

Một số chủ đề trong Lý thuyết đồ thị

Lê Xuân Thành

Mục lục

- 1 Một số khái niệm cơ bản
- 2 Duyệt đồ thị
- 3 Đồ thị Euler
- 4 Đồ thị Hamilton
- 5 Đồ thị phẳng
- 6 Cây và cây khung
- 7 Đường đi ngắn nhất

Mục lục

1 Một số khái niệm cơ bản

2 Duyệt đồ thị

3 Đồ thị Euler

4 Đồ thị Hamilton

5 Đồ thị phẳng

6 Cây và cây khung

7 Đường đi ngắn nhất

Nội dung

- 1.1 Khái niệm đồ thị
- 1.2 Tính phổ biến của đồ thị
- 1.3 Các thuật ngữ cơ bản của đồ thị vô hướng
- 1.4 Các thuật ngữ cơ bản của đồ thị có hướng
- 1.5 Biểu diễn đồ thị
- 1.6 Liên thông trong đồ thị vô hướng
- 1.7 Liên thông trong đồ thị có hướng
- 1.8 Đangkan cầu đồ thị
- 1.9 Một số đồ thị đặc biệt

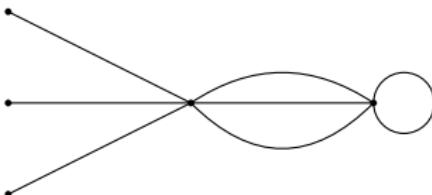
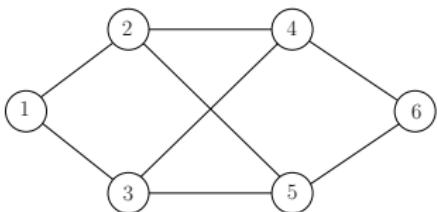
Đồ thị vô hướng

Định nghĩa

Một *đồ thị vô hướng* G được cấu thành bởi

- một tập hợp không rỗng V gồm các *đỉnh*, và
- một tập hợp $E \subseteq \{\{u, v\} \mid u, v \in V\}$ gồm các *cạnh*.

- Tập đỉnh V có thể vô hạn, tập cạnh E có thể rỗng
- Đồ thị vô hướng thường được biểu diễn trực quan như sau:
 - các đỉnh được biểu diễn bởi các vòng tròn hoặc các điểm
 - các cạnh được biểu diễn bởi các đường thẳng hoặc đường cong nối các đỉnh



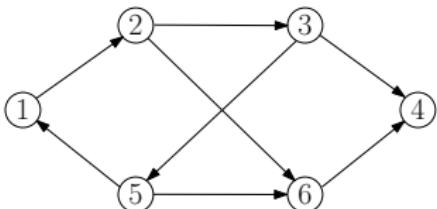
Đồ thị có hướng

Định nghĩa

Một *đồ thị có hướng* G được cấu thành bởi

- một tập hợp khác rỗng N gồm các *đỉnh* (nốt), và
- một tập hợp $A \subseteq \{(u, v) \mid u, v \in N\}$ gồm các *cung*.

- Tập đỉnh N có thể vô hạn, tập cung A có thể rỗng
- Đồ thị có hướng thường được biểu diễn trực quan như sau:
 - các đỉnh được biểu diễn bởi các vòng tròn hoặc các điểm
 - các cung được biểu diễn bởi các mũi tên nối các đỉnh

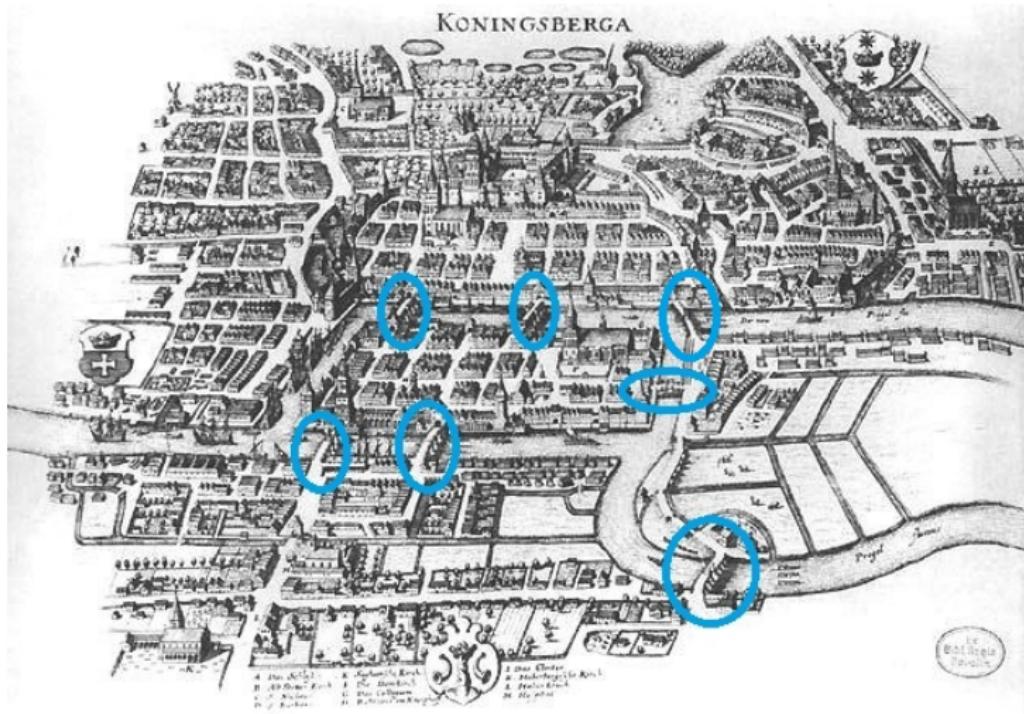


Nội dung

- 1.1 Khái niệm đồ thị
- 1.2 Tính phổ biến của đồ thị
- 1.3 Các thuật ngữ cơ bản của đồ thị vô hướng
- 1.4 Các thuật ngữ cơ bản của đồ thị có hướng
- 1.5 Biểu diễn đồ thị
- 1.6 Liên thông trong đồ thị vô hướng
- 1.7 Liên thông trong đồ thị có hướng
- 1.8 Đangkan cầu đồ thị
- 1.9 Một số đồ thị đặc biệt

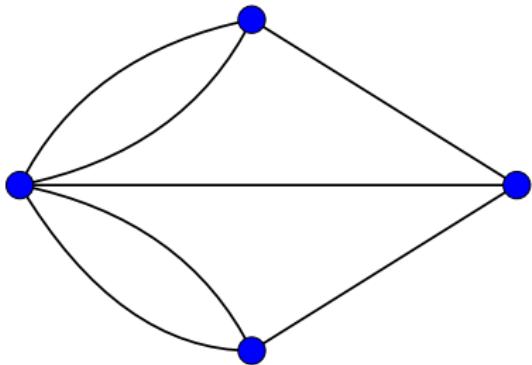
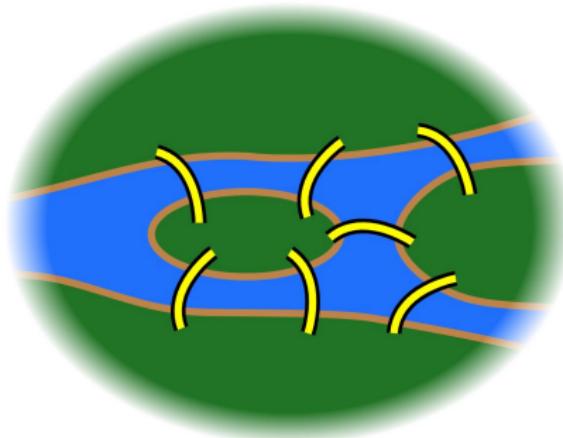
Tính phổ biến của đồ thị

- 1736: Bài toán Euler về 7 cây cầu ở Königsberg



Tính phổ biến của đồ thị

- 1736: Bài toán Euler về 7 cây cầu ở Königsberg



Tính phổ biến của đồ thị

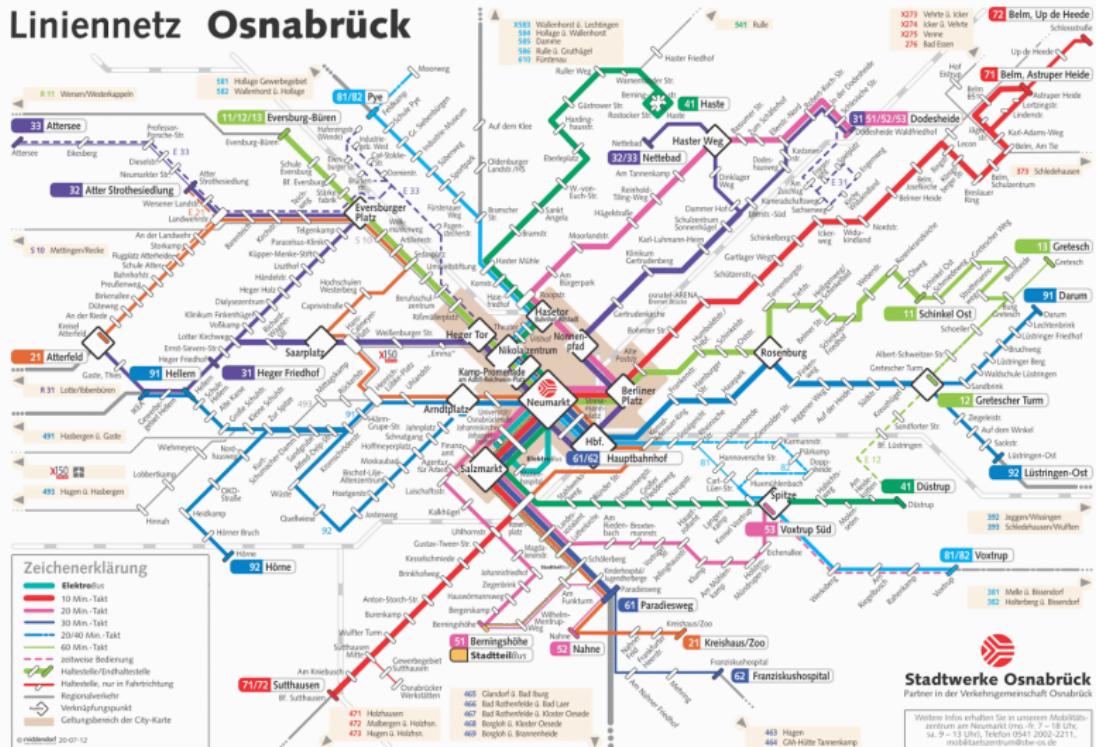
- 1847: Kirchhoff¹ mô hình hóa các mạch điện như các đồ thị
- 1857: Cayley² biểu diễn cấu trúc của C_nH_{n+2} như các đồ thị
- Đồ thị được sử dụng trong:
 - *Mạng xã hội*: mô hình hóa các cấu trúc xã hội dựa trên các kiểu quan hệ giữa người với người
 - *Mạng truyền thông*: mô hình hóa các thiết bị và kết nối truyền thông
 - *Mạng thông tin*: mô hình hóa các trang web và kết nối giữa chúng
 - *Mạng vận tải*: mô hình hóa các nhà ga và các tuyến vận tải
 - và rất nhiều lĩnh vực khác ...

¹Gustav Robert Kirchhoff (12.03.1824–17.10.1887): nhà vật lý người Đức

²Arthur Cayley (16.08.1821–26.01.1895): nhà toán học người Anh

Tính phổ biến của đồ thị

Liniennetz Osnabrück



 Stadtwerke Osnabrück

Partner in der Verkehrsgemeinschaft Osnabrück

Weitere Infos erhalten Sie in unserem Mobilitätszentrum am Neumarkt (mo.-fr. 7 - 18 Uhr, sa. 9 - 13 Uhr), Telefon 0541 2002-2211, mobilitaetszentrum@stw-ss.de

Nội dung

- 1.1 Khái niệm đồ thị
- 1.2 Tính phổ biến của đồ thị
- 1.3 Các thuật ngữ cơ bản của đồ thị vô hướng
- 1.4 Các thuật ngữ cơ bản của đồ thị có hướng
- 1.5 Biểu diễn đồ thị
- 1.6 Liên thông trong đồ thị vô hướng
- 1.7 Liên thông trong đồ thị có hướng
- 1.8 Đangkan cầu đồ thị
- 1.9 Một số đồ thị đặc biệt

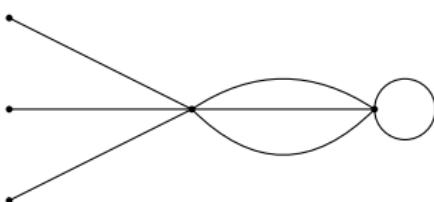
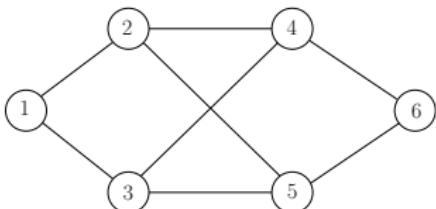
Các thuật ngữ cơ bản của đồ thị vô hướng

Cho trước một **đồ thị vô hướng** $G = (V, E)$.

Định nghĩa

- *Cấp* của G là số đỉnh của G (tức là $|V|$).
- *Cỡ* của G là số cạnh của G (tức là $|E|$).

Minh họa các khái niệm trên trong các ví dụ sau:

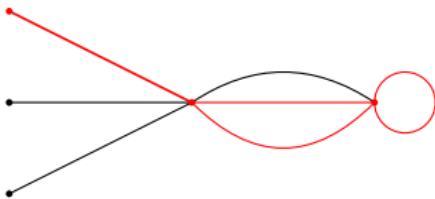
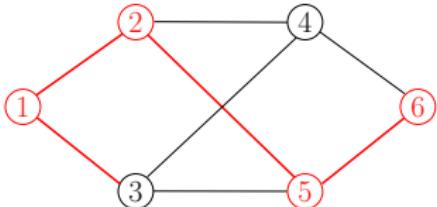


Các thuật ngữ cơ bản của đồ thị vô hướng

Cho trước một **đồ thị vô hướng** $G = (V, E)$.

Định nghĩa

- Một *đồ thị con* của G là một đồ thị vô hướng $G^* = (V^*, E^*)$ với $V^* \subseteq V$ và $E^* \subseteq E$.
- Một *đồ thị con thực sự* của G là một đồ thị con G^* thỏa mãn $G^* \neq G$.



Các thuật ngữ cơ bản của đồ thị vô hướng

Cho trước một **đồ thị vô hướng** $G = (V, E)$.

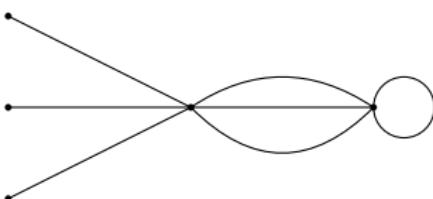
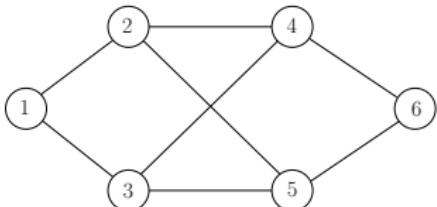
Định nghĩa

Hai đỉnh $u, v \in V$ được gọi là *kề nhau* nếu $e := \{u, v\} \in E$.

Khi đó ta nói

- cạnh e *kề (liên thuộc)* với đỉnh u và v ,
- cạnh e *nối* các đỉnh u và v ,
- đỉnh u, v *kề (liên thuộc)* với cạnh e ,
- đỉnh u là một hàng xóm của đỉnh v (và ngược lại).

Minh họa các khái niệm trên trong các ví dụ sau:



Các thuật ngữ cơ bản của đồ thị vô hướng

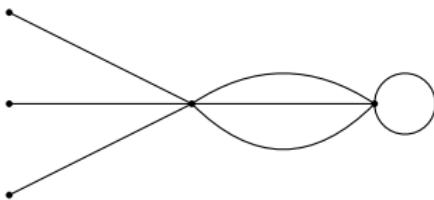
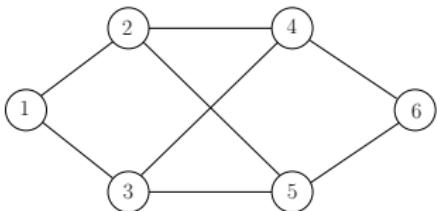
Cho trước một **đồ thị vô hướng** $G = (V, E)$.

Định nghĩa

- Tập hợp tất cả các hàng xóm của $v \in V$, ký hiệu là $N(v)$, được gọi là *lân cận* của v .
- Tập hợp tất cả các hàng xóm của các đỉnh trong $A \subseteq V$, ký hiệu là $N(A)$, được gọi là lân cận của tập đỉnh A .

Công thức: $N(A) = \bigcup_{v \in A} N(v)$

Minh họa các khái niệm trên trong các ví dụ sau:



Các thuật ngữ cơ bản của đồ thị vô hướng

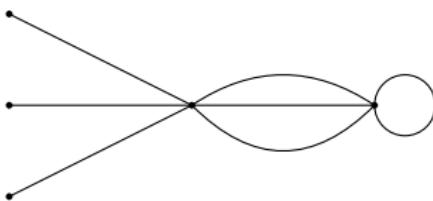
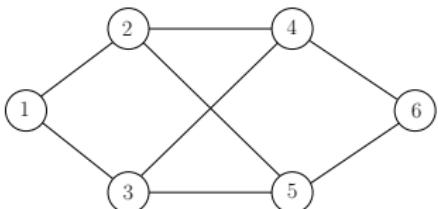
Cho trước một **đồ thị vô hướng** $G = (V, E)$.

Định nghĩa

Bậc của đỉnh $v \in V$, ký hiệu $\deg(v)$, là số cạnh kề với v . Trong trường hợp có khuyên $\{v, v\} \in E$, khuyên này đóng góp 2 đơn vị vào bậc của v .

- Đỉnh có bậc 0 được gọi là *đỉnh cô lập*
- Đỉnh có bậc 1 được gọi là *pendant*

Minh họa các khái niệm trên trong các ví dụ sau:



Một số kết quả cơ bản

Cho trước một đồ thị vô hướng $G = (V, E)$.

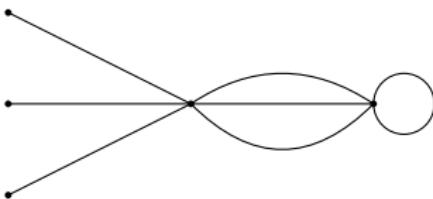
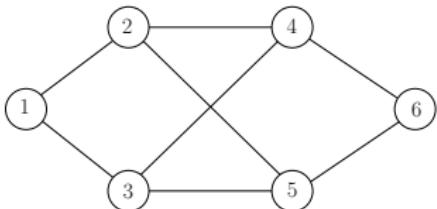
Định lý bắt tay

$$2|E| = \sum_{v \in V} \deg(v).$$

Hệ quả

Một đồ thị vô hướng phải có chẵn đỉnh có bậc lẻ.

Kiểm tra các kết quả trên với các ví dụ sau:



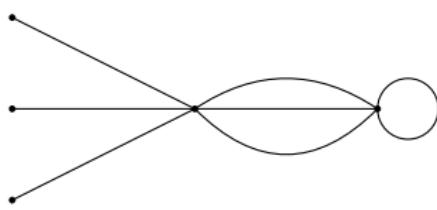
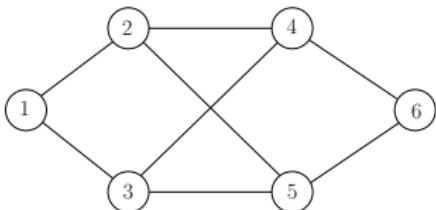
Các thuật ngữ cơ bản của đồ thị vô hướng

Định nghĩa

Một đồ thị vô hướng $G = (V, E)$ được gọi là *đơn* nếu

- không có khuyên (cạnh có dạng $\{v, v\}$ với $v \in V$) trong E , và
- có nhiều nhất một cạnh kề với mỗi cặp đỉnh.

Ví dụ: *Đồ thị đơn (trái)*



Từ bây giờ,
với đồ thị vô hướng ta chỉ xét đồ thị đơn.

Nội dung

- 1.1 Khái niệm đồ thị
- 1.2 Tính phổ biến của đồ thị
- 1.3 Các thuật ngữ cơ bản của đồ thị vô hướng
- 1.4 Các thuật ngữ cơ bản của đồ thị có hướng
- 1.5 Biểu diễn đồ thị
- 1.6 Liên thông trong đồ thị vô hướng
- 1.7 Liên thông trong đồ thị có hướng
- 1.8 Đangkan cầu đồ thị
- 1.9 Một số đồ thị đặc biệt

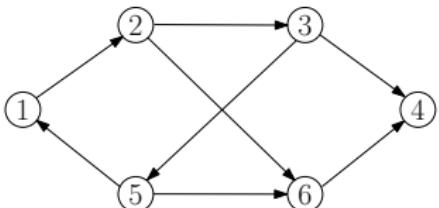
Các thuật ngữ cơ bản của đồ thị có hướng

Cho trước một **đồ thị có hướng** $G = (N, A)$.

Định nghĩa

- *Bậc* của G là số đỉnh của G (tức là $|N|$).
- *Cỡ* của G là số cung của G (tức là $|A|$).

Minh họa các khái niệm trên trong các ví dụ sau:

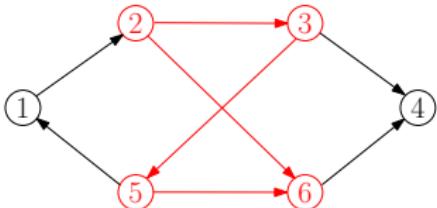


Các thuật ngữ cơ bản của đồ thị có hướng

Cho trước một **đồ thị có hướng** $G = (N, A)$.

Định nghĩa

- Một *đồ thị con* của G là một đồ thị có hướng $G^* = (N^*, A^*)$ với $N^* \subseteq N$ và $A^* \subseteq A$.
- Một *đồ thị con thực sự* của G là một đồ thị con G^* thỏa mãn $G^* \neq G$.



Các thuật ngữ cơ bản của đồ thị có hướng

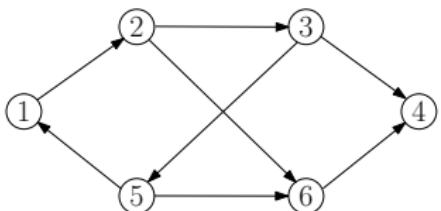
Cho trước một **đồ thị có hướng** $G = (N, A)$.

Định nghĩa

Khi $a = (u, v) \in A$, ta nói:

- đỉnh u kề tới đỉnh v , đỉnh v kề từ đỉnh u ,
- đỉnh u, v kề (*liên thuộc*) với cung a ,
- cung a kề (*liên thuộc*) với đỉnh u, v ,
- đỉnh u là *đỉnh bắt đầu* của cung a , đỉnh v là *đỉnh kết thúc* của a .

Minh họa các khái niệm trên trong các ví dụ sau:



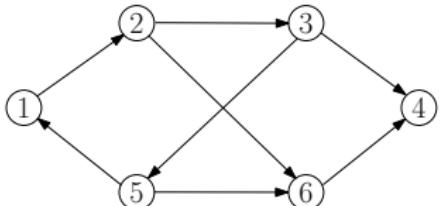
Các thuật ngữ cơ bản của đồ thị có hướng

Cho trước một **đồ thị có hướng** $G = (N, A)$.

Định nghĩa

- **Bậc vào** của một đỉnh $u \in N$, ký hiệu $\deg^-(u)$, là số cung nhận u là đỉnh kết thúc.
- **Bậc ra** của một đỉnh $u \in N$, ký hiệu $\deg^+(u)$, là số cung nhận u là đỉnh bắt đầu.
- Trong trường hợp có khuyên $(u, u) \in A$, khuyên này đóng góp 1 đơn vị vào bậc vào và 1 đơn vị vào bậc ra của u .

Minh họa các khái niệm trên trong các ví dụ sau:



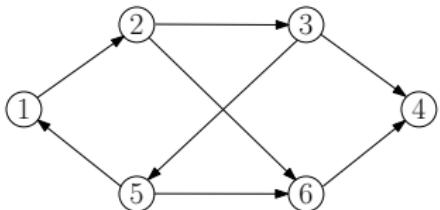
Basic results

Cho trước một đồ thị có hướng $G = (N, A)$.

Định lý

$$\sum_{v \in N} \deg^-(v) = \sum_{v \in N} \deg^+(v) = |A|.$$

Kiểm tra kết quả trên với các ví dụ sau đây:



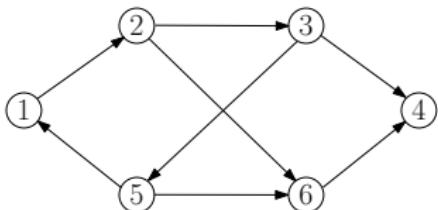
Các thuật ngữ cơ bản của đồ thị có hướng

Định nghĩa

Một đồ thị có hướng $G = (N, A)$ được gọi là **đơn** nếu

- không có khuyên (cung có dạng (v, v) với $v \in V$) trong A , và
- có nhiều nhất một cung kề với mỗi cặp đỉnh.

Ví dụ: Đồ thị đơn (trái)



Từ bây giờ,
với đồ thị có hướng ta chỉ xét đồ thị đơn.

Nội dung

- 1.1 Khái niệm đồ thị
- 1.2 Tính phổ biến của đồ thị
- 1.3 Các thuật ngữ cơ bản của đồ thị vô hướng
- 1.4 Các thuật ngữ cơ bản của đồ thị có hướng
- 1.5 **Biểu diễn đồ thị**
- 1.6 Liên thông trong đồ thị vô hướng
- 1.7 Liên thông trong đồ thị có hướng
- 1.8 Đangkan cầu đồ thị
- 1.9 Một số đồ thị đặc biệt

Ma trận kề của đồ thị vô hướng

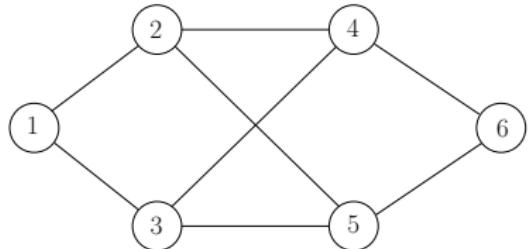
Cho trước một **đồ thị vô hướng** $G = (V, E)$ với $V = \{v_1, \dots, v_n\}$.

Định nghĩa

Ma trận kề của G là ma trận $C = (c_{ij}) \in \{0, 1\}^{n \times n}$ xác định bởi

$$\begin{aligned} c_{ij} = 1 & \quad \text{nếu } \{v_i, v_j\} \in E, \\ c_{ij} = 0 & \quad \text{nếu } \{v_i, v_j\} \notin E. \end{aligned}$$

Chú ý: ma trận kề của đồ thị vô hướng là đối xứng.



$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Ma trận kề của đồ thị vô hướng

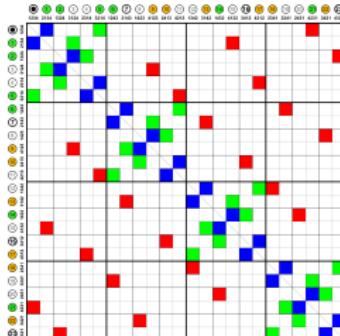
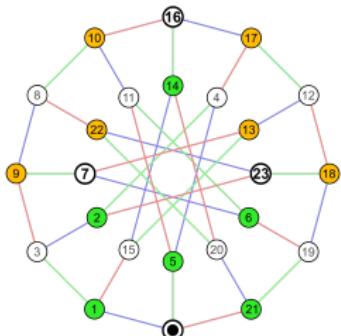
Cho trước một **đồ thị vô hướng** $G = (V, E)$ với $V = \{v_1, \dots, v_n\}$.

Định nghĩa

Ma trận kề của G là ma trận $C = (c_{ij}) \in \{0, 1\}^{n \times n}$ xác định bởi

$$\begin{aligned} c_{ij} = 1 & \quad \text{nếu } \{v_i, v_j\} \in E, \\ c_{ij} = 0 & \quad \text{nếu } \{v_i, v_j\} \notin E. \end{aligned}$$

Chú ý: ma trận kề của đồ thị vô hướng là đối xứng.



Ma trận kề của đồ thị có hướng

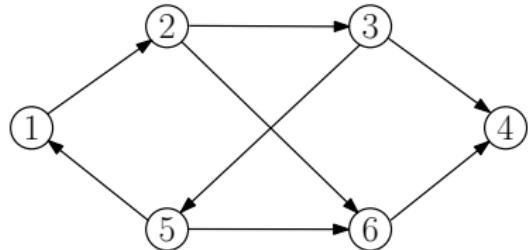
Cho trước một **đồ thị có hướng** $G = (N, A)$ với $N = \{u_1, \dots, u_n\}$.

Định nghĩa

Ma trận kề của G là ma trận $C = (c_{ij}) \in \{0, 1\}^{n \times n}$ xác định bởi

$$\begin{aligned} c_{ij} = 1 & \quad \text{nếu } (u_i, u_j) \in A, \\ c_{ij} = 0 & \quad \text{nếu } (u_i, u_j) \notin A. \end{aligned}$$

Chú ý: ma trận kề của đồ thị có hướng có thể không đối xứng.



