

Tuần 1 - Đồ thị cơ bản

* Tự học - Vẽ đồ thị vô hướng - ma trận kề

Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

▼ Đặt cờ

Vẽ đồ thị **vô hướng** có 6 đỉnh và ma trận kề (mở rộng) như sau:

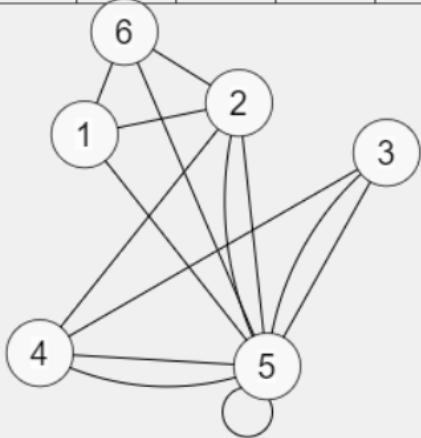
```
0 1 0 0 1 1  
1 0 0 1 2 1  
0 0 0 1 2 0  
0 1 1 0 2 0  
1 2 2 2 1 1  
1 1 0 0 1 0
```

Chú ý

- Đồ thị có thể có đa cung. Nếu có x cung giữa đỉnh u và v thì ô $(u, v) = x$ và ô $(v, u) = x$
- Đồ thị có thể có khuyên. Nếu có khuyên (u, u) , thì ô $(u, u) = 1$

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

Help Clear shift Delete Edit Undo Red Black



* Tự học - Vẽ đồ thị vô hướng - danh sách đỉnh kề

Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

Đặt cờ

Vẽ đồ thị **vô hướng** có 6 đỉnh và danh sách đỉnh kề của các đỉnh như sau:

1: 1 4

2: 2 3 4 4 6

3: 2 4

4: 1 2 2 3

5: 6

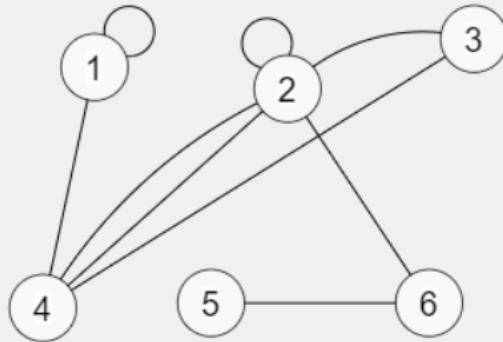
6: 2 5

Chú ý

- Đồ thị có thể có đa cung.
- Đồ thị có thể có khuyên.

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

Help Clear shift Delete Edit Undo Red Black



* Tự học - Vẽ đồ thị vô hướng - danh sách cung

Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

Đặt cờ

Vẽ đồ thị **vô hướng** có 5 đỉnh và gồm các cung sau:

1 2

1 2

1 5

2 5

2 5

3 3

3 4

Chú ý

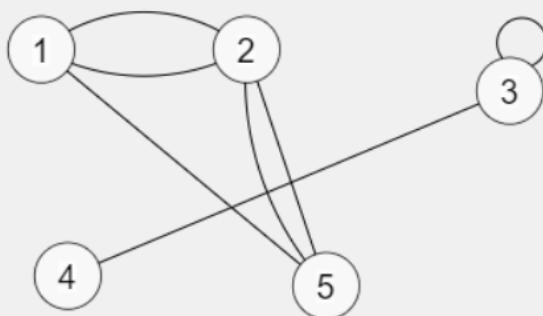
- Đồ thị có thể có đa cung.
- Đồ thị có thể có khuyên.

For example:

Test	Result
#test	1: 2 2 5
	2: 1 1 5 5
	3: 3 4
	4: 3
	5: 1 2 2

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

Help Clear shift Delete Edit Undo Red Black



* Tự học - Vẽ đồ thị có hướng - ma trận kề

Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

Đặt cờ

Vẽ đồ thị **có hướng** có 6 đỉnh và ma trận kề (mở rộng) như sau:

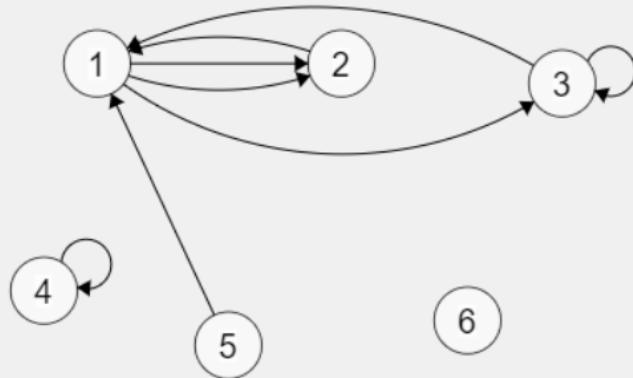
```
0 2 1 0 0 0
1 0 0 0 0 0
1 0 1 0 0 0
0 0 0 1 0 0
1 0 0 0 0 0
0 0 0 0 0 0
```

Chú ý

- Đồ thị có thể có đa cung. Nếu có x cung giữa đỉnh u và v thì ô $(u, v) = x$
- Đồ thị có thể có khuyên. Nếu có khuyên (u, u) , thì ô $(u, u) = 1$

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

Help Clear shift Delete Edit Undo Red Black



* Tự học - Vẽ đồ thị có hướng - danh sách đỉnh kề

Câu hỏi 1

Đúng

Đạt điểm 0,50
trên 1,00

Đặt cờ

Vẽ đồ thị **có hướng** có 5 đỉnh và danh sách các đỉnh kề của các đỉnh như sau:

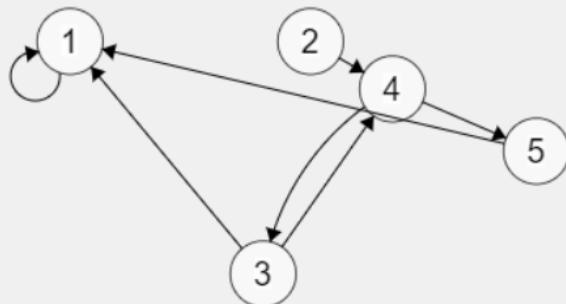
- 1: 1
- 2: 4
- 3: 1 4
- 4: 3 5
- 5: 1

Chú ý

- Đồ thị có thể có đa cung.
- Đồ thị có thể có khuyên.

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

Help Clear shift Delete Edit Undo Red Black



* Tự học - Vẽ đồ thị có hướng - danh sách cung

Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

Đặt cờ

Vẽ đồ thị **có hướng** có 6 đỉnh và gồm các cung sau:

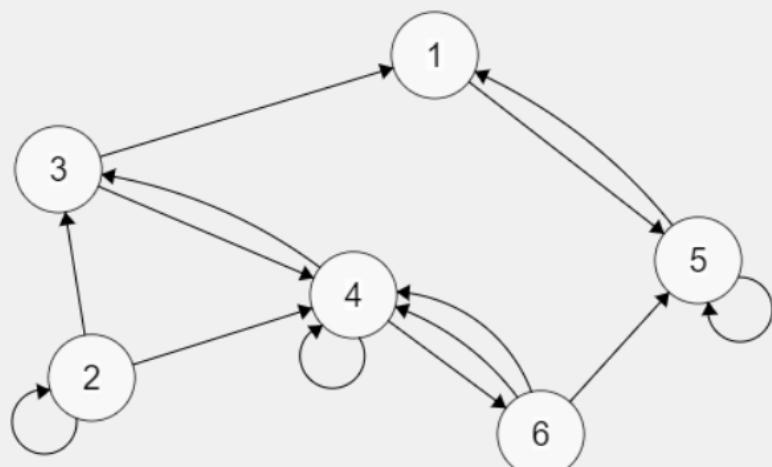
1 5
2 2
2 3
2 4
3 1
3 4
4 3
4 4
4 6
5 1
5 5
6 4
6 4
6 5

Chú ý

- Đồ thị có thể có đa cung.
- Đồ thị có thể có khuyên.

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

Help Clear shift Delete Edit Undo Red Black



* Tự học - Kiểm tra 2 đồ thị đẳng cấu với nhau (giống nhau về mặt cấu trúc)

Câu hỏi 1

Đúng

Đạt điểm 1.00
trên 1.00

Đặt cờ

Cho hai đồ thị G_1 và G_2 như hình vẽ. Hãy kiểm tra xem hai đồ thị này có đẳng cấu với nhau hay không. Nếu có, chỉ ra song ánh ứng nếu không, chỉ ra 1 lý do bất kỳ. Các lý do có thể là:

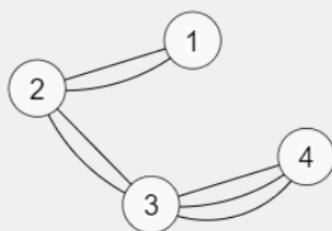
- Chuỗi bậc của G_2 khác chuỗi bậc của G_1 .
- Không tìm được song sánh.

Answer: (penalty regime: 75, 100, ... %)

Reset answer

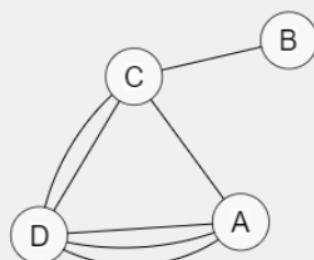
Đồ thị G_1

Help Clear shift Delete Edit Undo Red Black



Đồ thị G_2

Help Clear shift Delete Edit Undo Red Black



Chuỗi bậc

Chuỗi bậc (degree sequence): là danh sách các bậc của một đồ thị sắp xếp theo thứ tự giảm dần, ví dụ: 4, 4, 3, 2, 1

Tìm chuỗi bậc của G_1 & G_2

Chuỗi bậc của G_1 : 5,4,3,2

Chuỗi bậc của G_2 : 5,4,4,1

Lý do

- Chuỗi bậc G_1 & G_2 khác nhau.
 Không tìm được song sánh.

* Tự học - LTĐT cơ bản

Câu hỏi 1

Đúng

Đạt điểm 0,90
trên 1,00

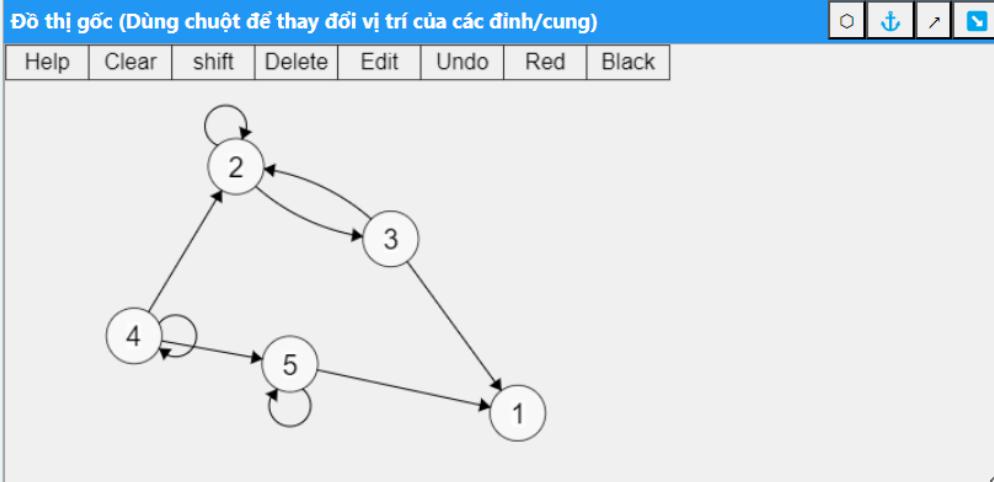
Đặt cờ

Cho đồ thị **có hướng** gồm 5 đỉnh và 9 cung như hình vẽ.

Hãy hoàn thành các bài tập bên dưới.

Answer: (penalty regime: 75, 100, ... %)

Reset answer



1. Loại đồ thị (Graph type)

Đồ thị trên thuộc loại nào?

- Đơn đồ thị có hướng (Simple directed graph)
- Đa đồ thị có hướng không khuyên (Directed multigraph)
- Đa đồ thị có hướng có khuyên (Quiver)

2. Kề nhau (adjacent)

Theo quy ước (xem slides): **đỉnh u "kề với" đỉnh v** (nhưng chưa chắc v "kề với" u) \Leftrightarrow có cung (u, v) .

Đánh dấu vào các câu đúng, bỏ trống các câu sai.

- 2 kề với 3.
- 3 kề với 3.
- 5 kề với 5.

3. Đỉnh kề/lân cận (adjacent vertices or neighbors)

Đỉnh v là đỉnh kề/lân cận (adjacent vertex/neighbor) của đỉnh u \Leftrightarrow u kề với v.

Liệt kê các đỉnh kề của các đỉnh bên dưới, ngăn cách nhau bằng dấu phẩy, bỏ qua các đỉnh trùng nhau.

- Các đỉnh kề của 1:
- Các đỉnh kề của 3:
- Các đỉnh kề của 2:

4. Bậc (degree)

- **Bậc vào của đỉnh u, kí hiệu $\deg^-(u)$, là số cung đi đến u.**
- **Bậc ra của đỉnh u, kí hiệu $\deg^+(u)$, là số cung đi ra khỏi u.**

Cho biết bậc vào và bậc ra của các đỉnh bên dưới.

- $\deg^-(5) =$
- $\deg^-(3) =$
- $\deg^-(2) =$

- $\deg^+(5) =$
- $\deg^+(3) =$
- $\deg^+(2) =$

Câu hỏi 2

Đúng

Đạt điểm 1,00
trên 1,00

Đặt cờ

Cho đồ thị **vô hướng** gồm **6** đỉnh và **8** cung như hình vẽ.

Hãy hoàn thành các bài tập bên dưới.

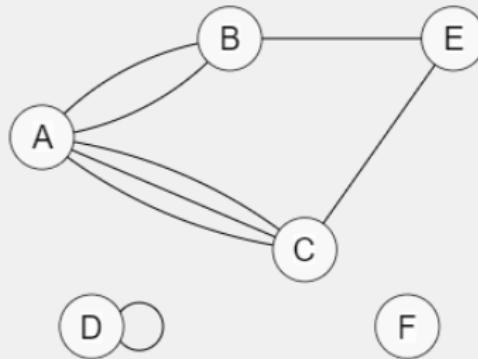
Answer: (penalty regime: 75, 100, ... %)

Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)



Help Clear shift Delete Edit Undo Red Black



1. Loại đồ thị (Graph type)

Đồ thị trên thuộc loại nào?

- Đơn đồ thị (Simple graph)
- Đa đồ thị (Multigraph)
- Giả đồ thị (Pseudograph)

2. Kề nhau (adjacent)

Hai đỉnh u và v được gọi là **kề nhau (adjacent) \Leftrightarrow có cung (u, v)** .

Đánh dấu vào các câu đúng, bỏ trống các câu sai.

- A và D kề nhau.
- F và D kề nhau.
- D và B kề nhau.

3. Đỉnh kề/lân cận (adjacent vertices or neighbors)

Đỉnh v là đỉnh kề/lân cận (adjacent vertex/neighbor) của đỉnh $u \Leftrightarrow u$ và v kề nhau.

Liệt kê các đỉnh kề của các đỉnh bên dưới, ngăn cách nhau bằng dấu phẩy, bỏ qua các đỉnh trùng nhau.

- Các đỉnh kề của E: B,C
- Các đỉnh kề của B: A,E
- Các đỉnh kề của D: D

B,C
A,E
D

4. Bậc (degree)

Bậc của đỉnh u , kí hiệu $\deg(u)$, là số cung liên thuộc với u (khuyên được tính 2 lần).

Cho biết bậc của các đỉnh bên dưới.

- $\deg(B) =$
- $\deg(C) =$
- $\deg(E) =$

3
4
2

Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

☛ Đặt cờ

Cho đồ thị **có hướng** gồm 5 đỉnh và 8 cung như hình vẽ.

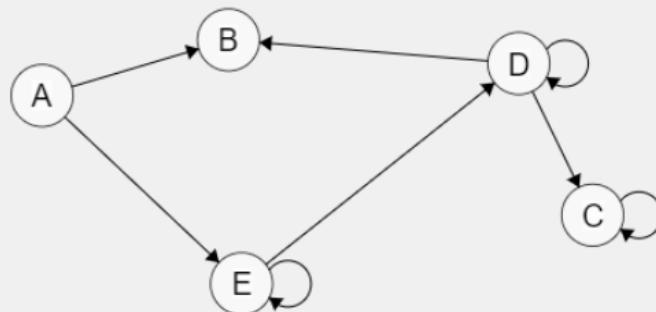
Hãy hoàn thành các bài tập bên dưới.

Answer: (penalty regime: 75, 100, ... %)

Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo Red Black

**1. Loại đồ thị (Graph type)**

Đồ thị trên thuộc loại nào?

- Đơn đồ thị có hướng (Simple directed graph)
- Đa đồ thị có hướng không khuyên (Directed multigraph)
- Đa đồ thị có hướng có khuyên (Quiver)

2. Kề nhau (adjacent)*Theo quy ước (xem slides): đỉnh u "kề với" đỉnh v (nhưng chưa chắc v "kề với" u) \Leftrightarrow có cung (u, v) .*

Đánh dấu ✓ vào các câu đúng, bỏ trống các câu sai.

- E kề với E.
- B kề với E.
- A kề với A.

3. Đỉnh kề/lân cận (adjacent vertices or neighbors)*Đỉnh v là đỉnh kề/lân cận (adjacent vertex/neighbor) của đỉnh u \Leftrightarrow u kề với v.*

Liệt kê các đỉnh kề của các đỉnh bên dưới, ngăn cách nhau bằng dấu phẩy, bỏ qua các đỉnh trùng nhau.

- Các đỉnh kề của B: _____
- Các đỉnh kề của E: _____
- Các đỉnh kề của D: _____

E,D
B,C,D

4. Độ (degree)

- *Bậc vào của đỉnh u, kí hiệu $\deg^-(u)$, là số cung đi đến u.*
- *Bậc ra của đỉnh u, kí hiệu $\deg^+(u)$, là số cung đi ra khỏi u.*

Cho biết bậc vào và bậc ra của các đỉnh bên dưới.

- $\deg^-(A) =$ _____
- $\deg^-(B) =$ _____
- $\deg^-(C) =$ _____

0
2
2

- $\deg^+(A) =$ _____
- $\deg^+(B) =$ _____
- $\deg^+(C) =$ _____

2
0
1

Câu hỏi 2

Đúng

Đạt điểm 0,90
trên 1,00

☛ Đặt cờ

Cho đồ thị **vô hướng** gồm **6** đỉnh và **9** cung như hình vẽ.

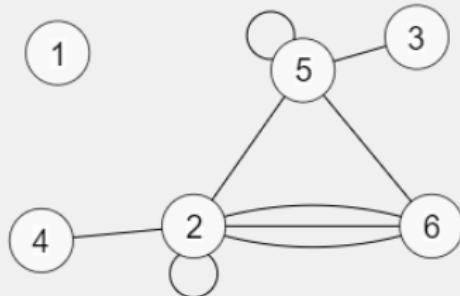
Hãy hoàn thành các bài tập bên dưới.

Answer: (penalty regime: 75, 100, ... %)

Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo Red Black

**1. Loại đồ thị (Graph type)**

Đồ thị trên thuộc loại nào?

- Đơn đồ thị (Simple graph)
- Đa đồ thị (Multigraph)
- Giả đồ thị (Pseudograph)

2. Kề nhau (adjacent)

Hai đỉnh u và v được gọi là kề nhau (adjacent) \Leftrightarrow có cung (u, v).

Đánh dấu ✓ vào các câu đúng, bỏ trống các câu sai.

- 4 và 2 kề nhau.
- 2 và 6 kề nhau.
- 3 và 5 kề nhau.

3. Đỉnh kề/lân cận (adjacent vertices or neighbors)

Đỉnh v là đỉnh kề/lân cận (adjacent vertex/neighbor) của đỉnh u \Leftrightarrow u và v kề nhau.

Liệt kê các đỉnh kề của các đỉnh bên dưới, ngăn cách nhau bằng dấu phẩy, bỏ qua các đỉnh trùng nhau.

- Các đỉnh kề của 4:
- Các đỉnh kề của 2:
- Các đỉnh kề của 5:

4. Độ (degree)

Bộ của đỉnh u, kí hiệu $\deg(u)$, là số cung liên thuộc với u (khuyên được tính 2 lần).

Cho biết bộ của các đỉnh bên dưới.

- $\deg(6) =$
- $\deg(4) =$
- $\deg(5) =$

Tuần 2 - Duyệt đồ thị

* Bài tập lý thuyết: DFS - Duyệt đồ thị theo chiều sâu dùng ngăn xếp

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
⚐ Đặt cờ

Cho đồ thị có hướng gồm 6 đỉnh và 8 cung như bên dưới.

Hãy áp dụng thuật toán duyệt đồ thị theo chiều sâu sử dụng ngăn xếp (stack) để duyệt đồ thị trên, bắt đầu từ đỉnh A. Với mỗi bước lặp, cho biết đỉnh nào được lấy ra khỏi stack, có làm gì trên đỉnh đó không, những đỉnh nào sẽ được thêm vào stack, nội dung của stack. Ghi các thông tin này vào bảng như sau:

- Cột **u**, ghi đỉnh được lấy ra từ đỉnh stack
- Cột **Duyệt/bỏ qua**, nếu duyệt **u** thì ghi thứ tự của đỉnh được duyệt, thứ tự duyệt tính từ 1. Nếu đỉnh này đã duyệt rồi ghi **bỏ qua**
- Cột **Các đỉnh kề của u**, liệt kê **các đỉnh kề chưa được duyệt của u** theo thứ tự nhỏ đến lớn, ngăn cách với nhau bằng dấu phẩy, ví dụ: A,B,D
- Cột **Stack**, liệt kê các đỉnh đang có trong stack, ngăn cách nhau bằng dấu phẩy (đỉnh stack ở phía bên phải), vd: B,C,E. Đỉnh E đang nằm trên đỉnh stack

Quy ước

- Đỉnh stack nằm phía tay PHẢI.
- Sử dụng thuật toán DFS phiên bản 2 bài tập lý thuyết: đỉnh đã được duyệt sẽ không được thêm vào ngăn xếp nữa.
- Do bản chất của thuật toán, một đỉnh có thể ở trong ngăn xếp nhiều lần.

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

Reset answer

Đồ thị gốc

Áp dụng thuật toán duyệt đồ thị theo chiều sâu sử dụng stack và ghi kết quả vào bảng:

	u	Duyệt/bỏ qua	Các đỉnh kề của u	Stack
Khởi tạo				A
1	A	1	C, E	C, E
2	E	2	D	C, D
3	D	3	B	C, B
4	B	4	F	C, F
5	F	5		C
6	C	6		
7				
8				
...				

Add row Delete row

Câu hỏi 1

Đúng

Đạt điểm 0.88
trên 1.00

Đặt cờ

Cho đồ thị **vô hướng** gồm 5 đỉnh và 6 cung như bên dưới.

Hãy áp dụng **thuật toán duyệt đồ thị theo chiều sâu sử dụng ngăn xếp (stack)** để duyệt đồ thị trên, bắt đầu từ đỉnh **C**. Với mỗi bước lặp, cho biết đỉnh nào được lấy ra khỏi stack, có làm gì trên đỉnh đó không, những đỉnh nào sẽ được thêm vào stack, nội dung của stack. Ghi các thông tin này vào bảng như sau:

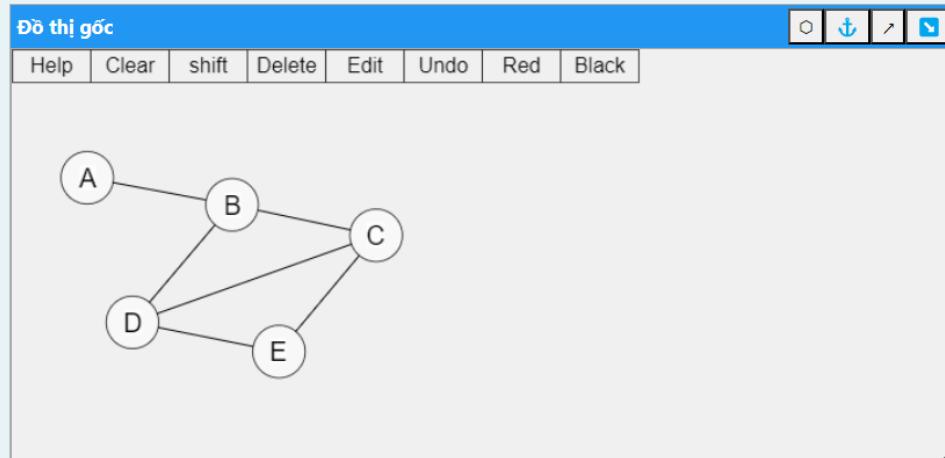
- Cột **u**, ghi đỉnh được lấy ra từ đỉnh stack
- Cột **Duyệt/bỏ qua**, nếu duyệt **u** thì ghi thứ tự của đỉnh được duyệt, thứ tự duyệt tính từ 1. Nếu đỉnh này đã duyệt rồi ghi **bỏ qua**
- Cột **Các đỉnh kề của u**, liệt kê **các đỉnh kề chưa được duyệt của u** theo thứ tự nhỏ đến lớn, ngăn cách với nhau bằng dấu phẩy, ví dụ: A,B,D
- Cột **stack**, liệt kê các đỉnh đang có trong stack, ngăn cách nhau bằng dấu phẩy (đỉnh stack ở phía bên phải), vd: B,C,E. Đỉnh E đang nằm trên đỉnh stack

Quy ước

- Đỉnh stack nằm phía tay PHẢI.
- Sử dụng thuật toán DFS phiên bản 2 bài tập lý thuyết: đỉnh đã được duyệt sẽ không được thêm vào ngăn xếp nữa.
- Do bản chất của thuật toán, một đỉnh có thể ở trong ngăn xếp nhiều lần.

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

Reset answer



Áp dụng thuật toán duyệt đồ thị theo chiều sâu sử dụng stack và ghi kết quả vào bảng:

	u	Duyệt/bỎ qua	Các đỉnh kề của u	Stack
Khởi tạo				C
1	C	1	B, D, E	B, D, E
2	E	2	D	B, D, D
3	D	3	B	B, D, B
4	B	4	A	B, D, A
5	A	5		B, D
6	D	bỎ qua		B
...	B	bỎ qua		

Add row Delete row

* Bài tập lý thuyết: DFS - Duyệt đồ thị theo chiều sâu dùng đệ quy

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
Đặt cờ

Xem 1 thuật toán đệ quy duyệt đồ thị theo chiều sâu (**kiểm tra ngoài vòng lặp**) như sau:

```
void DFS(int u) {
    //1. Xét u đã duyệt chưa
    if (mark[u] == 1) //nếu u đã duyệt
        return; //ta. Bỏ qua

    mark[u] = 1; //1b. Duyệt u và

    //2. Xét các đỉnh kề chưa duyệt của u
    for (v là các đỉnh kề chưa duyệt của u)
        DFS(v); //Gọi đệ quy duyệt v
}
```

Thuật toán trên gồm 2 bước chính:

1. Xét u đã duyệt hay chưa? Nếu đã duyệt, bấm "1a. Bỏ qua". Ngược lại, bấm "1b. Duyệt".
2. Nếu ở bước 1 chọn "1b. Duyệt" thì làm tiếp bước 2, nếu không bỏ qua bước 2 (vì đã return). Ở bước 2, lần lượt xét từng đỉnh kề chưa được duyệt của u để gọi đệ quy duyệt nó.
 - o Hãy liệt kê các đỉnh kề chưa được duyệt của u vào ô "**Với các đỉnh kề chưa duyệt của u**", ngăn cách nhau bằng dấu phẩy. Nếu u không có đỉnh kề hoặc các đỉnh kề của u đều đã được duyệt thì để trống.
 - o Bấm nút "Xét".

