
30
31

```

```

32 int adjacent(Graph *pG, int u, int v) {
33     return pG->A[u][v] > 0;
34 }
35
36 #define WHITE 0
37 #define GRAY 1
38 #define BLACK 2
39
40 int color[MAX_N]; //Lưu trạng thái của các đỉnh
41 int has_circle; //Đồ thị chứa trình hay không
42
43 void DFS(Graph *pG, int u, int p) {
44     //1. Tô màu đang duyệt cho u
45     color[u] = GRAY;
46
47     //2. Xét các đỉnh kề của u
48     for (int v = 1; v <= pG->n; v++)
49         if (adjacent(pG, u, v)) {
50             if (v == p) //2a. Nếu v == p
51                 continue; //bỏ qua
52
53             if (color[v] == WHITE) //2a. Nếu v chưa duyệt
54                 DFS(pG, v, u); //gọi đệ quy duyệt nó
55             else if (color[v] == GRAY) //2b. v đang duyệt
56                 has_circle = 1; //chứa chu trình
57         }
58
59     //3. Tô màu duyệt xong cho u
60     color[u] = BLACK;
61 }
62
63
64 int main() {
65     //1. Khai báo đồ thị G
66     Graph G;
67     //2. Đọc dữ liệu và dựng đồ thị
68     int n, m, u, v;
69     scanf("%d%d", &n, &m);
70     init_graph(&G, n);
71     for (int e = 0; e < m; e++) {
72         scanf("%d%d", &u, &v);
73         add_edge(&G, u, v);
74     }
75
76     for (int u = 1; u <= G.n; u++)
77         color[u] = WHITE;
78     //2. Khởi tạo biến has_circle
79     has_circle = 0;
80     //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
81     for (int u = 1; u <= G.n; u++)
82         if (color[u] == WHITE) //u chưa duyệt
83             DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
84     //4. Kiểm tra has_circle
85
86     if (has_circle)
87         printf("CIRCLED\n");
88     else
89         printf("NO CIRCLE\n");
90
91     return 0;
92 }
93
94

```

Correct

Marks for this submission: 1.00/1.00.

Viết chương trình đọc vào một **đơn đồ thị vô hướng** và kiểm tra xem nó có chứa chu trình hay không. Nếu đồ thị có chu trình, in ra các đỉnh có trong chu trình.

Đầu vào (Input):

Dữ liệu đầu vào được nhập từ bàn phím với định dạng:

- Dòng đầu tiên chứa 2 số nguyên n và m, tương ứng là số đỉnh và số cung.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u, v mô tả cung (u, v).

Đầu ra (Output):

- Nếu đồ thị có chu trình, in ra các đỉnh có trong chu trình trên cùng 1 dòng, cách nhau 1 khoảng trắng. Đỉnh đầu và đỉnh cuối của chu trình giống nhau.
- Nếu đồ thị không chứa chu trình, in ra -1
- Nếu đồ thị có nhiều chu trình, chỉ cần in ra 1 chu trình bất kỳ.
- Xem thêm ví dụ bên dưới.

Trong ví dụ đầu tiên, ta tìm thấy chu trình 1 -> 3 -> 2 -> 1, vì thế chương trình phải in ra:

1 3 2 1

For example:

Input	Result
3 3 1 3 2 3 1 2	Chu trình hợp lệ.
7 9 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7	Chu trình hợp lệ.

Answer: (penalty regime: 10, 20, ... %)

```
1 #include <stdio.h>
2 #define max 100
3 typedef int ElementType;
4 int parent[max];
5 typedef struct
6 {
7     int n, m;
8     int A[max][max];
9 } Graph;
10
11 void init_graph(Graph *pG, int n)
12 {
13     pG->n = n;
14     pG->m = 0;
15     for (int u = 1; u <= n; u++)
16     {
17         for (int v = 1; v <= n; v++)
18         {
19             pG->A[u][v] = 0;
20         }
21     }
22 }
23
24 void add_edge(Graph *pG, int u, int v)
25 {
26     pG->A[u][v] = 1;
27     pG->A[v][u] = 1;
28     pG->m++;
29 }
30
31 int adjacent(Graph *pG, int u, int v)
32 {
33     return pG->A[u][v] > 0;
34 }
35
36 /* kiểm tra đồ thị có chứa chu trình*/
37 #define WHITE 0
38 #define GRAY 1
39 #define BLACK 2
40
41 int color[max]; // lưu trạng thái của các đỉnh
42 int has_circle; // đồ thị chứa chu trình hay không
43 int start, end;
44
45 void DFS(Graph *pG, int u, int p) {
46     color[u] = GRAY;
```



