CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Bài 09: Một số thuật toán đồ thị

Nguyễn Thị Tâm
nguyenthitam.hus@gmail.com

Ngày 18 tháng 4 năm 2025

Nội dung

Nội dung

Định nghĩa

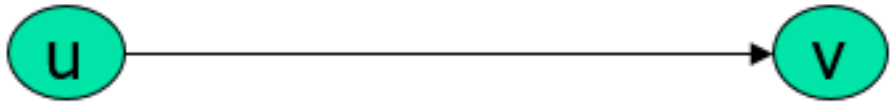
Đồ thị

Đồ thị G kí hiệu $G = (V, E)$ là cấu trúc rời rạc bao gồm hai tập

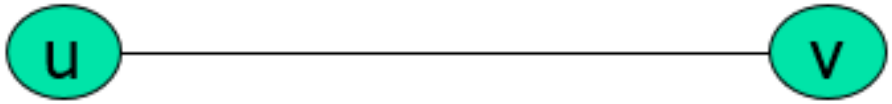
- Tập đỉnh $V(G)$ là tập hữu hạn khác rỗng
- Tập cạnh $E(V)$ là tập hữu hạn có thể là tập rỗng gồm các cặp $(u, v), u, v \in V$.

Các loại cạnh

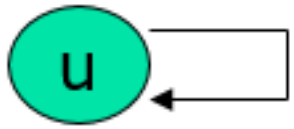
- Có hướng



- Vô hướng



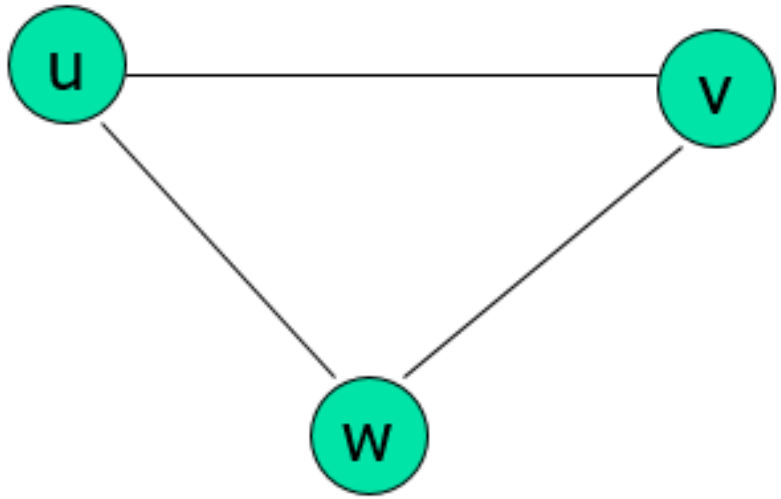
- Khuyên



Các loại đồ thị

Đồ thị vô hướng

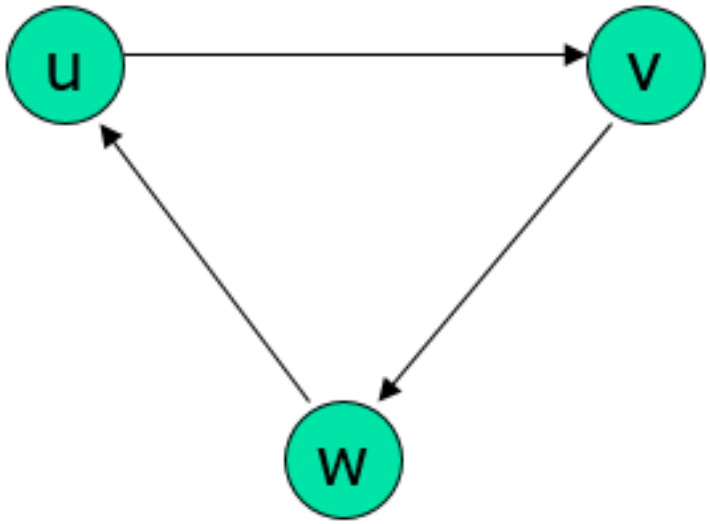
- Đơn (đa) đồ thị vô hướng $G = (V, E)$ là cặp gồm
  - Tập đỉnh V là tập hữu hạn các phần tử, các phần tử gọi là đỉnh
  - Tập cạnh E là tập các bộ không có thứ tự dạng $(u, v), u, v \in V, u \neq v$. Các phần tử của E được gọi là các cạnh.
- Ví dụ đồ thị dưới đây gồm:
  - Tập đỉnh $V = \{u, v, w\}$
  - Tập cạnh $E = \{(u, v), (u, w), (v, w)\}$



Các loại đồ thị

Đồ thị có hướng

- Đơn (đa) đồ thị có hướng $G = (V, E)$ là cặp gồm
  - Tập đỉnh V là tập hữu hạn các phần tử, các phần tử gọi là đỉnh
  - Tập cạnh E là tập các bộ không có thứ tự dạng $(u, v), u, v \in V, u \neq v$. Các phần tử của E được gọi là các cung hoặc các cạnh có hướng.
- Ví dụ đồ thị dưới đây gồm:
  - Tập đỉnh $V = \{u, v, w\}$
  - Tập cạnh $E = \{(u, v), (v, w), (w, u)\}$



Các thuật ngữ

Đồ thị vô hướng

- Đỉnh u và v được gọi là hai đỉnh kề/lân cận nhau nếu (u, v) là một cạnh của đồ thị
- Cạnh (u, v) gọi là liên thuộc với hai đỉnh u và v
- Đỉnh u, v được gọi là các đầu mút của cạnh (u, v)

Các thuật ngữ

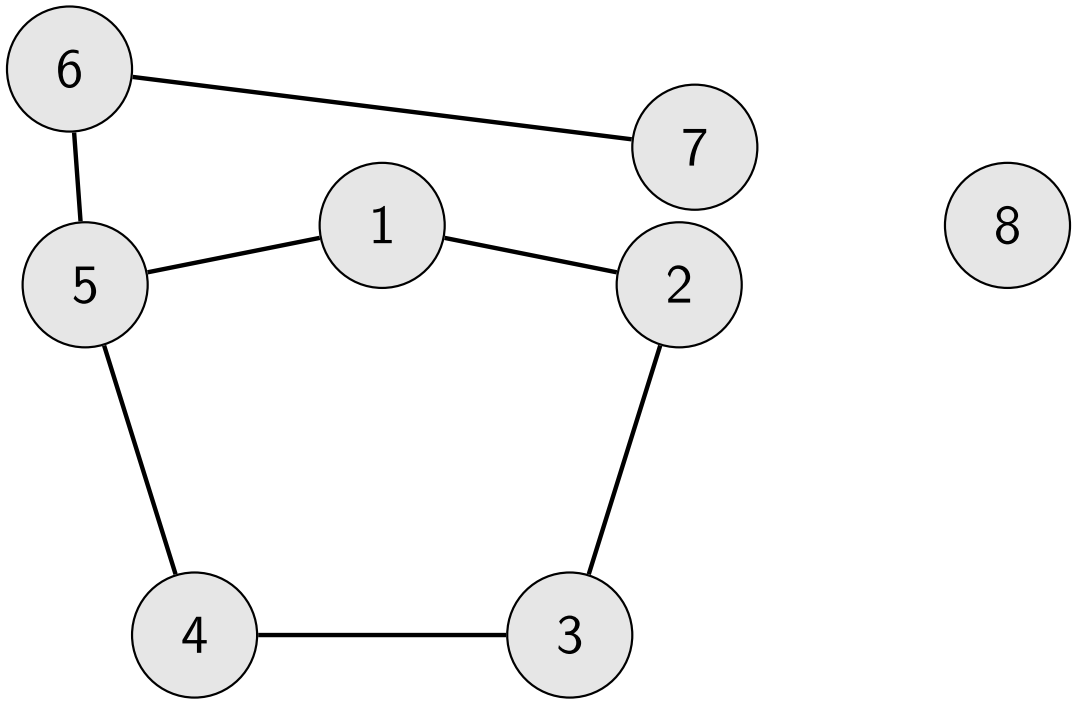
Đồ thị có hướng

- Đỉnh u và v được gọi là hai đỉnh kề/lân cận nhau nếu (u, v) là một cạnh của đồ thị
- Cạnh $e = (u, v)$, e đi ra khỏi u và đi vào v
- Đỉnh đầu của e là u, đỉnh cuối của e là v

Các thuật ngữ

Bậc của đỉnh

Giả sử đồ thị $G = (V, E)$ là đồ thị vô hướng, $v \in V$ là một đỉnh nào đó. Bậc của đỉnh v kí hiệu là $deg(v)$ là số cạnh kề với định đó. Đỉnh có bậc 0 được gọi là đỉnh cô lập (*isolated*). Đỉnh bậc 1 được gọi là đỉnh treo (*pendant*)



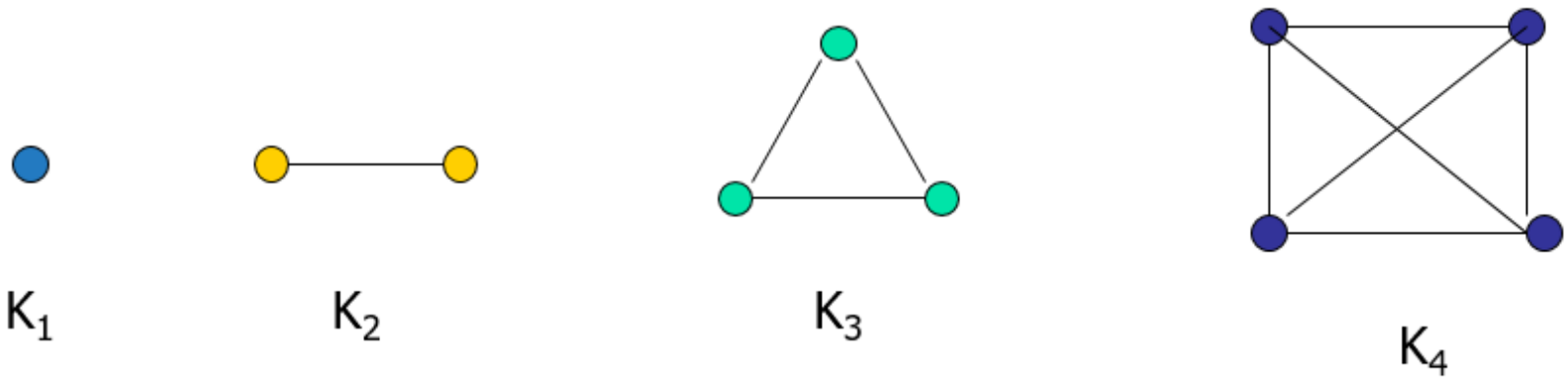
Hình 1: Đồ thị với 8 đỉnh, trong đó đỉnh 8 là đỉnh cô lập và đỉnh 7 là đỉnh treo.

Một số đồ thị đặc biệt

Đồ thị đầy đủ

Đồ thị đầy đủ - complete graph

Đồ thị đầy đủ n đỉnh kí hiệu là K_n là đồ thị vô hướng mà giữa hai đỉnh bất kì của nó luôn có cạnh nối.



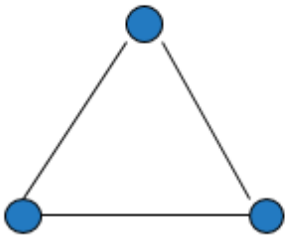
Câu hỏi: Vẽ đồ thị K_5, K_6, K_7, K_{12}. Đếm số cạnh của mỗi đồ thị trên.