Chỗ đồ thị **vô hướng** gồm 5 đỉnh và 8 cung như bên này bên dưới. Hãy áp dụng thuật toán đệ quy trên để duyệt đồ thị đã cho bắt đầu từ đỉnh **B**.

DFS(B);

Dựa vào kết quả duyệt đồ thị, vẽ cây duyệt đồ thị theo chiều sâu. Cây duyệt đồ thị bao gồm tất cả các đỉnh của đồ thị gốc và các cung (u, v) với u là đỉnh gọi DFS(v).

Quy ước

- Mỗi thี hiện của hàm **DFS(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.
- Hay làm bài theo thứ tự từ ngoài vào và từ trên xuống dưới.
- Nếu làm sai 1 bước, có thể bấm "**Lùi lại 1 bước**" để lùi lại bước trước đó.
- Liệt kê các đỉnh theo thứ tự **A, B, C, ...**

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

Đặt cờ

Thực hiện duyệt đệ quy theo chiều sâu bắt đầu từ đỉnh B
Lùi lại 1 bước Số bước: 14

DFS(B)

1. Nếu B đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. Bỏ qua 1b. Duyệt 1
2. Với các đỉnh kề chưa duyệt của B: C,D,E

Xét 2

DFS(C)

1. Nếu C đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. Bỏ qua 1b. Duyệt 3
2. Với các đỉnh kề chưa duyệt của C: A,D

Xét 4

DFS(A)

1. Nếu A đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt 5
2. Với các đỉnh kề chưa duyệt của A: D,E

Xét 6

DFS(D)

1. Nếu D đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt 7
2. Với các đỉnh kề chưa duyệt của D: E

Xét 8

DFS(E)

1. Nếu E đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt 9
2. Với các đỉnh kề chưa duyệt của E:

Xét 10

DFS(E)

1. Nếu E đã duyệt => BỎ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt 11
2. Với các đỉnh kề chưa duyệt của E:

Xét

DFS(D)

1. Nếu D đã duyệt => BỎ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt 12
2. Với các đỉnh kề chưa duyệt của D:

Xét

DFS(B)

1. Nếu B đã duyệt => BỎ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt 13
2. Với các đỉnh kề chưa duyệt của B:

Xét

DFS(E)

1. Nếu E đã duyệt => BỎ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt 14
2. Với các đỉnh kề chưa duyệt của E:

Xét

Vẽ cây duyệt đồ thị theo chiều sâu

Help Clear shift Delete Edit Undo Red Black

Câu hỏi 1
Đúng
Đạt điểm 1,00
Đặt cờ

Xét 1 thuật toán đệ quy duyệt đồ thị theo chiều sâu (**kiểm tra ngoài vòng lặp**) như sau:

```
void DFS(int u) {
    //1. Xét u đã duyệt chưa
    if (mark[u] == 1) //nếu u đã duyệt
        return; //1a. Bỏ qua

    mark[u] = 1; //1b. Duyệt u và

    //2. Xét các đỉnh kề chưa duyệt của u
    for (v là các đỉnh kề chưa duyệt của u)
        DFS(v); //đoạn đệ quy duyệt v
}
```

Thuật toán gồm 2 bước chính:

1. Xét u đã duyệt hay chưa? Nếu đã duyệt, bấm "**1a. Bỏ qua**". Ngược lại, bấm "**1b. Duyệt**".
2. Nếu ở bước 1 chọn "**1b. Duyệt**" thì làm tiếp bước 2, nếu không bỏ qua bước 2 (vì đã return). Ở bước 2, lần lượt xét từng đỉnh kề chưa được duyệt của u để gọi đệ quy duyệt nó.
 - Hãy liệt kê các đỉnh kề **chưa được duyệt** của u vào ô "**Với các đỉnh kề chưa duyệt của u**", ngăn cách nhau bằng dấu phẩy. Nếu u không có đỉnh kề hoặc các đỉnh kề của u đều đã được duyệt thì để trống.
 - Bấm nút "**Xét**".

Cho đồ thị **có hướng** gồm **6** đỉnh và **9** cung như bên dưới. Hãy áp dụng thuật toán đệ quy trên để duyệt đồ thị đã cho bắt đầu từ đỉnh **3**.

DFS(3);

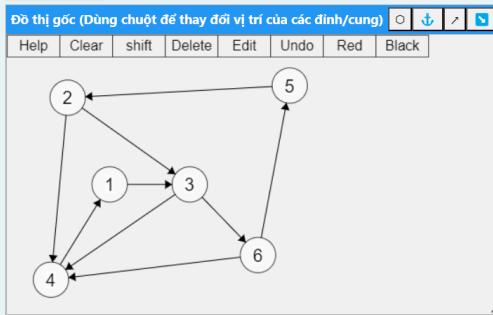
Dựa vào kết quả duyệt đồ thị, vẽ cây duyệt đồ thị theo chiều sâu. Cây duyệt đồ thị bao gồm tất cả cá đỉnh của đồ thị gốc và các cung (u, v) với u là đỉnh gọi DFS(v).

Quy ước

- Mỗi thể hiện của hàm **DFS(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, ...**

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

Reset answer



Thực hiện duyệt đệ quy theo chiều sâu bắt đầu từ đỉnh 3

Lùi lại 1 bước Số bước: **12**

DFS(3)

1. Nếu 3 đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. Bỏ qua **1b. Duyệt** **✓ 1**

2. Với các đỉnh kề chưa duyệt của 3: 4,6

Xét **✓ 2**

DFS(4)

1. Nếu 4 đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. Bỏ qua **1b. Duyệt** **✓ 3**

2. Với các đỉnh kề chưa duyệt của 4: 1

✓ 4

DFS(1)

1. Nếu 1 đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. Bỏ qua **1b. Duyệt** **✓ 5**

2. Với các đỉnh kề chưa duyệt của 1:

Xét **✓ 6**

DFS(6)

1. Nếu 6 đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. BỎ QUÁ **1b. DUYỆT** **✓ 7**

2. Với các đỉnh kề chưa duyệt của 6: 5

✓ 8

DFS(5)

1. Nếu 5 đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. BỎ QUÁ **1b. DUYỆT** **✓ 9**

2. Với các đỉnh kề chưa duyệt của 5: 2

Xét **✓ 10**

DFS(2)

1. Nếu 2 đã duyệt => Bỏ qua, ngược lại => Duyệt

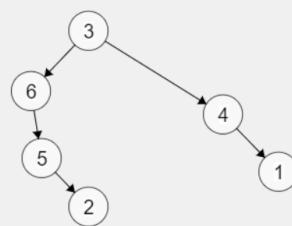
1a. BỎ QUÁ **1b. DUYỆT** **✓ 11**

2. Với các đỉnh kề chưa duyệt của 2:

Xét **✓ 12**

Vẽ cây duyệt đồ thị theo chiều sâu

Help Clear shift Delete Edit Undo Red Black



* Bài tập lý thuyết: BFS - Duyệt đồ thị theo chiều rộng dùng hàng đợi

Câu hỏi 1
Đúng
Đạt điểm 0,90
trên 1,00
☞ Đặt cờ

Cho đồ thị **có hướng** gồm 7 đỉnh và 12 cung như bên dưới.

Hãy áp dụng thuật toán duyệt đồ thị theo chiều rộng sử dụng **hàng đợi** để duyệt đồ thị trên, bắt đầu từ đỉnh **E**. Với mỗi bước lặp, cho biết đỉnh nào được lấy ra khỏi hàng đợi, có làm gì trên đỉnh đó không, những đỉnh nào sẽ được thêm vào hàng đợi, nội dung của hàng đợi. Ghi các thông tin này vào bảng như sau:

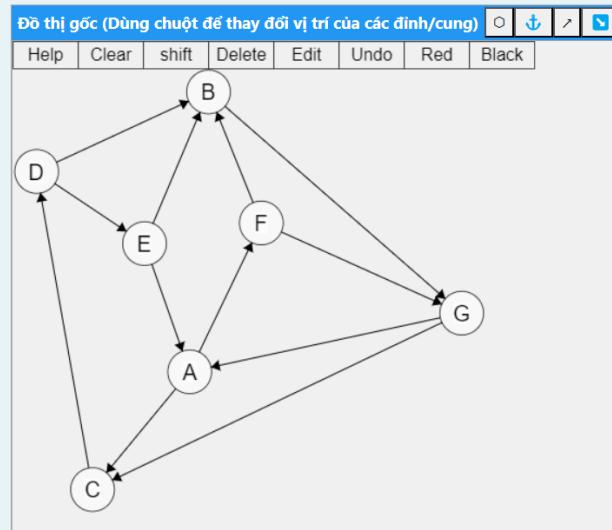
- Cột **u**, ghi đỉnh được lấy ra từ đầu hàng đợi
- Cột **Duyệt/bỏ qua**, nếu duyệt **u** thì ghi thứ tự của đỉnh được duyệt, thứ tự duyệt tính từ 1. Nếu đỉnh này đã duyệt rồi ghi **bỏ qua**.
- Cột **Các đỉnh kề của u**, liệt kê **các đỉnh kề chưa được duyệt của u** theo thứ tự nhô đến lớn, ngăn cách với nhau bằng dấu phẩy, ví dụ: A,B,D
- Cột **hàng đợi**, liệt kê các đỉnh đang có trong hàng đợi, ngăn cách nhau bằng dấu phẩy (đỉnh đầu hàng đợi nằm ở phía bên TRÁI), ví dụ: B,C,E. Đỉnh B đang nằm đầu hàng đợi.

Quy ước

- Đầu hàng đợi nằm phía tay TRÁI
- Do bản chất của thuật toán, một đỉnh chưa duyệt có thể có mặt nhiều lần trong hàng đợi.

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

Reset answer



Áp dụng thuật toán duyệt đồ thị theo chiều rộng sử dụng hàng đợi và ghi kết quả vào bảng:

	u	Duyệt/bỎ qua	Các đỉnh kề của u	Hàng đợi
Khởi tạo				E
1	E	1	A, B	A, B
2	A	2	C, F	B, C, F
3	B	3	G	C, F, G
4	C	4	D	F, G, D
5	F	5	G	G, D, G
6	G	6		D, G
7	D	7		G
8	G	bỎ qua		

Câu hỏi 1

Đúng

Đạt điểm 0,90
trên 1,00

☞ Đặt cờ

Cho đồ thị **vô hướng** gồm **6** đỉnh và **6** cung như bên dưới.

Hãy áp dụng **thuật toán duyệt đồ thị theo chiều rộng sử dụng hàng đợi** để duyệt đồ thị trên, bắt đầu từ đỉnh **5**. Với mỗi bước lặp, cho biết đỉnh nào được lấy ra khỏi hàng đợi, có làm gì trên đỉnh đó không, những đỉnh nào sẽ được thêm vào hàng đợi, nội dung của hàng đợi. Ghi các thông tin này vào bảng như sau:

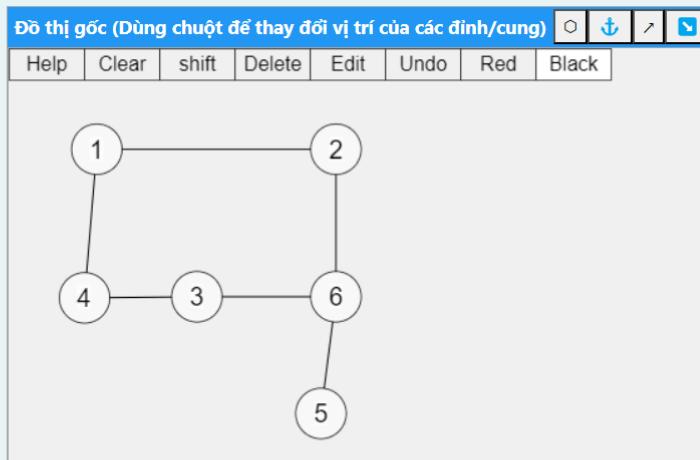
- Cột **u**, ghi đỉnh được lấy ra từ đầu hàng đợi
- Cột **Duyệt/bỏ qua**, nếu duyệt u thì ghi thứ tự của đỉnh được duyệt, thứ tự duyệt tính từ 1. Nếu đỉnh này đã duyệt rồi ghi **bỏ qua**.
- Cột **Các đỉnh kề của u**, liệt kê **các đỉnh kề chưa được duyệt của u** theo thứ tự nhỏ đến lớn, ngăn cách với nhau bằng dấu phẩy, ví dụ: 1,2,4
- Cột **hàng đợi**, liệt kê các đỉnh đang có trong hàng đợi, ngăn cách nhau bằng dấu phẩy (đỉnh đầu hàng đợi nằm ở phía bên TRÁI), ví dụ: 2,3,5. Đỉnh 2 đang nằm đầu hàng đợi.

Quy ước

- Đầu hàng đợi nằm phía tay TRÁI
- Do bản chất của thuật toán, một đỉnh chưa duyệt có thể có mặt nhiều lần trong hàng đợi.

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

Reset answer



Áp dụng thuật toán duyệt đồ thị theo chiều rộng sử dụng hàng đợi và ghi kết quả vào bảng:

	u	Duyệt/bỎ qua	Các đỉnh kề của u	Hàng đợi
Khởi tạo				5
1	5	1	6	6
2	6	2	2,3	2,3
3	2	3	1	3,1
4	3	4	4	1,4
5	1	5	4	4,4
6	4	6		4
...	4	bỎ qua		

Add row | Delete row

* DFS đệ quy (2 phiên bản: kiểm tra NGOÀI + TRONG vòng for)

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
Đặt cờ

Xét 1 thuật toán đệ quy duyệt đồ thị theo chiều sâu (**kiểm tra ngoài vòng lặp**) như sau:

```
void DFS(int u) {
    //1. Xét u đã duyệt chưa
    if (mark[u] == 1) //nếu u đã duyệt
        return; //1a. Bỏ qua

    mark[u] = 1; //1b. Duyệt u và

    //2. Xét các đỉnh kề chưa duyệt của u
    for (v là các đỉnh kề chưa duyệt của u)
        DFS(v); //Gọi đệ quy duyệt v
}
```

Thuật toán trên gồm 2 bước chính:

1. Xét u đã duyệt hay chưa? Nếu đã duyệt, bấm "**1a. Bỏ qua**". Ngược lại, bấm "**1b. Duyệt**".
2. Nếu ở bước 1 chọn "**1b. Duyệt**" thì làm tiếp bước 2, nếu không bỏ qua bước 2 (vì đã return). Ở bước 2, lần lượt xét từng đỉnh kề chưa được duyệt của u để gọi đệ quy duyệt nó.
 - Hãy liệt kê các đỉnh kề **chưa được duyệt** của u vào ô "**Với các đỉnh kề chưa duyệt của u**", ngăn cách nhau bằng dấu phẩy. Nếu u không có đỉnh kề hoặc các đỉnh kề của u đều đã được duyệt thì để trống.
 - Bấm nút "**Xét**".

Cho đồ thị **có hướng** gồm 5 đỉnh và 9 cung như bên dưới. Hãy áp dụng thuật toán đệ quy trên để duyệt đồ thị đã cho bắt đầu từ đỉnh 4.

[DFS\(4\);](#)

Dựa vào kết quả duyệt đồ thị, vẽ cây duyệt đồ thị theo chiều sâu. Cây duyệt đồ thị bao gồm tất cả các đỉnh của đồ thị gốc và các cung (u, v) với u là đỉnh gọi DFS(v).

Quy ước

- Mỗi thể hiện của hàm **DFS(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, ...**

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

[Reset answer](#)

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo Red Black

Thực hiện duyệt đệ quy theo chiều sâu bắt đầu từ đỉnh 4

[Lùi lại 1 bước](#) Số bước: 11

DFS(4)

1. Nếu 4 đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. Bỏ qua 1b. Duyệt ✓ 1
2. Với các đỉnh kề chưa duyệt của 4: 1,3,5 [Xét](#) ✓ 2

DFS(1)

1. Nếu 1 đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. Bỏ qua 1b. Duyệt ✓ 3
2. Với các đỉnh kề chưa duyệt của 1: 2 [Xét](#) ✓ 4

DFS(2)

1. Nếu 2 đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt ✓ 5
2. Với các đỉnh kề chưa duyệt của 2: [Xét](#) ✓ 6

DFS(3)

1. Nếu 3 đã duyệt => Bỏ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt ✓ 7
2. Với các đỉnh kề chưa duyệt của 3: 5 [Xét](#) ✓ 8

DFS(5)

1. Nếu 5 đã duyệt => BỎ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt ✓ 9
2. Với các đỉnh kề chưa duyệt của 5: [Xét](#) ✓ 10

DFS(5)

1. Nếu 5 đã duyệt => BỎ qua, ngược lại => Duyệt

1a. BỎ qua 1b. Duyệt ✗ 11
2. Với các đỉnh kèle chưa duyệt của 5: [Xét](#)

Vẽ cây duyệt đồ thị theo chiều sâu

Help Clear shift Delete Edit Undo Red Black

Câu hỏi 2
Đúng
Đạt điểm 1,00
trên 1,00
Đặt cờ

Xét thuật toán duyệt đồ thị theo chiều sâu sử dụng đệ quy (**kiểm tra trong vòng lặp**) như bên dưới.

```
void DFS(int u) {
    //1. Đánh dấu u đã duyệt
    mark[u] = 1;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u)
        if (v chưa duyệt) { // mark[v] == 0
            //2a. Gọi đệ quy duyệt v
            DFS(v);
        } else { //mark[v] == 1
            //2b. Bỏ qua
        }
}
```

Thuật toán duyệt này gồm 2 bước chính:

1. Duyệt (ví dụ in ra màn hình) và đánh dấu u đã duyệt.
2. Xét các đỉnh kề v của u, có 2 trường hợp xảy ra:
 - o v chưa duyệt => gọi đệ quy duyệt v.
 - o v đã duyệt => bỏ qua

Khởi tạo tất cả các đỉnh chưa duyệt.

Cho đồ thị **vô hướng** gồm 6 đỉnh và 9 cung như bên như bên dưới. Hãy áp dụng thuật toán duyệt đồ thị bên trên để duyệt G bắt đầu từ đỉnh C.

```
DFS();
```

Quy ước

- Mỗi thể hiện của hàm **DFS(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, ...**

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

Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

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

Thực hiện duyệt đệ quy theo chiều sâu bắt đầu từ đỉnh C

Lùi lại 1 bước Số bước: 30

DFS(C)

1. Đánh dấu C đã duyệt.

Đánh dấu 1

2. Với các đỉnh kề v của C: A,B,D,E,F

Xét 2

2a. v = A, chưa duyệt => Gọi đệ quy duyệt. Duyệt 3

DFS(A)

1. Đánh dấu A đã duyệt.

Đánh dấu 4

2. Với các đỉnh kề v của A: B,C

Xét 5

2a. v = B, chưa duyệt => Gọi đệ quy duyệt. Duyệt 6

DFS(B)

1. Đánh dấu B đã duyệt.

Đánh dấu 7

2. Với các đỉnh kề v của B: A,C,D

8

2b. v = A, đã duyệt rồi => Bỏ qua thời. Bỏ qua 9

2b. v = C, đã duyệt rồi => Bỏ qua thời. BỎ QUÁ 10

2a. v = D, chưa duyệt => Gọi đệ quy duyệt. Duyệt 11

DFS(D)

1. Đánh dấu D đã duyệt.

Đánh dấu 12

2. Với các đỉnh kề v của D: B,C,E

Xét 13

2b. v = B, đã duyệt rồi => Bỏ qua thời. BỎ QUÁ 14

2b. v = C, đã duyệt rồi => Bỏ qua thời. BỎ QUÁ 15

2a. v = E, chưa duyệt => Gọi đệ quy duyệt. Duyệt 16

DFS(E)

1. Đánh dấu E đã duyệt.

Đánh dấu 17

2. Với các đỉnh kề v của E: C,D,F

Xét 18

2b. v = C, đã duyệt rồi => BỎ QUÁ 19

2b. v = D, đã duyệt rồi => BỎ QUÁ 20

2a. v = F, chưa duyệt => Gọi đệ quy duyệt. Duyệt 21

DFS(F)

1. Đánh dấu F đã duyệt.

Đánh dấu 22

2. Với các đỉnh kề v của F: C,E

23

2b. v = C, đã duyệt rồi => BỎ QUÁ 24

2b. v = E, đã duyệt rồi => BỎ QUÁ 25

2b. v = C, đã duyệt rồi => BỎ QUÁ 26

2b. v = B, đã duyệt rồi => BỎ QUÁ 27

2b. v = D, đã duyệt rồi => BỎ QUÁ 28

2b. v = E, đã duyệt rồi => BỎ QUÁ 29

2b. v = F, đã duyệt rồi => BỎ QUÁ 30

Vẽ cây duyệt đồ thị theo chiều sâu

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

Tuần 3 - Tính liên thông của đồ thị

* Tự học - Áp dụng thuật toán Tarjan tìm các BPLT của đồ thị có hướng (ví dụ)

Câu hỏi 1
Đúng
Điểm 0.00
trên 1.00
T⁺ 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
    min_num[u] = k; min_num[u] = 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) {
            //Lưu 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) {
            //B2. 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:

- 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.
 - Xét các đỉnh kề 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
 - 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 dưới. Hãy áp dụng thuật toán Tarjan để tìm các thành phần liên thông mạnh của G bắt đầu từ đỉnh 1.

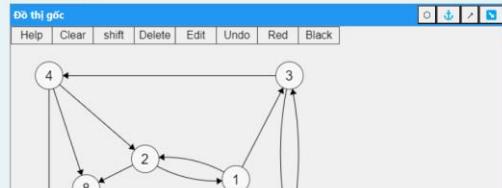
Dựa vào kết quả của thuật toán, về các thành phần liên thông tìm được. Với mỗi thành phần liên thông, vẽ các cung và các cung bén trong thành phần liên thông này. Nói cách khác, xoá bỏ các cung của đồ thị gốc mà 2 đỉnh của nó nằm ở 2 thành phần liên thông khác nhau.

Quy ước

- Mỗi thê hiện của hàm SCC(u) được minh họa bằng một khung chữ nhật. Việc gọi đệ quy được minh họa bằng một khung chữ nhật nhỏ hơn bên trong.
- Hay làm bài theo thứ tự từ ngoài vào trong và từ trên xuống dưới.
- Nếu lầm sai 1 bước, đó là bấm "Lùi lại 1 bước" để làm lại bước trước đó.
- Liệt kê các cách 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

Để lùi lại 1 bước | Số bước: 52

SCC(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 [X] 1
- Với các đỉnh kề v của 1: 2, 3
2a. v = '2' chưa duyệt [Duyệt nô] [X] 2

SCC(2) [---] [---]

 - 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 [X] 4
 - Với các đỉnh kề v của 2: 1, 3
2b. v = '1' duyệt rồi nhưng vẫn còn trên stack [Cập nhật min_num[u]] [X] 6
Cập nhật min_num[2] = min(min_num[2], num[1]) = 1 [Cập nhật] [X] 7

2a. v = '3' chưa duyệt [Duyệt nô] [X] 8

SCC(3) [---] [---]

 - Đánh số cho đỉnh 3 và đưa nó vào ngăn xếp S.
Gán num[3] = min_num[3] = k = 3 ; k++;
Đưa 3 vào S. Kết quả: S = 1, 2, 3
Thực hiện đánh số và đưa vào Stack [X] 9
 - Với các đỉnh kề v của 3: 4
2b. v = '4' chưa duyệt [Duyệt nô] [X] 10

2a. v = '7' chưa duyệt [Duyệt nô] [X] 11

SCC(7) [---] [---]

- 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, 3, 4
Thực hiện đánh số và đưa vào Stack [X] 12
- Với các đỉnh kề v của 7: 6, 8
2a. v = '6' chưa duyệt [Duyệt nô] [X] 13

SCC(6) [---] [---]

 - 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, 3, 4, 5
Thực hiện đánh số và đưa vào Stack [X] 15
 - Với các đỉnh kề v của 6: 7
2b. v = '7' duyệt rồi nhưng vẫn còn trên stack [Cập nhật min_num[u]] [X] 17
Cập nhật min_num[6] = min(min_num[6], num[7]) = 4 [Cập nhật] [X] 18

3. Kiểm tra num và min_num của 6
num[6] <= min_num[6] | Gán num[6] = min_num[6] [X] 19

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

3. Kiểm tra num và min_num của 7
num[7] <= min_num[7] | Gán num[7] = min_num[7] [X] 21

Lấy các đỉnh trong ngăn xếp ra cho đến khi lũy được 7
Các đỉnh được lũy ra: 4, 5, 6, 7
Nội dung còn lại của ngăn xếp: 1, 2, 3
[Cập nhật] [X] 22

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

3. Kiểm tra num và min_num của 8
num[8] <= min_num[8] | Gán num[8] = min_num[8] [X] 24

Lấy các đỉnh trong ngăn xếp ra cho đến khi lũy được 8
Các đỉnh được lũy ra: 8
Nội dung còn lại của ngăn xếp: 1, 2
[Cập nhật] [X] 25

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

3. Kiểm tra num và min_num của 2
num[2] <= min_num[2] | Gán num[2] = min_num[2] [X] 27

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

2a. v = '3' chưa duyệt [Duyệt nô] [X] 29

SCC(3) [---] [---]

 - 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 [X] 30
 - Với các đỉnh kề v của 3: 4, 5
2a. v = '4' chưa duyệt [Duyệt nô] [X] 32

SCC(4) [---] [---]

 - 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 [X] 33
 - Với các đỉnh kề v của 4: 2, 7, 8
2b. v = '2' chưa duyệt rồi nhưng vẫn còn trên stack [Cập nhật min_num[u]] [X] 35
Cập nhật min_num[4] = min(min_num[4], num[2]) = 2 [Cập nhật] [X] 36

2c. v = '7' chưa duyệt rồi và không còn trên stack [X] 37

2c. v = '8' chưa duyệt rồi và không còn trên stack [X] 38

3. Kiểm tra num và min_num của 4
num[4] <= min_num[4] | Gán num[4] = min_num[4] [X] 39

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

2a. v = '5' chưa duyệt [Duyệt nô] [X] 41

SCC(5) [---] [---]

 - 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 [X] 42
 - Với các đỉnh kề v của 5: 6, 7
2a. v = '6' chưa duyệt [Duyệt nô] [X] 43

SCC(6) [---] [---]

 - Đánh số cho đỉnh 6 và đưa nó vào ngăn xếp S.
Gán num[6] = min_num[6] = k = 9 ; k++;
Đưa 6 vào S. Kết quả S = 1, 2, 3, 4, 5, 6
Thực hiện đánh số và đưa vào Stack [X] 44
 - Với các đỉnh kề v của 6: 7
2b. v = '7' chưa duyệt rồi và không còn trên stack [Cập nhật min_num[u]] [X] 46
Cập nhật min_num[6] = min(min_num[6], num[7]) = 6 [Cập nhật] [X] 47

2c. v = '5' chưa duyệt rồi và không còn trên stack [X] 48

3. Kiểm tra num và min_num của 6
num[6] <= min_num[6] | Gán num[6] = min_num[6] [X] 49

Cập nhật min_num[5] = min(min_num[5], min_num[6]) = 2 [Cập nhật] [X] 50

3. Kiểm tra num và min_num của 1
num[1] <= min_num[1] | Gán num[1] = min_num[1] [X] 51

Lấy các đỉnh trong ngăn xếp ra cho đến khi lũy được 1
Các đỉnh được lũy ra: 5, 4, 3, 2, 1
Nội dung còn lại của ngăn xếp:
[Cập nhật] [X] 52

Và các thành phần liên thông

* Tự học - Áp dụng thuật toán Tarjan tìm các BPLT của đồ thị có hướng (ngẫu nhiên)

Đang thi: 1
Đang thi: 1/30
Thời gian: 1:00
Tín hiệu: 100
Tỷ lệ: 100%

Thiết kế Tarjan tìm các thành phần liên thông mạnh (Tarjan's strongly connected Components algorithm) áp dụng thuật toán duyệt DFS theo chiều sâu (tổng độ rộng) kết hợp với danh sách các đỉnh và một ngăn xếp lưu trữ các thành phần liên thông mạnh (strongly connected components) gọi tắt là SCC.

```

void SSSCC(v) {
    // phán số cho đỉnh u, đưa u vào ngăn xếp
    min_num[v] = max_num[v] = k;
    push(v, w);
    mark[v] = true;

    // tách lynh nút các đỉnh kề của u
    for (e là các đỉnh kề của u) {
        if (!mark[e]) {
            // tra: hiện x = cắp nhận ta: min_num[e]
            min_num[e] = max_num[e] = min_num[v];
            SSSCC(e);
            l += 1; // v là đỉnh stack
        } else if (e == w) {
            // cắp nhận lại min_num[e]
            min_num[w] = max_num[w] = min_num[v];
        }
    }
    // tách kề của u, không thêm vào ngăn xếp
}

```

Thiết kế trên gồm 3 bước chính:

1. Dành số cho đỉnh, đưa u vào ngăn xếp. Sau khi dành số xong tăng k lên. Thông thường, k được khởi tạo bằng 1.
2. Kết các đỉnh kề của u, có 2 trường hợp xảy ra:
 - v > x: hiện x = cắp nhận ta: min_num[u]
 - v <= x: duyệt rõ những nút còn trên stack => cắp nhận lại min_num[u]
3. Kết thúc duyệt, không có đỉnh nào kề u là đỉnh khớp (articulation vertex hay điểm cõi chết).

