

Started on	Wednesday, 11 June 2025, 10:29 AM
State	Finished
Completed on	Sunday, 15 June 2025, 11:04 PM
Time taken	4 days 12 hours
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 đồ thị có hướng và áp dụng thuật toán Tarjan lên nó.

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 n dòng. Ở dòng thứ u, in ra 2 số nguyên num[u] và min_num[u] cách nhau 1 khoảng trắng.
- Xem thêm ví dụ bên dưới.

For example:

Input	Result
5 7	1 1
1 2	2 1
2 3	3 1
3 1	4 3
2 4	5 3
3 4	
4 5	
5 3	
8 13	1 1
1 2	2 1
1 3	4 2
2 1	5 2
2 8	8 4
3 4	7 6
3 5	6 6
4 2	3 3
4 7	
4 8	
5 3	
5 7	
6 7	
7 6	

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

```
1 #include <stdio.h>
2
3 #define MAX_SIZE 100
4
5 //----- Stack-----
6 typedef int ElementType;
7
8 typedef struct {
9     ElementType data[MAX_SIZE];
10     int top_idx;
11 } Stack;
12
13 void init_stack(Stack* pS) {
14     pS->top_idx = -1;
15 }
16
17 int is_empty(Stack* pS) {
18     return pS->top_idx == -1;
19 }
20
21 void push(Stack* pS, int x) {
22     pS->top_idx++;
23     pS->data[pS->top_idx] = x;
24 }
25
26 void pop(Stack* pS) {
27     pS->top_idx--;
28 }
29
30 ElementType top(Stack* pS) {
31     return pS->data[pS->top_idx];
32 }
```



```

32 }
33
34 //-----Graph-----
35 typedef struct {
36     int n, m;
37     int A[MAX_SIZE][MAX_SIZE];
38 } Graph;
39
40 void init_graph(Graph* pG, int n) {
41     pG->n = n;
42     pG->m = 0;
43
44     for (int i = 0; i <= n; i++) {
45         for (int j = 0; j <= n; j++) {
46             pG->A[i][j] = 0;
47         }
48     }
49 }
50
51 void add_edge(Graph* pG, int u, int v) {
52     pG->A[u][v]++;
53     // if (u != v) {
54     //     pG->A[v][u]++;
55     // }
56
57     pG->m++;
58 }
59
60 int adj(Graph* pG, int u, int v) {
61     return pG->A[u][v] > 0;
62 }
63
64 //-----Strong connection-----
65 int num[MAX_SIZE];
66 int min_num[MAX_SIZE];
67 int k;
68 Stack S;
69 int on_stack[MAX_SIZE];
70
71 void SCC(Graph* pG, int u) {
72     // 1. Đánh số và đưa vào stack
73     num[u] = k;
74     min_num[u] = k;
75     k++;
76     push(&S, u);
77     on_stack[u] = 1;
78
79     // 2. Duyệt các đỉnh kề
80     for (int v = 1; v <= pG->n; v++) {
81         if (adj(pG, u, v)) {
82             // chưa duyệt
83             if (num[v] < 0) {
84                 SCC(pG, v);
85                 min_num[u] = min_num[v] < min_num[u] ? min_num[v] : min_num[u];
86             } else if (on_stack[v]) {
87                 min_num[u] = num[v] < min_num[u] ? num[v] : min_num[u];
88             }
89         }
90     }
91
92     // 3. so sánh num và min num
93     if (num[u] == min_num[u]) {
94         int w;
95         do {
96             w = top(&S);
97             pop(&S);
98             on_stack[w] = 0;
99         } while (w != u);
100     }
101 }
102
103 int main() {
104     int n, m;
105     scanf("%d %d", &n, &m);
106
107     Graph G;
108     init_graph(&G, n);
109
110     for (int i = 0; i < m; i++) {

```

```

110     int u, v;
111         scanf("%d%d", &u, &v);
112         add_edge(&G, u, v);
113     }
114
115     init_stack(&S);
116     k = 1;
117
118     // chưa duyệt
119     for (int i = 0; i <= n; i++) {
120         num[i] = -1;
121     }
122
123     for (int i = 1; i <= G.n; i++) {
124         if (num[i] < 0) {
125             SCC(&G, i);
126         }
127     }
128
129     for (int i = 1; i <= G.n; i++) {
130         printf("%d %d\n", num[i], min_num[i]);
131     }
132 }