Một số đồ thị đặc biệt

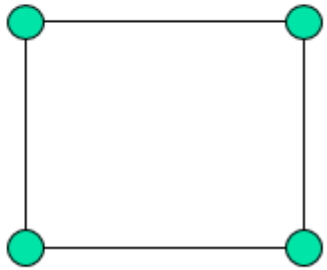
Đồ thị chu trình

Đồ thị chu trình - cycle graph

Đồ thị chu trình n đỉnh kí hiệu là C_n là đồ thị mà mọi đỉnh đều có bậc 2.



C_3



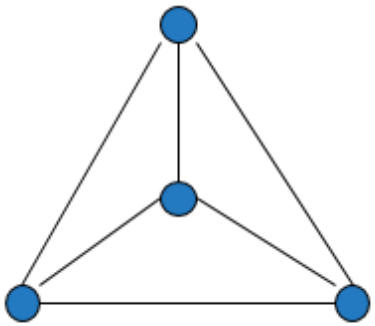
C_4

Một số đồ thị đặc biệt

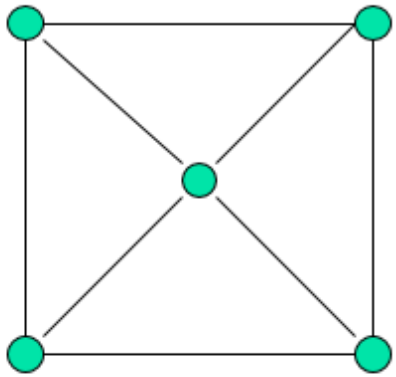
Đồ thị bánh xe

Đồ thị bánh xe - wheel graph

Đồ thị bánh xe n đỉnh kí hiệu là W_n được tạo thành từ đồ thị chu trình C_{n-1} bằng cách thêm 1 đỉnh và các cạnh nối đỉnh đó với tất cả các đỉnh còn lại



W_4



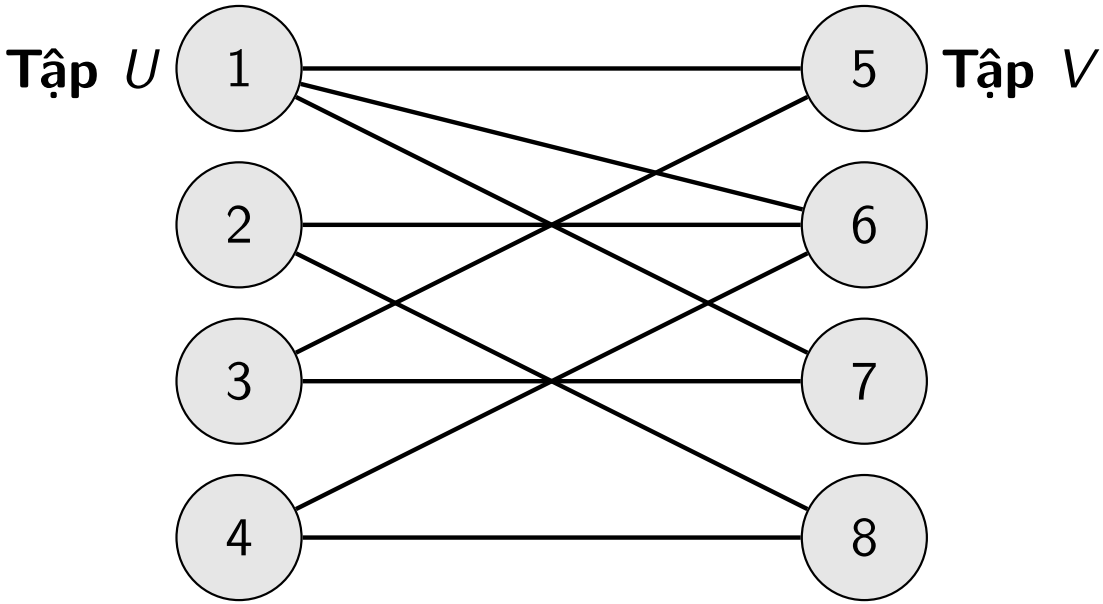
W_5

Một số đồ thị đặc biệt

Đồ thị hai phần

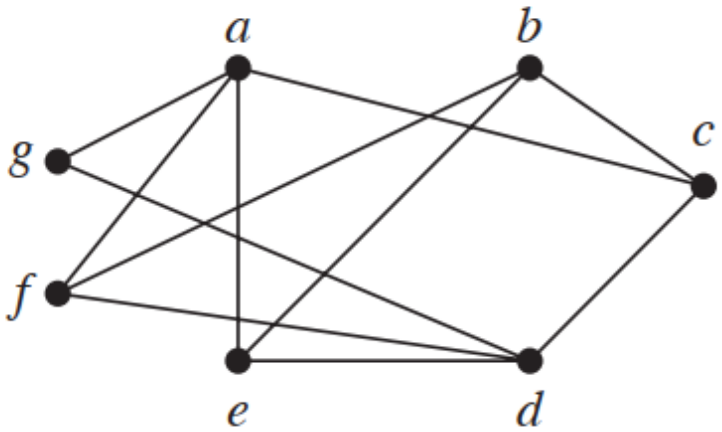
Đồ thị hai phần

Một đồ thị được gọi là hai phần nếu tập đỉnh V có thể phân hoạch thành hai tập V_1 và V_2 sao cho mỗi cạnh của đồ thị nối một đỉnh của V_1 với một đỉnh của V_2

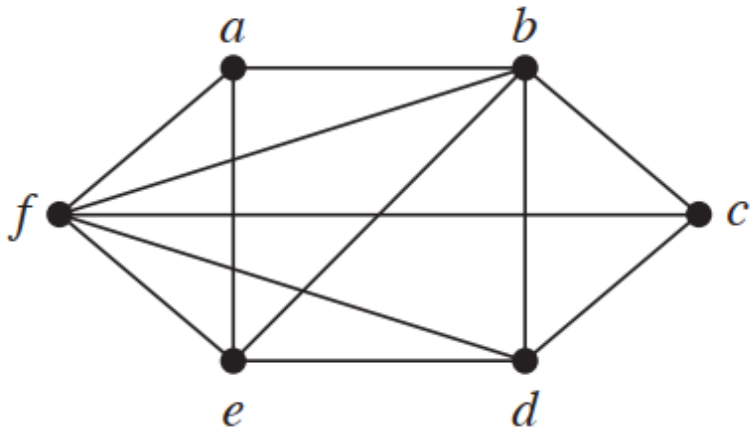


Hình 2: Đồ thị 2 phía với 8 đỉnh, chia thành hai tập $U = \{1, 2, 3, 4\}$ và $V = \{5, 6, 7, 8\}$.

Câu hỏi: Đồ thị nào dưới đây là đồ thị hai phần



G

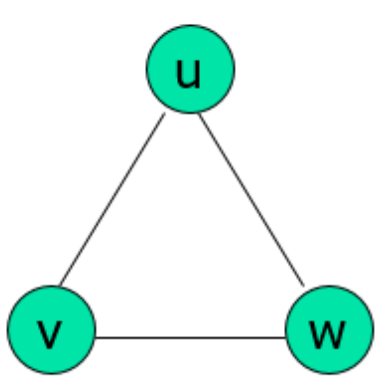


H

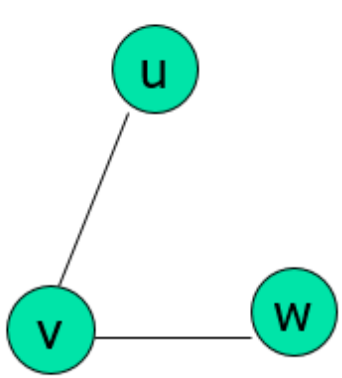
Đồ thị con

Đồ thị con

Một đồ thị con của đồ thị $G = (V, E)$ là một đồ thị $H = (V', E')$, trong đó V' là tập con của V, E' là tập con của E



G



H_1

H_2

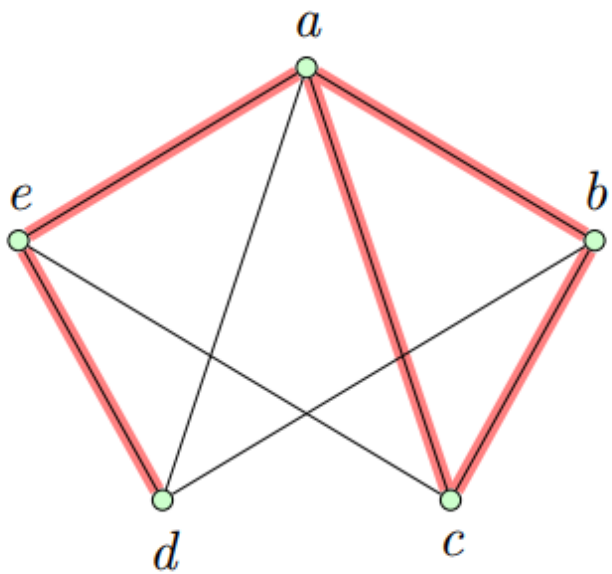
Đường đi và chu trình

Hành trình

Một hành trình trong đồ thị G là một dãy đỉnh

$$v_1, v_2, \ldots, v_k, \tag{1}$$

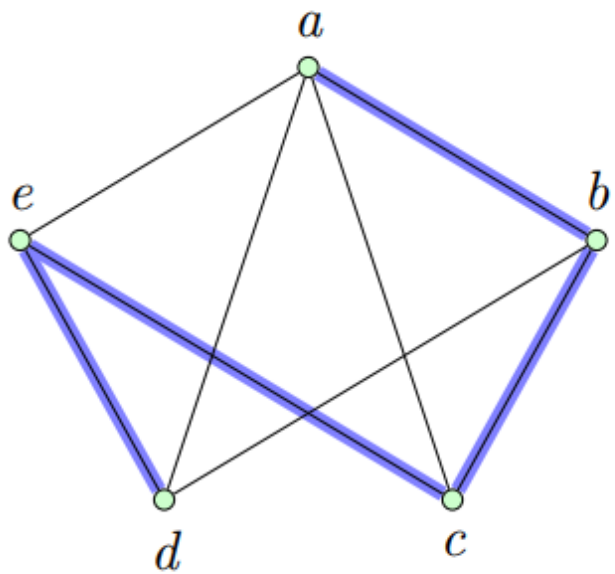
thỏa mãn v_i và v_{i+1} kề nhau (với $1 \leq i \leq k-1$)