Lưu ý là các đỉnh trong Stack sẽ cho đến khi u là đỉnh lấp ra. Các đỉnh được lấy ra này chính là thành phần liên thông mạnh của u.

Đoạn thi có hướng ghi: 7 đỉnh và 7 cung của bốn bến bến đỗ. Hãy áp dụng thuật toán Tarjan để tìm các thành phần liên thông mạnh của 4 bến đỗ đầu từ 2. Dựa vào kết quả thuật toán, các thành phần liên thông tìm được. Với mỗi thành phần, họ có tên là các đỉnh và các cung bùn trong thành phần liên thông này. Ngoài khác, xem xét độ dài của mỗi bến đỗ và số cửa sổ nằm ở bến phà liên thông khac.

Quy tắc:

- Mỗi bến đỗ của hanh động phải mìn và không chung chí nhau. Ví dụ: gđ 4 bến đỗ không bao giờ trên cùng chí nhau hơn nữa.
- Lưu ý bài thử thử bộ input vẫn trong tự bờ biển xuống dưới.
- Nếu làm sai 1 bước, có thể bị mất "Lỗi tại 1 bước". Để làm lại bước đó.
- Lết kèc định theo thứ tự 1, 2, 3, ...
- k được khai báo = 1.

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

Reset answer

Nhập giải

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

Đồ thị giao

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

Độ dài là 1 bước | Số bước: 45

SCC(1) [---] [---]

1. Dành số cho đỉnh 1 và đưa nó vào ngăn xếp.
Gán num[1] = min_num[1] = k = 1
Đưa 1 vào 5. Kết quả: S = 2

Thực hiện đánh số và đưa vào Stack [X] 1

2. Với các đỉnh kề của 1: 1,4,7

Độ dài là 1 bước | Số bước: 3

Độ dài là 1 bước | Số bước: 4

SCC(2) [---] [---]

1. Dành số cho đỉnh 2 và đưa nó vào ngăn xếp.
Gán num[2] = min_num[2] = k = 2
Đưa 2 vào 5. Kết quả: S = 2,3

Thực hiện đánh số và đưa vào Stack [X] 2

3. Kiểm tra num và min_num của 2

num[2] == min_num[2]
num[2] != min_num[2] X 3

Cập nhật min_num[2] = min(min_num[2], min_num[7]) = 2
Cập nhật min_num[7] = min(min_num[7], min_num[2]) = 2

Độ dài là 1 bước | Số bước: 3

SCC(3) [---] [---]

1. Dành số cho đỉnh 3 và đưa nó vào ngăn xếp.
Gán num[3] = min_num[3] = k = 3
Đưa 3 vào 5. Kết quả: S = 2,3,5

Thực hiện đánh số và đưa vào Stack [X] 3

4. Kiểm tra num và min_num của 3

num[3] == min_num[3]
num[3] != min_num[3] X 3

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

Độ dài là 1 bước | Số bước: 3

SCC(4) [---] [---]

1. Dành số cho đỉnh 4 và đưa nó vào ngăn xếp.
Gán num[4] = min_num[4] = k = 4
Đưa 4 vào 5. Kết quả: S = 2,3,5,7,6

Thực hiện đánh số và đưa vào Stack [X] 4

5. Kiểm tra num và min_num của 4

num[4] == min_num[4]
num[4] != min_num[4] X 5

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

Độ dài là 1 bước | Số bước: 4

SCC(5) [---] [---]

1. Dành số cho đỉnh 5 và đưa nó vào ngăn xếp.
Gán num[5] = min_num[5] = k = 5
Đưa 5 vào 6. Kết quả: S = 2,3,5,6,7

Thực hiện đánh số và đưa vào Stack [X] 5

6. Kiểm tra num và min_num của 5

num[5] == min_num[5]
num[5] != min_num[5] X 6

Cập nhật min_num[5] = min(min_num[5], min_num[7]) = 5
Cập nhật min_num[7] = min(min_num[7], min_num[5]) = 5

Độ dài là 1 bước | Số bước: 5

SCC(6) [---] [---]

1. Dành số cho đỉnh 6 và đưa nó vào ngăn xếp.
Gán num[6] = min_num[6] = k = 6
Đưa 6 vào 7. Kết quả: S = 2,3,5,7,6,4

Thực hiện đánh số và đưa vào Stack [X] 6

7. Kiểm tra num và min_num của 6

num[6] == min_num[6]
num[6] != min_num[6] X 7

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

Độ dài là 1 bước | Số bước: 6

SCC(7) [---] [---]

1. Dành số cho đỉnh 7 và đưa nó vào ngăn xếp.
Gán num[7] = min_num[7] = k = 7
Đưa 7 vào 5. Kết quả: S = 2,3,5,7,6,4,2

Thực hiện đánh số và đưa vào Stack [X] 7

8. Kiểm tra num và min_num của 7

num[7] == min_num[7]
num[7] != min_num[7] X 8

Cập nhật min_num[7] = min(min_num[7], min_num[5]) = 7
Cập nhật min_num[5] = min(min_num[5], min_num[7]) = 7

Độ dài là 1 bước | Số bước: 7

SCC(8) [---] [---]

1. Dành số cho đỉnh 8 và đưa nó vào ngăn xếp.
Gán num[8] = min_num[8] = k = 8
Đưa 8 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8

Thực hiện đánh số và đưa vào Stack [X] 8

9. Kiểm tra num và min_num của 8

num[8] == min_num[8]
num[8] != min_num[8] X 9

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

Độ dài là 1 bước | Số bước: 8

SCC(9) [---] [---]

1. Dành số cho đỉnh 9 và đưa nó vào ngăn xếp.
Gán num[9] = min_num[9] = k = 9
Đưa 9 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9

Thực hiện đánh số và đưa vào Stack [X] 9

10. Kiểm tra num và min_num của 9

num[9] == min_num[9]
num[9] != min_num[9] X 10

Cập nhật min_num[9] = min(min_num[9], min_num[7]) = 9
Cập nhật min_num[7] = min(min_num[7], min_num[9]) = 9

Độ dài là 1 bước | Số bước: 9

SCC(10) [---] [---]

1. Dành số cho đỉnh 10 và đưa nó vào ngăn xếp.
Gán num[10] = min_num[10] = k = 10
Đưa 10 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,10

Thực hiện đánh số và đưa vào Stack [X] 10

11. Kiểm tra num và min_num của 10

num[10] == min_num[10]
num[10] != min_num[10] X 11

Cập nhật min_num[10] = min(min_num[10], min_num[7]) = 10
Cập nhật min_num[7] = min(min_num[7], min_num[10]) = 10

Độ dài là 1 bước | Số bước: 10

SCC(11) [---] [---]

1. Dành số cho đỉnh 11 và đưa nó vào ngăn xếp.
Gán num[11] = min_num[11] = k = 11
Đưa 11 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11

Thực hiện đánh số và đưa vào Stack [X] 11

12. Kiểm tra num và min_num của 11

num[11] == min_num[11]
num[11] != min_num[11] X 12

Cập nhật min_num[11] = min(min_num[11], min_num[7]) = 11
Cập nhật min_num[7] = min(min_num[7], min_num[11]) = 11

Độ dài là 1 bước | Số bước: 11

SCC(12) [---] [---]

1. Dành số cho đỉnh 12 và đưa nó vào ngăn xếp.
Gán num[12] = min_num[12] = k = 12
Đưa 12 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12

Thực hiện đánh số và đưa vào Stack [X] 12

13. Kiểm tra num và min_num của 12

num[12] == min_num[12]
num[12] != min_num[12] X 13

Cập nhật min_num[12] = min(min_num[12], min_num[7]) = 12
Cập nhật min_num[7] = min(min_num[7], min_num[12]) = 12

Độ dài là 1 bước | Số bước: 12

SCC(13) [---] [---]

1. Dành số cho đỉnh 13 và đưa nó vào ngăn xếp.
Gán num[13] = min_num[13] = k = 13
Đưa 13 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13

Thực hiện đánh số và đưa vào Stack [X] 13

14. Kiểm tra num và min_num của 13

num[13] == min_num[13]
num[13] != min_num[13] X 14

Cập nhật min_num[13] = min(min_num[13], min_num[7]) = 13
Cập nhật min_num[7] = min(min_num[7], min_num[13]) = 13

Độ dài là 1 bước | Số bước: 13

SCC(14) [---] [---]

1. Dành số cho đỉnh 14 và đưa nó vào ngăn xếp.
Gán num[14] = min_num[14] = k = 14
Đưa 14 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14

Thực hiện đánh số và đưa vào Stack [X] 14

15. Kiểm tra num và min_num của 14

num[14] == min_num[14]
num[14] != min_num[14] X 15

Cập nhật min_num[14] = min(min_num[14], min_num[7]) = 14
Cập nhật min_num[7] = min(min_num[7], min_num[14]) = 14

Độ dài là 1 bước | Số bước: 14

SCC(15) [---] [---]

1. Dành số cho đỉnh 15 và đưa nó vào ngăn xếp.
Gán num[15] = min_num[15] = k = 15
Đưa 15 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15

Thực hiện đánh số và đưa vào Stack [X] 15

16. Kiểm tra num và min_num của 15

num[15] == min_num[15]
num[15] != min_num[15] X 16

Cập nhật min_num[15] = min(min_num[15], min_num[7]) = 15
Cập nhật min_num[7] = min(min_num[7], min_num[15]) = 15

Độ dài là 1 bước | Số bước: 15

SCC(16) [---] [---]

1. Dành số cho đỉnh 16 và đưa nó vào ngăn xếp.
Gán num[16] = min_num[16] = k = 16
Đưa 16 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16

Thực hiện đánh số và đưa vào Stack [X] 16

17. Kiểm tra num và min_num của 16

num[16] == min_num[16]
num[16] != min_num[16] X 17

Cập nhật min_num[16] = min(min_num[16], min_num[7]) = 16
Cập nhật min_num[7] = min(min_num[7], min_num[16]) = 16

Độ dài là 1 bước | Số bước: 16

SCC(17) [---] [---]

1. Dành số cho đỉnh 17 và đưa nó vào ngăn xếp.
Gán num[17] = min_num[17] = k = 17
Đưa 17 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17

Thực hiện đánh số và đưa vào Stack [X] 17

18. Kiểm tra num và min_num của 17

num[17] == min_num[17]
num[17] != min_num[17] X 18

Cập nhật min_num[17] = min(min_num[17], min_num[7]) = 17
Cập nhật min_num[7] = min(min_num[7], min_num[17]) = 17

Độ dài là 1 bước | Số bước: 17

SCC(18) [---] [---]

1. Dành số cho đỉnh 18 và đưa nó vào ngăn xếp.
Gán num[18] = min_num[18] = k = 18
Đưa 18 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18

Thực hiện đánh số và đưa vào Stack [X] 18

19. Kiểm tra num và min_num của 18

num[18] == min_num[18]
num[18] != min_num[18] X 19

Cập nhật min_num[18] = min(min_num[18], min_num[7]) = 18
Cập nhật min_num[7] = min(min_num[7], min_num[18]) = 18

Độ dài là 1 bước | Số bước: 18

SCC(19) [---] [---]

1. Dành số cho đỉnh 19 và đưa nó vào ngăn xếp.
Gán num[19] = min_num[19] = k = 19
Đưa 19 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19

Thực hiện đánh số và đưa vào Stack [X] 19

20. Kiểm tra num và min_num của 19

num[19] == min_num[19]
num[19] != min_num[19] X 20

Cập nhật min_num[19] = min(min_num[19], min_num[7]) = 19
Cập nhật min_num[7] = min(min_num[7], min_num[19]) = 19

Độ dài là 1 bước | Số bước: 19

SCC(20) [---] [---]

1. Dành số cho đỉnh 20 và đưa nó vào ngăn xếp.
Gán num[20] = min_num[20] = k = 20
Đưa 20 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20

Thực hiện đánh số và đưa vào Stack [X] 20

21. Kiểm tra num và min_num của 20

num[20] == min_num[20]
num[20] != min_num[20] X 21

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

Độ dài là 1 bước | Số bước: 20

SCC(21) [---] [---]

1. Dành số cho đỉnh 21 và đưa nó vào ngăn xếp.
Gán num[21] = min_num[21] = k = 21
Đưa 21 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21

Thực hiện đánh số và đưa vào Stack [X] 21

22. Kiểm tra num và min_num của 21

num[21] == min_num[21]
num[21] != min_num[21] X 22

Cập nhật min_num[21] = min(min_num[21], min_num[7]) = 21
Cập nhật min_num[7] = min(min_num[7], min_num[21]) = 21

Độ dài là 1 bước | Số bước: 21

SCC(22) [---] [---]

1. Dành số cho đỉnh 22 và đưa nó vào ngăn xếp.
Gán num[22] = min_num[22] = k = 22
Đưa 22 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22

Thực hiện đánh số và đưa vào Stack [X] 22

23. Kiểm tra num và min_num của 22

num[22] == min_num[22]
num[22] != min_num[22] X 23

Cập nhật min_num[22] = min(min_num[22], min_num[7]) = 22
Cập nhật min_num[7] = min(min_num[7], min_num[22]) = 22

Độ dài là 1 bước | Số bước: 22

SCC(24) [---] [---]

1. Dành số cho đỉnh 24 và đưa nó vào ngăn xếp.
Gán num[24] = min_num[24] = k = 24
Đưa 24 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24

Thực hiện đánh số và đưa vào Stack [X] 24

25. Kiểm tra num và min_num của 24

num[24] == min_num[24]
num[24] != min_num[24] X 25

Cập nhật min_num[24] = min(min_num[24], min_num[7]) = 24
Cập nhật min_num[7] = min(min_num[7], min_num[24]) = 24

Độ dài là 1 bước | Số bước: 24

SCC(26) [---] [---]

1. Dành số cho đỉnh 26 và đưa nó vào ngăn xếp.
Gán num[26] = min_num[26] = k = 26
Đưa 26 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26

Thực hiện đánh số và đưa vào Stack [X] 26

27. Kiểm tra num và min_num của 26

num[26] == min_num[26]
num[26] != min_num[26] X 27

Cập nhật min_num[26] = min(min_num[26], min_num[7]) = 26
Cập nhật min_num[7] = min(min_num[7], min_num[26]) = 26

Độ dài là 1 bước | Số bước: 26

SCC(28) [---] [---]

1. Dành số cho đỉnh 28 và đưa nó vào ngăn xếp.
Gán num[28] = min_num[28] = k = 28
Đưa 28 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28

Thực hiện đánh số và đưa vào Stack [X] 28

29. Kiểm tra num và min_num của 28

num[28] == min_num[28]
num[28] != min_num[28] X 29

Cập nhật min_num[28] = min(min_num[28], min_num[7]) = 28
Cập nhật min_num[7] = min(min_num[7], min_num[28]) = 28

Độ dài là 1 bước | Số bước: 28

SCC(30) [---] [---]

1. Dành số cho đỉnh 30 và đưa nó vào ngăn xếp.
Gán num[30] = min_num[30] = k = 30
Đưa 30 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30

Thực hiện đánh số và đưa vào Stack [X] 30

31. Kiểm tra num và min_num của 30

num[30] == min_num[30]
num[30] != min_num[30] X 31

Cập nhật min_num[30] = min(min_num[30], min_num[7]) = 30
Cập nhật min_num[7] = min(min_num[7], min_num[30]) = 30

Độ dài là 1 bước | Số bước: 30

SCC(32) [---] [---]

1. Dành số cho đỉnh 32 và đưa nó vào ngăn xếp.
Gán num[32] = min_num[32] = k = 32
Đưa 32 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32

Thực hiện đánh số và đưa vào Stack [X] 32

33. Kiểm tra num và min_num của 32

num[32] == min_num[32]
num[32] != min_num[32] X 33

Cập nhật min_num[32] = min(min_num[32], min_num[7]) = 32
Cập nhật min_num[7] = min(min_num[7], min_num[32]) = 32

Độ dài là 1 bước | Số bước: 32

SCC(34) [---] [---]

1. Dành số cho đỉnh 34 và đưa nó vào ngăn xếp.
Gán num[34] = min_num[34] = k = 34
Đưa 34 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34

Thực hiện đánh số và đưa vào Stack [X] 34

35. Kiểm tra num và min_num của 34

num[35] == min_num[35]
num[35] != min_num[35] X 35

Cập nhật min_num[35] = min(min_num[35], min_num[7]) = 35
Cập nhật min_num[7] = min(min_num[7], min_num[35]) = 35

Độ dài là 1 bước | Số bước: 35

SCC(36) [---] [---]

1. Dành số cho đỉnh 36 và đưa nó vào ngăn xếp.
Gán num[36] = min_num[36] = k = 36
Đưa 36 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36

Thực hiện đánh số và đưa vào Stack [X] 36

37. Kiểm tra num và min_num của 36

num[37] == min_num[37]
num[37] != min_num[37] X 37

Cập nhật min_num[37] = min(min_num[37], min_num[7]) = 37
Cập nhật min_num[7] = min(min_num[7], min_num[37]) = 37

Độ dài là 1 bước | Số bước: 37

SCC(38) [---] [---]

1. Dành số cho đỉnh 38 và đưa nó vào ngăn xếp.
Gán num[38] = min_num[38] = k = 38
Đưa 38 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36,38

Thực hiện đánh số và đưa vào Stack [X] 38

39. Kiểm tra num và min_num của 38

num[39] == min_num[39]
num[39] != min_num[39] X 39

Cập nhật min_num[39] = min(min_num[39], min_num[7]) = 39
Cập nhật min_num[7] = min(min_num[7], min_num[39]) = 39

Độ dài là 1 bước | Số bước: 39

SCC(40) [---] [---]

1. Dành số cho đỉnh 40 và đưa nó vào ngăn xếp.
Gán num[40] = min_num[40] = k = 40
Đưa 40 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36,38,40

Thực hiện đánh số và đưa vào Stack [X] 40

41. Kiểm tra num và min_num của 41

num[41] == min_num[41]
num[41] != min_num[41] X 41

Cập nhật min_num[41] = min(min_num[41], min_num[7]) = 41
Cập nhật min_num[7] = min(min_num[7], min_num[41]) = 41

Độ dài là 1 bước | Số bước: 41

SCC(42) [---] [---]

1. Dành số cho đỉnh 42 và đưa nó vào ngăn xếp.
Gán num[42] = min_num[42] = k = 42
Đưa 42 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36,38,40,42

Thực hiện đánh số và đưa vào Stack [X] 42

43. Kiểm tra num và min_num của 43

num[43] == min_num[43]
num[43] != min_num[43] X 43

Cập nhật min_num[43] = min(min_num[43], min_num[7]) = 43
Cập nhật min_num[7] = min(min_num[7], min_num[43]) = 43

Độ dài là 1 bước | Số bước: 43

SCC(44) [---] [---]

1. Dành số cho đỉnh 44 và đưa nó vào ngăn xếp.
Gán num[44] = min_num[44] = k = 44
Đưa 44 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36,38,40,42,44

Thực hiện đánh số và đưa vào Stack [X] 44

45. Kiểm tra num và min_num của 45

num[45] == min_num[45]
num[45] != min_num[45] X 45

Cập nhật min_num[45] = min(min_num[45], min_num[7]) = 45
Cập nhật min_num[7] = min(min_num[7], min_num[45]) = 45

Độ dài là 1 bước | Số bước: 45

SCC(46) [---] [---]

1. Dành số cho đỉnh 46 và đưa nó vào ngăn xếp.
Gán num[46] = min_num[46] = k = 46
Đưa 46 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36,38,40,42,44,46

Thực hiện đánh số và đưa vào Stack [X] 46

47. Kiểm tra num và min_num của 47

num[47] == min_num[47]
num[47] != min_num[47] X 47

Cập nhật min_num[47] = min(min_num[47], min_num[7]) = 47
Cập nhật min_num[7] = min(min_num[7], min_num[47]) = 47

Độ dài là 1 bước | Số bước: 47

SCC(48) [---] [---]

1. Dành số cho đỉnh 48 và đưa nó vào ngăn xếp.
Gán num[48] = min_num[48] = k = 48
Đưa 48 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36,38,40,42,44,46,48

Thực hiện đánh số và đưa vào Stack [X] 48

49. Kiểm tra num và min_num của 49

num[49] == min_num[49]
num[49] != min_num[49] X 49

Cập nhật min_num[49] = min(min_num[49], min_num[7]) = 49
Cập nhật min_num[7] = min(min_num[7], min_num[49]) = 49

Độ dài là 1 bước | Số bước: 49

SCC(50) [---] [---]

1. Dành số cho đỉnh 50 và đưa nó vào ngăn xếp.
Gán num[50] = min_num[50] = k = 50
Đưa 50 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50

Thực hiện đánh số và đưa vào Stack [X] 50

51. Kiểm tra num và min_num của 51

num[51] == min_num[51]
num[51] != min_num[51] X 51

Cập nhật min_num[51] = min(min_num[51], min_num[7]) = 51
Cập nhật min_num[7] = min(min_num[7], min_num[51]) = 51

Độ dài là 1 bước | Số bước: 51

SCC(52) [---] [---]

1. Dành số cho đỉnh 52 và đưa nó vào ngăn xếp.
Gán num[52] = min_num[52] = k = 52
Đưa 52 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52

Thực hiện đánh số và đưa vào Stack [X] 52

53. Kiểm tra num và min_num của 53

num[53] == min_num[53]
num[53] != min_num[53] X 53

Cập nhật min_num[53] = min(min_num[53], min_num[7]) = 53
Cập nhật min_num[7] = min(min_num[7], min_num[53]) = 53

Độ dài là 1 bước | Số bước: 53

SCC(54) [---] [---]

1. Dành số cho đỉnh 54 và đưa nó vào ngăn xếp.
Gán num[54] = min_num[54] = k = 54
Đưa 54 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54

Thực hiện đánh số và đưa vào Stack [X] 54

55. Kiểm tra num và min_num của 55

num[55] == min_num[55]
num[55] != min_num[55] X 55

Cập nhật min_num[55] = min(min_num[55], min_num[7]) = 55
Cập nhật min_num[7] = min(min_num[7], min_num[55]) = 55

Độ dài là 1 bước | Số bước: 55

SCC(56) [---] [---]

1. Dành số cho đỉnh 56 và đưa nó vào ngăn xếp.
Gán num[56] = min_num[56] = k = 56
Đưa 56 vào 5. Kết quả: S = 2,3,5,7,6,4,2,8,9,11,12,13,14,15,16,17,18,19,20,21,

* Tự học - for + DFS = Duyệt tất cả các đỉnh của đồ thị

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
Đặt cờ

Xét thuật toán duyệt đồ thị theo chiều sâu sử dụng đệ quy (**kiểm tra trong vòng lặp**) như bên dưới.

```
void DFS(int u) {
    //1. Đánh dấu u đã duyệt
    mark[u] = 1;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u)
        if (v chưa duyệt) { // mark[v] == 0
            //2a. Gọi đệ quy duyệt v
            DFS(v);
        } else { //mark[v] == 1
            //2b. Bỏ qua
        }
}
```

1. Duyệt (viết từ trái sang phải) và đánh dấu u đã duyệt.

2. Xét các đỉnh kề v của u, có 2 trường hợp xảy ra:

- v chưa duyệt => gọi đệ quy duyệt v.
- v đã duyệt => bỏ qua

Khởi tạo tất cả các đỉnh chưa duyệt.

