

5.1. Định nghĩa và khái niệm

Định nghĩa. Bộ đồ $G = \langle V, E \rangle$, trong đó V là tập hợp hữu hạn được gọi là tập đỉnh ($V = \{1, 2, \dots, n\}$), E là tập có thứ tự hoặc không có thứ tự các cặp đỉnh trong V được gọi là tập cạnh.

Đồ thị vô hướng. Đồ thị $G = \langle V, E \rangle$ được gọi là đồ thị vô hướng nếu các cạnh thuộc E là các bộ không tính đến thứ tự các đỉnh trong V .

Đơn đồ thị vô hướng. Đồ thị $G = \langle V, E \rangle$ được gọi là đơn đồ thị vô hướng nếu là đồ thị vô hướng và giữa hai đỉnh bất kỳ thuộc V có nhiều nhất một cạnh nối.

Đa đồ thị vô hướng. Đồ thị $G = \langle V, E \rangle$ được gọi là đơn đồ thị vô hướng nếu là đồ thị vô hướng và tồn tại một cặp đỉnh trong V có nhiều hơn một cạnh nối. Cạnh $e_1 \in E$, $e_2 \in E$ được gọi là cạnh bội nếu chúng cùng chung cặp đỉnh.

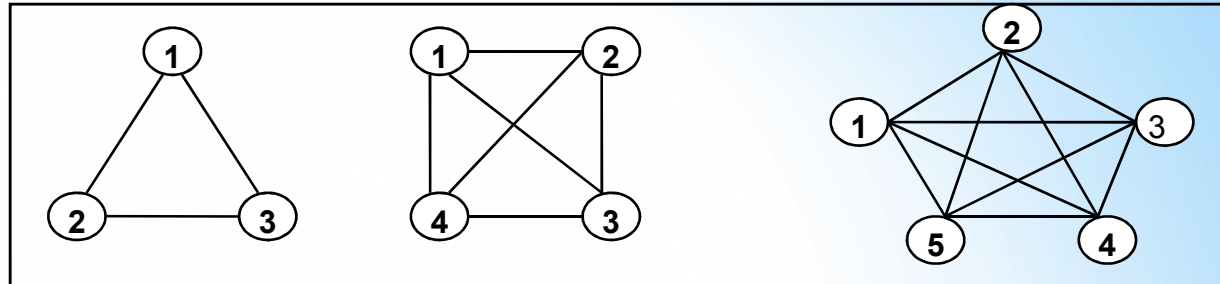
Giả đồ thị vô hướng. Đồ thị $G = \langle V, E \rangle$ bao gồm V là tập đỉnh, E là họ các cặp không có thứ tự gồm hai phần tử (hai phần tử không nhất thiết phải khác nhau) trong V được gọi là các cạnh. Cạnh e được gọi là khuyên nếu có dạng $e = (u, u)$, trong đó u là đỉnh nào đó thuộc V .

Đơn đồ thị có hướng. Đồ thị $G = \langle V, E \rangle$ bao gồm V là tập các đỉnh, E là tập các cặp có thứ tự gồm hai phần tử của V gọi là các cung. Giữa hai đỉnh bất kỳ của G tồn tại nhiều nhất một cung.

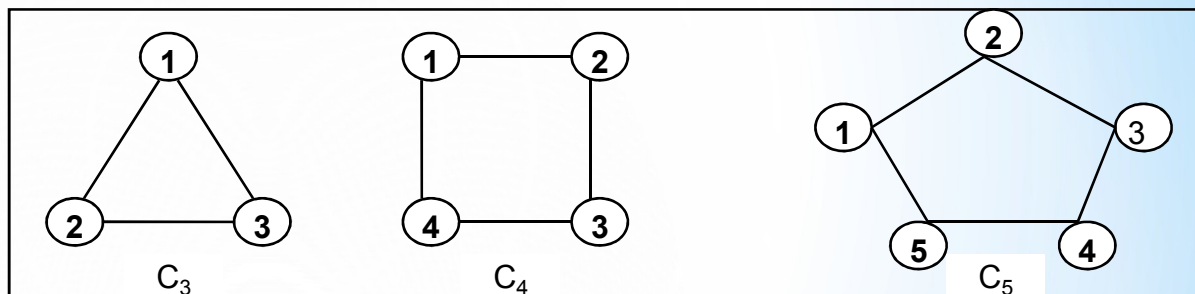
Đa đồ thị có hướng. Đồ thị $G = \langle V, E \rangle$ bao gồm V là tập đỉnh, E là cặp có thứ tự gồm hai phần tử của V được gọi là các cung. Hai cung e_1, e_2 tương ứng với cùng một cặp đỉnh được gọi là cung lặp.

Một số dạng đồ thị đặc biệt:

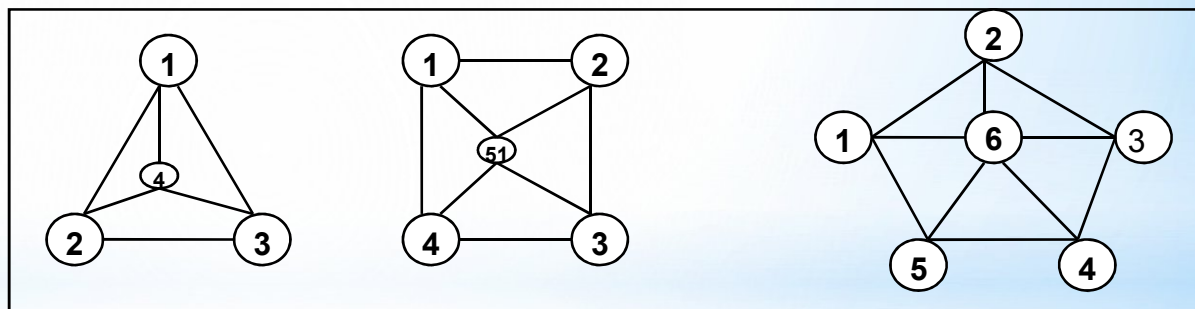
Đồ thị đầy đủ



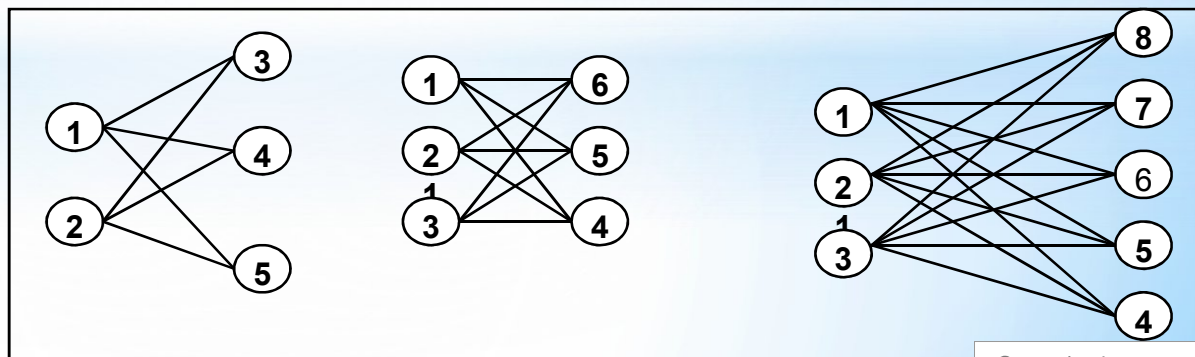
Đồ thị vòng



Đồ thị bánh xe



Đồ thị hai phía



Một số thuật ngữ trên đồ thị vô hướng:

Đỉnh kề. Hai đỉnh u và v của đồ thị vô hướng $G = \langle V, E \rangle$ được gọi là kề nhau nếu (u, v) là cạnh thuộc đồ thị G . Nếu $e = (u, v)$ là cạnh của đồ thị G thì ta nói cạnh này liên thuộc với hai đỉnh u và v , hoặc ta nói cạnh e nối đỉnh u với đỉnh v , đồng thời các đỉnh u và v sẽ được gọi là đỉnh đầu của cạnh (u, v) .

Bậc của đỉnh. Ta gọi bậc của đỉnh v trong đồ thị vô hướng là số cạnh liên thuộc với nó và ký hiệu là $\deg(v)$. Đỉnh có bậc là 0 được gọi là đỉnh cô lập. Đỉnh có bậc 1 được gọi là đỉnh treo.

Đường đi, chu trình. Đường đi độ dài n từ đỉnh u đến đỉnh v trên đồ thị vô hướng $G = \langle V, E \rangle$ là dãy $x_0, x_1, \dots, x_{n-1}, x_n$, trong đó n là số nguyên dương, $x_0 = u$, $x_n = v$, $(x_i, x_{i+1}) \in E$, $i = 0, 1, 2, \dots, n-1$. Đường đi có đỉnh đầu trùng với đỉnh cuối gọi là chu trình.

Tính liên thông. Đồ thị vô hướng được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kỳ của nó.

Thành phần liên thông. Đồ thị vô hướng liên thông thì số thành phần liên thông là 1. Đồ thị vô hướng không liên thông thì số liên thông của đồ thị là số các đồ thị con của nó liên thông.

Đỉnh trụ. Đỉnh $u \in V$ được gọi là đỉnh trụ nếu loại bỏ u cùng với các cạnh nối với u làm tăng thành phần liên thông của đồ thị.

Cạnh cầu. Cạnh $(u, v) \in E$ được gọi là cầu nếu loại bỏ (u, v) làm tăng thành phần liên thông của đồ thị.

Đỉnh rẽ nhánh. Đỉnh s được gọi là đỉnh rẽ nhánh (đỉnh thắt) của cặp đỉnh u, v nếu mọi đường đi từ u đến v đều qua s .

Một số thuật ngữ trên đồ thị có hướng:

Định nghĩa 1. Nếu $e=(u,v)$ là cung của đồ thị có hướng G thì ta nói hai đỉnh u và v là kề nhau, và nói cung (u, v) nối đỉnh u với đỉnh v , hoặc nói cung này đi ra khỏi đỉnh u và đi vào đỉnh v . Đỉnh u được gọi là đỉnh đầu, đỉnh v được gọi là đỉnh cuối của cung (u,v) .

Định nghĩa 2. Ta gọi bán bậc ra của đỉnh v trên đồ thị có hướng là số cung của đồ thị đi ra khỏi v và ký hiệu là $\deg^+(v)$. Ta gọi bán bậc vào của đỉnh v trên đồ thị có hướng là số cung của đồ thị đi vào v và ký hiệu là $\deg^-(v)$.

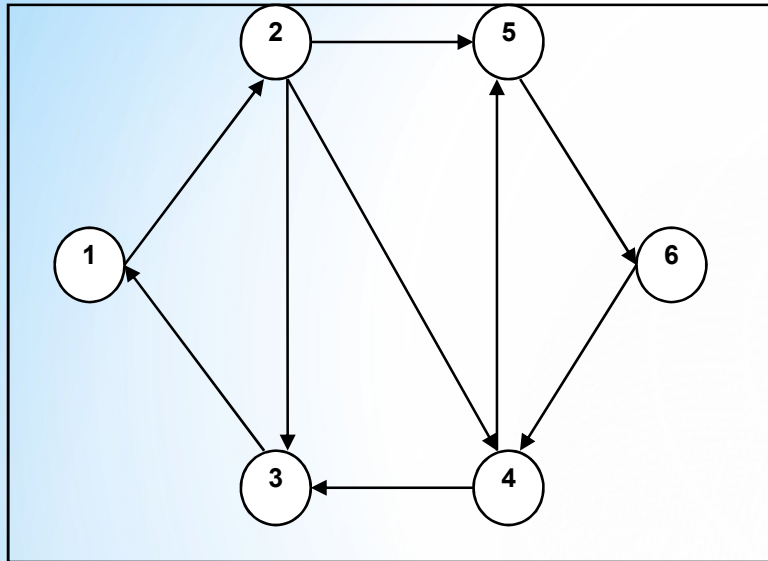
Định nghĩa 3. Đường đi độ dài n từ đỉnh u đến đỉnh v trong đồ thị có hướng $G=<V,A>$ là dãy x_0, x_1, \dots, x_n , trong đó, n là số nguyên dương, $u = x_0$, $v = x_n$, $(x_i, x_{i+1}) \in E$. Đường đi như trên có thể biểu diễn thành dãy các cung : (x_0, x_1) , (x_1, x_2) , \dots , (x_{n-1}, x_n) . Đỉnh u được gọi là đỉnh đầu, đỉnh v được gọi là đỉnh cuối của đường đi. Đường đi có đỉnh đầu trùng với đỉnh cuối ($u=v$) được gọi là một chu trình. Đường đi hay chu trình được gọi là đơn nếu như không có hai cạnh nào lặp lại.

Định nghĩa 4. Đồ thị có hướng $G=<V,E>$ được gọi là liên thông mạnh nếu giữa hai đỉnh bất kỳ $u \in V$, $v \in V$ đều có đường đi từ u đến v .

Định nghĩa 5. Ta gọi đồ thị vô hướng tương ứng với đồ thị có hướng $G=<V,E>$ là đồ thị tạo bởi G và bỏ hướng của các cạnh trong G . Khi đó, đồ thị có hướng $G=<V,E>$ được gọi là liên thông yếu nếu đồ thị vô hướng tương ứng với nó là liên thông.

