Started on	Sunday, 15 June 2025, 10:58 PM
State	Finished
Completed on	Sunday, 15 June 2025, 10:59 PM
Time taken	44 secs
Marks	2.00/2.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Viết chương trình đọc vào một đồ thị vô hướng và kiểm tra xem nó có phải là đồ thị phân đôi 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 của đồ thị.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u v, cách nhau 1 khoảng trắng, mô tả cung (u, v).

Đầu ra (Output)

- Nếu đồ thị là đồ thị phân đôi, in ra YES, ngược lại in ra NO.
- Xem thêm các ví dụ bên dưới.

For example:

Input	Result
3 2 1 2 2 3	YES
3 3 1 2 2 3 3 1	NO

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

```c
#include <stdio.h>

#define MAX 100

//---------------------Graph-------------------
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 i=0; i<=n;i++){
        for(int j=0; j<=n;j++){
            pG->A[i][j] = 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 adj(Graph* pG, int u, int v){
    return pG->A[u][v] > 0;
}

//---------Group-------
//----------------DFS--------------
#define NO_COLOR 0
#define RED 1
#define BLUE 2
#define SUM 3

int visited[MAX];
int color[MAX];
int is_conflict = 0;

void DFS(Graph* pG, int u, int color_u){
//  visited[u] = 1;
//  printf("%d ", u);
```

```
    for(int v=1; v<=pG->n; v++){

        //xet cac dinh ke cua u
        if(adj(pG,u,v)){
            int color_v = color[v];
            if(color_v==NO_COLOR){
                DFS(pG, v, SUM - color_u);
            } else if(color_v==color_u){
                is_conflict = 1;
            }
        }
    }
}

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

    init_graph(&G,n);

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

    for(int i = 0; i <= G.n; i++){
        visited[i] = 0;
        color[i] = NO_COLOR;
    }

    DFS(&G,1,BLUE);

    if(is_conflict){
        printf("NO");
    }else{
        printf("YES");
    }

}
```

	Input	Expected	Got	
✔	3 2 1 2 2 3	YES	YES	✔
✔	3 3 1 2 2 3 3 1	NO	NO	✔
✔	9 8 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9	YES	YES	✔

	Input	Expected	Got	
✔	8 9 1 2 1 3 1 4 1 5 2 7 1 8 5 7 3 6 4 6	YES	YES	✔

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 NO_COLOR 0
#define BLUE     1
#define RED      2

//Các biến hỗ trợ
int color[MAX_N];       //Lưu trạng thái của các đỉnh
int conflict;       //Có đụng độ trong khi tô màu không

//Tô màu các đỉnh của đồ thị bắt đầu từ đỉnh u với màu c
void colorize(Graph *pG, int u, int c) {
    //1. Tô màu c cho u
    color[u] = c;
    //2. Xét các đỉnh kề của u
    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v)) {
            if (color[v] == NO_COLOR)    //2a. Nếu v chưa có màu
                colorize(pG, v, 3 - c);  //tô màu nó ngược với c
            else if (color[v] == color[u]) //2b. u và v  cùng màu
                conflict = 1;            //đụng độ, không tô được
        }
}
```

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

    if (conflict)
        printf("NO\n");
    else
        printf("YES\n");

    return 0;
}
```

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

David là huấn luyện viên của một đội bóng gồm **N** thành viên. David muốn chia đội bóng thành hai nhóm. Để tăng tính đa dạng của các thành viên trong nhóm, David quyết định không xếp hai thành viên đã từng thi đấu với nhau vào chung một nhóm. Bạn hãy lập trình giúp David phân chia đội bóng.

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 thành viên và số cặp thành viên đã từng thi đấu với nhau.
- M dòng tiếp theo mỗi dòng chứa 2 số nguyên **u v** nói rằng 2 thành viên **u** và **v** đã từng thi đấu chung với nhau.

Đầu ra (Output):

- Nếu phân chia được, hãy in ra các thành viên của mỗi nhóm. Nhóm của thành viên 1 sẽ được in trước, nhóm còn lại in ra sau. Các thành viên trong nhóm được in ra theo thứ tự tăng dần và in trên 1 dòng. Hai thành viên cách nhau 1 khoảng trắng.
- Nếu không thể phân chia, in ra **IMPOSSIBLE**

Xem thêm các ví dụ bên dưới.

For example:

Input	Result
3 2 1 2 2 3	1 3 2
3 3 1 2 2 3 3 1	IMPOSSIBLE
9 8 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9	1 2 3 4 5 6 7 8 9