```

	Input	Expected	Got	
✓	5 7 1 2 2 3 3 1 2 4 3 4 4 5 5 3	1 1 2 1 3 1 4 3 5 3	1 1 2 1 3 1 4 3 5 3	✓
✓	8 13 1 2 1 3 2 1 2 8 3 4 3 5 4 2 4 7 4 8 5 3 5 7 6 7 7 6	1 1 2 1 4 2 5 2 8 4 7 6 6 6 3 3	1 1 2 1 4 2 5 2 8 4 7 6 6 6 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

```



```

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     if (pG->A[u][v] > 1)
24         printf("da cung (%d, %d)\n", u, v);
25     if (u == v)
26         printf("khuyen %d\n", u);
27
28
29     pG->m++;
30 }
31
32 int adjacent(Graph *pG, int u, int v) {
33     return pG->A[u][v] > 0;
34 }
35
36
37 #define MAX_SIZE 100
38 typedef int ElementType;
39 typedef struct {
40     ElementType data[MAX_SIZE];
41     int top_idx;
42 } Stack;
43 /* Hàm khởi tạo ngăn xếp rỗng */
44 void make_null_stack(Stack *pS) {
45     pS->top_idx = -1;
46 }
47 /* Hàm thêm phần tử u vào đỉnh ngăn xếp */
48 void push(Stack *pS, ElementType u) {
49     pS->top_idx++;
50     pS->data[pS->top_idx] = u;
51 }
52 /* Hàm xem phần tử trên đỉnh ngăn xếp */
53 ElementType top(Stack *pS) {
54     return pS->data[pS->top_idx];
55 }
56 /* Hàm xóa bỏ phần tử trên đỉnh ngăn xếp */
57 void pop(Stack *pS) {
58     pS->top_idx--;
59 }
60 /* Hàm kiểm tra ngăn xếp rỗng */
61 int empty(Stack *pS) {
62     return pS->top_idx == -1;
63 }
64
65
66
67 int min(int a, int b) {
68     return a < b ? a : b;
69 }
70
71
72 int num[MAX_N], min_num[MAX_N];
73 int k;
74 Stack S;
75 int on_stack[MAX_N];
76 int nb_cnt;
77
78
79 //Duyệt đồ thị bắt đầu từ đỉnh u
80 void SCC(Graph *pG, int u) {
81     //1. Đánh số u, đưa u vào ngăn xếp S
82     num[u] = min_num[u] = k; k++;
83     push(&S, u);
84     on_stack[u] = 1;
85     //2. Xét các đỉnh kề của u
86     for (int v = 1; v <= pG->n; v++) {
87         if (adjacent(pG, u, v)) {
88             if (num[v] < 0) {
89                 SCC(pG, v);
90                 min_num[u] = min(min_num[u], min_num[v]);
91             } else if (on_stack[v])
92                 min_num[u] = min(min_num[u], num[v]);
93         }
94     }
95 }
96
97 /* Hàm kiểm tra các đỉnh là đỉnh đầu hay đỉnh cuối */

```

```

95 //3. Kiểm tra u có phải là đỉnh khớp
96 if (num[u] == min_num[u]) {
97     //printf("Tìm được BPLT mạnh, %d là đỉnh khớp.\n", u);
98     nb_cnt++;
99     int w;
100 do { //Lấy các đỉnh trong S ra cho đến khi gặp u
101     w = top(&S);
102     pop(&S);
103     on_stack[w] = 0;
104     //printf("Lay %d.\n", w);
105
106 } while (w != u);
107 }
108 }
109
110
111
112
113 int main() {
114     //1. Khai báo đồ thị G
115     Graph G;
116     //2. Đọc dữ liệu và dựng đồ thị
117     int n, m, u, v;
118     scanf("%d%d", &n, &m);
119     init_graph(&G, n);
120 for (int e = 0; e < m; e++) {
121     scanf("%d%d", &u, &v);
122     add_edge(&G, u, v);
123 }
124
125 for (int u = 1; u <= G.n; u++)
126     num[u] = -1;
127
128
129 //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
130 k = 1; //1b. Tất cả đều chưa duyệt
131 make_null_stack(&S); //1c. Làm rỗng ngăn xếp
132 //2. Duyệt toàn bộ đồ thị để tìm BPLT mạnh
133 nb_cnt = 0;
134 for (int u = 1; u <= G.n; u++)
135     if (num[u] == -1) //u chưa duyệt
136         SCC(&G, u); //duyet nó
137
138
139 for (int u = 1; u <= n; u++)
140     printf("%d %d\n", num[u], min_num[u]);
141
142     return 0;
143 }
144
145

```

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 đồ thị có hướng và kiểm tra xem nó có liên thông mạnh 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)

- In ra màn hình **STRONG CONNECTED** nếu đồ thị liên thông mạnh, ngược lại in ra **DISCONNECTED**.
- Xem thêm ví dụ bên dưới.

For example:

Input	Result
5 7 1 2 2 3 3 1 2 4 3 4 4 5 5 3	STRONG CONNECTED
8 13 1 2 1 3 2 1 2 8 3 4 3 5 4 2 4 7 4 8 5 3 5 7 6 7 7 6	DISCONNECTED

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