5.2. Biểu diễn đồ thị

Biểu diễn bằng ma trận kề:



0	1	0	0	0	0
0	0	1	1	1	0
1	0	0	0	0	0
0	0	1	0	1	0
0	0	0	0	0	1
0	0	0	1	0	0

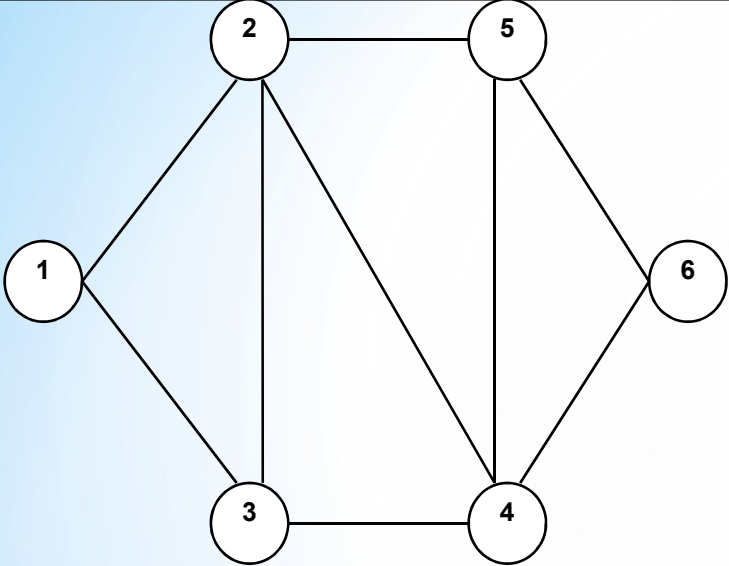
Ưu điểm của ma trận kề:

- Đơn giản dễ cài đặt trên máy tính bằng cách sử dụng một mảng hai chiều để biểu diễn ma trận kề;
- Dễ dàng kiểm tra được hai đỉnh u, v có kề với nhau hay không bằng đúng một phép so sánh ($a[u][v] \neq 0$?); và chúng ta chỉ mất đúng một phép so sánh.

Nhược điểm của ma trận kề:

- Lãng phí bộ nhớ: bất kể số cạnh nhiều hay ít ta cần n^2 đơn vị bộ nhớ để biểu diễn;
- Không thể biểu diễn được với các đồ thị có số đỉnh lớn (ví dụ triệu đỉnh);
- Để xem xét đỉnh u có những đỉnh kề nào cần mất n phép so sánh kể cả đỉnh u là đỉnh cô lập hoặc đỉnh treo.

Biểu diễn bằng danh sách cạnh: Sử dụng mảng, hoặc danh sách liên kết

	<u>Đỉnh đầu</u>	<u>Đỉnh cuối</u>
	1	2
	1	3
	2	3
	2	4
	2	5
	3	4
	4	5
	4	6
	5	6

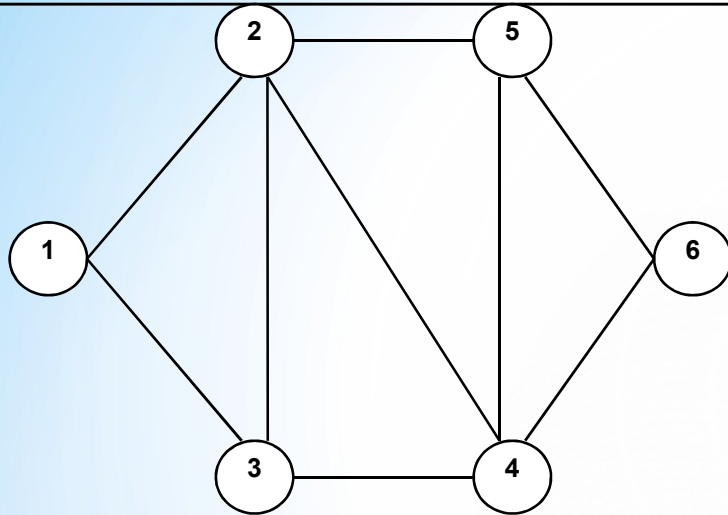
Ưu điểm của danh sách cạnh:

- Trong trường hợp đồ thị thưa ($m < 6n$), biểu diễn bằng danh sách cạnh tiết kiệm được không gian nhớ;
- Thuận lợi cho một số thuật toán chỉ quan tâm đến các cạnh của đồ thị.

Nhược điểm của danh sách cạnh:

- Khi cần duyệt các đỉnh kề với đỉnh u bắt buộc phải duyệt tất cả các cạnh của đồ thị. Điều này làm cho thuật toán có chi phí tính toán cao.

Biểu diễn bằng danh sách kề: Sử dụng mảng, hoặc danh sách liên kết



List(1) = {2, 3 }

List(2) = {1, 3, 4, 5 }

List(3) = {1, 2, 4 }

List(4) = {2, 3, 5, 6 }

List(5) = {2, 4, 6 }

List(6) = {4, 5 }

Ưu điểm của danh sách kề:

- Dễ dàng duyệt tất cả các đỉnh của một danh sách kề;
- Dễ dàng duyệt các cạnh của đồ thị trong mỗi danh sách kề;
- Tối ưu việc cài đặt các giải thuật trên đồ thị.

Nhược điểm của danh sách kề:

- Khó khăn cho người đọc có kỹ năng lập trình yếu. Vì khi biểu diễn đồ thị ta phải dùng một mảng, mỗi phần tử của nó là một danh sách liên kết (tham khảo thêm trong cài đặt cụ thể).

Một số qui ước trong cài đặt: Để thuận lợi trong cài đặt thuật toán ta qui ước các dạng biểu diễn đồ thị được tổ chức trong file theo khuôn dạng sau:

Biểu diễn ma trận kề:

- Dòng đầu tiên ghi lại số tự nhiên N là số đỉnh của đồ thị;
- N dòng kế tiếp ghi lại ma trận kề của đồ thị. Hai phần tử khác nhau của ma trận kề được ghi cách nhau một vài khoảng trống.

Biểu diễn bằng danh sách cạnh:

- Dòng đầu tiên ghi lại hai số tự nhiên N và M tương ứng với là số đỉnh và số cạnh của đồ thị. Hai số được viết cách nhau một vài khoảng trống.
- M dòng kế tiếp, mỗi dòng ghi lại một cạnh của đồ thị. Đỉnh đầu và đỉnh cuối của mỗi cạnh được viết cách nhau một vài khoảng trống.

Biểu diễn đồ thị bằng danh sách kề:

- Dòng đầu tiên ghi lại số đỉnh của đồ thị.
- Những dòng kế tiếp ghi lại trên mỗi dòng danh sách kề của đỉnh tương ứng. Danh sách kề của mỗi đỉnh được ghi theo khuôn dạng: <số lượng đỉnh kề của đỉnh tương ứng> + <danh sách đỉnh kề của đỉnh>. Các đỉnh (số) được viết cách nhau một vài khoảng trống.

Ví dụ: với dòng $u = 5$ ghi theo khuôn dạng

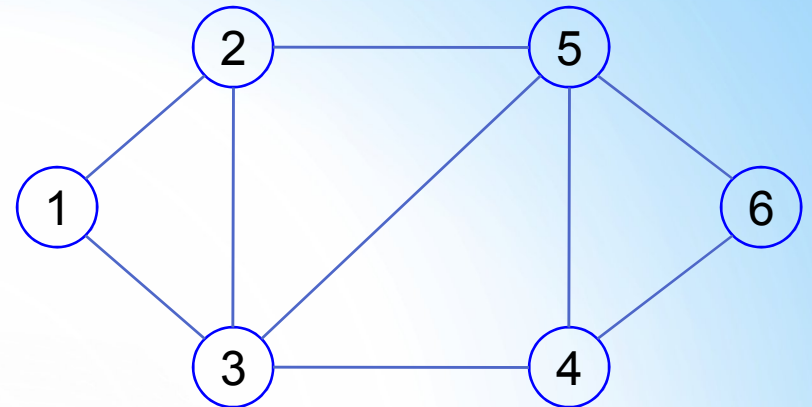
3 2 4 6

Ta hiểu là danh sách kề của đỉnh 5 gồm 3 đỉnh là đỉnh 2, 4, 6.

Ví dụ về các file biểu diễn đồ thị:

dothi.in

6					
0	1	1	0	0	0
1	0	1	0	1	0
1	1	0	0	1	0
0	1	1	0	1	1
0	1	1	1	0	1
0	0	0	1	1	0



dothi.in

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

dothi.in

6	8
1	2
1	3
2	3
2	5
3	4
3	5
4	5
4	6
5	6

BÀI TẬP 1. Chuyển đổi biểu diễn của đồ thị (vô hướng hoặc có hướng) thành các dạng khác nhau. Dữ liệu vào là các file có khuôn dạng như đã đề cập ở trên. Chương trình gồm có các chức năng chính như sau:

1. Chuyển đổi biểu diễn đồ thị từ dạng biểu diễn ma trận kề trong file dothi.in thành biểu diễn đồ thị dưới dạng danh sách cạnh trong file canh.out.
2. Chuyển đổi biểu diễn đồ thị từ dạng biểu diễn ma trận kề trong file dothi.in thành biểu diễn đồ thị dưới dạng danh sách kề trong file ke.out.
3. Chuyển đổi biểu diễn đồ thị từ dạng biểu diễn danh sách cạnh trong file dothi.in thành biểu diễn đồ thị dưới dạng ma trận kề trong file matranke.out.
4. Chuyển đổi biểu diễn đồ thị từ dạng biểu diễn danh sách cạnh trong file dothi.in thành biểu diễn đồ thị dưới dạng danh sách kề trong file ke.out.
5. Chuyển đổi biểu diễn đồ thị từ dạng biểu diễn danh sách kề trong file dothi.in thành biểu diễn đồ thị dưới dạng ma trận kề trong file matranke.out.
6. Chuyển đổi biểu diễn đồ thị từ dạng biểu diễn danh sách kề trong file dothi.in thành biểu diễn đồ thị dưới dạng danh sách cạnh trong file canh.out.

Chú ý sự khác biệt trong khi chuyển đổi giữa đồ thị vô hướng và đồ thị có hướng. Trong Những nội dung tiếp theo, ta sẽ làm việc với dữ liệu biểu diễn đồ thị trên máy tính và hạn chế sử dụng hình vẽ của đồ thị.

5.3. Thuật toán tìm kiếm theo chiều sâu

Đặt bài toán. Cho đồ thị vô hướng hoặc có hướng $G = \langle V, E \rangle$. Bài toán cơ bản đầu tiên của lý thuyết đồ thị là làm thế nào ta có thể duyệt (thăm) các đỉnh của đồ thị bắt đầu tại một đỉnh u tùy ý thuộc V .

Thuật toán tìm kiếm theo chiều sâu bắt đầu tại đỉnh $u \in V$ trên đồ thị được tiến hành như sau:

- Ghi nhận trạng thái các đỉnh lúc đầu đều chưa được duyệt bằng một mảng chuaxet[]: chuaxet[u] = True ứng với trạng thái đỉnh u chưa được duyệt, chuaxet[u] = False ứng với trạng thái đỉnh u đã được duyệt. Như vậy, trạng thái ban đầu của chuaxet[u] = True với mọi $u \in V$.
- Bắt đầu tại đỉnh tùy ý $u \in V$ ta thăm luôn đỉnh u. Sau đó ghi nhận đỉnh u đã được thăm bằng cách bật trạng thái chuaxet[u]=False.
- Duyệt trên tập đỉnh $Ke(u) = \{ v \in V : (u, v) \in E \}$ và duyệt theo chiều sâu tại đỉnh $v \in Ke(u)$ đầu tiên chưa được duyệt hay chuaxet[v] = true để tiếp tục duyệt.

5.3.1. Thuật toán đệ qui DFS(u)

```
void DFS ( u ) {  
    <thăm đỉnh u>; chuaxet[u] = False;  
    for (v=1; v<=n; v++) //V = {1, 2, ..., n}  
        if ( v ∈ Ke(u) and chuaxet[v])  
            DFS(v);  
}
```


5.3.2. Thuật toán DFS(u) dựa trên stack

Thuật toán DFS(u):

Begin

Bước 1 (Khởi tạo):

stack = \emptyset ; *// Khởi tạo stack là \emptyset*

Push(stack, u); *// Đưa đỉnh u vào ngăn xếp*

<Thăm đỉnh u>; *// Duyệt đỉnh u*

chuaxet[u] = False; *// Xác nhận đỉnh u đã duyệt*

Bước 2 (Lặp) :

while (stack $\neq \emptyset$) do

 s = Pop(stack); *// Loại đỉnh ở đầu ngăn xếp*

 for each t \in Ke(s) do *// Lấy mỗi đỉnh t \in Ke(s)*

 if (chuaxet[t]) then *// Nếu t đúng là chưa duyệt*

 <Thăm đỉnh t>; *// Duyệt đỉnh t*

 chuaxet[t] = False; *// Xác nhận đỉnh t đã duyệt*

 Push(stack, s); *// Đưa s vào stack*

 Push(stack, t); *// Đưa t vào stack*

 break; *// Chỉ lấy một đỉnh t*

 EndIf;

 EndFor;

EndWhile;

Bước 3 (Trả lại kết quả): Return(<Tập đỉnh đã duyệt>);

End.

