Câu hỏi **1**

Đúng

Đạt điểm 0,90 trên 1,00

Đặt cờ

Thuật toán Tarjan tìm các thành phần liên thông mạnh (Tarjan's strongly connected components algorithm) dựa trên thuật toán duyệt đồ thị theo chiều sâu (sử dụng đệ quy) kết hợp với đánh số các đỉnh và một ngăn xếp lưu trữ các thành phần liên thông mạnh (strongly connected components gọi tắt là SCC).

```
void SCC(int u) {
    //1. Đánh số cho đỉnh u, đưa u vào ngăn xếp
    num[u] = k; min_num[u] = k; k++;
    push(S, u); on_stack[u] = true;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u)
        if (v chưa duyệt) {
            //2a. Duyệt v và cập nhật lại min_num[u]
            SCC(v);
            min_num[u] = min(min_num[u], min_num[v]);
        } else if (v còn trên stack) {
            //2b. cập nhật lại min_num[u]
            min_num[u] = min(min_num[u], num[v]);
        } else {
            //2c. bỏ qua, không làm gì cả
        }

   //3. Kiểm tra num[u] == min_num[u]
   if (num[u] == min_num[u]) {
       //Lấy các đỉnh trong stack ra cho đến khi gặp u
       //Các đỉnh này thuộc về một thành phần liên thông
   }
}
```

Thuật toán trên gồm 3 bước chính:

1. Đánh số cho đỉnh u, đưa u vào ngăn xếp. Sau khi đánh số xong tăng k lên. Thông thường, k được khởi tạo bằng 1.
2. Xét các đỉnh kề v của u, có 3 trường hợp xảy ra:
   - v chưa duyệt => gọi đệ quy duyệt v và cập nhật lại min_num[u]
   - v duyệt rồi nhưng vẫn còn trên stack => cập nhật lại min_num[u]
   - v duyệt rồi và không còn trên stack => bỏ qua
3. Kiểm tra num[u] và min_num[u]: nếu bằng nhau thì u là đỉnh khớp (articulation vertex) hay đỉnh cắt (cut vertex)
   - Lần lượt lấy các đỉnh trong stack ra cho đến khi u được lấy ra. Các đỉnh được lấy ra này chính là thành phần liên thông mạnh chứa u.

Cho đồ thị **có hướng** gồm **8** đỉnh và **14** cung như bên như bên dưới. Hãy áp dụng thuật toán Tarjan để tìm các thành phần liên thông mạnh của G bắt đầu từ đỉnh **1**.

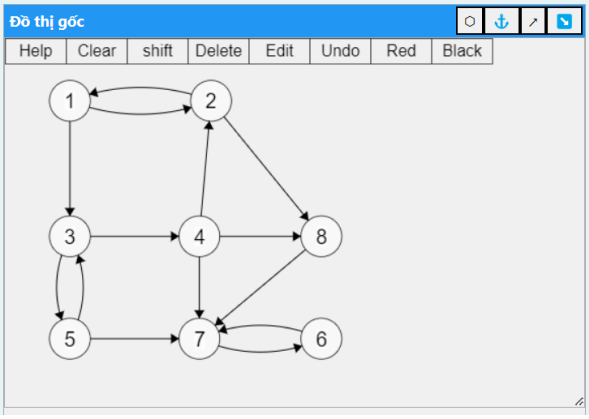
Dựa vào kết quả của thuật toán, vẽ các thành phần liên thông tìm được. Với mỗi thành phần liên thông, vẽ các đỉnh và các cung bên trong thành phần liên thông này. Nói cách khác, xoá bỏ các cung của đồ thị gốc mà 2 đỉnh của nó nằm ở 2 thành phần liên thông khác nhau.

Quy ước

- Mỗi thể hiện của hàm **SCC(u)** được minh họa bằng một khung chữ nhật. Việc gọi đệ quy được minh họa bằng một hình chữ nhật nhỏ hơn bên trong.
- Hãy làm bài theo thứ tự từ ngoài vào trong và từ trên xuống dưới.
- Nếu làm sai 1 bước, có thể bấm "**Lùi lại 1 bước**" để làm lại bước trước đó.
- Liệt kê các đỉnh theo thứ tự **1, 2, 3, ...**
- **k được khởi tạo = 1.**

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

Reset answer



Áp dụng thuật toán Tarjan bằng cách duyệt đệ quy theo chiều sâu bắt đầu từ đỉnh 1

Lùi lại 1 bước | Số bước: 52

SCC(1) ←→ →←

1. Đánh số cho đỉnh 1 và đưa nó vào ngăn xếp S.
   Gán num[1] = min_num[1] = k = 1 ; k++;
   Đưa 1 vào S. Kết quả: S = 1
   Thực hiện đánh số và đưa vào Stack ✓ 1
2. Với các đỉnh kề v của 1: 2,3 Xét ✓ 2

2a. v = '2' chưa duyệt Duyệt nó ✓ 3

SCC(2) ←→ →←

1. Đánh số cho đỉnh 2 và đưa nó vào ngăn xếp S.
   Gán num[2] = min_num[2] = k = 2 ; k++;
   Đưa 2 vào S. Kết quả: S = 1,2
   Thực hiện đánh số và đưa vào Stack ✓ 4
2. Với các đỉnh kề v của 2: 1,8 Xét ✓ 5

2b. v = '1' duyệt rồi nhưng vẫn còn trên stack Cập nhật min_num[u] ✓ 6
Cập nhật min_num[2] = min(min_num[2], num[1]) = 1
Cập nhật ✓ 7

2a. v = '8' chưa duyệt Duyệt nó ✓ 8

SCC(8) ←→ →←

1. Đánh số cho đỉnh 8 và đưa nó vào ngăn xếp S.
   Gán num[8] = min_num[8] = k = 3 ; k++;
   Đưa 8 vào S. Kết quả: S = 1,2,8
   Thực hiện đánh số và đưa vào Stack ✓ 9
2. Với các đỉnh kề v của 8: 7 Xét ✓ 10

2a. v = '7' chưa duyệt Duyệt nó ✓ 11

SCC(7) ←→ →←

1. Đánh số cho đỉnh 7 và đưa nó vào ngăn xếp S.
   Gán num[7] = min_num[7] = k = 4 ; k++;
   Đưa 7 vào S. Kết quả: S = 1,2,8,7
   Thực hiện đánh số và đưa vào Stack ✓ 12
2. Với các đỉnh kề v của 7: 6 Xét ✓ 13