$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Ma trận kề của đồ thị có hướng

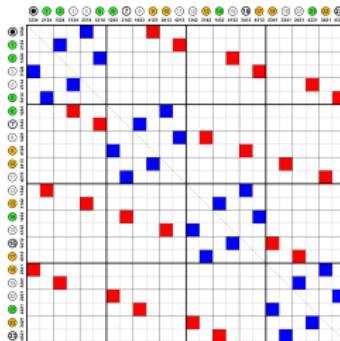
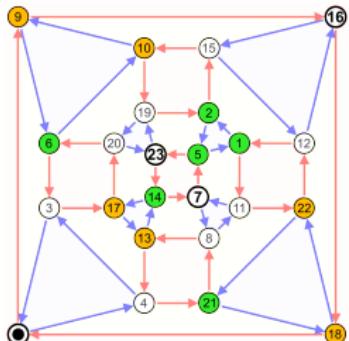
Cho trước một **đồ thị có hướng** $G = (N, A)$ với $N = \{u_1, \dots, u_n\}$.

Định nghĩa

Ma trận kề của G là ma trận $C = (c_{ij}) \in \{0, 1\}^{n \times n}$ xác định bởi

$$\begin{aligned} c_{ij} = 1 & \quad \text{nếu } (u_i, u_j) \in A, \\ c_{ij} = 0 & \quad \text{nếu } (u_i, u_j) \notin A. \end{aligned}$$

Chú ý: ma trận kề của đồ thị có hướng có thể không đối xứng.



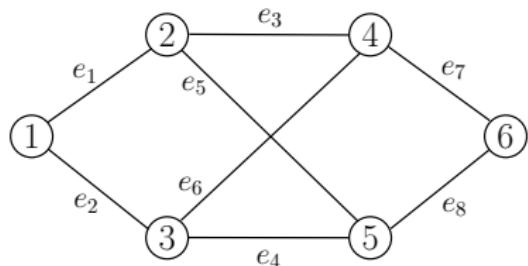
Ma trận liên thuộc của đồ thị vô hướng

Cho trước một **đồ thị vô hướng** $G = (V, E)$ với $V = \{v_1, \dots, v_n\}$.

Định nghĩa

Ma trận liên thuộc của G là ma trận $C = (c_{ij}) \in \{0, 1\}^{n \times m}$ xác định bởi

$$\begin{aligned} c_{ij} = 1 & \quad \text{nếu đỉnh } v_i \text{ liên thuộc với cạnh } e_j, \\ c_{ij} = 0 & \quad \text{nếu ngược lại.} \end{aligned}$$



$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Ma trận liên thuộc của đồ thị có hướng

Cho trước một **đồ thị có hướng** $G = (N, A)$
 với $N = \{u_1, \dots, u_n\}$, $A = \{a_1, \dots, a_m\}$.

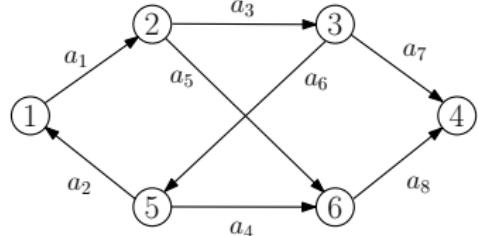
Định nghĩa

Ma trận liên thuộc của G là ma trận $C = (c_{ij}) \in \{0, 1\}^{n \times m}$ xác định bởi

$c_{ij} = 1$ nếu u_i là đỉnh bắt đầu của cung a_j ,

$c_{ij} = -1$ nếu u_i là đỉnh kết thúc của cung a_j ,

$c_{ij} = 0$ nếu u_i không liên thuộc với cung a_j .



$$\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \end{bmatrix}$$

Bình luận

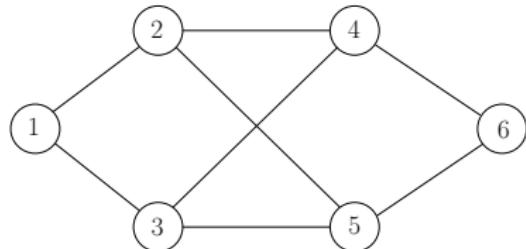
- Lợi ích của ma trận kề và ma trận liên thuộc:
 - Dễ dàng nhận biết các đỉnh kề nhau
 - Dễ dàng tính bậc của các đỉnh
- Hạn chế của ma trận kề và ma trận liên thuộc:
 - Thường là ma trận thưa
 - Sử dụng dung lượng bộ nhớ cố định (thường là lớn) khi lưu trữ trên máy tính

Danh sách cạnh của đồ thị vô hướng

Cho trước một **đồ thị vô hướng** $G = (V, E)$.

Định nghĩa

Danh sách cạnh của G là một mảng với các phần tử là các cạnh của G .



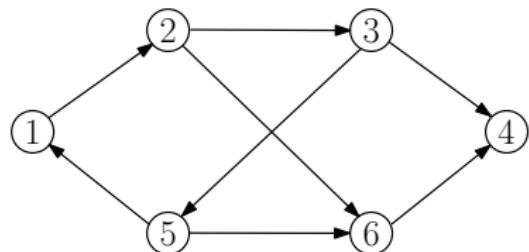
1	2
1	3
2	4
2	5
3	4
3	5
4	6
5	6

Danh sách cung của đồ thị có hướng

Cho trước một đồ thị có hướng $G = (N, A)$.

Định nghĩa

Danh sách cung của G là một mảng với các phần tử là các cung của G .



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

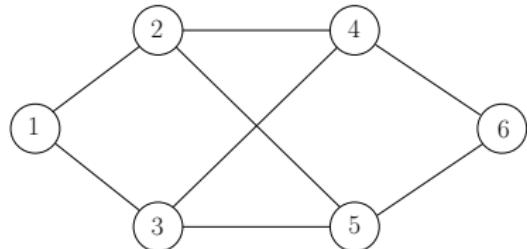
Danh sách kề của đồ thị vô hướng

Cho trước một **đồ thị vô hướng** $G = (V, E)$.

Định nghĩa

Danh sách kề của G là một mảng các danh mục, trong đó mỗi danh mục có dạng

$$v_i \in V, \quad \{u \in V \mid u \text{ là một hàng xóm của } v_i\}.$$



- 1, {2, 3}
- 2, {1, 4, 5}
- 3, {1, 4, 5}
- 4, {2, 3, 6}
- 5, {2, 3, 6}
- 6, {4, 5}

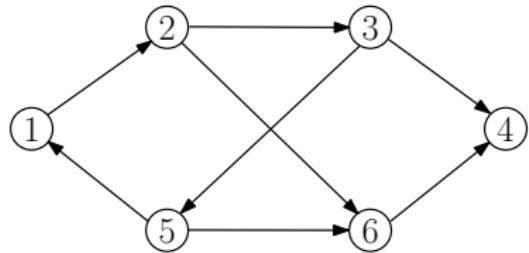
Danh sách kề của đồ thị có hướng

Cho trước một **đồ thị có hướng** $G = (N, A)$.

Định nghĩa

Danh sách kề của G là một mảng các danh mục, trong đó mỗi danh mục có dạng

$$u_i \in N, \quad \{u \in N \mid (u_i, u) \in A\}.$$



- 1, $\{2\}$
- 2, $\{3, 6\}$
- 3, $\{4, 5\}$
- 4, \emptyset
- 5, $\{1, 6\}$
- 6, $\{4\}$

Bình luận

- Lợi ích của danh sách cạnh, danh sách cung, danh sách kề:
 - Dễ dàng thêm, bớt cạnh / cung
 - Tiết kiệm bộ nhớ khi lưu trữ trên máy tính
- Hạn chế của danh sách cạnh, danh sách cung, danh sách kề:
 - Không thuận tiện khi tính toán bậc của các đỉnh
 - Không dễ để nhận biết các đỉnh kề nhau

Nội dung

- 1.1 Khái niệm đồ thị
- 1.2 Tính phổ biến của đồ thị
- 1.3 Các thuật ngữ cơ bản của đồ thị vô hướng
- 1.4 Các thuật ngữ cơ bản của đồ thị có hướng
- 1.5 Biểu diễn đồ thị
- 1.6 Liên thông trong đồ thị vô hướng
- 1.7 Liên thông trong đồ thị có hướng
- 1.8 Đangkan cầu đồ thị
- 1.9 Một số đồ thị đặc biệt

Đường đi trong đồ thị vô hướng

Cho trước một **đồ thị vô hướng** $G = (V, E)$.

Định nghĩa

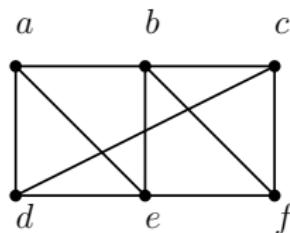
- Một *đường đi* có độ dài n từ $u \in V$ tới $v \in V$ là một dãy các đỉnh

$$x_0 \equiv u, x_1, \dots, x_n \equiv v \in V$$

thỏa mãn $\{x_{i-1}, x_i\} \in E$ với mọi $i = 1, \dots, n$.

- Một *chu trình* là một đường đi có độ dài dương, bắt đầu và kết thúc tại cùng một đỉnh.
- Một đường đi hoặc chu trình là *đơn* nếu nó đi qua mỗi cạnh nhiều nhất một lần.

Minh họa các khái niệm trên
trong ví dụ này:



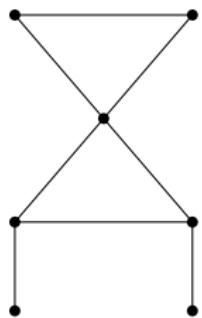
Liên thông trong đồ thị vô hướng

Định nghĩa

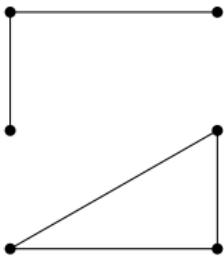
Một đồ thị vô hướng được G gọi là *liên thông* nếu có một đường đi nối hai đỉnh khác nhau bất kỳ trong đồ thị. Ngược lại, ta gọi G là *không liên thông*.

Định lý

Luôn có đường đi **đơn** nối hai đỉnh khác nhau bất kỳ trong một đồ thị vô hướng liên thông.



G_1



G_2

Thành phần liên thông của đồ thị vô hướng

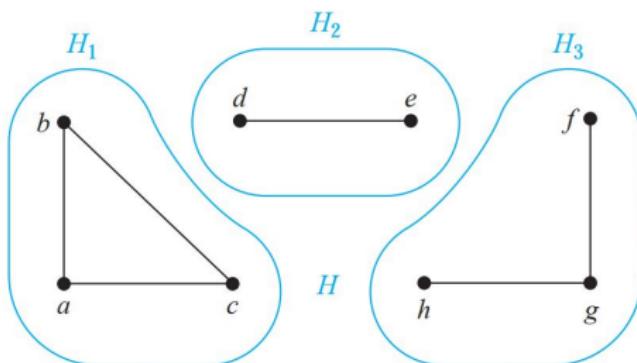
Cho trước một **đồ thị vô hướng** $G = (V, E)$.

Định nghĩa

Một *thành phần liên thông* của G là

- một đồ thị con liên thông của G , và
- không là đồ thị con thực sự của một đồ thị con liên thông của G .

Nôm na: một thành phần liên thông là một đồ thị con liên thông cực đại.

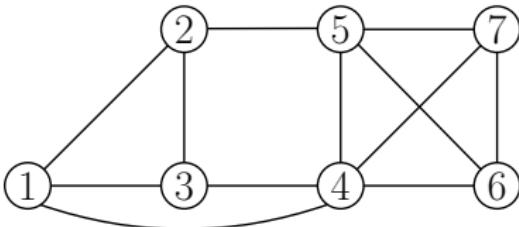
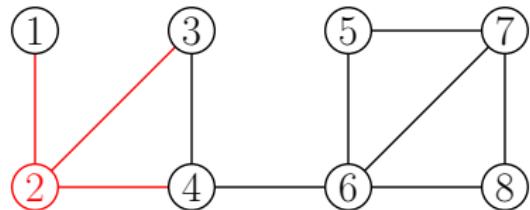


Liên thông đỉnh

Cho trước một đồ thị vô hướng $G = (V, E)$.

Định nghĩa

- Đỉnh $v \in V$ được gọi là *đỉnh cắt* hay *đỉnh khớp* của G nếu sau khi xóa v cùng các cạnh liên thuộc với nó ta thu được một đồ thị con với nhiều thành phần liên thông hơn.
- G được gọi là *không phân tách* được nếu nó liên thông và không có đỉnh cắt.

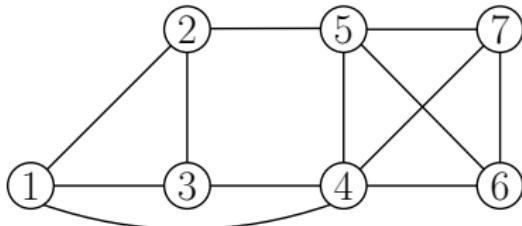
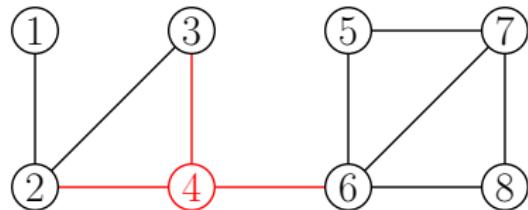


Liên thông đỉnh

Cho trước một đồ thị vô hướng $G = (V, E)$.

Định nghĩa

- Đỉnh $v \in V$ được gọi là *đỉnh cắt* hay *đỉnh khớp* của G nếu sau khi xóa v cùng các cạnh liên thuộc với nó ta thu được một đồ thị con với nhiều thành phần liên thông hơn.
- G được gọi là *không phân tách* được nếu nó liên thông và không có đỉnh cắt.

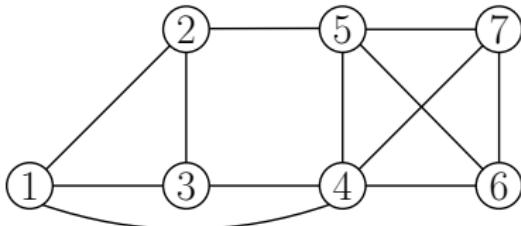
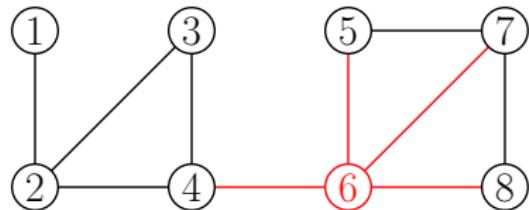


Liên thông đỉnh

Cho trước một đồ thị vô hướng $G = (V, E)$.

Định nghĩa

- Đỉnh $v \in V$ được gọi là *đỉnh cắt* hay *đỉnh khớp* của G nếu sau khi xóa v cùng các cạnh liên thuộc với nó ta thu được một đồ thị con với nhiều thành phần liên thông hơn.
- G được gọi là *không phân tách* được nếu nó liên thông và không có đỉnh cắt.

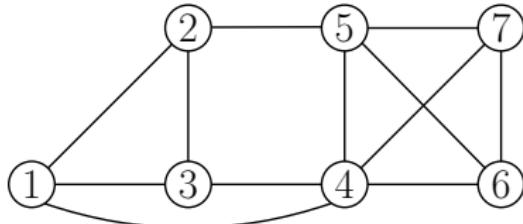
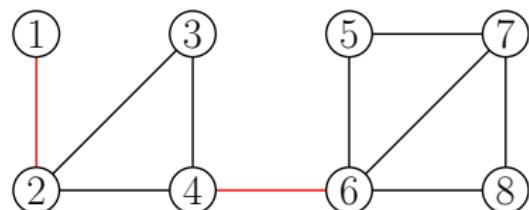


Liên thông cạnh

Cho trước một đồ thị vô hướng $G = (V, E)$.

Định nghĩa

Một cạnh $e \in E$ được gọi là *cạnh cắt* hay *cầu* của G nếu sau khi xóa cạnh e ta thu được một đồ thị con với nhiều thành phần liên thông hơn.



Nội dung

- 1.1 Khái niệm đồ thị
- 1.2 Tính phổ biến của đồ thị
- 1.3 Các thuật ngữ cơ bản của đồ thị vô hướng
- 1.4 Các thuật ngữ cơ bản của đồ thị có hướng
- 1.5 Biểu diễn đồ thị
- 1.6 Liên thông trong đồ thị vô hướng
- 1.7 **Liên thông trong đồ thị có hướng**
- 1.8 Đẳng cấu đồ thị
- 1.9 Một số đồ thị đặc biệt

Liên thông trong đồ thị có hướng

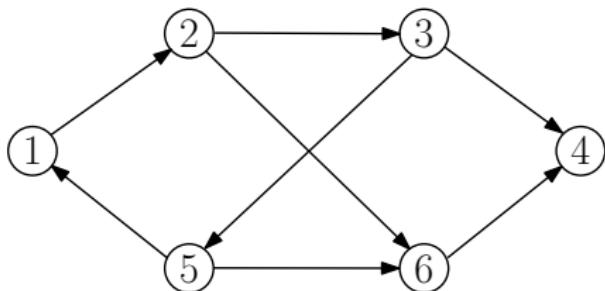
Cho trước một **đồ thị có hướng** $G = (N, A)$.

Định nghĩa

Xây dựng đồ thị vô hướng $G^* = (V, E)$ với $V \equiv N$ và

$$E = \{\{u, v\} \mid (u, v) \in A\}.$$

Đồ thị G^* được gọi là *đồ thị vô hướng nền* của G .



Liên thông trong đồ thị có hướng

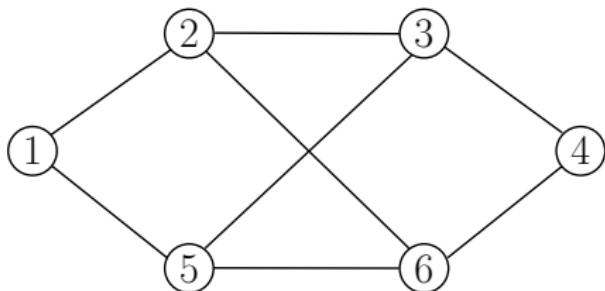
Cho trước một **đồ thị có hướng** $G = (N, A)$.

Định nghĩa

Xây dựng đồ thị vô hướng $G^* = (V, E)$ với $V \equiv N$ và

$$E = \{\{u, v\} \mid (u, v) \in A\}.$$

Đồ thị G^* được gọi là *đồ thị vô hướng nền* của G .



Đường đi trong đồ thị có hướng

Cho trước một **đồ thị có hướng** $G = (N, A)$.

Định nghĩa

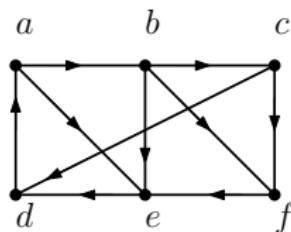
- Một *đường đi* có độ dài n từ $u \in N$ tới $v \in N$ là một dãy các đỉnh

$$x_0 \equiv u, x_1, \dots, x_n \equiv v \in N$$

thỏa mãn $(x_{i-1}, x_i) \in A$ với mọi $i = 1, \dots, n$.

- Một *chu trình* là một đường đi có độ dài dương, bắt đầu và kết thúc tại cùng một đỉnh.
- Một đường đi hoặc chu trình là *đơn* nếu nó đi qua mỗi cung không quá một lần.

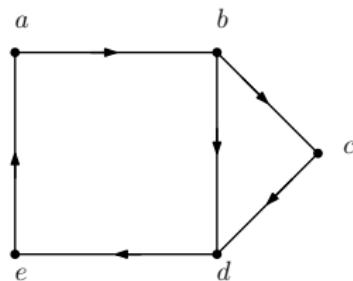
Minh họa các khái niệm trên
trong ví dụ này:



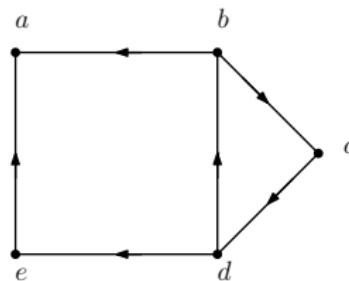
Liên thông trong đồ thị có hướng

Định nghĩa

- Một đồ thị có hướng G được gọi là *liên thông yếu* nếu có đường đi giữa hai đỉnh khác nhau bất kỳ trong đồ thị vô hướng nền của G .
- Một đồ thị có hướng được gọi là *liên thông mạnh* nếu với hai đỉnh khác nhau a, b bất kỳ ta luôn có
 - một đường đi từ a tới b , và
 - một đường đi từ b tới a .



Đồ thị có hướng liên thông mạnh



Đồ thị có hướng liên thông yếu

Nội dung

- 1.1 Khái niệm đồ thị
- 1.2 Tính phổ biến của đồ thị
- 1.3 Các thuật ngữ cơ bản của đồ thị vô hướng
- 1.4 Các thuật ngữ cơ bản của đồ thị có hướng
- 1.5 Biểu diễn đồ thị
- 1.6 Liên thông trong đồ thị vô hướng
- 1.7 Liên thông trong đồ thị có hướng
- 1.8 Đangkan cầu đồ thị
- 1.9 Một số đồ thị đặc biệt

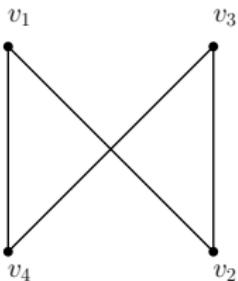
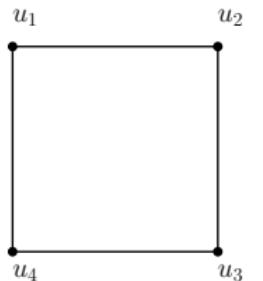
Đẳng cấu³ đồ thị

Định nghĩa

Hai đồ thị vô hướng $G_1 = (V_1, E_1)$ và $G_2 = (V_2, E_2)$ được gọi là *đẳng cấu* với nhau nếu có một song ánh $f: V_1 \rightarrow V_2$ thỏa mãn

$$\{f(u), f(v)\} \in E_2 \Leftrightarrow \{u, v\} \in E_1 \quad \forall u, v \in V_1.$$

Định nghĩa khái niệm đẳng cấu giữa hai đồ thị có hướng?



$$f(u_i) = v_i \quad (i=1, \dots, 4)$$

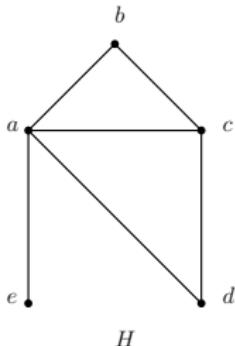
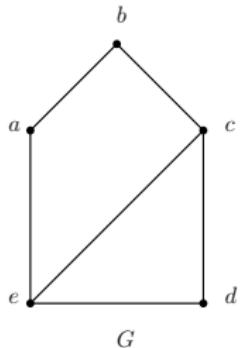
³ *Isomorphism*: thuật ngữ này có nguồn gốc từ tiếng Hy Lạp (*isos* nghĩa là “bằng nhau” và *morphe* nghĩa là “dạng”)

Đẳng cấu đồ thị

Chú ý:

- Số đỉnh
- Số cạnh
- Số đỉnh có cùng bậc

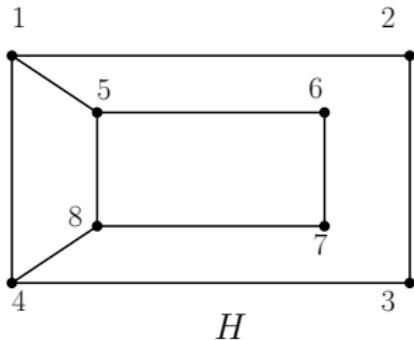
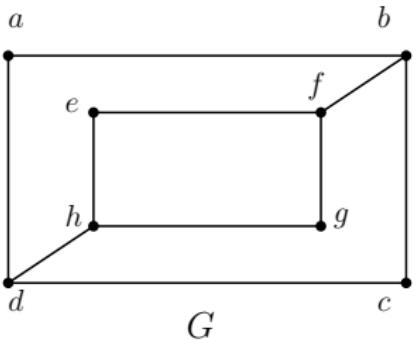
là bất biến qua đẳng cấu đồ thị.



G không đẳng cấu với H vì

- H có một đỉnh bậc 1
- G không có đỉnh bậc 1

Đẳng cấu đồ thị



G không đẳng cấu với H vì

- đỉnh a có bậc 2,
vậy nó phải tương ứng với một trong các đỉnh 2, 3, 6, 7;
- mỗi đỉnh 2, 3, 6, 7 kề với một đỉnh bậc 2;
- nhưng đỉnh a chỉ kề với các đỉnh bậc 3.

Nội dung

- 1.1 Khái niệm đồ thị
- 1.2 Tính phổ biến của đồ thị
- 1.3 Các thuật ngữ cơ bản của đồ thị vô hướng
- 1.4 Các thuật ngữ cơ bản của đồ thị có hướng
- 1.5 Biểu diễn đồ thị
- 1.6 Liên thông trong đồ thị vô hướng
- 1.7 Liên thông trong đồ thị có hướng
- 1.8 Đangkan cầu đồ thị
- 1.9 Một số đồ thị đặc biệt

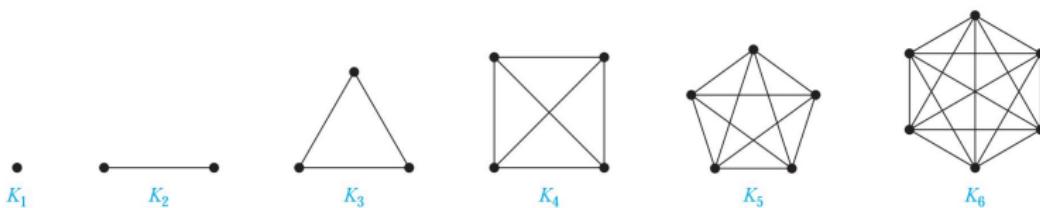
Đồ thị đầy đủ

Định nghĩa

Một *đồ thị đầy đủ* trên n đỉnh, ký hiệu K_n , là một đồ thị vô hướng chứa chính xác một cạnh nối bất kỳ cặp đỉnh khác nhau nào.

Notes:

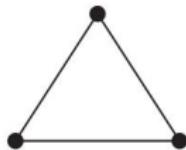
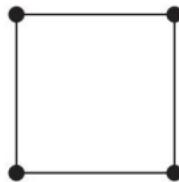
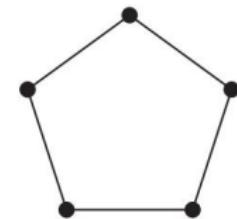
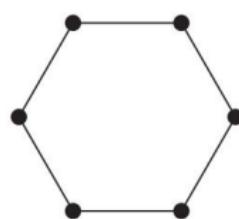
- K_n có $\frac{n(n-1)}{2}$ cạnh
- K_n không phân tách được



Đồ thị vòng

Định nghĩa

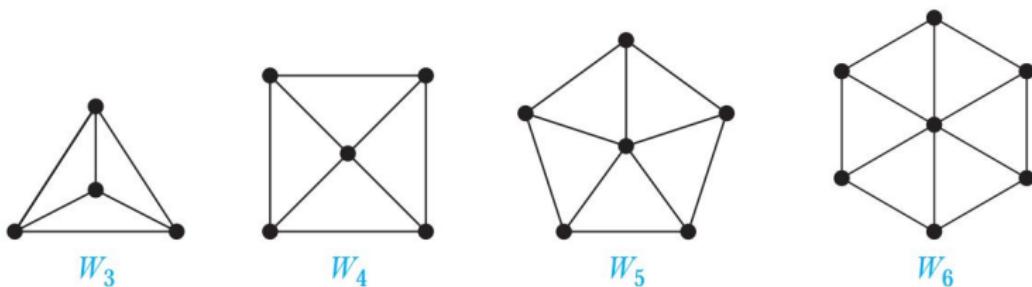
Một đồ thị vòng C_n trên $n \geq 3$ đỉnh v_1, v_2, \dots, v_n là một đồ thị vô hướng với các cạnh $\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$.

 C_3  C_4  C_5  C_6

Đồ thị bánh xe

Định nghĩa

Một đồ thị bánh xe W_n trên $n \geq 3$ đỉnh là một đồ thị vô hướng được tạo thành bằng cách thêm một đỉnh vào đồ thị vòng C_n và nối đỉnh đó với từng đỉnh của đồ thị vòng C_n bởi một cạnh mới.

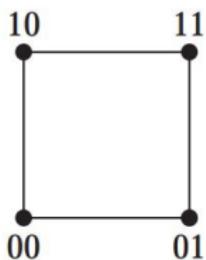
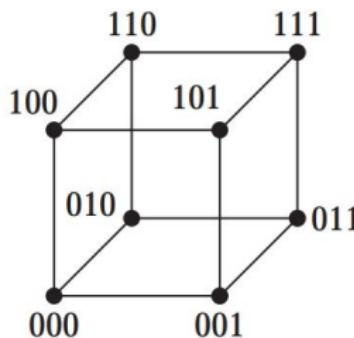


Đồ thị lập phương

Định nghĩa

Một đồ thị lập phương Q_n là một đồ thị vô hướng được tạo thành bởi

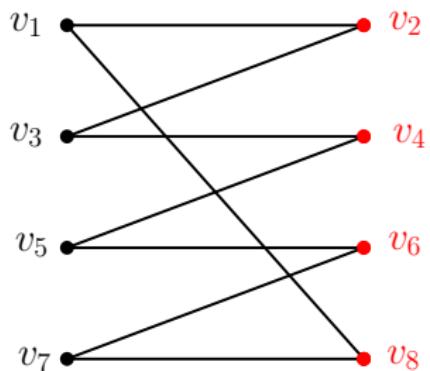
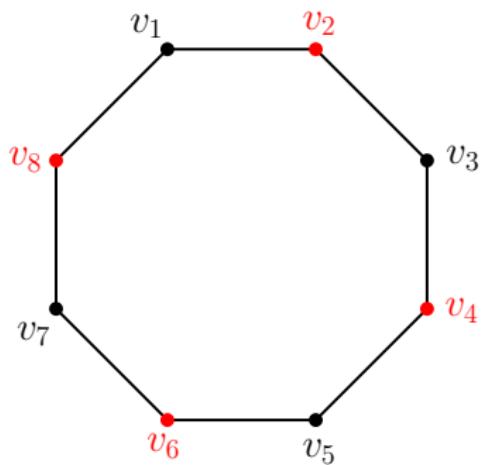
- 2^n đỉnh tương ứng với các xâu nhị phân độ dài n ,
- hai đỉnh được nối với nhau bởi một cạnh khi và chỉ khi các xâu nhị phân tương ứng với hai đỉnh đó khác nhau tại chính xác một vị trí.