Kiểm nghiệm thuật toán:

I

Ví dụ 2. Cho đồ thị gồm 13 đỉnh được biểu diễn dưới dạng ma trận kề như hình bên phải. Hãy cho biết kết quả thực hiện thuật toán trong Hình 3.1 bắt đầu tại đỉnh $u=1$? Chỉ rõ trạng thái của ngăn xếp và tập đỉnh được duyệt theo mỗi bước thực hiện của thuật toán?

0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	1	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	1	1	1	0	0	0	1	0
0	1	0	0	1	0	1	0	0	0	0	1	0
0	0	0	1	1	1	0	1	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	1	0	1	1	1
0	0	0	0	0	0	0	0	1	1	0	0	1
0	0	0	0	1	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	0

DFS(1) = 1, 2, 3, 4, 7, 5, 6, 12, 8, 10, 9, 11, 13

STT	Trạng thái stack	Các đỉnh được duyệt
1	1	1
2	1, 2	1, 2
3	1, 2, 3	1, 2, 3
4	1, 2, 3, 4	1, 2, 3, 4
5	1, 2, 3, 4, 7	1, 2, 3, 4, 7
6	1, 2, 3, 4, 7, 5	1, 2, 3, 4, 7, 5
7	1, 2, 3, 4, 7, 5, 6	1, 2, 3, 4, 7, 5, 6
8	1, 2, 3, 4, 7, 5, 6, 12	1, 2, 3, 4, 7, 5, 6, 12
9	1, 2, 3, 4, 7, 5, 6, 12, 8	1, 2, 3, 4, 7, 5, 6, 12, 8
10	1, 2, 3, 4, 7, 5, 6, 12, 10	1, 2, 3, 4, 7, 5, 6, 12, 8, 10
11	1, 2, 3, 4, 7, 5, 6, 12, 10, 9	1, 2, 3, 4, 7, 5, 6, 12, 8, 10, 9
12	1, 2, 3, 4, 7, 5, 6, 12, 10, 9, 11	1, 2, 3, 4, 7, 5, 6, 12, 8, 10, 9, 11
13	1, 2, 3, 4, 7, 5, 6, 12, 10, 9, 11, 13	1, 2, 3, 4, 7, 5, 6, 12, 8, 10, 9, 11, 13
14	∅	

5.4. Thuật toán tìm kiếm theo chiều rộng (BFS)

Thuật toán BFS(u):

Bước 1(Khởi tạo):

Queue = \emptyset ; // Khởi tạo hàng đợi rỗng
Push(Queue,u); //Đưa u vào hàng đợi
chuaxet[u] = False; //Ghi nhận đỉnh đã xét

Bước 2 (Lặp):

while (Queue $\neq \emptyset$) do //Lặp đến khi hàng đợi rỗng
 s = Pop(Queue); //Lấy s ra khỏi hàng đợi
 <Thăm đỉnh s>; //Thăm đỉnh s
 for each t \in Ke(s) do //Lặp trên danh sách kề của s
 if (chuaxet[t]) then //Nếu t chưa xét
 Push(Queue, t); //Đưa t vào hàng đợi
 chuaxet[t] = False; //Ghi nhận t đã xét
 EndIf ;
EndFor ;
EndWhile ;

Bước 3 (Trả lại kết quả) :

Return(<Tập đỉnh được duyệt>) ;

End.

Kiểm nghiệm thuật toán BFS(u):

Ví dụ 3. Cho đồ thị gồm 13 đỉnh được biểu diễn dưới dạng ma trận kề như hình bên phải. Hãy cho biết kết quả thực hiện thuật toán trong Hình 3.3 bắt đầu tại đỉnh $u=1$? Chỉ rõ trạng thái của hàng đợi và tập đỉnh được duyệt theo mỗi bước thực hiện của thuật toán?

0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	1	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	1	1	1	0	0	0	1	0
0	1	0	0	1	0	1	0	0	0	0	1	0
0	0	0	1	1	1	0	1	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	1	0	1	1	1
0	0	0	0	0	0	0	0	1	1	0	0	1
0	0	0	0	1	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	1	1	1	0	0



STT	Trạng thái Queue	Các đỉnh được duyệt
1	1	\emptyset
2	2, 3, 4	1
3	3, 4, 6	1, 2
4	4, 6, 5	1, 2, 3
5	6, 5, 7	1, 2, 3, 4
6	5, 7, 12	1, 2, 3, 4, 6
7	7, 12, 8	1, 2, 3, 4, 6, 5
8	12, 8	1, 2, 3, 4, 6, 5, 7
9	8, 10	1, 2, 3, 4, 6, 5, 7, 12
10	10	1, 2, 3, 4, 6, 5, 7, 12, 8
11	9, 11, 13	1, 2, 3, 4, 6, 5, 7, 12, 8, 10
12	11, 13	1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9
13	13	1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9, 11
14	\emptyset	1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9, 11, 13

Kết quả duyệt BFS(1) = { 1, 2, 3, 4, 6, 5, 7, 12, 8, 10, 9, 11, 13 }.

Độ phức tạp thuật toán DFS, BFS:

Độ phức tạp thuật toán DFS(u), BFS(u) phụ thuộc vào phương pháp biểu diễn đồ thị. Độ phức tạp thuật toán BFS(u) theo các dạng biểu diễn đồ thị như sau:

- Độ phức tạp thuật toán là $O(n^2)$ trong trường hợp đồ thị biểu diễn dưới dạng ma trận kề, với n là số đỉnh của đồ thị.
- Độ phức tạp thuật toán là $O(n.m)$ trong trường hợp đồ thị biểu diễn dưới dạng danh sách cạnh, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.
- Độ phức tạp thuật toán là $O(\max(n, m))$ trong trường hợp đồ thị biểu diễn dưới dạng danh sách kề, với n là số đỉnh của đồ thị, m là số cạnh của đồ thị.

5.5. Ứng dụng của tìm kiếm

- Duyệt tất cả các đỉnh của đồ thị.
- Tìm đường đi từ s đến t .
- Xác định các thành phần liên thông của đồ thị.
- Duyệt các đỉnh trụ của đồ thị.
- Duyệt các cạnh cầu của đồ thị.
- Duyệt tập đỉnh thất của cặp đỉnh
- Xây dựng cây khung của đồ thị.
- Xác định tính liên thông mạnh.
- Xác định các thành phần liên thông mạnh.
- Định chiều đồ thị.

Thuật toán duyệt tất cả các đỉnh của đồ thị:

Thuật toán Display-Graph:

begin

Bước 1 (Khởi tạo):

for ($u=1$; $u \leq n$; $u++$) do { *//thiết lập tất cả các đỉnh đều chưa duyệt*
chuaxet[u] = True;

Endfor;

Bước 2 (Lặp):

for ($u = 1$; $u \leq n$; $u++$) do { *//lặp trên tập đỉnh V*
if (chuaxet[u]) then
BFS (u); *//DFS(u); //*

endif;

endfor;

end.

Thuật toán duyệt tất cả các thành phần liên thông của đồ thị:

Thuật toán Conneted-Graph:

begin

Bước 1 (Khởi tạo):

Solt =0; *//thiết lập số thành phần liên thông ban đầu là 0*

for (u=1; u<=n; u++) do { *//thiết lập tất cả các đỉnh đều chưa duyệt*
chuaxet[u] = True;

Endfor;

Bước 2 (Lặp):

for (u =1; u≤ n; u++) do { *//lặp trên tập đỉnh V*

if (chuaxet[u]) then

Solt++; *//tăng thành phần liên thông*

<ghi nhận các đỉnh cùng một thành phần liên thông>;

BFS (u); *//DFS(u);*

endif;

endfor;

Bước 3 (trả lại kết quả):

Return(Solt);

end.

Ví dụ ta cần kiểm nghiệm thuật toán trên Hình 3.4 cho đồ thị được biểu diễn dưới dạng ma trận kề như dưới đây.

0	0	1	0	1	0	1	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	0	0
1	0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	0	0	1	0	1	0	0	0
1	0	1	0	1	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0	1	0	1	0
0	0	0	0	1	0	1	0	0	0	1	0	1
0	0	0	1	0	1	0	1	0	0	0	1	0
0	0	1	0	1	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	1	0	1	0	0

Thực hiện thuật toán DFS và BFS như được mô tả ở trên ta nhận được :

Thành phần liên thông 1: $BFS(1) = \{1, 3, 5, 7, 9, 11, 13\}$.

Thành phần liên thông 2: $BFS(2) = \{2, 4, 6, 8, 10, 12\}$.

Thuật toán tìm đường đi từ đỉnh s đến đỉnh t:

Thuật toán Length-DFS(s, t):

Begin

Bước 1 (Khởi tạo):

stack = \emptyset ; *//Khởi tạo stack là \emptyset*

Push(stack, s); *//Đưa đỉnh s vào ngăn xếp*

chuaxet[s] = False; *//Xác nhận đỉnh u đã duyệt*

Bước 2 (Lặp) :

while (stack $\neq \emptyset$) do

 u = Pop(stack); *//Loại đỉnh ở đầu ngăn xếp*

 for each v \in Ke(u) do *//Lấy mỗi đỉnh u \in Ke(v)*

 if (chuaxet[v]) then *//Nếu v đúng là chưa duyệt*

 chuaxet[v] = False; *// Xác nhận đỉnh v đã duyệt*

 Push(stack, u); *//Đưa u vào stack*

 Push(stack, v); *//Đưa v vào stack*

truoc[v] = u; *//Ghi nhận truoc[v] là u*

 break; *//Chỉ lấy một đỉnh t*

 EndIf;

 EndFor;

EndWhile;

Bước 3 (Trả lại kết quả):

Return(truoc[]); *//trả lại mảng ghi lại các đỉnh DFS đã duyệt*

End.

Thuật toán tìm đường đi từ đỉnh s đến đỉnh t:

Thuật toán Length-BFS(s, t):

Bước 1(Khởi tạo):

Queue = \emptyset ; *//Khởi tạo hàng đợi là rỗng*
Push(Queue,s); *//Đưa s vào hàng đợi*
chuaxet[s] = False; *//Bật trạng thái đỉnh s*

Bước 2 (Lặp):

while (Queue $\neq \emptyset$) do { *//lặp đến khi hàng đợi rỗng*
 u = Pop(Queue); *//lấy u ra khỏi hàng đợi*
 for each $v \in Ke(u)$ do { *//duyet trên các đỉnh v kề với u*
 if (chuaxet[v]) { *//nếu đỉnh v đúng là chưa được xét*
 Push(Queue, v); *//đưa v vào hàng đợi*
 chuaxet[v]=False; *//ghi nhận v đã xét*
 truoc[v]=u; *//Ghi nhận muốn đi đến v phải qua u*
 }
 EndIf ;
 EndFor ;
EndWhile ;

Bước 3 (Trả lại kết quả) :

Return(truoc[>]) ; *//trả lại mảng ghi lại các đỉnh BFS đã duyệt*

End.

Thuật toán ghi nhận đường đi từ đỉnh s đến đỉnh t:

Thuật toán Ghi-Nhan-Duong-Di (s, t) {

if (truoc[t] == 0) {

 <Không có đường đi từ s đến t>;

}

else {

 <Đưa ra đỉnh t>; //Đưa ra trước đỉnh t

 u = truoc[t]; //u là đỉnh trước khi đến được t

while (u ≠ s) { //Lặp nếu u chưa phải là s

 <Đưa ra đỉnh u>; //Đưa ra đỉnh u

 u = truoc[u]; // Lần ngược lại đỉnh truoc[u].

