

Started on	Friday, 13 June 2025, 11:07 PM
State	Finished
Completed on	Sunday, 15 June 2025, 5:26 PM
Time taken	1 day 18 hours
Marks	3.60/4.00
Grade	9.00 out of 10.00 (90%)

Viết chương trình đọc một đồ thị **vô hướng** từ bàn phím và đếm số bộ phận liên thông của đồ thị.

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 số bộ phận liên thông của đồ thị.

Gợi ý

- Xem tài liệu thực hành

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	1
2 1	
1 3	
2 4	

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

```
1 #include <stdio.h>
2 #define max 100
3 typedef int ElementType;
4 int mark[max];
5 typedef struct{
6     int n,m;
7     int A[max][max];
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 }
19
20 void add_edge(Graph *pG, int u, int v){
21     pG->A[u][v] = 1;
22     pG->A[v][u] = 1;
23     pG->m++;
24 }
25
26 int adjacent(Graph *pG, int u, int v){
27     return pG->A[u][v] > 0;
28 }
29
30 typedef struct{
31     ElementType data[max];
32     int top_idx;
33 }Stack;
34
35 void make_null_stack(Stack *pS){
36     pS->top_idx = -1;
37 }
38
39 void push(Stack *pS, ElementType u){
40     pS->top_idx++;
41     pS->data[pS->top_idx] = u;
42 }
43
44 ElementType top(Stack *pS){
45     return pS->data[pS->top_idx];
46 }
```



```

46 }
47
48 void pop(Stack *pS){
49     pS->top_idx--;
50 }
51
52 int empty(Stack *pS){
53     return pS->top_idx == -1;
54 }
55
56 void DFS(Graph *pG, int s){
57     Stack S;
58     make_null_stack(&S);
59
60     push(&S, s);
61
62     while (!empty(&S)){
63         int u = top(&S);
64         pop(&S);
65         if(mark[u] != 0)
66             continue;
67         mark[u] = 1;
68
69         for(int v=1; v <= pG->n; v++){
70             if(adjacent(pG,u,v)&& mark[v] == 0)
71                 push(&S,v);
72         }
73     }
74 }
75 int main (){
76     Graph G;
77     int n, m, u, v;
78     scanf("%d%d", &n, &m);
79     init_graph(&G, n);
80
81     for (int e = 0; e < m; e++) {
82         scanf("%d%d", &u, &v);
83         add_edge(&G, u, v);
84     }
85     for (int u = 1; u <= G.n; u++)
86         mark[u] = 0;
87     int cnt = 0;
88     for(int u = 1; u <= G.n; u++){
89         if(mark[u]==0){
90             DFS(&G,u);
91             cnt++;
92         }
93     }
94     printf("%d",cnt);
95     return 0;
96 }
97

```

	Input	Expected	Got	
✓	4 3 2 1 1 3 2 4	1	1	✓
✓	4 2 1 2 3 4	2	2	✓
✓	4 2 1 4 2 3	2	2	✓
✓	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	1	1	✓

	Input	Expected	Got	
✓	13 15 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	2	2	✓
✓	13 13 1 2 1 12 3 7 4 6 4 7 5 8 5 9 6 7 6 13 8 9 10 11 10 12 11 12	3	3	✓

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     pG->m++;
24 }
25
26 int adjacent(Graph *pG, int u, int v) {
27     return pG->A[u][v] > 0;
28 }
29
30
31
32
33 //Biến hỗ trợ dùng để lưu trạng thái của đỉnh: đã duyệt/chưa duyệt
34 int mark[MAX_N];
35
36 void DFS(Graph *pG, int u) {
37     //1. Đánh dấu u đã duyệt
38     //printf("Duyet %d\n", u); //Làm gì đó trên u
39     mark[u] = 1; //Đánh dấu nó đã duyệt
40
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) //Nếu v chưa duyệt
44             DFS(pG, v); //gọi đệ quy duyệt nó
45
46 }
47
48 int main() {
49     //1. Khai báo đồ thị G
50     Graph G;
51     //2. Đọc dữ liệu và dựng đồ thị
52     int n, m, u, v;

```