Answer: (penalty regime: 33.3, 66.7, ... %)

```
#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;
    pG->A[v][u] += 1;
    pG->m++;
}

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

#define NO_COLOR 0
#define BLUE     1
#define RED      2
```

```c
#define RED       2

int color[MAX_N];   // Lưu trạng thái màu của các đỉnh
int conflict = 0;   // Cờ kiểm tra đụng độ màu

void colorize(Graph *pG, int u, int c) {
    color[u] = c;
    for (int v = 1; v <= pG->n; v++) {
        if (adjacent(pG, u, v)) {
            if (color[v] == NO_COLOR)
                colorize(pG, v, 3 - c);
            else if (color[v] == color[u])
                conflict = 1;
        }
    }
}

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 i = 1; i <= G.n; i++)
        color[i] = NO_COLOR;
    conflict = 0;

    for (int i = 1; i <= G.n; i++)
        if (color[i] == NO_COLOR)
            colorize(&G, i, BLUE);

    if (conflict) {
        printf("IMPOSSIBLE\n");
    } else {
        // In nhóm BLUE trước (vì đỉnh 1 được tô màu BLUE từ đầu)
        for (int i = 1; i <= G.n; i++)
            if (color[i] == BLUE)
                printf("%d ", i);
        printf("\n");
        for (int i = 1; i <= G.n; i++)
            if (color[i] == RED)
                printf("%d ", i);
        printf("\n");
    }

    return 0;
}
```

Debug: source code from all test runs

Run 1

```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;
    pG->A[v][u] += 1;
    pG->m++;
}

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

#define NO_COLOR 0
#define BLUE     1
#define RED      2

int color[MAX_N];       // Lưu trạng thái màu của các đỉnh
int conflict = 0;       // Cờ kiểm tra đụng độ màu

void colorize(Graph *pG, int u, int c) {
    color[u] = c;
    for (int v = 1; v <= pG->n; v++) {
        if (adjacent(pG, u, v)) {
            if (color[v] == NO_COLOR)
                colorize(pG, v, 3 - c);
            else if (color[v] == color[u])
                conflict = 1;
        }
    }
}

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 i = 1; i <= G.n; i++)
        color[i] = NO_COLOR;
    conflict = 0;

    for (int i = 1; i <= G.n; i++)
        if (color[i] == NO_COLOR)
            colorize(&G, i, BLUE);

    if (conflict) {
        printf("IMPOSSIBLE\n");
    } else {
        // In nhóm BLUE trước (vì đỉnh 1 được tô màu BLUE từ đầu)
        for (int i = 1; i <= G.n; i++)
            if (color[i] == BLUE)
                printf("%d ", i);
```

```
        printf("\n");
        for (int i = 1; i <= G.n; i++)
            if (color[i] == RED)
                printf("%d ", i);
        printf("\n");
    }

    return 0;
}
```

Run 2

```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;
    pG->A[v][u] += 1;
    pG->m++;
}

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

#define NO_COLOR 0
#define BLUE     1
#define RED      2

int color[MAX_N];       // Lưu trạng thái màu của các đỉnh
int conflict = 0;       // Cờ kiểm tra đụng độ màu

void colorize(Graph *pG, int u, int c) {
    color[u] = c;
    for (int v = 1; v <= pG->n; v++) {
        if (adjacent(pG, u, v)) {
            if (color[v] == NO_COLOR)
                colorize(pG, v, 3 - c);
            else if (color[v] == color[u])
                conflict = 1;
        }
    }
}

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 i = 1; i <= G.n; i++)
        color[i] = NO_COLOR;
    conflict = 0;

    for (int i = 1; i <= G.n; i++)
        if (color[i] == NO_COLOR)
            colorize(&G, i, BLUE);

    if (conflict) {
        printf("IMPOSSIBLE\n");
    } else {
        // In nhóm BLUE trước (vì đỉnh 1 được tô màu BLUE từ đầu)
        for (int i = 1; i <= G.n; i++)
            if (color[i] == BLUE)
                printf("%d ", i);
```

```
        printf("\n");
        for (int i = 1; i <= G.n; i++)
            if (color[i] == RED)
                printf("%d ", i);
        printf("\n");
    }

    return 0;
}
```

Run 3

```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;
    pG->A[v][u] += 1;
    pG->m++;
}

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

#define NO_COLOR 0
#define BLUE     1
#define RED      2

int color[MAX_N];       // Lưu trạng thái màu của các đỉnh
int conflict = 0;       // Cờ kiểm tra đụng độ màu

void colorize(Graph *pG, int u, int c) {
    color[u] = c;
    for (int v = 1; v <= pG->n; v++) {
        if (adjacent(pG, u, v)) {
            if (color[v] == NO_COLOR)
                colorize(pG, v, 3 - c);
            else if (color[v] == color[u])
                conflict = 1;
        }
    }
}

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 i = 1; i <= G.n; i++)
        color[i] = NO_COLOR;
    conflict = 0;

    for (int i = 1; i <= G.n; i++)
        if (color[i] == NO_COLOR)
            colorize(&G, i, BLUE);

    if (conflict) {
        printf("IMPOSSIBLE\n");
    } else {
        // In nhóm BLUE trước (vì đỉnh 1 được tô màu BLUE từ đầu)
        for (int i = 1; i <= G.n; i++)
            if (color[i] == BLUE)
                printf("%d ", i);
```

```
        printf("\n");
        for (int i = 1; i <= G.n; i++)
            if (color[i] == RED)
                printf("%d ", i);
        printf("\n");
    }

    return 0;
}
```

	Input	Expected	Got	
✔	3 2 1 2 2 3	1 3 2	1 3 2	✔
✔	3 3 1 2 2 3 3 1	IMPOSSIBLE	IMPOSSIBLE	✔
✔	9 8 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

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

Jump to...

* Bài tập 13 - Tính liên thông mạnh ►