2a. v = '6' chưa duyệt Duyệt nó ✓ 14

SCC(6) ←→ →←

1. Đánh số cho đỉnh 6 và đưa nó vào ngăn xếp S.
   Gán num[6] = min_num[6] = k = 5 ; k++;
   Đưa 6 vào S. Kết quả: S = 1,2,8,7,6
   Thực hiện đánh số và đưa vào Stack ✓ 15
2. Với các đỉnh kề v của 6: 7 Xét ✓ 16

2b. v = '7' duyệt rồi nhưng vẫn còn trên stack Cập nhật min_num[u] ✓ 17
Cập nhật min_num[6] = min(min_num[6], num[7]) = 4 Cập nhật ✓ 18

3. Kiểm tra num và min_num của 6
   num[6] == min_num[6]
   num[6] != min_num[6] ✗ 19

Cập nhật min_num[7] = min(min_num[7], min_num[6]) = 4 Cập nhật ✓ 20

3. Kiểm tra num và min_num của 7
   num[7] == min_num[7] num[7] != min_num[7] ✓ 21
   Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được 7
   Các đỉnh được lấy ra: 6,7
   Nội dung còn lại của ngăn xếp: 1,2,8
   Cập nhật ✓ 22

Cập nhật min_num[8] = min(min_num[8], min_num[7]) = 3 Cập nhật ✓ 23

3. Kiểm tra num và min_num của 8
   num[8] == min_num[8] num[8] != min_num[8] ✓ 24
   Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được 8
   Các đỉnh được lấy ra: 8
   Nội dung còn lại của ngăn xếp: 1,2
   Cập nhật ✓ 25

Cập nhật min_num[2] = min(min_num[2], min_num[8]) = 1 Cập nhật ✓ 26

3. Kiểm tra num và min_num của 2
   num[2] == min_num[2] num[2] != min_num[2] ✗ 27

Cập nhật min_num[1] = min(min_num[1], min_num[2]) = 1
Cập nhật ✓ 28

2a. v = '3' chưa duyệt Duyệt nó ✓ 29

SCC(3) ←→ →←

1. Đánh số cho đỉnh 3 và đưa nó vào ngăn xếp S.
   Gán num[3] = min_num[3] = k = 6 ; k++;
   Đưa 3 vào S. Kết quả: S = 1,2,3
   Thực hiện đánh số và đưa vào Stack ✓ 30
2. Với các đỉnh kề v của 3: 4,5 Xét ✓ 31

2a. v = '4' chưa duyệt Duyệt nó ✓ 32

SCC(4) ←→ →←

1. Đánh số cho đỉnh 4 và đưa nó vào ngăn xếp S.
   Gán num[4] = min_num[4] = k = 7 ; k++;
   Đưa 4 vào S. Kết quả: S = 1,2,3,4
   Thực hiện đánh số và đưa vào Stack ✓ 33
2. Với các đỉnh kề v của 4: 2,7,8 Xét ✓ 34

2b. v = '2' duyệt rồi nhưng vẫn còn trên stack Cập nhật min_num[u] ✓ 35
Cập nhật min_num[4] = min(min_num[4], num[2]) = 2 Cập nhật ✓ 36

2c. v = '7' duyệt rồi và không còn trên stack Bỏ qua ✗ 37

2c. v = '8' duyệt rồi và không còn trên stack Bỏ qua ✗ 38

3. Kiểm tra num và min_num của 4
   num[4] == min_num[4] num[4] != min_num[4] ✗ 39

Cập nhật min_num[3] = min(min_num[3], min_num[4]) = 2 Cập nhật ✓ 40

2a. v = '5' chưa duyệt Duyệt nó ✓ 41

SCC(5) ←→ →←

1. Đánh số cho đỉnh 5 và đưa nó vào ngăn xếp S.
   Gán num[5] = min_num[5] = k = 8 ; k++;
   Đưa 5 vào S. Kết quả: S = 1,2,3,4,5
   Thực hiện đánh số và đưa vào Stack ✓ 42
2. Với các đỉnh kề v của 5: 3,7 Xét ✓ 43

2b. v = '3' duyệt rồi nhưng vẫn còn trên stack Cập nhật min_num[u] ✓ 44
Cập nhật min_num[5] = min(min_num[5], num[3]) = 6 Cập nhật ✓ 45

2c. v = '7' duyệt rồi và không còn trên stack Bỏ qua ✗ 46

3. Kiểm tra num và min_num của 5
   num[5] == min_num[5] num[5] != min_num[5] ✗ 47

Cập nhật min_num[3] = min(min_num[3], min_num[5]) = 2 Cập nhật ✓ 48

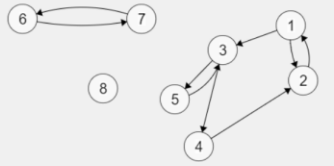
3. Kiểm tra num và min_num của 3
   num[3] == min_num[3] num[3] != min_num[3] ✗ 49

Cập nhật min_num[1] = min(min_num[1], min_num[3]) = 1
Cập nhật ✓ 50

3. Kiểm tra num và min_num của 1
   num[1] == min_num[1] num[1] != min_num[1] ✓ 51
   Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được 1
   Các đỉnh được lấy ra: 5,4,3,2,1
   Nội dung còn lại của ngăn xếp:
   Cập nhật ✓ 52

Vẽ các thành phần liên thông

Help | Clear | shift | Delete | Edit | Undo | Red | Black



Câu hỏi **1**

Đúng

Đạt điểm 1,00 trên 1,00

Đặt cờ

Thuật toán Tarjan tìm các thành phần liên thông mạnh (Tarjan's strongly connected components algorithm) dựa trên thuật toán duyệt đồ thị theo chiều sâu (sử dụng đệ quy) kết hợp với đánh số các đỉnh và một ngăn xếp lưu trữ các thành phần liên thông mạnh (strongly connected components gọi tắt là SCC).

```
void SCC(int u) {
    //1. Đánh số cho đỉnh u, đưa u vào ngăn xếp
    num[u] = k; min_num[u] = k; k++;
    push(S, u); on_stack[u] = true;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u)
        if (v chưa duyệt) {
            //2a. Duyệt v và cập nhật lại min_num[u]
            SCC(v);
            min_num[u] = min(min_num[u], min_num[v]);
        } else if (v còn trên stack) {
            //2b. cập nhật lại min_num[u]
            min_num[u] = min(min_num[u], num[v]);
        } else {
            //2c. bỏ qua, không làm gì cả
        }

   //3. Kiểm tra num[u] == min_num[u]
   if (num[u] == min_num[u]) {
       //Lấy các đỉnh trong stack ra cho đến khi gặp u
       //Các đỉnh này thuộc về một thành phần liên thông
   }
}
```