```

53 scanf("%d", &n, &m);
54 init_graph(&G, n);
55 for (int e = 0; e < m; e++) {
56     scanf("%d", &u, &v);
57     add_edge(&G, u, v);
58 }
59
60 //3. Khởi tạo mảng mark[u] = 0, với mọi u = 1, 2, ..., n
61 for (int u = 1; u <= G.n; u++) {
62     mark[u] = 0;
63 }
64
65 //4. Duyệt toàn bộ đồ thị
66 int cnt = 0;
67 for (int u = 1; u <= G.n; u++)
68     if (mark[u] == 0) {
69         DFS(&G, u);
70         cnt++;
71     }
72
73 printf("%d\n", cnt);
74
75 return 0;
76 }
77

```

Correct

Marks for this submission: 1.00/1.00.

Viết chương trình đọc một đồ thị **vô hướng** từ bàn phím và đếm số đỉnh của bộ phận liên thông của đỉnh 1.

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 số đỉnh của bộ phận liên thông của đỉnh 1.

Gợi ý

- Xem tài liệu thực hành.

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	4
2 1	
1 3	
2 4	

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

```
1 #include <stdio.h>
2 #define max 100
3 typedef int ElementType;
4 int mark[max];
5 int nb_u;
6 typedef struct{
7     int n,m;
8     int A[max][max];
9 }Graph;
10
11 void init_graph (Graph *pG, int n){
12     pG->n = n;
13     pG->m = 0;
14     for (int u = 1; u <= n; u++){
15         for (int v = 1; v <= n; v++){
16             pG->A[u][v] = 0;
17         }
18     }
19 }
20
21 void add_edge(Graph *pG, int u, int v){
22     pG->A[u][v] = 1;
23     pG->A[v][u] = 1;
24     pG->m++;
25 }
26
27 int adjacent(Graph *pG, int u, int v){
28     return pG->A[u][v] > 0;
29 }
30
31 typedef struct{
32     ElementType data[max];
33     int top_idx;
34 }Stack;
35
36 void make_null_stack(Stack *pS){
37     pS->top_idx = -1;
38 }
39
40 void push(Stack *pS, ElementType u){
41     pS->top_idx++;
42     pS->data[pS->top_idx] = u;
43 }
44
45 ElementType top(Stack *pS){
46     return pS->data[pS->top_idx];
47 }
```



```

46     return ps->data[ps->top_idx];
47 }
48
49 void pop(Stack *pS){
50     pS->top_idx--;
51 }
52
53 int empty(Stack *pS){
54     return pS->top_idx == -1;
55 }
56
57 void DFS(Graph *pG, int s){
58     Stack S;
59     make_null_stack(&S);
60
61     push(&S, s);
62
63     while (!empty(&S)){
64         int u = top(&S);
65         pop(&S);
66         if(mark[u] != 0)
67             continue;
68
69         mark[u] = 1;
70         nb_u ++;
71
72         for(int v=1; v <= pG->n; v++){
73             if(adjacent(pG,u,v)&& mark[v] == 0)
74                 push(&S,v);
75         }
76     }
77 }
78 int main (){
79     Graph G;
80     int n, m, u, v;
81     scanf("%d", &n, &m);
82     init_graph(&G, n);
83
84     for (int e = 0; e < m; e++) {
85         scanf("%d", &u, &v);
86         add_edge(&G, u, v);
87     }
88     for (int u = 1; u <= G.n; u++)
89         mark[u] = 0;
90
91     nb_u = 0;
92     DFS(&G,1);
93     printf("%d",nb_u);
94     return 0;
95 }
96

```

	Input	Expected	Got	
✓	4 3 2 1 1 3 2 4	4	4	✓
✓	4 2 1 2 3 4	2	2	✓
✓	4 2 1 4 2 3	2	2	✓
✓	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	13	13	✓

	Input	Expected	Got	
✓	13 15 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	10	10	✓
✓	13 13 1 2 1 12 3 7 4 6 4 7 5 8 5 9 6 7 6 13 8 9 10 11 10 12 11 12	5	5	✓

Passed all tests! ✓

Question author's solution (C):