```
1 #include <stdio.h>
2
3 #define MAX_SIZE 100
4
5 //----- Stack-----
6 typedef int ElementType;
7
8 ▾ typedef struct {
9     ElementType data[MAX_SIZE];
10     int top_idx;
11 } Stack;
12
13 ▾ void init_stack(Stack* pS) {
14     pS->top_idx = -1;
15 }
16
17 ▾ int is_empty(Stack* pS) {
18     return pS->top_idx == -1;
19 }
20
21 ▾ void push(Stack* pS, int x) {
22     pS->top_idx++;
23     pS->data[pS->top_idx] = x;
24 }
25
26 ▾ void pop(Stack* pS) {
27     pS->top_idx--;
28 }
29
30 ▾ ElementType top(Stack* pS) {
31     return pS->data[pS->top_idx];
32 }
```



```

32 }
33
34 //-----Graph-----
35 typedef struct {
36     int n, m;
37     int A[MAX_SIZE][MAX_SIZE];
38 } Graph;
39
40 void init_graph(Graph* pG, int n) {
41     pG->n = n;
42     pG->m = 0;
43
44     for (int i = 0; i <= n; i++) {
45         for (int j = 0; j <= n; j++) {
46             pG->A[i][j] = 0;
47         }
48     }
49 }
50
51 void add_edge(Graph* pG, int u, int v) {
52     pG->A[u][v]++;
53     // if (u != v) {
54     //     pG->A[v][u]++;
55     // }
56
57     pG->m++;
58 }
59
60 int adj(Graph* pG, int u, int v) {
61     return pG->A[u][v] > 0;
62 }
63
64 //-----Strong connection-----
65 int num[MAX_SIZE];
66 int min_num[MAX_SIZE];
67 int k;
68 Stack S;
69 int on_stack[MAX_SIZE];
70 int cnt;
71
72 void SCC(Graph* pG, int u) {
73     // 1. Đánh số và đưa vào stack
74     num[u] = k;
75     min_num[u] = k;
76     k++;
77     push(&S, u);
78     on_stack[u] = 1;
79
80     // 2. Duyệt các đỉnh kề
81     for (int v = 1; v <= pG->n; v++) {
82         if (adj(pG, u, v)) {
83             // chưa duyệt
84             if (num[v] < 0) {
85                 SCC(pG, v);
86                 min_num[u] = min_num[v] < min_num[u] ? min_num[v] : min_num[u];
87             } else if (on_stack[v]) {
88                 min_num[u] = num[v] < min_num[u] ? num[v] : min_num[u];
89             }
90         }
91     }
92
93     // 3. so sánh num và min num
94     if (num[u] == min_num[u]) {
95         cnt++;
96
97         int w;
98         do {
99             w = top(&S);
100             pop(&S);
101             on_stack[w] = 0;
102         } while (w != u);
103     }
104 }
105
106 int main() {
107     int n, m;
108     scanf("%d %d", &n, &m);
109
110     Graph G;

```