}

 <Đưa ra nốt đỉnh s>;

}

}

Thuật toán kiểm tra tính liên thông mạnh trên đồ thị có hướng:

Boolean Strong-Connective ($G = \langle V, E \rangle$) {

```
Relnit(); //  $\forall u \in V$ : chuaxet[u] = True;  
for each  $u \in V$  do { // Lấy mỗi đỉnh thuộc  $V$   
    if (DFS( $u$ )  $\neq V$  ) then // Nếu DFS( $u$ )  $\neq V$  hoặc BFS( $u$ )  $\neq V$   
        return(False); // Đồ thị không liên thông mạnh  
    endif;  
    Relnit(); // Khởi tạo lại các mảng chuaxet[]  
endfor;  
return(True); // Đồ thị liên thông mạnh
```

}

Thuật toán duyệt đỉnh trụ:

Thuật toán Duyệt-Tru ($G = \langle V, E \rangle$) {

```
Relnit(); //  $\forall u \in V$ : chuaxet[u] = True;  
for each  $v \in V$  do { // Lấy mỗi đỉnh  $u$  tập đỉnh  $V$   
    chuaxet[v] = False; // Cấm DFS hoặc BFS duyệt vào đỉnh  $v$   
    if (DFS( $u$ )  $\neq V \setminus \{v\}$  ) then // Duyệt DFS hoặc BFS tại đỉnh  $u \neq v$   
        <Ghi nhận  $v$  là trụ>;  
    endif ;  
    Relnit(); // Khởi tạo lại các mảng chuaxet[]  
endfor;
```

}

Bài toán duyệt các thành phần liên thông mạnh của đồ thị

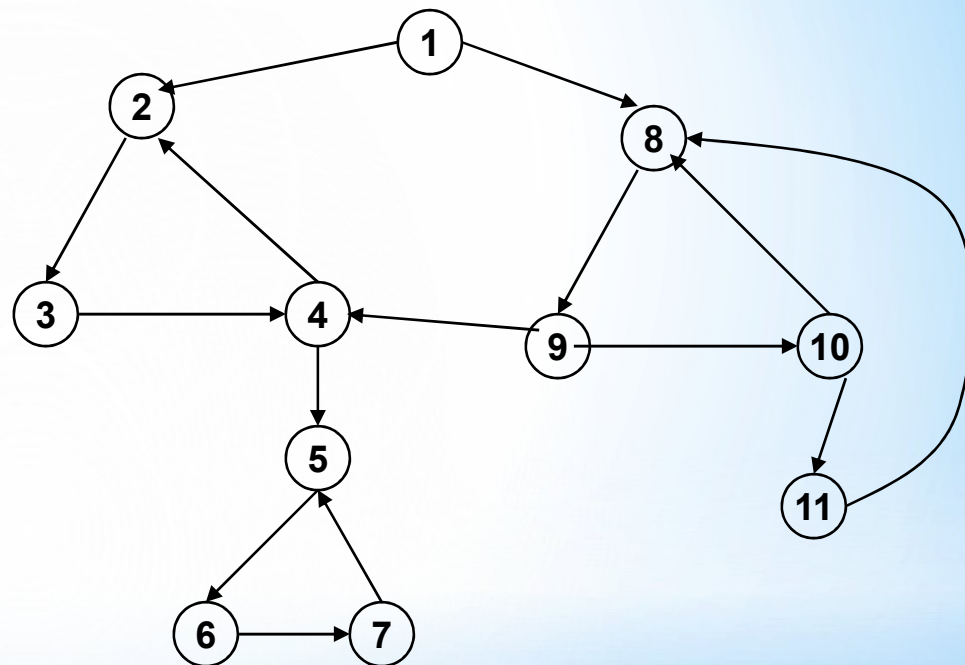
Đặt bài toán. Cho đồ thị có hướng $G = \langle V, E \rangle$. Hãy liệt kê tất cả các thành phần liên thông mạnh của đồ thị.

Input:

11	15
1	2
1	8
2	3
3	4
4	2
4	5
5	6
6	7
7	5
8	9
9	4
9	10
10	8
10	11
11	8

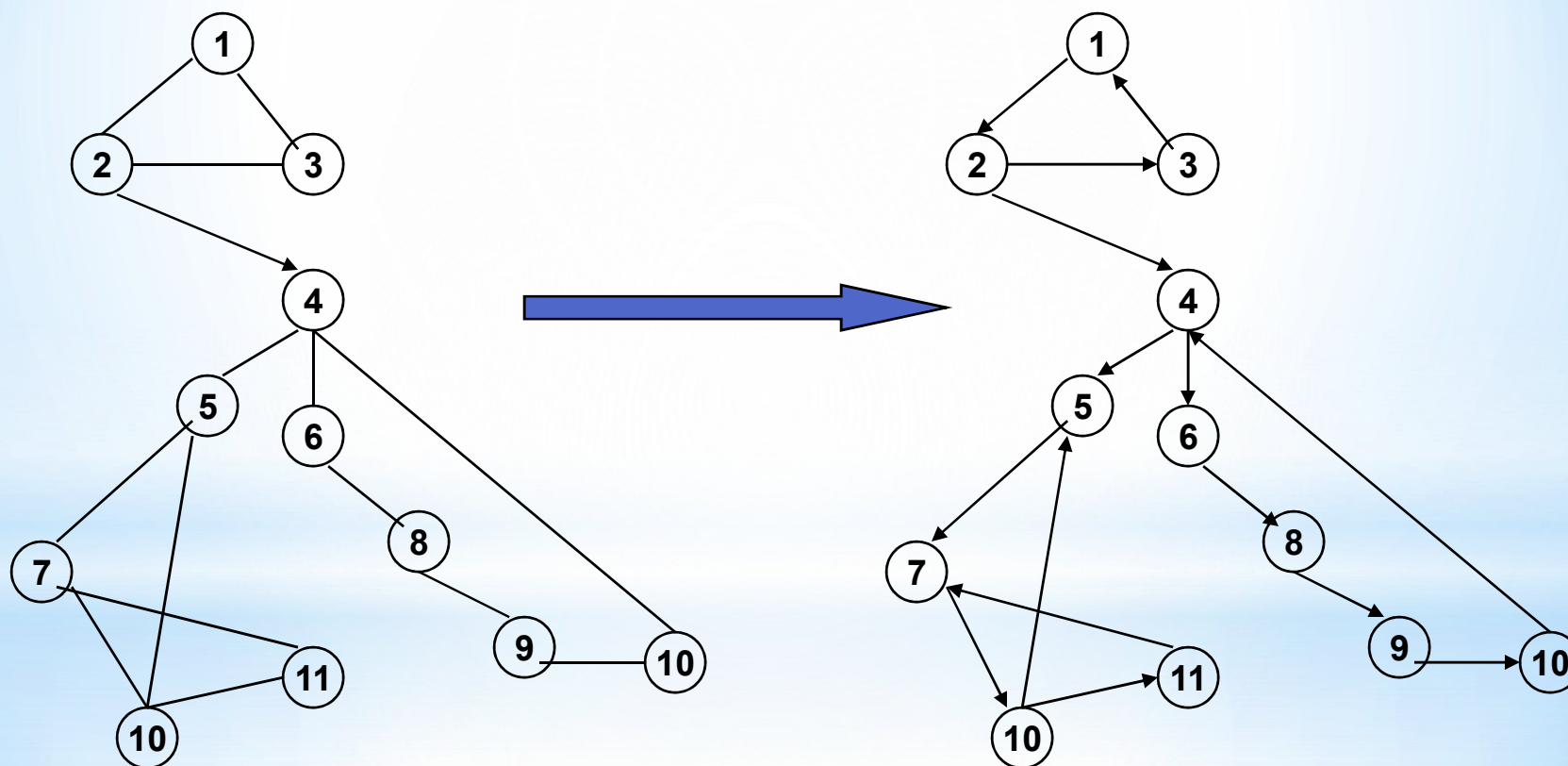
Output:

TPLTM1: 4 3 2
TPLTM2: 7 6 5
TPLTM3: 11 10 9 8
TPLTM4: 1



Định chiều đồ thị và bài toán liệt kê cầu

Bài toán . Cho đồ thị vô hướng liên thông $G = \langle V, E \rangle$. Hãy định hướng lại các cạnh của G sao cho ta có thể nhận được đồ thị có hướng với tối thiểu các thành phần liên thông mạnh.

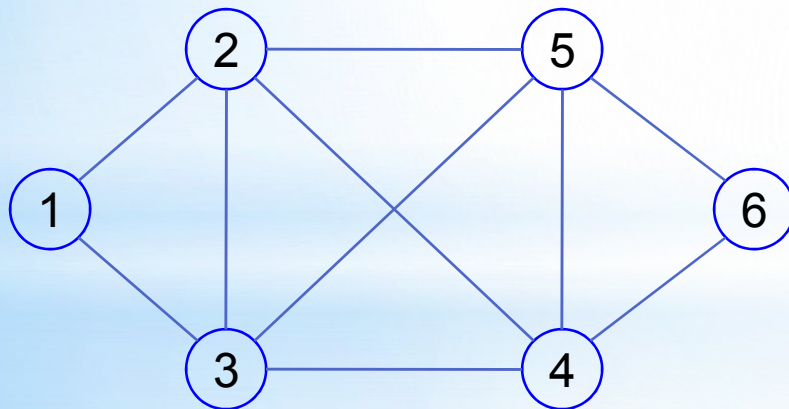


5.5. Đồ thị Euler- Đồ thị nửa Euler

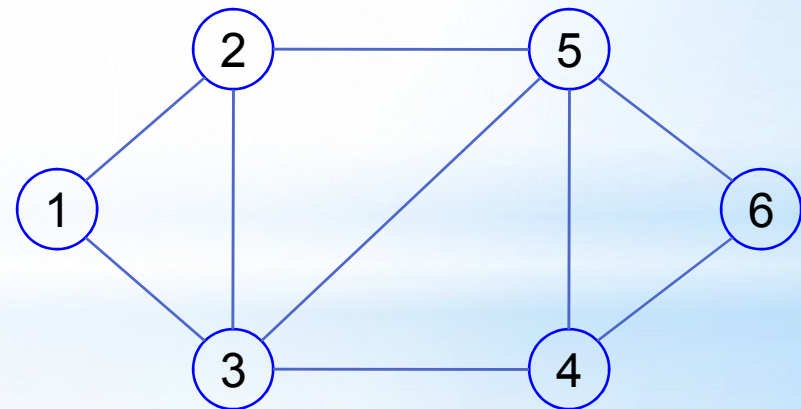
Định nghĩa. Chu trình đơn trong đồ thị G đi qua mỗi cạnh của đồ thị đúng một lần được gọi là chu trình Euler. Đường đi đơn trong G đi qua mỗi cạnh của nó đúng một lần được gọi là đường đi Euler. Đồ thị được gọi là đồ thị Euler nếu nó có chu trình Euler. Đồ thị có đường đi Euler được gọi là nửa Euler. Thuật toán xây dựng chu trình Euler trên đồ thị có hướng Euler bắt đầu tại đỉnh u .

Ví dụ :

- Đồ thị $G1$ là Euler vì $G1$ có đường đi Euler: 1-2-3-4-5-2-4-6-5-3-1
- Đồ thị $G2$ là nửa Euler vì $G1$ có đường đi Euler: 2-1-3-2-5-3-4-6-5-4.



Đồ thị $G1$



Đồ thị $G2$

5.5.1. Điều kiện cần và đủ để đồ thị là Euler và Semmi-Euler

Định lý 1 (Đồ thị vô hướng):

- Đồ thị vô hướng liên thông $G = \langle V, E \rangle$ là Euler khi và chỉ khi tất cả các đỉnh của V đều có bậc chẵn.
- Đồ thị vô hướng liên thông $G = \langle V, E \rangle$ là nửa Euler nhưng không là Euler khi và chỉ khi G có đúng hai đỉnh bậc lẻ. Đường đi Euler xuất phát tại một đỉnh bậc lẻ và kết thúc tại đỉnh bậc lẻ còn lại.

Định lý 2 (Đồ thị có hướng):

- Đồ thị có hướng liên thông yếu $G = \langle V, E \rangle$ là Euler khi và chỉ khi mọi đỉnh của G đều có bán đỉnh bậc ra bằng bán đỉnh bậc vào.
- Đồ thị có hướng liên thông yếu $G = \langle V, E \rangle$ là nửa Euler nhưng không là Euler khi và chỉ khi tồn tại đúng hai đỉnh u, v sao cho bán đỉnh bậc ra của u trừ bán đỉnh bậc vào của u bằng bán đỉnh bậc vào của v trừ bán đỉnh bậc ra của v và bằng 1. Các đỉnh khác u, v còn lại có bán đỉnh bậc ra bằng bán đỉnh bậc vào. Đường đi Euler trên đồ thị sẽ xuất phát tại u và kết thúc tại v .

5.5.2. Thuật toán kiểm tra đồ thị là Euler và Semmi-Euler

Thuật toán kiểm tra đồ thị vô hướng:

```
Boolean Check-Euler( A[], n ) { //A là ma trận kề, n là số đỉnh của đồ thị
    int H; //H là tổng theo hàng;
    int dem1=0, dem2=0; //dem1 là số đỉnh bậc chẵn; dem2 là số đỉnh bậc lẻ
    for (int u=1; u<=n; u++){ //duyệt  $u \in V$ .
        H = 0; //thiết lập H=0;
        for (int v=1; v<=n; v++) { // duyệt  $v \in V$ .
            H = H + A[u][v]; //tổng các phần tử của hàng u.
        }
        if ( H % 2 ==0) dem1 ++; //tăng số đỉnh bậc chẵn lên 1.
        else {
            dem2 ++; // tăng số đỉnh bậc lẻ lên 1.
            x = u; //ghi nhận đỉnh bậc lẻ.
        }
    }
    if (dem1==n ) return(1); //nếu tất cả các đỉnh đều có bậc chẵn.
    else if (dem2 ==2) return (-1); //nếu chỉ có hai đỉnh bậc lẻ.
    return (0);
}
```

Kiểm nghiệm thuật toán: Giả sử ta cần kiểm tra đồ thị vô hướng liên thông G1 và G2 được biểu diễn dưới dạng ma trận kề A[][] có phải là Euler hay không.

Đồ thị vô hướng liên thông G1:

- $u = 1$: $H = \deg(1) = 2$; $dem1 = 1$; $dem2 = 0$;
- $u = 2$: $H = \deg(2) = 4$; $dem1 = 2$; $dem2 = 0$;
- $u = 3$: $H = \deg(3) = 4$; $dem1 = 3$; $dem2 = 0$;
- $u = 4$: $H = \deg(4) = 4$; $dem1 = 4$; $dem2 = 0$;
- $u = 5$: $H = \deg(5) = 4$; $dem1 = 5$; $dem2 = 0$;
- $u = 6$: $H = \deg(6) = 2$; $dem1 = 6$; $dem2 = 0$;

Vì $dem1 = 6$ hay tất cả các đỉnh của G đều có bậc chẵn nên hàm $Check-Euler(A, 6) = 1$. Nói cách khác G là Euler.