Đường đi và chu trình

Đường đi

Hành trình mà trong đó mọi đỉnh đều khác nhau được gọi là **đường đi**



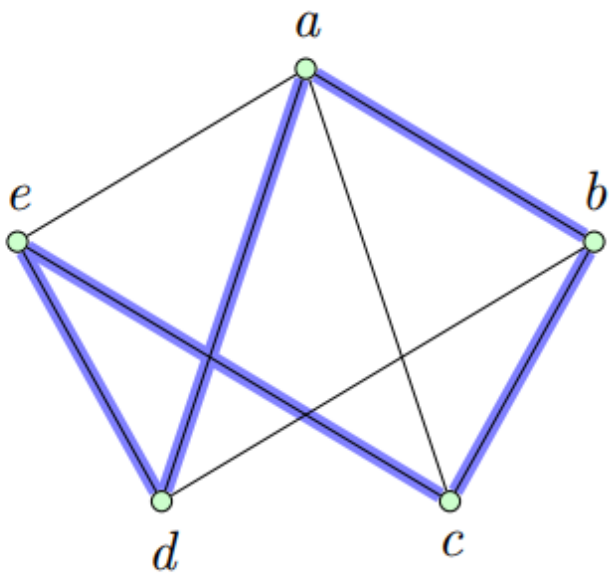
Đường đi và chu trình

Chu trình

Một hành trình

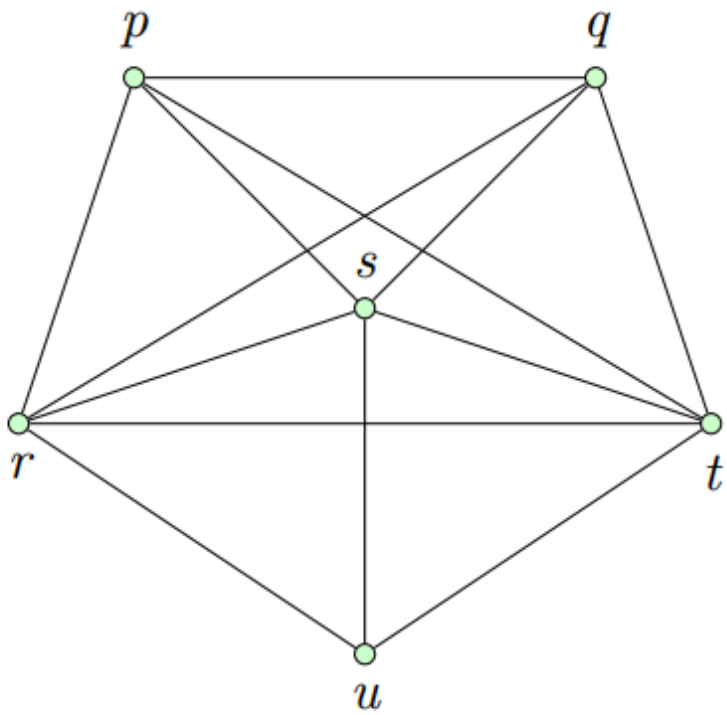
$$v_1, v_2, \ldots, v_{r+1}, \tag{2}$$

trong đó mọi đỉnh đều phân biệt ngoại trừ $v_1 = v_{r+1}$ được gọi là một **chu trình**. Ví nó có r đỉnh phân biệt và r cạnh nên ta cũng thường gọi nó là r-**chu trình**, hay chu trình **độ dài** r



Bài tập

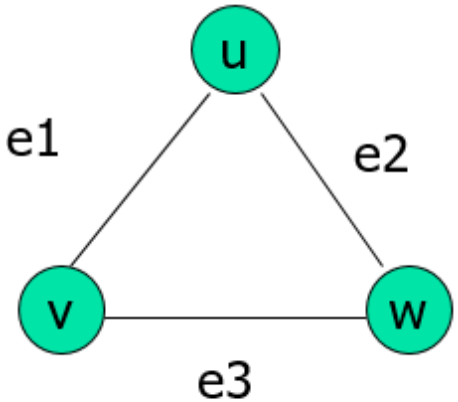
- Hình dưới đây thể hiện các địa điểm thú vị trên đảo Wanda và đường đi giữa chúng.
- Hãy tìm đường đi trên đảo để tham quan mỗi địa điểm đúng một lần và trở về vị trí xuất phát



Ma trận kề

Biểu diễn đồ thị vô hướng

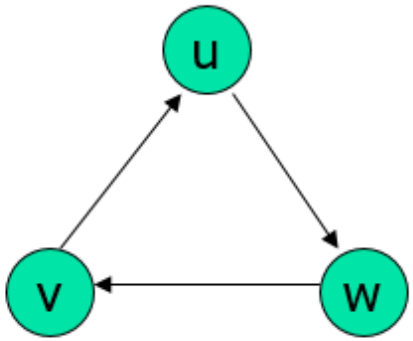
- Cho đồ thị $G = (V, E)$. Ma trận kề của đồ thị G là ma trận A kích thước $|V| \times |V|$, trong đó mỗi cạnh $e(u, v)$ tương ứng với các phần tử khác 0 ở vị trí (u, v) của ma trận.



	v	u	w
v	0	1	1
u	1	0	1
w	1	1	0

Ma trận kề

Biểu diễn đồ thị có hướng



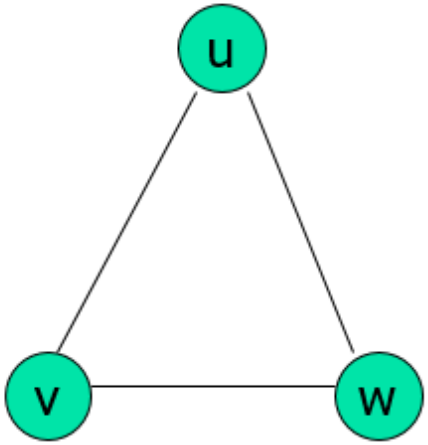
	v	u	w
v	0	1	0
u	0	0	1
w	1	0	0

Danh sách cạnh - Danh sách kề

Danh sách cạnh Liệt kê danh sách các cạnh
Danh sách kề

- Khi cần biểu diễn đồ thị thưa (đồ thị có ít cạnh) ta thường sửa dụng danh sách kề để biểu diễn đồ thị.
- Danh sách kề của đỉnh v là tập

$$Adj(v) = \{u|(u, v) \in E\} \tag{3}$$



Đỉnh	Danh sách kề
u	v , w
v	w, u
w	u , v

Duyệt đồ thị là gì

Duyệt đồ thị

Duyệt đồ thị (*Graph searching* hoặc *Graph Travesal*) là việc duyệt qua mỗi đỉnh và mỗi cạnh của đồ thị

Ứng dụng

- Xây dựng các thuật toán khảo sát tính chất của đồ thị
- Là thành phần cơ bản của nhiều thuật toán
- Tìm kiếm theo chiều rộng
- Tìm kiếm theo chiều sâu

Ý tưởng chung

Trong quá trình thực hiện thuật toán, mỗi đỉnh ở một trong ba trạng thái

- Chưa thăm/ chưa được đi qua
- Đã thăm/Đã được đi qua, nhưng chưa duyệt xong
- Đã duyệt xong

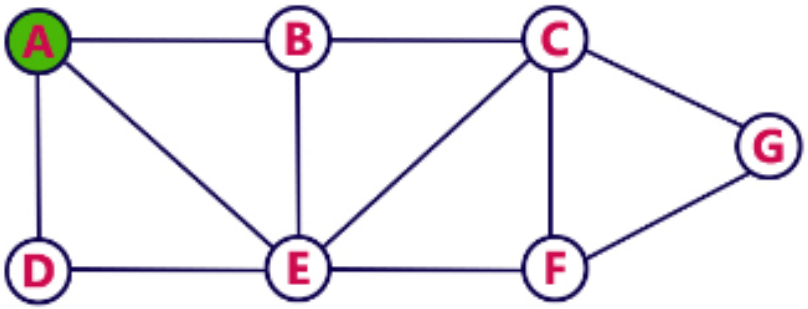
Quá trình duyệt được bắt đầu từ một đỉnh v nào đó. Ta sẽ khảo sát các đỉnh đạt được tới từ v

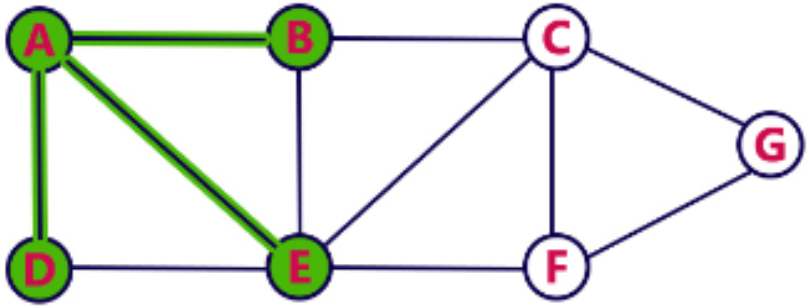
- Ban đầu, mỗi đỉnh đều được đánh dấu là chưa thăm
- Đỉnh đã được thăm sẽ chuyển thành đỉnh đã thăm nhưng chưa duyệt xong
- Khi tất cả các đỉnh kề của một đỉnh v đã được thăm, đỉnh v sẽ được đánh dấu thành đỉnh đã thăm

Tìm kiếm theo chiều rộng

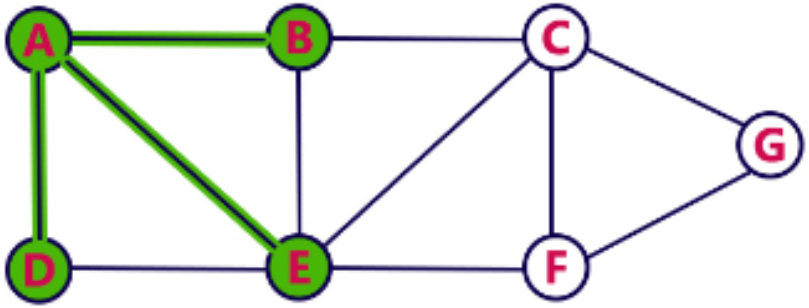
- Bước 1: Khởi tạo
  - Các đỉnh đều ở trạng thái chưa được đánh dấu. Ngoại trừ đỉnh nguồn s đã được đánh dấu.
  - Một hàng đợi ban đầu chỉ chứa 1 phần tử là s
- Bước 2: Lặp lại các bước sau cho đến khi hàng đợi rỗng:
  - Lấy đỉnh u ra khỏi hàng đợi.
  - Xét tất cả những đỉnh v kề với u mà chưa được đánh dấu, với mỗi đỉnh v đó:
    - Đánh dấu v đã thăm.
    - Lưu lại vết đường đi từ u đến v
    - Đẩy v vào trong hàng đợi (đỉnh v sẽ chờ được duyệt tại những bước sau).
- Bước 3: Truy vết tìm đường đi.