```

110 init_graph(&G, n);
111
112 for (int i = 0; i < m; i++) {
113     int u, v;
114     scanf("%d%d", &u, &v);
115     add_edge(&G, u, v);
116 }
117
118 init_stack(&S);
119 k = 1;
120
121 // chưa duyệt
122 for (int i = 0; i <= n; i++) {
123     num[i] = -1;
124 }
125
126 cnt = 0;
127 for (int i = 1; i <= G.n; i++) {
128     if (num[i] < 0) {
129         SCC(&G, i);
130     }
131 }
132
133 if (cnt > 1) {
134     printf("DISCONNECTED");
135 } else {
136     printf("STRONG CONNECTED");
137 }
138 }

```

	Input	Expected	Got	
✓	5 7 1 2 2 3 3 1 2 4 3 4 4 5 5 3	STRONG CONNECTED	STRONG CONNECTED	✓
✓	8 13 1 2 1 3 2 1 2 8 3 4 3 5 4 2 4 7 4 8 5 3 5 7 6 7 7 6	DISCONNECTED	DISCONNECTED	✓

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     if (pG->A[u][v] > 1)
24         printf("da cung (%d, %d)\n", u, v);
25     if (u == v)
26         printf("khuyen %d\n", u);
27
28     pG->m++;
29 }
30
31 int adjacent(Graph *pG, int u, int v) {
32     return pG->A[u][v] > 0;
33 }
34
35
36
37 #define MAX_SIZE 100
38 typedef int ElementType;
39 typedef struct {
40     ElementType data[MAX_SIZE];
41     int top_idx;
42 } Stack;
43 /* Hàm khởi tạo ngăn xếp rỗng */
44 void make_null_stack(Stack *pS) {
45     pS->top_idx = -1;
46 }
47 /* Hàm thêm phần tử u vào đỉnh ngăn xếp */
48 void push(Stack *pS, ElementType u) {
49     pS->top_idx++;
50     pS->data[pS->top_idx] = u;
51 }
52 /* Hàm xem phần tử trên đỉnh ngăn xếp */
53 ElementType top(Stack *pS) {
54     return pS->data[pS->top_idx];
55 }
56 /* Hàm xóa bỏ phần tử trên đỉnh ngăn xếp */
57 void pop(Stack *pS) {
58     pS->top_idx--;
59 }
60 /* Hàm kiểm tra ngăn xếp rỗng */
61 int empty(Stack *pS) {
62     return pS->top_idx == -1;
63 }
64
65
66
67 int min(int a, int b) {
68     return a < b ? a : b;
69 }
70
71
72 int num[MAX_N], min_num[MAX_N];
73 int k;
74 Stack S;
75 int on_stack[MAX_N];
76 int nb_cnt;
77
78
79 //Duyệt đồ thị bắt đầu từ đỉnh u
80 void SCC(Graph *pG, int u) {
81     //1. Đánh số u, đưa u vào ngăn xếp S
82     num[u] = min_num[u] = k; k++;
83     push(&S, u);
84     on_stack[u] = 1;
85     //2. Xét các đỉnh kề của u
86     for (int v = 1; v <= pG->n; v++) {
87         if (adjacent(pG, u, v)) {

```

```

88     if (num[v] < 0) {
89         SCC(pG, v);
90         min_num[u] = min(min_num[u], min_num[v]);
91     } else if (on_stack[v])
92         min_num[u] = min(min_num[u], num[v]);
93     }
94 }
95 //3. Kiểm tra u có phải là đỉnh khớp
96 if (num[u] == min_num[u]) {
97     //printf("Tìm được BPLT mạnh, %d là đỉnh khớp.\n", u);
98     nb_cnt++;
99     int w;
100     do { //Lấy các đỉnh trong S ra cho đến khi gặp u
101         w = top(&S);
102         pop(&S);
103         on_stack[w] = 0;
104         //printf("Lay %d.\n", w);
105
106     } while (w != u);
107 }
108 }
109
110
111
112
113 int main() {
114     //1. Khai báo đồ thị G
115     Graph G;
116     //2. Đọc dữ liệu và dựng đồ thị
117     int n, m, u, v;
118     scanf("%d%d", &n, &m);
119     init_graph(&G, n);
120     for (int e = 0; e < m; e++) {
121         scanf("%d%d", &u, &v);
122         add_edge(&G, u, v);
123     }
124
125     for (int u = 1; u <= G.n; u++)
126         num[u] = -1;
127
128
129     //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
130     k = 1; //1b. Tất cả đều chưa duyệt
131     make_null_stack(&S); //1c. Làm rỗng ngăn xếp
132     //2. Duyệt toàn bộ đồ thị để tìm BPLT mạnh
133     nb_cnt = 0;
134     for (int u = 1; u <= G.n; u++)
135         if (num[u] == -1) //u chưa duyệt
136             SCC(&G, u); //duyet nó
137
138
139     if (nb_cnt == 1)
140         printf("STRONG CONNECTED\n");
141     else
142         printf("DISCONNECTED\n");
143
144     return 0;
145 }
146
147

```

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 đồ thị có hướng và đếm số bộ phận liên thông mạnh của nó.