Đồ thị vô hướng liên thông G2:

- $u = 1$: $H = \deg(1) = 2$; $dem1 = 1$; $dem2 = 0$;
- $u = 2$: $H = \deg(2) = 3$; $dem1 = 1$; $dem2 = 1$; $x = 2$;
- $u = 3$: $H = \deg(3) = 4$; $dem1 = 2$; $dem2 = 1$;
- $u = 4$: $H = \deg(4) = 3$; $dem1 = 2$; $dem2 = 2$; $x = 4$;
- $u = 5$: $H = \deg(5) = 4$; $dem1 = 3$; $dem2 = 2$;
- $u = 6$: $H = \deg(6) = 2$; $dem1 = 4$; $dem2 = 2$;

Vì $dem2 = 2$ hay G có đúng hai đỉnh bậc lẻ nên hàm $Check-Euler(A, 6) = -1$. Nói cách khác G là nửa Euler và đường đi Euler xuất phát tại đỉnh $x = 4$ và kết thúc tại đỉnh 2.

Đồ thị G1

0	1	1	0	0	0
1	0	1	1	1	0
1	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	0	1
0	0	0	1	1	0

Đồ thị G2

0	1	1	0	0	0
1	0	1	0	1	0
1	1	0	1	1	0
0	0	1	0	1	1
0	1	1	1	0	1
0	0	0	1	1	0

Thuật toán kiểm tra đồ thị có hướng:

```
Boolean Check-Euler( A[], n ) { //A là ma trận kề, n là số đỉnh của đồ thị
    int H, C; //H là tổng theo hàng; C là tổng theo cột
    int dem1=0, dem2=0, dem3;
    for (int u=1; u<=n; u++){ //duyệt  $u \in V$ .
        H = 0; C = 0; //thiết lập H=0; C=0
        for (int v=1; v<=n; v++) { // duyệt  $v \in V$ .
            H = H + A[u][v]; //tổng các phần tử của hàng u.
            C = C + A[v][u]; //tổng các phần tử của hàng u.
        }
        if ( H ==C) dem1 ++; //Nếu bán đỉnh bậc ra bằng bán đỉnh bậc vào.
        else if (H-C ==1) { //nếu bán đỉnh bậc ra trừ bán đỉnh bậc vào là 1
            dem2 ++; // tăng số đỉnh bậc lẻ lên 1.
            x = u; //ghi nhận đỉnh u.
        }
        else if ((C-H ==1) { //nếu bán đỉnh bậc vào trừ bán đỉnh bậc ra là 1
            dem3 ++; // tăng số đỉnh bậc lẻ lên 1.
            y = u; //ghi nhận đỉnh v.
        }
    }
    if (dem1==n ) return(1); //nếu tất cả các đỉnh đều có  $\deg^+(u)=\deg^-(u)$ .
    else if (dem2 ==1 && dem3==1) return (-1); //nếu chỉ tồn tại u và v.
    return (0);
}
```


Kiểm nghiệm thuật toán: Giả sử ta cần kiểm tra đồ thị có hướng liên thông yếu G1 và G2 được biểu diễn dưới dạng ma trận kề A[][] có phải là Euler hay không.

Đồ thị có hướng liên thông yếu G1: $H = \deg^+(u)$; $C = \deg^-(u)$

- $u = 1$: $H = 1$; $C = 1$; $\text{dem1} = 1$; $\text{dem2} = 0$; $\text{dem3} = 0$;
- $u = 2$: $H = 2$; $C = 2$; $\text{dem1} = 2$; $\text{dem2} = 0$; $\text{dem3} = 0$;
- $u = 3$: $H = 2$; $C = 2$; $\text{dem1} = 3$; $\text{dem2} = 0$; $\text{dem3} = 0$;
- $u = 4$: $H = 2$; $C = 2$; $\text{dem1} = 4$; $\text{dem2} = 0$; $\text{dem3} = 0$;
- $u = 5$: $H = 2$; $C = 2$; $\text{dem1} = 5$; $\text{dem2} = 0$; $\text{dem3} = 0$;
- $u = 6$: $H = 1$; $C = 1$; $\text{dem1} = 6$; $\text{dem2} = 0$; $\text{dem3} = 0$;

Vì vậy hàm $\text{Check_Euler}(A, 6) = 1$. Nói cách khác G là Euler.

Đồ thị có hướng liên thông yếu G2:

- $u = 1$: $H = 1$; $C = 1$; $\text{dem1} = 1$; $\text{dem2} = 0$; $\text{dem3} = 0$;
- $u = 2$: $H = 2$; $C = 2$; $\text{dem1} = 2$; $\text{dem2} = 0$; $\text{dem3} = 0$;
- $u = 3$: $H = 2$; $C = 2$; $\text{dem1} = 3$; $\text{dem2} = 0$; $\text{dem3} = 0$;
- $u = 4$: $H = 1$; $C = 2$; $\text{dem1} = 3$; $\text{dem2} = 0$; $\text{dem3} = 1$; $y = 4$;
- $u = 5$: $H = 2$; $C = 1$; $\text{dem1} = 3$; $\text{dem2} = 1$; $\text{dem3} = 1$; $x = 5$;
- $u = 6$: $H = 1$; $C = 1$; $\text{dem1} = 4$; $\text{dem2} = 1$; $\text{dem3} = 0$;

Vì vậy hàm $\text{Check_Euler}(A, 6) = -1$. Nói cách khác G là nửa Euler.

Đồ thị G1

0	1	0	0	0	0
0	0	0	1	1	0
1	1	0	0	0	0
0	0	1	0	1	0
0	0	1	0	0	1
0	0	0	1	0	0

Đồ thị G2

0	1	0	0	0	0
0	0	0	1	1	0
1	1	0	0	0	0
0	0	1	0	0	0
0	0	1	0	0	1
0	0	0	1	0	0

5.5.3. Thuật toán tìm chu trình Euler

Thuật toán Euler-Cycle(u):

Bước 1 (Khởi tạo) :

stack = \emptyset ; //Khởi tạo một stack bắt đầu là \emptyset

CE = \emptyset ; //Khởi tạo mảng CE bắt đầu là \emptyset

Push (stack, u) ; //Đưa đỉnh u vào ngăn xếp

Bước 2 (Lặp) :

while (stack $\neq \emptyset$) do { //Lặp cho đến khi stack rỗng

 s = get(stack); //Lấy đỉnh ở đầu ngăn xếp

 if (Ke(s) $\neq \emptyset$) then { // Nếu danh sách Ke(s) chưa rỗng

 t = < Đỉnh đầu tiên trong Ke(s) >;

 Push(stack, t) ; //Đưa t vào stack;

 E = E \ (s,t); // Loại bỏ cạnh (s,t);

 }

 else { //Trường hợp Ke(s)= \emptyset

 s = Pop(stack); // Đưa s ra khỏi ngăn xếp

 s \Rightarrow CE; //Đưa s sang CE

 }

}

Bước 3 (Trả lại kết quả) :

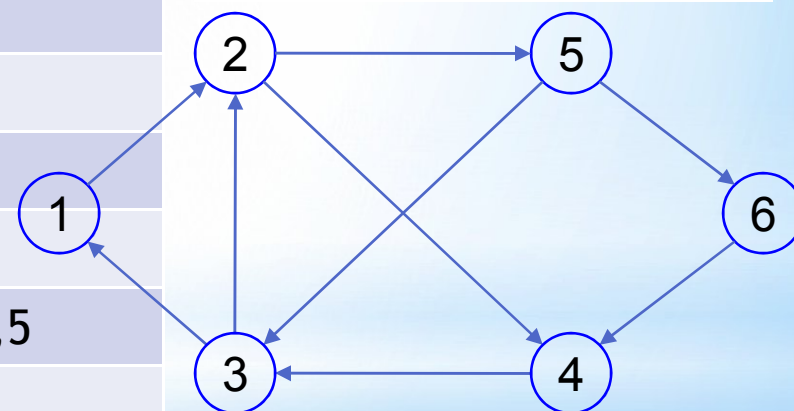
<Lật ngược lại các đỉnh trong CE ta được chu trình Euler> ;

Kiểm nghiệm thuật toán: Euler-Path(): u=5

STT	Trạng thái stack?	Giá trị CE?
1	5	ϕ
2	5,3	ϕ
3	5,3,1	ϕ
4	5,3,1,2	ϕ
5	5,3,1,2,4	ϕ
6	5,3,1,2,4,3	ϕ
7	5,3,1,2,4,3,2	ϕ
8	5,3,1,2,4,3,2,5	ϕ
9	5,3,1,2,4,3,2,5,6	ϕ
10	5,3,1,2,4,3,2,5,6,4	ϕ
11	ϕ	4,6,5,2,3,4,2,1,3,5
15	Lật ngược lại các đỉnh: CE= 5 - 3 - 1 - 2 - 4 - 3 - 2 - 5 - 6 - 4.	

Đồ thị G1

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0



Thuật toán tìm đường đi euler:

Thuật toán Euler-Path (u):

- u là đỉnh bậc lẻ đầu tiên nếu G là đồ thị vô hướng
- u là đỉnh có $\deg^+(u) - \deg(u) = 1$.

Bước 1 (Khởi tạo) :

```
stack =  $\emptyset$  ; //Khởi tạo một stack bắt đầu là  $\emptyset$   
dCE =  $\emptyset$  ; //Khởi tạo mảng dCE bắt đầu là  $\emptyset$   
Push (stack, u) ; //Đưa đỉnh u vào ngăn xếp
```

Bước 2 (Lặp) :

```
while (stack  $\neq \emptyset$  ) do { //Lặp cho đến khi stack rỗng  
    s = get(stack); //Lấy đỉnh ở đầu ngăn xếp  
    if ( Ke(s)  $\neq \emptyset$  ) then { // Nếu danh sách Ke(s) chưa rỗng  
        t = < Đỉnh đầu tiên trong Ke(s)>;  
        Push(stack, t) ; //Đưa t vào stack;  
        E = E \ (s,t); // Loại bỏ cạnh (s,t);  
    }  
    else { //Trường hợp Ke(s)= $\emptyset$   
        s = Pop(stack); // Đưa s ra khỏi ngăn xếp  
        s  $\Rightarrow$  dCE; //Đưa s sang dCE  
    }  
}
```

Bước 3 (Trả lại kết quả) :

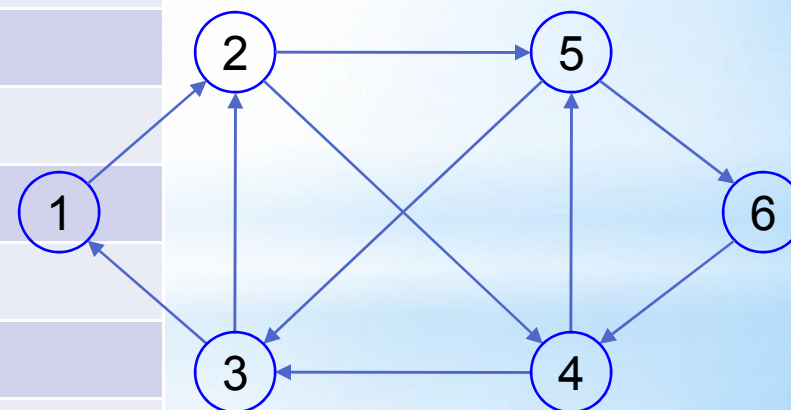
<Lật ngược lại các đỉnh trong dCE ta được đường đi Euler> ;

Kiểm nghiệm thuật toán: Euler-Cutle(1)

STT	Trạng thái stack?	Giá trị CE?
1	1	ϕ
2	1,2	ϕ
3	1,2,4	ϕ
4	1,2,4,3	ϕ
5	1,2,4,3, 1	ϕ
6	1,2,4,3	1
7	1,2,4,3,2	1
8	1,2,4,3,2,5	1
9	1,2,4,3,2,5,3	1
10	1,2,4,3,2,5	1,3
11	1,2,4,3,2,5,6	1,3
12	1,2,4,3,2,5,6,4	1,3
13	1,2,4,3,2,5,6,4,5	1,3
14	ϕ	1,3,5,4,6,5,2,3,4,2,1
15	Lật ngược lại các đỉnh: CE= 1-2-4-3-2-5-6-4-5-3-1.	

Đồ thị G1

0	1	0	0	0	0
0	0	0	1	1	0
1	1	0	0	0	0
0	0	1	0	1	0
0	0	1	0	0	1
0	0	0	1	0	0



5.6. Đồ thị Hamilton

Định nghĩa. Đường đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần được gọi là đường đi Hamilton. Chu trình bắt đầu tại một đỉnh v nào đó qua tất cả các đỉnh còn lại mỗi đỉnh đúng một lần sau đó quay trở lại v được gọi là chu trình Hamilton. Đồ thị có chu trình Hamilton được gọi là đồ thị Hamilton. Đồ thị có đường đi Hamilton được gọi là đồ thị nửa Hamilton.

Để liệt kê tất cả các chu trình Hamilton ta thực hiện theo thuật toán sau:

```
Thuật toán Hamilton( int k) {  
    /* Liệt kê các chu trình Hamilton của đồ thị bằng cách phát triển dãy đỉnh  
    (X[1], X[2], . . . , X[k-1] ) của đồ thị  $G = (V, E)$  */  
    for  $y \in Ke(X[k-1])$  {  
        if ( $k == n+1$ ) and ( $y == v0$ ) then  
            Ghinhan(X[1], X[2], . . . , X[n], v0);  
        else {  
            X[k]=y; chuaxet[y] = false;  
            Hamilton(k+1);  
            chuaxet[y] = true;  
        }  
    }  
}
```

5.7. Cây khung của đồ thị

5.7.1. Đặt bài toán

Định nghĩa. Ta gọi cây là đồ thị vô hướng liên thông không có chu trình. Đồ thị không liên thông được gọi là rừng.