Thuật toán trên gồm 3 bước chính:

1. Đánh số cho đỉnh u, đưa u vào ngăn xếp. Sau khi đánh số xong tăng k lên. Thông thường, k được khởi tạo bằng 1.
2. Xét các đỉnh kề v của u, có 3 trường hợp xảy ra:
   - v chưa duyệt => gọi đệ quy duyệt v và cập nhật lại min_num[u]
   - v duyệt rồi nhưng vẫn còn trên stack => cập nhật lại min_num[u]
   - v duyệt rồi và không còn trên stack => bỏ qua
3. Kiểm tra num[u] và min_num[u]: nếu bằng nhau thì u là đỉnh khớp (articulation vertex) hay đỉnh cắt (cut vertex)
   - Lần lượt lấy các đỉnh trong stack ra cho đến khi u được lấy ra. Các đỉnh được lấy ra này chính là thành phần liên thông mạnh chứa u.

Cho đồ thị **có hướng** gồm **5** đỉnh và **8** cung như bên như bên dưới. Hãy áp dụng thuật toán Tarjan để tìm các thành phần liên thông mạnh của G bắt đầu từ đỉnh **C**.

Dựa vào kết quả của thuật toán, vẽ các thành phần liên thông tìm được. Với mỗi thành phần liên thông, vẽ các đỉnh và các cung bên trong thành phần liên thông này. Nói cách khác, xoá bỏ các cung của đồ thị gốc mà 2 đỉnh của nó nằm ở 2 thành phần liên thông khác nhau.

Quy ước

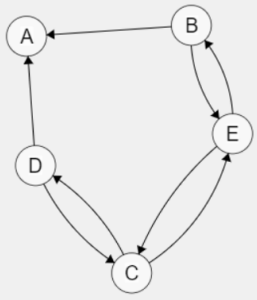
- Mỗi thể hiện của hàm **SCC(u)** được minh họa bằng một khung chữ nhật. Việc gọi đệ quy được minh họa bằng một hình chữ nhật nhỏ hơn bên trong.
- Hãy làm bài theo thứ tự từ ngoài vào trong và từ trên xuống dưới.
- Nếu làm sai 1 bước, có thể bấm "**Lùi lại 1 bước**" để làm lại bước trước đó.
- Liệt kê các đỉnh theo thứ tự **A, B, C, ...**
- **k được khởi tạo = 1.**

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

Reset answer

Đồ thị gốc

Help | Clear | shift | Delete | Edit | Undo | Red | Black



Áp dụng thuật toán Tarjan bằng cách duyệt đệ quy theo chiều sâu bắt đầu từ đỉnh C

Lùi lại 1 bước | Số bước: 32

SCC(C) ←→ →←

1. Đánh số cho đỉnh C và đưa nó vào ngăn xếp S.

Gán num[C] = min_num[C] = k = 1 ; k++;

Đưa C vào S. Kết quả: S = C

Thực hiện đánh số và đưa vào Stack ✓ 1

2. Với các đỉnh kề v của C: D,E Xét ✓ 2

2a. v = 'D' chưa duyệt Duyệt nó ✓ 3

SCC(D) ←→ →←

1. Đánh số cho đỉnh D và đưa nó vào ngăn xếp S.

Gán num[D] = min_num[D] = k = 2 ; k++;

Đưa D vào S. Kết quả: S = C,D

Thực hiện đánh số và đưa vào Stack ✓ 4

2. Với các đỉnh kề v của D: A,C Xét ✓ 5

2a. v = 'A' chưa duyệt Duyệt nó ✓ 6

SCC(A) ←→ →←

1. Đánh số cho đỉnh A và đưa nó vào ngăn xếp S.

Gán num[A] = min_num[A] = k = 3 ; k++;

Đưa A vào S. Kết quả: S = C,D,A

Thực hiện đánh số và đưa vào Stack ✓ 7

2. Với các đỉnh kề v của A: Xét ✓ 8

3. Kiểm tra num và min_num của A

num[A] == min_num[A] num[A] != min_num[A] ✓ 9

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được A

Các đỉnh được lấy ra: A

Nội dung còn lại của ngăn xếp: C,D

Cập nhật ✓ 10

Cập nhật min_num[D] = min(min_num[D], min_num[A]) = 2 Cập nhật ✓ 11

2b. v = 'C' duyệt rồi nhưng vẫn còn trên stack Cập nhật min_num[u] ✓ 12

Cập nhật min_num[D] = min(min_num[D], num[C]) = 1 Cập nhật ✓ 13

3. Kiểm tra num và min_num của D

num[D] == min_num[D] num[D] != min_num[D] ✗ 14

Cập nhật min_num[C] = min(min_num[C], min_num[D]) = 1 Cập nhật ✓ 15

2a. v = 'E' chưa duyệt Duyệt nó ✓ 16

SCC(E) ←→ →←

1. Đánh số cho đỉnh E và đưa nó vào ngăn xếp S.

Gán num[E] = min_num[E] = k = 4 ; k++;

Đưa E vào S. Kết quả: S = C,D,E

Thực hiện đánh số và đưa vào Stack ✓ 17

2. Với các đỉnh kề v của E: B,C Xét ✓ 18

2a. v = 'B' chưa duyệt Duyệt nó ✓ 19

SCC(B) ←→ →←

1. Đánh số cho đỉnh B và đưa nó vào ngăn xếp S.