Để duyệt toàn bộ đồ thị ta sử dụng vòng lặp sau:

```
for (u = 1; i <= n; u++)
    if (u chưa duyệt)
        DFS(u); //Gọi duyệt u
```

Cho đồ thị **vô hướng** gồm 5 đỉnh và 5 cung như bên như bên dưới. Hãy áp dụng thuật toán duyệt đồ thị bên trên để duyệt toàn bộ các đỉnh của G.

Quy ước

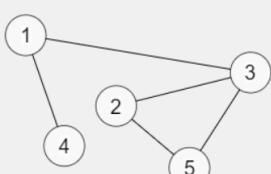
- Mỗi thể hiện của hàm **DFS(u)** được minh họa bằng một khung chữ nhật. Việc gọi đệ quy được minh họa bằng một hình chữ nhật nhỏ hơn bên trong.
- Hãy làm bài theo thứ tự từ ngoài vào trong và từ trên xuống dưới.
- Nếu làm sai 1 bước, có thể bấm "**Lùi lại 1 bước**" để làm lại bước trước đó.
- Liệt kê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

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

Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo Red Black



Thực hiện duyệt đệ quy theo chiều sâu toàn bộ đồ thị
Lùi lại 1 bước Số bước: 25

```
for (u = 1; i <= 5; u++) {

    u = 1, Duyệt Bỏ qua 1
    DFS(1)
    1. Đánh dấu 1 đã duyệt.
    Dánh dấu 2
    2. Với các đỉnh kề v của 1: 3,4 Xét 3
    2a. v = 3, chưa duyệt => Gọi đệ quy duyệt. Duyệt 4
    DFS(3)
    1. Đánh dấu 3 đã duyệt.
    Dánh dấu 5
    2. Với các đỉnh kề v của 3: 1,2,5 Xét 6
    2b. v = 1, đã duyệt rồi => Bỏ qua thôi. Bỏ qua 7
    2a. v = 2, chưa duyệt => Gọi đệ quy duyệt. Duyệt 8
    DFS(2)
    1. Đánh dấu 2 đã duyệt.
    Dánh dấu 9
    2. Với các đỉnh kề v của 2: 3,5 Xét 10
    2b. v = 3, đã duyệt rồi => Bỏ qua thôi. Bỏ qua 11
    2a. v = 5, chưa duyệt => Gọi đệ quy duyệt. Duyệt 12
    DFS(5)
    1. Đánh dấu 5 đã duyệt.
    Dánh dấu 13
    2. Với các đỉnh kề v của 5: 2,3 Xét 14
    2b. v = 2, đã duyệt rồi => Bỏ qua thôi. Bỏ qua 15
    2b. v = 3, đã duyệt rồi => Bỏ qua thôi. Bỏ qua 16
    2a. v = 4, chưa duyệt => Gọi đệ quy duyệt. Duyệt 18
    DFS(4)
    1. Đánh dấu 4 đã duyệt.
    Dánh dấu 19
    2. Với các đỉnh kề v của 4: 1 Xét 20
    2b. v = 1, đã duyệt rồi => Bỏ qua thôi. BỎ QUÁ 21
}

u = 2, Duyệt BỎ QUÁ 22
u = 3, Duyệt BỎ QUÁ 23
u = 4, Duyệt BỎ QUÁ 24
u = 5, Duyệt BỎ QUÁ 25
}

Vẽ cây duyệt đồ thị theo chiều sâu
```

Help Clear shift Delete Edit Undo Red Black

Câu hỏi 1
Đúng
Đạt điểm 0,90
trên 1,00
☞ Đặt cờ

Xét thuật toán duyệt đồ thị theo chiều sâu sử dụng đệ quy (**kiểm tra trong vòng lặp**) như bên dưới.

```
void DFS(int u) {
    //1. Đánh dấu u đã duyệt
    mark[u] = 1;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u)
        if (v chưa duyệt) { // mark[v] == 0
            //2a. Gọi đệ quy duyệt v
            DFS(v);
        } else { //mark[v] == 1
            //2b. Bỏ qua
        }
}
```

Thuật toán duyệt này gồm 2 bước chính:

1. Duyệt (ví dụ in ra màn hình) và đánh dấu u đã duyệt.
2. Xét các đỉnh kề v của u, có 2 trường hợp xảy ra:
 - o v chưa duyệt => gọi đệ quy duyệt v.
 - o v đã duyệt => bỏ qua

Khởi tạo tất cả các đỉnh chưa duyệt.

Để duyệt toàn bộ đồ thị ta sử dụng vòng lặp sau:

```
for (u = 1; i <= n; u++)
    if (u chưa duyệt)
        DFS(u); //Gọi duyệt u
```

Cho đồ thị **vô hướng** gồm 5 đỉnh và 4 cung như bên dưới. Hãy áp dụng thuật toán duyệt đồ thị bên trên để duyệt toàn bộ các đỉnh của G.

Quy ước

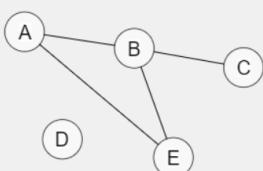
- Mỗi thể hiện của hàm **DFS(u)** được minh họa bằng một khung chữ nhật. Việc gọi đệ quy được minh họa bằng một hình chữ nhật nhỏ hơn bên trong.
- Hãy làm bài theo thứ tự từ ngoài vào trong và từ trên xuống dưới.
- Nếu làm sai 1 bước, có thể bấm "Lùi lại 1 bước" để làm lại bước trước đó.
- Liệt kê các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

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

Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo Red Black



Thực hiện duyệt đệ quy theo chiều sâu toàn bộ đồ thị

☞ Lùi lại 1 bước Số bước: 23

for (u = A; i <= E; u++) {

u = A, ✓ 1

DFS(A)

1. Đánh dấu A đã duyệt.

✓ 2

2. Với các đỉnh kề v của A: B,E ✓ 3

2a. v = B, chưa duyệt => Gọi đệ quy duyệt. ✓ 4

DFS(B)

1. Đánh dấu B đã duyệt.

✓ 5

2. Với các đỉnh kề v của B: A,C,E ✓ 6

2a. v = A, đã duyệt rồi => Bỏ qua thôi. X 7

DFS(C)

1. Đánh dấu C đã duyệt.

✓ 9

2. Với các đỉnh kề v của C: B ✓ 10

2a. v = B, đã duyệt rồi => Bỏ qua thôi. X 11

DFS(E)

1. Đánh dấu E đã duyệt.

✓ 13

2. Với các đỉnh kề v của E: A,B ✓ 14

2b. v = A, đã duyệt rồi => Bỏ qua thôi. X 15

2b. v = B, đã duyệt rồi => Bỏ qua thôi. X 16

2b. v = E, đã duyệt rồi => Bỏ qua thôi. X 17

u = B, X 18

u = C, X 19

u = D, ✓ 20

DFS(D)

1. Đánh dấu D đã duyệt.

✓ 21

2. Với các đỉnh kề v của D: ✓ 22

u = E, X 23

}

Vẽ cây duyệt đồ thị theo chiều sâu

Help Clear shift Delete Edit Undo Red Black

Tuần 4 - Tính liên thông và ứng dụng (Hướng dẫn thực hành)

* Tự học - Kiểm tra đồ thị CÓ HƯỚNG chứa chu trình

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
Đặt cờ

Xét thuật toán kiểm tra đồ thị (có hướng) chứa chu trình dựa trên thuật toán duyệt đồ thị theo chiều sâu (sử dụng đệ quy) kết hợp với tô màu các đỉnh như bên dưới.

```
#define WHITE 0
#define BLACK 1
#define GRAY 2
int has_cycle = 0;
void DFS(int u) {
    //1. Tô màu xám (GRAY) cho đỉnh u, u đang duyệt
    color[u] = GRAY;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u) {
        if (v chưa duyệt) { // v có màu WHITE
            //2a. Gọi đệ quy duyệt v
            DFS(v);
        } else if (v có màu GRAY) { //v còn đang duyệt,
            //2b. Có chu trình: v -> ... -> u -> v, dừng!
            has_cycle = 1;
            return;
        } else {
            //2c. bỏ qua, không làm gì cả
        }
    }

    //3. Duyệt xong: tô màu BLACK (DUYỆT XONG) cho u.
    color[u] = BLACK;
}
```

Thuật toán này dựa ý tưởng: giả sử, đỉnh chưa duyệt có màu WHITE (TRẮNG), đỉnh đang duyệt có màu GRAY (XÁM), đỉnh duyệt xong có màu BLACK (ĐEN). Trong quá trình duyệt, nếu ta gặp 1 đỉnh v có màu GRAY có nghĩa là ta đã đi 1 vòng từ v rồi trở về chính nó.

Thuật toán duyệt này gồm 3 bước chính:

1. Bắt đầu duyệt: tô màu GRAY (ĐANG DUYỆT) cho u.
2. Xét các đỉnh kề v của u, có 3 trường hợp xảy ra:
 - v có màu WHITE => gọi đệ quy duyệt v.
 - v có màu GRAY => có chu trình, dừng!
 - v có màu BLACK => duyệt xong rồi, bỏ qua
3. Duyệt xong: tô màu BLACK (DUYỆT XONG) cho u.

Khởi tạo tất cả các đỉnh đều có màu WHITE (CHÚA DUYỆT). Kết thúc thuật toán, nếu **has_cycle = 1** thì G chứa chu trình.

Cho đồ thị **có hướng** gồm 6 đỉnh và 10 cung như bên dưới. Hãy áp dụng thuật toán kiểm tra chu trình để kiểm tra đồ thị G có chứa chu trình không, bắt đầu từ đỉnh 5.

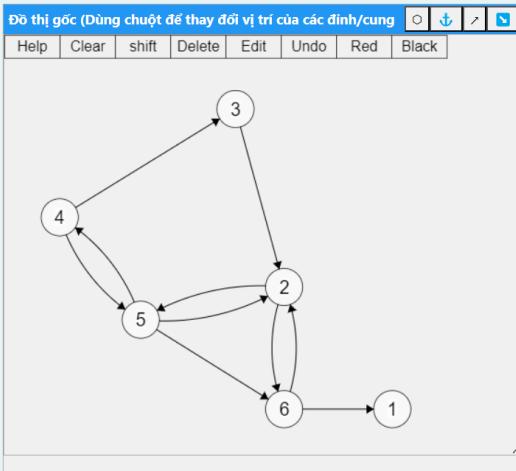
DFS(5);

Quy ước

- Mỗi thể hiện của hàm **DFS(u)** được minh họa bằng một khung chữ nhật. Việc gọi đệ quy được minh họa bằng một hình chữ nhật nhỏ nhô 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, ...**
- **Để đơn giản, khi phát hiện chứa chu trình hãy bỏ qua tất cả các bước còn lại.**

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

Reset answer



Áp dụng thuật toán kiểm tra chu trình bằng cách duyệt đệ quy theo chiều sâu kết hợp với tô màu bắt đầu từ đỉnh 5.

Lùi lại 1 bước | Số bước: 6

DFS(5)

1. Bắt đầu duyệt, tô màu ĐANG DUYỆT cho 5.

color[5] = GRAY (XÁM)

Tô màu 1

2. Với các đỉnh kề v của 5: 2,4,6

với v = '2' có màu WHITE (TRẮNG): chưa duyệt => Gọi đệ quy duyệt.

3

DFS(2)

1. Bắt đầu duyệt, tô màu ĐANG DUYỆT cho 2.

color[2] = GRAY (XÁM)

Tô màu 4

2. Với các đỉnh kề v của 2: 5,6

với v = '5', có màu GRAY (XÁM): đang duyệt => Phát hiện chu trình!!!

6

2a. v = '6' có màu WHITE (TRẮNG): chưa duyệt => Gọi đệ quy duyệt.

2b. v = '6', có màu GRAY (XÁM): đang duyệt => Phát hiện chu trình!!!

2c. v = '6', có màu BLACK (ĐEN): duyệt xong rồi => Bỏ qua thôi.

3. Duyệt xong, tô màu ĐÃ DUYỆT cho 2.

color[2] = WHITE (TRẮNG)

Tô màu

2a. v = '4' có màu WHITE (TRẮNG): chưa duyệt => Gọi đệ quy duyệt.

7

2b. v = '4', có màu GRAY (XÁM): đang duyệt => Phát hiện chu trình!!!

8

2c. v = '4', có màu BLACK (ĐEN): duyệt xong rồi => Bỏ qua thôi.

9

3. Duyệt xong, tô màu ĐÃ DUYỆT cho 5.

color[5] = WHITE (TRẮNG)

Tô màu

* Tự học - Kiểm tra đồ thị VÔ HƯỚNG chứa chu trình

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
Đặt câu

Xét thuật toán kiểm tra đồ thị (vô hướng) chứa chu trình dựa trên thuật toán duyệt đồ thị theo chiều sâu (sử dụng đệ quy) kết hợp với tô màu các đỉnh như bên dưới. Thuật toán này khác với thuật toán kiểm tra chu trình trên đồ thị có hướng một chút ở chỗ bổ sung thêm kiểm tra v có phải là cha của u không (tránh trường hợp v gọi duyệt và v gọi duyệt).

```
#define WHITE 0
#define BLACK 1
#define GRAY 2
int has_cycle = 0;
void DFS(int u, int parent) {
    //1. Tô màu xám (GRAY) cho đỉnh  $u$ ,  $u$  đang duyệt
    color[u] = GRAY;

    //2. Lần lượt xét các đỉnh kề của  $u$ 
    for (v là các đỉnh kề của  $u$ ) {
        if (v == parent) { //v vừa mới gọi duyệt  $u$ , giờ  $u$  gọi duyệt  $v$ 
            //2a. Bỏ qua
        } else if (v chưa duyệt) { //v có màu WHITE
            //2b. Gọi đệ quy duyệt  $v$ 
            DFS(v, u);
        } else if (v có màu GRAY) { //v còn đang duyệt,
            //2c. Có chu trình:  $v \rightarrow \dots \rightarrow u \rightarrow v$ , dừng!
            has_cycle = 1;
            return;
        } else {
            //2d. bỏ qua, không làm gì cả
        }
    }
    //3. Tô màu BLACK cho  $u$ 
    color[u] = BLACK;
}
```

Thuật toán này dựa ý tưởng: giả sử, đỉnh chưa duyệt có màu WHITE (TRẮNG), đỉnh đang duyệt có màu GRAY (XÂM), đỉnh duyệt xong có màu BLACK (ĐEN). Trong quá trình duyệt, nếu ta gặp 1 đỉnh v có màu GRAY có nghĩa là ta đã đi 1 vòng từ v rồi trở về chính nó.

Thuật toán duyệt này gồm 3 bước chính:

1. Bắt đầu duyệt: tô màu GRAY (ĐANG DUYỆT) cho u .
2. Xét các đỉnh kề v của u , có 3 trường hợp xảy ra (kiểm tra theo thứ tự):
 - $v == parent \Rightarrow v$ là cha của u , bỏ qua.
 - v có màu WHITE \Rightarrow gọi đệ quy duyệt v .
 - v có màu GRAY \Rightarrow có chu trình, dừng!
 - v có màu BLACK \Rightarrow duyệt xong rồi, bỏ qua
3. Duyệt xong: tô màu BLACK (DUYỆT XONG) cho u .

Khởi tạo tất cả các đỉnh đều có màu WHITE (CHƯA DUYỆT). Kết thúc thuật toán, nếu **has_cycle = 1** thì G chứa chu trình.

Cho đồ thị vô hướng gồm 6 đỉnh và 5 cung như bên như bên dưới. Hãy áp dụng thuật toán kiểm tra chu trình để kiểm tra đồ thị G có chứa chu trình không, bắt đầu từ đỉnh 4.

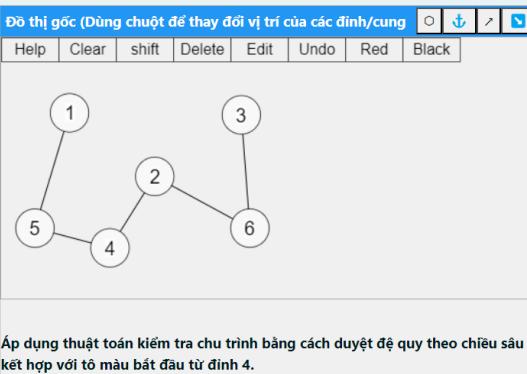
DFS(4, -1);

Quy ước

- Mỗi thê hiện của hàm **DFS(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 lùi 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, ...
- Để đơn giản, khi phát hiện chứa chu trình hãy bỏ qua tất cả các bước còn lại.**

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

Reset answer



DFS(4, #)

1. Bắt đầu duyệt, tô màu ĐANG DUYỆT cho 4.
color[4] = GRAY (XÂM)
Tô màu 1

2. Với các đỉnh kề v của 4: 2,5
Xét 2

DFS(2, 4)

1. Bắt đầu duyệt, tô màu ĐANG DUYỆT cho 2.
color[2] = GRAY (XÂM)
Tô màu 4

2. Với các đỉnh kề v của 2: 4,6
Xét 5

2a. $v = '4'$, là cha của $u \Rightarrow$ Bỏ qua. Bỏ qua X 6

2b. $v = '6'$ có màu WHITE (TRẮNG): chưa duyệt \Rightarrow Gọi đệ quy duyệt.
Duyệt 7

DFS(6, 2)

1. Bắt đầu duyệt, tô màu ĐANG DUYỆT cho 6.
color[6] = GRAY (XÂM)
Tô màu 8

2. Với các đỉnh kề v của 6: 2,3
Xét 9

2a. $v = '2'$, là cha của $u \Rightarrow$ Bỏ qua. Bỏ qua X 10

2b. $v = '3'$ có màu WHITE (TRẮNG): chưa duyệt \Rightarrow Gọi đệ quy duyệt.
Duyệt 11

DFS(3, 6)

1. Bắt đầu duyệt, tô màu ĐANG DUYỆT cho 3.
color[3] = GRAY (XÂM)
Tô màu 12

2. Với các đỉnh kề v của 3: 6
Xét 13

2a. $v = '6'$, là cha của $u \Rightarrow$ Bỏ qua. Bỏ qua X 14

3. Duyệt xong, tô màu ĐÃ DUYỆT cho 3.
color[3] = BLACK (ĐEN)
Tô màu 15

3. Duyệt xong, tô màu ĐÃ DUYỆT cho 6.
color[6] = BLACK (ĐEN)
Tô màu 16

3. Duyệt xong, tô màu ĐÃ DUYỆT cho 2.
color[2] = BLACK (ĐEN)
Tô màu 17

2b. $v = '5'$ có màu WHITE (TRẮNG): chưa duyệt \Rightarrow Gọi đệ quy duyệt.
Duyệt 18

DFS(5, 4)

1. Bắt đầu duyệt, tô màu ĐANG DUYỆT cho 5.
color[5] = GRAY (XÂM)
Tô màu 19

2. Với các đỉnh kề v của 5: 1,4
Xét 20

2a. $v = '1'$ có màu WHITE (TRẮNG): chưa duyệt \Rightarrow Gọi đệ quy duyệt.
Duyệt 21

DFS(1, 5)

1. Bắt đầu duyệt, tô màu ĐANG DUYỆT cho 1.
color[1] = GRAY (XÂM)
Tô màu 22

2. Với các đỉnh kề v của 1: 5
Xét 23

2a. $v = '5'$, là cha của $u \Rightarrow$ Bỏ qua. Bỏ qua X 24

3. Duyệt xong, tô màu ĐÃ DUYỆT cho 1.
color[1] = BLACK (ĐEN)
Tô màu 25

2a. $v = '4'$, là cha của $u \Rightarrow$ Bỏ qua. BỎ QUÁ X 26

3. Duyệt xong, tô màu ĐÃ DUYỆT cho 5.
color[5] = BLACK (ĐEN)
Tô màu 27

3. Duyệt xong, tô màu ĐÃ DUYỆT cho 4.
color[4] = BLACK (ĐEN)
Tô màu 28

* Tự học - Kiểm tra đồ thị PHÂN ĐÔI

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
▼ Đặt cờ

Xét thuật toán kiểm tra đồ thị phân đôi dựa trên thuật toán duyệt đồ thị theo chiều sâu (sử dụng đệ quy) kết hợp với tô màu các đỉnh như bên dưới.

```
#define NONE -1
#define BLUE 0
#define RED 1

int conflict = 0;
void Colorize(int u, int c) {
    //1. Tô màu c cho đỉnh u
    color[u] = c;

    //2. Lần lượt xét các đỉnh kề của u
    for (v là các đỉnh kề của u)
        if (v chưa duyệt) { // v chưa có màu, color[v] == NONE
            //2a. Duyệt v bằng màu ngược với màu c, !c
            Colorize(v, !c);
        } else if (màu của v giống màu u) { // color[u] == color[v]
            //2b. giống màu => không thể tô màu, dừng!
            conflict = 1;
            return;
        } else {
            //2c. bỏ qua, không làm gì cả
        }
}
}
```

Thuật toán này dựa trên tính chất "G là đồ thị phân đôi khi và chỉ khi ta có thể tô màu các đỉnh bằng hai màu sao cho hai đỉnh kề có màu khác nhau".

Thuật toán tô màu gồm 2 bước chính:

1. Tô màu c cho đỉnh u.
2. Xét các đỉnh kề v của u, có 3 trường hợp xảy ra:
 - v chưa duyệt (chưa có màu) => gọi đệ quy tô màu !c cho v.
 - v duyệt rồi và có màu giống với màu của u => dừng đệ, không thể tô, dừng!
 - v duyệt rồi và có màu khác với màu của u => bỏ qua

Kết thúc thuật toán, nếu conflict = 0 thì G là đồ thị phân đôi.

Cho đồ thị **vô hướng** gồm **6** đỉnh và **9** cung như bên dưới. Hãy áp dụng thuật toán tô màu các đỉnh để kiểm tra đồ thị G có phải là đồ thị phân đôi hay không, bắt đầu từ đỉnh **D** được tô màu **ĐEN**.

Colorize(D, ĐEN);

Nếu G là đồ thị phân đôi, hãy sắp xếp lại các đỉnh có màu ĐEN nằm bên trái, các đỉnh có màu còn lại ở bên phải. Các đỉnh có màu giống nhau được xếp ngang hàng nhau.

Quy ước

- Mỗi thê hiện của hàm **Colorize(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, ...**
- **Để đơn giản, khi dung độ xảy ra, hãy bỏ qua tất cả các bước còn lại.**

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

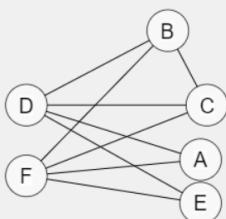
Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Nếu đồ thị này là đồ thị phân đôi, sau khi tô màu xong hãy sắp xếp lại các đỉnh của nó về hai phía. Các đỉnh có màu ĐEN ở bên trái, các đỉnh có màu TRẮNG ở bên phải.

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo Red Black



Áp dụng thuật toán kiểm tra đồ thị phân đôi bằng cách duyệt đệ quy theo chiều sâu kết hợp với tô màu bắt đầu từ đỉnh D được tô màu ĐEN.

Lùi lại 1 bước | Số bước: 18

Colorize(D, ĐEN) 1. Tô màu cho đỉnh D. color[D] = ĐEN <input type="checkbox"/> Tô màu ✓ 1	
2. Với các đỉnh kề v của D: A,B,C,E Xét ✓ 2 2a. v = A, chưa có màu <input type="checkbox"/> Tô màu nő ✓ 3	
Colorize(A, <color>) 1. Tô màu cho đỉnh A. color[A] = TRẮNG <input type="checkbox"/> Tô màu ✓ 4	
2. Với các đỉnh kề v của A: D,F Xét ✓ 5 2c. v = D, có màu rồi và khác màu của A <input type="checkbox"/> Bỏ qua X 6	
2a. v = F, chưa có màu <input type="checkbox"/> Tô màu nő ✓ 7	
Colorize(F, <color>) 1. Tô màu cho đỉnh F. color[F] = ĐEN <input type="checkbox"/> Tô màu ✓ 8	
2. Với các đỉnh kề v của F: A,B,C,E Xét ✓ 9 2c. v = A, có màu rồi và khác màu của F <input type="checkbox"/> Bỏ qua X 10	
2a. v = B, chưa có màu <input type="checkbox"/> Tô màu nő ✓ 11	
Colorize(B, <color>) 1. Tô màu cho đỉnh B. color[B] = TRẮNG <input type="checkbox"/> Tô màu ✓ 12	
2. Với các đỉnh kề v của B: C,D,F Xét ✓ 13 2a. v = C, chưa có màu <input type="checkbox"/> Tô màu nő ✓ 14	
Colorize(C, <color>) 1. Tô màu cho đỉnh C. color[C] = ĐEN <input type="checkbox"/> Tô màu ✓ 15	
2. Với các đỉnh kề v của C: B,D,F Xét ✓ 16 2c. v = B, có màu rồi và khác màu của C <input type="checkbox"/> Bỏ qua X 17	
2b. v = D, có màu rồi và giống màu của C => dung độ!!! <input type="checkbox"/> Dung ✓ 18	
2a. v = F, chưa có màu <input type="checkbox"/> Tô màu nő 2b. v = F, có màu rồi và giống màu của C => dung độ!!! <input type="checkbox"/> Dung 2c. v = F, có màu rồi và khác màu của C <input type="checkbox"/> Bỏ qua	
2a. v = D, chưa có màu <input type="checkbox"/> Tô màu nő 2b. v = D, có màu rồi và giống màu của B => dung độ!!! <input type="checkbox"/> Dung 2c. v = D, có màu rồi và khác màu của B <input type="checkbox"/> Bỏ qua	
2a. v = F, chưa có màu <input type="checkbox"/> Tô màu nő 2b. v = F, có màu rồi và giống màu của B => dung độ!!! <input type="checkbox"/> Dung 2c. v = F, có màu rồi và khác màu của B <input type="checkbox"/> BỎ QUÁ	
2a. v = C, chưa có màu <input type="checkbox"/> Tô màu nő 2b. v = C, có màu rồi và giống màu của F => dung độ!!! <input type="checkbox"/> Dung 2c. v = C, có màu rồi và khác màu của F <input type="checkbox"/> BỎ QUÁ	
2a. v = E, chưa có màu <input type="checkbox"/> Tô màu nő 2b. v = E, có màu rồi và giống màu của F => dung độ!!! <input type="checkbox"/> Dung 2c. v = E, có màu rồi và khác màu của F <input type="checkbox"/> BỎ QUÁ	
2a. v = B, chưa có màu <input type="checkbox"/> Tô màu nő 2b. v = B, có màu rồi và giống màu của D => dung độ!!! <input type="checkbox"/> Dung 2c. v = B, có màu rồi và khác màu của D <input type="checkbox"/> BỎ QUÁ	
2a. v = C, chưa có màu <input type="checkbox"/> Tô màu nő 2b. v = C, có màu rồi và giống màu của D => dung độ!!! <input type="checkbox"/> Dung 2c. v = C, có màu rồi và khác màu của D <input type="checkbox"/> BỎ QUÁ	
2a. v = E, chưa có màu <input type="checkbox"/> Tô màu nő 2b. v = E, có màu rồi và giống màu của D => dung độ!!! <input type="checkbox"/> Dung 2c. v = E, có màu rồi và khác màu của D <input type="checkbox"/> BỎ QUÁ	

Tuần 6 - Đường đi ngắn nhất trên đồ thị

* Tự học - Biểu diễn đồ thị có trọng số bằng phương pháp ma trận trọng số (VÔ HƯỚNG)

Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

Đặt cờ

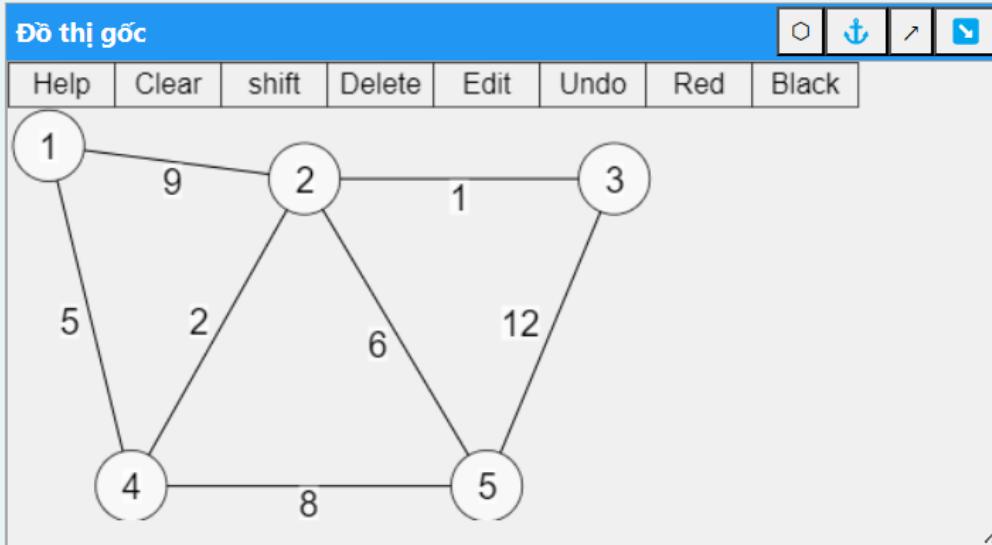
Cho đồ thị **vô hướng, có trọng số** gồm 5 đỉnh và 7 cung như hình vẽ.