 Q_1  Q_2  Q_3

Đồ thị hai phía

Định nghĩa

Một đồ thị vô hướng G được gọi là *đồ thị hai phía* nếu tập đỉnh của G có thể phân hoạch thành hai tập con rời nhau V_1, V_2 sao cho mỗi cạnh của G đều liên thuộc với một đỉnh trong V_1 và một đỉnh trong V_2 .

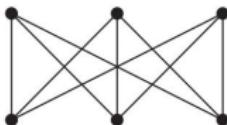
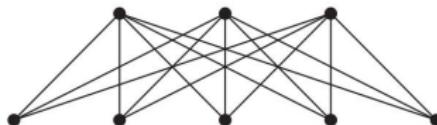
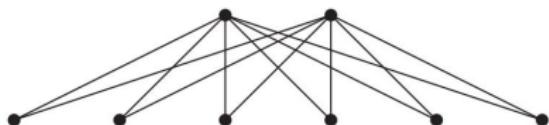


Đồ thị hai phía đầy đủ

Định nghĩa

Một *đồ thị hai phía đầy đủ* $K_{m,n}$ là một đồ thị hai phía trên hai tập đỉnh rời nhau V_1, V_2 sao cho

- $|V_1| = m, |V_2| = n$, và
- mỗi đỉnh trong V_1 đều kề với một đỉnh trong V_2 .

 $K_{2,3}$  $K_{3,3}$  $K_{3,5}$  $K_{2,6}$

Mục lục

1 Một số khái niệm cơ bản

2 Duyệt đồ thị

3 Đồ thị Euler

4 Đồ thị Hamilton

5 Đồ thị phẳng

6 Cây và cây khung

7 Đường đi ngắn nhất

Nội dung

- 2.1 Tìm kiếm theo chiều sâu
- 2.2 Tìm kiếm theo chiều rộng
- 2.3 Áp dụng: tìm đường đi trong đồ thị
- 2.4 Áp dụng: kiểm tra tính liên thông

Tìm kiếm theo chiều sâu

Về thuật toán tìm kiếm theo chiều sâu trên đồ thị (DFS):

- Một thuật toán duyệt qua mỗi đỉnh của đồ thị một lượt
- Độ phức tạp tính toán $O(\text{cấp đồ thị} + \text{cỡ đồ thị})$
- Sử dụng lược đồ quay lui

```
procedure DFS(v)
    for mỗi đỉnh chưa duyệt w kề với v do
        DFS(w);
    end for
```

Ghi chú:

- Trong phần này ta chỉ xét đồ thị vô hướng
- Với một số thay đổi, có thể áp dụng DFS cho đồ thị có hướng
- Tập hợp đỉnh phải được sắp xếp trước

Giả mã của DFS

Thuật toán DFS trên đồ thị $G = (V, E)$

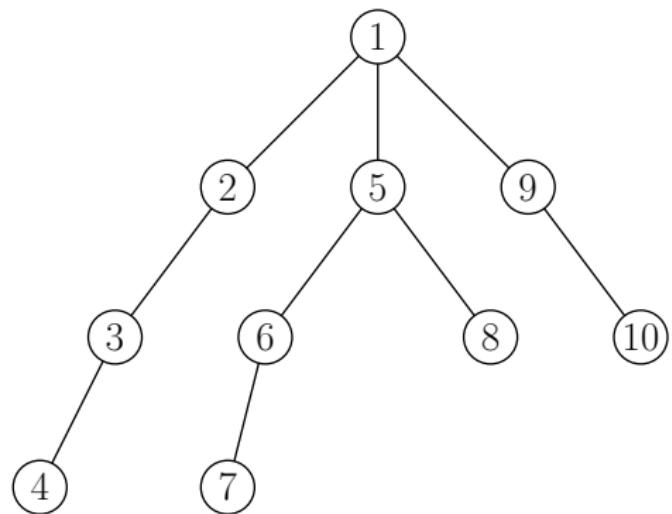
- 1: $\text{visited}[v] := \text{false}$ với mọi $v \in V$;
- 2: **for** mỗi $v \in V$ **do**
- 3: **if** $\text{visited}[v] = \text{false}$ **then**
- 4: $\text{DFS}(v)$;
- 5: **end if**
- 6: **end for**

Thuật toán $\text{DFS}(v)$

- 1: In thông tin của đỉnh v ;
- 2: $\text{visited}[v] := \text{true}$;
- 3: **for** mỗi hàng xóm u của v **do**
- 4: **if** $\text{visited}[u] = \text{false}$ **then**
- 5: $\text{DFS}(u)$;
- 6: **end if**
- 7: **end for**

Ví dụ cho DFS

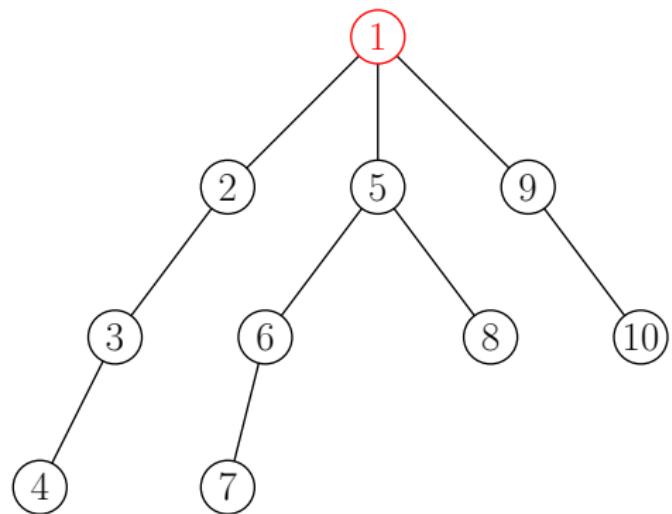
Thứ tự duyệt các đỉnh
theo DFS



- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

Ví dụ cho DFS

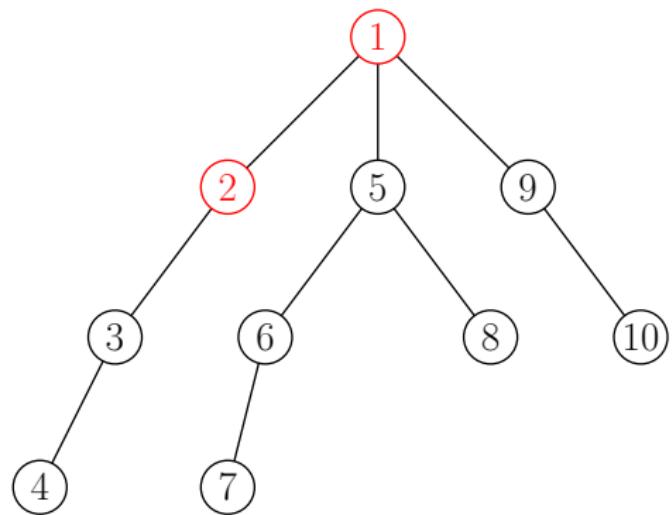
Thứ tự duyệt các đỉnh
theo DFS



- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

Ví dụ cho DFS

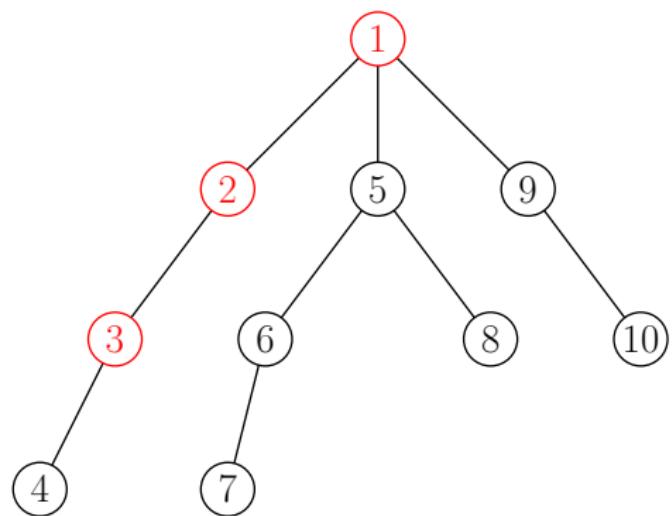
Thứ tự duyệt các đỉnh
theo DFS



- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

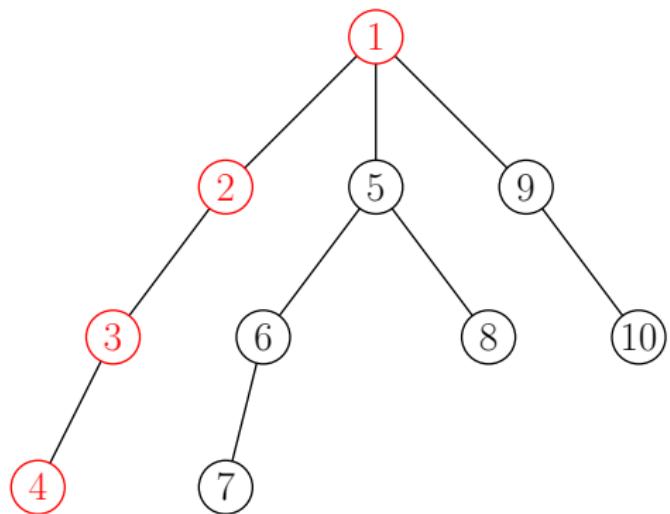
Ví dụ cho DFS

Thứ tự duyệt các đỉnh
theo DFS



- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

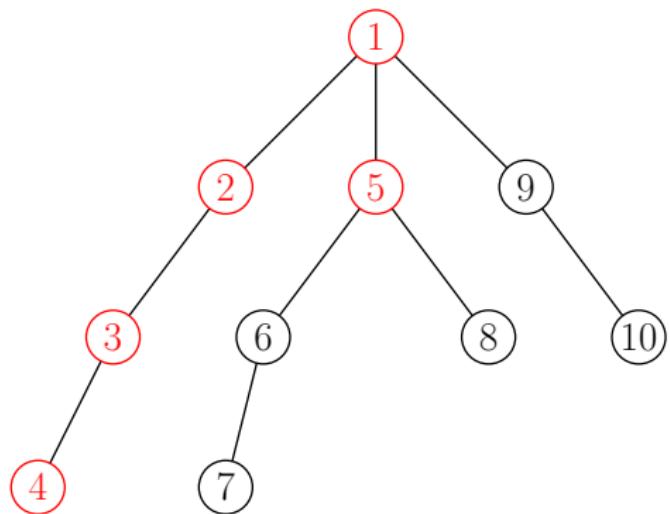
Ví dụ cho DFS



Thứ tự duyệt các đỉnh
theo DFS

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

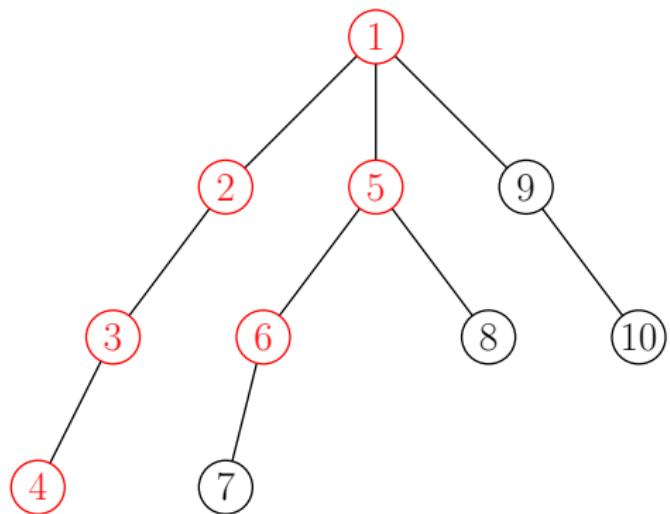
Ví dụ cho DFS



Thứ tự duyệt các đỉnh
theo DFS

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

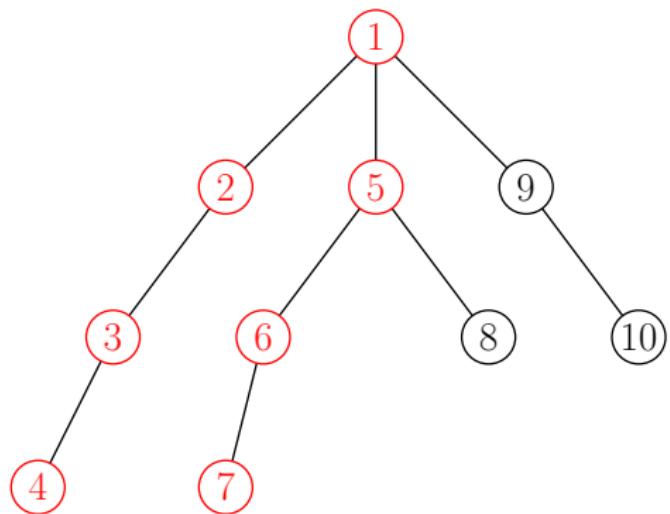
Ví dụ cho DFS



Thứ tự duyệt các đỉnh
theo DFS

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

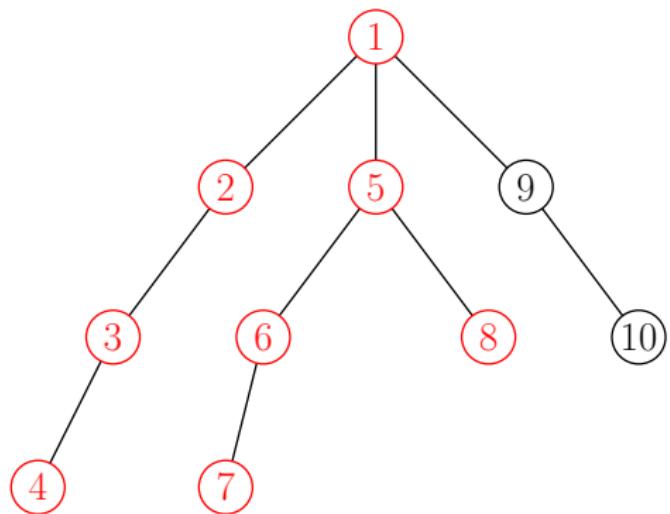
Ví dụ cho DFS



Thứ tự duyệt các đỉnh
theo DFS

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

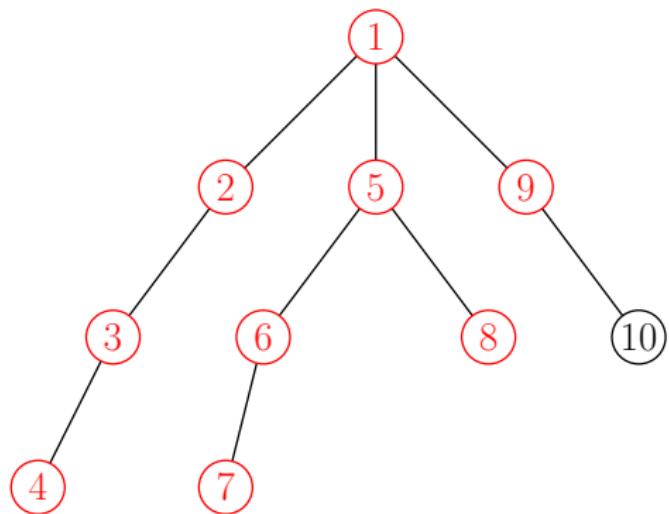
Ví dụ cho DFS



Thứ tự duyệt các đỉnh
theo DFS

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

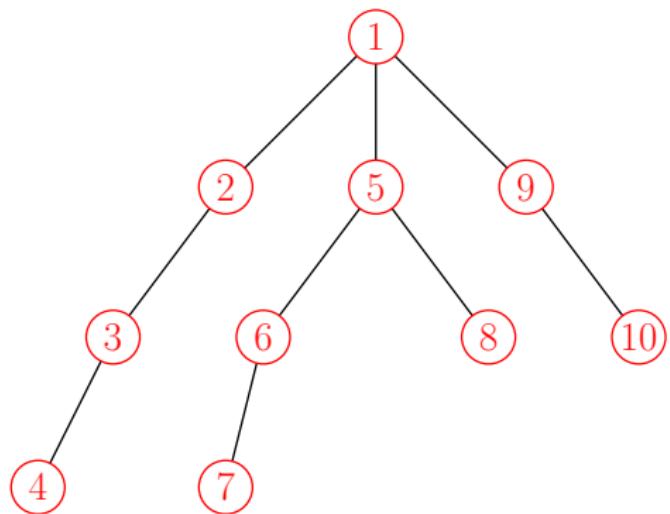
Ví dụ cho DFS



Thứ tự duyệt các đỉnh
theo DFS

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

Ví dụ cho DFS



Thứ tự duyệt các đỉnh
theo DFS

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

Nội dung

- 2.1 Tìm kiếm theo chiều sâu
- 2.2 Tìm kiếm theo chiều rộng
- 2.3 Áp dụng: tìm đường đi trong đồ thị
- 2.4 Áp dụng: kiểm tra tính liên thông

Tìm kiếm theo chiều rộng

Về thuật toán tìm kiếm theo chiều rộng (BFS) trên đồ thị [liên thông](#):

- Một thuật toán duyệt qua mỗi đỉnh của đồ thị một lượt
- Độ phức tạp tính toán $O(\text{cấp đồ thị} + \text{cỡ đồ thị})$
- Lược đồ:

procedure BFS

Tạo một hàng đợi Q để chứa các đỉnh chưa xử lý;

Đẩy một đỉnh của đồ thị vào Q ;

while $Q \neq \emptyset$ **do**

Duyệt đỉnh chưa xử lý đầu tiên trong hàng đợi Q ,

đánh dấu đỉnh đó đã được xử lý;

Đẩy các hàng xóm của đỉnh vừa được xử lý vào Q ;

end while

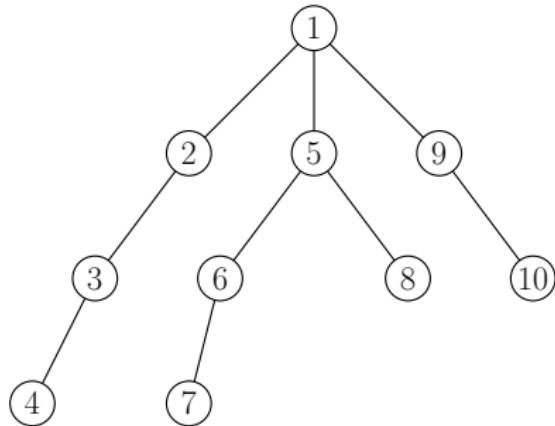
Ghi chú:

- Trong phần này ta chỉ xét đồ thị vô hướng
- Với một số thay đổi, có thể áp dụng DFS cho đồ thị có hướng
- Tập hợp đỉnh phải được sắp xếp trước

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



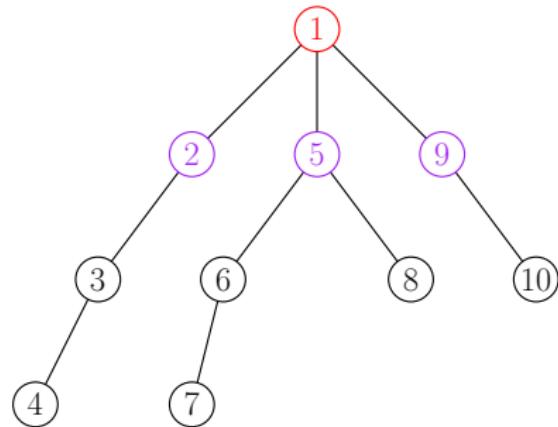
Mảng Q từ **first** đến **last**
sau mỗi bước:

- | | |
|------------------|------------------|
| • 1, 2, 5, 9 | • 6, 8, 10, 4, 7 |
| • 2, 5, 9, 3 | • 8, 10, 4, 7 |
| • 5, 9, 3, 6, 8 | • 10, 4, 7 |
| • 9, 3, 6, 8, 10 | • 4, 7 |
| • 3, 6, 8, 10, 4 | • 7 |

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



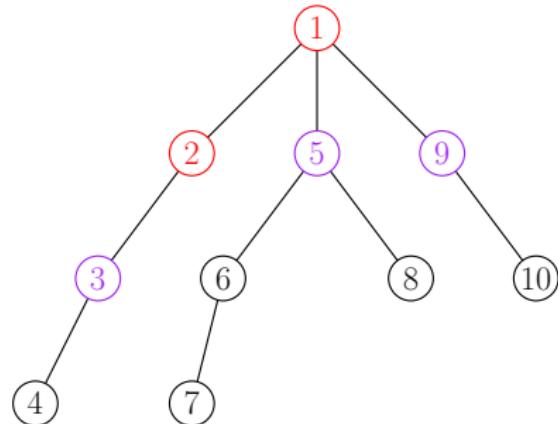
Mảng Q từ **first** đến **last**
sau mỗi bước:

- 1, 2, 5, 9 • 6, 8, 10, 4, 7
- 2, 5, 9, 3 • 8, 10, 4, 7
- 5, 9, 3, 6, 8 • 10, 4, 7
- 9, 3, 6, 8, 10 • 4, 7
- 3, 6, 8, 10, 4 • 7

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



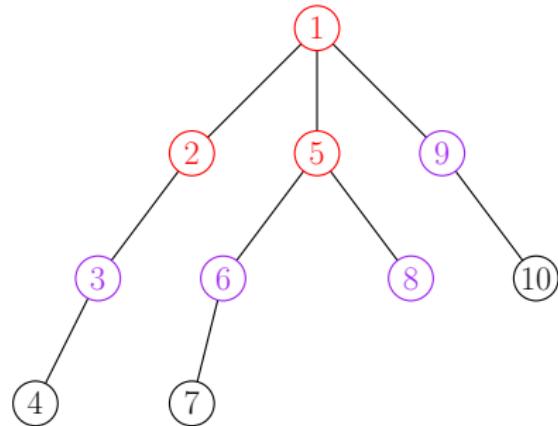
Mảng Q từ **first** đến **last**
sau mỗi bước:

- 1, 2, 5, 9 • 6, 8, 10, 4, 7
- 2, 5, 9, 3 • 8, 10, 4, 7
- 5, 9, 3, 6, 8 • 10, 4, 7
- 9, 3, 6, 8, 10 • 4, 7
- 3, 6, 8, 10, 4 • 7

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



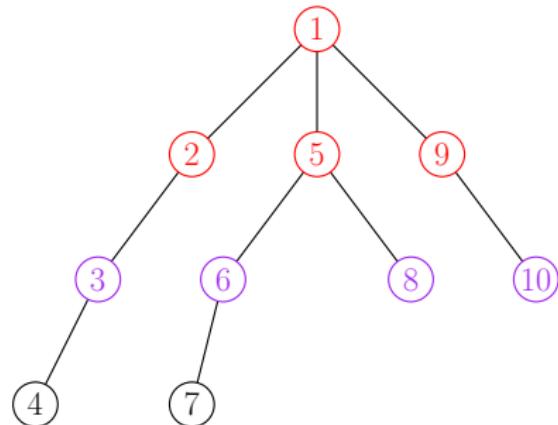
Mảng Q từ **first** đến **last**
sau mỗi bước:

- 1, 2, 5, 9 • 6, 8, 10, 4, 7
- 2, 5, 9, 3 • 8, 10, 4, 7
- 5, 9, 3, 6, 8 • 10, 4, 7
- 9, 3, 6, 8, 10 • 4, 7
- 3, 6, 8, 10, 4 • 7

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



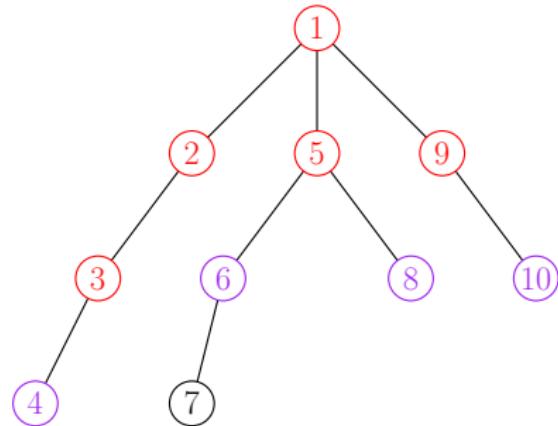
Mảng Q từ **first** đến **last**
sau mỗi bước:

- 1, 2, 5, 9 • 6, 8, 10, 4, 7
- 2, 5, 9, 3 • 8, 10, 4, 7
- 5, 9, 3, 6, 8 • 10, 4, 7
- 9, 3, 6, 8, 10 • 4, 7
- 3, 6, 8, 10, 4 • 7

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



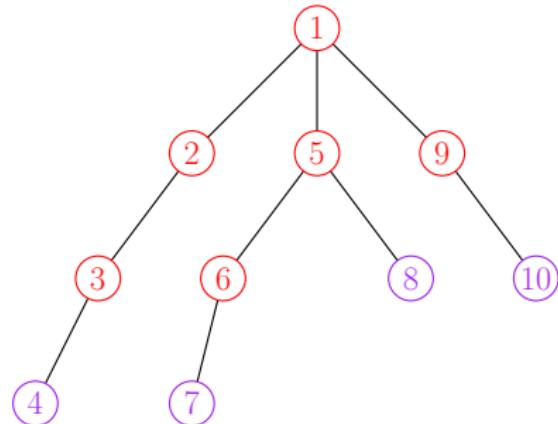
Mảng Q từ **first** đến **last**
sau mỗi bước:

- 1, 2, 5, 9 • 6, 8, 10, 4, 7
- 2, 5, 9, 3 • 8, 10, 4, 7
- 5, 9, 3, 6, 8 • 10, 4, 7
- 9, 3, 6, 8, 10 • 4, 7
- 3, 6, 8, 10, 4 • 7

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



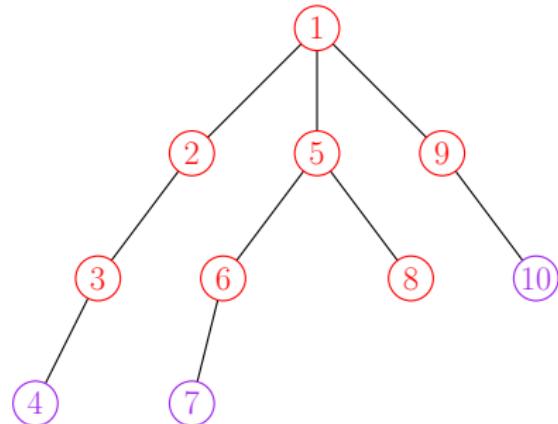
Mảng Q từ **first** đến **last**
sau mỗi bước:

- | | |
|------------------|------------------|
| • 1, 2, 5, 9 | • 6, 8, 10, 4, 7 |
| • 2, 5, 9, 3 | • 8, 10, 4, 7 |
| • 5, 9, 3, 6, 8 | • 10, 4, 7 |
| • 9, 3, 6, 8, 10 | • 4, 7 |
| • 3, 6, 8, 10, 4 | • 7 |

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



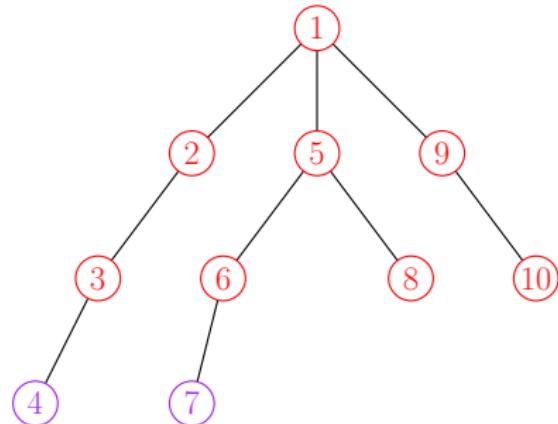
Mảng Q từ **first** đến **last**
sau mỗi bước:

- 1, 2, 5, 9 • 6, 8, 10, 4, 7
- 2, 5, 9, 3 • 8, 10, 4, 7
- 5, 9, 3, 6, 8 • 10, 4, 7
- 9, 3, 6, 8, 10 • 4, 7
- 3, 6, 8, 10, 4 • 7

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



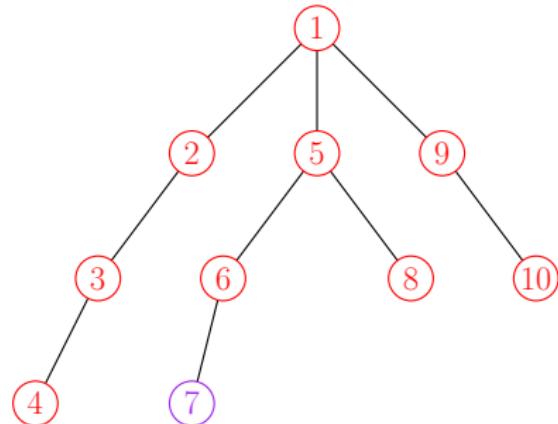
Mảng Q từ **first** đến **last**
sau mỗi bước:

- 1, 2, 5, 9 • 6, 8, 10, 4, 7
- 2, 5, 9, 3 • 8, 10, 4, 7
- 5, 9, 3, 6, 8 • 10, 4, 7
- 9, 3, 6, 8, 10 • 4, 7
- 3, 6, 8, 10, 4 • 7