Gán num[B] = min_num[B] = k = 5 ; k++;

Đưa B vào S. Kết quả: S = C,D,E,B

Thực hiện đánh số và đưa vào Stack ✓ 20

2. Với các đỉnh kề v của B: A,E Xét ✓ 21

2c. v = 'A' duyệt rồi và không còn trên stack Bỏ qua ✗ 22

2b. v = 'E' duyệt rồi nhưng vẫn còn trên stack

Cập nhật min_num[u] ✓ 23

Cập nhật min_num[B] = min(min_num[B], num[E]) = 4 Cập nhật ✓ 24

3. Kiểm tra num và min_num của B

num[B] == min_num[B] num[B] != min_num[B] ✗ 25

Cập nhật min_num[E] = min(min_num[E], min_num[B]) = 4 Cập nhật ✓ 26

2b. v = 'C' duyệt rồi nhưng vẫn còn trên stack Cập nhật min_num[u] ✓ 27

Cập nhật min_num[E] = min(min_num[E], num[C]) = 1 Cập nhật ✓ 28

3. Kiểm tra num và min_num của E

num[E] == min_num[E] num[E] != min_num[E] ✗ 29

Cập nhật min_num[C] = min(min_num[C], min_num[E]) = 1 Cập nhật ✓ 30

3. Kiểm tra num và min_num của C

num[C] == min_num[C] num[C] != min_num[C] ✓ 31

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được C

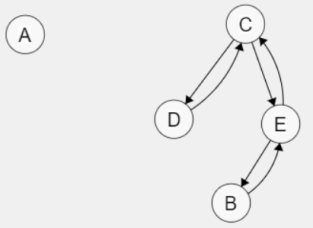
Các đỉnh được lấy ra: B,E,D,C

Nội dung còn lại của ngăn xếp:

Cập nhật ✓ 32

Vẽ các thành phần liên thông

Help | Clear | shift | Delete | Edit | Undo | Red | Black



Câu hỏi **1**

Đúng

Đạt điểm 1,00 trên 1,00

Đặt cờ

Thuật toán Tarjan tìm các thành phần liên thông mạnh (Tarjan's strongly connected components algorithm) dựa trên thuật toán duyệt đồ thị theo chiều sâu (sử dụng đệ quy) kết hợp với đánh số các đỉnh và một ngăn xếp lưu trữ các thành phần liên thông mạnh (strongly connected components gọi tắt là SCC).

```
void SCC(int u) {
    //1. Đánh số cho đỉnh u, đưa u vào ngăn xếp
    num[u] = k; min_num[u] = k; k++;
    push(S, u); on_stack[u] = true;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u)
        if (v chưa duyệt) {
            //2a. Duyệt v và cập nhật lại min_num[u]
            SCC(v);
            min_num[u] = min(min_num[u], min_num[v]);
        } else if (v còn trên stack) {
            //2b. cập nhật lại min_num[u]
            min_num[u] = min(min_num[u], num[v]);
        } else {
            //2c. bỏ qua, không làm gì cả
        }

   //3. Kiểm tra num[u] == min_num[u]
   if (num[u] == min_num[u]) {
       //Lấy các đỉnh trong stack ra cho đến khi gặp u
       //Các đỉnh này thuộc về một thành phần liên thông
   }
}
```

Thuật toán trên gồm 3 bước chính:

1. Đánh số cho đỉnh u, đưa u vào ngăn xếp. Sau khi đánh số xong tăng k lên. Thông thường, k được khởi tạo bằng 1.
2. Xét các đỉnh kề v của u, có 3 trường hợp xảy ra:
   - v chưa duyệt => gọi đệ quy duyệt v và cập nhật lại min_num[u]
   - v duyệt rồi nhưng vẫn còn trên stack => cập nhật lại min_num[u]
   - v duyệt rồi và không còn trên stack => bỏ qua
3. Kiểm tra num[u] và min_num[u]: nếu bằng nhau thì u là đỉnh khớp (articulation vertex) hay đỉnh cắt (cut vertex)
   - Lần lượt lấy các đỉnh trong stack ra cho đến khi u được lấy ra. Các đỉnh được lấy ra này chính là thành phần liên thông mạnh chứa u.

Cho đồ thị **có hướng** gồm **7** đỉnh và **10** cung như bên như bên dưới. Hãy áp dụng thuật toán Tarjan để tìm các thành phần liên thông mạnh của G bắt đầu từ đỉnh **B**.

Dựa vào kết quả của thuật toán, vẽ các thành phần liên thông tìm được. Với mỗi thành phần liên thông, vẽ các đỉnh và các cung bên trong thành phần liên thông này. Nói cách khác, xoá bỏ các cung của đồ thị gốc mà 2 đỉnh của nó nằm ở 2 thành phần liên thông khác nhau.

Quy ước

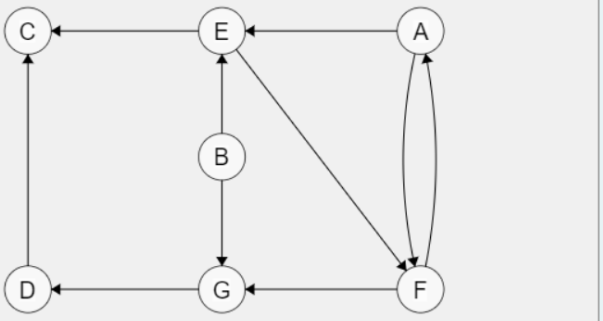
- Mỗi thể hiện của hàm **SCC(u)** được minh họa bằng một khung chữ nhật. Việc gọi đệ quy được minh họa bằng một hình chữ nhật nhỏ hơn bên trong.
- Hãy làm bài theo thứ tự từ ngoài vào trong và từ trên xuống dưới.
- Nếu làm sai 1 bước, có thể bấm "**Lùi lại 1 bước**" để làm lại bước trước đó.
- Liệt kê các đỉnh theo thứ tự **A, B, C, ...**
- **k được khởi tạo = 1.**

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

Reset answer

Đồ thị gốc

Help | Clear | shift | Delete | Edit | Undo | Red | Black



Áp dụng thuật toán Tarjan bằng cách duyệt đệ quy theo chiều sâu bắt đầu từ đỉnh B

Lùi lại 1 bước Số bước: 44

SCC(B) ←→ →←

1. Đánh số cho đỉnh B và đưa nó vào ngăn xếp S.

Gán num[B] = min_num[B] = k = 1 ; k++;

Đưa B vào S. Kết quả: S = B

Thực hiện đánh số và đưa vào Stack ✓ 1

2. Với các đỉnh kề v của B: E,G Xét ✓ 2

2a. v = 'E' chưa duyệt Duyệt nó ✓ 3

SCC(E) ←→ →←

1. Đánh số cho đỉnh E và đưa nó vào ngăn xếp S.