```

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     pG->m++;
24 }
25
26 int adjacent(Graph *pG, int u, int v) {
27     return pG->A[u][v] > 0;
28 }
29
30
31
32
33 //Biến hỗ trợ dùng để lưu trạng thái của đỉnh: đã duyệt/chưa duyệt
34 int mark[MAX_N];
35 int nb_u;
36
37 void DFS(Graph *pG, int u) {
38     //1. Đánh dấu u đã duyệt
39     //printf("Duyệt %d\n", u); //Làm gì đó trên u
40     mark[u] = 1; //Đánh dấu nó đã duyệt
41     nb_u++;
42
43     //2. Xét các đỉnh kề của u
44     for (int v = 1; v <= pG->n; v++)
45         if (adjacent(pG, u, v) && mark[v] == 0) //Nếu v chưa duyệt
46             DFS(pG, v); //gọi đệ quy duyệt nó
47 }
48
49
50 int main() {
51     //1. Khai báo đồ thị G
52     Graph G;

```




```

53 //2. Đọc dữ liệu và dựng đồ thị
54 int n, m, u, v;
55 scanf("%d%d", &n, &m);
56 init_graph(&G, n);
57 for (int e = 0; e < m; e++) {
58     scanf("%d%d", &u, &v);
59     add_edge(&G, u, v);
60 }
61
62 //3. Khởi tạo mảng mark[u] = 0, với mọi u = 1, 2, ..., n
63 for (int u = 1; u <= G.n; u++) {
64     mark[u] = 0;
65 }
66
67 //4. Duyệt đồ thị từ đỉnh 1
68 nb_u = 0;
69 DFS(&G, 1);
70
71 printf("%d\n", nb_u);
72
73 return 0;
74 }
75
76

```

Correct

Marks for this submission: 1.00/1.00.

Viết chương trình đọc một đồ thị vô hướng từ bàn phím và đếm số đỉnh của bộ phận liên thông của đỉnh **s**. Đỉnh **s** được đọc từ bàn phím.

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**).
- Dòng cuối cùng chứa đỉnh **s**.

Đầu ra (Output)

- In ra số đỉnh của bộ phận liên thông của đỉnh **s**.

Gợi ý

- Xem tài liệu thực hành.

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 3	4

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

```
1  #include <stdio.h>
2  #define max 100
3  typedef int ElementType;
4  int mark[max];
5  int nb_u;
6  typedef struct{
7      int n,m;
8      int A[max][max];
9  }Graph;
10
11 void init_graph (Graph *pG, int n){
12     pG->n = n;
13     pG->m = 0;
14     for (int u = 1; u <= n; u++){
15         for (int v = 1; v <= n; v++){
16             pG->A[u][v] = 0;
17         }
18     }
19 }
20
21 void add_edge(Graph *pG, int u, int v){
22     pG->A[u][v] = 1;
23     pG->A[v][u] = 1;
24     pG->m++;
25 }
26
27 int adjacent(Graph *pG, int u, int v){
28     return pG->A[u][v] > 0;
29 }
30
31 typedef struct{
32     ElementType data[max];
33     int top_idx;
34 }Stack;
35
36 void make_null_stack(Stack *pS){
37     pS->top_idx = -1;
38 }
39
40 void push(Stack *pS, ElementType u){
41     pS->top_idx++;
42     pS->data[pS->top_idx] = u;
43 }
```