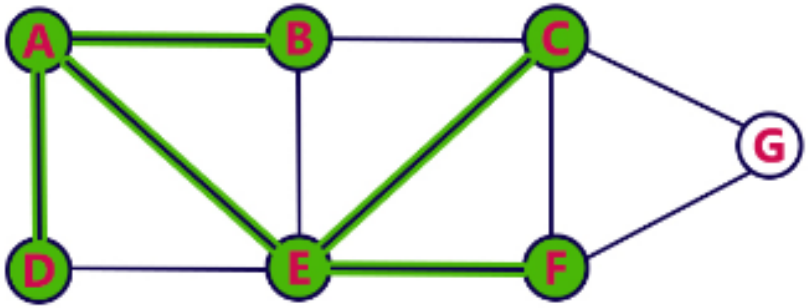
Tìm kiếm theo chiều rộng



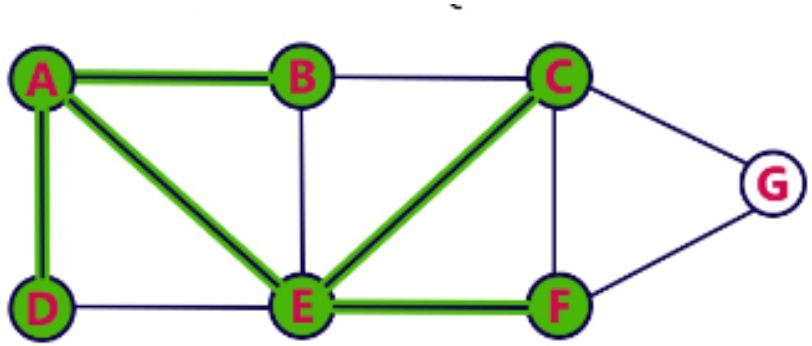


Tìm kiếm theo chiều rộng



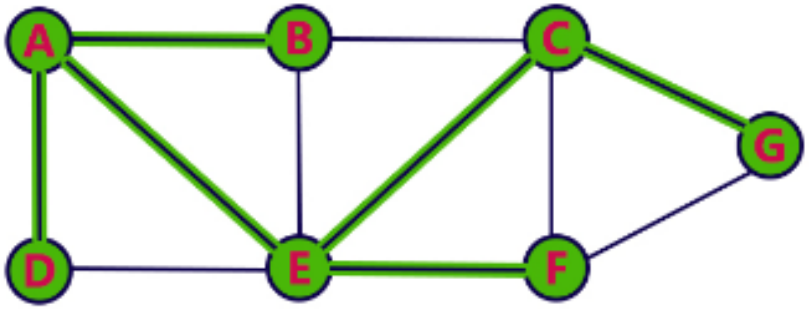


Tìm kiếm theo chiều rộng



Queue

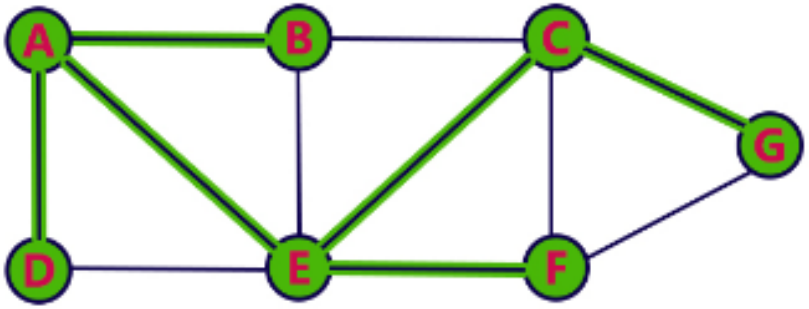
				C	F	

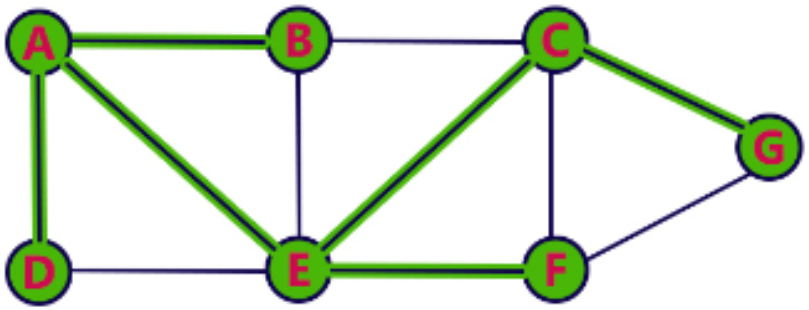


Queue

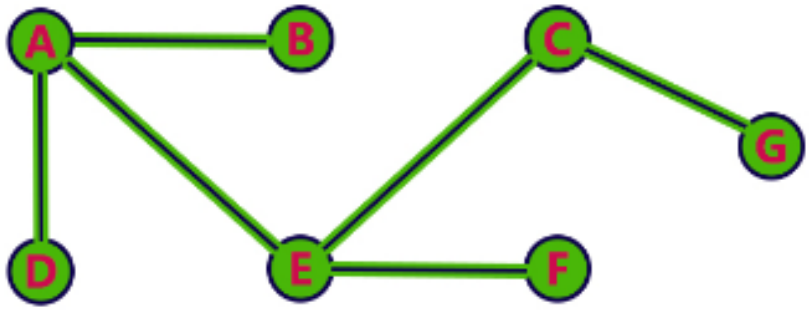
					F	G

Tìm kiếm theo chiều rộng



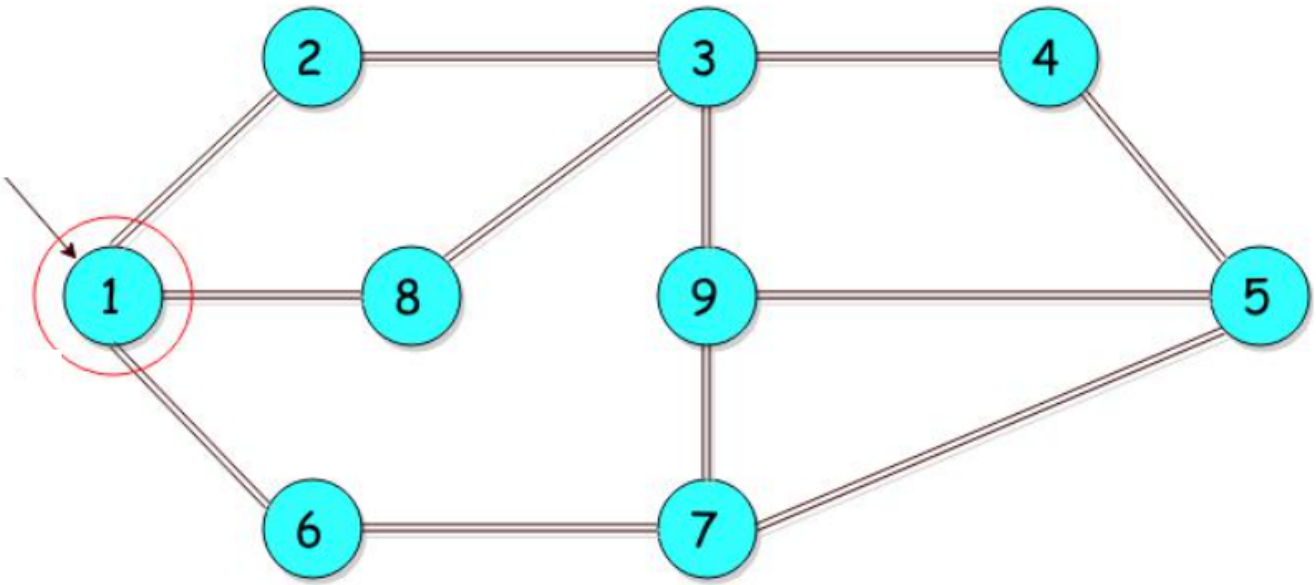


Tìm kiếm theo chiều rộng



Bài tập

Duyệt theo chiều rộng đồ thị sau:

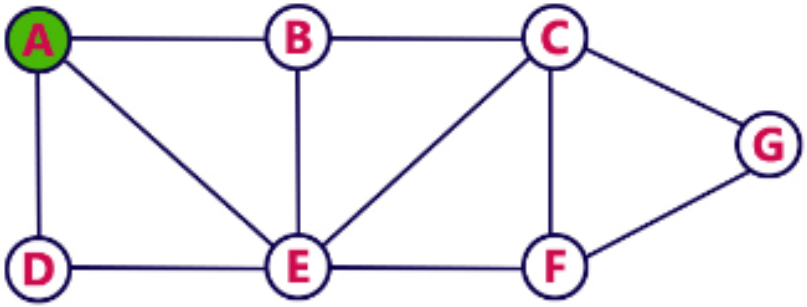


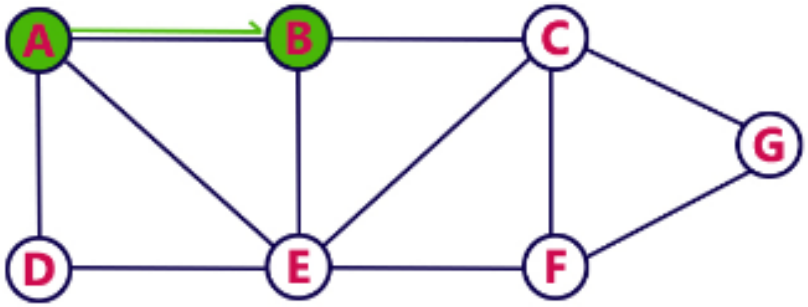
Tìm kiếm theo chiều sâu

Depth-first search - DFS

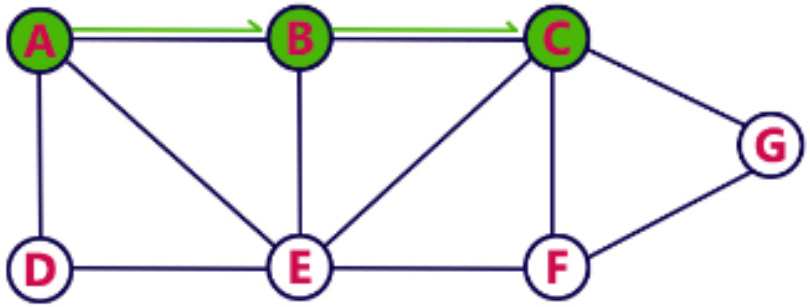
- Ta bắt đầu từ đỉnh s, đánh đấu đỉnh này "đã thăm". Sau đó ta đánh dấu s là đỉnh hiện tại u.
- Đi theo cạnh (u,v) bất kỳ.
  - Nếu cạnh (u,v) dẫn chúng ta đến đỉnh "đã thăm" v, ta quay trở về u.
  - Nếu đỉnh v là đỉnh mới, ta di chuyển đến v. Đánh dấu v là "đã thăm". Đặt v thành đỉnh hiện hành và lặp lại các bước.
- Cuối cùng, ta có thể đi đến một đỉnh mà tại đó tất cả các cạnh kề với nó đều dẫn chúng ta đến các đỉnh "đã thăm". Khi đó, ta sẽ quay lui bằng cách quay lại cho đến khi trở lại một đỉnh kề với một cạnh còn chưa được thăm. Lại tiếp tục quy trình như trên.
- Thuật toán dừng khi trở về s và không còn cạnh nào kề với nó chưa được thăm