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 số BPLT mạnh của đồ thị.
- Xem thêm ví dụ bên dưới.

For example:

Input	Result
5 7 1 2 2 3 3 1 2 4 3 4 4 5 5 3	1
8 13 1 2 1 3 2 1 2 8 3 4 3 5 4 2 4 7 4 8 5 3 5 7 6 7 7 6	3

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

```
1 #include <stdio.h>
2
3 #define MAX_SIZE 100
4
5 //----- Stack-----
6 typedef int ElementType;
7
8 typedef struct {
9     ElementType data[MAX_SIZE];
10     int top_idx;
11 } Stack;
12
13 void init_stack(Stack* pS) {
14     pS->top_idx = -1;
15 }
16
17 int is_empty(Stack* pS) {
18     return pS->top_idx == -1;
19 }
20
21 void push(Stack* pS, int x) {
22     pS->top_idx++;
23     pS->data[pS->top_idx] = x;
24 }
25
26 void pop(Stack* pS) {
27     pS->top_idx--;
28 }
29
30 ElementType top(Stack* pS) {
31     return pS->data[pS->top_idx];
32 }
```



```

32 }
33
34 //-----Graph-----
35 typedef struct {
36     int n, m;
37     int A[MAX_SIZE][MAX_SIZE];
38 } Graph;
39
40 void init_graph(Graph* pG, int n) {
41     pG->n = n;
42     pG->m = 0;
43
44     for (int i = 0; i <= n; i++) {
45         for (int j = 0; j <= n; j++) {
46             pG->A[i][j] = 0;
47         }
48     }
49 }
50
51 void add_edge(Graph* pG, int u, int v) {
52     pG->A[u][v]++;
53     // if (u != v) {
54     //     pG->A[v][u]++;
55     // }
56
57     pG->m++;
58 }
59
60 int adj(Graph* pG, int u, int v) {
61     return pG->A[u][v] > 0;
62 }
63
64 //-----Strong connection-----
65 int num[MAX_SIZE];
66 int min_num[MAX_SIZE];
67 int k;
68 Stack S;
69 int on_stack[MAX_SIZE];
70 int cnt;
71
72 void SCC(Graph* pG, int u) {
73     // 1. Đánh số và đưa vào stack
74     num[u] = k;
75     min_num[u] = k;
76     k++;
77     push(&S, u);
78     on_stack[u] = 1;
79
80     // 2. Duyệt các đỉnh kề
81     for (int v = 1; v <= pG->n; v++) {
82         if (adj(pG, u, v)) {
83             // chưa duyệt
84             if (num[v] < 0) {
85                 SCC(pG, v);
86                 min_num[u] = min_num[v] < min_num[u] ? min_num[v] : min_num[u];
87             } else if (on_stack[v]) {
88                 min_num[u] = num[v] < min_num[u] ? num[v] : min_num[u];
89             }
90         }
91     }
92
93     // 3. so sánh num và min num
94     if (num[u] == min_num[u]) {
95         cnt++;
96
97         int w;
98         do {
99             w = top(&S);
100             pop(&S);
101             on_stack[w] = 0;
102         } while (w != u);
103     }
104 }
105
106 int main() {
107     int n, m;
108     scanf("%d %d", &n, &m);
109
110     Graph G;

```

```

110 init_graph(&G, n);
111
112 for (int i = 0; i < m; i++) {
113     int u, v;
114     scanf("%d%d", &u, &v);
115     add_edge(&G, u, v);
116 }
117
118 init_stack(&S);
119 k = 1;
120
121 // chưa duyệt
122 for (int i = 0; i <= n; i++) {
123     num[i] = -1;
124 }
125
126 cnt = 0;
127 for (int i = 1; i <= G.n; i++) {
128     if (num[i] < 0) {
129         SCC(&G, i);
130     }
131 }
132
133 printf("%d", cnt);
134 }