Hãy biểu diễn đồ thị trên bằng phương pháp "**Ma trận trọng số**" (weighted matrix).

- $A[u][v] = \text{trọng số cung } (u, v)$, nếu có cung (u, v)
- $A[u][v] = \text{NO_EDGE}$, nếu không có cung (u, v)

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

Reset answer



Ma trận trọng số tương ứng của đồ thị trên

Nếu $A[u][v] = \text{NO_EDGE}$, có thể bỏ trống.

	1	2	3	4	5
1		9		5	
2	9		1	2	6
3		1			12
4	5	2			8
5		6	12	8	

Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

Đặt cờ

Cho đồ thị **có hướng, có trọng số** gồm 5 đỉnh và 6 cung như hình vẽ.

Hãy biểu diễn đồ thị trên bằng phương pháp "**Ma trận trọng số**" (weighted matrix).

- $A[u][v] = \text{trọng số cung } (u, v)$, nếu có cung (u, v)
- $A[u][v] = \text{NO_EDGE}$, nếu không có cung (u, v)

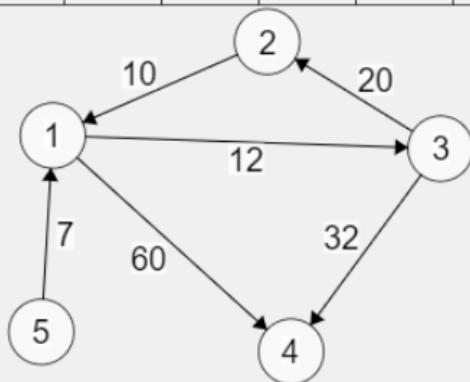
Answer: (penalty regime: 10, 20, ... %)

Reset answer

Đồ thị gốc



Help Clear shift Delete Edit Undo Red Black



Ma trận trọng số tương ứng của đồ thị trên

Nếu $A[u][v] = \text{NO_EDGE}$, có thể bỏ trống.

	1	2	3	4	5
1			12	60	
2	10				
3		20		32	
4					
5	7				

* Tự học - Biểu diễn đồ thị có trọng số bằng phương pháp danh sách cung

Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

Đặt cờ

Cho đồ thị **vô hướng, có trọng số** gồm 5 đỉnh và 7 cung như hình vẽ.

Hãy biểu diễn đồ thị trên bằng phương pháp "**Danh sách cung**" (edge list).

- Liệt kê các cung và trọng số của nó.

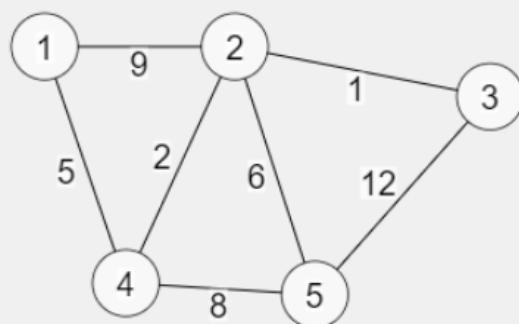
Answer: (penalty regime: 10, 20, ... %)

Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)



Help Clear shift Delete Edit Undo Red Black



Danh sách các cung (u, v, w), kèm theo trọng số, có trong đồ thị trên

1. (1 , 2 , 9)
2. (1 , 4 , 5)
3. (2 , 3 , 1)
4. (2 , 4 , 2)
5. (2 , 5 , 6)
6. (3 , 5 , 12)
7. (4 , 5 , 8)

* Tự học - Bài tập: Áp dụng thuật toán Moore - Dijkstra tìm đường đi ngắn nhất trên đồ thị (vd)

Câu hỏi 1
Đúng
Đạt điểm 0.80
trên 1.00
Đặt cờ

Cho đồ thị **có hướng** có trọng số không âm gồm **6** đỉnh và **10** cung như bên dưới.

Hãy áp dụng **thuật toán Moore - Dijkstra** để tìm các đường đi ngắn nhất từ đỉnh **1** đến các đỉnh khác trên đồ thị. Ở mỗi vòng lặp i ghi lại kết quả trung gian vào các ô tương ứng. Mỗi ô ở cột u ghi hai giá trị $\pi[u]$ và $p[u]$ cách nhau bằng dấu /, ví dụ: cột 3 được ghi là 4/6 thì nghĩa là $\pi[3] = 4$ và $p[3] = 6$.

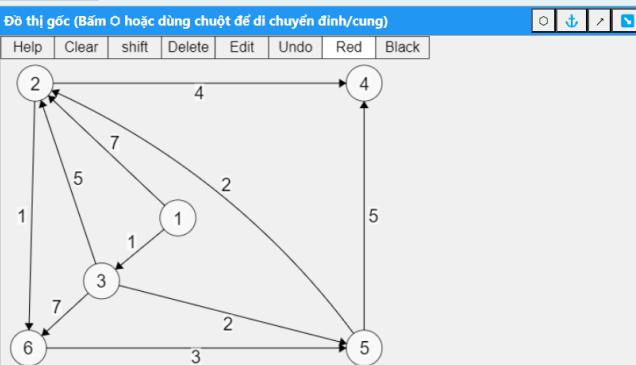
Dựa vào các $p[u]$ sau cùng, hãy vẽ cây đường đi ngắn nhất. Cây đường đi ngắn nhất gồm tất cả các đỉnh của đồ thị gốc và các cung ($p[u], u$).

Quy ước

- Sử dụng **oo** (hai ký tự o) để biểu diễn giá trị vô cùng.
- Nếu giá trị $p[u]$ chưa có, có thể bỏ trống, ghi -1 hoặc ghi -.
- Đỉnh không cập nhật nữa thì bỏ trống ở cột đó hoặc cũng có thể ghi lại giống hệt hàng bên trên.
- Đánh dấu đỉnh bằng dấu *
- Nếu có 2 đỉnh có cùng giá trị π thì chọn đỉnh có số thứ tự nhỏ.
- Nếu không chọn được u (không có đường đi từ s đến u), thì dừng.
- Cột công việc có thể không ghi.

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

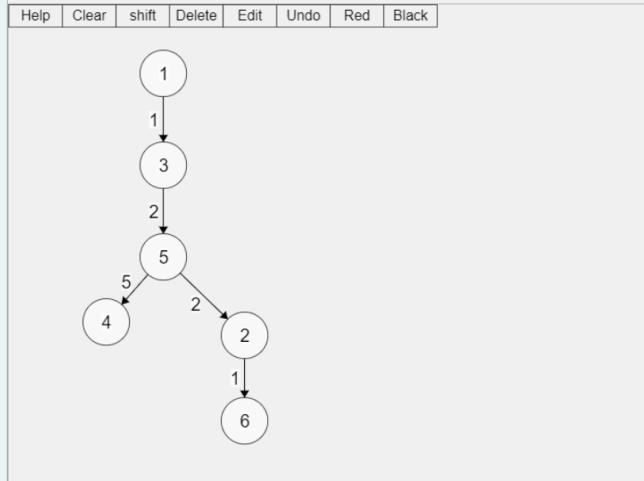
Reset answer



1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0	oo	oo	oo	oo	oo	
1	*	7/1	1/1				
2		6/3	*		3/3	8/3	
3		5/5		8/5	*		
4			*			6/2	
5						*	

2. Vẽ cây đường đi ngắn nhất



* Tự học - Bài tập: Áp dụng thuật toán Moore - Dijkstra tìm đường đi ngắn nhất trên đồ thị (random)

Câu hỏi 1
Đúng
Đạt điểm 1.00
Đặt cờ

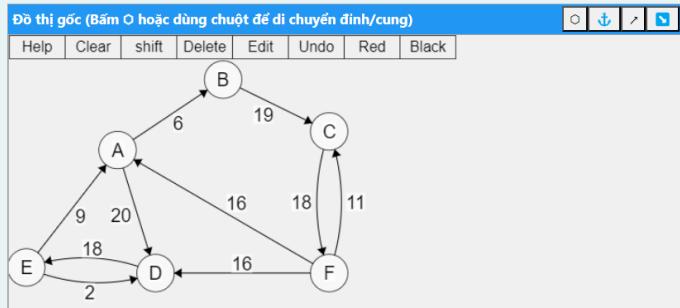
Cho đồ thị có hướng có trọng số không âm gồm 6 đỉnh và 10 cung như bên dưới.
Hãy áp dụng thuật toán Moore - Dijkstra để tìm các đường đi ngắn nhất từ đỉnh **B** đến các đỉnh khác trên đồ thị. Ở mỗi vòng lặp i hãy kết quả trung gian vào các ô tương ứng. Mỗi ô ở cột u ghi hai giá trị $\pi[u]$ và $p[u]$ cách nhau bằng dấu /, ví dụ: cột C được ghi là 4/F thì có nghĩa là $\pi[C] = 4$ và $p[C] = F$.
Dựa vào các $p[u]$ sau cùng, hãy vẽ cây đường đi ngắn nhất. Cây đường đi ngắn nhất gồm tất cả các đỉnh của đồ thị gốc và các cung ($p[u], u$).

Quy ước

- Sử dụng **oo** (hai ký tự o) để biểu diễn giá trị vô cùng.
- Nếu giá trị $p[u]$ chưa có, có thể bỏ trống, ghi -1 hoặc ghi -.
- Đỉnh không cập nhật nữa thì bỏ trống ở cột đó hoặc cũng có thể ghi lại giống hệt hàng bên trên.
- Đánh dấu đỉnh bằng dấu *
- Nếu có 2 đỉnh có cùng giá trị π thì chọn đỉnh có số thứ tự nhỏ.
- Nếu không chọn được u (không có đường đi từ s đến u), thì dừng.
- Cột công việc có thể không ghi.

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

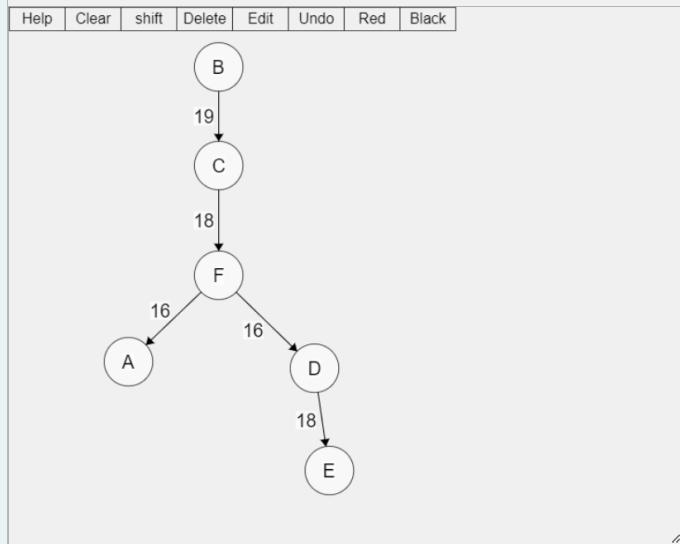
Reset answer



1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng

	A	B	C	D	E	F	Công việc
Khởi tạo	oo	0/-	oo	oo	oo	oo	
1		*	19/B				
2			*			37/C	
3	53/F			53/F		*	
4	*						
5				*	71/D		

2. Vẽ cây đường đi ngắn nhất



Câu hỏi 1

Đúng

Đạt điểm 1,00
trên 1,00

▼ Đặt cờ

Cho đồ thị **vô hướng** có trọng số không âm gồm **6** đỉnh và **9** cung như bên dưới.

Hãy áp dụng thuật toán Moore - Dijkstra để tìm các đường đi ngắn nhất từ đỉnh **4** đến các đỉnh khác trên đồ thị. Ở mỗi vòng lặp i ghi lại kết quả trung gian vào các ô tương ứng. Mỗi ô ở cột u ghi hai giá trị $\pi[u]$ và $p[u]$ cách nhau bằng dấu /, ví dụ: cột 3 được ghi là 4/6 thì có nghĩa là $\pi[3] = 4$ và $p[3] = 6$.

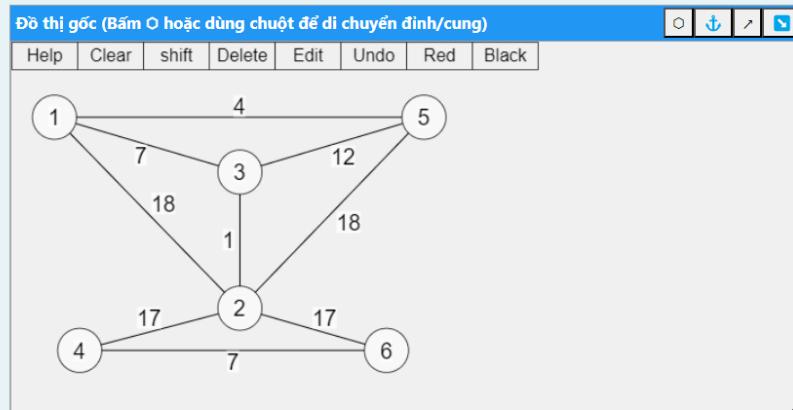
Dựa vào các $p[u]$ sau cùng, hãy vẽ cây đường đi ngắn nhất. Cây đường đi ngắn nhất gồm tất cả các đỉnh của đồ thị gốc và các cung $(p[u], u)$.

Quy ước

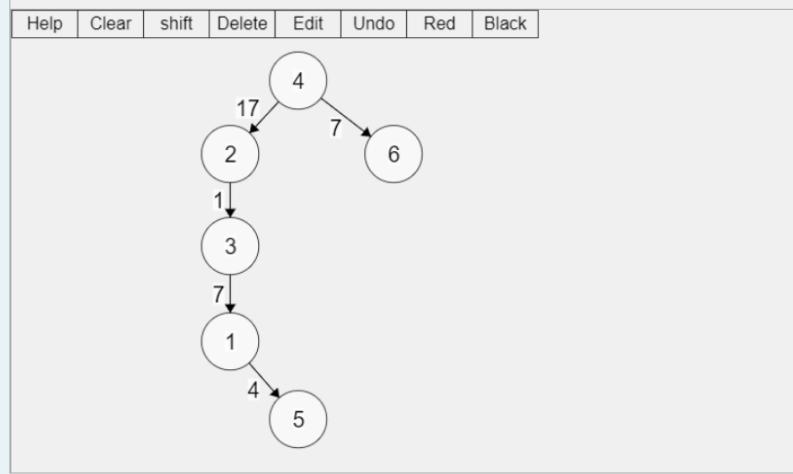
- Sử dụng **oo** (hai ký tự o) để biểu diễn giá trị vô cùng.
- Nếu giá trị $p[u]$ chưa có, có thể bỏ trống, ghi -1 hoặc ghi -.
- Đỉnh không cập nhật nữa thì bỏ trống ở cột đó hoặc cũng có thể ghi lại giống hệt hàng bên trên.
- Đánh dấu đỉnh bằng dấu *
- Nếu có 2 đỉnh có cùng giá trị π thì chọn đỉnh có số thứ tự nhỏ.
- Nếu không chọn được u (không có đường đi từ s đến u), thì dừng.
- Cột công việc có thể không ghi.

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

Reset answer

**1. Áp dụng thuật toán Moore - Dijkstra và ghi kết quả vào bảng**

	1	2	3	4	5	6	Công việc
Khởi tạo	oo	oo	oo	0/-	oo	oo	
1		17/4		*		7/4	
2						*	
3	35/2	*	18/2		35/2		
4	25/3		*		30/3		
5	*				29/1		

2. Vẽ cây đường đi ngắn nhất

Tuần 8 - Thứ tự topo & Ứng dụng

* Tự học - Thứ tự topo (ngẫu nhiên)

Câu hỏi 1

Đúng

Đạt điểm 0,90
trên 1,00

Đặt cờ

Cho đồ thị **có hướng không chu trình (DAG)** gồm 7 đỉnh và 11 cung bên dưới.

Sắp xếp các đỉnh của đồ thị theo thứ tự topo.

Nhắc lại: *thứ tự topo của các đỉnh là một cách sắp xếp các đỉnh sao cho với mỗi cung (u, v) , đỉnh đầu u phải đứng trước đỉnh cuối v .*

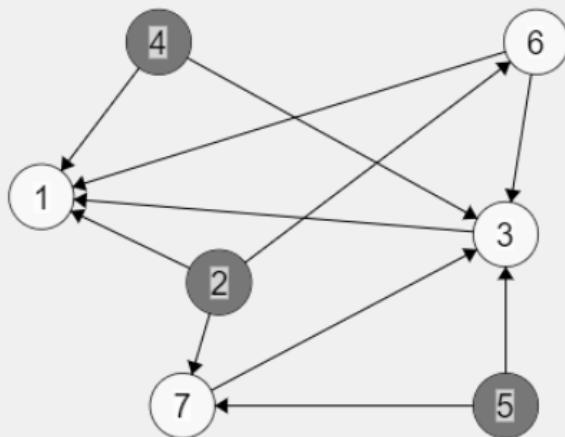
Answer: (penalty regime: 10, 20, ... %)

Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)



Help Clear shift Delete Edit Undo Red Black



Liệt kê các đỉnh theo thứ tự topo, ngăn cách các đỉnh bằng dấu phẩy

2,4,5,6,7,3,1

* Tự học - Xếp hạng đồ thị (ví dụ)

Câu hỏi 1
Đúng
Đạt điểm 0,80
trên 1,00
Đặt cờ

Cho đồ thị có hướng không chu trình (DAG) gồm 7 đỉnh và 12 cung bên dưới.

Hãy áp dụng [thuật toán xếp hạng đồ thị](#) để xếp hạng các đỉnh đồ thị này. Ở mỗi vòng lặp k ghi lại kết quả trung gian vào các ô tương ứng.

- Ở mỗi hàng k,
 - các cột u từ 1 đến 7 ghi **d[u]**: bậc vào của đỉnh u sau bước lặp k. Nếu ở bước k đỉnh u được xếp hạng thì **ghi dấu * vào ô (k, u)**
 - cột **S[k+1]** ghi các đỉnh gốc mới (các đỉnh có $d[u] = 0$) sau bước lặp k, cách nhau bằng dấu phẩy, ví dụ: **2, 5**
 - Cột **Công việc** ghi các công việc đã thực hiện trong bước k. Cột này không bắt buộc phải ghi
- Ở hàng **Kết quả**, ghi hạng của các đỉnh

Quy ước

- Đỉnh không cập nhật thì bỏ trống ở cột đó hoặc cũng có thể ghi lại giá trị giống hệt hàng bên trên

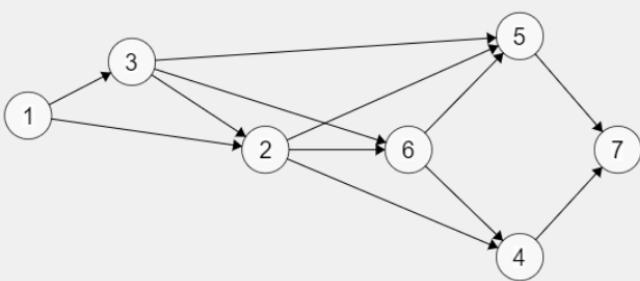
Answer: (penalty regime: 10, 20, ... %)

Reset answer

Sau khi xếp hạng xong, sắp xếp lại vị trí các đỉnh theo thứ tự của hạng từ trái sang phải

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

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



Áp dụng thuật toán xếp hạng và ghi kết quả vào bảng

	1	2	3	4	5	6	7	S[k+1]	Công việc
Khởi tạo	0	2	1	2	3	2	2	1	
0	*	1	0					3	
1		0	*		2	1		2	
2		*		1	1	0		6	
3				0	0	*		4,5	
4				*	*		0	7	
5							*		
Kết quả:	0	2	1	4	4	3	5		

* Tự học - Xếp hạng đồ thị (ngẫu nhiên)

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
Reset cờ

Cho đồ thị có hướng không chu trình (DAG) gồm 7 đỉnh và 14 cung bên dưới.

Hãy áp dụng thuật toán xếp hạng đồ thị để xếp hạng các đỉnh đồ thị này. Ở mỗi vòng lặp k ghi lại kết quả trung gian vào các ô tương ứng.

- Ở mỗi hàng k,
 - các cột u từ 1 đến 7 ghi $d[u]$: bậc vào của đỉnh u sau bước lặp k. Nếu ở bước k đỉnh u được xếp hạng thì **ghi dấu * vào ô (k, u)**
 - cột $S[k+1]$ ghi các đỉnh gốc mới (các đỉnh có $d[u] = 0$) sau bước lặp k, cách nhau bằng dấu phẩy, ví dụ: 2, 5
 - Cột **Công việc** ghi các công việc đã thực hiện trong bước k. Cột này không bắt buộc phải ghi
- Ở hàng **Kết quả**, ghi hạng của các đỉnh

Quy ước

- Đỉnh không cập nhật thì bỏ trống ở cột đó hoặc cũng có thể ghi lại giá trị giống hệt hàng bên trên

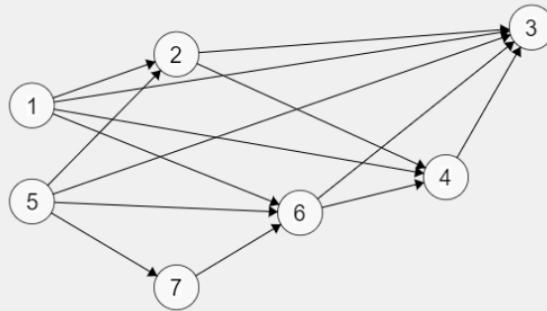
Answer: (penalty regime: 10, 20, ... %)

Reset answer

Sau khi xếp hạng xong, sắp xếp lại vị trí các đỉnh theo thứ tự của hạng từ trái sang phải

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

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



Áp dụng thuật toán xếp hạng và ghi kết quả vào bảng

	1	2	3	4	5	6	7	$S[k+1]$	Công việc
Khởi tạo	0	2	5	3	0	3	1	1,5	
0	*	0	3	2	*	1	0	2,7	
1		*	2	1		0	*	6	
2			1	0		*		4	
3			0	*				3	
4			*						
5									
6									
Kết quả:	0	1	4	3	0	2	1		

* Tự học - Quản lý dự án (ví dụ)

Câu hỏi 1
Đúng
Đạt điểm 0.91
trên 1.00
☞ Đạt cờ

Cho bảng công việc của một dự án như bên dưới.

Hãy vẽ đồ thị mô hình hóa bài toán quản lý dự án này. Vẽ lại đồ thị sau khi đã xếp hạng.

Tính thời gian sớm nhất có thể bắt đầu công việc u: $t[u]$

Tính thời gian trễ nhất có thể bắt đầu công việc u: $T[u]$

Quy ước

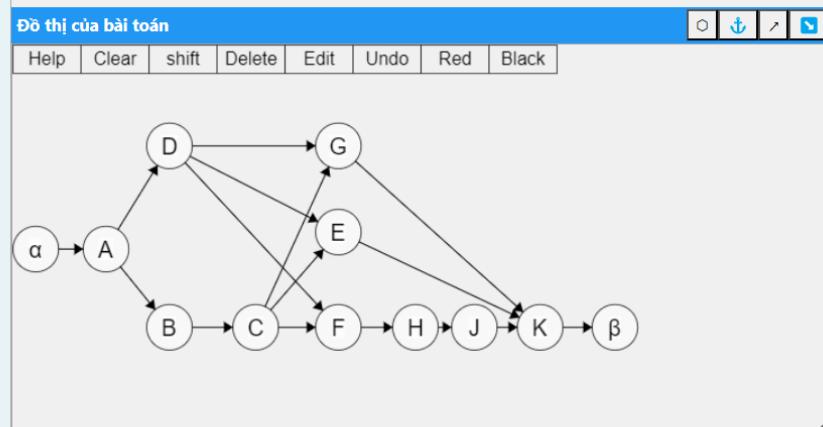
- Hai công việc giả là α và β . Để gõ được ký hiệu này, gõ $\backslash\alpha$ và $\backslash\beta$ vào định tương ứng.

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

Bảng công việc

Công việc	Mô tả	Thời gian hoàn thành (d[u])	Công việc trước đó
A	Các công việc hồ	7	
B	Dựng khung cho mái	3	A
C	Lớp mái	1	B
D	Lắp đặt hệ thống vệ sinh, chiếu sáng	8	A
E	Trang trí mặt tiền	2	C, D
F	Ráp cửa sổ	1	C, D
G	Trang hoàng vườn	1	C, D
H	Làm trần	2	F
J	Sơn phết	2	H
K	Chuyển nhà	1	E, G, J

1. Vẽ đồ thị mô hình hóa bài toán (sắp xếp các đỉnh sao cho đúng với hạng của chúng)



2. Tính $t[u]$ và $T[u]$

Đánh dấu * vào ô tương ứng với công việc then chốt

	α	A	B	C	D	E	F	G	H	J	K	β
$d[u]$	0	7	3	1	8	2	1	1	2	2	1	0
$t[u]$	0	0	7	10	7	15	15	15	16	18	20	21
$T[u]$	0	0	11	14	7	18	15	19	16	18	20	21
CV then chốt	*	*			*		*		*	*	*	

* Tự học - Quản lý dự án (ngẫu nhiên)

Câu hỏi 1

Đúng

Đạt điểm 0,97
trên 1,00

Đặt cờ

Cho bảng công việc của một dự án như bên dưới.