Cây được coi là dạng đồ thị đơn giản nhất của đồ thị. Định lý sau đây cho ta một số tính chất của cây.

Định lý. Giả sử $T = \langle V, E \rangle$ là đồ thị vô hướng n đỉnh. Khi đó những khẳng định sau là tương đương

- T là một cây;
- T không có chu trình và có $n-1$ cạnh;
- T liên thông và có đúng $n-1$ cạnh;
- T liên thông và mỗi cạnh của nó đều là cầu;
- Giữa hai đỉnh bất kỳ của T được nối với nhau bởi đúng một đường đi đơn;
- T không chứa chu trình nhưng nếu thêm vào nó một cạnh ta thu được đúng một chu trình;

Định nghĩa 2. Cho G là đồ thị vô hướng liên thông. Ta gọi đồ thị con T của G là một cây khung của G (Cây bao trùm) nếu T thoả mãn hai điều kiện:

- T là một cây;
- Tập đỉnh của T đúng bằng tập đỉnh của G .

Trong lý thuyết đồ thị, người ta qua tâm đến hai bài toán cơ bản về cây:

Bài toán 1. Cho đồ thị vô hướng $G = \langle V, E \rangle$. Hãy xây dựng một cây khung của đồ thị bắt đầu tại đỉnh $u \in V$.

Bài toán 2. Cho đồ thị vô hướng $G = \langle V, E \rangle$ có trọng số. Hãy xây dựng cây khung có độ dài nhỏ nhất.

Bài toán 1 được giải quyết bằng các thuật toán tìm kiếm cơ bản: thuật toán DFS hoặc BFS. Bài toán 2 được giải quyết bằng thuật toán Kruskal hoặc PRIM.

5.7.2. Xây dựng cây khung của đồ thị dựa vào DFS :

Thuật toán Tree-DFS(u) {

Bước 1 (khởi tạo):

Stack = ϕ ; sc = 0; T = ϕ ; //Khởi tạo stack, số cạnh, và tập cạnh cây

Push(Stack, u); chuaxet[u] = False;

Bước 2 (Lặp):

While (Stack $\neq \phi$) do { //lặp đến khi stack rỗng

s = Pop(Stack); //Lấy s ra khỏi ngăn xếp

For t \in Ke(s) do { //Duyệt các đỉnh t kề với s

If (chuaxet[s]) { //nếu t đúng là chưa xét

T = T \cup (s,t); sc++; chuaxet[t] = False;

Push(Stack, s); Push(Stack, t); break;

}

}

}

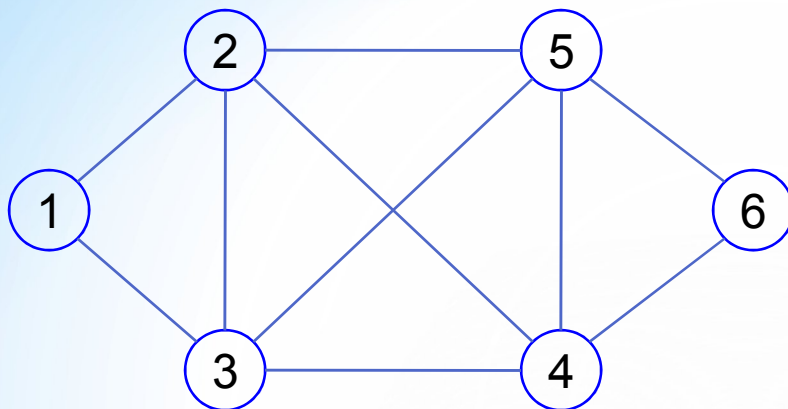
Bước 3 (Trả lại kết quả):

if (sc < n-1) <Đồ thị không liên thông>;

else return (T);

}

Kiểm nghiệm thuật toán: Tree-DFS(1)



Đồ thị G1

0	1	1	0	0	0
1	0	1	1	1	0
1	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	0	1
0	0	0	1	1	0

STT	Trạng thái stack?	Tập cạnh T=?
1	1	$T = \emptyset$
2	1,2	$T = T \cup (1,2); sc=1$
3	1,2,3	$T = T \cup (2,3); sc=2$
4	1,2,3, 4	$T = T \cup (3,4); sc=3$
5	1,2,3, 4, 5	$T = T \cup (4,5); sc=4$
6	1,2,3, 4, 5, 6	$T = T \cup (5,6); sc=5$
7	\emptyset	
Kết luận: $T = \{(1,2), (2,3), (3,4), (4,5), (5,6)\}$		

Bài tập:

Input: $G = \langle V, E \rangle$ là đồ thị được biểu diễn dưới dạng danh sách cạnh.

Output: Cây khung $H = \langle V, T \rangle$ của G hoặc thông báo đồ thị không liên thông.

Yêu cầu:

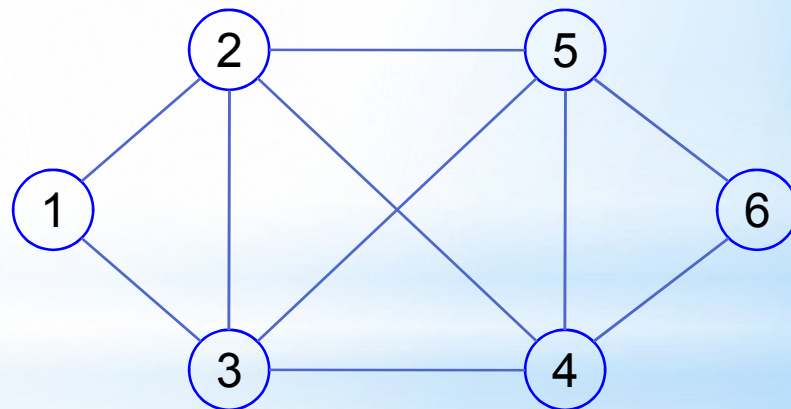
- 1) Sử dụng các thao tác trên stack sử dụng mảng (Version 1)?
- 2) Sử dụng các thao tác trên stack sử dụng danh sách liên kết đơn (Version 2)?
- 3) Sử dụng các thao tác trên stack sử dụng STL C++ (Version 3)?
- 4) Dữ liệu vào được tự động nhập từ file dothi.in. Kết quả ra tự động ghi vào file Cay.out theo khuôn dạng dưới đây:

Input:

6	10
1	2
1	3
2	3
2	4
2	5
3	4
3	5
4	5
4	6
5	6

Output:

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



5.7.3. Xây dựng cây khung của đồ thị dựa vào BFS :

Thuật toán Tree-BFS(u) {

Bước 1 (khởi tạo):

Queue = ϕ ; sc = 0; T = ϕ ; *//Khởi tạo queue, số cạnh, và tập cạnh cây*

Push(Queue, u); chuaxet[u] = False;

Bước 2 (Lặp):

While (Queue $\neq \phi$) do {*//lặp đến khi hàng đợi rỗng*

s = Pop(Queue); *//Lấy s ra khỏi hàng đợi*

For t \in Ke(s) do { *//Duyệt các đỉnh t kề với s*

If (chuaxet[s]) { *//nếu t đúng là chưa xét*

T = T \cup (s,t); *//Thêm cạnh (s,t) vào cây*

sc++; *//Tăng số cạnh lên 1*

Push(Queue, t); chuaxet[t] = False;

}

}

}

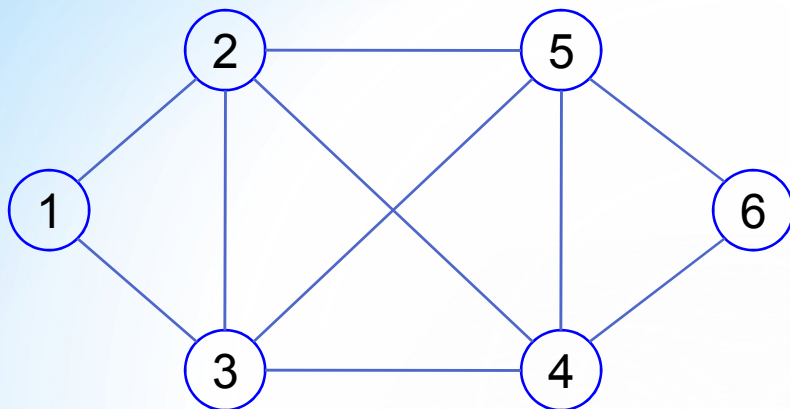
Bước 3 (Trả lại kết quả):

if (sc < n-1) <Đồ thị không liên thông>;

else return (T);

}

Kiểm nghiệm thuật toán: Tree-BFS(1)



Đồ thị G1

0	1	1	0	0	0
1	0	1	1	1	0
1	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	0	1
0	0	0	1	1	0

STT	Trạng thái queue?	Tập cạnh T=?
1	1	$T = \emptyset$
2	2, 3	$T = T \cup \{(1,2), (1,3)\}; sc=2$
3	3, 4, 5	$T = T \cup \{(2,4), (2,5)\}; sc=4$
4	4, 5	$T = T \cup \emptyset; sc=4$
5	5, 6	$T = T \cup (4,6); sc=5$
6	6	$T = T \cup \emptyset; sc=5$
7	\emptyset	
Kết luận: $T = \{(1,2), (1,3), (2,3), (2,4), (4,6)\}$		

Bài tập:

Input: $G = \langle V, E \rangle$ là đồ thị được biểu diễn dưới dạng danh sách cạnh.

Output: Cây khung $H = \langle V, T \rangle$ của G hoặc thông báo đồ thị không liên thông.

Yêu cầu:

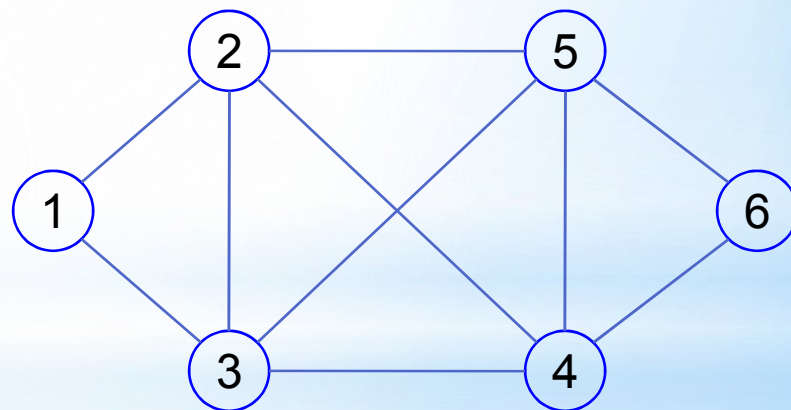
- 1) Sử dụng các thao tác trên Queue sử dụng mảng (Version 1)?
- 2) Sử dụng các thao tác trên Queue sử dụng danh sách liên kết đơn (Version 2)?
- 3) Sử dụng các thao tác trên Queue sử dụng STL C++ (Version 3)?
- 4) Dữ liệu vào được tự động nhập từ file dothi.in. Kết quả ra tự động ghi vào file Cay.out theo khuôn dạng dưới đây:

Input:

6	10
1	2
1	3
2	3
2	4
2	5
3	4
3	5
4	5
4	6
5	6

Output:

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



5.7.4. Bài toán tìm cây khung nhỏ nhất

Bài toán tìm cây khung nhỏ nhất là một trong những bài toán đồ thị có ứng dụng trong nhiều lĩnh vực khác nhau của thực tế. Bài toán được phát biểu như dưới đây.

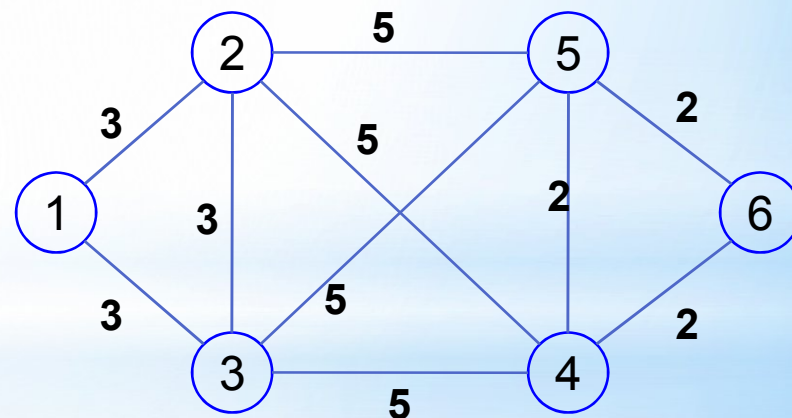
Đặt bài toán. Cho $G = \langle V, E \rangle$ là đồ thị vô hướng liên thông với tập đỉnh $V = \{1, 2, \dots, n\}$ và tập cạnh E gồm m cạnh. Mỗi cạnh e của đồ thị được gán với một số không âm $c(e)$ được gọi là độ dài cạnh. Giả sử $H = \langle V, T \rangle$ là một cây khung của đồ thị G . Ta gọi độ dài $c(H)$ của cây khung H được tính bằng tổng độ dài các cạnh. Bài toán được đặt ra là, trong số các cây khung của đồ thị hãy tìm cây khung có độ dài nhỏ nhất của đồ thị.