Gán num[E] = min_num[E] = k = 2 ; k++;

Đưa E vào S. Kết quả: S = B,E

Thực hiện đánh số và đưa vào Stack ✓ 4

2. Với các đỉnh kề v của E: C,F Xét ✓ 5

2a. v = 'C' chưa duyệt Duyệt nó ✓ 6

SCC(C) ←→ →←

1. Đánh số cho đỉnh C và đưa nó vào ngăn xếp S.

Gán num[C] = min_num[C] = k = 3 ; k++;

Đưa C vào S. Kết quả: S = B,E,C

Thực hiện đánh số và đưa vào Stack ✓ 7

2. Với các đỉnh kề v của C: Xét ✓ 8

3. Kiểm tra num và min_num của C

num[C] == min_num[C] num[C] != min_num[C] ✓ 9

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được C

Các đỉnh được lấy ra: C

Nội dung còn lại của ngăn xếp: B,E

Cập nhật ✓ 10

Cập nhật min_num[E] = min(min_num[E], min_num[C]) = 2 Cập nhật ✓ 11

2a. v = 'F' chưa duyệt Duyệt nó ✓ 12

SCC(F) ←→ →←

1. Đánh số cho đỉnh F và đưa nó vào ngăn xếp S.

Gán num[F] = min_num[F] = k = 4 ; k++;

Đưa F vào S. Kết quả: S = B,E,F

Thực hiện đánh số và đưa vào Stack ✓ 13

2. Với các đỉnh kề v của F: A,G Xét ✓ 14

2a. v = 'A' chưa duyệt Duyệt nó ✓ 15

SCC(A) ←→ →←

1. Đánh số cho đỉnh A và đưa nó vào ngăn xếp S.

Gán num[A] = min_num[A] = k = 5 ; k++;

Đưa A vào S. Kết quả: S = B,E,F,A

Thực hiện đánh số và đưa vào Stack ✓ 16

2. Với các đỉnh kề v của A: E,F Xét ✓ 17

2b. v = 'E' duyệt rồi nhưng vẫn còn trên stack

Cập nhật min_num[u] ✓ 18

Cập nhật min_num[A] = min(min_num[A], num[E]) = 2 Cập nhật ✓ 19

2b. v = 'F' duyệt rồi nhưng vẫn còn trên stack

Cập nhật min_num[u] ✓ 20

Cập nhật min_num[A] = min(min_num[A], num[F]) = 2 Cập nhật ✓ 21

3. Kiểm tra num và min_num của A

num[A] == min_num[A] num[A] != min_num[A]

✗ 22

Cập nhật min_num[F] = min(min_num[F], min_num[A]) = 2 Cập nhật ✓ 23

2a. v = 'G' chưa duyệt Duyệt nó ✓ 24

SCC(G) ←→ →←

1. Đánh số cho đỉnh G và đưa nó vào ngăn xếp S.

Gán num[G] = min_num[G] = k = 6 ; k++;

Đưa G vào S. Kết quả: S = B,E,F,A,G

Thực hiện đánh số và đưa vào Stack ✓ 25

2. Với các đỉnh kề v của G: D

Xét ✓ 26

2a. v = 'D' chưa duyệt Duyệt nó ✓ 27

SCC(D) ←→ →←

1. Đánh số cho đỉnh D và đưa nó vào ngăn xếp S.

Gán num[D] = min_num[D] = k = 7 ; k++;

Đưa D vào S. Kết quả: S = B,E,F,A,G,D

Thực hiện đánh số và đưa vào Stack ✓ 28

2. Với các đỉnh kề v của D: C Xét ✓ 29

2c. v = 'C' duyệt rồi và không còn trên stack Bỏ qua ✗ 30

3. Kiểm tra num và min_num của D

num[D] == min_num[D]

num[D] != min_num[D] ✓ 31

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được D

Các đỉnh được lấy ra: D

Nội dung còn lại của ngăn xếp: B,E,F,A,G

Cập nhật ✓ 32

Cập nhật min_num[G] = min(min_num[G], min_num[D]) = 6 Cập nhật ✓ 33

3. Kiểm tra num và min_num của G

num[G] == min_num[G] num[G] != min_num[G] ✓ 34

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được G

Các đỉnh được lấy ra: G

Nội dung còn lại của ngăn xếp: B,E,F,A

Cập nhật ✓ 35

Cập nhật min_num[F] = min(min_num[F], min_num[G]) = 2 Cập nhật ✓ 36

3. Kiểm tra num và min_num của F

num[F] == min_num[F] num[F] != min_num[F] ✗ 37

Cập nhật min_num[E] = min(min_num[E], min_num[F]) = 2 Cập nhật ✓ 38

3. Kiểm tra num và min_num của E

num[E] == min_num[E] num[E] != min_num[E] ✓ 39

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được E

Các đỉnh được lấy ra: A,F,E

Nội dung còn lại của ngăn xếp: B

Cập nhật ✓ 40

Cập nhật min_num[B] = min(min_num[B], min_num[E]) = 1 Cập nhật ✓ 41

2c. v = 'G' duyệt rồi và không còn trên stack Bỏ qua ✗ 42

3. Kiểm tra num và min_num của B

num[B] == min_num[B] num[B] != min_num[B] ✓ 43

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được B

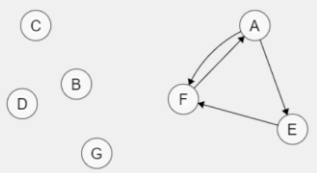
Các đỉnh được lấy ra: B

Nội dung còn lại của ngăn xếp:

Cập nhật ✓ 44

Vẽ các thành phần liên thông

Help | Clear | shift | Delete | Edit | Undo | Red | Black



Câu hỏi **1**

Đúng

Đạt điểm 1,00 trên 1,00

Đặt cờ

Thuật toán Tarjan tìm các thành phần liên thông mạnh (Tarjan's strongly connected components algorithm) dựa trên thuật toán duyệt đồ thị theo chiều sâu (sử dụng đệ quy) kết hợp với đánh số các đỉnh và một ngăn xếp lưu trữ các thành phần liên thông mạnh (strongly connected components gọi tắt là SCC).

```
void SCC(int u) {
    //1. Đánh số cho đỉnh u, đưa u vào ngăn xếp
    num[u] = k; min_num[u] = k; k++;
    push(S, u); on_stack[u] = true;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u)
        if (v chưa duyệt) {
            //2a. Duyệt v và cập nhật lại min_num[u]
            SCC(v);
            min_num[u] = min(min_num[u], min_num[v]);
        } else if (v còn trên stack) {
            //2b. cập nhật lại min_num[u]
            min_num[u] = min(min_num[u], num[v]);
        } else {
            //2c. bỏ qua, không làm gì cả
        }

   //3. Kiểm tra num[u] == min_num[u]
   if (num[u] == min_num[u]) {
       //Lấy các đỉnh trong stack ra cho đến khi gặp u
       //Các đỉnh này thuộc về một thành phần liên thông
   }
}
```