```

	Input	Expected	Got	
✓	5 7 1 2 2 3 3 1 2 4 3 4 4 5 5 3	1	1	✓
✓	8 13 1 2 1 3 2 1 2 8 3 4 3 5 4 2 4 7 4 8 5 3 5 7 6 7 7 6	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++)

```



```

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     if (pG->A[u][v] > 1)
24         printf("da cung (%d, %d)\n", u, v);
25     if (u == v)
26         printf("khuyen %d\n", u);
27
28     pG->m++;
29 }
30
31 int adjacent(Graph *pG, int u, int v) {
32     return pG->A[u][v] > 0;
33 }
34
35
36
37 #define MAX_SIZE 100
38 typedef int ElementType;
39 typedef struct {
40     ElementType data[MAX_SIZE];
41     int top_idx;
42 } Stack;
43 /* Hàm khởi tạo ngăn xếp rỗng */
44 void make_null_stack(Stack *pS) {
45     pS->top_idx = -1;
46 }
47 /* Hàm thêm phần tử u vào đỉnh ngăn xếp */
48 void push(Stack *pS, ElementType u) {
49     pS->top_idx++;
50     pS->data[pS->top_idx] = u;
51 }
52 /* Hàm xem phần tử trên đỉnh ngăn xếp */
53 ElementType top(Stack *pS) {
54     return pS->data[pS->top_idx];
55 }
56 /* Hàm xóa bỏ phần tử trên đỉnh ngăn xếp */
57 void pop(Stack *pS) {
58     pS->top_idx--;
59 }
60 /* Hàm kiểm tra ngăn xếp rỗng */
61 int empty(Stack *pS) {
62     return pS->top_idx == -1;
63 }
64
65
66
67 int min(int a, int b) {
68     return a < b ? a : b;
69 }
70
71
72 int num[MAX_N], min_num[MAX_N];
73 int k;
74 Stack S;
75 int on_stack[MAX_N];
76 int nb_cnt;
77
78
79 //Duyệt đồ thị bắt đầu từ đỉnh u
80 void SCC(Graph *pG, int u) {
81     //1. Đánh số u, đưa u vào ngăn xếp S
82     num[u] = min_num[u] = k; k++;
83     push(&S, u);
84     on_stack[u] = 1;
85     //2. Xét các đỉnh kề của u
86     for (int v = 1; v <= pG->n; v++) {
87         if (adjacent(pG, u, v)) {
88             if (num[v] < 0) {
89                 SCC(pG, v);
90                 min_num[u] = min(min_num[u], min_num[v]);
91             } else if (on_stack[v])
92                 min_num[u] = min(min_num[u], num[v]);
93         }
94     }
95 }

```

```

92         min_num[u] = min(min_num[u], num[v]);
93     }
94 }
95 //3. Kiểm tra u có phải là đỉnh khớp
96 if (num[u] == min_num[u]) {
97     //printf("Tìm được BPLT mạnh, %d là đỉnh khớp.\n", u);
98     nb_cnt++;
99     int w;
100     do { //Lấy các đỉnh trong S ra cho đến khi gặp u
101         w = top(&S);
102         pop(&S);
103         on_stack[w] = 0;
104         //printf("Lay %d.\n", w);
105     } while (w != u);
106 }
107 }
108 }
109
110
111
112
113 int main() {
114     //1. Khai báo đồ thị G
115     Graph G;
116     //2. Đọc dữ liệu và dựng đồ thị
117     int n, m, u, v;
118     scanf("%d%d", &n, &m);
119     init_graph(&G, n);
120     for (int e = 0; e < m; e++) {
121         scanf("%d%d", &u, &v);
122         add_edge(&G, u, v);
123     }
124
125     for (int u = 1; u <= G.n; u++)
126         num[u] = -1;
127
128
129     //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
130     k = 1; //1b. Tất cả đều chưa duyệt
131     make_null_stack(&S); //1c. Làm rỗng ngăn xếp
132     //2. Duyệt toàn bộ đồ thị để tìm BPLT mạnh
133     nb_cnt = 0;
134     for (int u = 1; u <= G.n; u++)
135         if (num[u] == -1) //u chưa duyệt
136             SCC(&G, u); //duyet nó
137
138
139     printf("%d\n", nb_cnt);
140
141     return 0;
142 }
143
144

```

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 đồ thị có hướng và tìm bộ phận liên thông mạnh 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 màn hình số đỉnh của BPLT mạnh có nhiều đỉnh nhất.
- Xem thêm ví dụ bên dưới.

For example:

Input	Result
5 6 1 2 2 3 3 1 2 4 3 4 5 3	3
8 13 1 2 1 3 2 1 2 8 3 4 3 5 4 2 4 7 4 8 5 3 5 7 6 7 7 6	5

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