Hãy vẽ đồ thị mô hình hóa bài toán quản lý dự án này. Vẽ lại đồ thị sau khi đã xếp hạng.

Tính thời gian sớm nhất có thể bắt đầu công việc u: $t[u]$

Tính thời gian trễ nhất có thể bắt đầu công việc u: $T[u]$

Quy ước

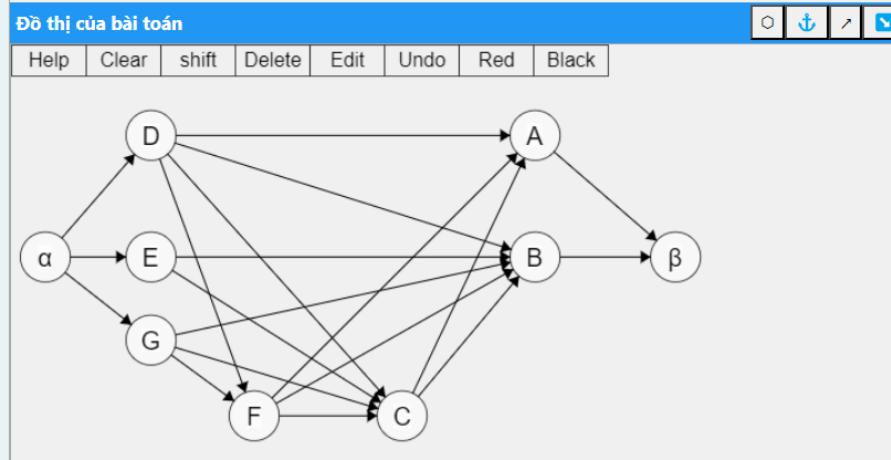
- Hai công việc giả là α và β . Để gõ được ký hiệu này, gõ \alpha và \beta vào định tương ứng.

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

Bảng công việc

Công việc	Mô tả	Thời gian hoàn thành (d[u])	Công việc trước đó
A	Công việc 1	18	F, D, C
B	Công việc 2	5	C, G, E, F, D
C	Công việc 3	11	D, G, F, E
D	Công việc 4	5	
E	Công việc 5	5	
F	Công việc 6	14	G, D
G	Công việc 7	7	

1. Vẽ đồ thị mô hình hóa bài toán (sắp xếp các đỉnh sao cho đúng với hạng của chúng)



2. Tính $t[u]$ và $T[u]$

Đánh dấu * vào ô tương ứng với công việc then chốt

	α	A	B	C	D	E	F	G	β
$d[u]$	0	18	5	11	5	5	14	7	0
$t[u]$	0	32	32	21	0	0	7	0	50
$T[u]$	0	32	45	21	2	16	7	0	50
CV then chốt	*	*	*	*			*	*	*

Tuần 11 - Cây

* Tự học - Áp dụng thuật toán Kruskal (cơ bản, ví dụ 1)

Câu hỏi 1
Đúng
Đạt điểm 1.00
trên 1.00
Đặt cờ

Cho đồ thị vô hướng có trọng số gồm **6** đỉnh và **10** cung như bên dưới.

Hãy áp dụng **thuật toán Kruskal** để tìm cây khung vô hướng nhỏ nhất (cây khung có tổng trọng số nhỏ nhất).

Ghi kết quả trung gian vào bảng.

Bước 1 (sắp xếp): Sắp xếp các cung theo trọng số tăng dần (đứng ra là không giảm, nhưng nói tăng dần cho dễ nhớ).

- Các cột **u, v, w** ghi các cung (u, v) và trọng số (w) của chúng theo thứ tự trọng số tăng dần.

Bước 2 (lặp): Lần lượt xét từng cung theo thứ tự đã sắp xếp ở bước 1, với mỗi cung xem xét thêm nó vào cây hay không. Mọi cung sẽ được thêm vào cây nếu như thêm nó vào không tạo thành chu trình.

- Cột **Thêm vào cây/ghи thêm** (hoặc **có** hoặc **x** hoặc **yes**) nếu cung này được thêm vào cây, ghi **không** (hoặc **không thêm** hoặc **no**) nếu không thêm cung này vào cây. Hãy vẽ hình trên giấy và dùng mắt kiểm tra xem việc thêm này có tạo chu trình hay không.

Bước 3 (Vẽ cây): Dựa vào các cung được chọn thêm vào cây trong bước 2, hãy vẽ cây khung nhỏ nhất trong phần **Cây khung nhỏ nhất**. Cây khung nhỏ nhất gồm tất cả các đỉnh của đồ thị gốc và các cung được thêm vào cây.

Quy ước

- Hai cung có trọng số giống nhau thì ghi cung nào trước cũng được.

Chú ý

- Cây kết quả phụ thuộc vào thứ tự sắp xếp của các cung.

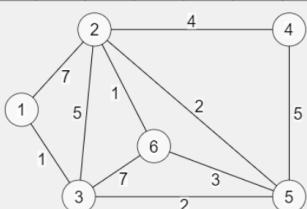
Answer: (penalty regime: 10, 20, ... %)

Reset answer

Đồ thị gốc (Dùng chuột để thay đổi vị trí của các đỉnh/cung)



Help Clear shift Delete Edit Undo Red Black

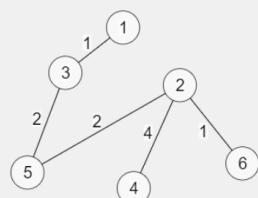


Áp dụng thuật toán Kruskal và ghi kết quả vào bảng

	u	v	w	Thêm vào cây?
1	1	3	1	x
2	2	6	1	x
3	2	5	2	x
4	3	5	2	x
5	5	6	3	no
6	2	4	4	x
7	2	3	5	no
8	4	5	5	no
9	1	2	7	no
10	3	6	7	no

Cây khung nhỏ nhất

Help Clear shift Delete Edit Undo Red Black



* Tự học - Áp dụng thuật toán Kruskal (cơ bản, ví dụ 2)

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
1% Đạt cơ

Cho đồ thị vô hướng có trọng số gồm 11 đỉnh và 21 cung như bên dưới.
Hãy áp dụng thuật toán Kruskal để tìm cây khung vô hướng nhỏ nhất (cây khung có tổng trọng số nhỏ nhất).
Ghi kết quả trung gian vào bảng.
Bước 1 (sắp xếp): Sắp xếp các cung theo trọng số tăng dần (đứng ra là không giảm, nhưng nón tăng dần cho dễ nhớ).

- Các cột u , v , w ghi các cung (u, v) và trọng số (w) của chúng theo thứ tự trọng số tăng dần.

Bước 2 (lập): Lần lượt xét từng cung theo thứ tự đã sắp xếp ở bước 1, với mỗi cung xem xét thêm nó vào cây hay không. Một cung sẽ được thêm vào cây nếu như thêm nó vào không tạo thành chu trình.

- Cột Thêm vào cây ghi thêm:** (hoặc **có** hoặc **x** hoặc **yes**) nếu cung này được thêm vào cây, ghi **không** (hoặc **không thêm** hoặc **no**) nếu không thêm cung này vào cây. Hãy vẽ hình trên giấy và dùng mắt kiểm tra xem việc thêm này có tạo chu trình hay không.

Bước 3 (Vẽ cây): Dựa vào các cung được chọn thêm vào cây trong bước 2, hãy vẽ cây khung nhỏ nhất trong phần **Cây khung nhỏ nhất**. Cây khung nhỏ nhất gồm tất cả các đỉnh của đồ thị gốc và các cung được thêm vào cây.

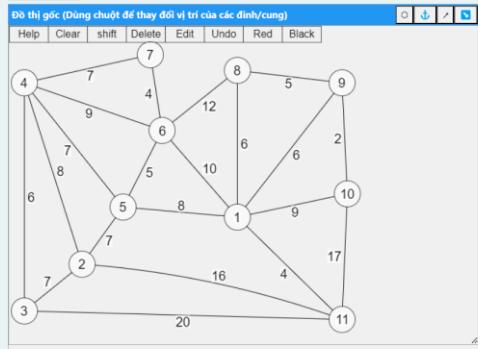
Quy ước:

- Hai cung có trọng số giống nhau thì chỉ xem nền trước cung đứng.

- Cây kết quả phụ thuộc vào thứ tự sắp xếp của các cung.

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

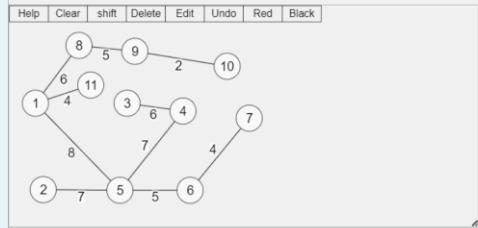
Reset answer



Áp dụng thuật toán Kruskal và ghi kết quả vào bảng

	u	v	w	Thêm vào cây?
1	9	10	2	x
2	1	11	4	x
3	6	7	4	x
4	5	6	5	x
5	8	9	5	x
6	1	8	6	x
7	1	9	6	no
8	3	4	6	x
9	2	5	7	x
10	4	5	7	x
11	4	7	7	no
12	2	3	7	no
13	2	4	8	no
14	1	5	8	x
15	4	6	9	no
16	1	10	9	no
17	1	6	10	no
18	6	8	12	no
19	2	11	16	no
20	10	11	17	no
21	3	11	20	no

Cây khung nhỏ nhất



* Tự học - Áp dụng thuật toán Kruskal (cơ bản, ngẫu nhiên)

Câu hỏi 1
Đúng
Đạt điểm 1.00
Đặt cờ

Cho đồ thị vô hướng có trọng số gồm 6 đỉnh và 9 cung như bên dưới.
Hãy áp dụng thuật toán Kruskal để tìm cây khung vô hướng nhỏ nhất (cây khung có tổng trọng số nhỏ nhất).
Ghi kết quả trung gian vào bảng.

Bước 1 (sắp xếp): Sắp xếp các cung theo trọng số tăng dần (đúng ra là không giảm, nhưng nói tăng dần cho dễ nhớ).

- Các cột u, v, w ghi các cung (u, v) và trọng số (w) của chúng theo thứ tự trọng số tăng dần.

Bước 2 (lặp): Lần lượt xét từng cung theo thứ tự đã sắp xếp ở bước 1, với mỗi cung xem xét thêm nó vào cây hay không. Một cung sẽ được thêm vào cây nếu như thêm nó vào không tạo thành chu trình.

Cột Thêm vào cây ghi thêm (hoặc **có** hoặc **x** hoặc **yes**) nếu cung này được thêm vào cây, ghi **không** (hoặc **không thêm** hoặc **no**) nếu không thêm cung này vào cây. Hãy vẽ hình trên giấy và dùng mắt kiểm tra xem việc thêm này có tạo chu trình hay không.

Bước 3 (Vẽ cây): Đưa vào các cung được chọn thêm vào cây trong bước 2, hãy vẽ cây khung nhỏ nhất trong phần **Cây khung nhỏ nhất**. Cây khung nhỏ nhất gồm tất cả các đỉnh của đồ thị gốc và các cung được thêm vào cây.

Quy ước

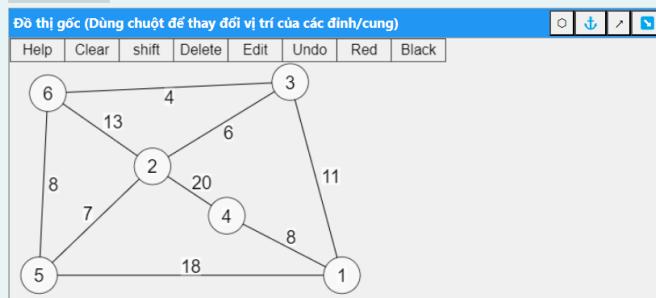
- Hai cung có trọng số giống nhau thì ghi cung nào trước cũng được.

Chú ý

- Cây kết quả phụ thuộc vào thứ tự sắp xếp của các cung.

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

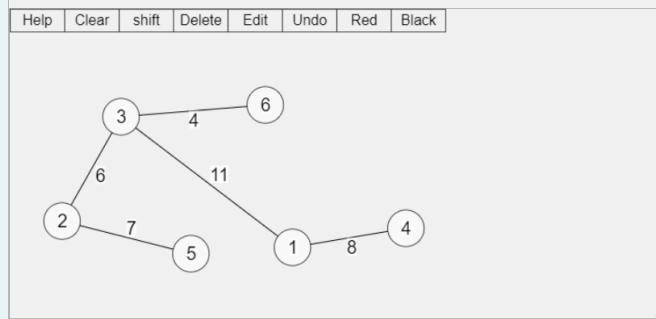
Reset answer



Áp dụng thuật toán Kruskal và ghi kết quả vào bảng

	u	v	w	Thêm vào cây?
1	3	6	4	x
2	2	3	6	x
3	2	5	7	x
4	1	4	8	x
5	5	6	8	no
6	1	3	11	x
7	2	6	13	no
8	1	5	18	no
9	2	4	20	no

Cây khung nhỏ nhất



* Tự học - Áp dụng thuật toán Kruskal (nâng cao, ví dụ)

Câu hỏi 1
Đúng
Đạt điểm 0.90
trên 1.00
Tài liệu

Cho đồ thị vô hướng có trọng số gồm 6 đỉnh và 10 cung như bên dưới.

Hãy áp dụng thuật toán Kruskal để tìm cây khung vô hướng nhỏ nhất (cây khung có tổng trọng số nhỏ nhất).
Ghi kết quả trung gian vào bảng.

Bước 1 (sắp xếp): Sắp xếp các cung theo trọng số tăng dần (đứng ra là không giảm, nhưng nói tăng dần cho dễ nhớ).

- Các cột **u**, **v**, **w** ghi các cung (u, v) và trọng số (w) của chúng theo thứ tự trọng số tăng dần.

Bước 2 (lặp): Lần lượt xét từng cung theo thứ tự đã sắp xếp ở bước 1, với mỗi cung xem xét thêm nó vào cây hay không. Một cung sẽ được thêm vào cây nếu như thêm nó vào không tạo thành chu trình.

Để kiểm tra việc thêm cung có tạo chu trình hay không ta dùng 1 mảng parent để quản lý các bộ phận liên thông của từng kết quả (kết thúc thuật toán, rừng này sẽ hợp nhất thành cây). Từ một đỉnh u, nếu lần theo parent[u] rồi parent[parent[u]], ... ta sẽ đến đỉnh gốc của bộ phận liên thông (BPLT) chứa u. Khi đó, ta gán parent[u] = u với mọi u.

- Cột **root_u** ghi chỉ số của đỉnh gốc của BPLT của u. Cột **root_v** ghi chỉ số của đỉnh gốc của BPLT.
- Nếu root_u = root_v, thì thêm cung (u, v) sẽ tạo thành chu trình. Vì thế ở cột **Thêm vào cây** ta ghi **không** (hoặc **không thêm** hoặc **no**).
- Ngược lại, nếu root_u ≠ root_v, làm các công việc sau:
 - Ở cột **Thêm vào cây**, ghi **thêm** (hoặc **có** hoặc **yes**). Khi thêm cung (u, v) vào rừng kết quả, ta phải hợp nhất BPLT chứa u và BPLT chứa v thành một BPLT duy nhất. Để đơn giản, trong bài tập này ta quy ước: **đem gốc của u làm con của gốc u**, có nghĩa là gán **parent[root_v] = root_u** (xem slides bài giảng).
 - Cập nhật lại hình vẽ trong phần **Quản lý các BPLT**.

Bước 3 (Vẽ cây): Dưa vào các cung được chọn thêm vào cây trong bước 2, hãy vẽ cây khung nhỏ nhất trong phần **Cây khung nhỏ nhất**. Cây khung nhỏ nhất gồm tất cả các đỉnh của đồ thị gốc và các cung được thêm vào cây.

Quy ước:

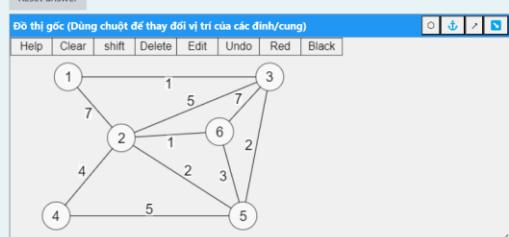
- Hai cung có trọng số giống nhau thì ghi cung nào trước cũng được.

Chú ý:

- Cây kết quả phụ thuộc vào thứ tự sắp xếp của các cung.

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

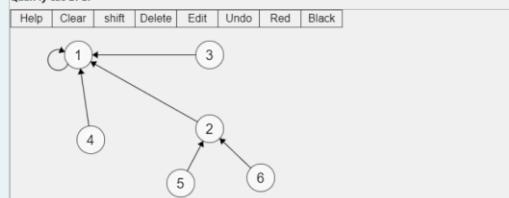
Reset answer



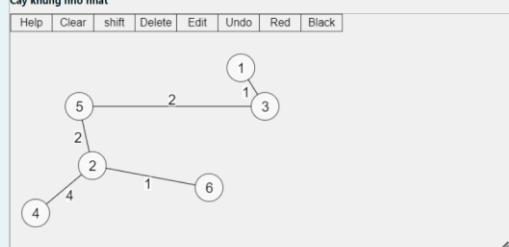
Áp dụng thuật toán Kruskal và ghi kết quả vào bảng

	u	v	w	root_u	root_v	Thêm vào cây?
1	1	3	1	1	3	x
2	2	6	1	2	6	x
3	2	5	2	2	5	x
4	3	5	2	1	2	x
5	5	6	3	1	1	no
6	2	4	4	1	4	x
7	2	3	5	1	1	no
8	4	5	5	1	1	no
9	1	2	7	1	1	no
10	3	6	7	1	1	no

Quản lý các BPLT



Cây khung nhỏ nhất



* Tự học - Áp dụng thuật toán Kruskal (nâng cao, ngẫu nhiên)

Câu hỏi 1
Đang
Đạt điểm 1,00
trên 1,00
1% Đạt cơ

Cho đồ thị vô hướng có trọng số gồm 6 đỉnh và 10 cung như bên dưới.

Hãy áp dụng thuật toán Kruskal để tìm cây khung nhỏ nhất (cây khung có tổng trọng số nhỏ nhất). Ghi kết quả trung gian vào bảng.

Bước 1 (sắp xếp): Sắp xếp các cung theo trọng số tăng dần (dùng ra là không giảm, nhưng nói tăng dần cho dễ nhớ).

- Các cột u , v , w ghi các cung (u, v) và trọng số (w) của chúng theo thứ tự trọng số tăng dần.

Bước 2 (lặp): Lần lượt xét từng cung theo thứ tự đã sắp xếp ở bước 1, với mỗi cung xem xét thêm nó vào cây hay không. Một cung sẽ được thêm vào cây nếu như thêm nó vào không tạo thành chu trình.

Để kiểm tra việc thêm cung có tạo chu trình hay không ta dùng 1 mảng parent để quản lý các bộ phận liên thông của rừng kết quả (kết thúc thuật toán, rừng này sẽ hợp nhất thành cây). Từ một đỉnh u , nếu lần theo parent[u] rồi parent[parent[u]] ... ta sẽ đến đỉnh gốc của bộ phận liên thông (BPLT) chứa u . Khi đó, ta gán parent[u] = u với mọi u .

- Cột $root_u$ ghi chỉ số của đỉnh gốc của BPLT của u . Cột $root_v$ ghi chỉ số của đỉnh gốc của BPLT của v .
- Nếu $root_u = root_v$, thì thêm cung (u, v) sẽ tạo thành chu trình. Vì thế ở cột **Thêm vào cây ta ghi không** (hoặc **không thêm** hoặc **no**).
- Ngược lại, nếu $root_u \neq root_v$, làm các công việc sau:
 - Ở cột **Thêm vào cây**, ghi **thêm** (hoặc **có** hoặc **x** hoặc **yes**). Khi thêm cung (u, v) vào rừng kết quả, ta phải hợp nhất BPLT chứa u và BPLT chứa v thành một BPLT duy nhất. Để đơn giản, trong bài này ta quy định **đỉnh gốc của lâm can của gốc u** , có nghĩa là gán $parent[root_v] = root_u$ (xem slides bài giảng).
 - Cập nhật lại hình vẽ trong phần **Quản lý các BPLT**.

Bước 3 (Vẽ cây): Dựa vào các cung được chọn thêm vào cây trong bước 2, hãy vẽ cây khung nhỏ nhất trong phần **Cây khung nhỏ nhất**. Cây khung nhỏ nhất gồm tất cả các đỉnh của đồ thị gốc và các cung được thêm vào cây.

Quy ước

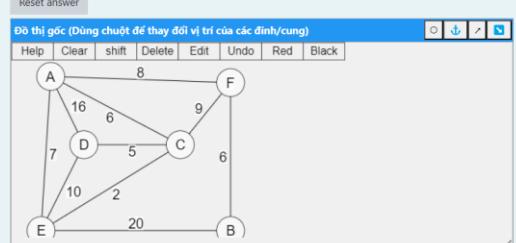
- Hai cung có trọng số giống nhau thì ghi cung nào trước cũng được.

Chú ý

- Cây kết quả phụ thuộc vào thứ tự sắp xếp của các cung.

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

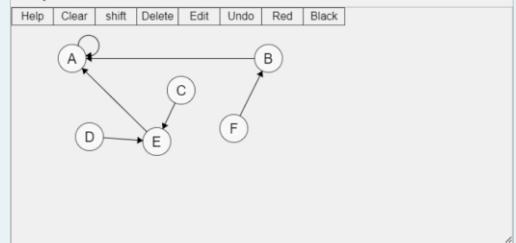
Reset answer



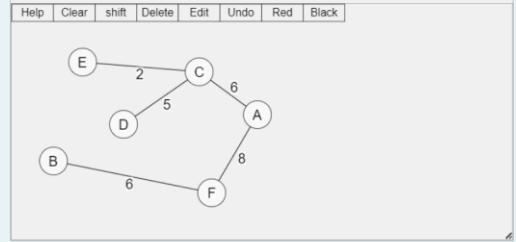
Áp dụng thuật toán Kruskal và ghi kết quả vào bảng

	u	v	w	root_u	root_v	Thêm vào cây?
1	E	C	2	E	C	x
2	C	D	5	E	D	x
3	A	C	6	A	E	x
4	B	F	6	B	F	x
5	A	E	7	A	A	no
6	A	F	8	A	B	x
7	C	F	9	A	A	no
8	D	E	10	A	A	no
9	A	D	16	A	A	no
10	B	E	20	A	A	no

Quản lý các BPLT



Cây khung nhỏ nhất



* Tự học - Áp dụng thuật toán Prim (ví dụ)

Câu hỏi 1
Đúng
Đạt điểm 1.00
trên 1.00
Đặt cờ

Cho đồ thị vô hướng có trọng số không âm gồm 6 đỉnh và 10 cung như bên dưới.

Hãy áp dụng thuật toán Prim để tìm cây khung nhỏ nhất từ đỉnh 1. Ở mỗi vòng lặp i ghi lại kết quả trung gian vào các ô tương ứng. Mỗi ô ở cột u ghi hai giá trị $\pi[u]$ và $p[u]$ cách nhau bằng dấu /, ví dụ: cột 3 được ghi là 4/6 thì có nghĩa là $\pi[3] = 4$ và $p[3] = 6$.

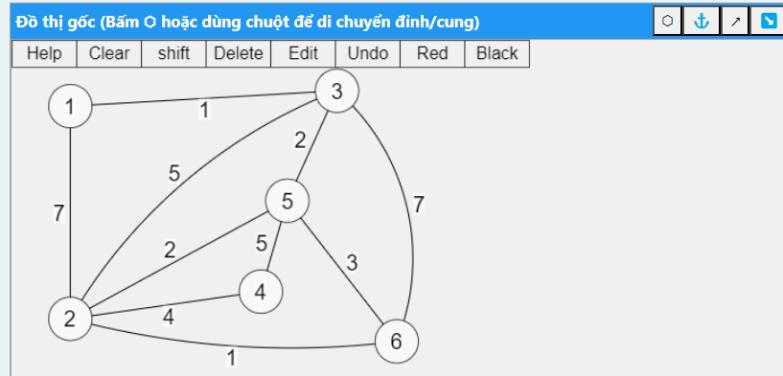
Dựa vào các $p[u]$ sau cùng, hãy vẽ cây khung nhỏ nhất. Cây khung nhỏ nhất gồm tất cả các đỉnh của đồ thị gốc và các cung $(p[u], u)$.

Quy ước

- Sử dụng oo (hai ký tự o) để biểu diễn giá trị vô cùng.
- Nếu giá trị $p[u]$ chưa có, có thể bỏ trống, ghi -1 hoặc ghi -.
- Đỉnh không cập nhật nữa thì bỏ trống ở cột đó hoặc cũng có thể ghi lại giống hệt hàng bên trên.
- Đánh dấu đỉnh bằng dấu *
- Nếu có 2 đỉnh có cùng giá trị π thì chọn đỉnh có số thứ tự nhỏ.
- Cột công việc có thể không ghi.

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

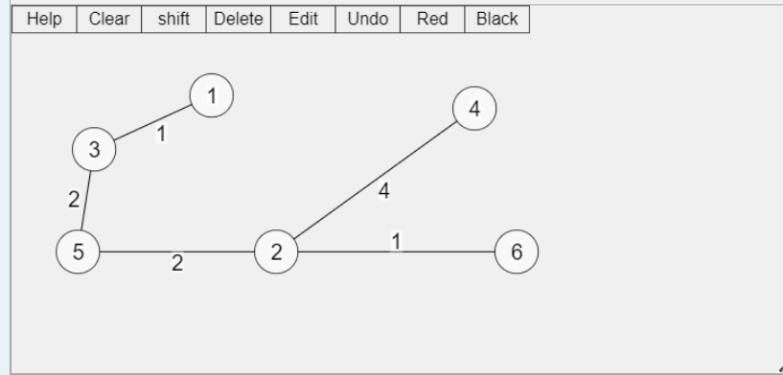
Reset answer



1. Áp dụng thuật toán Prim và ghi kết quả vào bảng

	1	2	3	4	5	6	Công việc
Khởi tạo	0/-	oo	oo	oo	oo	oo	
1	*	7/1	1/1				
2		5/3	*		2/3	7/3	
3		2/5		5/5	*	3/5	
4		*		4/2		1/2	
5						*	

2. Vẽ cây khung nhỏ nhất



* Tự học - Áp dụng thuật toán Prim (ngẫu nhiên)

Câu hỏi 1
Đúng
Đạt điểm 1.00
trên 1.00
▼ Đặt cờ

Cho đồ thị **vô hướng** có trọng số không âm gồm 6 đỉnh và 10 cung như bên dưới.