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



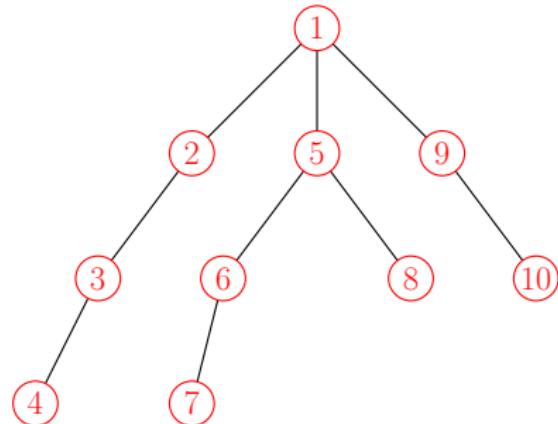
Mảng Q từ **first** đến **last**
sau mỗi bước:

- 1, 2, 5, 9 • 6, 8, 10, 4, 7
- 2, 5, 9, 3 • 8, 10, 4, 7
- 5, 9, 3, 6, 8 • 10, 4, 7
- 9, 3, 6, 8, 10 • 4, 7
- 3, 6, 8, 10, 4 • 7

Giả mã của BFS và ví dụ

Thuật toán BFS trên
đồ thị liên thông $G = (V, E)$

- 1: Tạo mảng Q ;
- 2: $\text{first} := 1$; $\text{last} := 1$;
- 3: $Q[\text{first}] :=$ một đỉnh của G ;
- 4: **while** $\text{first} \leq \text{last}$ **do**
- 5: $v := Q[\text{first}]$;
- 6: In thông tin của v ;
- 7: $\text{first} := \text{first} + 1$;
- 8: **for** mỗi hàng xóm u của v **do**
- 9: $\text{last} := \text{last} + 1$;
- 10: $Q[\text{last}] := u$;
- 11: **end for**
- 12: **end while**



Mảng Q từ **first** đến **last**
sau mỗi bước:

- 1, 2, 5, 9 • 6, 8, 10, 4, 7
- 2, 5, 9, 3 • 8, 10, 4, 7
- 5, 9, 3, 6, 8 • 10, 4, 7
- 9, 3, 6, 8, 10 • 4, 7
- 3, 6, 8, 10, 4 • 7

Nội dung

- 2.1 Tìm kiếm theo chiều sâu
- 2.2 Tìm kiếm theo chiều rộng
- 2.3 Áp dụng: tìm đường đi trong đồ thị
- 2.4 Áp dụng: kiểm tra tính liên thông

Tìm đường đi trong đồ thị

- **Cho trước:**

- một đồ thị vô hướng liên thông $G = (V, E)$
- hai đỉnh phân biệt s, t

- **Bài toán:** tìm một đường đi từ s tới t trong G

- **Ý tưởng:**

- Sử dụng DFS hoặc BFS
- Sử dụng một mảng để lưu các đỉnh đã duyệt và đỉnh trước mỗi đỉnh trên đường đi
- Bắt đầu tìm kiếm từ s , dừng tìm kiếm khi tới t

Tìm đường đi trong đồ thị bằng DFS

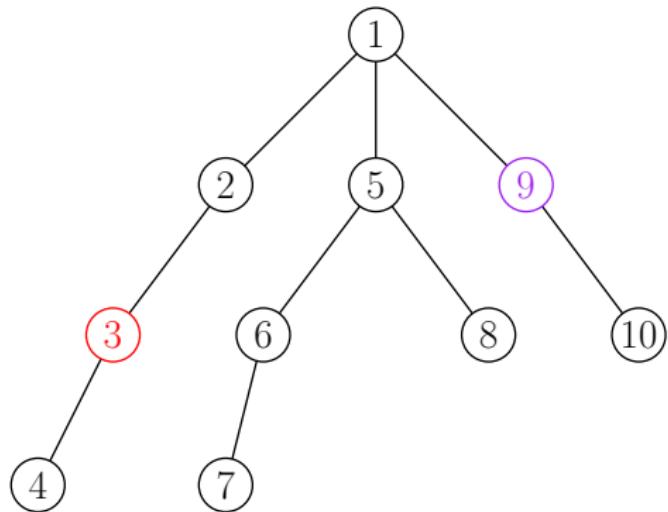
Thuật toán Tìm một đường đi từ s tới t bằng DFS

- 1: $\text{previous}[v] := \emptyset$ với mọi $v \in V$;
 - 2: $\text{previous}[s] := s$;
 - 3: $\text{found} := \text{false}$;
 - 4: $\text{DFS}(s)$;
 - 5: **if** $\text{found} = \text{true}$ **then** in đường đi (sử dụng mảng previous); **end if**
-

Thuật toán $\text{DFS}(v)$

- 1: **for** mỗi hàng xóm u của v **do**
 - 2: **if** $\text{previous}[u] = \emptyset$ **then**
 - 3: $\text{previous}[u] := v$;
 - 4: **if** $u = t$ **then** $\text{found} := \text{true}$; exit; **end if**
 - 5: $\text{DFS}(u)$;
 - 6: **end if**
 - 7: **end for**
-

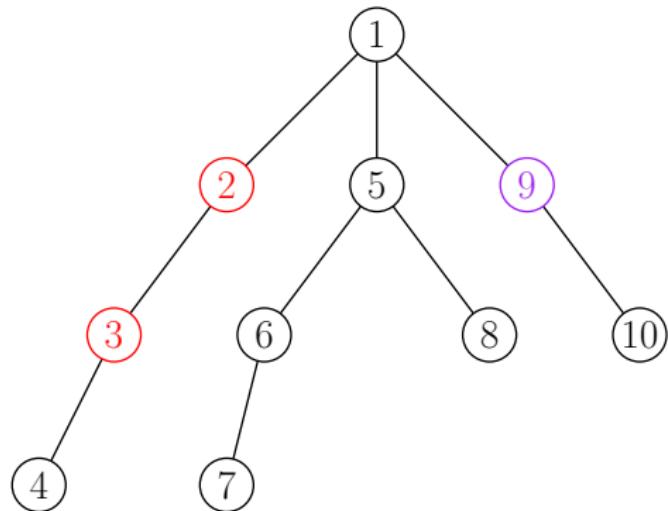
Tìm đường đi trong đồ thị bằng DFS: ví dụ



Tìm một đường
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3

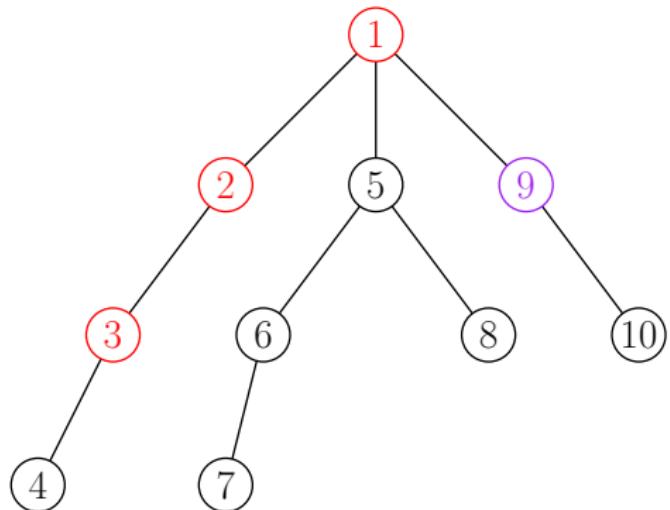
Tìm đường đi trong đồ thị bằng DFS: ví dụ



Tìm một đường
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3

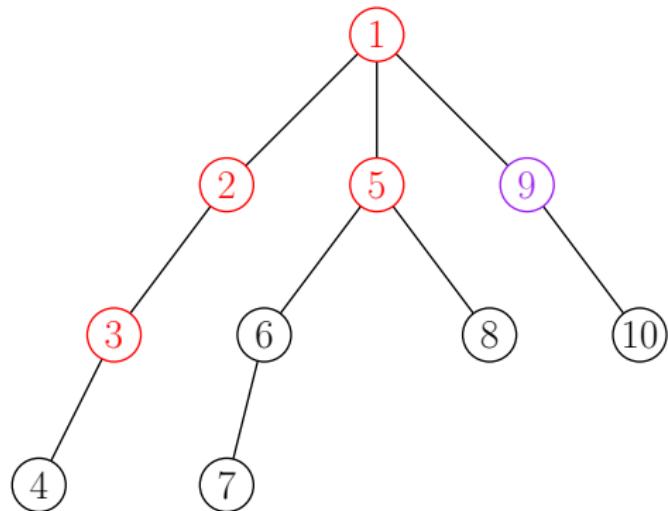
Tìm đường đi trong đồ thị bằng DFS: ví dụ



Tìm một đường
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	1	2

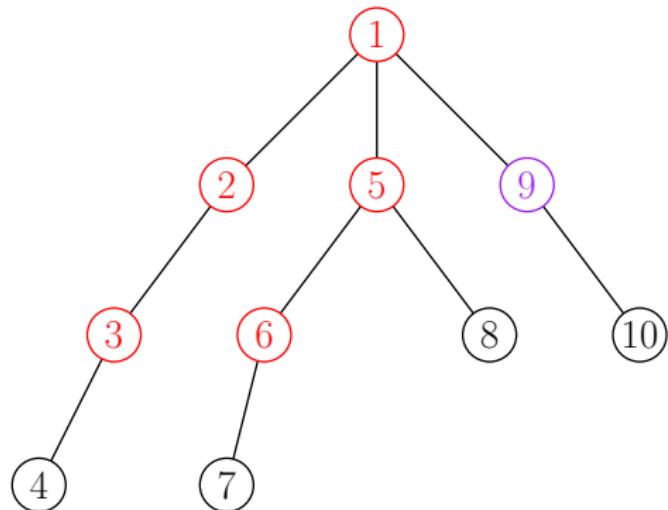
Tìm đường đi trong đồ thị bằng DFS: ví dụ



Tìm một đường
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	1	2
4	5	1

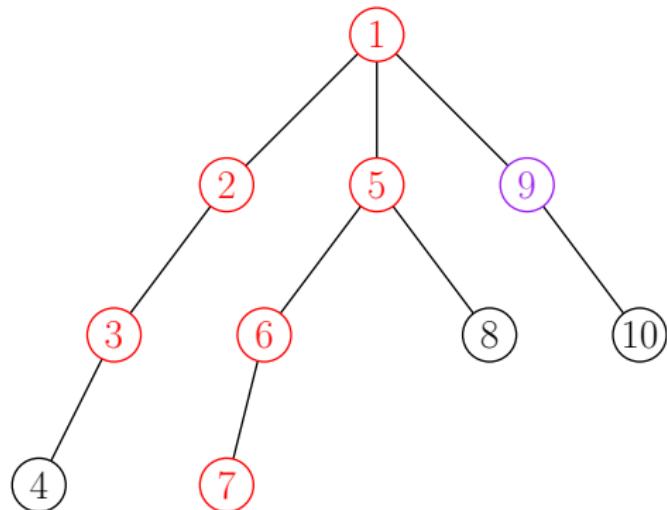
Tìm đường đi trong đồ thị bằng DFS: ví dụ



Tìm một đường
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	1	2
4	5	1
5	6	5

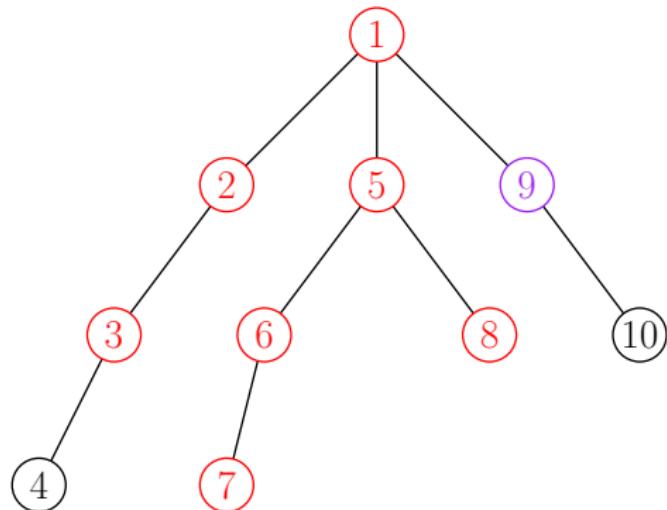
Tìm đường đi trong đồ thị bằng DFS: ví dụ



Tìm một đường
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	1	2
4	5	1
5	6	5
6	7	6

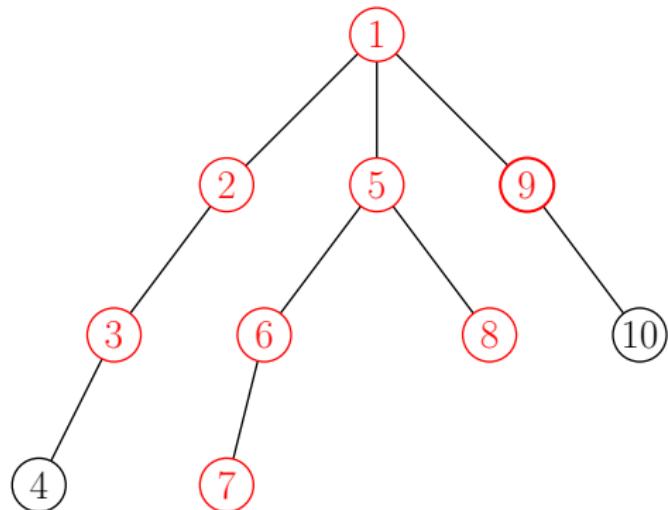
Tìm đường đi trong đồ thị bằng DFS: ví dụ



Tìm một đường
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	1	2
4	5	1
5	6	5
6	7	6
7	8	5

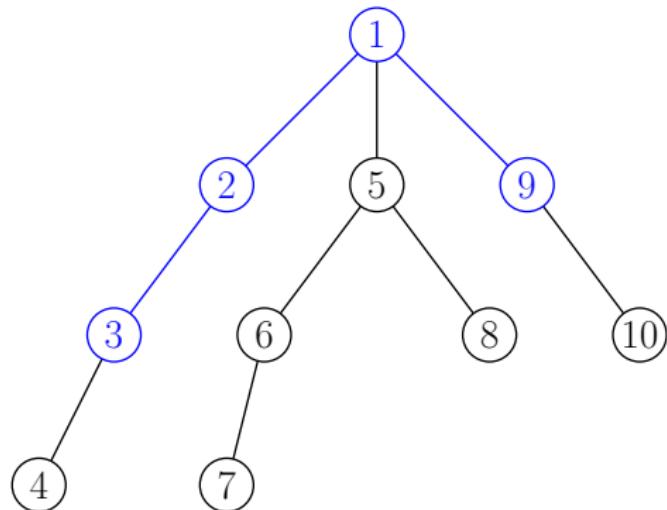
Tìm đường đi trong đồ thị bằng DFS: ví dụ



Tìm một đường
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	1	2
4	5	1
5	6	5
6	7	6
7	8	5
8	9	1

Tìm đường đi trong đồ thị bằng DFS: ví dụ



Tìm một đường
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	1	2
4	5	1
5	6	5
6	7	6
7	8	5
8	9	1

Tìm đường đi trong đồ thị bằng BFS

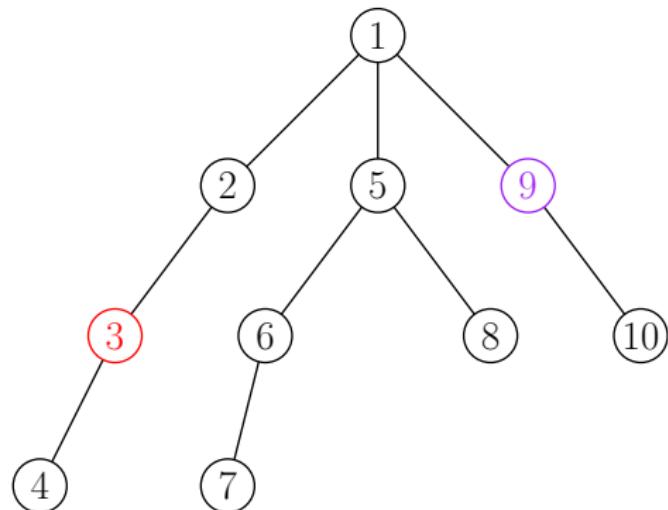
Thuật toán Tìm một đường đi từ s tới t bằng BFS

```

1: previous[v] :=  $\emptyset$  với mọi  $v \in V$ ;
2: previous[s] := s;
3: found := false;
4: first := 1; last := 1; Q[first] := s;
5: while first  $\leq$  last do
6:   v := Q[first];
7:   first := first + 1;
8:   for mỗi hàng xóm  $u$  của  $v$  do
9:     if previous[u] =  $\emptyset$  then
10:      previous[u] = v;
11:      if  $u = t$  then found := true; exit; end if
12:      last := last + 1;
13:      Q[last] := u;
14:    end if
15:  end for
16: end while
17: if found = true then in đường đi (sử dụng mảng previous); end if

```

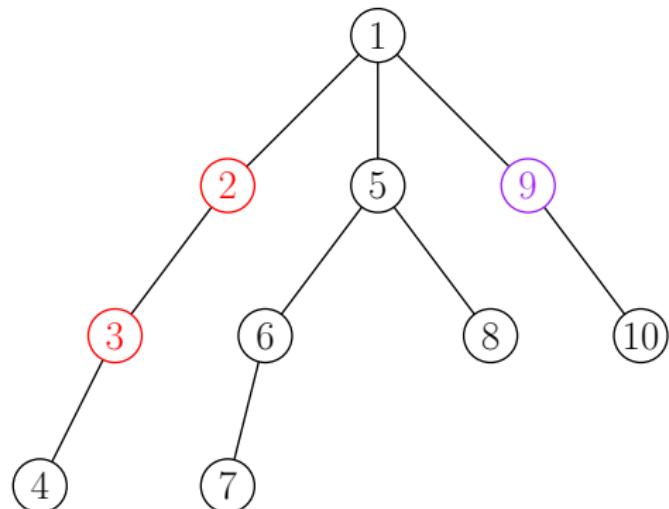
Tìm đường đi trong đồ thị bằng BFS: ví dụ



Tìm một đường đi
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3

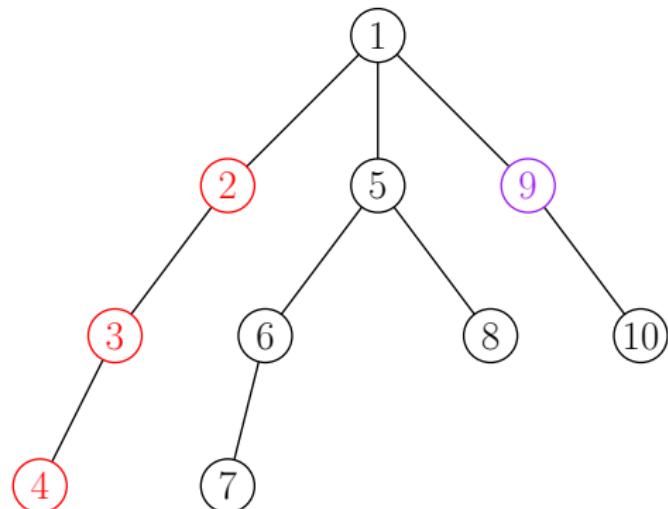
Tìm đường đi trong đồ thị bằng BFS: ví dụ



Tìm một đường đi
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3

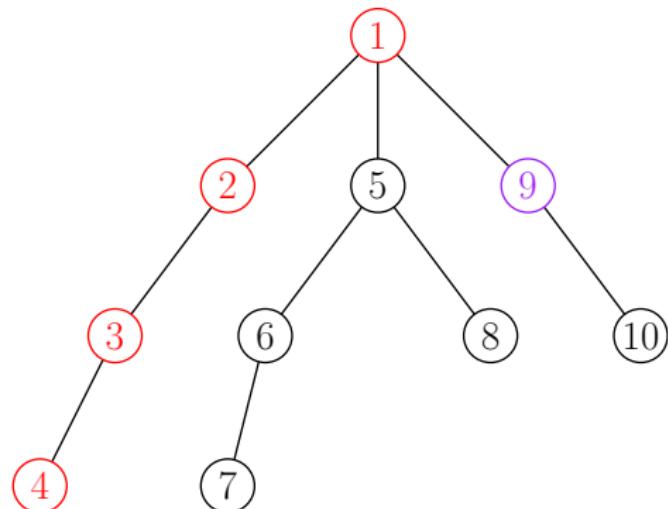
Tìm đường đi trong đồ thị bằng BFS: ví dụ



Tìm một đường đi
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	4	3

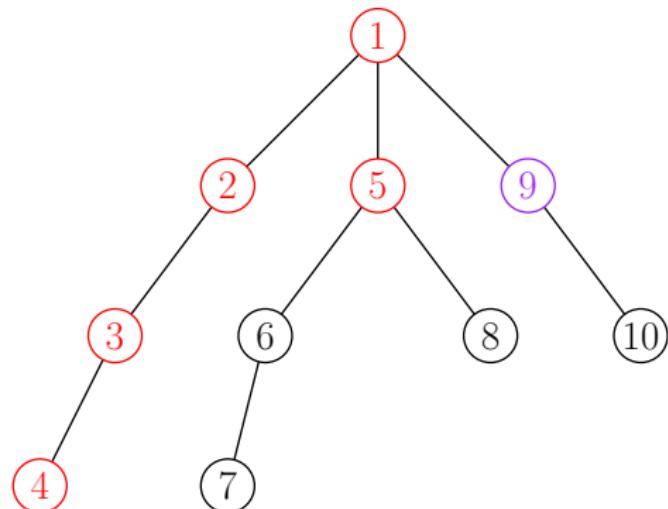
Tìm đường đi trong đồ thị bằng BFS: ví dụ



Tìm một đường đi
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	4	3
4	1	2

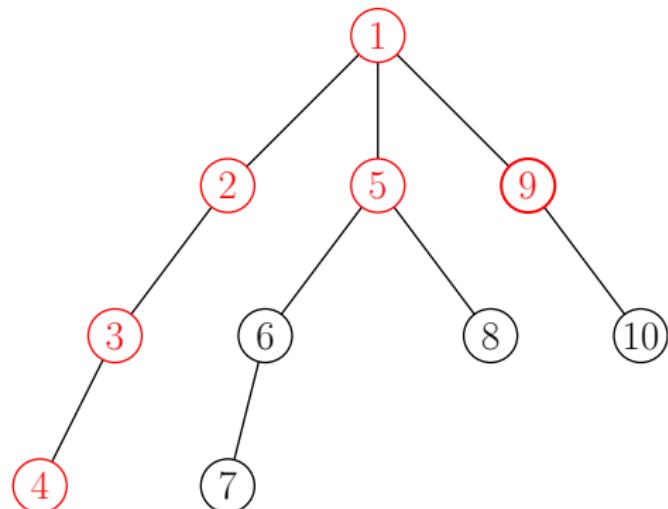
Tìm đường đi trong đồ thị bằng BFS: ví dụ



Tìm một đường đi
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	4	3
4	1	2
5	5	1

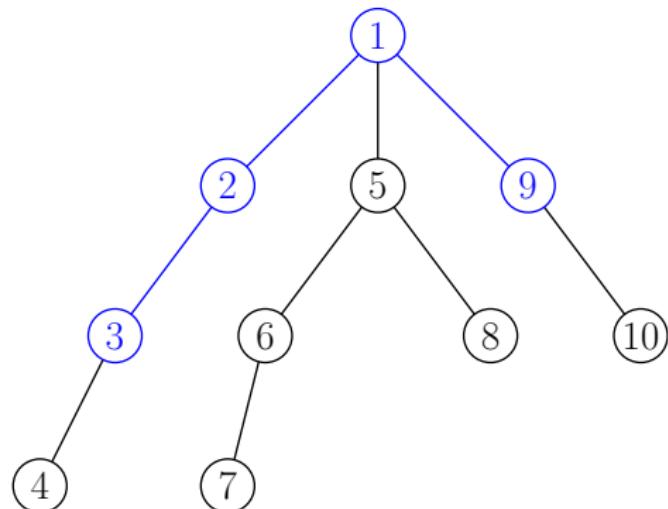
Tìm đường đi trong đồ thị bằng BFS: ví dụ



Tìm một đường đi
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	4	3
4	1	2
5	5	1
6	9	1

Tìm đường đi trong đồ thị bằng BFS: ví dụ



Tìm một đường đi
từ đỉnh 3 đến đỉnh 9

Bước	Đỉnh duyệt	Đỉnh trước
1	3	3
2	2	3
3	4	3
4	1	2
5	5	1
6	9	1

Nội dung

- 2.1 Tìm kiếm theo chiều sâu
- 2.2 Tìm kiếm theo chiều rộng
- 2.3 Áp dụng: tìm đường đi trong đồ thị
- 2.4 Áp dụng: kiểm tra tính liên thông

Kiểm tra tính liên thông

- **Cho trước:** một đồ thị vô hướng $G = (V, E)$
- **Bài toán:** đếm số thành phần liên thông của G
- **Ý tưởng:**
 - Mỗi lần duyệt DFS hoặc BFS cho một thành phần liên thông
 - Sử dụng một biến đếm số thành phần liên thông
 - Sử dụng một mảng để lưu thứ tự của thành phần liên thông mà mỗi đỉnh thuộc vào

Kiểm tra tính liên thông bằng DFS

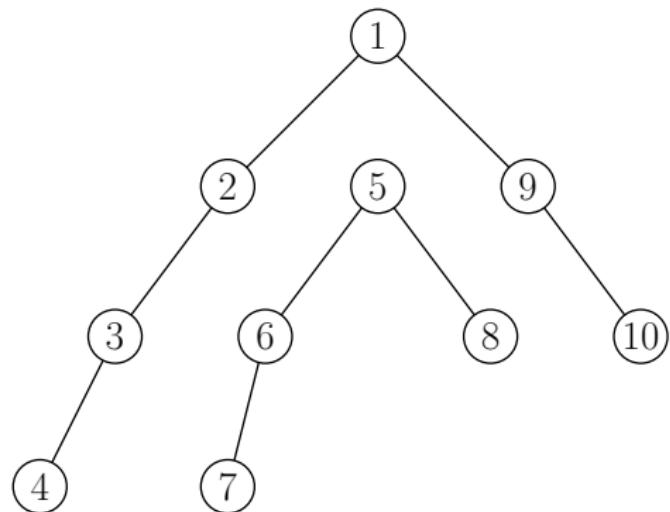
Thuật toán Kiểm tra tính liên thông bằng DFS

```
1: count := 0;  
2: component[v] := 0 với mọi  $v \in V$ ;  
3: for mỗi đỉnh  $v$  do  
4:   if component[v] = 0 then  
5:     count := count + 1;  
6:     DFS(v);  
7:   end if  
8: end for
```

Thuật toán DFS(v)

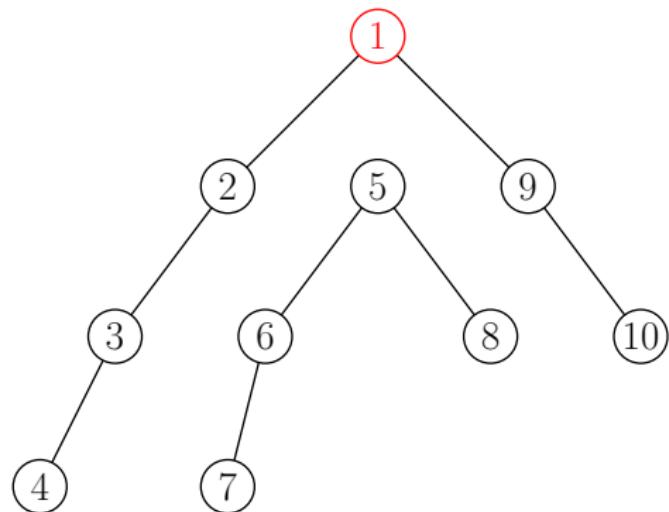
```
1: component[v] := count;  
2: for mỗi hàng xóm  $u$  của  $v$  do  
3:   DFS( $u$ );  
4: end for
```

Kiểm tra tính liên thông bằng DFS: ví dụ



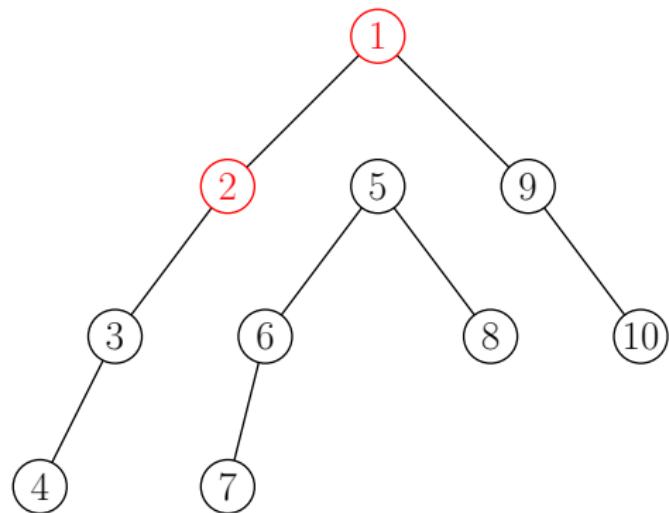
Bước	Đỉnh duyệt	Thành phần liên thông

Kiểm tra tính liên thông bằng DFS: ví dụ



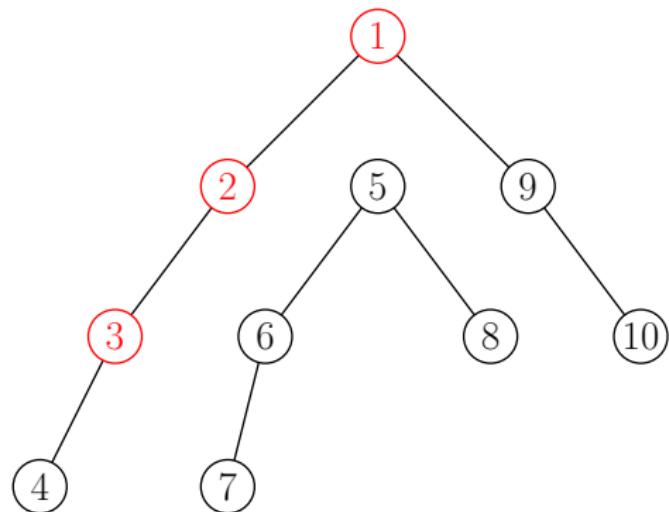
Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1