```
1  #include <stdio.h>
2
3  #define MAX_SIZE 100
4
5  //----- Stack-----
6  typedef int ElementType;
7
8  typedef struct {
9      ElementType data[MAX_SIZE];
10     int top_idx;
11 } Stack;
12
13 void init_stack(Stack* pS) {
14     pS->top_idx = -1;
15 }
16
17 int is_empty(Stack* pS) {
18     return pS->top_idx == -1;
19 }
20
21 void push(Stack* pS, int x) {
22     pS->top_idx++;
23     pS->data[pS->top_idx] = x;
24 }
25
26 void pop(Stack* pS) {
27     pS->top_idx--;
28 }
29
30 ElementType top(Stack* pS) {
31     return pS->data[pS->top_idx];
32 }
```



```

33 //-----Graph-----
34 typedef struct {
35     int n, m;
36     int A[MAX_SIZE][MAX_SIZE];
37 } Graph;
38
39
40 void init_graph(Graph* pG, int n) {
41     pG->n = n;
42     pG->m = 0;
43
44     for (int i = 0; i <= n; i++) {
45         for (int j = 0; j <= n; j++) {
46             pG->A[i][j] = 0;
47         }
48     }
49 }
50
51 void add_edge(Graph* pG, int u, int v) {
52     pG->A[u][v]++;
53     // if (u != v) {
54     //     pG->A[v][u]++;
55     // }
56
57     pG->m++;
58 }
59
60 int adj(Graph* pG, int u, int v) {
61     return pG->A[u][v] > 0;
62 }
63
64 //-----Strong connection-----
65 int num[MAX_SIZE];
66 int min_num[MAX_SIZE];
67 int k;
68 Stack S;
69 int on_stack[MAX_SIZE];
70 int cnt, max;
71
72 void SCC(Graph* pG, int u) {
73     // 1. Đánh số và đưa vào stack
74     num[u] = k;
75     min_num[u] = k;
76     k++;
77     push(&S, u);
78     on_stack[u] = 1;
79
80     // 2. Duyệt các đỉnh kề
81     for (int v = 1; v <= pG->n; v++) {
82         if (adj(pG, u, v)) {
83             // chưa duyệt
84             if (num[v] < 0) {
85                 SCC(pG, v);
86                 min_num[u] = min_num[v] < min_num[u] ? min_num[v] : min_num[u];
87             } else if (on_stack[v]) {
88                 min_num[u] = num[v] < min_num[u] ? num[v] : min_num[u];
89             }
90         }
91     }
92
93     // 3. so sánh num và min num
94     if (num[u] == min_num[u]) {
95         cnt = 0;
96         int w;
97         do {
98             cnt++;
99             w = top(&S);
100             pop(&S);
101             on_stack[w] = 0;
102         } while (w != u);
103         max = cnt > max ? cnt : max;
104     }
105 }
106
107 int main() {
108     int n, m;
109     scanf("%d %d", &n, &m);
110
111     Graph G;

```