Input:

6	10	
1	2	3
1	3	3
2	3	3
2	4	5
2	5	5
3	4	5
3	5	5
4	5	2
4	6	2
5	6	2

Output:

6	5	15
4	5	2
4	6	2
1	2	3
1	3	3
2	4	5



5.7.4.1. Thuật toán Kruskal

Thuật toán Kruskal:

Begin

Bước 1 (Khởi tạo):

$T = \emptyset$; // Khởi tạo tập cạnh cây khung là \emptyset

$d(H) = 0$; // Khởi tạo độ dài nhỏ nhất cây khung là 0

Bước 2 (Sắp xếp):

<Sắp xếp các cạnh của đồ thị theo thứ tự giảm dần của trọng số>;

Bước 3 (Lặp):

while ($|T| < n-1$ && $E \neq \emptyset$) do { // Lặp nếu $E \neq \emptyset$ và $|T| < n-1$

$e = \text{<Cạnh có độ dài nhỏ nhất>};$

$E = E \setminus \{e\}$; // Loại cạnh e ra khỏi đồ thị

if ($T \cup \{e\}$ không tạo nên chu trình) then {

$T = T \cup \{e\}$; // Kết nạp e vào tập cạnh cây khung

$d(H) = d(H) + d(e)$; // Độ dài của tập cạnh cây khung

endif;

endwhile;

Bước 4 (Trả lại kết quả):

if ($|T| < n-1$) then <Đồ thị không liên thông>;

else

Return(T , $d(H)$);

end.

Kiểm nghiệm thuật toán Kruskal: Giả sử ta cần xây dựng cây khung nhỏ nhất trên đồ thị được biểu diễn dưới dạng ma trận trọng số dưới đây:

∞	2	1	3	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	2	∞	∞	5	5	∞	∞	∞	∞	∞	∞
1	2	∞	4	∞	5	∞	∞	∞	∞	∞	∞	∞
3	∞	4	∞	5	5	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	5	∞	6	∞	∞	∞	6	∞	∞	∞
∞	5	5	5	6	∞	6	6	6	6	∞	∞	∞
∞	5	∞	∞	∞	6	∞	6	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	6	6	∞	7	∞	∞	7	7
∞	∞	∞	∞	∞	6	∞	7	∞	7	7	∞	∞
∞	∞	∞	∞	6	6	∞	∞	7	∞	7	7	∞
∞	∞	∞	∞	∞	∞	∞	∞	7	7	∞	8	∞
∞	∞	∞	∞	∞	∞	∞	7	∞	7	8	∞	8
∞	∞	∞	∞	∞	∞	∞	7	∞	∞	∞	8	∞

Bước 1: $T = \phi$; $D(T) = 0$;

Bước 2. Sắp xếp các cạnh theo thứ tự tăng dần của trọng số

Đầu	Cuối	Tr.Số
1	2	2
1	3	1
1	4	3
2	3	2
2	6	5
2	7	5
3	4	4
3	6	5
4	5	5
4	6	5
5	6	6
5	10	6
6	7	6
6	8	6
6	9	6
6	10	6
7	8	6
8	9	7
8	12	7
8	13	7
9	10	7
9	11	7
10	11	7
10	12	7
11	12	8
12	13	8

Đầu	Cuối	Tr.Số
1	3	1
1	2	2
2	3	2
1	4	3
3	4	4
2	6	5
2	7	5
3	6	5
4	5	5
4	6	5
5	6	6
5	10	6
6	7	6
6	8	6
6	9	6
6	10	6
7	8	6
8	9	7
8	12	7
8	13	7
9	10	7
9	11	7
10	11	7
10	12	7
11	12	8
12	13	8

Bước 3 (lắp) :

STT	Cạnh được xét	$T \cup e$
1	$E \setminus (1,3)$	$T = T \cup (1,3); D(T) = 1$
2	$E = E(1,2)$	$T = T \cup (1,2); D(T) = 1+2=3$
3	$E = E(2,3)$	<i>Tạo nên chu trình</i>
4	$E = E(1,4)$	$T = T \cup (1,4); D(T) = 3+3=6$
5	$E = E(3,4)$	<i>Tạo nên chu trình</i>
6	$E = E(2,6)$	$T = T \cup (2,6); D(T) = 6+5=11$
7	$E = E(2,7)$	$T = T \cup (2,7); D(T) = 11+5=16$
8	$E = E(3,6)$	<i>Tạo nên chu trình</i>
9	$E = E(4,5)$	$T = T \cup (4,5); D(T) = 16+5=21$
10	$E = E(4,6)$	<i>Tạo nên chu trình</i>
11	$E = E(5,6)$	<i>Tạo nên chu trình</i>
12	$E = E(5,10)$	$T = T \cup (5,10); D(T) = 21+6=27$
13	$E = E(6,7)$	<i>Tạo nên chu trình</i>
14	$E = E(6,8)$	$T = T \cup (6,8); D(T) = 27+6=33$
15	$E = E(6,9)$	$T = T \cup (6,9); D(T) = 33+6=39$
16	$E = E(6,10)$	<i>Tạo nên chu trình</i>
17	$E = E(7,8)$	<i>Tạo nên chu trình</i>
18	$E = E(8,9)$	<i>Tạo nên chu trình</i>
19	$E = E(8,12)$	$T = T \cup (8,12); D(T) = 39+7=46$
20	$E = E(8,13)$	$T = T \cup (8,13); D(T) = 46+7=53$
21	$E = E(9,10)$	<i>Tạo nên chu trình</i>
22	$E = E(9,11)$	$T = T \cup (9,11); D(T) = 53+7=60$
Bước lắp kết thúc vì $ T > N-1 = 12$		

Bước 4 : Trả lại kết quả:

$$T = \{ (1,3), (1,2), (1,4), (2,6), (2,7), (4,5), (5,10), (6,8), (6,9), (8,12), (8,13), (9,11) \}$$

$$D(T) = 1 + 2 + 3 + 5 + 5 + 5 + 6 + 6 + 6 + 7 + 7 + 7 = 60$$

5.7.4.2. Thuật toán PRIM

Thuật toán PRIM (s):

Begin:

Bước 1 (Khởi tạo):

$V_H = \{s\}$; //Tập đỉnh cây khung thiết lập ban đầu là s

$V = V \setminus \{s\}$; //Tập đỉnh V được bớt đi s

$T = \emptyset$; //Tập cạnh cây khung thiết lập ban đầu là \emptyset

$d(H) = 0$; //Độ dài cây khung được thiết lập là 0

Bước 2 (Lặp):

while ($V \neq \emptyset$) do {

$e = \langle u, v \rangle$: cạnh có độ dài nhỏ nhất thỏa mãn $u \in V, v \in V_H$;

$d(H) = d(H) + d(e)$; // Thiết lập độ dài cây khung nhỏ nhất

$T = T \cup \{e\}$; //Kết nạp e vào cây khung

$V = V \setminus \{u\}$; // Tập đỉnh V bớt đi đỉnh u

$V_H = V_H \cup \{u\}$; // Tập đỉnh V_H thêm vào đỉnh u

endwhile;

Bước 3 (Trả lại kết quả):

if ($|T| < n-1$) then <Đồ thị không liên thông>;

else Return(T, d(H));

End.

Kiểm nghiệm thuật toán PRIM: Giả sử ta cần xây dựng cây khung nhỏ nhất trên đồ thị được biểu diễn dưới dạng ma trận trọng số dưới đây:

∞	2	1	3	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	2	∞	∞	5	5	∞	∞	∞	∞	∞	∞
1	2	∞	4	∞	5	∞	∞	∞	∞	∞	∞	∞
3	∞	4	∞	5	5	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	5	∞	6	∞	∞	∞	6	∞	∞	∞
∞	5	5	5	6	∞	6	6	6	6	∞	∞	∞
∞	5	∞	∞	∞	6	∞	6	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	6	6	∞	7	∞	∞	7	7
∞	∞	∞	∞	∞	6	∞	7	∞	7	7	∞	∞
∞	∞	∞	∞	6	6	∞	∞	7	∞	7	7	∞
∞	∞	∞	∞	∞	∞	∞	∞	7	7	∞	8	∞
∞	∞	∞	∞	∞	∞	∞	7	∞	7	8	∞	8
∞	∞	∞	∞	∞	∞	∞	7	∞	∞	∞	8	∞

Bước khởi tạo: $T = \emptyset$; $D(T) = 0$; $V = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13$; $V_H = 1$

$e = (v, t) \mid v \in V, t \in V_T \text{ có độ dài nhỏ nhất}$	$V \setminus v = ?$	$V_H \cup v = ?$	$T, D(T)$
(1,3)	2,4,5,6,7,8,9,10,11,12,13	1,3	$T = T \cup (1,3)$ $D(T) = 0 + 1$
(1,2)	4,5,6,7,8,9,10,11,12,13	1,2,3	$T = T \cup (1,2)$ $D(T) = 1 + 2 = 3$
(1,4)	5,6,7,8,9,10,11,12,13	1,2,3,4	$T = T \cup (1,4)$ $D(T) = 3 + 3 = 6$
(2,6)	5, 7,8,9,10,11,12,13	1,2,3,4,6	$T = T \cup (2,6)$ $D(T) = 6 + 5 = 11$
(2,7)	5, 8,9,10,11,12,13	1,2,3,4,6,7	$T = T \cup (2,7)$ $D(T) = 11 + 5 = 16$
(4,5)	8,9,10,11,12,13	1,2,3,4,5, 6,7	$T = T \cup (4,5)$ $D(T) = 16 + 5 = 21$
(5,10)	8,9,11,12,13	1,2,3,4,5, 6,7,10	$T = T \cup (5,10)$ $D(T) = 21 + 6 = 27$
(6,8)	9,11,12,13	1,2,3,4,5, 6,7,8,10	$T = T \cup (6,8)$ $D(T) = 27 + 6 = 33$
(6,9)	11,12,13	1,2,3,4,5, 6,7,8,9,10	$T = T \cup (6,9)$ $D(T) = 33 + 6 = 39$
(8,12)	11,13	1,2,3,4,5, 6,7,8,9,10,12	$T = T \cup (8,12)$ $D(T) = 39 + 7 = 46$
(8,13)	11	1,2,3,4,5, 6,7,8,9,10,12,13	$T = T \cup (8,13)$ $D(T) = 46 + 7 = 53$
(9,11)	\emptyset	1,2,3,4,5, 6,7,8,9,10,12,13,11	$T = T \cup (9,11)$ $D(T) = 53 + 7 = 60$
$V = \emptyset$: kết thúc bước lặp			

Kết quả: $T = \{ (1,3), (1,2), (1,4), (2,6), (2,7), (4,5), (5,10), (6,8), (6,9), (8,12), (8,13), (9,11) \}$
 $D(T) = 1 + 2 + 3 + 5 + 5 + 5 + 6 + 6 + 6 + 7 + 7 + 7 = 60$

5.8. Bài toán tìm đường đi ngắn nhất

Phát biểu bài toán. Xét đồ thị $G = \langle V, E \rangle$; trong đó $|V| = n$, $|E| = m$. Với mỗi cạnh $(u, v) \in E$, ta đặt tương ứng với nó một số thực $A[u][v]$ được gọi là trọng số của cạnh. Ta sẽ đặt $A[u, v] = \infty$ nếu $(u, v) \notin E$. Nhiệm vụ của bài toán là tìm đường đi ngắn nhất từ một đỉnh xuất phát $s \in V$ (đỉnh nguồn) đến đỉnh cuối $t \in V$ (đỉnh đích). Đường đi như vậy được gọi là đường đi ngắn nhất từ s đến t , độ dài của đường đi $d(s, t)$ được gọi là khoảng cách ngắn nhất từ s đến t (trong trường hợp tổng quát $d(s, t)$ có thể âm). Nếu như không tồn tại đường đi từ s đến t thì độ dài đường đi $d(s, t) = \infty$.

Trường hợp 1. Nếu s cố định và t thay đổi, khi đó bài toán được phát biểu dưới dạng tìm đường đi ngắn nhất từ s đến tất cả các đỉnh còn lại trên đồ thị. Đối với đồ thị có trọng số không âm, bài toán luôn có lời giải bằng thuật toán Dijkstra. Đối với đồ thị có trọng số âm nhưng không tồn tại chu trình âm, bài toán có lời giải bằng thuật toán Bellman-Ford. Trong trường hợp đồ thị có chu trình âm, bài toán không có lời giải.

Trường hợp 2. Nếu s thay đổi và t cũng thay đổi, khi đó bài toán được phát biểu dưới dạng tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh của đồ thị. Bài toán luôn có lời giải trên đồ thị không có chu trình âm. Đối với đồ thị có trọng số không âm, bài toán được giải quyết bằng cách thực hiện lặp lại n lần thuật toán Dijkstra. Đối với đồ thị không có chu trình âm, bài toán có thể giải quyết bằng thuật toán Floyd.

