Started on	Sunday, 15 June 2025, 5:26 PM
State	Finished
Completed on	Sunday, 15 June 2025, 7:55 PM
Time taken	2 hours 29 mins
Marks	1.90/2.00
Grade	**9.50** out of 10.00 (**95**%)

Question **1**

Correct

Mark 0.90 out of 1.00

Có n hòn đảo và m cây cầu. Mỗi cây cầu bắt qua 2 hòn đảo. Một hôm chúa đảo tự hỏi là với các cây cầu hiện tại thì đứng ở một hòn đảo bất kỳ có thể nào đi đến được tất cả các hòn đảo khác mà không cần dùng đến thuyền hay không.

Hãy giúp chúa đảo viết chương trình kiểm tra.

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 đảo và số cây cầu.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u v nói rằng có 1 cây cầu bắt qua hai hòn đảo u và v.

Đầu ra (Output):

- Nếu có thể đi được in ra màn hình **YES**, ngược lại in ra **NO**.

For example:

Input	Result
4 3 1 2 2 3 3 4	YES

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

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

void DFS(Graph *pG, int u,int v)
{
    mark[u] = 1;
    for (int v = 1; v <= pG->n; v++)
        if (adjacent(pG, u, v) && mark[v] == 0)
            DFS(pG, v, u);
}

int connected(Graph *pG)
{
    for (int u = 1; u <= pG->n; u++)
        mark[u] = 0;
```

```
    DFS(pG,1,-1);
    for (int u = 1; u <= pG->n; u++)
        if (mark[u] == 0)
            return 0; // đồ thì ko liên thông
    return 1;         // đồ thị liên thông
}
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++)
        mark[i] = 0;

    if (connected(&G) == 1)
        printf("YES");
    else
        printf("NO");
    return 0;
}
```

	Input	Expected	Got	
✔	4 3 1 2 2 3 3 4	YES	YES	✔
✔	4 2 3 4 1 2	NO	NO	✔
✔	4 3 2 3 1 4	NO	NO	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00. Accounting for previous tries, this gives **0.90/1.00**.

Question **2**

Correct

Mark 1.00 out of 1.00

Tôn Ngộ Không là một trong các nhân vật chính của truyện Tây du ký. Khi còn ở Hoa Quả Sơn, Tôn Ngộ Không là vua của loài khỉ (Mỹ Hầu Vương). Hoa Quả Sơn có rất nhiều cây ăn trái, nên loài khỉ rất thích. Do đặc tính của mình, khỉ không thích đi mà chỉ thích chuyền từ cây này sang cây khác. Tuy nhiên, nếu khoảng cách giữa hai cây quá xa thi chúng không thể chuyền qua lại được.

Đường đường là vua của loài khỉ, Tôn Ngộ Không muốn vạch ra một kế hoạch hái trái cây trên tất cả các cây có trên Hoa Quả Sơn mà không cần phải nhảy xuống đất. Tôn Ngộ Không dự định sẽ bắt đầu leo lên một cây, hái trái của cây này, sau đó chuyền qua một cây kế tiếp hái trái của này và tiếp tục như thế cho đến khi tất cả các cây đều được hái trái. Một cây có thể được chuyền qua chuyền lại nhiều lần.

Hãy giúp Tôn Ngộ Không kiểm tra xem kế hoạch này có thể thực hiện được không.

Đầu vào (Input):

Giả sử số lượng cây ăn trái ở Hoa Quả Sơn là n cây và được đánh số từ 1 đến n.

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 cây và số cặp cây có thể chuyền qua lại.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u v, cách nhau 1 khoảng trắng, nói rằng có thể chuyền từ cây u sang cây v hoặc chuyền từ cây v sang cây u.

Đầu ra (Output):

- Nếu kế hoạch của Tôn Ngộ Không có thể thực hiện được **DUOC**, ngược lại in ra **KHONG**.

Xem thêm ví dụ bên dưới. Trong ví dụ đầu tiên, Tôn Ngộ Không bắt đầu từ cây 1, chuyền qua cây 2, sau đó chuyền ngược về 1, chuyền tiếp sang 3 và sau cùng là sang 4.

For example:

Input	Result
4 3 2 1 1 3 3 4	DUOC
4 2 1 2 3 4	KHONG

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

```c
#include <stdio.h>
#define max 100
typedef int ElementType;
int mark[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)
{
```

```
33      return pG->A[u][v] > 0;
34  }
35
36  // Dựng cây duyệt đồ thị theo chiều sâu dùng đệ quy
37  void DFS(Graph *pG, int u,int v)
38  {
39      // 1. đánh dấu u đã duyệt
40      mark[u] = 1;
41      // 2. xét các đỉnh kề của u
42      for (int v = 1; v <= pG->n; v++)
43          if (adjacent(pG, u, v) && mark[v] == 0)
44              DFS(pG, v, u);
45  }
46
47  int connected(Graph *pG)
48  {
49      for (int u = 1; u <= pG->n; u++)
50          mark[u] = 0;
51      DFS(pG,1,-1);
52      for (int u = 1; u <= pG->n; u++)
53          if (mark[u] == 0)
54              return 0; // đồ thì ko liên thông
55      return 1;         // đồ thị liên thông
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 i = 1; i <= G.n; i++)
70          mark[i] = 0;
71
72      if (connected(&G) == 1)
73          printf("DUOC");
74      else
75          printf("KHONG");
76      return 0;
77  }
78
```

	Input	Expected	Got	
✔	4 3 2 1 1 3 3 4	DUOC	DUOC	✔
✔	4 2 1 2 3 4	KHONG	KHONG	✔
✔	4 3 1 4 2 3	KHONG	KHONG	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ * Bài tập 8. Bộ phận liên thông

Jump to...

* Bài tập 10. Kiểm tra đồ thị chứa chu trình ►