Kiểm tra tính liên thông bằng DFS: ví dụ



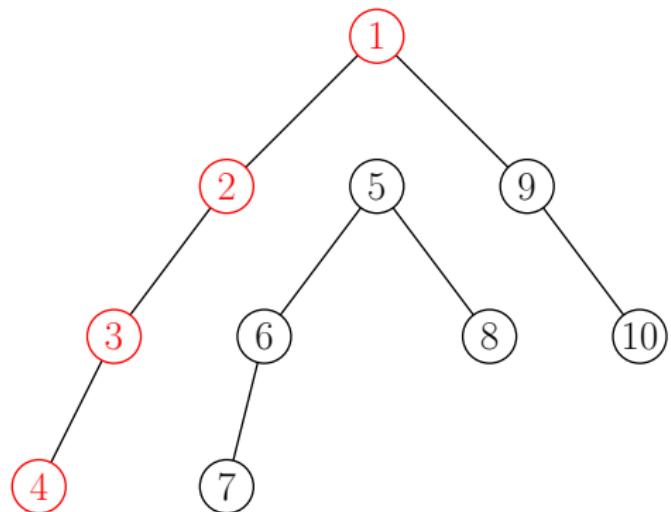
Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1
2	2	1

Kiểm tra tính liên thông bằng DFS: ví dụ



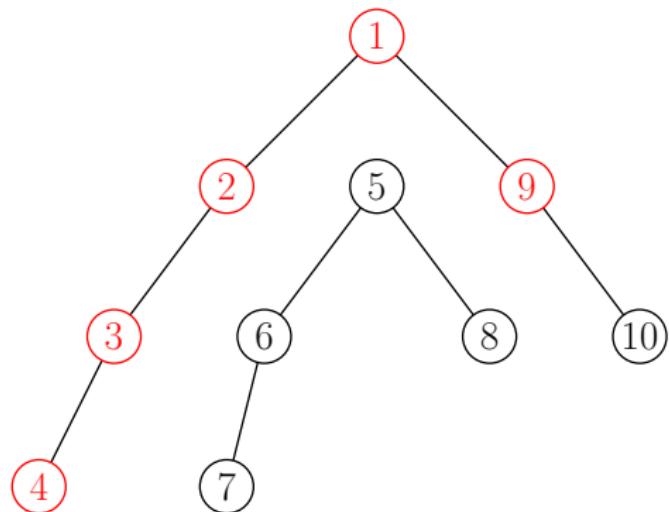
Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1
2	2	1
3	3	1

Kiểm tra tính liên thông bằng DFS: ví dụ



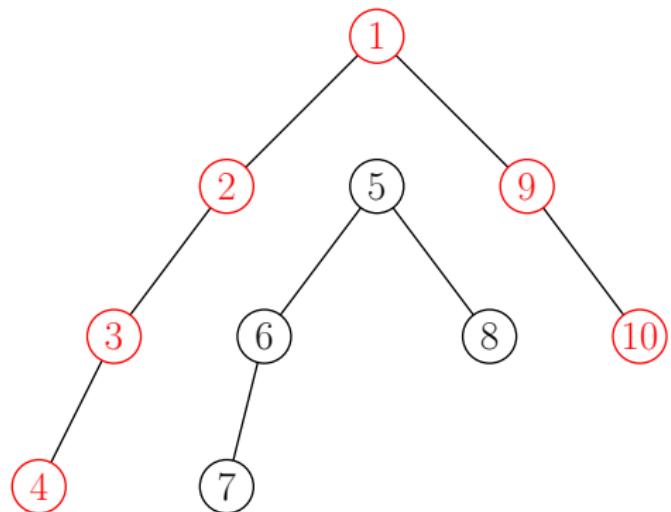
Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1
2	2	1
3	3	1
4	4	1

Kiểm tra tính liên thông bằng DFS: ví dụ



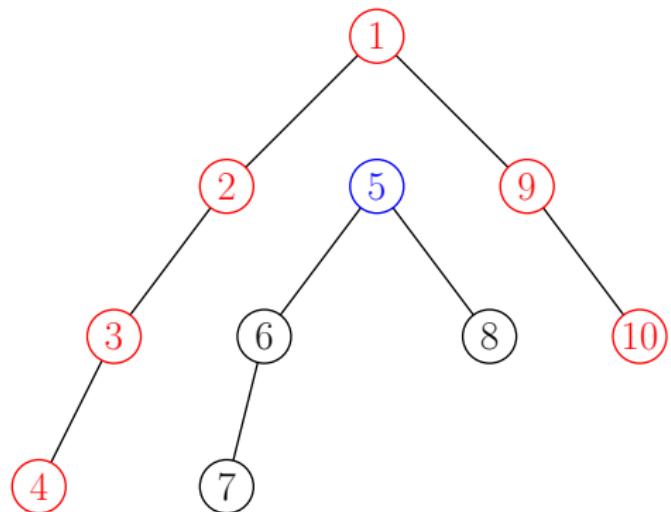
Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1
2	2	1
3	3	1
4	4	1
5	9	1

Kiểm tra tính liên thông bằng DFS: ví dụ



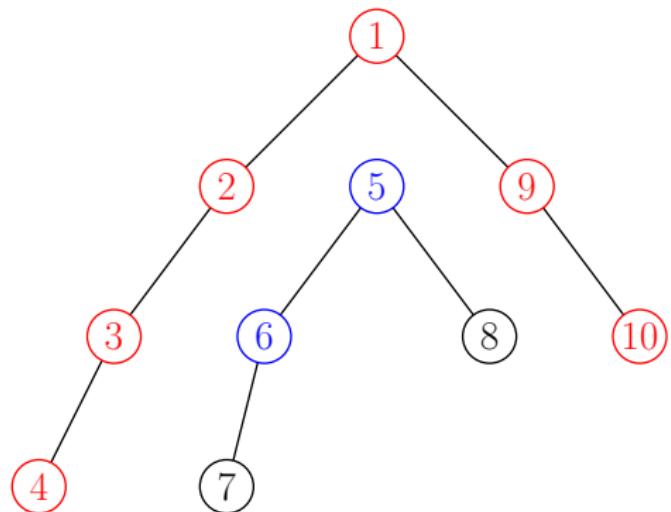
Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1
2	2	1
3	3	1
4	4	1
5	9	1
6	10	1

Kiểm tra tính liên thông bằng DFS: ví dụ



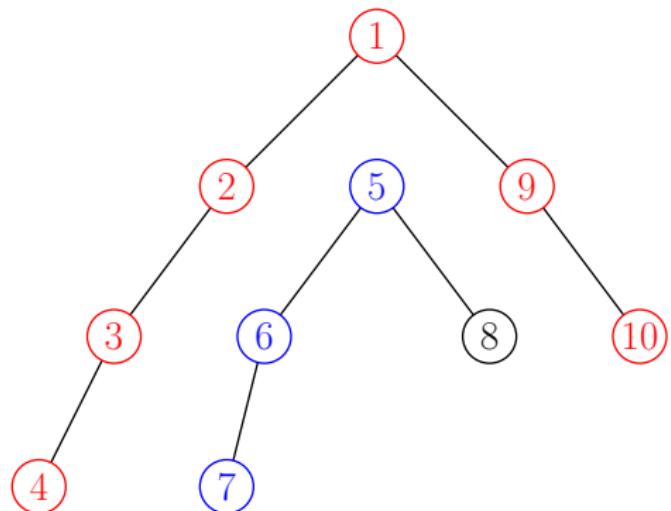
Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1
2	2	1
3	3	1
4	4	1
5	9	1
6	10	1
7	5	2

Kiểm tra tính liên thông bằng DFS: ví dụ



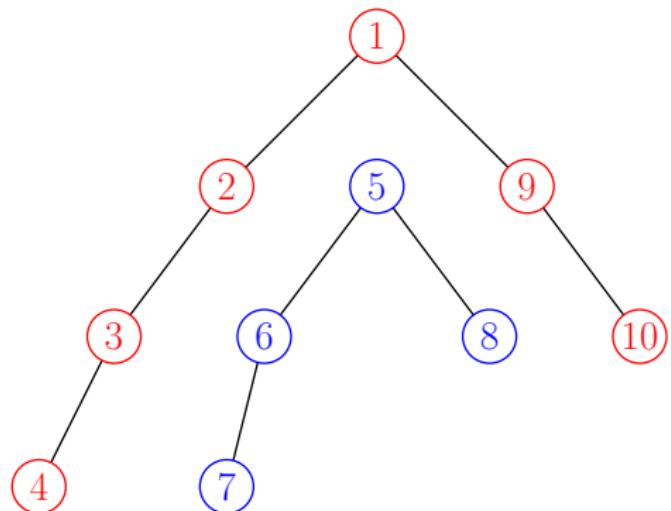
Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1
2	2	1
3	3	1
4	4	1
5	9	1
6	10	1
7	5	2
8	6	2

Kiểm tra tính liên thông bằng DFS: ví dụ



Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1
2	2	1
3	3	1
4	4	1
5	9	1
6	10	1
7	5	2
8	6	2
9	7	2

Kiểm tra tính liên thông bằng DFS: ví dụ



Bước	Đỉnh duyệt	Thành phần liên thông
1	1	1
2	2	1
3	3	1
4	4	1
5	9	1
6	10	1
7	5	2
8	6	2
9	7	2
10	8	2

Kiểm tra tính liên thông bằng BFS

Thuật toán Kiểm tra tính liên thông bằng BFS

```

1: count := 0;
2: component[v] := 0 với mọi  $v \in V$ ;
3: for mỗi đỉnh  $v$  do
4:   if component[v] = 0 then
5:     count := count + 1;
6:     BFS(v);
7:   end if
8: end for

```

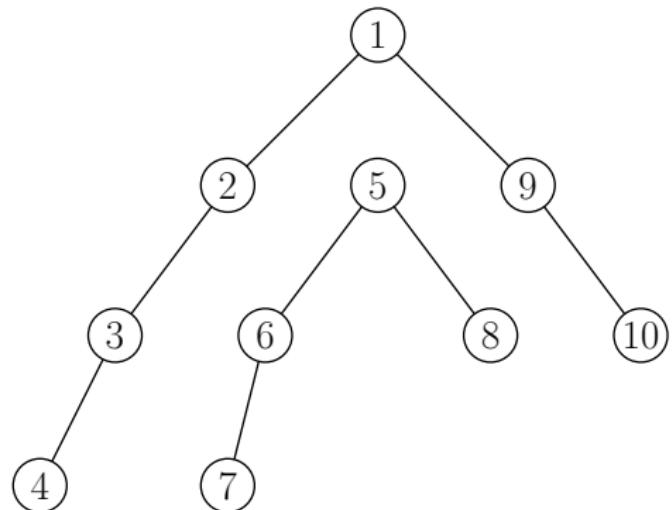
Thuật toán BFS(v)

```

1: component[v] := count;
2: first := 1; last := 1;
3: Q[first] := v;
4: while first  $\leq$  last do
5:   w := Q[first];
6:   first := first + 1;
7:   for mỗi hàng xóm  $u$  của  $w$  do
8:     last := last + 1;
9:     Q[last] := u;
10:    component[u] := count;
11:   end for
12: end while

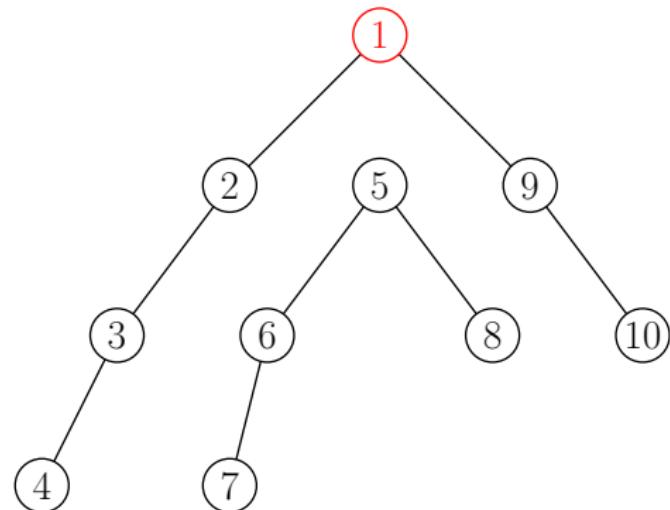
```

Kiểm tra tính liên thông bằng BFS: ví dụ



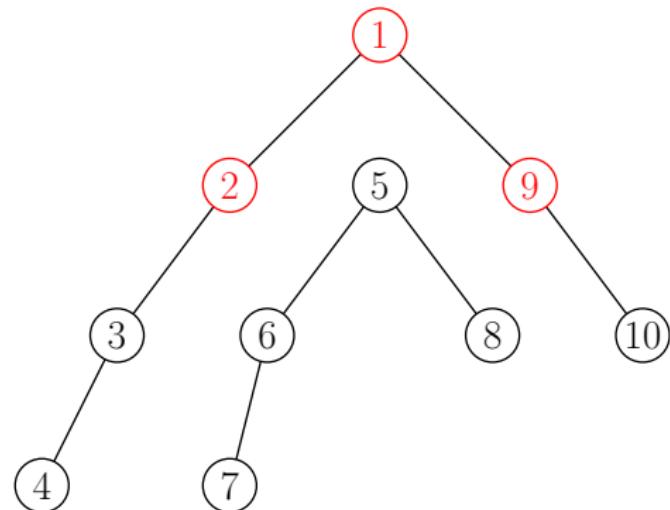
- Lần gọi BFS đầu tiên:
 - count = 1,
component 1
- Lần gọi BFS thứ hai:
 - count = 2,
component 2

Kiểm tra tính liên thông bằng BFS: ví dụ



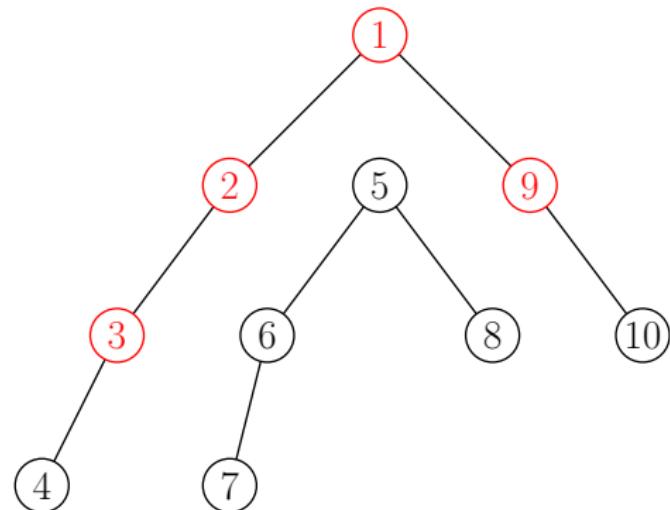
- Lần gọi BFS đầu tiên:
 - count = 1,
component 1
 - 1,
- Lần gọi BFS thứ hai:
 - count = 2,
component 2

Kiểm tra tính liên thông bằng BFS: ví dụ



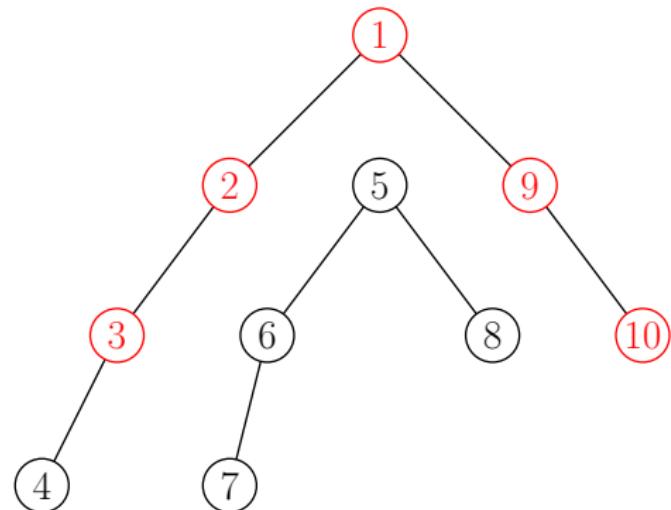
- Lần gọi BFS đầu tiên:
 - count = 1,
component 1
 - 1, 2, 9,
- Lần gọi BFS thứ hai:
 - count = 2,
component 2

Kiểm tra tính liên thông bằng BFS: ví dụ



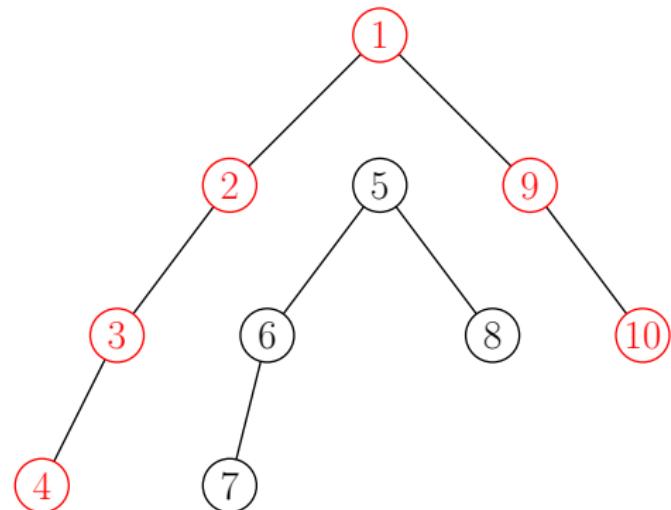
- Lần gọi BFS đầu tiên:
 - count = 1,
component 1
 - 1, 2, 9, 3,
- Lần gọi BFS thứ hai:
 - count = 2,
component 2

Kiểm tra tính liên thông bằng BFS: ví dụ



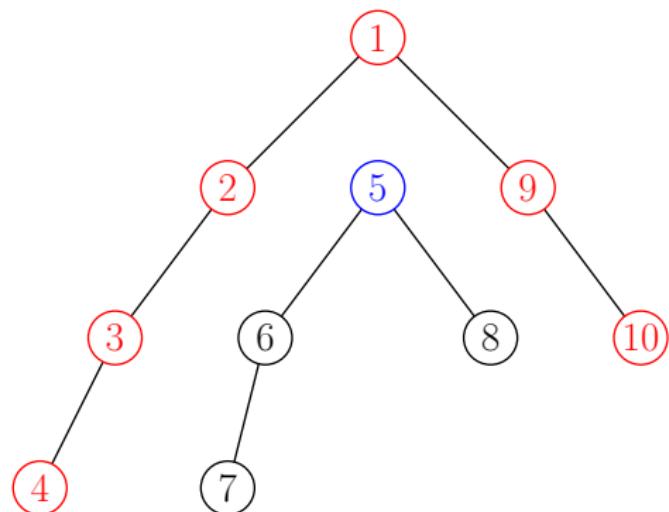
- Lần gọi BFS đầu tiên:
 - count = 1,
component 1
 - 1, 2, 9, 3, 10,
- Lần gọi BFS thứ hai:
 - count = 2,
component 2

Kiểm tra tính liên thông bằng BFS: ví dụ



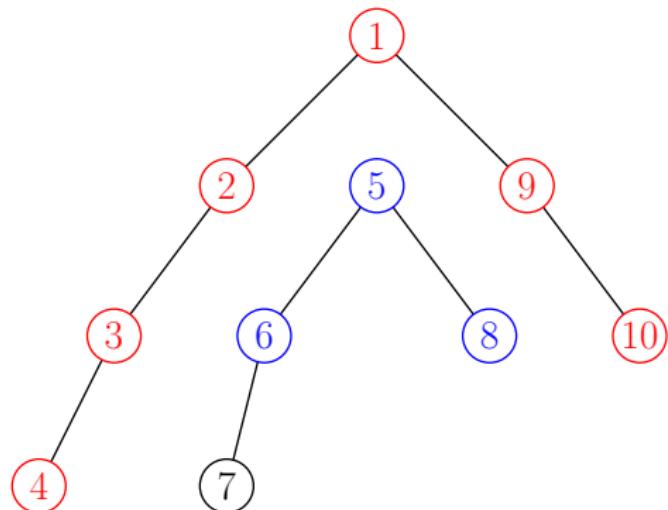
- Lần gọi BFS đầu tiên:
 - count = 1,
component 1
 - 1, 2, 9, 3, 10, 4
- Lần gọi BFS thứ hai:
 - count = 2,
component 2

Kiểm tra tính liên thông bằng BFS: ví dụ



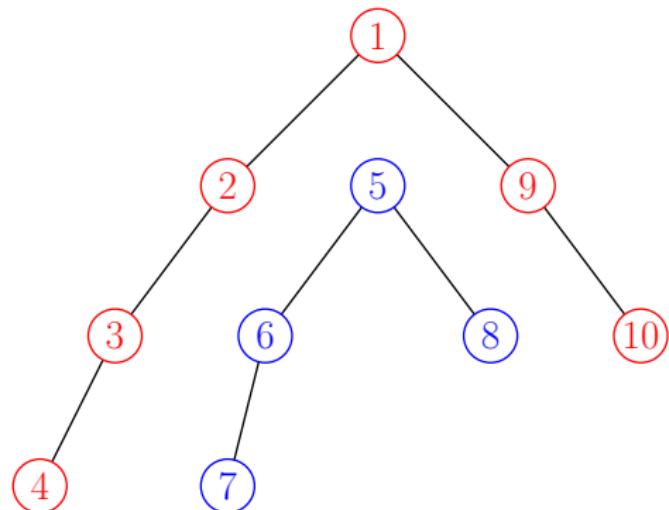
- Lần gọi BFS đầu tiên:
 - count = 1,
component 1
 - 1, 2, 9, 3, 10, 4
- Lần gọi BFS thứ hai:
 - count = 2,
component 2
 - 5,

Kiểm tra tính liên thông bằng BFS: ví dụ



- Lần gọi BFS đầu tiên:
 - count = 1,
component 1
 - 1, 2, 9, 3, 10, 4
- Lần gọi BFS thứ hai:
 - count = 2,
component 2
 - 5, 6, 8,

Kiểm tra tính liên thông bằng BFS: ví dụ



- Lần gọi BFS đầu tiên:
 - count = 1,
component 1
 - 1, 2, 9, 3, 10, 4
- Lần gọi BFS thứ hai:
 - count = 2,
component 2
 - 5, 6, 8, 7

Mục lục

1 Một số khái niệm cơ bản

2 Duyệt đồ thị

3 Đồ thị Euler

4 Đồ thị Hamilton

5 Đồ thị phẳng

6 Cây và cây khung

7 Đường đi ngắn nhất

Nhắc lại

Cho trước một đồ thị vô hướng $G = (V, E)$.

Định nghĩa

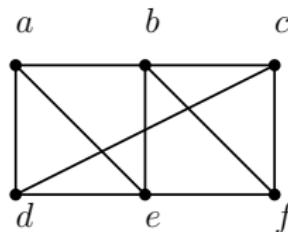
- Một *đường đi* có độ dài n từ $u \in V$ tới $v \in V$ là một dãy các đỉnh

$$x_0 \equiv u, x_1, \dots, x_n \equiv v \in V$$

thỏa mãn $\{x_{i-1}, x_i\} \in E$ với mọi $i = 1, \dots, n$.

- Một *chu trình* là một đường đi có độ dài dương, bắt đầu và kết thúc tại cùng một đỉnh.
- Một đường đi hoặc chu trình là *đơn* nếu nó đi qua mỗi cạnh nhiều nhất một lần.

Minh họa các khái niệm trên
trong ví dụ này:



Đường đi Euler, chu trình Euler

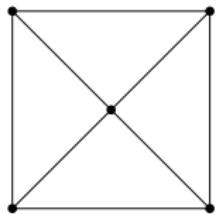
Cho trước một đồ thị vô hướng $G = (V, E)$.

Định nghĩa

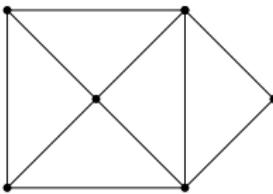
Một *đường đi Euler* (tương ứng, *chu trình Euler*) trong G là một đường đi (tương ứng, chu trình) **đơn** chứa **tất cả các cạnh** của G .

Định nghĩa

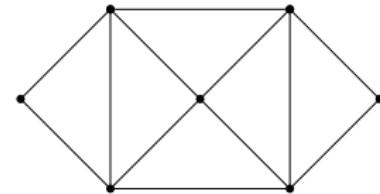
- G được gọi là một *đồ thị nửa Euler* nếu nó có một đường đi Euler.
- G được gọi là một *đồ thị Euler* nếu nó có một chu trình Euler.



Không nửa Euler



Nửa Euler



Euler

Đường đi Euler, chu trình Euler

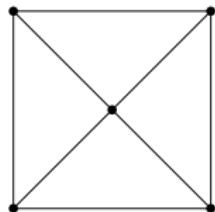
Cho trước một đồ thị vô hướng $G = (V, E)$.

Định nghĩa

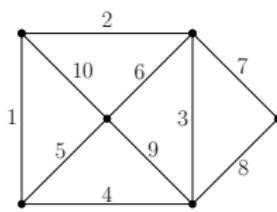
Một *đường đi Euler* (tương ứng, *chu trình Euler*) trong G là một đường đi (tương ứng, chu trình) **đơn** chứa **tất cả các cạnh** của G .

Định nghĩa

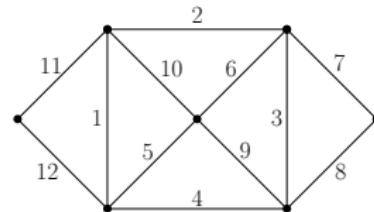
- G được gọi là một *đồ thị nửa Euler* nếu nó có một đường đi Euler.
- G được gọi là một *đồ thị Euler* nếu nó có một chu trình Euler.



Không nửa Euler



Nửa Euler



Euler

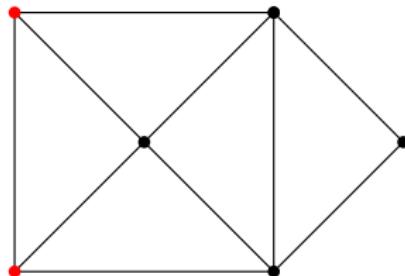
Định lý Euler

Cho trước một đồ thị vô hướng liên thông $G = (V, E)$.

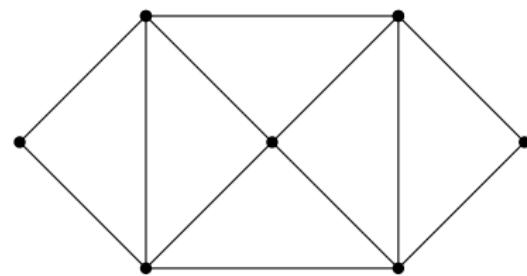
Định lý Euler (1736)

- G là một đồ thị Euler **khi và chỉ khi** mọi đỉnh của G đều có bậc chẵn.
- G là một đồ thị nửa Euler **khi và chỉ khi** G có nhiều nhất hai đỉnh bậc lẻ.

Chứng minh. Bài tập về nhà!



Nửa Euler



Euler

Thuật toán Hierholzer⁴

- **Đầu vào:** một đồ thị vô hướng liên thông $G = (V, E)$ với tất cả các đỉnh có bậc chẵn
- **Đầu ra:** chu trình Euler của G
- **Lược đồ thuật toán Hierholzer:**

Tìm một chu trình C trong G ;

Đặt H là đồ thị G sau khi xóa các cạnh của C ;

while H còn chứa cạnh **do**

Chọn một đỉnh của C liên thuộc với một cạnh của H ;

Xây dựng một chu trình con với đỉnh đã chọn;

Xóa các cạnh của chu trình con nêu trên khỏi H ;

Thêm chu trình con nêu trên vào C ;

end while

- *Chú ý:* với một số biến đổi, ta có thể tìm đường Euler trong đồ thị nửa Euler

⁴Carl Hierholzer (02.10.1840–13.09.1871): nhà toán học người Đức

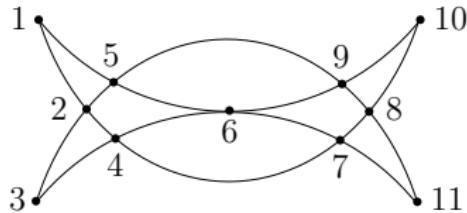
Thuật toán Hierholzer: giả mã và ví dụ

Thuật toán Hierholzer

```

1: order := 0; top := 1;
2: C[top] := một đỉnh  $v$ ;
3: while top > 0 do
4:    $w := C[\text{top}]$ ;
5:   if  $w$  có một hàng xóm  $u$  then
6:     top := top + 1;
7:      $C[\text{top}] := u$ ;
8:     xóa cạnh  $\{w, u\}$ ;
9:   else
10:    top := top - 1;
11:    order := order + 1;
12:     $E[\text{order}] := w$ ;
13:  end if
14: end while

```



Vòng đầu tiên:

- $C: 1, 2, 3, 4, 2, 5, 1$
- $E: 1$

Vòng thứ hai:

- $C: 1, 2, 3, 4, 2, 5, 6, 4, 7, 6, 9, 5$
- $E: 1, 5$

Vòng thứ ba:

- $C: 1, 2, 3, 4, 2, 5, 6, 4, 7, 6, 9, 8, 7, 11, 8, 10, 9$
- $E: 1, 5, 9$

Vòng cuối:

- $E: 1, 5, 9, 10, 8, 11, 7, 8, 9, 6, 7, 4, 6, 5, 2, 4, 3, 2, 1$

- C : chu trình tạm thời trong lược đồ
- E : mảng lưu trữ chu trình Euler

Mục lục

1 Một số khái niệm cơ bản

2 Duyệt đồ thị

3 Đồ thị Euler

4 Đồ thị Hamilton

5 Đồ thị phẳng

6 Cây và cây khung

7 Đường đi ngắn nhất

Đường đi Hamilton, chu trình Hamilton

Cho trước một đồ thị vô hướng liên thông $G = (V, E)$.

