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**%)

Question **1**

Correct

Mark 1.00 out of 1.00

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, ... %)

```c
#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 cùa 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 đề 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;
}
```

Debug: source code from all test runs

Run 1

```c
#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 cùa 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 đề 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 cùa 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 đề 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 3

```c
#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 cùa 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 đề 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 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 cùa 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 đề 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 5

```c
#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 cùa 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 đề 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 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 cùa 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 đề 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):

```c
#include <stdio.h>

/* Khai báo CTDL Graph*/
#define MAX_N 100
typedef struct {
    int n, m;
    int A[MAX_N][MAX_N];
} 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;
    //if (u != v)
    //    pG->A[v][u] += 1;

    if (pG->A[u][v] > 1)
        printf("da cung (%d, %d)\n", u, v);
    if (u == v)
        printf("khuyen %d\n", u);


    pG->m++;
}

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

#define WHITE 0
#define GRAY  1
#define BLACK 2

int color[MAX_N];       //Lưu trạng thái của các đỉnh
```

```c
int has_circle;     //Đồ thị chứa trình hay không

void DFS(Graph *pG, int u) {
    //1. Tô màu dang 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 (color[v] == WHITE)      //2a. Nếu v chưa duyệt
                DFS(pG, v);         //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() {
    //1. Khai báo đồ thị G
    Graph G;
    //2. Đọc dữ liệu và dựng đồ thị
    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);     //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;
}
```

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 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, ... %)

```c
#include <stdio.h>

/* Khai báo CTDL Graph*/
#define MAX_N 100
typedef struct {
    int n, m;
    int A[MAX_N][MAX_N];
} 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;
    //if (u != v)
    //    pG->A[v][u] += 1;

    if (pG->A[u][v] > 1)
        printf("da cung (%d, %d)\n", u, v);
    if (u == v)
        printf("khuyen %d\n", u);


    pG->m++;
}

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

#define WHITE 0
#define GRAY  1
#define BLACK 2

int color[MAX_N];       //Lưu trạng thái của các đỉnh
int parent[MAX_N];      //Lưu trạng thái của các đỉnh
int has_circle;     //Đồ thị chứa trình hay không
int start, end;

void DFS(Graph *pG, int u, int p) {
    //1. Tô màu đang duyệt cho u
```

```
    color[u] = GRAY;
    parent[u] = p;

    //2. Xét các đỉnh kề của u
    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, u);          //gọi đệ quy duyệt nó
            else if (color[v] == GRAY) { //2b. v đang duyệt
                has_circle = 1;     //chứa chu trình
                start = u;
                end = v;
            }
        }

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


int main() {
    //1. Khai báo đồ thị G
    Graph G;
    //2. Đọc dữ liệu và dựng đồ thị
    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) {
        int A[100], i = 0;
        A[0] = start;
        int u = start;
        do {
            u = parent[u];
            i++;
            A[i] = u;
        } while (u != v);

        for (int j = i; j >= 0; j--)
            printf("%d ", A[j]);

        printf("%d\n", A[i]);
    } else
        printf("-1\n");

    return 0;
}
```

	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):

```
#include <stdio.h>

/* Khai báo CTDL Graph*/
#define MAX_N 100
typedef struct {
    int n, m;
    int A[MAX_N][MAX_N];
} 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;
    //if (u != v)
    //    pG->A[v][u] += 1;

    if (pG->A[u][v] > 1)
        printf("da cung (%d, %d)\n", u, v);
    if (u == v)
        printf("khuyen %d\n", u);


    pG->m++;
}

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

#define WHITE 0
#define GRAY  1
#define BLACK 2

int color[MAX_N];       //Lưu trạng thái của các đỉnh
int parent[MAX_N];      //Lưu trạng thái của các đỉnh
int has_circle;     //Đồ thị chứa trình hay không
int start, end;

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

    //2. Xét các đỉnh kề của u
    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, u);          //gọi đệ quy duyệt nó
            else if (color[v] == GRAY) { //2b. v đang duyệt
                has_circle = 1;     //chứa chu trình
                start = u;
```

```
                end = v;
            }
        }

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


int main() {
    //1. Khai báo đồ thị G
    Graph G;
    //2. Đọc dữ liệu và dựng đồ thị
    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) {
        int A[100], i = 0;
        A[0] = start;
        int u = start;
        do {
            u = parent[u];
            i++;
            A[i] = u;
        } while (u != v);

        for (int j = i; j >= 0; j--)
            printf("%d ", A[j]);

        printf("%d\n", A[i]);
    } else
        printf("-1\n");

    return 0;
}
```

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 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, ... %)