Thuật toán trên gồm 3 bước chính:

1. Đánh số cho đỉnh u, đưa u vào ngăn xếp. Sau khi đánh số xong tăng k lên. Thông thường, k được khởi tạo bằng 1.
2. Xét các đỉnh kề v của u, có 3 trường hợp xảy ra:
   - v chưa duyệt => gọi đệ quy duyệt v và cập nhật lại min_num[u]
   - v duyệt rồi nhưng vẫn còn trên stack => cập nhật lại min_num[u]
   - v duyệt rồi và không còn trên stack => bỏ qua
3. Kiểm tra num[u] và min_num[u]: nếu bằng nhau thì u là đỉnh khớp (articulation vertex) hay đỉnh cắt (cut vertex)
   - Lần lượt lấy các đỉnh trong stack ra cho đến khi u được lấy ra. Các đỉnh được lấy ra này chính là thành phần liên thông mạnh chứa u.

Cho đồ thị **có hướng** gồm **5** đỉnh và **6** cung như bên như bên dưới. Hãy áp dụng thuật toán Tarjan để tìm các thành phần liên thông mạnh của G bắt đầu từ đỉnh **B**.

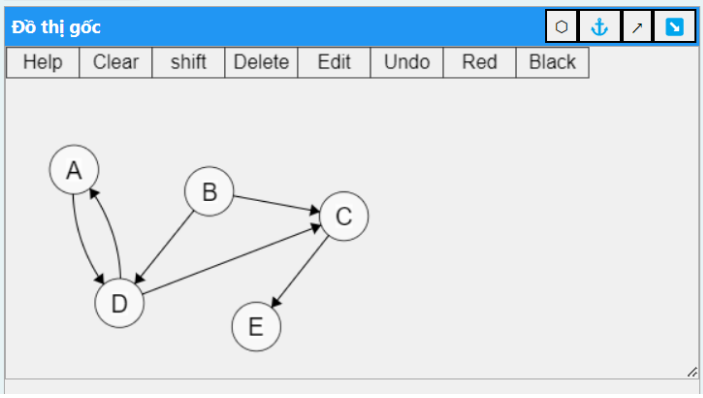
Dựa vào kết quả của thuật toán, vẽ các thành phần liên thông tìm được. Với mỗi thành phần liên thông, vẽ các đỉnh và các cung bên trong thành phần liên thông này. Nói cách khác, xoá bỏ các cung của đồ thị gốc mà 2 đỉnh của nó nằm ở 2 thành phần liên thông khác nhau.

Quy ước

- Mỗi thể hiện của hàm **SCC(u)** được minh họa bằng một khung chữ nhật. Việc gọi đệ quy được minh họa bằng một hình chữ nhật nhỏ hơn bên trong.
- Hãy làm bài theo thứ tự từ ngoài vào trong và từ trên xuống dưới.
- Nếu làm sai 1 bước, có thể bấm "**Lùi lại 1 bước**" để làm lại bước trước đó.
- Liệt kê các đỉnh theo thứ tự **A, B, C, ...**
- **k được khởi tạo = 1.**

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

Reset answer



Áp dụng thuật toán Tarjan bằng cách duyệt đệ quy theo chiều sâu bắt đầu từ đỉnh B

Lùi lại 1 bước Số bước: **30**

SCC(B) ←→ →←

1. Đánh số cho đỉnh B và đưa nó vào ngăn xếp S.

Gán num[B] = min_num[B] = k = 1 ; k++;

Đưa B vào S. Kết quả: S = B

Thực hiện đánh số và đưa vào Stack ✓ 1

2. Với các đỉnh kề v của B: C,D Xét ✓ 2

2a. v = 'C' chưa duyệt Duyệt nó ✓ 3

SCC(C) ←→ →←

1. Đánh số cho đỉnh C và đưa nó vào ngăn xếp S.

Gán num[C] = min_num[C] = k = 2 ; k++;

Đưa C vào S. Kết quả: S = B,C

Thực hiện đánh số và đưa vào Stack ✓ 4

2. Với các đỉnh kề v của C: E Xét ✓ 5

2a. v = 'E' chưa duyệt Duyệt nó ✓ 6

SCC(E) ←→ →←

1. Đánh số cho đỉnh E và đưa nó vào ngăn xếp S.

Gán num[E] = min_num[E] = k = 3 ; k++;

Đưa E vào S. Kết quả: S = B,C,E

Thực hiện đánh số và đưa vào Stack ✓ 7

2. Với các đỉnh kề v của E: Xét ✓ 8

3. Kiểm tra num và min_num của E

num[E] == min_num[E] num[E] != min_num[E] ✓ 9

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được E

Các đỉnh được lấy ra: E

Nội dung còn lại của ngăn xếp: B,C

Cập nhật ✓ 10

Cập nhật min_num[C] = min(min_num[C], min_num[E]) = 2 Cập nhật ✓ 11

3. Kiểm tra num và min_num của C

num[C] == min_num[C] num[C] != min_num[C] ✓ 12

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được C

Các đỉnh được lấy ra: C

Nội dung còn lại của ngăn xếp: B

Cập nhật ✓ 13

Cập nhật min_num[B] = min(min_num[B], min_num[C]) = 1 Cập nhật ✓ 14

2a. v = 'D' chưa duyệt Duyệt nó ✓ 15

SCC(D) ←→ →←

1. Đánh số cho đỉnh D và đưa nó vào ngăn xếp S.

Gán num[D] = min_num[D] = k = 4 ; k++;

Đưa D vào S. Kết quả: S = B,D

Thực hiện đánh số và đưa vào Stack ✓ 16

2. Với các đỉnh kề v của D: A,C Xét ✓ 17

2a. v = 'A' chưa duyệt Duyệt nó ✓ 18

SCC(A) ←→ →←

1. Đánh số cho đỉnh A và đưa nó vào ngăn xếp S.