```

111 init_graph(&G, n);
112
113 for (int i = 0; i < m; i++) {
114     int u, v;
115     scanf("%d%d", &u, &v);
116     add_edge(&G, u, v);
117 }
118
119 init_stack(&S);
120 k = 1;
121
122 // chưa duyệt
123 for (int i = 0; i <= n; i++) {
124     num[i] = -1;
125 }
126
127 cnt = 0;
128 max = cnt;
129 for (int i = 1; i <= G.n; i++) {
130     if (num[i] < 0) {
131         SCC(&G, i);
132     }
133 }
134
135 printf("%d", max);
136 }

```

	Input	Expected	Got	
✓	5 6 1 2 2 3 3 1 2 4 3 4 5 3	3	3	✓
✓	8 13 1 2 1 3 2 1 2 8 3 4 3 5 4 2 4 7 4 8 5 3 5 7 6 7 7 6	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 u = 1; u <= n; u++)
15         for (int v = 1; v <= n; v++)
16             pG->A[u][v] = 0;
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     if (pG->A[u][v] > 1)
24         printf("da cung (%d, %d)\n", u, v);
25     if (u == v)
26         printf("khuyen %d\n", u);
27
28
29     pG->m++;
30 }
31
32 int adjacent(Graph *pG, int u, int v) {
33     return pG->A[u][v] > 0;
34 }
35
36
37 #define MAX_SIZE 100
38 typedef int ElementType;
39 typedef struct {
40     ElementType data[MAX_SIZE];
41     int top_idx;
42 } Stack;
43 /* Hàm khởi tạo ngăn xếp rỗng */
44 void make_null_stack(Stack *pS) {
45     pS->top_idx = -1;
46 }
47 /* Hàm thêm phần tử u vào đỉnh ngăn xếp */
48 void push(Stack *pS, ElementType u) {
49     pS->top_idx++;
50     pS->data[pS->top_idx] = u;
51 }
52 /* Hàm xem phần tử trên đỉnh ngăn xếp */
53 ElementType top(Stack *pS) {
54     return pS->data[pS->top_idx];
55 }
56 /* Hàm xóa bỏ phần tử trên đỉnh ngăn xếp */
57 void pop(Stack *pS) {
58     pS->top_idx--;
59 }
60 /* Hàm kiểm tra ngăn xếp rỗng */
61 int empty(Stack *pS) {
62     return pS->top_idx == -1;
63 }
64
65
66
67 int min(int a, int b) {
68     return a < b ? a : b;
69 }
70
71
72 int num[MAX_N], min_num[MAX_N];
73 int k;
74 Stack S;
75 int on_stack[MAX_N];
76 int max_cnt;
77
78
79 //Duyệt đồ thị bắt đầu từ đỉnh u
80 void SCC(Graph *pG, int u) {
81     //1. Đánh số u, đưa u vào ngăn xếp S
82     num[u] = min_num[u] = k; k++;
83     push(&S, u);
84     on_stack[u] = 1;
85     //2. Xét các đỉnh kề của u
86     for (int v = 1; v <= pG->n; v++) {
87         if (adjacent(pG, u, v)) {
88             if (num[v] < 0) {
89                 SCC(pG, v);
90                 min_num[u] = min(min_num[u], min_num[v]);
91             } else if (on_stack[v]) {

```

```

91         } else if (on_stack[v])
92             min_num[u] = min(min_num[u], num[v]);
93     }
94 }
95 //3. Kiểm tra u có phải là đỉnh khớp
96 if (num[u] == min_num[u]) {
97     //printf("Tìm được BPLT mạnh, %d là đỉnh khớp.\n", u);
98     int nb_cnt = 0;
99     int w;
100     do { //Lấy các đỉnh trong S ra cho đến khi gặp u
101         w = top(&S);
102         pop(&S);
103         on_stack[w] = 0;
104         //printf("Lấy %d.\n", w);
105         nb_cnt++;
106     } while (w != u);
107     if (nb_cnt > max_cnt)
108         max_cnt = nb_cnt;
109 }
110 }
111 }
112
113
114
115
116 int main() {
117     //1. Khai báo đồ thị G
118     Graph G;
119     //2. Đọc dữ liệu và dựng đồ thị
120     int n, m, u, v;
121     scanf("%d%d", &n, &m);
122     init_graph(&G, n);
123     for (int e = 0; e < m; e++) {
124         scanf("%d%d", &u, &v);
125         add_edge(&G, u, v);
126     }
127
128     for (int u = 1; u <= G.n; u++)
129         num[u] = -1;
130
131
132     //3. Duyệt toàn bộ đồ thị để kiểm tra chu trình
133     k = 1; //1b. Tất cả đều chưa duyệt
134     make_null_stack(&S); //1c. Làm rỗng ngăn xếp
135     //2. Duyệt toàn bộ đồ thị để tìm BPLT mạnh
136     max_cnt = 0;
137     for (int u = 1; u <= G.n; u++)
138         if (num[u] == -1) //u chưa duyệt
139             SCC(&G, u); //duyet nó
140
141
142     printf("%d\n", max_cnt);
143
144     return 0;
145 }
146
147

```

Correct

Marks for this submission: 1.00/1.00.

◀ * Bài tập 12 - Kiểm tra đồ thị phân
đôi

Jump to...

* Bài tập 14 - Ứng dụng liên thông
mạnh ▶

