Started on	Friday, 13 June 2025, 4:40 PM
State	Finished
Completed on	Friday, 13 June 2025, 11:07 PM
Time taken	6 hours 27 mins
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Viết chương trình đọc một đồ thị **vô hướng** từ bàn phím và kiểm tra xem đồ thị có liên thông không.

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 liên thông in ra **CONNECTED**, ngược lại in ra **DISCONNECTED**.

Gợi ý

- Duyệt đồ thị từ một đỉnh. Sau khi duyệt, kiểm tra xem có đỉnh nào chưa được không.

Hướng dẫn đọc dữ liệu và chạy thử chương trình

- Để chạy thử chương trình, bạn nên tạo một tập tin **dt.txt** chứa đồ thị cần kiểm tra.
- Thêm dòng **freopen("dt.txt", "r", stdin);** vào ngay sau hàm main(). Khi nộp bài, nhớ gỡ bỏ dòng này ra.
- Có thể sử dụng đoạn chương trình đọc dữ liệu mẫu sau đây:

```
freopen("dt.txt", "r", stdin); //Khi nộp bài nhớ bỏ dòng này.
Graph G;
int n, m, u, v, w, e;
scanf("%d%d", &n, &m);
init_graph(&G, n);

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

For example:

Input	Result
4 3 2 1 1 3 2 4	CONNECTED

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

```
#include <stdio.h>
#define max 100

int mark[max];
int parent[max];
typedef struct{
    int n,m;
    int A[max][max];
}Graph;
typedef struct{
    int u,p;
}ElementType;
typedef struct{
    ElementType data[max];
    int front,rear;
}Queue;

//khoi tao hang doi
void make_null_queue (Queue *pQ){
    pQ->front = 0;
    pQ->rear = -1;
}

//dua ptu vao cuoi
void enqueue (Queue *pQ, ElementType u){
    pQ->rear++;
    pQ->data[pQ->rear] = u;
}

ElementType front (Queue *pQ){
```

```
    return pQ->data[pQ->front];
}

void dequeue (Queue *pQ){
    pQ->front++;
}

int empty (Queue *pQ){
    return pQ->front > pQ->rear;
}

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;
}


void BFS(Graph *pG, int s){
    Queue Q;
    make_null_queue (&Q);

    ElementType pair;
    pair.u = s;
    pair.p = -1;
    enqueue(&Q,pair);
    while (!empty(&Q)){
        ElementType pair = front(&Q);
        int u = pair.u;
        int p = pair.p;
        dequeue(&Q);

        if (mark[u] != 0){
            continue;
        }
        // printf ("%d\n", u);
        mark[u] = 1;
        parent[u]= p;
        //duyệt các cạnh kề của uu
        for (int v = 1; v <= pG->n; v++){
            if (adjacent(pG,u,v)){
                ElementType pair;
                pair.u =v;
                pair.p =u;
                enqueue (&Q,pair);
            }
        }
    }
}

//kiểm tra tính liên thông
int connected(Graph *pG){
    for(int u = 1; u < pG->n; u++)
    //khởi tạo tất cả đỉnh chưa duyệt
        mark[u] = 0;
    //2. duyệt đồ thị từ đỉnh bất kì
    BFS(pG,1);
    //3. kiểm tra xem có đỉnh nào chưa được duyệt không
    for(int u=1; u < pG->n; u++)
        if(mark[u] == 0)
            return 0;
    return 1;
}
int main (){
```

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

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

    for (int i = 1; i <= n; i++) {
        if (mark[i] == 0) {
            BFS(&G, i);

        }
    }

    if(connected(&G)==1)
        printf("CONNECTED");
    else
        printf("DISCONNECTED");
}
```

	Input	Expected	Got	
✔	4 3 2 1 1 3 2 4	CONNECTED	CONNECTED	✔
✔	4 2 1 2 3 4	DISCONNECTED	DISCONNECTED	✔
✔	4 2 1 4 2 3	DISCONNECTED	DISCONNECTED	✔
✔	13 16 1 4 1 2 1 12 2 4 3 7 4 6 4 7 5 8 5 9 6 7 6 13 8 9 10 11 10 12 11 12 5 6	CONNECTED	CONNECTED	✔

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;

```

```c
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;

    pG->m++;
}

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




//Biến hỗ trợ dùng để lưu trạng thái của đỉnh: đã duyệt/chưa duyệt
int mark[MAX_N];

void DFS(Graph *pG, int u) {
    //1. Đánh dấu u đã duyệt
    //printf("Duyet %d\n", u);  //Làm gì đó trên u
    mark[u] = 1;            //Đánh dấu nó đã duyệt

    //2. Xét các đỉnh kề của u
    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v) && mark[v] == 0) //Nếu v chưa duyệt
            DFS(pG, v);                 //gọi đệ quy duyệt nó

}

// Kiểm tra pG có liên thông không
int connected(Graph *pG) {
    //1. Khởi tạo tất cả đỉnh đều chưa duyệt
    for (int u = 1; u <= pG->n; u++)
        mark[u] = 0;
    //2. Duyệt đồ thị từ đỉnh bất kỳ, ví dụ: 1
    DFS(pG, 1);
    //3. Kiểm tra xem có đỉnh nào chưa duyệt không
    for (int u = 1; u <= pG->n; u++)
        if (mark[u] == 0)   //Vẫn còn đỉnh chưa duyệt
            return 0;   //Đồ thị không liên thông, thoát luôn

    return 1;           //Tất cả các đỉnh đều đã duyệt => liên thông
}


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);
    }

    //3. Khởi tạo mảng mark[u] = 0, với mọi u = 1, 2, ..., n
    for (int u = 1; u <= G.n; u++) {
        mark[u] = 0;
    }

    //4. Gọi hàm connected để kiểm tra

    if (connected(&G))
        printf("CONNECTED\n");
    else
        printf("DISCONNECTED\n");

```

```
88      return 0;
89  }
90
91
```

Correct

Marks for this submission: 1.00/1.00.

◄ * Bài tập 6. Xây dựng cây duyệt đồ thị

Jump to...

* Bài tập 8. Bộ phận liên thông ►