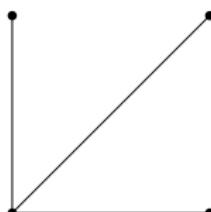
Định nghĩa

- Một *đường đi Hamilton* trong G là một đường đi **đơn** qua **mỗi đỉnh** của G chính xác một lần.
- Một *chu trình Hamilton* trong G là một chu trình **đơn** qua **mỗi đỉnh** của G chính xác một lần.

Định nghĩa

- G được gọi là *đồ thị nửa Hamilton* nếu nó có một đường Hamilton.
- G được gọi là *đồ thị Hamilton* nếu nó có một chu trình Hamilton.

Không nửa Hamilton



Đường đi Hamilton, chu trình Hamilton

Cho trước một đồ thị vô hướng liên thông $G = (V, E)$.

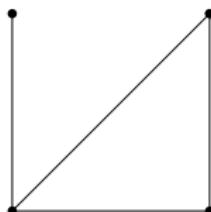
Định nghĩa

- Một *đường đi Hamilton* trong G là một đường đi **đơn** qua **mỗi đỉnh** của G chính xác một lần.
- Một *chu trình Hamilton* trong G là một chu trình **đơn** qua **mỗi đỉnh** của G chính xác một lần.

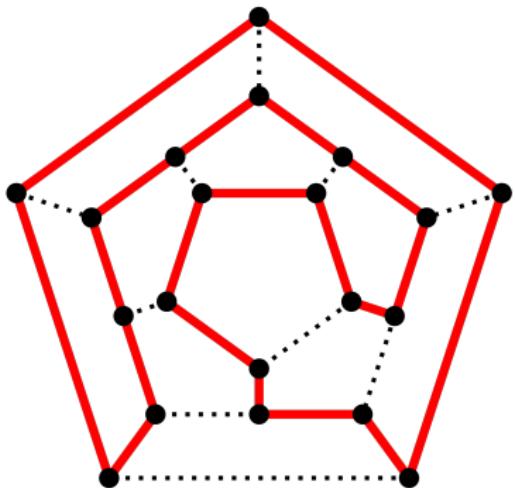
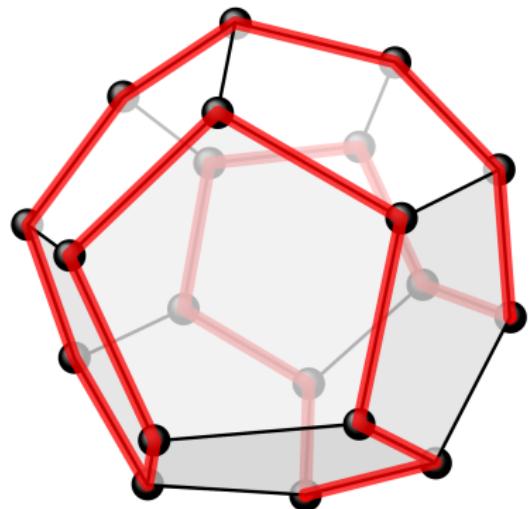
Định nghĩa

- G được gọi là *đồ thị nửa Hamilton* nếu nó có một đường Hamilton.
- G được gọi là *đồ thị Hamilton* nếu nó có một chu trình Hamilton.

Nửa Hamilton



Một ví dụ về đồ thị Hamilton⁵



⁵ Câu đố Icosian, đặt ra từ 1857 bởi nhà toán học người Ai-len William Rowan Hamilton (03.08.1805–02.09.1865)

Các điều kiện đủ cho sự tồn tại của chu trình Hamilton

Cho trước một đồ thị vô hướng liên thông $G = (V, E)$ với $|V| = n \geq 3$.

Định lý Dirac⁶ (1952)

Nếu mọi đỉnh của G có bậc không dưới $\frac{n}{2}$, thì G có một chu trình Hamilton.

Định lý Ore⁷ (1960)

Nếu $\deg(u) + \deg(v) \geq n$ với mọi cặp đỉnh **không kề nhau** $u, v \in V$, thì G có một chu trình Hamilton.

⁶Gabriel Andrew Dirac (13.03.1925–20.07.1984): nhà toán học người Hungary - Anh

⁷Øystein Ore (07.10.1899–13.08.1968): nhà toán học người Na Uy

Các điều kiện đủ cho sự tồn tại của chu trình Hamilton

Cho trước một đồ thị vô hướng liên thông $G = (V, E)$ với $|V| = n \geq 3$.

Định lý Pósa⁸ (1962)

Giả sử các đỉnh của G được sắp xếp theo thứ tự bậc tăng dần:

$$\deg(v_1) \leq \deg(v_2) \leq \dots \leq \deg(v_n).$$

Nếu với mỗi $k < \frac{n}{2}$ ta có $k < \deg(v_k)$, thì G có một chu trình Hamilton.

Ghi chú:

Định lý Pósa \implies Định lý Ore \implies Định lý Dirac

⁸Lajos Pósa (born 09.12.1947): nhà toán học người Hungary

Mục lục

1 Một số khái niệm cơ bản

2 Duyệt đồ thị

3 Đồ thị Euler

4 Đồ thị Hamilton

5 Đồ thị phẳng

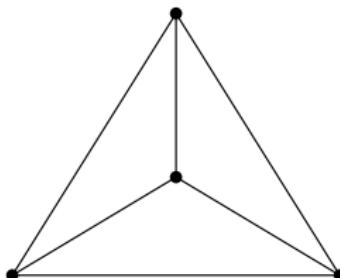
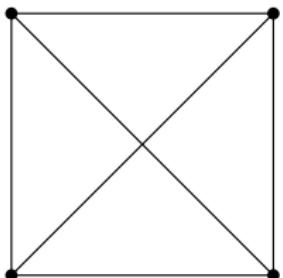
6 Cây và cây khung

7 Đường đi ngắn nhất

Đồ thị phẳng

Định nghĩa

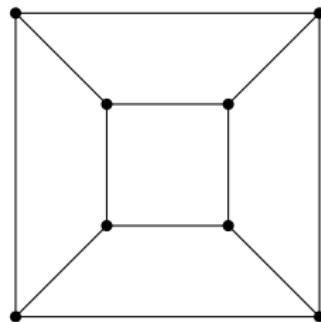
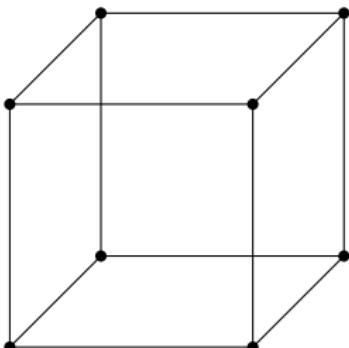
- Một đồ thị vô hướng liên thông được gọi là *đồ thị phẳng* nếu nó có thể vẽ được trên một mặt phẳng mà các cạnh chỉ giao nhau tại các đỉnh.
- Một đồ thị có hướng liên thông được gọi là *đồ thị phẳng* nếu đồ thị vô hướng nền của nó phẳng.



Đồ thị phẳng

Định nghĩa

- Một đồ thị vô hướng liên thông được gọi là *đồ thị phẳng* nếu nó có thể vẽ được trên một mặt phẳng mà các cạnh chỉ giao nhau tại các đỉnh.
- Một đồ thị có hướng liên thông được gọi là *đồ thị phẳng* nếu đồ thị vô hướng nền của nó phẳng.

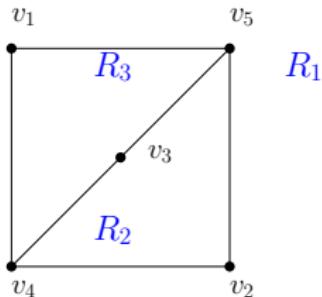
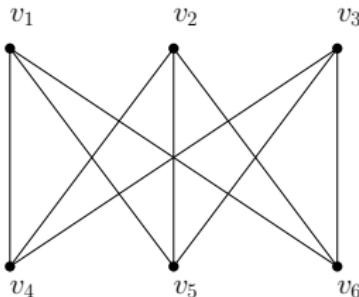


Đồ thị phẳng

Ví dụ: Chứng minh rằng $K_{3,3}$ không phẳng.

Chứng minh. Giả sử $K_{3,3}$ là đồ thị phẳng, khi đó:

- v_1, v_4, v_5, v_2 tạo thành một chu trình.
- Nếu v_3 nằm trong miền tạo bởi chu trình trên:
 - Nếu v_6 ở R_1 , thì $\{v_3, v_6\}$ phải cắt chu trình.
 - Nếu v_6 ở R_2 , thì $\{v_1, v_6\}$ phải cắt hoặc $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
 - Nếu v_6 ở R_3 , thì $\{v_2, v_6\}$ phải cắt $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
- Nếu v_3 nằm ngoài miền tạo bởi chu trình trên:
 - Lập luận tương tự (bài tập về nhà!)

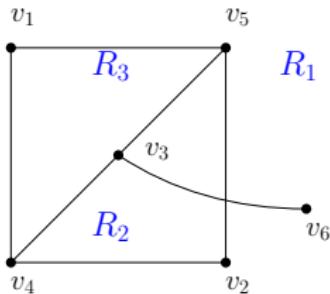
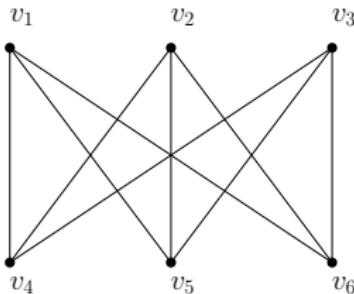


Đồ thị phẳng

Ví dụ: Chứng minh rằng $K_{3,3}$ không phẳng.

Chứng minh. Giả sử $K_{3,3}$ là đồ thị phẳng, khi đó:

- v_1, v_4, v_5, v_2 tạo thành một chu trình.
- Nếu v_3 nằm trong miền tạo bởi chu trình trên:
 - Nếu v_6 ở R_1 , thì $\{v_3, v_6\}$ phải cắt chu trình.
 - Nếu v_6 ở R_2 , thì $\{v_1, v_6\}$ phải cắt hoặc $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
 - Nếu v_6 ở R_3 , thì $\{v_2, v_6\}$ phải cắt $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
- Nếu v_3 nằm ngoài miền tạo bởi chu trình trên:
 - Lập luận tương tự (bài tập về nhà)

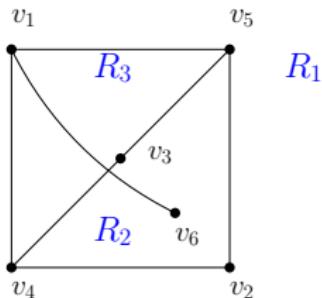
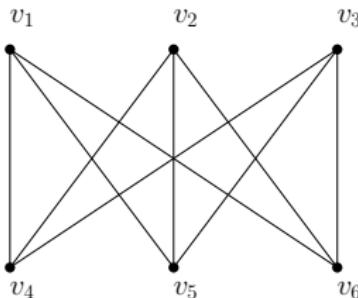


Đồ thị phẳng

Ví dụ: Chứng minh rằng $K_{3,3}$ không phẳng.

Chứng minh. Giả sử $K_{3,3}$ là đồ thị phẳng, khi đó:

- v_1, v_4, v_5, v_2 tạo thành một chu trình.
- Nếu v_3 nằm trong miền tạo bởi chu trình trên:
 - Nếu v_6 ở R_1 , thì $\{v_3, v_6\}$ phải cắt chu trình.
 - Nếu v_6 ở R_2 , thì $\{v_1, v_6\}$ phải cắt hoặc $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
 - Nếu v_6 ở R_3 , thì $\{v_2, v_6\}$ phải cắt $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
- Nếu v_3 nằm ngoài miền tạo bởi chu trình trên:
 - Lập luận tương tự (bài tập về nhà)

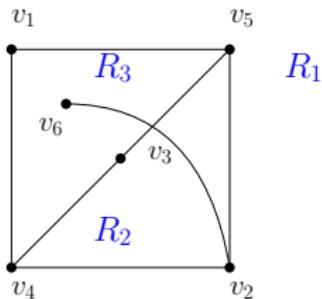
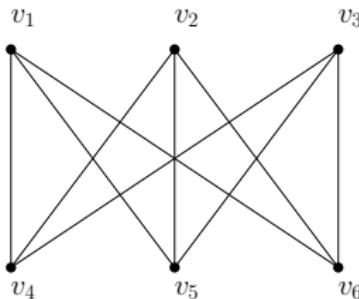


Đồ thị phẳng

Ví dụ: Chứng minh rằng $K_{3,3}$ không phẳng.

Chứng minh. Giả sử $K_{3,3}$ là đồ thị phẳng, khi đó:

- v_1, v_4, v_5, v_2 tạo thành một chu trình.
- Nếu v_3 nằm trong miền tạo bởi chu trình trên:
 - Nếu v_6 ở R_1 , thì $\{v_3, v_6\}$ phải cắt chu trình.
 - Nếu v_6 ở R_2 , thì $\{v_1, v_6\}$ phải cắt hoặc $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
 - Nếu v_6 ở R_3 , thì $\{v_2, v_6\}$ phải cắt $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
- Nếu v_3 nằm ngoài miền tạo bởi chu trình trên:
 - Lập luận tương tự (bài tập về nhà)



Đồ thị phẳng

Ví dụ: Chứng minh rằng $K_{3,3}$ không phẳng.

Chứng minh. Giả sử $K_{3,3}$ là đồ thị phẳng, khi đó:

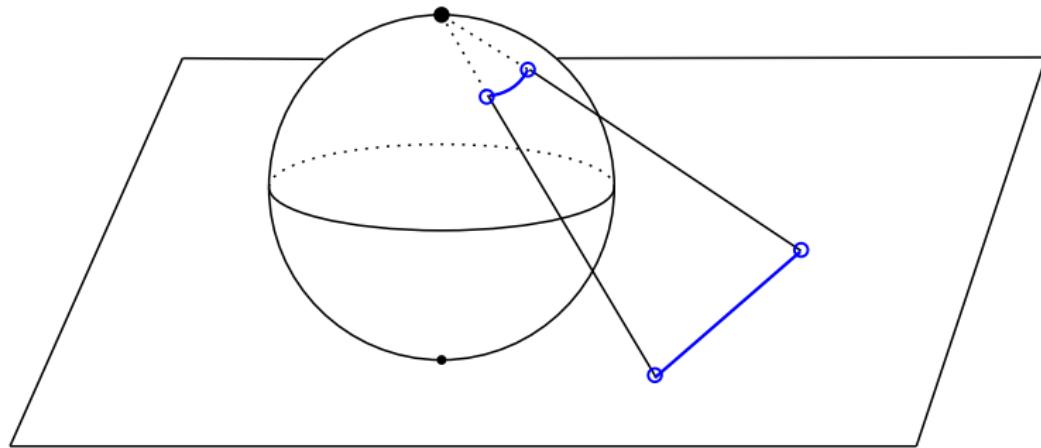
- v_1, v_4, v_5, v_2 tạo thành một chu trình.
- Nếu v_3 nằm trong miền tạo bởi chu trình trên:
 - Nếu v_6 ở R_1 , thì $\{v_3, v_6\}$ phải cắt chu trình.
 - Nếu v_6 ở R_2 , thì $\{v_1, v_6\}$ phải cắt hoặc $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
 - Nếu v_6 ở R_3 , thì $\{v_2, v_6\}$ phải cắt $\{v_3, v_4\}$ hoặc $\{v_3, v_5\}$.
- Nếu v_3 nằm ngoài miền tạo bởi chu trình trên:
 - Lập luận tương tự ([bài tập về nhà!](#))

Nhúng phẳng và nhúng cầu

Định lý

Một đồ thị là phẳng khi và chỉ khi có thể nhúng nó lên một mặt cầu mà các cạnh chỉ giao nhau tại các đỉnh.

Ý tưởng chứng minh.



Công thức Euler

Định lý (công thức Euler)

Cho G là một đồ thị đơn, liên thông, phẳng với v đỉnh và e cạnh. Gọi f là số miền trên mặt phẳng được phân hoạch bởi các cạnh của G . Khi đó ta có

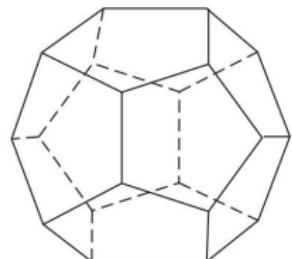
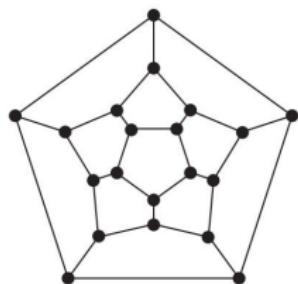
$$v - e + f = 2.$$

Chứng minh. Xem Chương 13 trong
[Proofs from THE BOOK \(sixth edition\).](#)
 Springer, 2018.

Hệ quả (phiên bản đa diện của công thức Euler)

Cho P là một đa diện với v đỉnh, e cạnh, và f mặt. Ta có

$$v - e + f = 2.$$



$$v = 20, e = 30, f = 12$$

Công thức Euler: Hệ quả 1

Hệ quả

Cho $G = (V, E)$ là một đồ thị đơn, liên thông, phẳng với $|V| \geq 3$. Ta có

$$|E| \leq 3|V| - 6.$$

Chứng minh.

- Gọi *bậc* của một miền là số cạnh tạo thành biên của miền đó (một cạnh xuất hiện 2 lần trên biên đóng góp 2 đơn vị vào bậc).
- G chia mặt phẳng thành f miền, mỗi miền có bậc ≥ 3 .
- Vì $2|E| = \sum_{\text{tất cả các miền } R} \deg(R) \geq 3f$, ta có $f \leq \frac{2}{3}|E|$.
- Theo công thức Euler, $f = 2 - |V| + |E|$, so $2 - |V| + |E| \leq \frac{2}{3}|E|$, hay tương đương $|E| \leq 3|V| - 6$.

Áp dụng.

K_5 không phẳng, vì với K_5 ta có $|V| = 5, |E| = 10 > 3|V| - 6 = 9$.

Công thức Euler: Hệ quả 2

Hệ quả

Cho $G = (V, E)$ là một đồ thị đơn, liên thông, phẳng. Khi đó G có một đỉnh có bậc không vượt quá 5.

Chứng minh.

- Trường hợp $|V| = 1$ hoặc $|V| = 2$: hiển nhiên. Xét $|V| \geq 3$.
- Giả sử mọi đỉnh của G đều có bậc ≥ 6 .
- Theo định lý bắt tay, ta có $2|E| = \sum_{v \in V} \deg(v)$, vì vậy $2|E| \geq 6|V|$.
- Theo hệ quả trước, ta có $|E| \leq 3|V| - 6$, vì vậy $2|E| \leq 6|V| - 12$ (mâu thuẫn với giả thiết).

Áp dụng.

K_n (với $n \geq 6$) không phẳng, vì mọi đỉnh đều có bậc > 5 .

Phép phân chia cơ bản và đồng cấu đồ thị

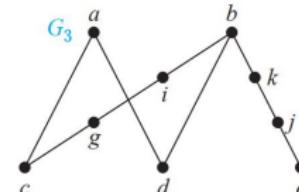
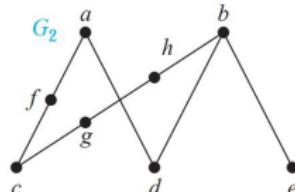
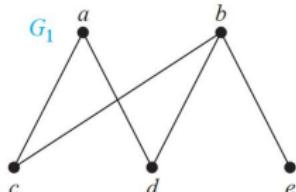
Định nghĩa

Một phép phân chia cơ bản trên một đồ thị vô hướng đơn G là một hành động bao gồm:

- xóa một cạnh $\{u, v\}$, sau đó
- thêm một đỉnh mới w , và
- thêm hai cạnh mới $\{u, w\}$ và $\{v, w\}$.

Định nghĩa

Hai đồ thị vô hướng đơn được gọi là *đồng cấu* nếu chúng nhận được từ cùng một đồ thị sau một dãy các phép phân chia cơ bản.



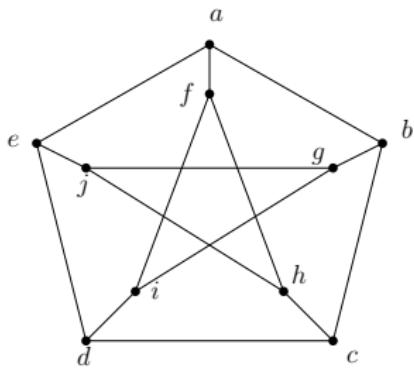
Định lý Kuratowski¹⁰

Định lý

Một đồ thị **không phẳng** khi và chỉ khi nó có một đồ thị con đồng cấu với $K_{3,3}$ hoặc K_5 .

Áp dụng.

Đồ thị Petersen⁹ không là đồ thị phẳng.



⁹ Julius Peter Christian Petersen (16.06.1839–05.08.1910): nhà toán học người Đan Mạch

¹⁰ Kazimierz Kuratowski (02.02.1896–18.06.1980): nhà toán học Ba Lan

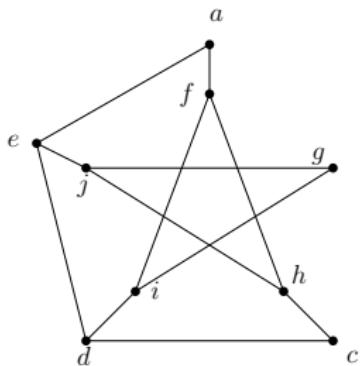
Định lý Kuratowski¹⁰

Định lý

Một đồ thị **không phẳng** khi và chỉ khi nó có một đồ thị con đồng cấu với $K_{3,3}$ hoặc K_5 .

Áp dụng.

Đồ thị Petersen⁹ không là đồ thị phẳng.



⁹ Julius Peter Christian Petersen (16.06.1839–05.08.1910): nhà toán học người Đan Mạch

¹⁰ Kazimierz Kuratowski (02.02.1896–18.06.1980): nhà toán học Ba Lan

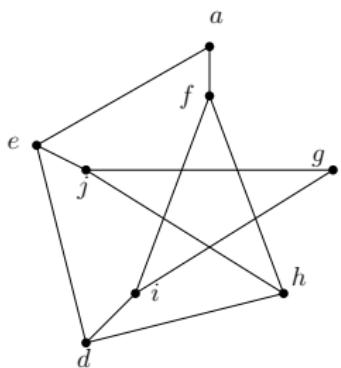
Định lý Kuratowski¹⁰

Định lý

Một đồ thị **không phẳng** khi và chỉ khi nó có một đồ thị con đồng cấu với $K_{3,3}$ hoặc K_5 .

Áp dụng.

Đồ thị Petersen⁹ không là đồ thị phẳng.



⁹ Julius Peter Christian Petersen (16.06.1839–05.08.1910): nhà toán học người Đan Mạch

¹⁰ Kazimierz Kuratowski (02.02.1896–18.06.1980): nhà toán học Ba Lan

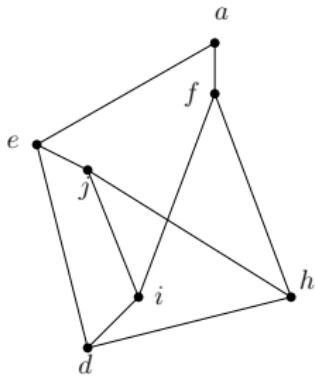
Định lý Kuratowski¹⁰

Định lý

Một đồ thị **không phẳng** khi và chỉ khi nó có một đồ thị con đồng cấu với $K_{3,3}$ hoặc K_5 .

Áp dụng.

Đồ thị Petersen⁹ không là đồ thị phẳng.



⁹ Julius Peter Christian Petersen (16.06.1839–05.08.1910): nhà toán học người Đan Mạch

¹⁰ Kazimierz Kuratowski (02.02.1896–18.06.1980): nhà toán học Ba Lan

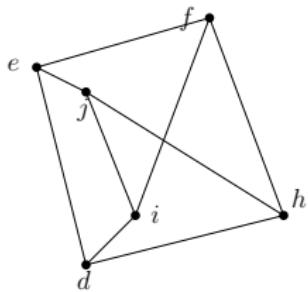
Định lý Kuratowski¹⁰

Định lý

Một đồ thị **không phẳng** khi và chỉ khi nó có một đồ thị con đồng cấu với $K_{3,3}$ hoặc K_5 .

Áp dụng.

Đồ thị Petersen⁹ không là đồ thị phẳng.



⁹ Julius Peter Christian Petersen (16.06.1839–05.08.1910): nhà toán học người Đan Mạch

¹⁰ Kazimierz Kuratowski (02.02.1896–18.06.1980): nhà toán học Ba Lan

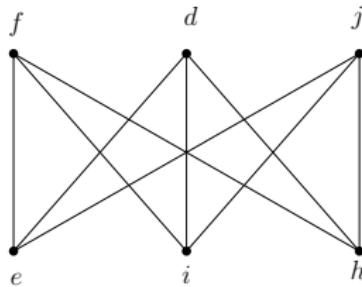
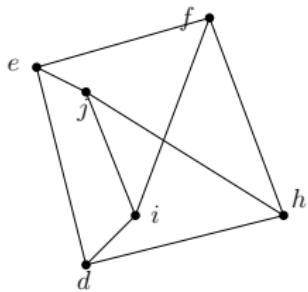
Định lý Kuratowski¹⁰

Định lý

Một đồ thị **không phẳng** khi và chỉ khi nó có một đồ thị con đồng cấu với $K_{3,3}$ hoặc K_5 .

Áp dụng.

Đồ thị Petersen⁹ không là đồ thị phẳng.



⁹ Julius Peter Christian Petersen (16.06.1839–05.08.1910): nhà toán học người Đan Mạch

¹⁰ Kazimierz Kuratowski (02.02.1896–18.06.1980): nhà toán học Ba Lan

Mục lục

1 Một số khái niệm cơ bản

2 Duyệt đồ thị

3 Đồ thị Euler

4 Đồ thị Hamilton

5 Đồ thị phẳng

6 Cây và cây khung

7 Đường đi ngắn nhất

Nội dung

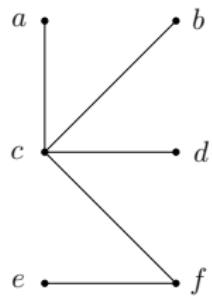
- 6.1 Cây và rừng
- 6.2 Cây khung (tối thiểu)
- 6.3 Thuật toán Kruskal
- 6.4 Thuật toán Prim

Cây

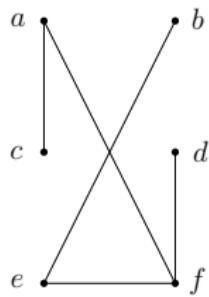
Định nghĩa

Một cây là một đồ thị vô hướng liên thông không có chu trình đơn

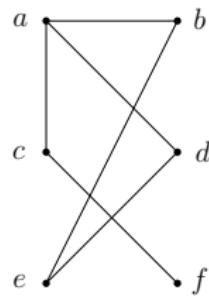
Chú ý: Cây luôn là đồ thị đơn



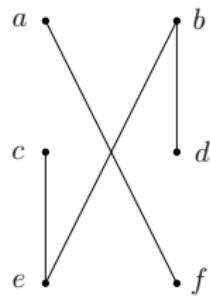
Tree



Tree



Not a tree



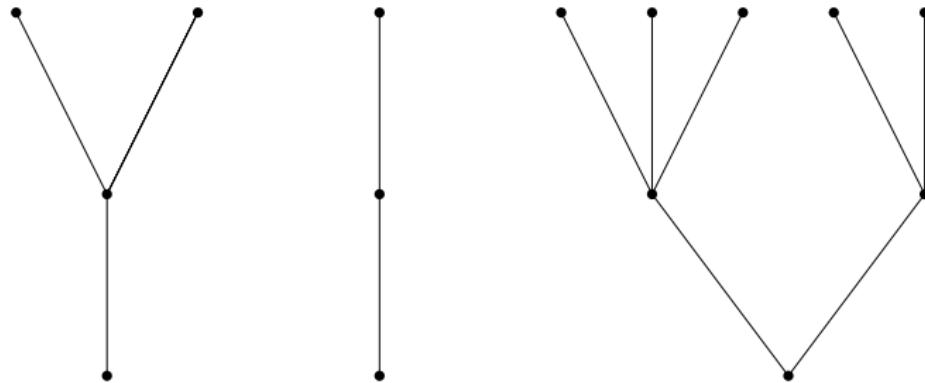
Not a tree

Rừng

Định nghĩa

Một đồ thị vô hướng được gọi là *rừng* nếu mỗi thành phần liên thông của nó là một cây

Ví dụ về *rừng* có 3 cây:



Cấu trúc cây

Định lý

Cho $G = (V, E)$ là một đồ thị vô hướng đơn.

Khi đó các khẳng định sau là tương đương.