```
#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 cùa các đỉnh
int has_circle;// đồ thị chứa chu trình hay không


void DFS(Graph *pG, int u, int p) {
    //1. Tô màu dang 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;
}
```

Debug: source code from all test runs

Run 1

```c
#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 cùa các đỉnh
int has_circle;// đồ thị chứa chu trình hay không


void DFS(Graph *pG, int u, int p) {
        //1. Tô màu dang 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

```c
#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 cùa các đỉnh
int has_circle;// đồ thị chứa chu trình hay không


void DFS(Graph *pG, int u, int p) {
        //1. Tô màu dang 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

```c
#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 cùa các đỉnh
int has_circle;// đồ thị chứa chu trình hay không


void DFS(Graph *pG, int u, int p) {
        //1. Tô màu dang 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

```c
#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 cùa các đỉnh
int has_circle;// đồ thị chứa chu trình hay không


void DFS(Graph *pG, int u, int p) {
        //1. Tô màu dang 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

```c
#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 cùa các đỉnh
int has_circle;// đồ thị chứa chu trình hay không


void DFS(Graph *pG, int u, int p) {
        //1. Tô màu dang 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

```c
#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 cùa các đỉnh
int has_circle;// đồ thị chứa chu trình hay không


void DFS(Graph *pG, int u, int p) {
        //1. Tô màu dang 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):

```c
#include <stdio.h>

/* Khai báo CTDL Graph*/
#define MAX_N 100
typedef struct {
    int n, m;
    int A[MAX_N][MAX_N];
} 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;
    if (u != v)
        pG->A[v][u] += 1;

    if (pG->A[u][v] > 1)
        printf("da cung (%d, %d)\n", u, v);
    if (u == v)
        printf("khuyen %d\n", u);


    pG->m++;
}

```

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

#define WHITE 0
#define GRAY  1
#define BLACK 2

int color[MAX_N];       //Lưu trạng thái của các đỉnh
int has_circle;     //Đồ thị chứa trình hay không

void DFS(Graph *pG, int u, int p) {
    //1. Tô màu dang 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() {
    //1. Khai báo đồ thị G
    Graph G;
    //2. Đọc dữ liệu và dựng đồ thị
    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;
}
```

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 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, ... %)

```c
#include <stdio.h>
#define max 100
typedef int ElementType;
int parent[max];
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
int start, end;

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

```
    parent[u] = p;

    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v)) {
            if (v == p)
                continue;

            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
                start = u;
                end = v;
            }
        }

    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)  //u chưa duyệt
            DFS(&G, u, -1);     //gọi DFS(&G, u) để duyệt từ u
    //4. Kiểm tra has_circle

    if (has_circle) {
        int A[100], i = 0;
        A[0] = start;
        int u = start;
        do {
            u = parent[u];
            i++;
            A[i] = u;
        } while (u != v);

        for (int j = i; j >= 0; j--)
            printf("%d ", A[j]);

        printf("%d\n", A[i]);
    } else
        printf("-1\n");

    return 0;
}
```

	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):

```
#include <stdio.h>

/* Khai báo CTDL Graph*/
#define MAX_N 100
typedef struct {
    int n, m;
    int A[MAX_N][MAX_N];
} 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;
    if (u != v)
        pG->A[v][u] += 1;

    if (pG->A[u][v] > 1)
        printf("da cung (%d, %d)\n", u, v);
    if (u == v)
        printf("khuyen %d\n", u);


    pG->m++;
}

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

#define WHITE 0
#define GRAY  1
#define BLACK 2

int color[MAX_N];       //Lưu trạng thái của các đỉnh
int parent[MAX_N];      //Lưu trạng thái của các đỉnh
int has_circle;     //Đồ thị chứa trình hay không
int start, end;

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

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

            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
                start = u;
                end = v;
            }
        }

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


int main() {
    //1. Khai báo đồ thị G
    Graph G;
    //2. Đọc dữ liệu và dựng đồ thị
    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) {
        int A[100], i = 0;
        A[0] = start;
        int u = start;
        do {
            u = parent[u];
            i++;
            A[i] = u;
        } while (u != v);

        for (int j = i; j >= 0; j--)
            printf("%d ", A[j]);

        printf("%d\n", A[i]);
    } else
        printf("-1\n");

    return 0;
}
```

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 ►