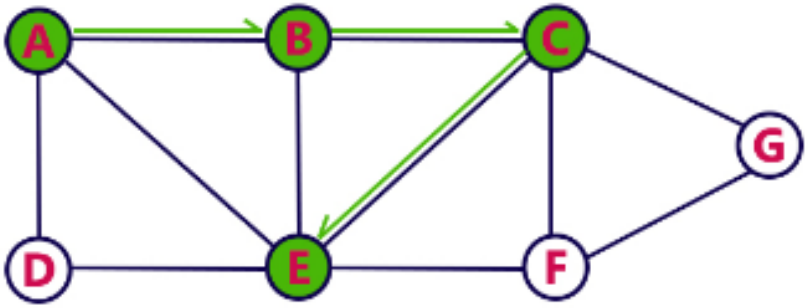
Tìm kiếm theo chiều sâu



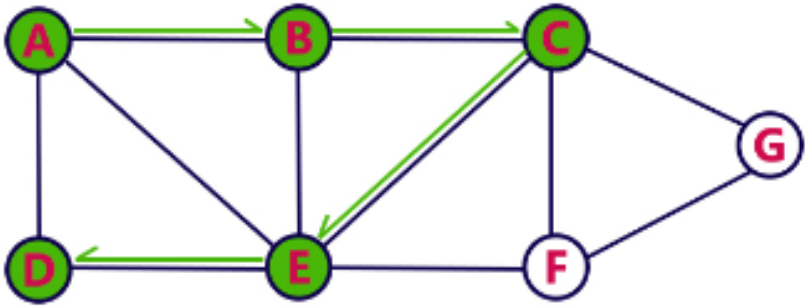


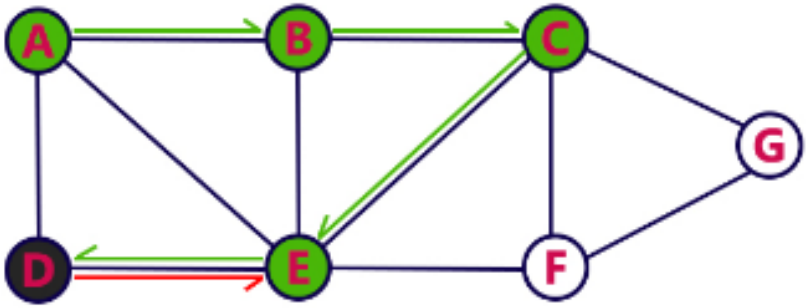
Tìm kiếm theo chiều sâu

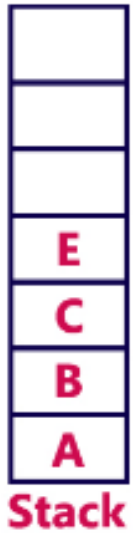




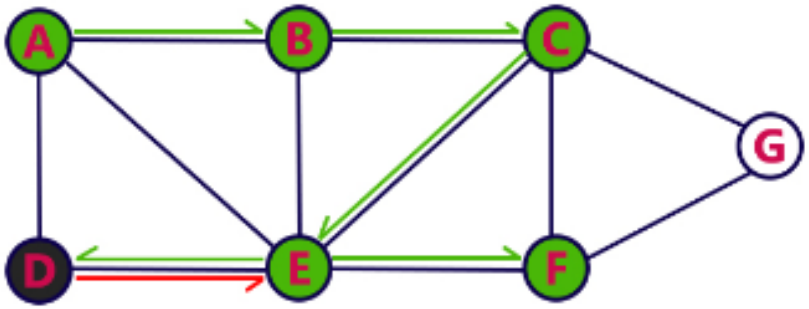
Tìm kiếm theo chiều sâu

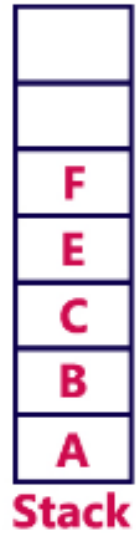


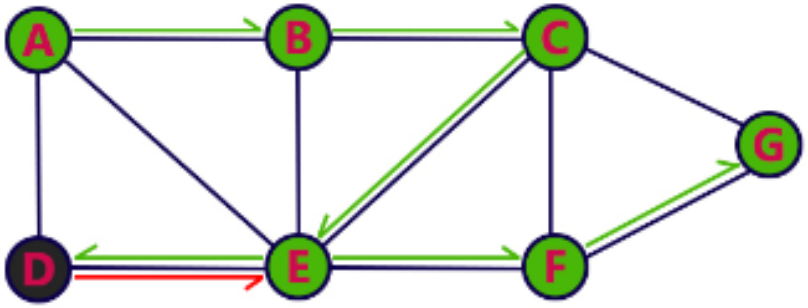




Tìm kiếm theo chiều sâu

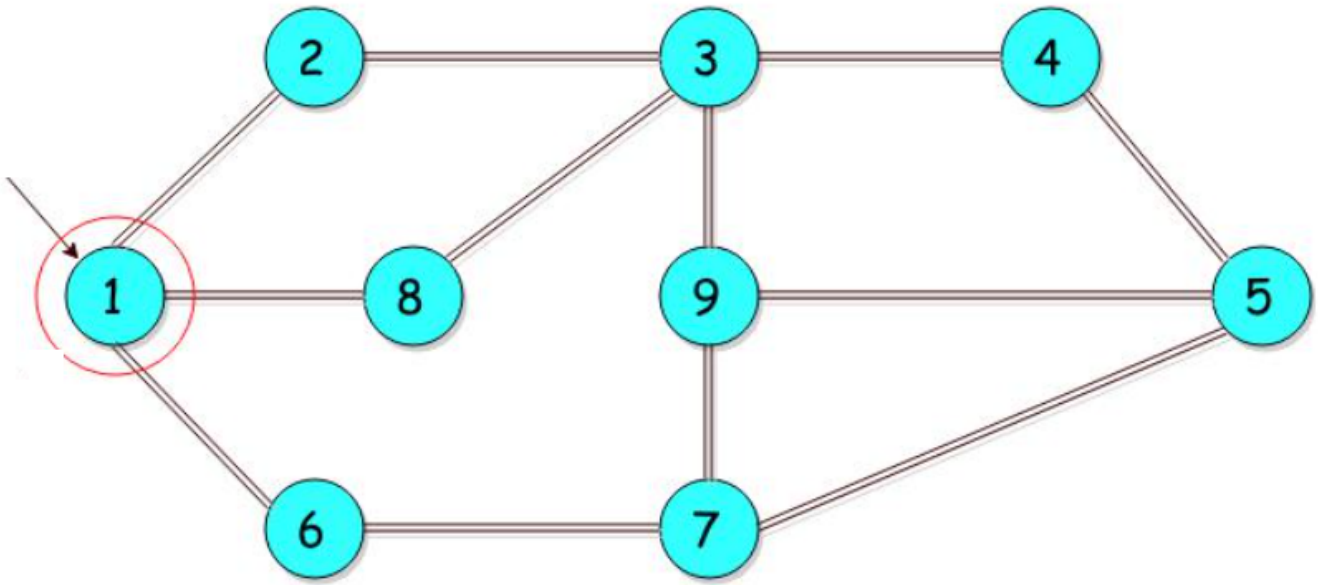






Bài tập

Duyệt theo chiều sâu đồ thị sau:



Nội dung

Giới thiệu bài toán

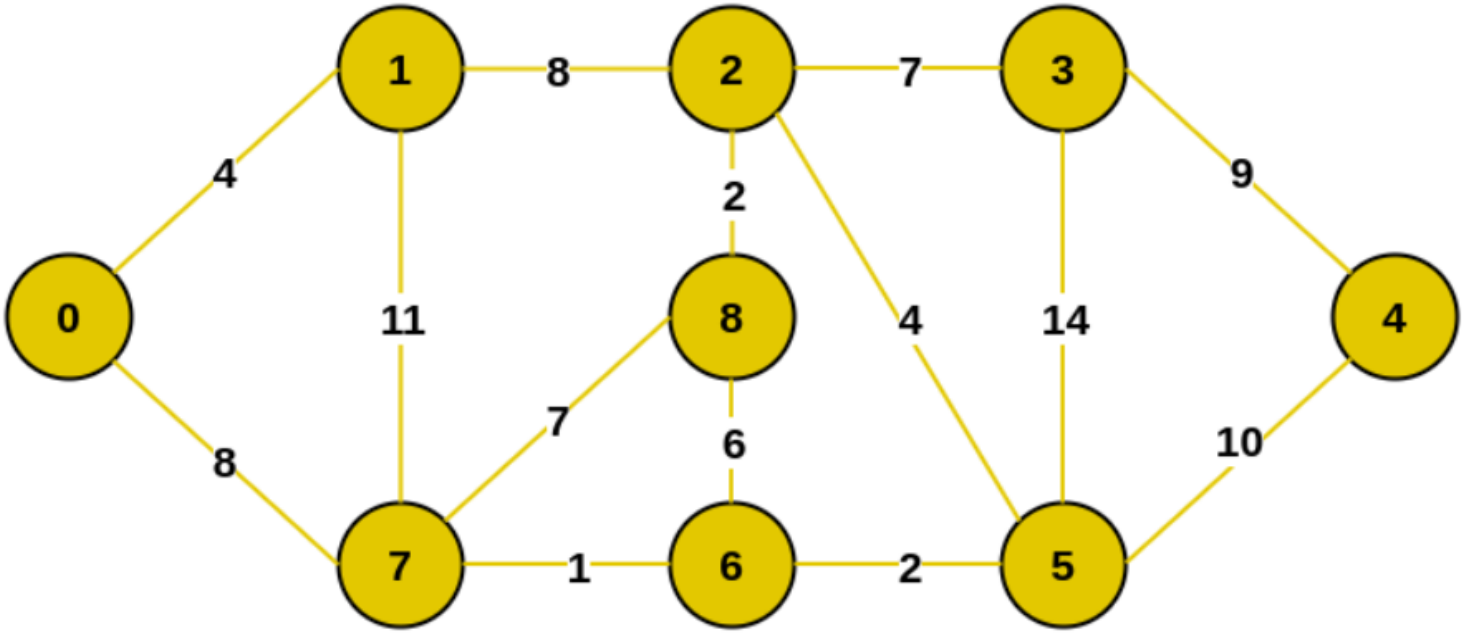
Bài toán cây khung (cây bao trùm) nhỏ nhất của đồ thị là một trong số những bài toán tối ưu trên đồ thị có nhiều ứng dụng khác nhau trong đời sống.

- Bài toán xây dựng đường giao thông: giả sử ta muốn xây dựng một hệ thống đường nối n thành phố sao cho giữa các thành phố bất kì luôn có đường đi. Bài toán đặt ra là tìm cây khung nhỏ nhất trên đồ thị, mỗi thành phố ứng với một đỉnh sao cho tổng chi phí xây dựng đường đi là nhỏ nhất.
- Bài toán nối mạng máy tính: Cần nối mạng một hệ thống gồm n máy tính đánhsố từ 1 đến n. Biết chi phí nối máy i với máy j là $c[i, j], i, j = 1, 2, ..., n$ (thông thường chi phí này phụ thuộc vào độ dài cáp nối cần sử dụng). Hãy tìm cách nối mạng sao cho tổng chi phí nối mạng là nhỏ nhất.

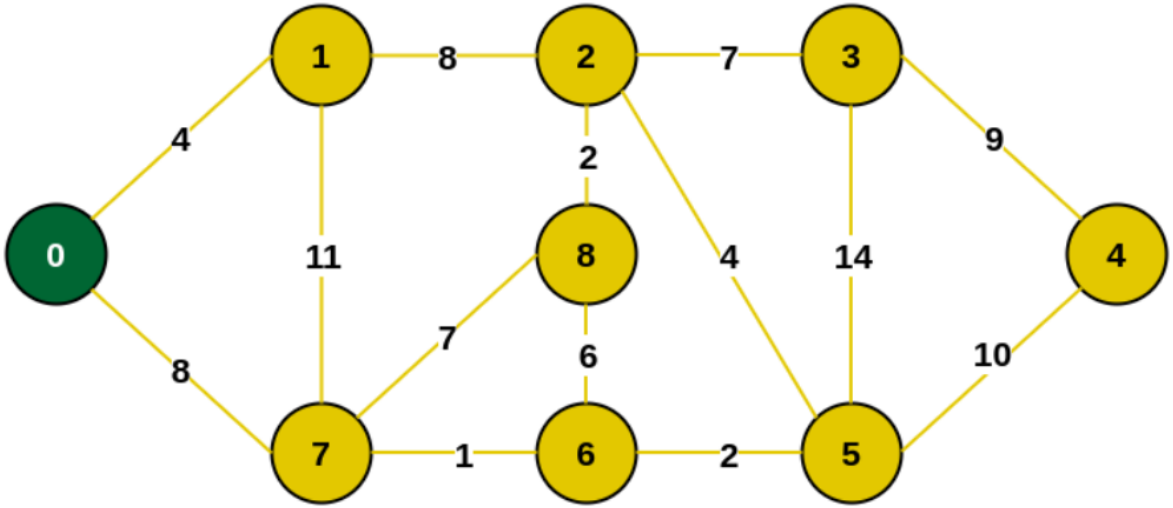
Thuật toán Prim

Ý tưởng thuật toán:

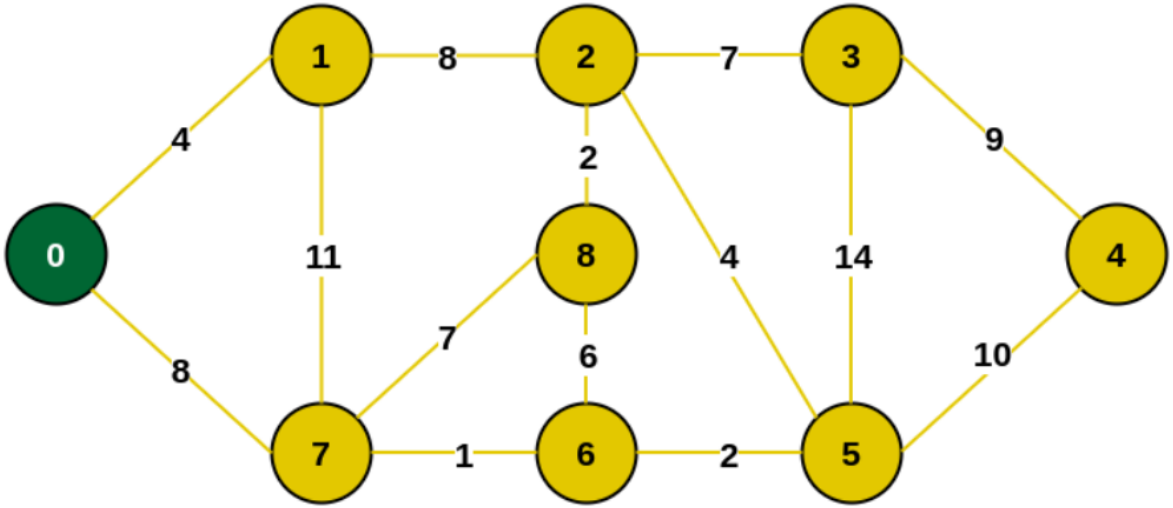
- Tại mỗi bước: chọn một đỉnh chưa được xét, sao cho đỉnh đó kề và gần nhất với các đỉnh đã được xét
- Ví dụ minh họa:

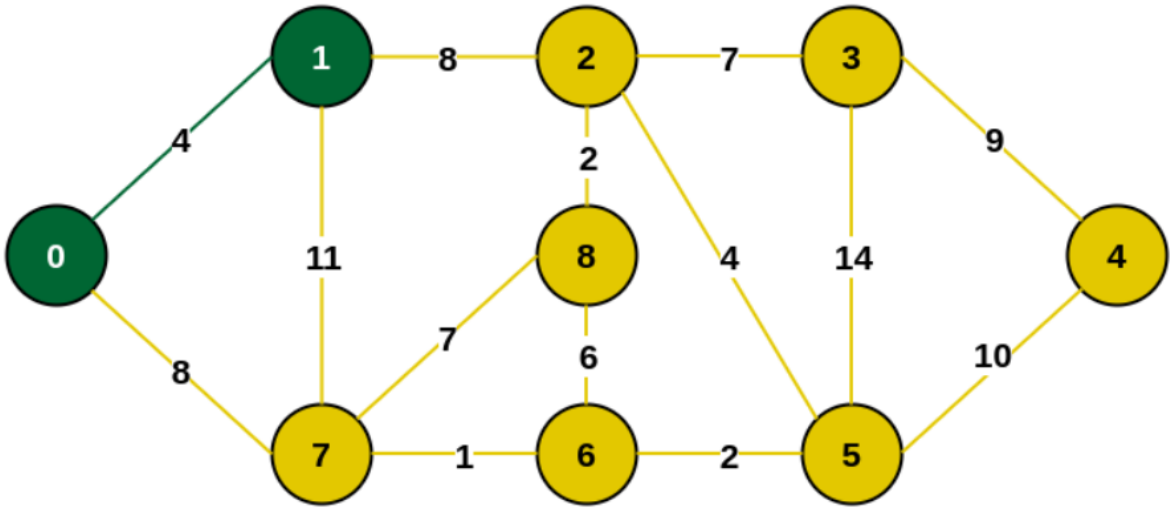


Ví dụ

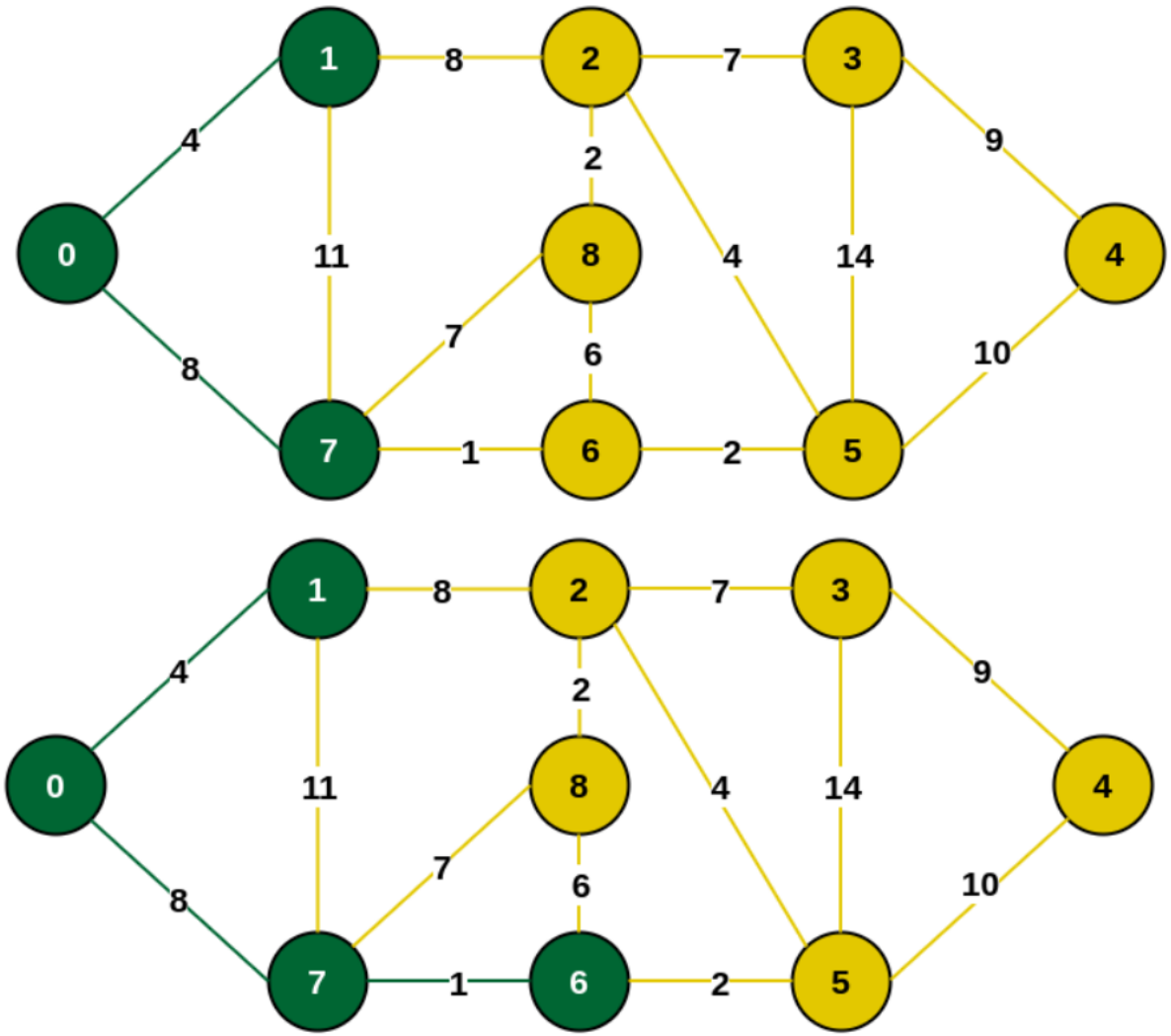


Ví dụ

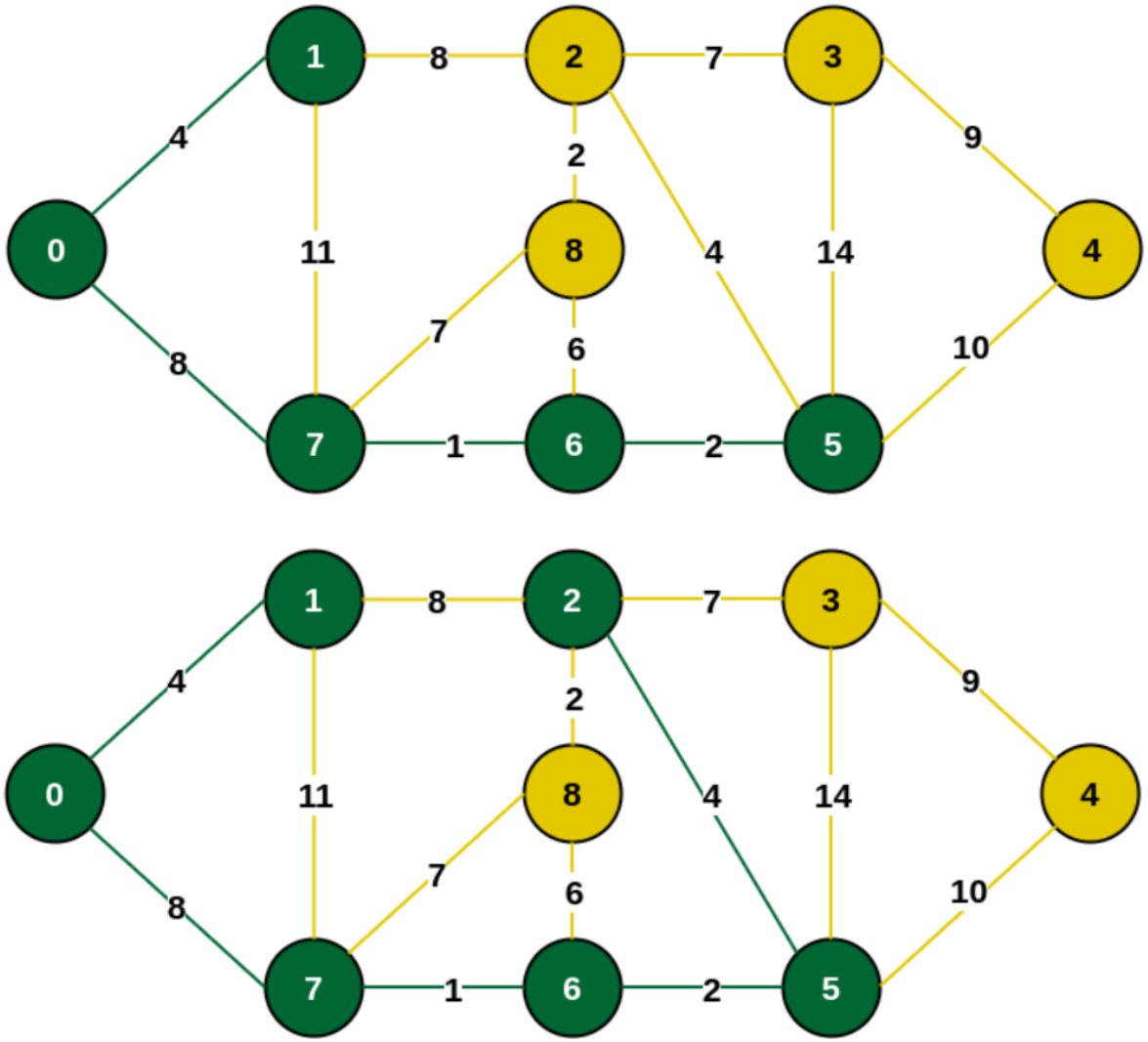

0
1
2
3
4
5
6
7
8
4
8
8
11
7
2
4
14
9
6
10
1
2


0
1
2
3
4
5
6
7
8
4
8
8
11
7
2
4
14
9
6
10
1
2

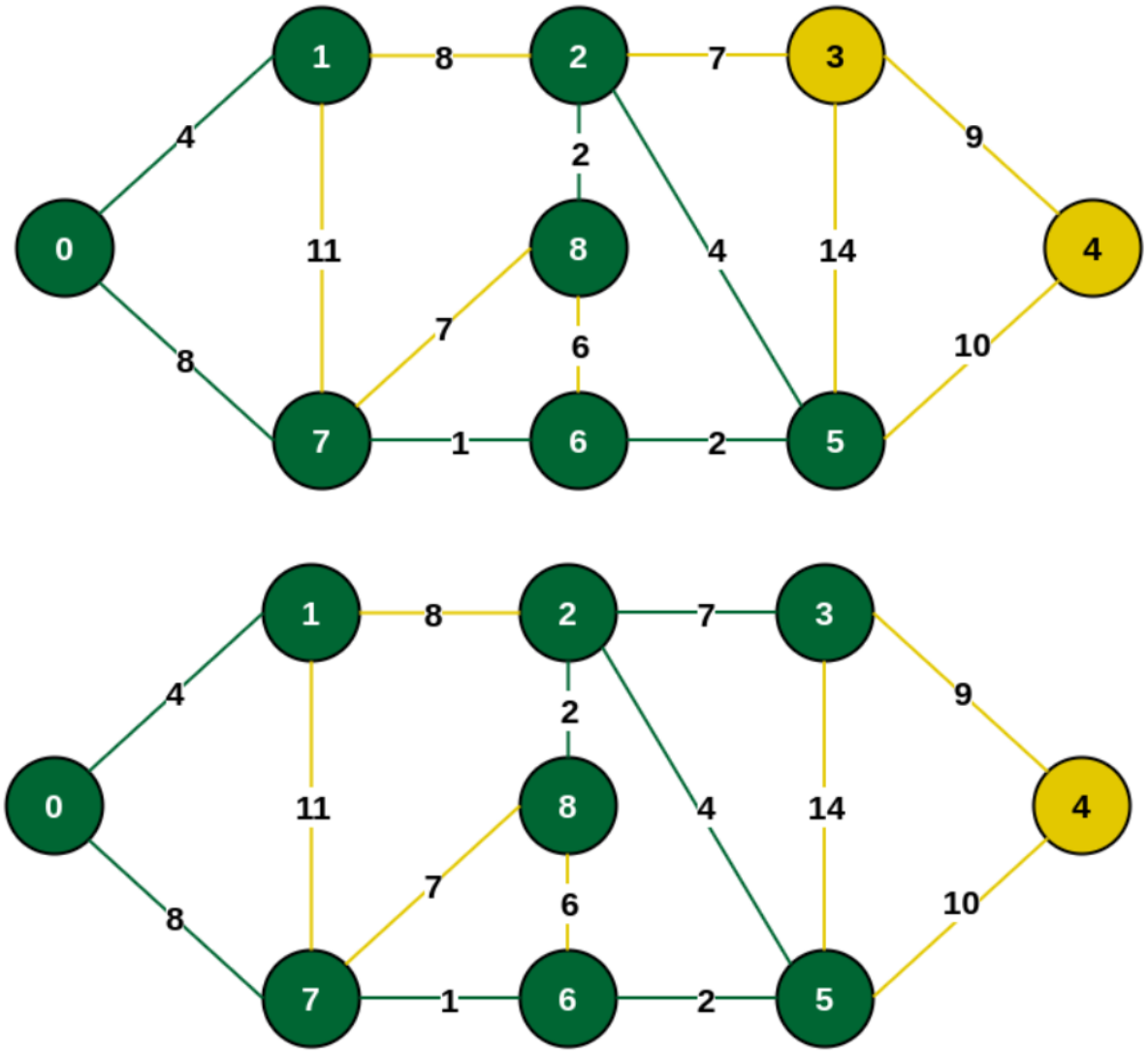
Ví dụ


0
1
2
3
4
5
6
7
8
4
8
8
11
7
2
4
14
9
6
10
1
2
0
1
2
3
4
5
6
7
8
4
8
8
11
7
2
4
14
9
6
10
1
2

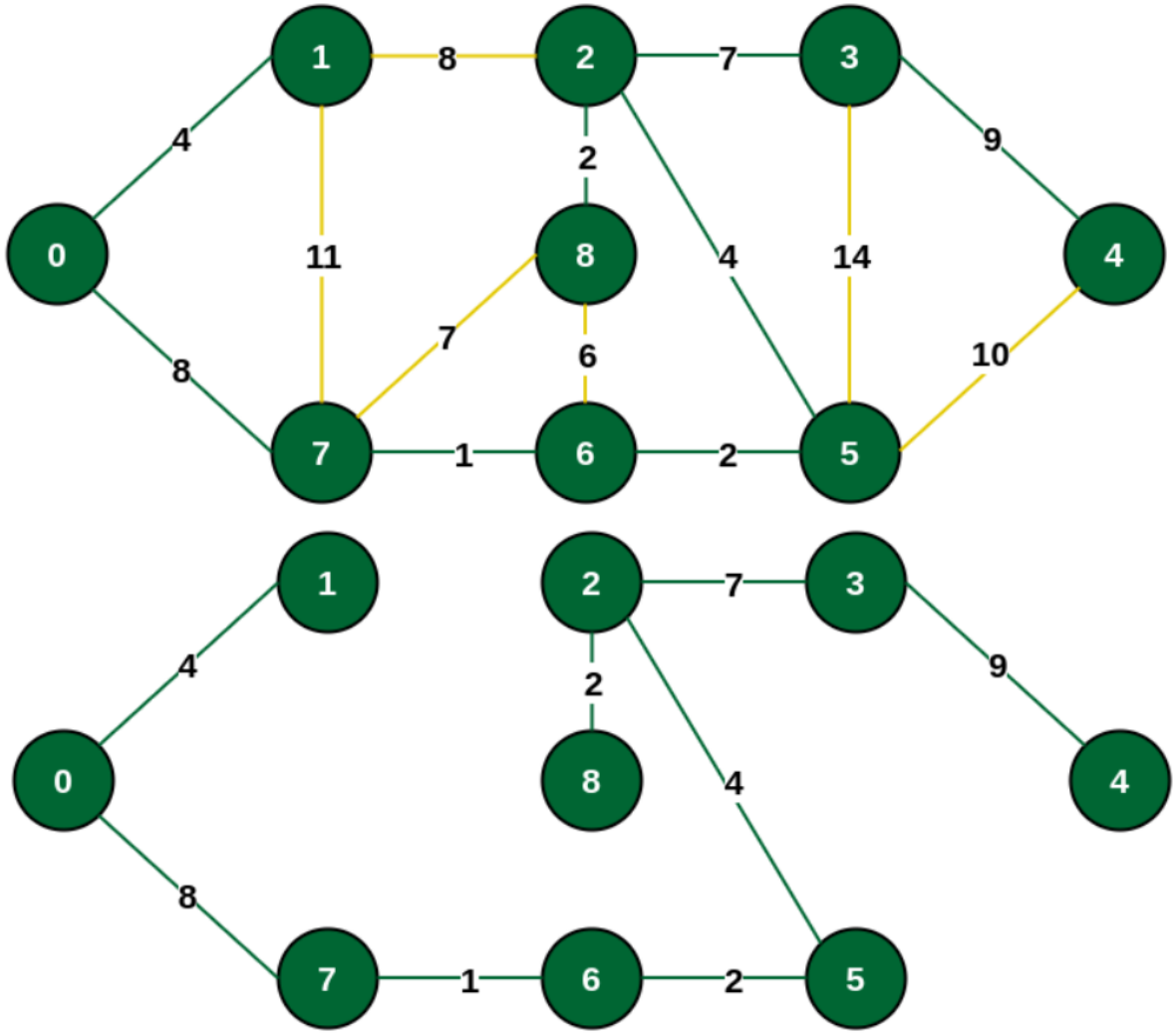
Ví dụ


0
1
2
3
4
5
6
7
8
4
8
8
11
7
2
4
14
9
6
10
1
2
0
1
2
3
4
5
6
7
8
4
8
8
11
7
2
4
14
9
6
10
1
2