```

44
45 ElementType top(Stack *pS){
46     return pS->data[pS->top_idx];
47 }
48
49 void pop(Stack *pS){
50     pS->top_idx--;
51 }
52
53 int empty(Stack *pS){
54     return pS->top_idx == -1;
55 }
56
57 void DFS(Graph *pG, int s){
58     Stack S;
59     make_null_stack(&S);
60
61     push(&S, s);
62
63     while (!empty(&S)){
64         int u = top(&S);
65         pop(&S);
66         if(mark[u] != 0)
67             continue;
68
69         mark[u] = 1;
70         nb_u ++;
71
72         for(int v=1; v <= pG->n; v++){
73             if(adjacent(pG,u,v)&& mark[v] == 0)
74                 push(&S,v);
75         }
76     }
77 }
78 int main (){
79     Graph G;
80     int n, m, u, v;
81     scanf("%d%d", &n, &m);
82     init_graph(&G, n);
83
84     for (int e = 0; e < m; e++) {
85         scanf("%d%d", &u, &v);
86         add_edge(&G, u, v);
87     }
88     for (int u = 1; u <= G.n; u++)
89         mark[u] = 0;
90
91     nb_u =0;
92     int s;
93     scanf("%d",&s);
94     DFS(&G,s);
95     printf("%d",nb_u);
96     return 0;
97 }
98

```

	Input	Expected	Got	
✓	4 3 2 1 1 3 2 4 3	4	4	✓
✓	4 2 1 2 3 4 3	2	2	✓
✓	4 3 1 2 2 3 3 1 3	3	3	✓

	Input	Expected	Got	
✓	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 5	13	13	✓
✓	13 15 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 6	10	10	✓
✓	13 13 1 2 1 12 3 7 4 6 4 7 5 8 5 9 6 7 6 13 8 9 10 11 10 12 11 12 5	3	3	✓
✓	13 14 1 2 1 12 3 7 3 12 4 6 4 7 5 8 5 9 6 7 6 13 8 9 10 11 10 12 11 12 2	10	10	✓

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     pG->m++;
24 }
25
26 int adjacent(Graph *pG, int u, int v) {
27     return pG->A[u][v] > 0;
28 }
29
30
31
32
33 //Biến hỗ trợ dùng để lưu trạng thái của đỉnh: đã duyệt/chưa duyệt
34 int mark[MAX_N];
35 int nb_u;
36
37 void DFS(Graph *pG, int u) {
38     //1. Đánh dấu u đã duyệt
39     //printf("Duyet %d\n", u); //Làm gì đó trên u
40     mark[u] = 1;           //Đánh dấu nó đã duyệt
41     nb_u++;
42
43     //2. Xét các đỉnh kề của u
44     for (int v = 1; v <= pG->n; v++)
45         if (adjacent(pG, u, v) && mark[v] == 0) //Nếu v chưa duyệt
46             DFS(pG, v);           //gọi đệ quy duyệt nó
47 }
48
49
50 int main() {
51     //1. Khai báo đồ thị G
52     Graph G;
53     //2. Đọc dữ liệu và dựng đồ thị
54     int n, m, u, v, s;
55     scanf("%d%d", &n, &m);
56     init_graph(&G, n);
57     for (int e = 0; e < m; e++) {
58         scanf("%d%d", &u, &v);
59         add_edge(&G, u, v);
60     }
61
62     scanf("%d", &s);
63     //3. Khởi tạo mảng mark[u] = 0, với mọi u = 1, 2, ..., n
64     for (int u = 1; u <= G.n; u++) {
65         mark[u] = 0;
66     }
67
68     //4. Duyệt đồ thị từ đỉnh 1
69     nb_u = 0;
70     DFS(&G, s);
71
72     printf("%d\n", nb_u);
73
74     return 0;
75 }
76
77

```

Correct

Marks for this submission: 1.00/1.00.

//



Viết chương trình đọc một đồ thị vô hướng từ bàn phím, tìm bộ phận liên thông có nhiều đỉnh nhất.

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 số đỉnh của bộ phận liên thông có nhiều đỉnh nhất.

Gợi ý

- Xem tài liệu thực hành.