Gán num[A] = min_num[A] = k = 5 ; k++;

Đưa A vào S. Kết quả: S = B,D,A

Thực hiện đánh số và đưa vào Stack ✓ 19

2. Với các đỉnh kề v của A: D Xét ✓ 20

2b. v = 'D' duyệt rồi nhưng vẫn còn trên stack

Cập nhật min_num[u] ✓ 21

Cập nhật min_num[A] = min(min_num[A], num[D]) = 4 Cập nhật ✓ 22

3. Kiểm tra num và min_num của A

num[A] == min_num[A] num[A] != min_num[A] ✗ 23

Cập nhật min_num[D] = min(min_num[D], min_num[A]) = 4 Cập nhật ✓ 24

2c. v = 'C' duyệt rồi và không còn trên stack Bỏ qua ✗ 25

3. Kiểm tra num và min_num của D

num[D] == min_num[D] num[D] != min_num[D] ✓ 26

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được D

Các đỉnh được lấy ra: A,D

Nội dung còn lại của ngăn xếp: B

Cập nhật ✓ 27

Cập nhật min_num[B] = min(min_num[B], min_num[D]) = 1 Cập nhật ✓ 28

3. Kiểm tra num và min_num của B

num[B] == min_num[B] num[B] != min_num[B] ✓ 29

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được B

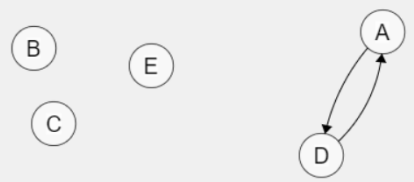
Các đỉnh được lấy ra: B

Nội dung còn lại của ngăn xếp:

Cập nhật ✓ 30

Vẽ các thành phần liên thông

Help | Clear | shift | Delete | Edit | Undo | Red | Black



Câu hỏi **1**

Đúng

Đạt điểm 0,90 trên 1,00

Đặt cờ

Thuật toán Tarjan tìm các thành phần liên thông mạnh (Tarjan's strongly connected components algorithm) dựa trên thuật toán duyệt đồ thị theo chiều sâu (sử dụng đệ quy) kết hợp với đánh số các đỉnh và một ngăn xếp lưu trữ các thành phần liên thông mạnh (strongly connected components gọi tắt là SCC).

```
void SCC(int u) {
    //1. Đánh số cho đỉnh u, đưa u vào ngăn xếp
    num[u] = k; min_num[u] = k; k++;
    push(S, u); on_stack[u] = true;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u)
        if (v chưa duyệt) {
            //2a. Duyệt v và cập nhật lại min_num[u]
            SCC(v);
            min_num[u] = min(min_num[u], min_num[v]);
        } else if (v còn trên stack) {
            //2b. cập nhật lại min_num[u]
            min_num[u] = min(min_num[u], num[v]);
        } else {
            //2c. bỏ qua, không làm gì cả
        }

   //3. Kiểm tra num[u] == min_num[u]
   if (num[u] == min_num[u]) {
       //Lấy các đỉnh trong stack ra cho đến khi gặp u
       //Các đỉnh này thuộc về một thành phần liên thông
   }
}
```

Thuật toán trên gồm 3 bước chính:

1. Đánh số cho đỉnh u, đưa u vào ngăn xếp. Sau khi đánh số xong tăng k lên. Thông thường, k được khởi tạo bằng 1.
2. Xét các đỉnh kề v của u, có 3 trường hợp xảy ra:
   - v chưa duyệt => gọi đệ quy duyệt v và cập nhật lại min_num[u]
   - v duyệt rồi nhưng vẫn còn trên stack => cập nhật lại min_num[u]
   - v duyệt rồi và không còn trên stack => bỏ qua
3. Kiểm tra num[u] và min_num[u]: nếu bằng nhau thì u là đỉnh khớp (articulation vertex) hay đỉnh cắt (cut vertex)
   - Lần lượt lấy các đỉnh trong stack ra cho đến khi u được lấy ra. Các đỉnh được lấy ra này chính là thành phần liên thông mạnh chứa u.

Cho đồ thị **có hướng** gồm **7** đỉnh và **11** cung như bên như bên dưới. Hãy áp dụng thuật toán Tarjan để tìm các thành phần liên thông mạnh của G bắt đầu từ đỉnh **5**.

Dựa vào kết quả của thuật toán, vẽ các thành phần liên thông tìm được. Với mỗi thành phần liên thông, vẽ các đỉnh và các cung bên trong thành phần liên thông này. Nói cách khác, xoá bỏ các cung của đồ thị gốc mà 2 đỉnh của nó nằm ở 2 thành phần liên thông khác nhau.

Quy ước

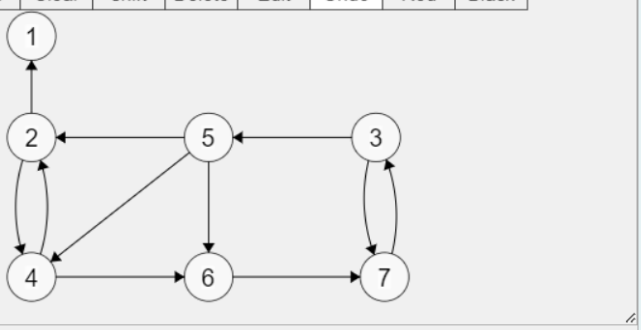
- Mỗi thể hiện của hàm **SCC(u)** được minh họa bằng một khung chữ nhật. Việc gọi đệ quy được minh họa bằng một hình chữ nhật nhỏ hơn bên trong.
- Hãy làm bài theo thứ tự từ ngoài vào trong và từ trên xuống dưới.
- Nếu làm sai 1 bước, có thể bấm "**Lùi lại 1 bước**" để làm lại bước trước đó.
- Liệt kê các đỉnh theo thứ tự **1, 2, 3, ...**
- **k được khởi tạo = 1.**

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

Reset answer

Đồ thị gốc

Help | Clear | shift | Delete | Edit | Undo | Red | Black



Áp dụng thuật toán Tarjan bằng cách duyệt đệ quy theo chiều sâu bắt đầu từ đỉnh 5

Lùi lại 1 bước | Số bước: 45

SCC(5) ←→ →←

1. Đánh số cho đỉnh 5 và đưa nó vào ngăn xếp S.