5.8.1. Thuật toán Dijkstra

Thuật toán Dijkstra (s): // $s \in V$ là một đỉnh bất kỳ của $G = \langle V, E \rangle$

Begin

Bước 1 (Khởi tạo):

$d[s]=0$; // Gán nhãn của đỉnh s là 0

$T = V \setminus \{s\}$; // T là tập đỉnh có nhãn tạm thời

for each $v \in V$ do { // Sử dụng s gán nhãn cho các đỉnh còn lại

$d[v] = A[s, v]$;

$truooc[v]=s$;

endfor;

Bước 2 (Lặp):

while ($T \neq \emptyset$) do {

 Tìm đỉnh $u \in T$ sao cho $d[u] = \min \{ d[z] \mid z \in T \}$;

$T = T \setminus \{u\}$; // cố định nhãn đỉnh u

 for each $v \in T$ do { // Sử dụng u , gán nhãn lại cho các đỉnh

 if ($d[v] > d[u] + A[u, v]$) then {

$d[v] = d[u] + A[u, v]$; // Gán lại nhãn cho đỉnh v ;

$truooc[v] = u$;

 endif;

 endfor;

endwhile;

Bước 3 (Trả lại kết quả):

Return ($d[s], truooc[s]$);

End.

Kiểm nghiệm thuật toán Dijkstra: Giả sử ta cần đường đi ngắn nhất từ đỉnh q đến tất cả các đỉnh còn lại của đồ thị được biểu diễn dưới dạng ma trận trọng số dưới đây:

∞	2	8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	2	∞	∞	∞	9	∞	∞	∞	∞	∞	∞
∞	∞	∞	6	∞	8	1	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	1	7	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	1	∞	∞	9	8	∞	∞	∞	∞
∞	∞	∞	∞	∞	2	∞	2	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	9	∞	∞	2	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	6	∞	9	8
∞	∞	∞	∞	7	6	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	6	7	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	7	∞	∞

Kết quả các bước thực hiện thuật toán:

Bước	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6	Đỉnh 7	Đỉnh 8	Đỉnh 9	Đỉnh 10	Đỉnh 11	Đỉnh 12	Đỉnh 13
1	$\langle 0,1 \rangle$	$\langle 2,1 \rangle$	$\langle 8,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$
2	*	$\langle 2,1 \rangle$	$\langle 4,2 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle 11,2 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$
3	*	*	$\langle 4,2 \rangle$	$\langle 10,3 \rangle$	$\langle \infty,1 \rangle$	$\langle 12,3 \rangle$	$\langle 5,3 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$
4	*	*	*	$\langle 10,3 \rangle$	$\langle \infty,1 \rangle$	$\langle 7,7 \rangle$	$\langle 5,3 \rangle$	$\langle 7,7 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$
5	*	*	*	$\langle 10,3 \rangle$	$\langle 8,6 \rangle$	$\langle 7,7 \rangle$	*	$\langle 7,7 \rangle$	$\langle 15,6 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$
6	*	*	*	$\langle 10,3 \rangle$	$\langle 8,6 \rangle$	*	*	$\langle 7,7 \rangle$	$\langle 15,6 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle 9,8 \rangle$	$\langle \infty,1 \rangle$
7	*	*	*	$\langle 10,3 \rangle$	$\langle 8,6 \rangle$	*	*	*	$\langle 15,6 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle 9,8 \rangle$	$\langle \infty,1 \rangle$
8	*	*	*	$\langle 10,3 \rangle$	*	*	*	*	$\langle 15,6 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	$\langle 9,8 \rangle$	$\langle 11,12 \rangle$
9	*	*	*	$\langle 10,3 \rangle$	*	*	*	*	$\langle 15,6 \rangle$	$\langle \infty,1 \rangle$	$\langle \infty,1 \rangle$	*	$\langle 11,12 \rangle$
10	*	*	*	*	*	*	*	*	$\langle 15,6 \rangle$	$\langle \infty,1 \rangle$	$\langle 18,13 \rangle$	*	$\langle 11,12 \rangle$
11	*	*	*	*	*	*	*	*	$\langle 15,6 \rangle$	$\langle 21,9 \rangle$	$\langle 18,13 \rangle$	*	*
12	*	*	*	*	*	*	*	*	*	$\langle 21,9 \rangle$	$\langle 18,13 \rangle$	*	*
13	*	*	*	*	*	*	*	*	*	$\langle 21,9 \rangle$	*	*	*

Kết quả :

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 2: 2. Đường đi: 1-2.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 3: 4. Đường đi: 1-2-3.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 4: 10. Đường đi: 1-2-3-10.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 5: 8. Đường đi: 1-2-3-7-6-5.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 6: 7. Đường đi: 1-2-3-7-6.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 7: 5. Đường đi: 1-2-3-7.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 8: 7. Đường đi: 1-2-3-7-8.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 9: 15. Đường đi: 1-2-3-7-6-9.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 10: 21. Đường đi: 1-2-3-7-6-9-10.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 11: 18. Đường đi: 1-2-3-7-8-12-13-11.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 12: 18. Đường đi: 1-2-3-7-8-12.

Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 13: 11. Đường đi: 1-2-3-7-8-12-13.

5.8.2. Thuật toán Bellman-Ford

Thuật toán Bellman-Ford (s): // $s \in V$ là đỉnh bất kỳ của đồ thị

Begin:

Bước 1 (Khởi tạo):

```
for  $v \in V$  do { // Sử dụng s gán nhãn cho các đỉnh  $v \in V$   
     $D[v] = A[s][v]$ ;  
     $Truoc[v] = s$ ;  
}
```

Bước 2 (Lặp):

```
 $D[s] = 0$ ;  $K = 1$ ;  
while ( $K \leq N - 2$ ) { //  $N - 2$  vòng lặp  
    for  $v \in V \setminus \{s\}$  do { // Lấy mỗi đỉnh  $v \in V \setminus s$   
        for  $u \in V$  do { // Gán nhãn cho v  
            if ( $D[v] > D[u] + A[u][v]$ ) {  
                 $D[v] = D[u] + A[u][v]$ ;  
                 $Truoc[v] = u$ ;  
            }  
        }  
    }  
}  $K = K + 1$ ;  
endwhile;
```

Bước 3 (Trả lại kết quả):

```
Return(  $D[v], Truoc[v] : v \in U$  );
```

End.

Kiểm nghiệm thuật toán Bellman-Ford:

$$A = \begin{bmatrix} \infty & 1 & \infty & \infty & 3 \\ \infty & \infty & 3 & 3 & 8 \\ \infty & \infty & \infty & 1 & -5 \\ \infty & \infty & 2 & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty \end{bmatrix}$$

Khi đó, kết quả thực hiện theo thuật toán ta được kết quả sau:

Vòng lặp K=1:

$$v=2; D[2] = 1$$

$$D[1] + A[1, 2] = 0 + 1 \text{ (Không nhỏ hơn 1)}$$

$$D[2] + A[2, 2] = 1 + \infty > 1$$

$$D[3] + A[3, 2] = \infty + \infty > 1$$

$$D[4] + A[4, 2] = \infty + \infty > 1$$

$$D[5] + A[5, 2] = \infty + \infty > 1$$

$$v=3; D[3] = \infty$$

$$D[1] + A[1, 3] = 0 + \infty$$

$$D[2] + A[2, 3] = 1 + 3 = 4 < \infty \text{ (Thay } D[3] = 4, \text{ Trước}[3] = 2)$$

$$D[3] + A[3, 3] = 4 + \infty > 4$$

$$D[4] + A[4, 3] = \infty + 2 > 4$$

$$D[5] + A[5, 3] = \infty + \infty > 4$$

$$v=4; D[4] = \infty$$

$$D[1] + A[1, 4] = 0 + \infty$$

$$D[2] + A[2, 4] = 1 + 3 = 4 < \infty \text{ (Thay } D[4] = 4, \text{ Trước}[4] = 2)$$

$$D[3] + A[3, 4] = 4 + 1 = 5 > 4$$

$$D[4] + A[4, 4] = 4 + \infty > 4$$

$$D[5] + A[5, 4] = \infty + 4 > 4$$

$$v=5; D[5] = 3$$

$$D[1] + A[1, 5] = 0 + 3 \text{ (Không nhỏ hơn 3)}$$

$$D[2] + A[2, 5] = 1 + 8 = 9 > 3$$

$$D[3] + A[3, 5] = 4 - 5 = -1 < 3 \text{ (Thay } D[5] = -1, \text{ Trước}[5] = 3)$$

$$D[4] + A[4, 5] = 4 + \infty > -1$$

$$D[5] + A[5, 5] = -1 + \infty > -1$$

Kiểm nghiệm thuật toán Bellman-Ford:

Vòng lặp K=2:

$v=2; D[2] = 1$

$D[1] + A[1, 2] = 0+1$ (Không nhỏ hơn 1)

$D[2] + A[2, 2] = 1 + \infty > 1$

$D[3] + A[3, 2] = 4 + \infty > 1$

$D[4] + A[4, 2] = 4 + \infty > 1$

$D[5] + A[5, 2] = -1 + \infty > 1$

$v=3; D[3] = 4$

$D[1] + A[1, 3] = 0+\infty > 4$

$D[2] + A[2, 3] = 1 + 3 = 4$ (Không nhỏ hơn 4)

$D[3] + A[3, 3] = 4 + \infty > 4$

$D[4] + A[4, 3] = 4 + 2 > 4$

$D[5] + A[5, 3] = -1 + \infty > 4$

$v=4; D[4] = 4$

$D[1] + A[1, 4] = 0+\infty > 4$

$D[2] + A[2, 4] = 1 + 3 = 4$ (Không nhỏ hơn 4)

$D[3] + A[3, 4] = 4 + 1 > 4$

$D[4] + A[4, 4] = 4 + \infty > 4$

$D[5] + A[5, 4] = -1 + 4 = 3 < 4$ (Thay $D[4] = 5$, Trước[4] = 5)

$v=5; D[5] = -1$

$D[1] + A[1, 5] = 0+\infty > -1$

$D[2] + A[2, 5] = 1 + 3 = -1$

$D[3] + A[3, 5] = 4 + 1 > -1$

$D[4] + A[4, 5] = 3 + \infty > -1$

$D[5] + A[5, 5] = -1 + \infty > -1$

Kiểm nghiệm thuật toán Bellman-Ford:

Vòng lặp K=3:

$v=2; D[2] = 1$

$D[1] + A[1, 2] = 0+1$ (Không nhỏ hơn 1)

$D[2] + A[2, 2] = 1 + \infty > 1$

$D[3] + A[3, 2] = 4 + \infty > 1$

$D[4] + A[4, 2] = 3 + \infty > 1$

$D[5] + A[5, 2] = -1 + \infty > 1$

$v=3; D[3] = 4$

$D[1] + A[1, 3] = 0+\infty > 4$

$D[2] + A[2, 3] = 1 + 3 = 4$ (Không nhỏ hơn 4)

$D[3] + A[3, 3] = 4 + \infty > 4$

$D[4] + A[4, 3] = 3 + 2 > 4$

$D[5] + A[5, 3] = -1 + \infty > 4$

$v=4; D[4] = 3$

$D[1] + A[1, 4] = 0+\infty > 3$

$D[2] + A[2, 4] = 1 + 3 = 3$

$D[3] + A[3, 4] = 4 + 1 > 3$

$D[4] + A[4, 4] = 3 + \infty > 3$

$D[5] + A[5, 4] = -1 + 4 = 3$ (Không nhỏ hơn 3)

$v=5; D[5] = -1$

$D[1] + A[1, 5] = 0+\infty > -1$

$D[2] + A[2, 5] = 1 + 3 = -1$

$D[3] + A[3, 5] = 4 + 1 > -1$

$D[4] + A[4, 5] = 3 + \infty > -1$

$D[5] + A[5, 5] = -1 + \infty > -1$

Kiểm nghiệm thuật toán Bellman-Ford:

K=?	D[1], Truoc[1]	D[2], Truoc[2]	D[3], Truoc[3]	D[4], Truoc[4]	D[5], Truoc[5]
	$\langle 0, 1 \rangle$	$\langle 1, 1 \rangle$	$\langle \infty, 1 \rangle$	$\langle \infty, 1 \rangle$	$\langle 3, 1 \rangle$
1	$\langle 0, 1 \rangle$	$\langle 1, 1 \rangle$	$\langle 4, 2 \rangle$	$\langle 4, 2 \rangle$	$\langle -1, 3 \rangle$
2	$\langle 0, 1 \rangle$	$\langle 1, 1 \rangle$	$\langle 4, 2 \rangle$	$\langle 3, 5 \rangle$	$\langle -1, 3 \rangle$
3	$\langle 0, 1 \rangle$	$\langle 1, 1 \rangle$	$\langle 4, 2 \rangle$	$\langle 3, 5 \rangle$	$\langle -1, 3 \rangle$

5.8.2. Thuật toán Floyed

Thuật toán Floy:

Begin:

Bước 1 (Khởi tạo):

```
for (i=1; i ≤ n; i++) {  
    for (j =1; j ≤ n; j++) {  
        d[i,j] = a[i, j];  
        p[i,j] = i;  
    }  
}
```

Bước 2 (lặp) :

```
for (k=1; k ≤ n; k++) {  
    for (i=1; i ≤ n; i++){  
        for (j =1; j ≤ n; j++) {  
            if (d[i,j] > d[i, k] + d[k, j]) {  
                d[i, j] = d[i, k] + d[k, j];  
                p[i,j] = p[k, j];  
            }  
        }  
    }  
}
```

Bước 3 (Trả lại kết quả):

Return (p([i,j], d[i,j]: i, j ∈ V);