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	4
2 1	
1 3	
2 4	

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

```
1 #include <stdio.h>
2 #define max 100
3 typedef int ElementType;
4 int mark[max];
5 int nb_u;
6 typedef struct{
7     int n,m;
8     int A[max][max];
9 }Graph;
10
11 void init_graph (Graph *pG, int n){
12     pG->n = n;
13     pG->m = 0;
14     for (int u = 1; u <= n; u++){
15         for (int v = 1; v <= n; v++){
16             pG->A[u][v] = 0;
17         }
18     }
19 }
20
21 void add_edge(Graph *pG, int u, int v){
22     pG->A[u][v] = 1;
23     pG->A[v][u] = 1;
24     pG->m++;
25 }
26
27 int adjacent(Graph *pG, int u, int v){
28     return pG->A[u][v] > 0;
29 }
30
31 typedef struct{
32     ElementType data[max];
33     int top_idx;
34 }Stack;
35
36 void make_null_stack(Stack *pS){
37     pS->top_idx = -1;
38 }
39
40 void push(Stack *pS, ElementType u){
41     pS->top_idx++;
42     pS->data[pS->top_idx] = u;
43 }
44
45 ElementType top(Stack *pS){
46     return pS->data[pS->top_idx];
47 }
```



```

46     return ps->data[ps->top_idx];
47 }
48
49 void pop(Stack *pS){
50     pS->top_idx--;
51 }
52
53 int empty(Stack *pS){
54     return pS->top_idx == -1;
55 }
56
57 void DFS(Graph *pG, int s){
58     Stack S;
59     make_null_stack(&S);
60
61     push(&S, s);
62
63     while (!empty(&S)){
64         int u = top(&S);
65         pop(&S);
66         if(mark[u] != 0)
67             continue;
68         mark[u] = 1;
69         nb_u ++;
70         for(int v=1; v <= pG->n; v++){
71             if(adjacent(pG,u,v)&& mark[v] == 0)
72                 push(&S,v);
73         }
74     }
75 }
76 int main (){
77     Graph G;
78     int n, m, u, v;
79     scanf("%d%d", &n, &m);
80     init_graph(&G, n);
81
82     for (int e = 0; e < m; e++) {
83         scanf("%d%d", &u, &v);
84         add_edge(&G, u, v);
85     }
86     for (int u = 1; u <= G.n; u++)
87         mark[u] = 0;
88
89     int max_cnt = 0;
90     for(int u = 1; u <= G.n; u++)
91         if(mark[u]==0){
92             nb_u = 0;
93             DFS(&G,u);
94             if(nb_u > max_cnt)
95                 max_cnt = nb_u;
96         }
97
98     printf("%d",max_cnt);
99     return 0;
100 }
101

```

	Input	Expected	Got	
✓	4 3 2 1 1 3 2 4	4	4	✓
✓	4 2 1 2 3 4	2	2	✓
✓	4 3 1 2 2 3 3 1	3	3	✓

	Input	Expected	Got	
✓	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	13	13	✓
✓	13 15 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	10	10	✓
✓	13 13 1 2 1 12 3 7 4 6 4 7 5 8 5 9 6 7 6 13 8 9 10 11 10 12 11 12	5	5	✓
✓	13 14 1 2 1 12 3 7 3 12 4 6 4 7 5 8 5 9 6 7 6 13 8 9 10 11 10 12 11 12	10	10	✓

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 }
```




```

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     pG->m++;
24 }
25
26 int adjacent(Graph *pG, int u, int v) {
27     return pG->A[u][v] > 0;
28 }
29
30
31
32
33 //Biến hỗ trợ dùng để lưu trạng thái của đỉnh: đã duyệt/chưa duyệt
34 int mark[MAX_N];
35 int nb_u;
36
37 void DFS(Graph *pG, int u) {
38     //1. Đánh dấu u đã duyệt
39     //printf("Duyet %d\n", u); //Làm gì đó trên u
40     mark[u] = 1; //Đánh dấu nó đã duyệt
41     nb_u++;
42
43     //2. Xét các đỉnh kề của u
44     for (int v = 1; v <= pG->n; v++)
45         if (adjacent(pG, u, v) && mark[v] == 0) //Nếu v chưa duyệt
46             DFS(pG, v); //gọi đệ quy duyệt nó
47 }
48
49
50 int main() {
51     //1. Khai báo đồ thị G
52     Graph G;
53     //2. Đọc dữ liệu và dựng đồ thị
54     int n, m, u, v, s;
55     scanf("%d", &n, &m);
56     init_graph(&G, n);
57     for (int e = 0; e < m; e++) {
58         scanf("%d", &u, &v);
59         add_edge(&G, u, v);
60     }
61
62     scanf("%d", &s);
63     //3. Khởi tạo mảng mark[u] = 0, với mọi u = 1, 2, ..., n
64     for (int u = 1; u <= G.n; u++) {
65         mark[u] = 0;
66     }
67
68     //4. Duyệt đồ thị từ đỉnh 1
69     int max_cnt = 0;
70     for (int u = 1; u <= G.n; u++)
71         if (mark[u] == 0) {
72             nb_u = 0;
73             DFS(&G, u);
74             if (nb_u > max_cnt)
75                 max_cnt = nb_u;
76         }
77
78     printf("%d\n", max_cnt);
79
80     return 0;
81 }
82
83

```

Correct

Marks for this submission: 1.00/1.00. Accounting for previous tries, this gives **0.60/1.00**.

◀ * Bài tập 7. Kiểm tra đồ thị vô hướng liên thông

Jump to...

* Bài tập 9. Ứng dụng liên thông ▶