```

47     parent[u] = p;
48
49     for (int v = 1; v <= pG->n; v++)
50     {
51         if (adjacent(pG, u, v)) {
52             if (v == p)
53                 continue;
54
55             if (color[v] == WHITE) //2a. Nếu v chưa duyệt
56                 DFS(pG, v, u); //gọi đệ quy duyệt nó
57             else if (color[v] == GRAY) { //2b. v đang duyệt
58                 has_circle = 1; //chứa chu trình
59                 start = u;
60                 end = v;
61             }
62         }
63     }
64     color[u] = BLACK;
65 }
66
67 int main() {
68     Graph G;
69     int n, m, u, v;
70     scanf("%d%d", &n, &m);
71     init_graph(&G, n);
72     for (int e = 0; e < m; e++) {
73         scanf("%d%d", &u, &v);
74         add_edge(&G, u, v);
75     }
76
77     for (int u = 1; u <= G.n; u++)
78         color[u] = WHITE;
79     has_circle = 0;
80     for (int u = 1; u <= G.n; u++)
81         if (color[u] == WHITE) //u chưa duyệt
82             DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
83     //4. Kiểm tra has_circle
84
85     if (has_circle) {
86         int A[100], i = 0;
87         A[0] = start;
88         int u = start;
89         do {
90             u = parent[u];
91             i++;
92             A[i] = u;
93         } while (u != v);
94
95         for (int j = i; j >= 0; j--)
96             printf("%d ", A[j]);
97
98         printf("%d\n", A[i]);
99     } else
100         printf("-1\n");
101
102     return 0;
103 }
104
105

```

	Input	Expected	Got	
✓	3 3 1 3 2 3 1 2	Chu trình hợp lệ.	Chu trình hợp lệ.	✓
✓	7 9 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7	Chu trình hợp lệ.	Chu trình hợp lệ.	✓

	Input	Expected	Got	
✓	7 11 1 2 1 3 2 4 2 5 2 6 3 2 5 3 3 6 4 7 7 5 6 4	Chu trình hợp lệ.	Chu trình hợp lệ.	✓
✓	3 2 1 2 2 3	-1	-1	✓
✓	4 3 1 2 2 3 4 2	-1	-1	✓
✓	4 4 1 2 2 3 3 4 4 1	Chu trình hợp lệ.	Chu trình hợp lệ.	✓

Passed all tests! ✓

Question author's solution (Python3):

```

1 #include <stdio.h>
2
3 /* Khai báo CSDL Graph*/
4 #define MAX_N 100
5 typedef struct {
6     int n, m;
7     int A[MAX_N][MAX_N];
8 } Graph;
9
10 void init_graph(Graph *pG, int n) {
11     pG->n = n;
12     pG->m = 0;
13     for (int u = 1 ; u <= n; u++)
14         for (int v = 1 ; v <= n; v++)
15             pG->A[u][v] = 0;
16 }
17
18 void add_edge(Graph *pG, int u, int v) {
19     pG->A[u][v] += 1;
20     if (u != v)
21         pG->A[v][u] += 1;
22
23     if (pG->A[u][v] > 1)
24         printf("da cung (%d, %d)\n", u, v);
25     if (u == v)
26         printf("khuyen %d\n", u);
27
28     pG->m++;
29 }
30
31
32 int adjacent(Graph *pG, int u, int v) {
33     return pG->A[u][v] > 0;
34 }
35
36 #define WHITE 0
37 #define GRAY 1
38 #define BLACK 2
39
40 int color[MAX_N]; //Lưu trạng thái của các đỉnh
41 int parent[MAX_N]; //Lưu trạng thái của các đỉnh
42 int has_circle; //Đồ thị chứa trình hay không
43 int start, end;
44
45 void DFS(Graph *pG, int u, int p) {
46     //1. Tô màu đang duyệt cho u
47     color[u] = GRAY;
48     parent[u] = p;
49
50     //2. Xét các đỉnh kề của u
51     for (int v = 1; v <= pG->n; v++)
52         if (adjacent(pG, u, v)) {
53             if (v == p)
54                 continue;
55
56             if (color[v] == WHITE) //2a. Nếu v chưa duyệt
57                 DFS(pG, v, u); //gọi đệ quy duyệt nó

```

```

58         else if (color[v] == GRAY) { //2b. v đang duyệt
59             has_circle = 1; //chứa chu trình
60             start = u;
61             end = v;
62         }
63     }
64
65     //3. Tô màu duyệt xong cho u
66     color[u] = BLACK;
67 }
68
69
70 int main() {
71     //1. Khai báo đồ thị G
72     Graph G;
73     //2. Đọc dữ liệu và dựng đồ thị
74     int n, m, u, v;
75     scanf("%d%d", &n, &m);
76     init_graph(&G, n);
77     for (int e = 0; e < m; e++) {
78         scanf("%d%d", &u, &v);
79         add_edge(&G, u, v);
80     }
81
82     for (int u = 1; u <= G.n; u++)
83         color[u] = WHITE;
84     //2. Khởi tạo biến has_circle
85     has_circle = 0;
86     //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
87     for (int u = 1; u <= G.n; u++)
88         if (color[u] == WHITE) //u chưa duyệt
89             DFS(&G, u, -1); //gọi DFS(&G, u) để duyệt từ u
90     //4. Kiểm tra has_circle
91
92     if (has_circle) {
93         int A[100], i = 0;
94         A[0] = start;
95         int u = start;
96         do {
97             u = parent[u];
98             i++;
99             A[i] = u;
100         } while (u != v);
101
102         for (int j = i; j >= 0; j--)
103             printf("%d ", A[j]);
104
105         printf("%d\n", A[i]);
106     } else
107         printf("-1\n");
108
109     return 0;
110 }
111
112

```

Correct

Marks for this submission: 1.00/1.00.

◀ * Bài tập 9. Ứng dụng liên thông

Jump to...

* Bài tập 11 - Ứng dụng kiểm tra chu trình ▶