Hãy áp dụng thuật toán Prim để tìm cây khung nhỏ nhất từ đỉnh 6. Ở mỗi vòng lặp i ghi lại kết quả trung gian vào các ô tương ứng. Mỗi ô ở cột u ghi hai giá trị $\pi[u]$ và $p[u]$ cách nhau bằng dấu /, ví dụ: cột 3 được ghi là 4/6 thì có nghĩa là $\pi[3] = 4$ và $p[3] = 6$.

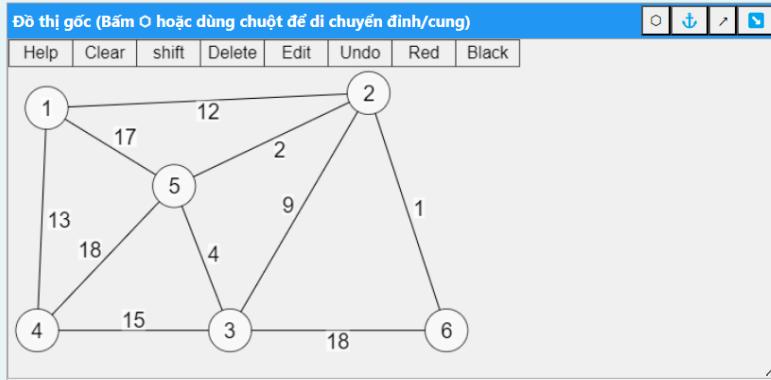
Dựa vào các $p[u]$ sau cùng, hãy vẽ cây khung nhỏ nhất. Cây khung nhỏ nhất gồm tất cả các đỉnh của đồ thị và các cung $(p[u], u)$.

Quy ước

- Sử dụng **oo** (hai ký tự o) để biểu diễn giá trị vô cùng.
- Nếu giá trị $p[u]$ chưa có, có thể bỏ trống, ghi -1 hoặc ghi -.
- Đỉnh không cập nhật nữa thì bỏ trống ở cột đó hoặc cũng có thể ghi lại giống hệt hàng bên trên.
- Đánh dấu đỉnh bằng dấu *
- Nếu có 2 đỉnh có cùng giá trị π thì chọn đỉnh có số thứ tự nhỏ.
- Cột công việc có thể không ghi.

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

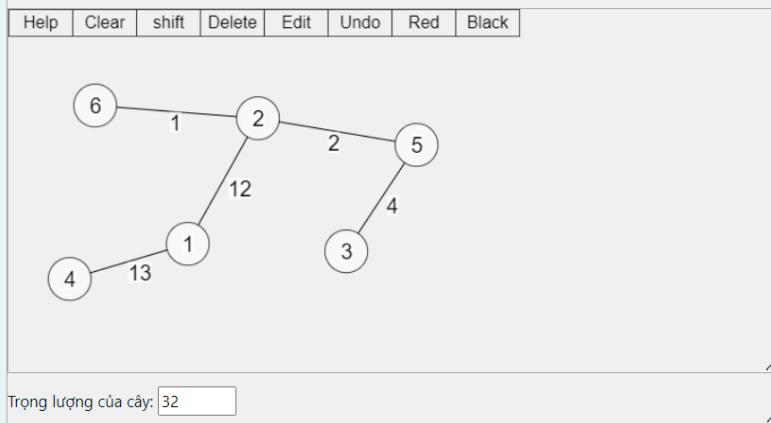
Reset answer



1. Áp dụng thuật toán Prim và ghi kết quả vào bảng

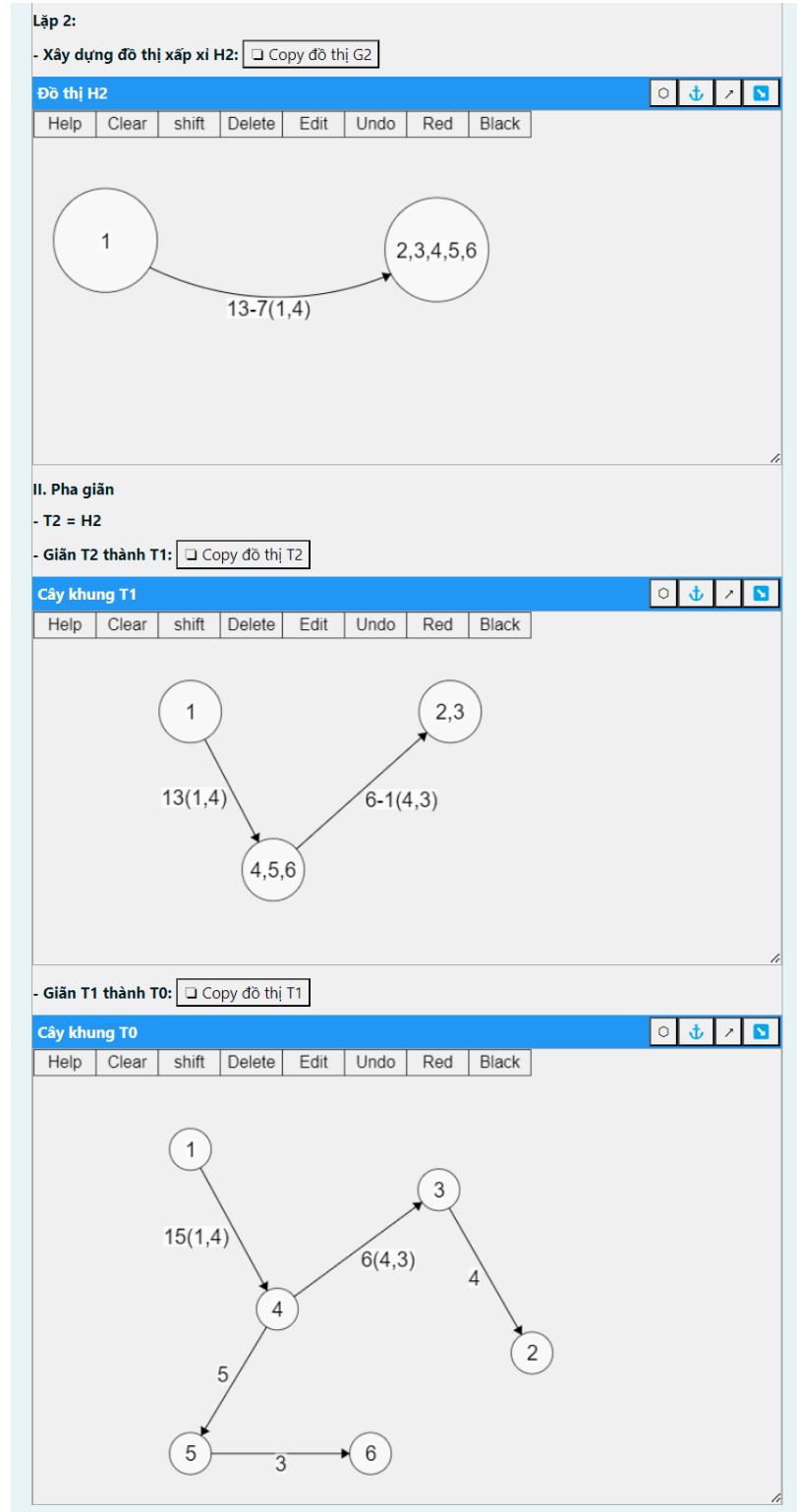
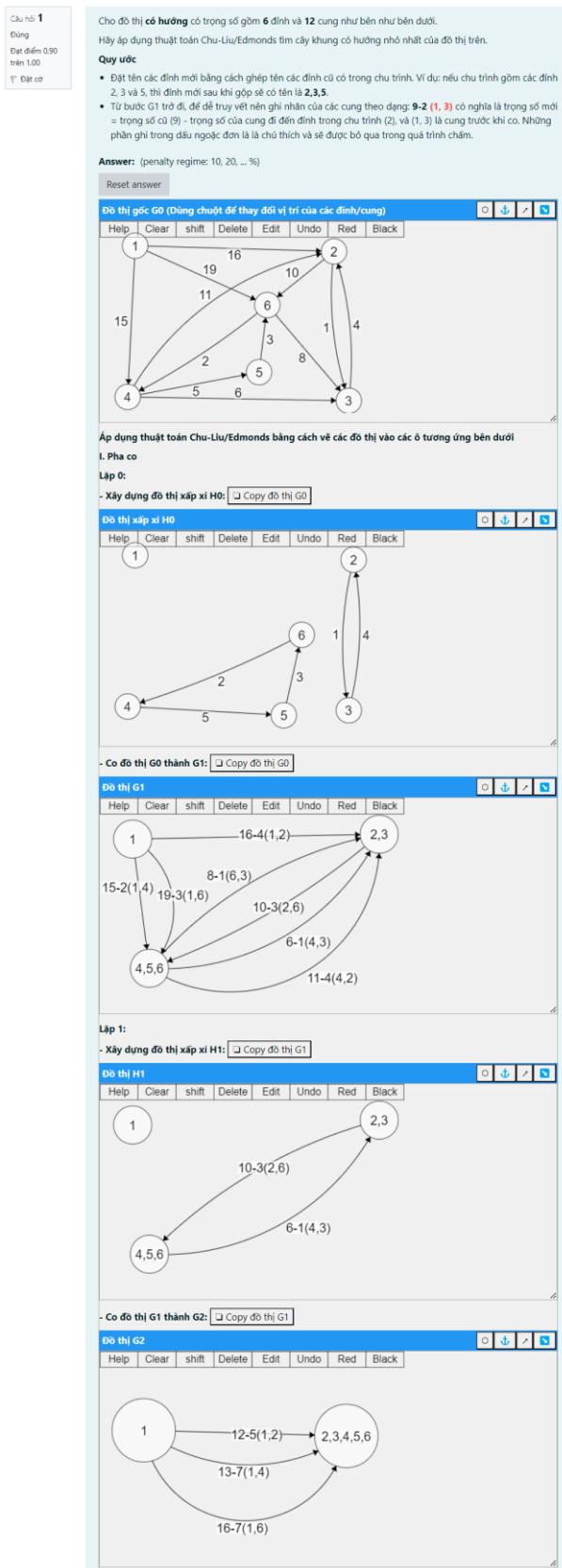
	1	2	3	4	5	6	Công việc
Khởi tạo	oo	oo	oo	oo	oo	0/-	
1		1/6	18/6			*	
2	12/2	*	9/2		2/2		
3			4/5	18/5	*		
4			*	15/3			
5	*			13/1			

2. Vẽ cây khung nhỏ nhất



Tuần 12 - Cây (tiếp theo)

* Tự học - Áp dụng thuật toán Chu-Liu/Edmonds (ví dụ)



* Tự học - Áp dụng thuật toán Chu-Liu/Edmonds (ngẫu nhiên)

Câu hỏi 1
Đúng
Đạt điểm 1,00
trên 1,00
Vì Đặt cờ

Cho đồ thị có hướng có trọng số gồm 6 đỉnh và 11 cung như bên dưới.

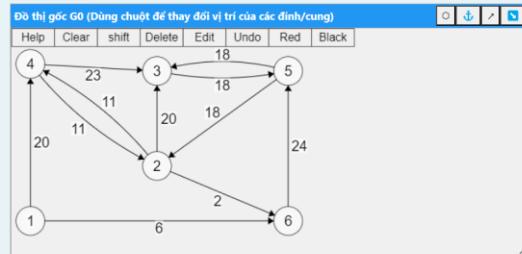
Hãy áp dụng thuật toán Chu-Liu/Edmonds tìm cây khung có hướng nhỏ nhất của đồ thị trên.

Quy ước

- Đặt tên các đỉnh mới bằng cách ghép tên các đỉnh cũ có trong chu trình. Ví dụ: nếu chu trình gồm các đỉnh 2, 3 và 5, thì đỉnh mới sau khi gộp sẽ có tên là 2,3,5.
- Tù bước G1 trả đũa, để dễ truy vết nên ghi nhận của các cung theo dạng: 9-2 (1, 3) có nghĩa là trọng số mới = trọng số cũ (9) - trọng số của cung đi đến đỉnh trong chu trình (2), và (1, 3) là cung trước khi co. Những phần ghi trong dấu ngoặc đơn là chú thích và sẽ được bỏ qua trong quá trình chấm.

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

Reset answer

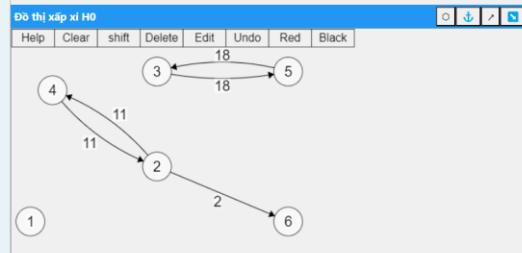


Áp dụng thuật toán Chu-Liu/Edmonds bằng cách vẽ các đồ thị vào các ô tương ứng bên dưới

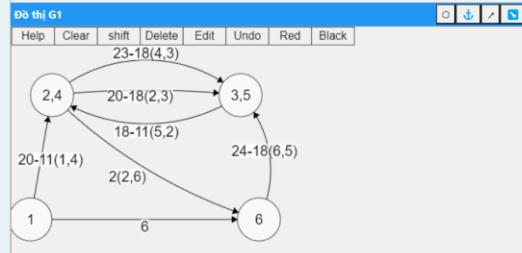
I. Pha co

Lập 0:

- Xây dựng đồ thị xấp xỉ H0: Copy đồ thị G0

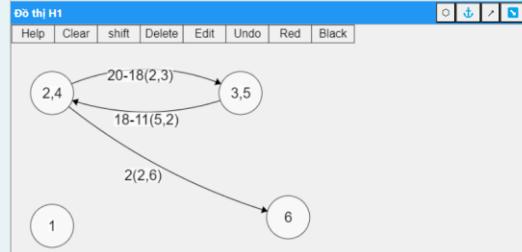


- Co đồ thị G0 thành G1: Copy đồ thị G0

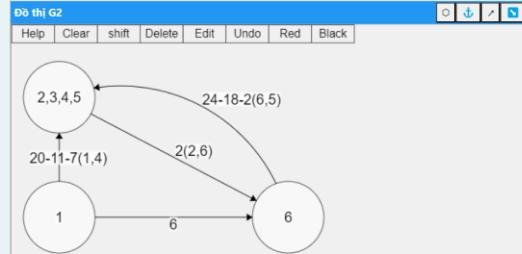


Lập 1:

- Xây dựng đồ thị xấp xỉ H1: Copy đồ thị G1

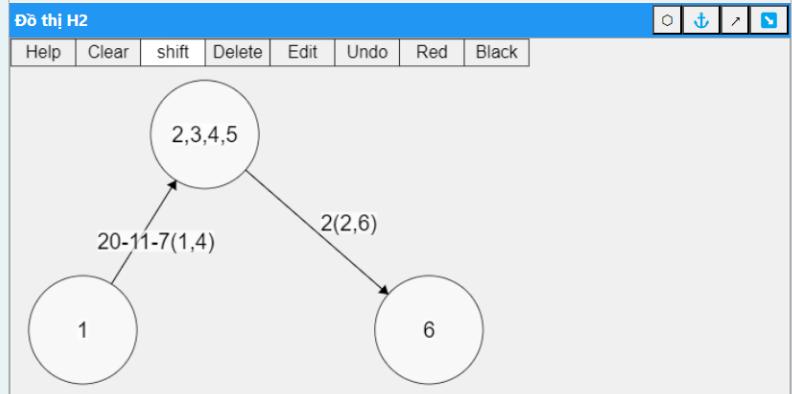


- Co đồ thị G1 thành G2: Copy đồ thị G1



Lập 2:

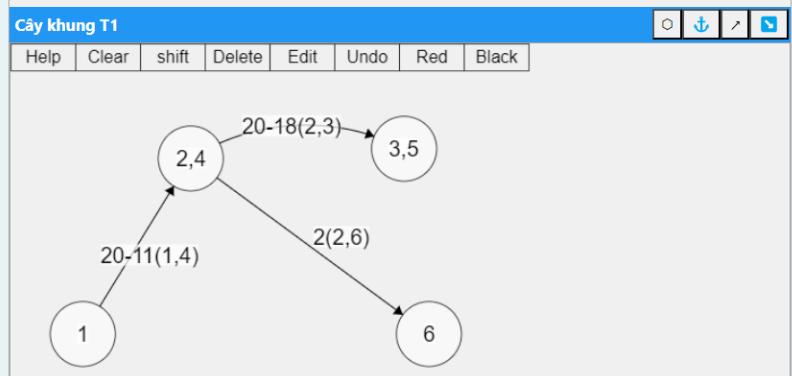
- Xây dựng đồ thị xấp xỉ H2: Copy đồ thị G2



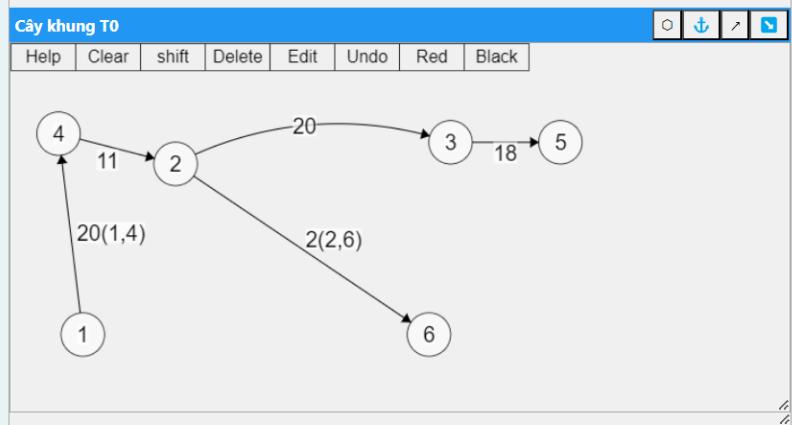
II. Pha giãn

- T2 = H2

- Giãn T2 thành T1: Copy đồ thị T2



- Giãn T1 thành T0: Copy đồ thị T1



Câu hỏi 1
Đúng
Đạt điểm 0.80 trên 1.00
! Đặt cờ

Chó đồ thị có hướng có trọng số gồm 6 đỉnh và 11 cung như bên dưới.

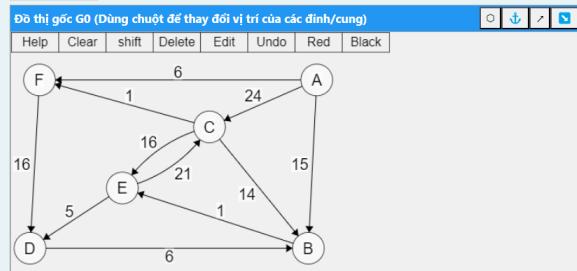
Hãy áp dụng thuật toán Chu-Liu/Edmonds tìm cây khung có hướng nhỏ nhất của đồ thị trên.

Quy ước

- Đặt tên các đỉnh mới bằng cách ghép tên các đỉnh cũ trong chu trình. Ví dụ: nếu chu trình gồm các đỉnh B, C và E, thì đỉnh mới sau khi gộp sẽ có tên là **B,C,E**.
- Từ bước G1 trở đi, để dễ truy vết nên ghi nhận của các cung theo dạng: **9-2 (A, C)** có nghĩa là trọng số mới = trọng số cũ (9) - trọng số của cung đi đến đỉnh trong chu trình (2), và (A, C) là cung trước khi co. Nhữngh phần ghi trong dấu ngoặc đơn là chủ thích và sẽ được bỏ qua trong quá trình chấm.

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

Reset answer

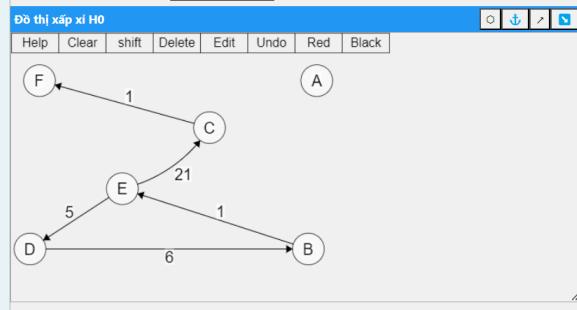


Áp dụng thuật toán Chu-Liu/Edmonds bằng cách vẽ các đồ thị vào các ô tương ứng bên dưới

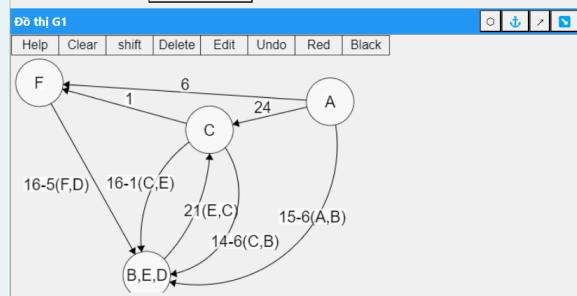
I. Pha co

Lập 0:

- Xây dựng đồ thị xấp xỉ H0: Copy đồ thị G0

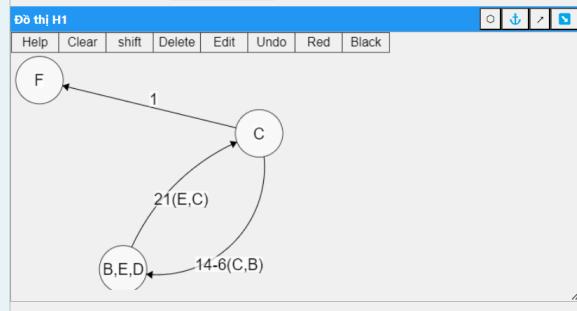


- Co đồ thị G0 thành G1: Copy đồ thị G0

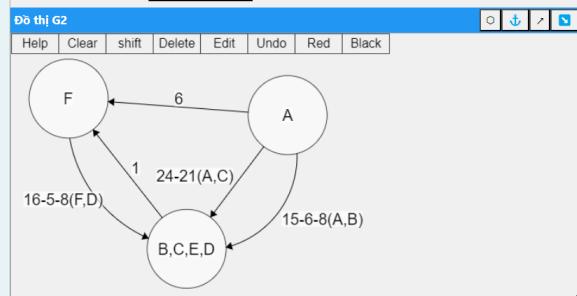


Lập 1:

- Xây dựng đồ thị xấp xỉ H1: Copy đồ thị G1

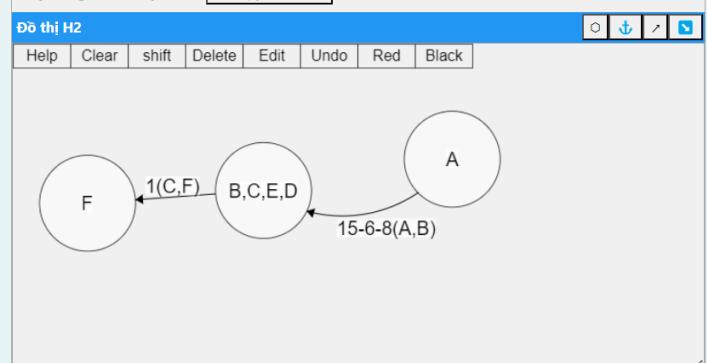


- Co đồ thị G1 thành G2: Copy đồ thị G1



Lập 2:

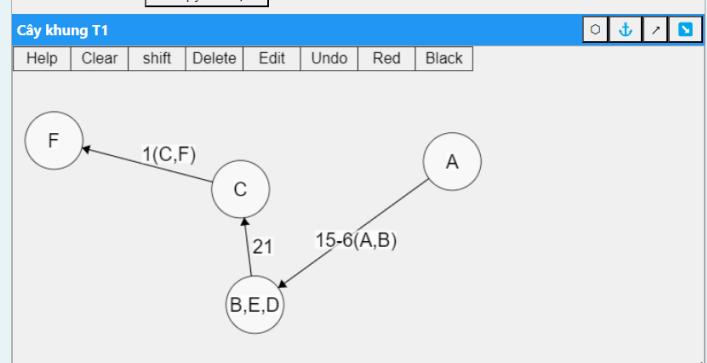
- Xây dựng đồ thị xấp xỉ H2: Copy đồ thị G2



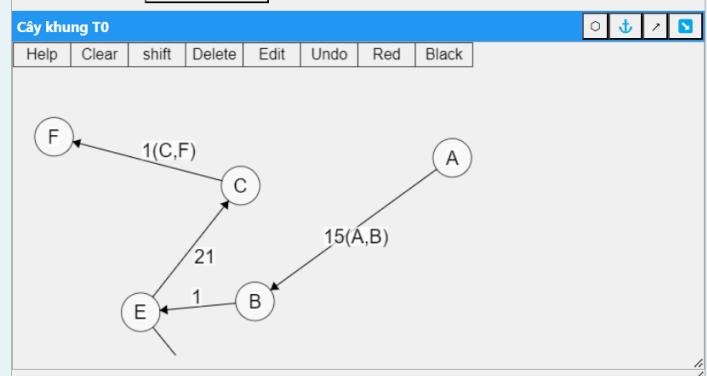
II. Pha giãn

- T2 = H2

- Giãn T2 thành T1: Copy đồ thị T2



- Giãn T1 thành T0: Copy đồ thị T1



Tuần 14 - Luồng cực đại trên mạng

* [Tự học] - 1. Khởi tạo luồng

Câu hỏi 1

Đúng
Đạt điểm 1,00
trên 1,00

Đặt cờ

Cho mạng với đỉnh phát $s = 1$ và đỉnh thu $t = 6$ được biểu diễn bằng đồ thị như bên dưới.

Hãy gán các luồng (số nguyên) trên cung sao cho tạo thành một luồng hợp lệ có giá trị lớn hơn 11.

Quy ước

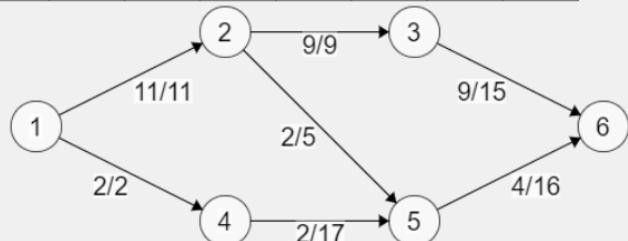
- Ghi luồng trên cung theo mẫu: f/c hoặc f (ví dụ: 3/5 hoặc 3) với f là luồng trên cung và c là khả năng thông qua của cung.

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

Reset answer

Mạng (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo Red Black



Câu hỏi 1

Đúng
Đạt điểm 1,00
trên 1,00

Đặt cờ

Cho mạng với đỉnh phát $s = 1$ và đỉnh thu $t = 6$ được biểu diễn bằng đồ thị như bên dưới.

Hãy gán các luồng (số nguyên) trên cung sao cho tạo thành một luồng hợp lệ có giá trị lớn hơn 11.