- (i) G là một cây.
- (ii) G không có chu trình, và $|E| = |V| - 1$.
- (iii) G liên thông, và $|E| = |V| - 1$.
- (iv) G liên thông, và mọi cạnh của G đều là cạnh cắt.
- (v) Hai đỉnh bất kỳ của G có thể nối với nhau bởi duy nhất một đường đơn.
- (vi) G không có chu trình, và nếu thêm một cạnh vào G ta có một chu trình đơn.

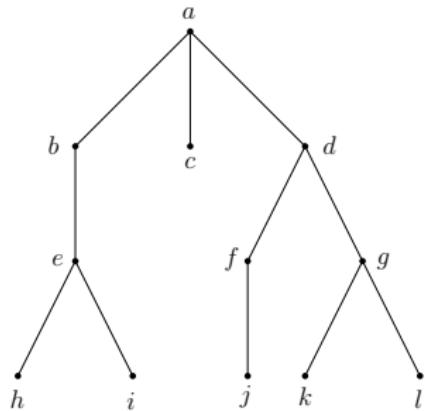
Cây có gốc

Định nghĩa

Một cây có gốc là một cây trong đó có một đỉnh được quy ước là gốc, và mỗi cạnh được định hướng trùng với hướng đi của đường đi đơn duy nhất từ gốc tới mỗi đỉnh.

Cho $T = (V, E)$ là một cây có gốc tại a .

- **Phụ huynh** của đỉnh $v \in V \setminus \{a\}$ là đỉnh duy nhất kề với v trên đường duy nhất từ a tới v .
- Nếu u là phụ huynh của v , thì v được gọi là **con** of u .
- Các đỉnh có cùng phụ huynh được gọi là **anh em**.



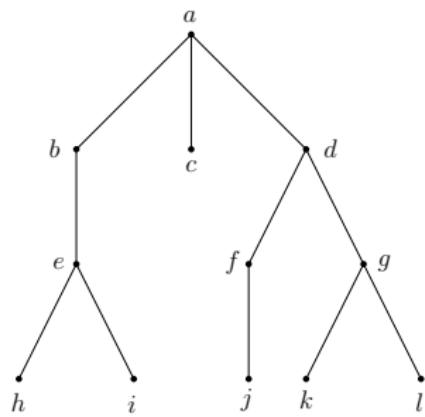
Cây có gốc

Định nghĩa

Một cây có gốc là một cây trong đó có một đỉnh được quy ước là gốc, và mỗi cạnh được định hướng trùng với hướng đi của đường đi đơn duy nhất từ gốc tới mỗi đỉnh.

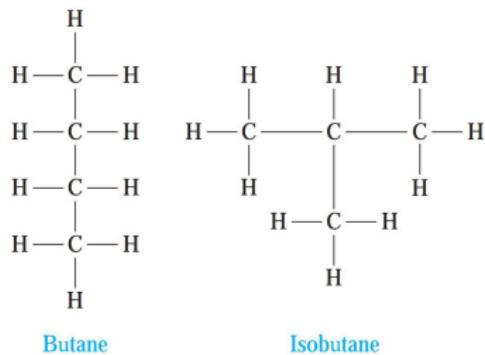
Cho $T = (V, E)$ là một cây có gốc tại a .

- **Tổ tiên** của $v \in V \setminus \{a\}$ là các đỉnh nằm trên đường đi từ a đến phụ huynh của v .
- **Hậu duệ** của $v \in V$ là các đỉnh nhận v là một tổ tiên.
- $v \in V$ được gọi là một **lá** của T nếu nó không có con.
- Các đỉnh có con được gọi là **đỉnh trong**.



Mô hình cây

- Cayley¹¹ phát hiện mô hình cây năm 1857 khi liệt kê các đồng đẳng của C_nH_{2n+2} , còn gọi là hydrocarbon no



- Mô hình cây biểu diễn hệ thống thư mục trong máy tính
- Mô hình cây biểu diễn gia phả

¹¹Arthur Cayley (16.08.1821–26.01.1895): nhà toán học người Anh

Nội dung

- 6.1 Cây và rừng
- 6.2 Cây khung (tối thiểu)
- 6.3 Thuật toán Kruskal
- 6.4 Thuật toán Prim

Cây khung

Định nghĩa

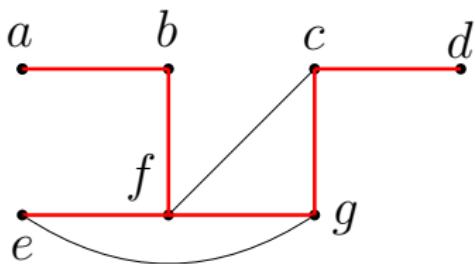
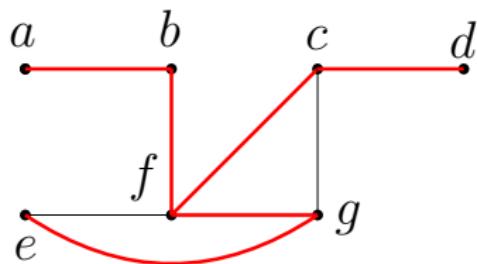
Cho G là một đồ thị đơn. Một *cây khung* của G là một đồ thị con H thỏa mãn H là một cây và H chứa mọi đỉnh của G .

Định lý

Một đồ thị đơn là liên thông khi và chỉ khi nó có một cây khung.

Định lý Cayley

Số cây khung của đồ thị đầy đủ K_n là n^{n-2} .

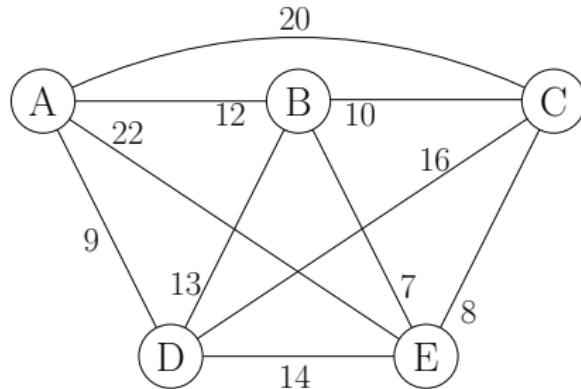


Cây khung tối thiểu: Phát biểu bài toán

Cho trước: một đồ thị vô hướng, liên thông, có trọng số với

- tập hợp đỉnh N , tập hợp cạnh E ,
- trọng số c_{ij} trên mỗi cạnh $\{i,j\} \in E$

Mục tiêu: Tìm một *cây khung tối thiểu* của G , tức là
một cây khung có tổng trọng số các cạnh nhỏ nhất

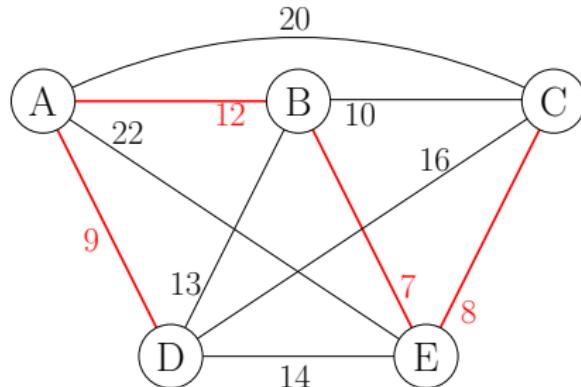


Cây khung tối thiểu: Phát biểu bài toán

Cho trước: một đồ thị vô hướng, liên thông, có trọng số với

- tập hợp đỉnh N , tập hợp cạnh E ,
- trọng số c_{ij} trên mỗi cạnh $\{i,j\} \in E$

Mục tiêu: Tìm một cây khung tối thiểu của G , tức là
một cây khung có tổng trọng số các cạnh nhỏ nhất



Nội dung

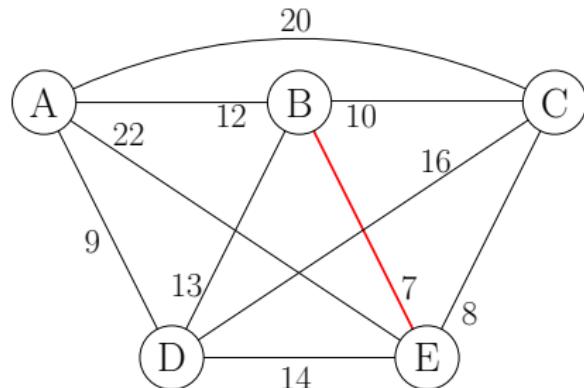
- 6.1 Cây và rừng
- 6.2 Cây khung (tối thiểu)
- 6.3 Thuật toán Kruskal¹²
- 6.4 Thuật toán Prim

¹²Joseph Bernard Kruskal (29.01.1928–19.10.2010): nhà toán học, thống kê học, khoa học máy tính, bác sĩ tâm lý người Mỹ

Thuật toán Kruskal qua một ví dụ

Thực thi theo tinh thần tham lam

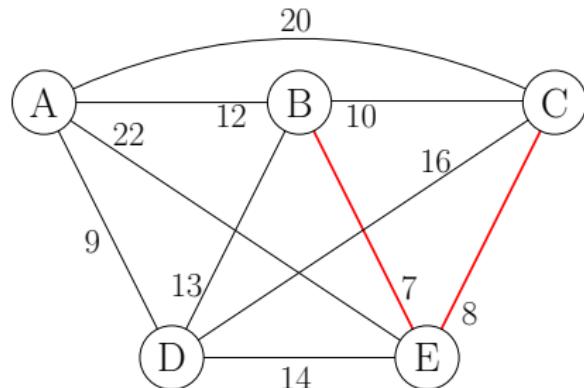
- Bắt đầu với cạnh có trọng số nhỏ nhất
- Từng bước thêm các cạnh thỏa mãn đồng thời
 - chưa được chọn
 - có trọng số nhỏ nhất
 - không tạo thành chu trình với các cạnh đã chọn
- Dừng khi $|N| - 1$ cạnh được chọn



Thuật toán Kruskal qua một ví dụ

Thực thi theo tinh thần tham lam

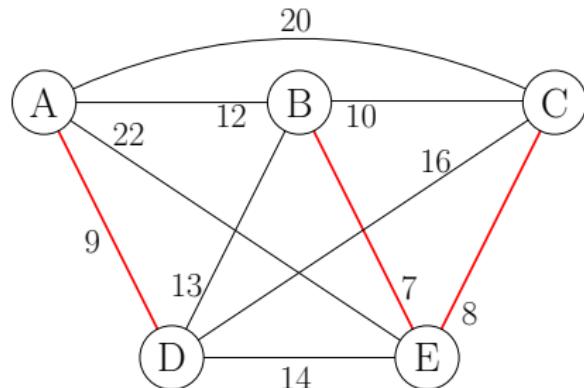
- Bắt đầu với cạnh có trọng số nhỏ nhất
- Từng bước thêm các cạnh thỏa mãn đồng thời
 - chưa được chọn
 - có trọng số nhỏ nhất
 - không tạo thành chu trình với các cạnh đã chọn
- Dừng khi $|N| - 1$ cạnh được chọn



Thuật toán Kruskal qua một ví dụ

Thực thi theo tinh thần tham lam

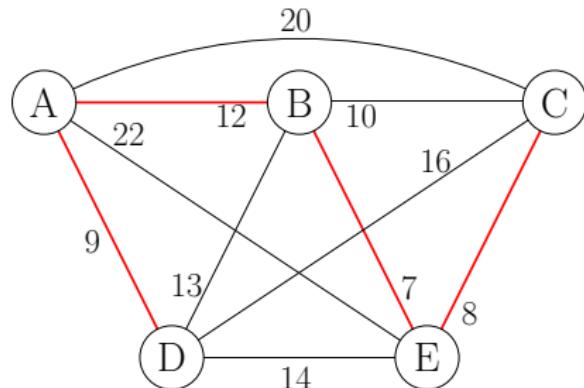
- Bắt đầu với cạnh có trọng số nhỏ nhất
- Từng bước thêm các cạnh thỏa mãn đồng thời
 - chưa được chọn
 - có trọng số nhỏ nhất
 - không tạo thành chu trình với các cạnh đã chọn
- Dừng khi $|N| - 1$ cạnh được chọn



Thuật toán Kruskal qua một ví dụ

Thực thi theo tinh thần tham lam

- Bắt đầu với cạnh có trọng số nhỏ nhất
- Từng bước thêm các cạnh thỏa mãn đồng thời
 - chưa được chọn
 - có trọng số nhỏ nhất
 - không tạo thành chu trình với các cạnh đã chọn
- Dừng khi $|N| - 1$ cạnh được chọn



Thuật toán Kruskal: giả mã

Require: Tìm một cây khung tối thiểu của một đồ thị vô hướng, liên thông, có trọng số $G = (N, E)$

- 1: $T := \emptyset$, $\text{selected}(e) := \text{false}$ với mọi $e \in E$
- 2: **while** $|T| \leq |N| - 1$ **do**
- 3: Tìm cạnh $e \in E$ với trọng số nhỏ nhất thỏa mãn
- 4: $\text{selected}(e) = \text{false}$
- 5: **if** $T \cup \{e\}$ không chứa chu trình đơn **then**
- 6: $\text{selected}(e) := \text{true}$
- 7: $T := T \cup \{e\}$
- 8: **end if**
- 9: **end while**

Nội dung

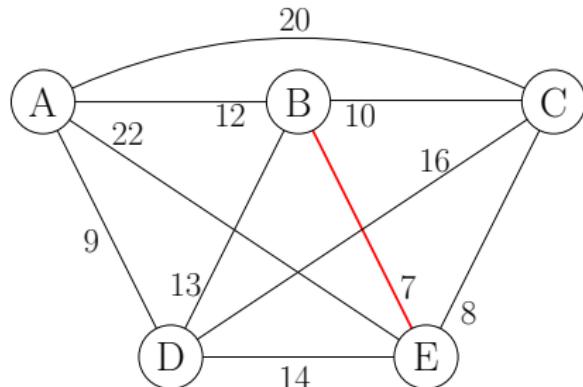
- 6.1 Cây và rừng
- 6.2 Cây khung (tối thiểu)
- 6.3 Thuật toán Kruskal
- 6.4 Thuật toán Prim¹³

¹³Robert Clay Prim (sinh ngày 25.09.1921): nhà toán học và khoa học máy tính người Mỹ

Thuật toán Prim qua một ví dụ

Thực thi theo tinh thần tham lam

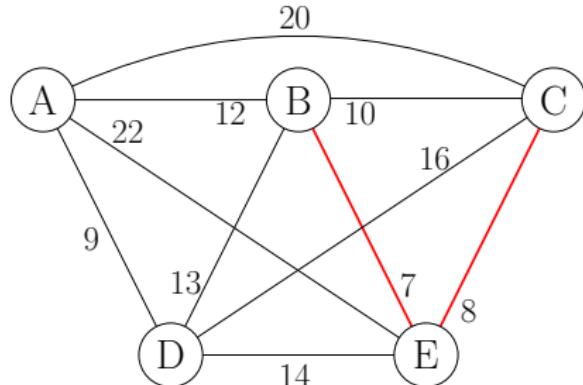
- Bắt đầu với cạnh có trọng số nhỏ nhất
- Từng bước thêm các cạnh thỏa mãn đồng thời
 - chưa được chọn
 - có trọng số nhỏ nhất
 - kề với một đỉnh của một cạnh đã được chọn
 - không tạo thành chu trình với các cạnh đã chọn
- Dừng khi $|N| - 1$ cạnh được chọn



Thuật toán Prim qua một ví dụ

Thực thi theo tinh thần tham lam

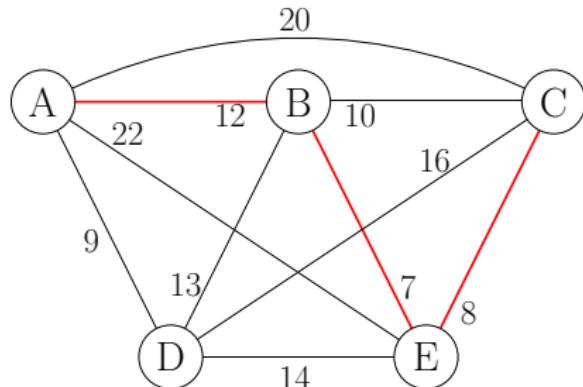
- Bắt đầu với cạnh có trọng số nhỏ nhất
- Từng bước thêm các cạnh thỏa mãn đồng thời
 - chưa được chọn
 - có trọng số nhỏ nhất
 - kề với một đỉnh của một cạnh đã được chọn
 - không tạo thành chu trình với các cạnh đã chọn
- Dừng khi $|N| - 1$ cạnh được chọn



Thuật toán Prim qua một ví dụ

Thực thi theo tinh thần tham lam

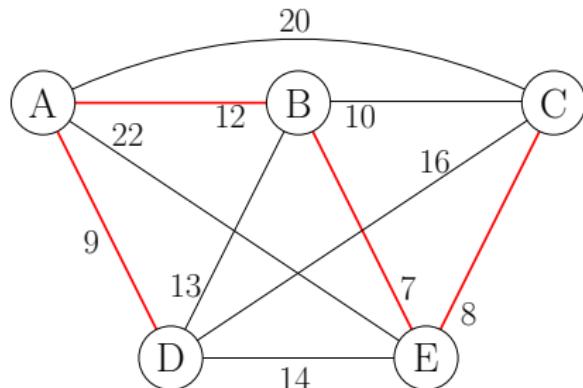
- Bắt đầu với cạnh có trọng số nhỏ nhất
- Từng bước thêm các cạnh thỏa mãn đồng thời
 - chưa được chọn
 - có trọng số nhỏ nhất
 - kề với một đỉnh của một cạnh đã được chọn
 - không tạo thành chu trình với các cạnh đã chọn
- Dừng khi $|N| - 1$ cạnh được chọn



Thuật toán Prim qua một ví dụ

Thực thi theo tinh thần tham lam

- Bắt đầu với cạnh có trọng số nhỏ nhất
- Từng bước thêm các cạnh thỏa mãn đồng thời
 - chưa được chọn
 - có trọng số nhỏ nhất
 - kề với một đỉnh của một cạnh đã được chọn
 - không tạo thành chu trình với các cạnh đã chọn
- Dừng khi $|N| - 1$ cạnh được chọn



Thuật toán Prim: giả mã

Require: Tìm một cây khung tối thiểu của một đồ thị vô hướng, liên thông, có trọng số $G = (N, E)$

- 1: $T := \emptyset$, $\text{selected}(e) := \text{false}$ với mọi $e \in E$
- 2: **while** $|T| \leq |N| - 1$ **do**
- 3: Tìm $e \in E$ kề với một đỉnh của T , với trọng số nhỏ nhất và $\text{selected}(e) = \text{false}$
- 4: **if** $T \cup \{e\}$ không chứa chu trình đơn **then**
- 5: $\text{selected}(e) := \text{true}$
- 6: $T := T \cup \{e\}$
- 7: **end if**
- 8: **end while**

Về thuật toán Kruskal và thuật toán Prim

- Thuật toán Prim được giới thiệu từ 1930 bởi Jarník¹⁴, và được Prim độc lập giới thiệu năm 1957
- Thuật toán Kruskal được giới thiệu năm 1956
- Độ phức tạp tính toán:
 - Thuật toán Kruskal: $O(|E| \log |E|)$
 - Thuật toán Prim: $O(|E| \log |N|)$

Thuật toán Kruskal thường được áp dụng cho đồ thị *thưa*

¹⁴Vojtěch Jarník (22.12.1897–22.09.1970): nhà toán học người Séc

Mục lục

1 Một số khái niệm cơ bản

2 Duyệt đồ thị

3 Đồ thị Euler

4 Đồ thị Hamilton

5 Đồ thị phẳng

6 Cây và cây khung

7 Đường đi ngắn nhất

Nội dung

7.1 Phát biểu bài toán

7.2 Thuật toán Dijkstra

7.3 Thuật toán Ford-Bellman

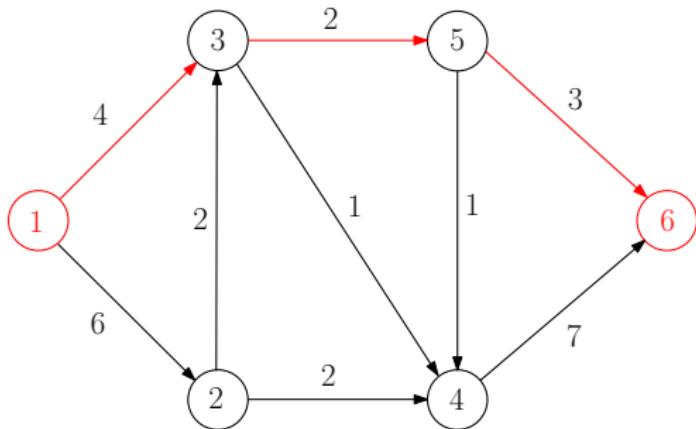
Phát biểu bài toán

Cho trước:

- một đồ thị có hướng $G = (N, A)$
- độ dài c_{ij} cho mỗi cung $(i, j) \in A$
- hai đỉnh phân biệt $s, t \in N$

Định nghĩa: Độ dài một đường đi = tổng độ dài các cung trên đường đi

Mục tiêu: Tìm (một) đường đi ngắn nhất từ s tới t



Nội dung

7.1 Phát biểu bài toán

7.2 Thuật toán Dijkstra¹⁵

7.3 Thuật toán Bellman-Ford

¹⁵ Edsger Wybe Dijkstra (11.05.1930–06.08.2002): nhà khoa học máy tính, kỹ sư phần mềm, kỹ sư hệ thống, nhà luận khoa học người Hà Lan. Ông là một trong những người tiên phong trong lĩnh vực khoa học máy tính.

Giả thiết

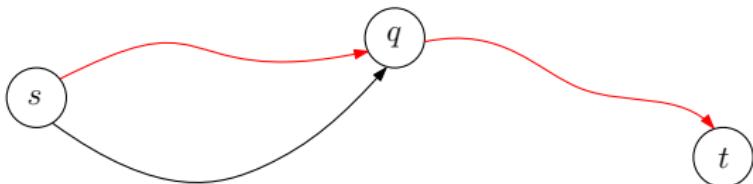
- Mọi cung có độ dài không âm
- Có đường đi từ s tới tất cả các đỉnh khác

Thuật toán Dijkstra: nguyên lý

- Tìm đường đi ngắn nhất từ s tới **tất cả các đỉnh khác**
- Ý tưởng dựa trên

Tính chất 1

Nếu $s = i_1 - i_2 - \dots - i_h = t$ là một đường đi ngắn nhất từ s tới t , thì với mọi $q = 2, \dots, h-1$ ta có $s = i_1 - i_2 - \dots - i_q$ là một đường đi ngắn nhất từ s tới i_q .



Tính chất 2

Gọi $d(i)$ là độ dài đường đi ngắn nhất từ s tới i . Khi đó, P là một đường đi ngắn nhất từ s tới j khi và chỉ khi $d(j) = d(i) + c_{ij}$ với mọi $(i, j) \in P$.

Thuật toán Dijkstra qua một ví dụ

Gán nhãn:

- Mỗi đỉnh i có một nhãn (*khoảng cách*) $d(i)$
- Đỉnh vĩnh cửu*: nhãn = độ dài đường ngắn nhất từ nguồn
- Đỉnh tạm thời*:
nhãn = cận trên của độ dài đường đi ngắn nhất từ nguồn

Minh họa thuật toán:

- Bước 0: nhãn tạm thời $d(s) = 0$,
 $d(i) = \infty \forall i \neq s$
- Mỗi bước:
 - Chọn đỉnh:
 $i :=$ đỉnh có nhãn tạm thời nhỏ nhất, chuyển thành đỉnh vĩnh cửu
 - Cập nhật khoảng cách:
 Nếu $(i, j) \in A$ và $d(j) > d(i) + c_{ij}$,
 thì đặt $d(j) := d(i) + c_{ij}$
- Dừng khi mọi đỉnh là vĩnh cửu

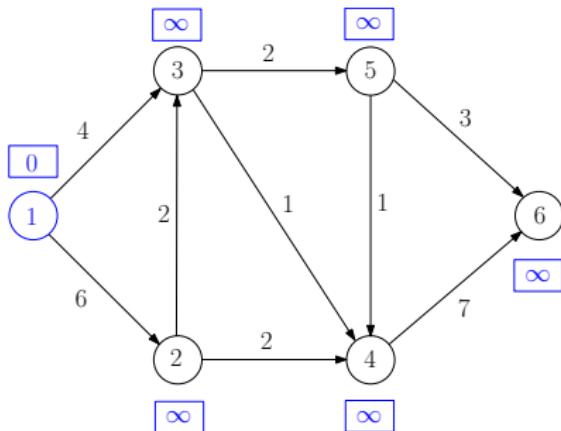
Thuật toán Dijkstra qua một ví dụ

Gán nhãn:

- Mỗi đỉnh i có một nhãn (*khoảng cách*) $d(i)$
- Đỉnh vĩnh cửu*: nhãn = độ dài đường ngắn nhất từ nguồn
- Đỉnh tạm thời*:
nhãn = cận trên của độ dài đường đi ngắn nhất từ nguồn

Minh họa thuật toán:

- Bước 0: nhãn tạm thời $d(s) = 0$,
 $d(i) = \infty \forall i \neq s$
- Mỗi bước:
 - Chọn đỉnh:
 $i :=$ đỉnh có nhãn tạm thời nhỏ nhất, chuyển thành đỉnh vĩnh cửu
 - Cập nhật khoảng cách:
 Nếu $(i, j) \in A$ và $d(j) > d(i) + c_{ij}$,
 thì đặt $d(j) := d(i) + c_{ij}$
- Dừng khi mọi đỉnh là vĩnh cửu



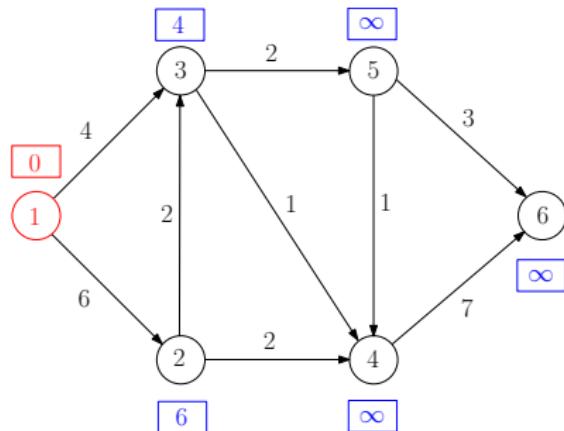
Thuật toán Dijkstra qua một ví dụ

Gán nhãn:

- Mỗi đỉnh i có một nhãn (*khoảng cách*) $d(i)$
- Đỉnh vĩnh cửu*: nhãn = độ dài đường ngắn nhất từ nguồn
- Đỉnh tạm thời*:
nhãn = cận trên của độ dài đường đi ngắn nhất từ nguồn

Minh họa thuật toán:

- Bước 0: nhãn tạm thời $d(s) = 0$,
 $d(i) = \infty \forall i \neq s$
- Mỗi bước:
 - Chọn đỉnh:
 $i :=$ đỉnh có nhãn tạm thời nhỏ nhất, chuyển thành đỉnh vĩnh cửu
 - Cập nhật khoảng cách:
 Nếu $(i, j) \in A$ và $d(j) > d(i) + c_{ij}$,
 thì đặt $d(j) := d(i) + c_{ij}$
- Dừng khi mọi đỉnh là vĩnh cửu



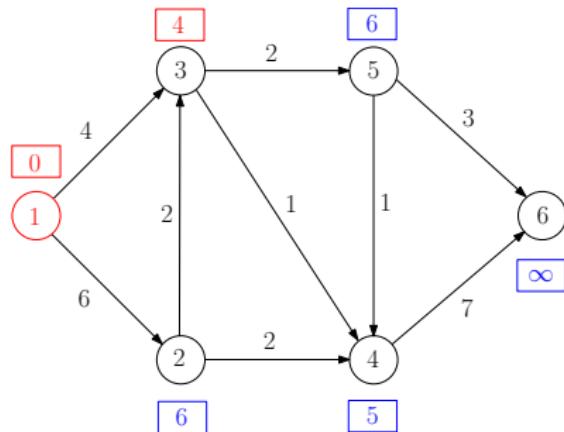
Thuật toán Dijkstra qua một ví dụ

Gán nhãn:

- Mỗi đỉnh i có một nhãn (*khoảng cách*) $d(i)$
- Đỉnh vĩnh cửu*: nhãn = độ dài đường ngắn nhất từ nguồn
- Đỉnh tạm thời*:
nhãn = cận trên của độ dài đường đi ngắn nhất từ nguồn

Minh họa thuật toán:

- Bước 0: nhãn tạm thời $d(s) = 0$,
 $d(i) = \infty \forall i \neq s$
- Mỗi bước:
 - Chọn đỉnh:
 $i :=$ đỉnh có nhãn tạm thời nhỏ nhất, chuyển thành đỉnh vĩnh cửu
 - Cập nhật khoảng cách:
 Nếu $(i, j) \in A$ và $d(j) > d(i) + c_{ij}$,
 thì đặt $d(j) := d(i) + c_{ij}$
- Dừng khi mọi đỉnh là vĩnh cửu



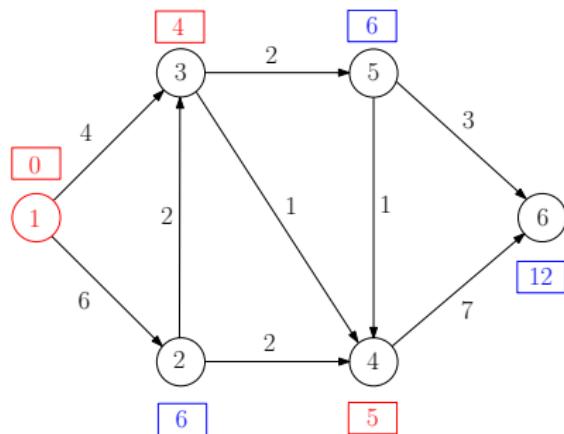
Thuật toán Dijkstra qua một ví dụ

Gán nhãn:

- Mỗi đỉnh i có một nhãn (*khoảng cách*) $d(i)$
- Đỉnh vĩnh cửu*: nhãn = độ dài đường ngắn nhất từ nguồn
- Đỉnh tạm thời*:
nhãn = cận trên của độ dài đường đi ngắn nhất từ nguồn

Minh họa thuật toán:

- Bước 0: nhãn tạm thời $d(s) = 0$,
 $d(i) = \infty \forall i \neq s$
- Mỗi bước:
 - Chọn đỉnh:
 $i :=$ đỉnh có nhãn tạm thời nhỏ nhất, chuyển thành đỉnh vĩnh cửu
 - Cập nhật khoảng cách:
 Nếu $(i, j) \in A$ và $d(j) > d(i) + c_{ij}$,
 thì đặt $d(j) := d(i) + c_{ij}$
- Dừng khi mọi đỉnh là vĩnh cửu



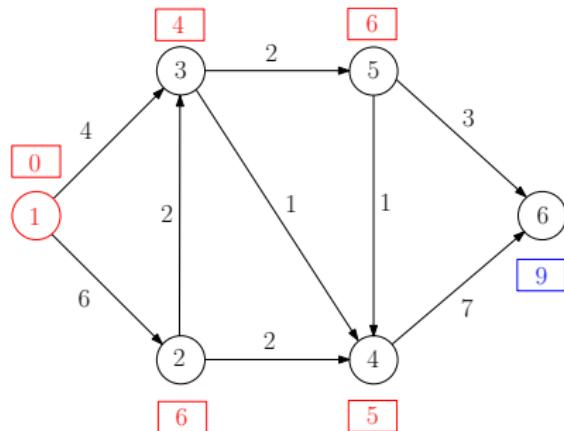
Thuật toán Dijkstra qua một ví dụ

Gán nhãn:

- Mỗi đỉnh i có một nhãn (*khoảng cách*) $d(i)$
- Đỉnh vĩnh cửu*: nhãn = độ dài đường ngắn nhất từ nguồn
- Đỉnh tạm thời*:
nhãn = cận trên của độ dài đường đi ngắn nhất từ nguồn

Minh họa thuật toán:

- Bước 0: nhãn tạm thời $d(s) = 0$,
 $d(i) = \infty \forall i \neq s$
- Mỗi bước:
 - Chọn đỉnh:
 $i :=$ đỉnh có nhãn tạm thời nhỏ nhất, chuyển thành đỉnh vĩnh cửu
 - Cập nhật khoảng cách:
 Nếu $(i, j) \in A$ và $d(j) > d(i) + c_{ij}$,
 thì đặt $d(j) := d(i) + c_{ij}$
- Dừng khi mọi đỉnh là vĩnh cửu



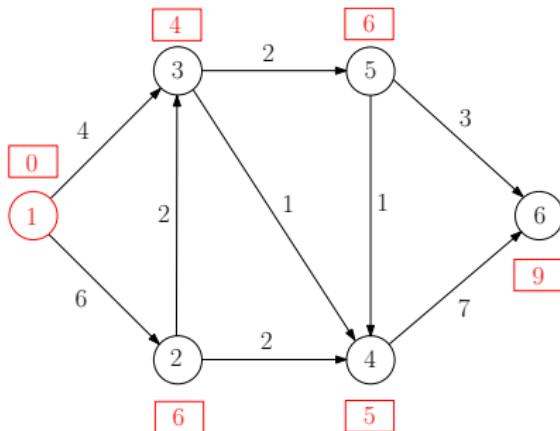
Thuật toán Dijkstra qua một ví dụ

Gán nhãn:

- Mỗi đỉnh i có một nhãn (*khoảng cách*) $d(i)$
- Đỉnh vĩnh cửu*: nhãn = độ dài đường ngắn nhất từ nguồn
- Đỉnh tạm thời*:
nhãn = cận trên của độ dài đường đi ngắn nhất từ nguồn

Minh họa thuật toán:

- Bước 0: nhãn tạm thời $d(s) = 0$,
 $d(i) = \infty \forall i \neq s$
- Mỗi bước:
 - Chọn đỉnh:
 $i :=$ đỉnh có nhãn tạm thời nhỏ nhất, chuyển thành đỉnh vĩnh cửu
 - Cập nhật khoảng cách:
 Nếu $(i, j) \in A$ và $d(j) > d(i) + c_{ij}$,
 thì đặt $d(j) := d(i) + c_{ij}$
- Dừng khi mọi đỉnh là vĩnh cửu



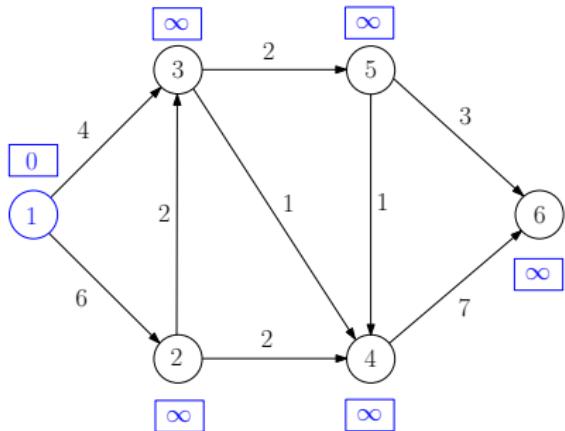
Thuật toán Dijkstra: giả mă

Require: Đồ thị có hướng $G = (N, A)$ với độ dài cạnh không âm, đỉnh nguồn $s \in N$, tìm đường ngắn nhất từ s tới các đỉnh khác

- 1: $S := \emptyset, \bar{S} := N, d(s) := 0, pred(s) := s, d(i) = \infty \forall i \in N$
- 2: **while** $|S| < |N|$ **do**
- 3: $i := \operatorname{argmin}\{d(j) \mid j \in \bar{S}\}$
- 4: $S := S \cup \{i\}, \bar{S} := \bar{S} \setminus \{i\}$
- 5: **for** mỗi $j \in N$ với $(i, j) \in A$ **do**
- 6: **if** $d(j) > d(i) + c_{ij}$ **then**
- 7: $d(j) := d(i) + c_{ij}, \quad pred(j) := i$
- 8: **end if**
- 9: **end for**
- 10: **end while**

Độ phức tạp tính toán: $O(|N|^2)$

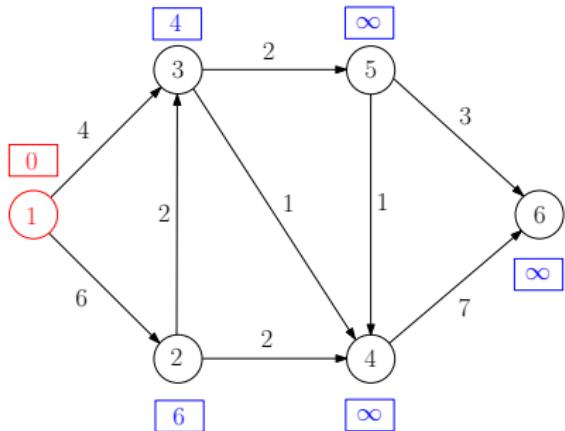
Thuật toán Dijkstra: bảng nhän



- đỏ: vĩnh cửu
- xanh: tạm thời
- $d(i)$, $pred(i)$

Bước	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
0	0	infinity	infinity	infinity	infinity	infinity

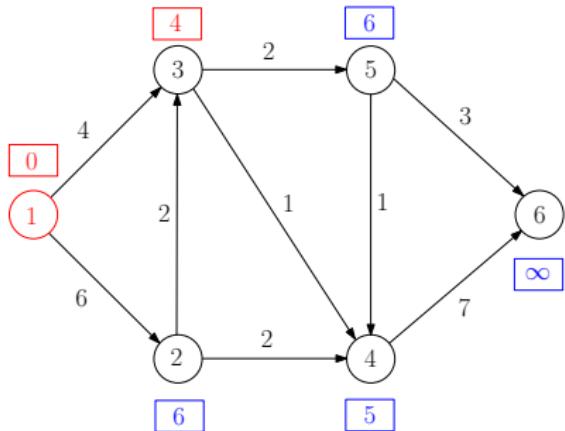
Thuật toán Dijkstra: bảng nhän



- đỏ: vĩnh cửu
- xanh: tạm thời
- $d(i)$, $pred(i)$

Bước	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
0	0	∞	∞	∞	∞	∞
1	0	6, 1	4, 1	∞	∞	∞

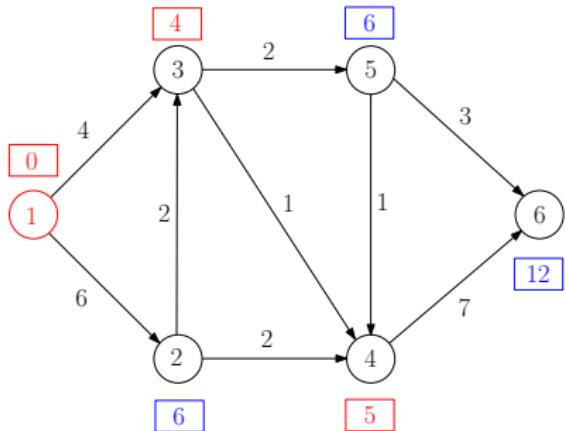
Thuật toán Dijkstra: bảng nhãnh



- đỏ: vĩnh cửu
- xanh: tạm thời
- $d(i)$, $pred(i)$

Bước	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
0	0	∞	∞	∞	∞	∞
1	0	6, 1	4, 1	∞	∞	∞
2	0	6, 1	4, 1	5, 3	6, 3	∞

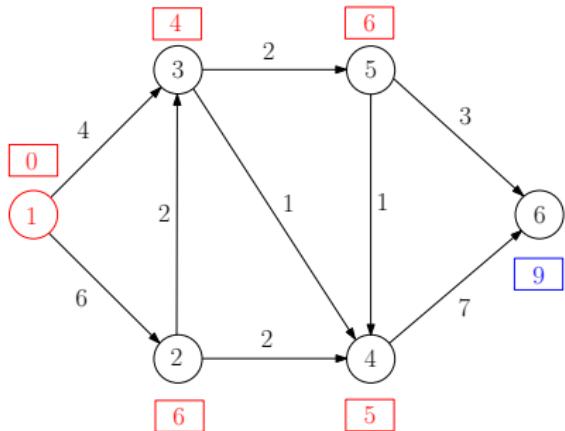
Thuật toán Dijkstra: bảng nhãnh



- đỏ: vĩnh cửu
- xanh: tạm thời
- $d(i)$, $pred(i)$

Bước	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
0	0	∞	∞	∞	∞	∞
1	0	6, 1	4, 1	∞	∞	∞
2	0	6, 1	4, 1	5, 3	6, 3	∞
3	0	6, 1	4, 1	5, 3	6, 3	12, 4

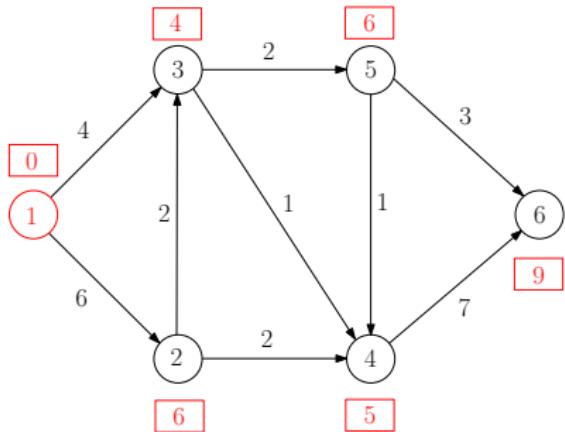
Thuật toán Dijkstra: bảng nhãnh



- đỏ: vĩnh cửu
- xanh: tạm thời
- $d(i)$, $pred(i)$

Bước	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
0	0	∞	∞	∞	∞	∞
1	0	6, 1	4, 1	∞	∞	∞
2	0	6, 1	4, 1	5, 3	6, 3	∞
3	0	6, 1	4, 1	5, 3	6, 3	12, 4
4	0	6, 1	4, 1	5, 3	6, 3	9, 5

Thuật toán Dijkstra: bảng nhän



- đỏ: vĩnh cửu
- xanh: tạm thời
- $d(i)$, $pred(i)$

Bước	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
0	0	∞	∞	∞	∞	∞
1	0	6, 1	4, 1	∞	∞	∞
2	0	6, 1	4, 1	5, 3	6, 3	∞
3	0	6, 1	4, 1	5, 3	6, 3	12, 4
4	0	6, 1	4, 1	5, 3	6, 3	9, 5
6	0	6, 1	4, 1	5, 3	6, 3	9, 5

Nội dung

7.1 Phát biểu bài toán

7.2 Thuật toán Dijkstra

7.3 Thuật toán Bellman¹⁶-Ford¹⁷

¹⁶Richard Ernest Bellman (26.08.1920– 19.03.1984): nhà toán học người Mỹ

¹⁷Lester Randolph Ford Jr. (23.09.1927–26.02.2017): nhà toán học người Mỹ

Giả thiết

- Độ dài cung có thể là số âm
- Đồ thị không chứa chu trình có độ dài âm
- Có đường đi từ s tới các đỉnh khác

Thuật toán Bellman-Ford: nguyên lý

Quy hoạch động

$$d^k(j) := \min\{d^{k-1}(j), \min_{(i,j) \in A} \{d^{k-1}(i) + c_{ij}\}\}$$

trong đó $d^k(j)$ là độ dài một đường ngắn nhất từ s tới j với $\leq k$ cung

Các bước tính toán

- Đầu tiên, tính khoảng cách của các đường đi ngắn nhất từ s với ≤ 1 cung
- Tiếp theo, tính khoảng cách của các đường đi ngắn nhất từ s với ≤ 2 cung
- ...
- Cuối cùng, tính khoảng cách của các đường đi ngắn nhất từ s với $\leq |V| - 1$ cung

(có nhiều nhất $|V| - 1$ cung trên một đường đi đơn trong đồ thị)

Thuật toán Bellman-Ford qua một ví dụ

Gán nhãn:

- Tại mỗi bước k , mỗi đỉnh i có một *nhãn* (*khoảng cách*) $d^k(i)$

Minh họa thuật toán:

- Bước 0: $d^0(s) = 0, d^0(i) = \infty \forall i \neq s$
- Bước $k = 1, \dots, |V| - 1$:
 - Với mỗi $(i, j) \in A$ với $j \neq s$:
 nếu $d^{k-1}(j) > d^{k-1}(i) + c_{ij}$,
 thì đặt $d^k(j) := d^{k-1}(i) + c_{ij}$,
 ngược lại đặt $d^k(j) := d^{k-1}(j)$

Chú ý:

Nếu với k nào đó ta có
 $d^k(j) = d^{k-1}(j) \forall j \in N$,
thì có thể dừng tại bước k

Thuật toán Bellman-Ford qua một ví dụ

Gán nhãn:

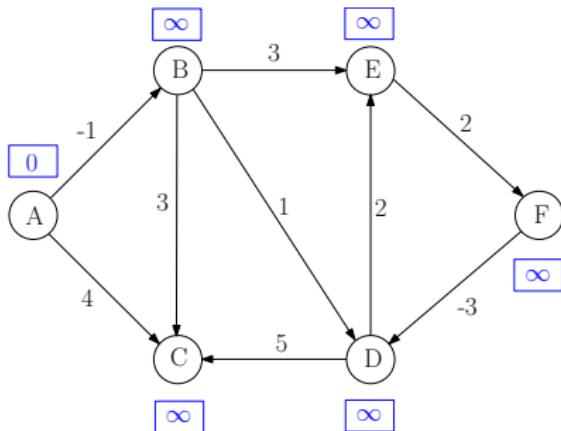
- Tại mỗi bước k , mỗi đỉnh i có một *nhãn* (khoảng cách) $d^k(i)$

Mô hình thuật toán:

- Bước 0: $d^0(s) = 0, d^0(i) = \infty \forall i \neq s$
- Bước $k = 1, \dots, |V| - 1$:
 - Với mỗi $(i, j) \in A$ với $j \neq s$:
 nếu $d^{k-1}(j) > d^{k-1}(i) + c_{ij}$,
 thì đặt $d^k(j) := d^{k-1}(i) + c_{ij}$,
 ngược lại đặt $d^k(j) := d^{k-1}(j)$

Chú ý:

Nếu với k nào đó ta có
 $d^k(j) = d^{k-1}(j) \forall j \in N$,
thì có thể dừng tại bước k



Thuật toán Bellman-Ford qua một ví dụ

Gán nhãn:

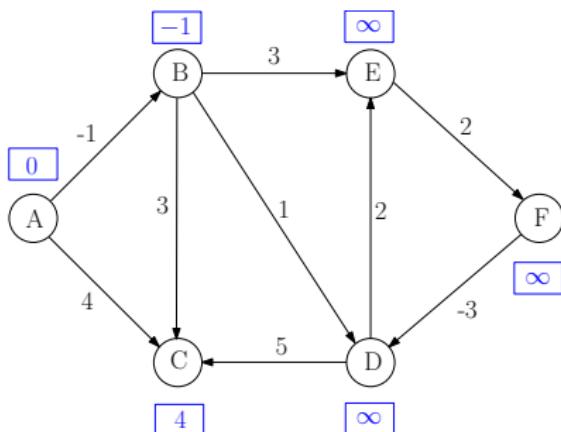
- Tại mỗi bước k , mỗi đỉnh i có một *nhãn* (khoảng cách) $d^k(i)$

Mô hình thuật toán:

- Bước 0: $d^0(s) = 0, d^0(i) = \infty \forall i \neq s$
- Bước $k = 1, \dots, |V| - 1$:
 - Với mỗi $(i, j) \in A$ với $j \neq s$:
 nếu $d^{k-1}(j) > d^{k-1}(i) + c_{ij}$,
 thì đặt $d^k(j) := d^{k-1}(i) + c_{ij}$,
 ngược lại đặt $d^k(j) := d^{k-1}(j)$

Chú ý:

Nếu với k nào đó ta có
 $d^k(j) = d^{k-1}(j) \forall j \in N$,
thì có thể dừng tại bước k



Thuật toán Bellman-Ford qua một ví dụ

Gán nhãn:

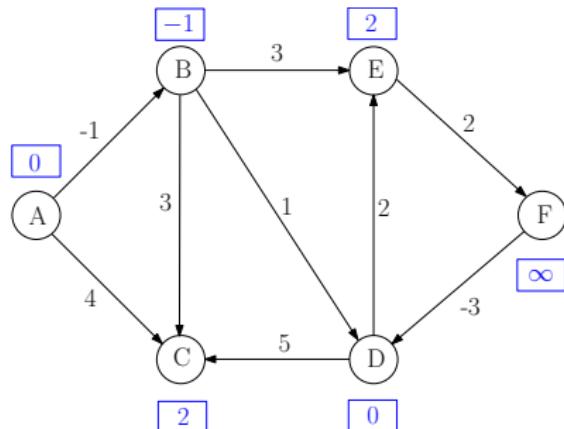
- Tại mỗi bước k , mỗi đỉnh i có một *nhãn* (khoảng cách) $d^k(i)$

Mô hình thuật toán:

- Bước 0: $d^0(s) = 0, d^0(i) = \infty \forall i \neq s$
- Bước $k = 1, \dots, |V| - 1$:
 - Với mỗi $(i, j) \in A$ với $j \neq s$:
 nếu $d^{k-1}(j) > d^{k-1}(i) + c_{ij}$,
 thì đặt $d^k(j) := d^{k-1}(i) + c_{ij}$,
 ngược lại đặt $d^k(j) := d^{k-1}(j)$

Chú ý:

Nếu với k nào đó ta có
 $d^k(j) = d^{k-1}(j) \forall j \in N$,
thì có thể dừng tại bước k



Thuật toán Bellman-Ford qua một ví dụ

Gán nhãn:

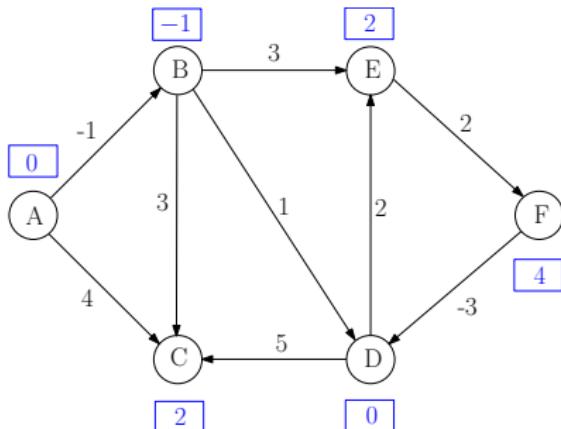
- Tại mỗi bước k , mỗi đỉnh i có một *nhãn* (khoảng cách) $d^k(i)$

Mô hình thuật toán:

- Bước 0: $d^0(s) = 0, d^0(i) = \infty \forall i \neq s$
- Bước $k = 1, \dots, |V| - 1$:
 - Với mỗi $(i, j) \in A$ với $j \neq s$:
 nếu $d^{k-1}(j) > d^{k-1}(i) + c_{ij}$,
 thì đặt $d^k(j) := d^{k-1}(i) + c_{ij}$,
 ngược lại đặt $d^k(j) := d^{k-1}(j)$

Chú ý:

Nếu với k nào đó ta có
 $d^k(j) = d^{k-1}(j) \forall j \in N$,
thì có thể dừng tại bước k



Thuật toán Bellman-Ford qua một ví dụ

Gán nhãn:

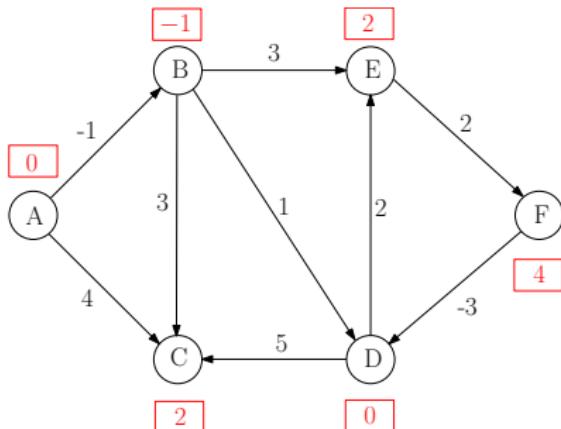
- Tại mỗi bước k , mỗi đỉnh i có một *nhãn* (khoảng cách) $d^k(i)$

Mô hình thuật toán:

- Bước 0: $d^0(s) = 0, d^0(i) = \infty \forall i \neq s$
- Bước $k = 1, \dots, |V| - 1$:
 - Với mỗi $(i, j) \in A$ với $j \neq s$:
 nếu $d^{k-1}(j) > d^{k-1}(i) + c_{ij}$,
 thì đặt $d^k(j) := d^{k-1}(i) + c_{ij}$,
 ngược lại đặt $d^k(j) := d^{k-1}(j)$

Chú ý:

Nếu với k nào đó ta có
 $d^k(j) = d^{k-1}(j) \forall j \in N$,
thì có thể dừng tại bước k



Thuật toán Bellman-Ford: giả mă

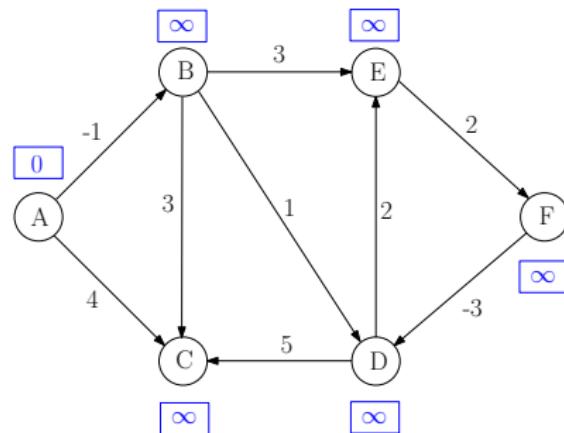
Require: Đồ thị có hướng $G = (N, A)$ với độ dài cạnh không âm, đỉnh nguồn $s \in N$, tìm đường ngắn nhất từ s tới các đỉnh khác

```

1:  $d(s) := 0, pred(s) := s, d(i) = \infty \forall i \in N$ 
2:  $k := 1$ 
3: while  $k \leq |N| - 1$  do
4:   for mỗi  $(i, j) \in A$  do
5:     if  $d(j) > d(i) + c_{ij}$  then
6:        $d(j) := d(i) + c_{ij}, \quad pred(j) := i$ 
7:     end if
8:   end for
9:    $k := k + 1$ 
10: end while
```

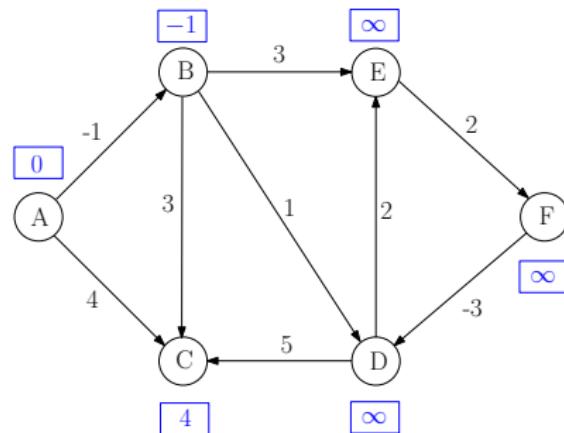
Độ phức tạp tính toán: $O(|N||A|)$

Thuật toán Bellman-Ford: bảng nhän



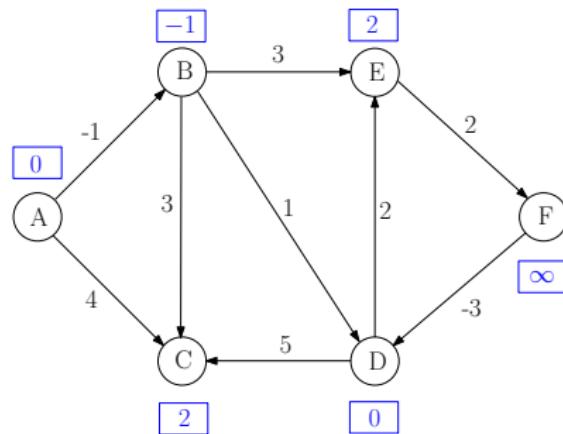
Bước	Đỉnh A	Đỉnh B	Đỉnh C	Đỉnh D	Đỉnh E	Đỉnh F
0	0	∞	∞	∞	∞	∞

Thuật toán Bellman-Ford: bảng nhãm



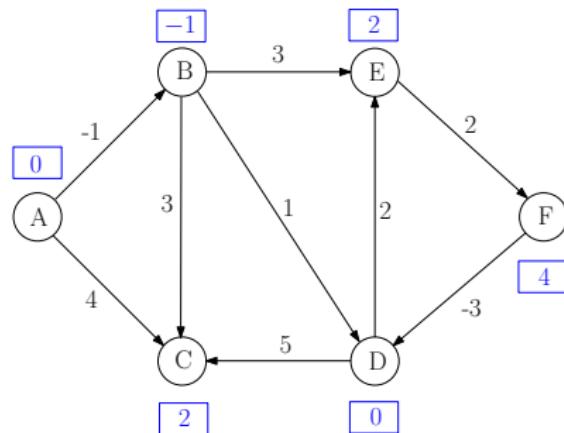
Bước	Đỉnh A	Đỉnh B	Đỉnh C	Đỉnh D	Đỉnh E	Đỉnh F
0	0	∞	∞	∞	∞	∞
1	0	-1, A	4, A	∞	∞	∞

Thuật toán Bellman-Ford: bảng nhãm



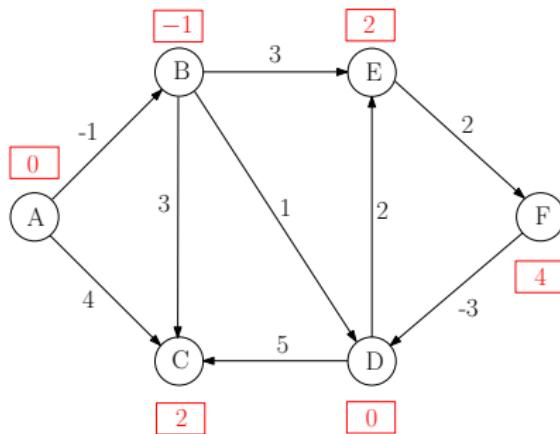
Bước	Đỉnh A	Đỉnh B	Đỉnh C	Đỉnh D	Đỉnh E	Đỉnh F
0	0	∞	∞	∞	∞	∞
1	0	-1, A	4, A	∞	∞	∞
2	0	-1, A	2, B	0, B	2, B	∞

Thuật toán Bellman-Ford: bảng nhãm



Bước	Đỉnh A	Đỉnh B	Đỉnh C	Đỉnh D	Đỉnh E	Đỉnh F
0	0	∞	∞	∞	∞	∞
1	0	-1, A	4, A	∞	∞	∞
2	0	-1, A	2, B	0, B	2, B	∞
3	0	-1, A	2, B	0, B	2, B	4, E

Thuật toán Bellman-Ford: bảng nhãm



Bước	Đỉnh A	Đỉnh B	Đỉnh C	Đỉnh D	Đỉnh E	Đỉnh F
0	0	∞	∞	∞	∞	∞
1	0	-1, A	4, A	∞	∞	∞
2	0	-1, A	2, B	0, B	2, B	∞
3	0	-1, A	2, B	0, B	2, B	4, E
3	0	-1, A	2, B	0, B	2, B	4, E

Trân trọng cảm ơn!