Gán num[5] = min_num[5] = k = 1 ; k++;

Đưa 5 vào S. Kết quả: S = 5

Thực hiện đánh số và đưa vào Stack ✓ 1

2. Với các đỉnh kề v của 5: 2,4,6 Xét ✓ 2

2a. v = '2' chưa duyệt Duyệt nó ✓ 3

SCC(2) ←→ →←

1. Đánh số cho đỉnh 2 và đưa nó vào ngăn xếp S.

Gán num[2] = min_num[2] = k = 2 ; k++;

Đưa 2 vào S. Kết quả: S = 5,2

Thực hiện đánh số và đưa vào Stack ✓ 4

2. Với các đỉnh kề v của 2: 1,4 Xét ✓ 5

2a. v = '1' chưa duyệt Duyệt nó ✓ 6

SCC(1) ←→ →←

1. Đánh số cho đỉnh 1 và đưa nó vào ngăn xếp S.

Gán num[1] = min_num[1] = k = 3 ; k++;

Đưa 1 vào S. Kết quả: S = 5,2,1

Thực hiện đánh số và đưa vào Stack ✓ 7

2. Với các đỉnh kề v của 1: Xét ✓ 8

3. Kiểm tra num và min_num của 1

num[1] == min_num[1] num[1] != min_num[1] ✓ 9

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được 1

Các đỉnh được lấy ra: 1

Nội dung còn lại của ngăn xếp: 5,2

Cập nhật ✓ 10

Cập nhật min_num[2] = min(min_num[2], min_num[1]) = 2 Cập nhật ✓ 11

2a. v = '4' chưa duyệt Duyệt nó ✓ 12

SCC(4) ←→ →←

1. Đánh số cho đỉnh 4 và đưa nó vào ngăn xếp S.

Gán num[4] = min_num[4] = k = 4 ; k++;

Đưa 4 vào S. Kết quả: S = 5,2,4

Thực hiện đánh số và đưa vào Stack ✓ 13

2. Với các đỉnh kề v của 4: 2,6 Xét ✓ 14

2b. v = '2' duyệt rồi nhưng vẫn còn trên stack

Cập nhật min_num[u] ✓ 15

Cập nhật min_num[4] = min(min_num[4], num[2]) = 2 Cập nhật ✓ 16

2a. v = '6' chưa duyệt Duyệt nó ✓ 17

SCC(6) ←→ →←

1. Đánh số cho đỉnh 6 và đưa nó vào ngăn xếp S.

Gán num[6] = min_num[6] = k = 5 ; k++;

Đưa 6 vào S. Kết quả: S = 5,2,4,6

Thực hiện đánh số và đưa vào Stack ✓ 18

2. Với các đỉnh kề v của 6: 7

Xét ✓ 19

2a. v = '7' chưa duyệt Duyệt nó ✓ 20

SCC(7) ←→ →←

1. Đánh số cho đỉnh 7 và đưa nó vào ngăn xếp S.

Gán num[7] = min_num[7] = k = 6 ; k++;

Đưa 7 vào S. Kết quả: S = 5,2,4,6,7

Thực hiện đánh số và đưa vào Stack ✓ 21

2. Với các đỉnh kề v của 7: 3 Xét ✓ 22

2a. v = '3' chưa duyệt Duyệt nó ✓ 23

SCC(3) ←→ →←

1. Đánh số cho đỉnh 3 và đưa nó vào ngăn xếp S.

Gán num[3] = min_num[3] = k = 7 ; k++;

Đưa 3 vào S. Kết quả: S = 5,2,4,6,7,3

Thực hiện đánh số và đưa vào Stack ✓ 24

2. Với các đỉnh kề v của 3: 5,7 Xét ✓ 25

2b. v = '5' duyệt rồi nhưng vẫn còn trên stack

Cập nhật min_num[u] ✓ 26

Cập nhật min_num[3] = min(min_num[3], num[5]) = 1 Cập nhật ✓ 27

2b. v = '7' duyệt rồi nhưng vẫn còn trên stack

Cập nhật min_num[u] ✓ 28

Cập nhật min_num[3] = min(min_num[3], num[7]) = 1 Cập nhật ✓ 29

3. Kiểm tra num và min_num của 3

num[3] == min_num[3]

num[3] != min_num[3] ✗ 30

Cập nhật min_num[7] = min(min_num[7], min_num[3]) = 1 Cập nhật ✓ 31

3. Kiểm tra num và min_num của 7

num[7] == min_num[7]

num[7] != min_num[7] ✗ 32

Cập nhật min_num[6] = min(min_num[6], min_num[7]) = 1 Cập nhật ✓ 33

3. Kiểm tra num và min_num của 6

num[6] == min_num[6] num[6] != min_num[6]

✗ 34

Cập nhật min_num[4] = min(min_num[4], min_num[6]) = 1 Cập nhật ✓ 35

3. Kiểm tra num và min_num của 4

num[4] == min_num[4] num[4] != min_num[4] ✗ 36

Cập nhật min_num[2] = min(min_num[2], min_num[4]) = 1 Cập nhật ✓ 37

3. Kiểm tra num và min_num của 2

num[2] == min_num[2] num[2] != min_num[2] ✗ 38

Cập nhật min_num[5] = min(min_num[5], min_num[2]) = 1 Cập nhật ✓ 39

2b. v = '4' duyệt rồi nhưng vẫn còn trên stack Cập nhật min_num[u] ✓ 40

Cập nhật min_num[5] = min(min_num[5], num[4]) = 1 Cập nhật ✓ 41

2b. v = '6' duyệt rồi nhưng vẫn còn trên stack Cập nhật min_num[u] ✓ 42

Cập nhật min_num[5] = min(min_num[5], num[6]) = 1 Cập nhật ✓ 43

3. Kiểm tra num và min_num của 5

num[5] == min_num[5] num[5] != min_num[5] ✓ 44

Lấy các đỉnh trong ngăn xếp ra cho đến khi lấy được 5

Các đỉnh được lấy ra: 3,7,6,4,2,5

Nội dung còn lại của ngăn xếp:

Cập nhật ✓ 45

Vẽ các thành phần liên thông

Help | Clear | shift | Delete | Edit | Undo | Red | Black