Quy ước

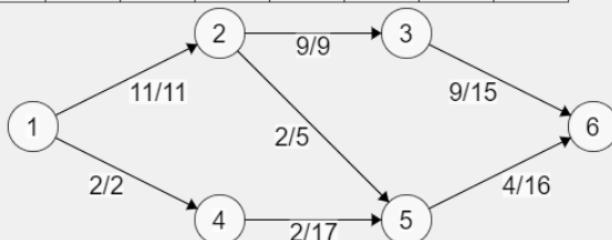
- Ghi luồng trên cung theo mẫu: f/c hoặc f (ví dụ: 3/5 hoặc 3) với f là luồng trên cung và c là khả năng thông qua của cung.

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

Reset answer

Mạng (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

Help Clear shift Delete Edit Undo Red Black



* [Tự học] - 2. Áp dụng phương pháp Ford - Fulkerson

Câu hỏi 1
Đúng
Đạt điểm 0.90
trên 1.00
T⁺ Đặt cờ

Cho mạng với đỉnh phát $s = A$ và đỉnh thu $t = F$ được biểu diễn bằng đồ thị như bên dưới. Trọng số của các cung chính là khả năng thông qua của nó.

Hãy áp dụng phương pháp Ford - Fulkerson tìm luồng cực đại trong mạng trên.

Quy ước

- Ghi luồng trên cung theo mẫu: t/c hoặc f (ví dụ: $3/5$ hoặc 3) với t là luồng trên cung và c là khả năng thông qua cung.
- Đường đang luồng dùng ký hiệu \rightarrow để ngăn cách các đỉnh, ví dụ: $A \rightarrow C \rightarrow F$

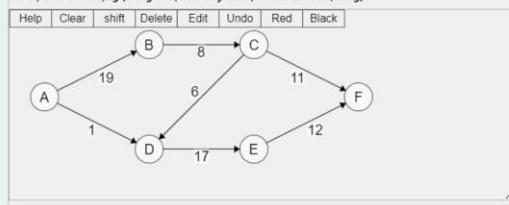
For example:

Test	Result
1. Khởi tạo luồng hợp lệ 2. Lặp tìm đường tăng luồng & tăng luồng 3. Luồng cực đại và lát cắt hợp nhất	

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

Reset answer

Đồ thị biểu diễn mạng (Dùng chuột để thay đổi vị trí của các đỉnh/cung)



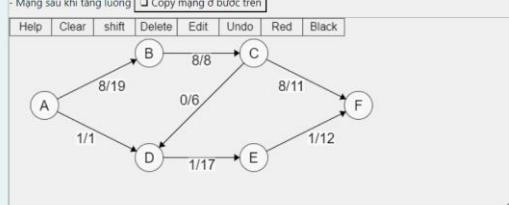
Áp dụng phương pháp Ford-Fulkerson tìm luồng cực đại trong mạng trên.

I. Khởi tạo một luồng hợp lệ bất kỳ có giá trị không vượt quá 1

II. Lập tìm đường tăng luồng + tăng luồng

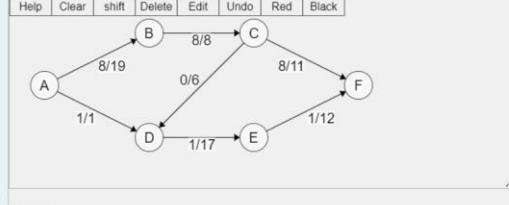
Lần lập 1

- Tìm một đường tăng luồng (augmenting path) bất kỳ: $A \rightarrow B \rightarrow C \rightarrow F$ ví dụ: $A \rightarrow C \rightarrow F$
- Lượng luồng tăng thêm (bottle neck capacity): 8
- Mạng sau khi tăng luồng Copy mạng ở bước trên



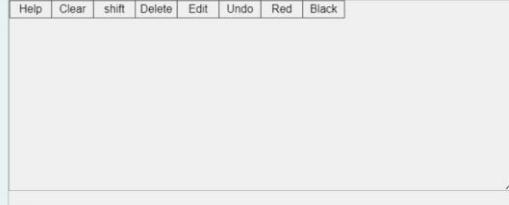
Lần lập 2

- Tìm một đường tăng luồng (augmenting path) bất kỳ: $A \rightarrow C \rightarrow F$ ví dụ: $A \rightarrow C \rightarrow F$
- Lượng luồng tăng thêm (bottle neck capacity): 8
- Mạng sau khi tăng luồng Copy mạng ở bước trên



Lần lập 3

- Tìm một đường tăng luồng (augmenting path) bất kỳ: $A \rightarrow C \rightarrow F$ ví dụ: $A \rightarrow C \rightarrow F$
- Lượng luồng tăng thêm (bottle neck capacity): 8
- Mạng sau khi tăng luồng Copy mạng ở bước trên



III. Kết quả

Luồng cực đại: 9

Lát cắt hợp nhất tách s và t (ngăn cách các đỉnh bằng dấu phẩy), ví dụ A, C, D, E :

S = [A,B] và T = [C,D,E,F]

Câu hỏi 1
Đúng
Đạt điểm 0.90
trên 1.00
T⁺ Đặt cờ

Cho mạng với đỉnh phát $s = 1$ và đỉnh thu $t = 6$ được biểu diễn bằng đồ thị như bên dưới. Trọng số của các cung chính là khả năng thông qua của nó.

Hãy áp dụng phương pháp Ford - Fulkerson tìm luồng cực đại trong mạng trên.

Quy ước

- Ghi luồng trên cung theo mẫu: t/c hoặc f (ví dụ: $3/5$ hoặc 3) với t là luồng trên cung và c là khả năng thông qua cung.
- Đường đang luồng dùng ký hiệu \rightarrow để ngăn cách các đỉnh, ví dụ: $1 \rightarrow 3 \rightarrow 6$

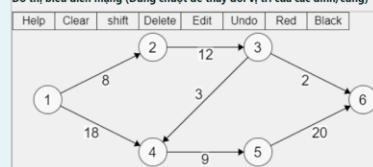
For example:

Test	Result
1. Khởi tạo luồng hợp lệ 2. Lập tìm đường tăng luồng & tăng luồng 3. Luồng cực đại và lát cắt hợp nhất	

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

Reset answer

Đồ thị biểu diễn mạng (Dùng chuột để thay đổi vị trí của các đỉnh/cung)



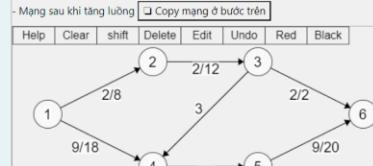
Áp dụng phương pháp Ford-Fulkerson tìm luồng cực đại trong mạng trên.

I. Khởi tạo một luồng hợp lệ bất kỳ có giá trị không vượt quá 2

II. Lập tìm đường tăng luồng + tăng luồng

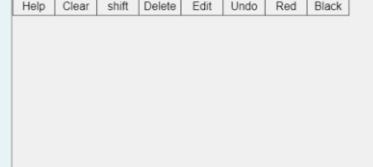
Lần lập 1

- Tìm một đường tăng luồng (augmenting path) bất kỳ: $1 \rightarrow 4 \rightarrow 5 \rightarrow 6$ ví dụ: $1 \rightarrow 3 \rightarrow 6$
- Lượng luồng tăng thêm (bottle neck capacity): 11
- Mạng sau khi tăng luồng Copy mạng ở bước trên



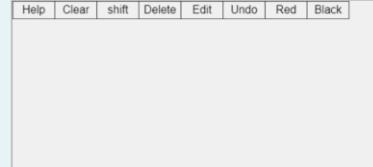
Lần lập 2

- Tìm một đường tăng luồng (augmenting path) bất kỳ: $1 \rightarrow 3 \rightarrow 6$ ví dụ: $1 \rightarrow 3 \rightarrow 6$
- Lượng luồng tăng thêm (bottle neck capacity): 11
- Mạng sau khi tăng luồng Copy mạng ở bước trên



III. Kết quả

- Tìm một đường tăng luồng (augmenting path) bất kỳ: $1 \rightarrow 3 \rightarrow 6$ ví dụ: $1 \rightarrow 3 \rightarrow 6$
- Lượng luồng tăng thêm (bottle neck capacity): 11
- Mạng sau khi tăng luồng Copy mạng ở bước trên



III. Kết quả

Luồng cực đại: 11

Lát cắt hợp nhất tách s và t (ngăn cách các đỉnh bằng dấu phẩy), ví dụ 1, 3, 4 :

S = [1,2,3,4] và T = [5,6]

* [Tự học] - 3. Gán nhãn tìm đường tăng luồng

Câu hỏi 1
Đúng
Đạt điểm 0,85
trên 1,00
 Đặt cờ

Cho mạng với đỉnh phát $s = A$ và đỉnh thu $t = F$ được biểu diễn bằng đồ thị như bên dưới.

Giả sử mạng đã được khởi tạo với luồng như được ghi trên các cung. Hãy áp dụng thuật toán Ford - Fulkerson tìm đường tăng luồng bằng cách gán các định dùng hàng đợi (theo thuật toán Edmonds-Karp).

Vẽ lại mạnq sau khi đã tăng luồng.

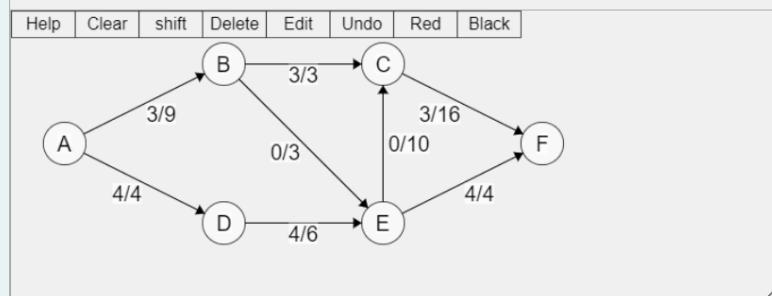
Quy ước

- Ghi nhãn các đỉnh theo mẫu: (+, A, oo) hoặc (-, D, 5)
 - Cột **hàng đợi** ghi các đỉnh đang có trong hàng đợi, ngăn cách bằng dấu phẩy. Đầu hàng đợi (front) nằm bên trái. Ví dụ: B, D
 - Ghi luồng trên cung theo mẫu: f/c hoặc f (ví dụ: 3/5 hoặc 3) với f là luồng trên cung và c là khả năng thông qua cung.

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

[Reset answer](#)

Mạng và luồng khởi tạo (Dùng chuột để thay đổi vị trí của các đỉnh/cung)

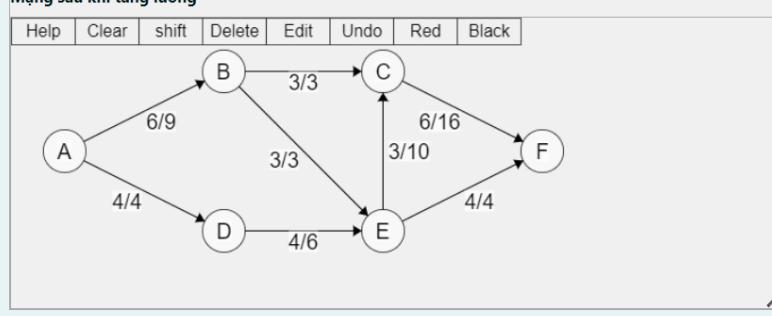


Gán nhãn các đỉnh dùng hàng đợi

- Đường tăng luồng (augmenting path): A->B->E->C->F ví dụ: A -> C -> F

- **Lực lượng tăng thêm** (bottle neck capacity):

Mang sau khi tăng luồng



Câu hỏi 1

Đúng

Đạt điểm 0.90
trên 1.00

Đặt cờ

Cho mạng với đỉnh phát s = 1 và đỉnh thu t = 6 được biểu diễn bằng đồ thị như bên dưới.

Giả sử mạng đã được khởi tạo với luồng như được ghi trên các cung. Hãy áp dụng thuật toán Ford - Fulkerson tìm đường tăng luồng bằng cách gán các đỉnh dùng hàng đợi (theo thuật toán Edmonds-Karp).

Vẽ lại mạng sau khi đã tăng luồng.

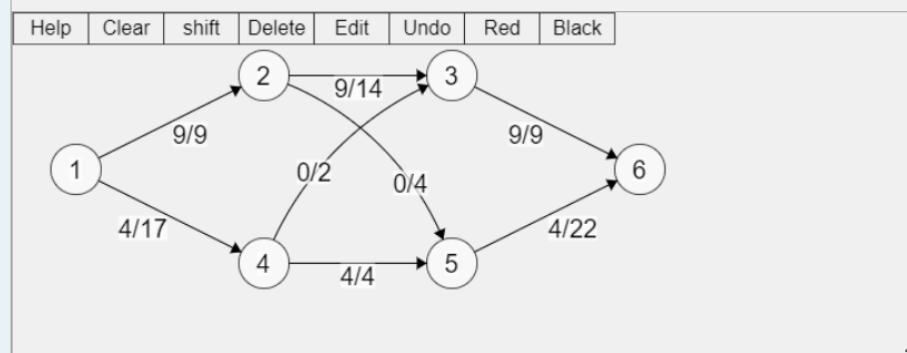
Quy ước

- Ghi nhãn các đỉnh theo mẫu: (+, 1, oo) hoặc (-, 4, 5)
- Cột **hàng đợi** ghi các đỉnh đang có trong hàng đợi, ngăn cách bằng dấu phẩy. Đầu hàng đợi (front) nằm bên trái. Ví dụ: 2, 4
- Ghi luồng trên cung theo mẫu: f/c hoặc f (ví dụ: 3/5 hoặc 3) với f là luồng trên cung và c là khả năng thông qua của cung.

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

Reset answer

Mạng và luồng khởi tạo (Dùng chuột để thay đổi vị trí của các đỉnh/cung)



Gán nhãn các đỉnh dùng hàng đợi

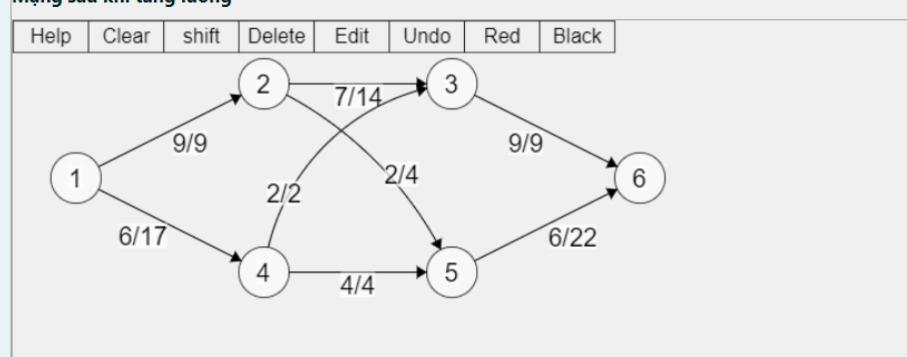
	1	2	3	4	5	6	Hàng đợi
Khởi tạo	(+, 1, oo)						1
1				(+, 1, 13)			4
2			(+, 4, 2)				3
3		(-, 3, 2)					2
4					(+, 2, 2)		5
5						(+, 5, 2)	6

Add row Delete row

- Đường tăng luồng (augmenting path): 1->4->3->2->5->6 ví dụ: 1 -> 3 -> 6

- Lượng luồng tăng thêm (bottle neck capacity): 2

Mạng sau khi tăng luồng



* [Tự học] - 4. Áp dụng thuật toán Edmonds-Karp (Ford-Fulkerson hoàn chỉnh)

Câu hỏi 1
Đúng
Đạt điểm 0,70
trên 1,00
Đặt cờ

Cho mạng với đỉnh phát $s = 1$ và đỉnh thu $t = 6$ được biểu diễn bằng đồ thị như bên dưới.
Hãy áp dụng thuật toán Ford - Fulkerson tìm luồng cực đại trong mạng. Thuật toán gồm 2 bước chính:

- Khởi tạo một luồng hợp lệ bất kỳ (thường là luồng có giá trị 0)
- Lặp để tìm tăng luồng. Việc tìm đường tăng luồng được thực hiện bằng cách gán các đỉnh thành hàng đợi (theo thuật toán Edmonds-Karp)

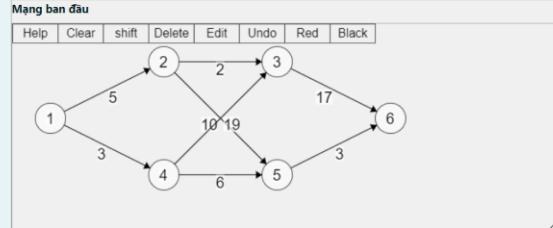
Sau bước gán nhãn, nếu tìm được đường tăng luồng, hãy vẽ lại mạng sau khi tăng luồng và tiếp tục. Ngược lại nếu không tìm được đường tăng luồng, thuật toán kết thúc, hãy biết lát cắt hẹp nhất (S, T) và giá trị luồng cực đại trong mạng.

Quy ước

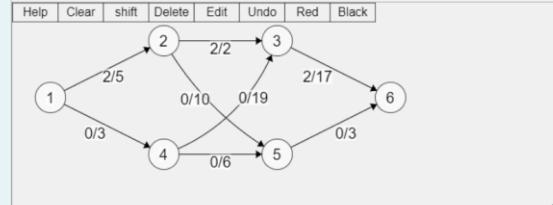
- Ghi nhãn các đỉnh theo mẫu: $(d[u], p[u], o[u])$ ví dụ: $(+, 1, oo)$, $(-, 4, 5)$
- Cột Hàng đợi ghi các đỉnh đang có trong hàng đợi, ngăn cách bằng dấu phẩy. Đầu hàng đợi (front) nằm bên trái. Ví dụ: $(+, 1, 3)$
- Ghi luồng trên cung theo mẫu: f/c hoặc f (ví dụ: $3/5$ hoặc 3) với f là luồng trên cung và c là khả năng thông qua cung
- Khi xét đỉnh để gán nhãn, gán nhãn đỉnh liên quan đến cung thuận trước, cung nghịch sau, đỉnh có thứ tự nhỏ trước, đỉnh có thứ tự lớn sau

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

Reset answer



I. Khởi tạo một luồng hợp lệ có giá trị không vượt quá 2



II. Lặp

Lần lặp 1

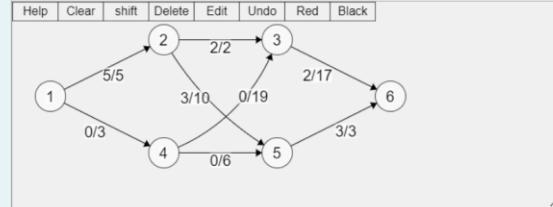
1.1 Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo	$(+, 1, oo)$						1
1		$(+, 1, 3)$		$(+, 1, 3)$			2, 4
2					$(+, 2, 3)$		4, 5
3			$(+, 4, 3)$				5, 3
4						$(+, 5, 3)$	3, 6
5							6

Tìm được đường tăng luồng Không có đường tăng luồng

- Đường tăng luồng (augmenting path): $1 \rightarrow 2 \rightarrow 5 \rightarrow 6$ ví dụ: $1 \rightarrow 3 \rightarrow 6$
- Lượng luồng tăng thêm (bottle neck capacity): 3

1.2 Mạng sau khi tăng luồng



Lần lặp 2

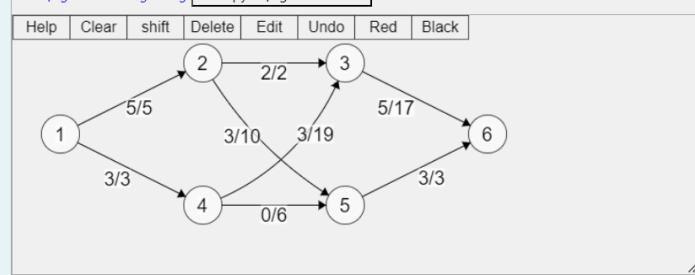
1.1 Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo	$(+, 1, oo)$						1
1					$(+, 1, 3)$		4
2					$(+, 4, 3)$		3, 5
3			$(-, 3, 2)$				$(+, 3, 3)$, 5, 6, 2
4							6, 2
5							2

Tìm được đường tăng luồng Không có đường tăng luồng

- Đường tăng luồng (augmenting path): $1 \rightarrow 4 \rightarrow 3 \rightarrow 6$ ví dụ: $1 \rightarrow 3 \rightarrow 6$
- Lượng luồng tăng thêm (bottle neck capacity): 3

1.2 Mạng sau khi tăng luồng



Lần lặp 3

1.1 Gán nhãn các đỉnh dùng hàng đợi

	1	2	3	4	5	6	Hàng đợi
Khởi tạo	$(+, 1, oo)$						1
1							
2							
3							
4							
5							

Tìm được đường tăng luồng Không có đường tăng luồng

- Luồng cực đại: 8
- Lát cắt hẹp nhất (S, T) tách s và t (ngăn cách các đỉnh bằng dấu phẩy, ví dụ: 1, 3, 4):
 - $S = 1$
 - $T = 2,3,4,5,6$

III. Kết quả

Câu hỏi 1

Đúng

Đạt điểm 0.80

trên 1.00

Còn lại 0

Cho mạng với đỉnh phát $s = A$ và đỉnh thu $t = F$ được biểu diễn bằng đồ thị như bên dưới.

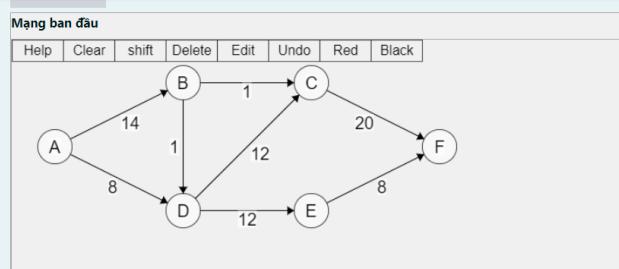
Hãy áp dụng thuật toán Ford - Fulkerson tìm luồng cực đại trong mạng. Thuật toán gồm 2 bước chính:

- Khởi tạo một luồng hợp lệ bất kỳ (thường là luồng có giá trị 0)
- Lặp để tìm tăng luồng. Việc tìm đường tăng luồng được thực hiện bằng cách gán các đỉnh dùng **hàng đợi** (theo thuật toán Edmonds-Karp)

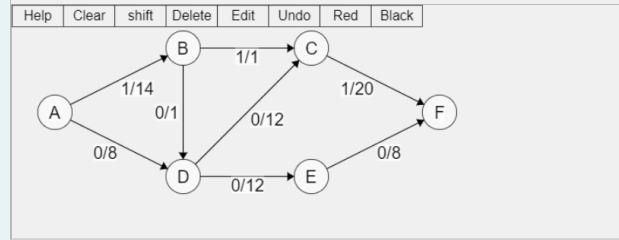
Sau bước gán nhãn, nếu tìm được đường tăng luồng, hãy vẽ lại mạng sau khi tăng luồng và tiếp tục. Ngược lại nếu không tìm được đường tăng luồng, thuật toán kết thúc, hãy cho biết lát cắt hẹp nhất (S, T) và giá trị luồng cực đại trong mạng.

Quy ước

- Ghi nhãn các đỉnh theo mẫu: $(d[u], p[u], o[u])$ ví dụ: $(+, A, \infty)$, $(-, D, 5)$
- Cột **Hàng đợi** ghi các đỉnh đang có trong hàng đợi, ngăn cách bằng dấu phẩy. Đầu hàng đợi (front) nằm bên trái. Ví dụ: B, D
- Ghi luồng trên cung theo mẫu: f/c hoặc f (ví dụ: $3/5$ hoặc 3) với f là luồng trên cung và c là khả năng thông qua của cung
- Khi xét đỉnh để gán nhãn, gán nhãn đỉnh liên quan đến **cung thuận trước**, **cung nghịch sau**, **đỉnh có thứ tự nhỏ trước**, **đỉnh có thứ tự lớn sau**



I. Khởi tạo một luồng hợp lệ có giá trị không vượt quá 1



II. Lặp

1. Lần lập 1

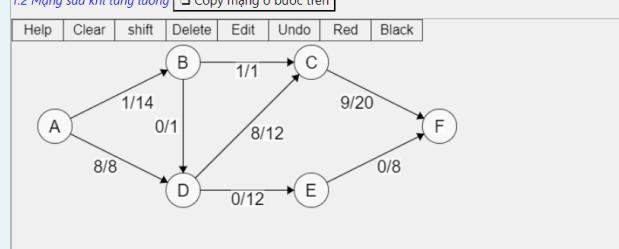
1.1 Gán nhãn các đỉnh dùng hàng đợi

	A	B	C	D	E	F	Hàng đợi
Khởi tạo	$(+, A, \infty)$						A
1		$(+, A, 13)$		$(+, A, 8)$			B, D
2							D
3							C, E
4							E, F
5							F

Tìm được đường tăng luồng Không có đường tăng luồng

- Đường tăng luồng (augmenting path): $A \rightarrow B \rightarrow D \rightarrow C \rightarrow F$ ví dụ: $A \rightarrow C \rightarrow F$
- Lượng luồng tăng thêm (bottle neck capacity): 8

1.2 Mạng sau khi tăng luồng Copy mạng ở bước trên



Lần lập 2

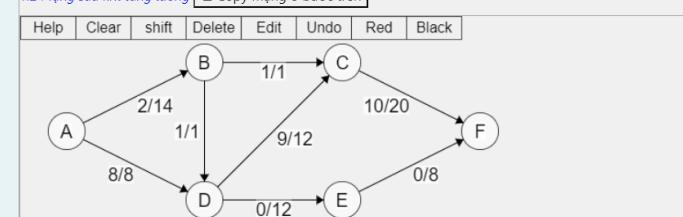
1.1 Gán nhãn các đỉnh dùng hàng đợi

	A	B	C	D	E	F	Hàng đợi
Khởi tạo	$(+, A, \infty)$						A
1		$(+, A, 13)$					B
2							D
3							C, E
4							F
5							

Tìm được đường tăng luồng Không có đường tăng luồng

- Đường tăng luồng (augmenting path): $A \rightarrow B \rightarrow D \rightarrow C \rightarrow F$ ví dụ: $A \rightarrow C \rightarrow F$
- Lượng luồng tăng thêm (bottle neck capacity): 1

1.2 Mạng sau khi tăng luồng Copy mạng ở bước trên



Lần lập 3

1.1 Gán nhãn các đỉnh dùng hàng đợi

	A	B	C	D	E	F	Hàng đợi
Khởi tạo	$(+, A, \infty)$						A
1		$(+, A, 12)$					B
2							
3							
4							
5							

Tìm được đường tăng luồng Không có đường tăng luồng

- Luồng cực đại: 10
- Lát cắt hẹp nhất (S, T) tách s và t (ngăn cách các đỉnh bằng dấu phẩy, ví dụ: A, C, D):
- $S = A, B$
- $T = C, D, E, F$

III. Kết quả