Ví dụ



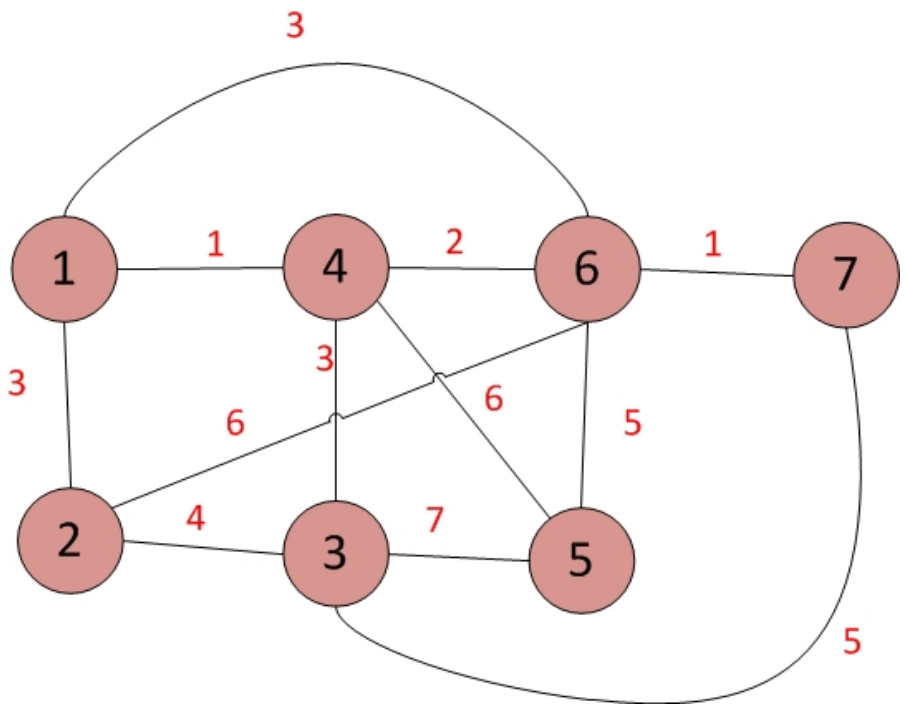
Ví dụ



Thuật toán Kruskal

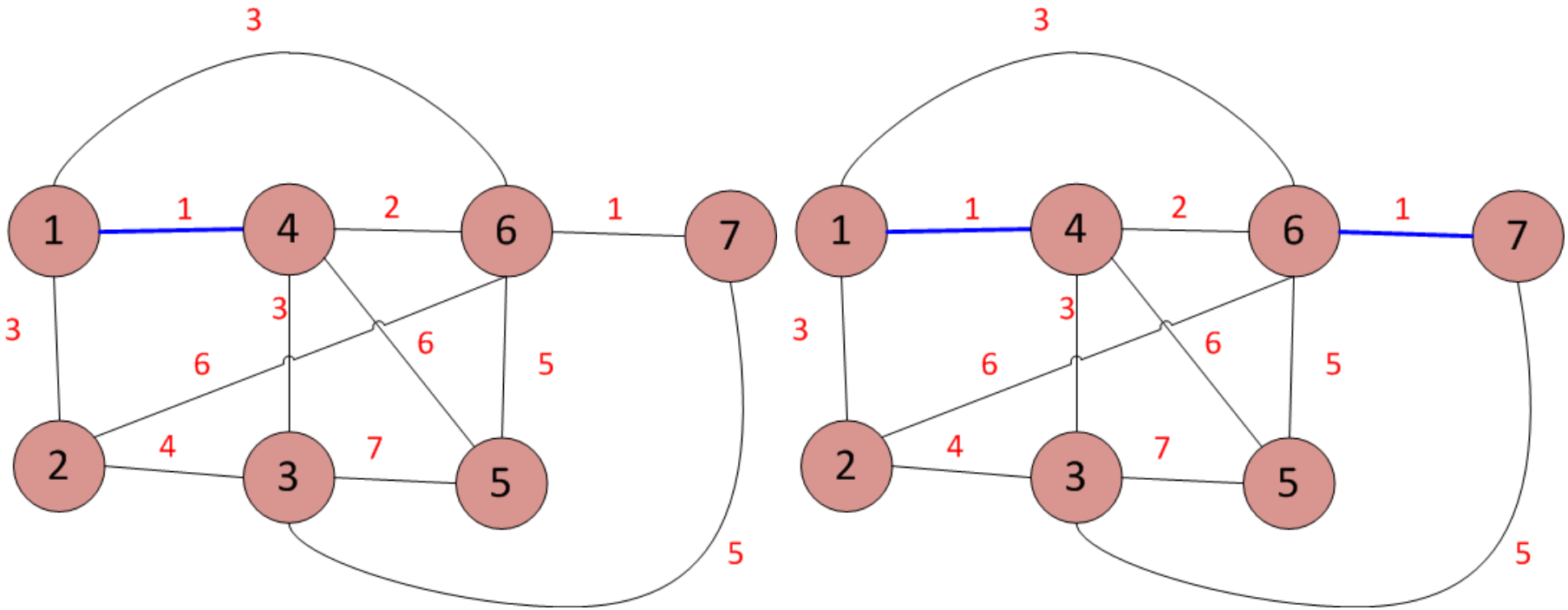
- Bước 1: Sắp xếp các cạnh của đồ thị theo thứ tự trọng số tăng dần.
- Bước 2: Khởi tạo $T := \emptyset$
- Bước 3: Lần lượt lấy từng cạnh thuộc danh sách đã sắp xếp. Nếu $T \cup \{e\}$ không chứa chu trình thì gán $T := T \cup \{e\}$.
- Bước 4: Nếu T đủ $n - 1$ phần tử thì dừng, ngược lại làm tiếp bước 3.

Ví dụ

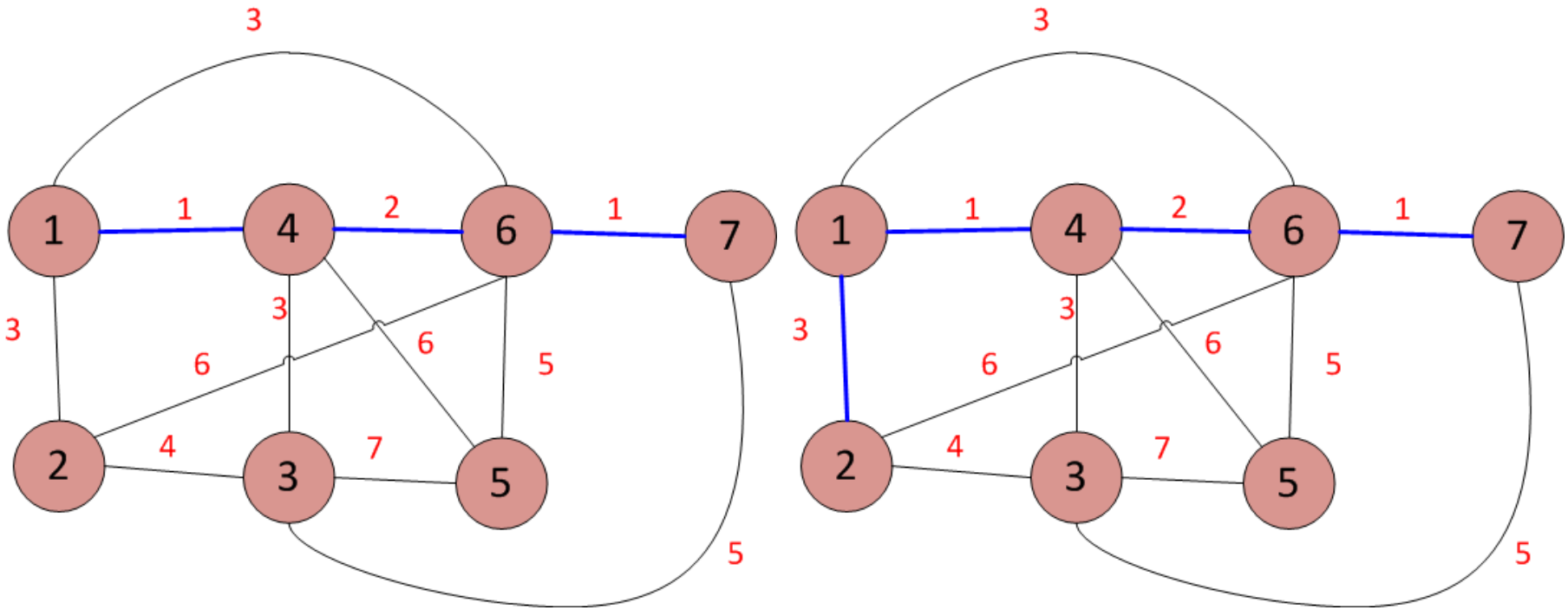


Điểm đầu	Điểm cuối	Trọng số
1	4	1
6	7	1
4	6	2
1	2	3
1	6	3
3	4	3
2	3	4
3	7	5
5	6	5
2	6	6
4	5	6
3	5	7

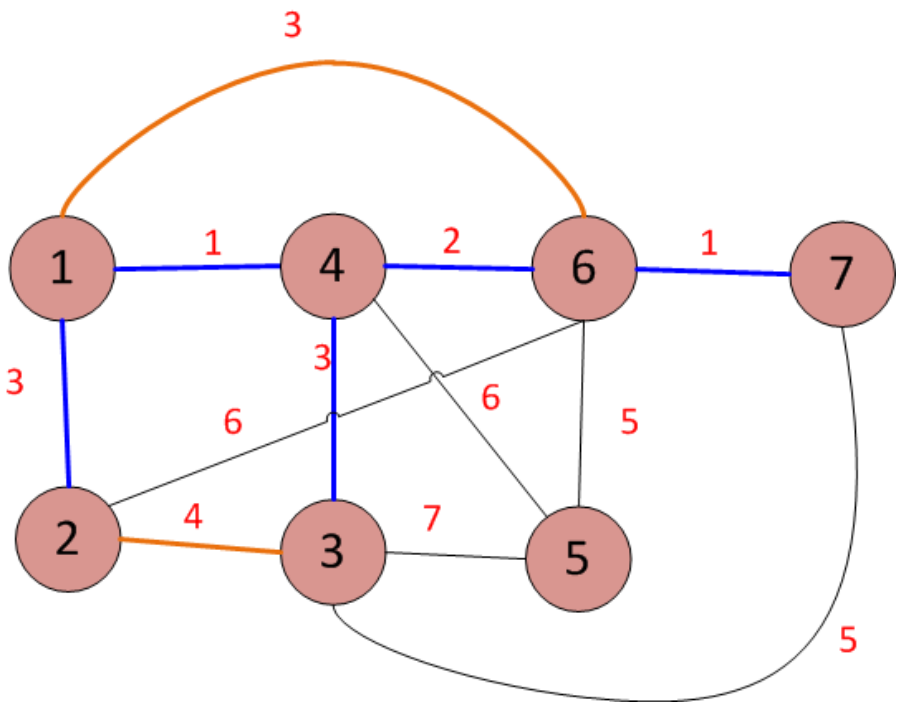
Ví dụ

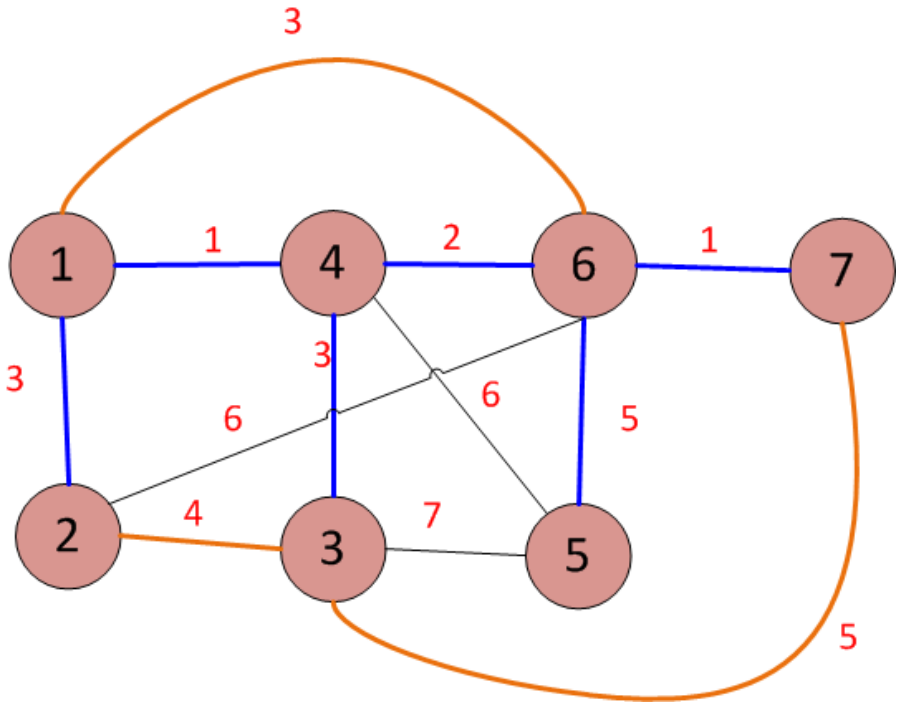


Ví dụ



Ví dụ





Bài toán đường đi ngắn nhất

Ba dạng đường đi ngắn nhất cơ bản

- Tìm đường đi ngắn nhất giữa hai đỉnh cho trước
- Tìm đường đi ngắn nhất giữa một đỉnh nguồn s tới tất cả các đỉnh còn lại
- Tìm đường đi ngắn nhất giữa hai đỉnh bất kì

Thuật toán Dijkstra

Thuật toán Dijkstra

Tìm đường đi ngắn nhất từ một đỉnh nguồn s đến tất cả các đỉnh còn lại trên đồ thị với trọng số các cạnh không âm

Ý tưởng

Thuật toán sử dụng chiến lược tham lam (greedy): Tại mỗi bước, nó chọn đỉnh có khoảng cách nhỏ nhất từ đỉnh nguồn (trong số các đỉnh chưa được xử lý) và cập nhật khoảng cách đến các đỉnh lân cận của đỉnh đó.

Nguyên lý hoạt động

- Duy trì một tập các đỉnh đã được xử lý (đã tìm được đường đi ngắn nhất).
- Ban đầu, khoảng cách từ đỉnh nguồn đến chính nó là 0, và đến các đỉnh khác là vô cùng.
- Mỗi bước, chọn đỉnh chưa được xử lý có khoảng cách nhỏ nhất từ nguồn, thêm nó vào tập đã xử lý, và cập nhật khoảng cách đến các đỉnh lân cận.

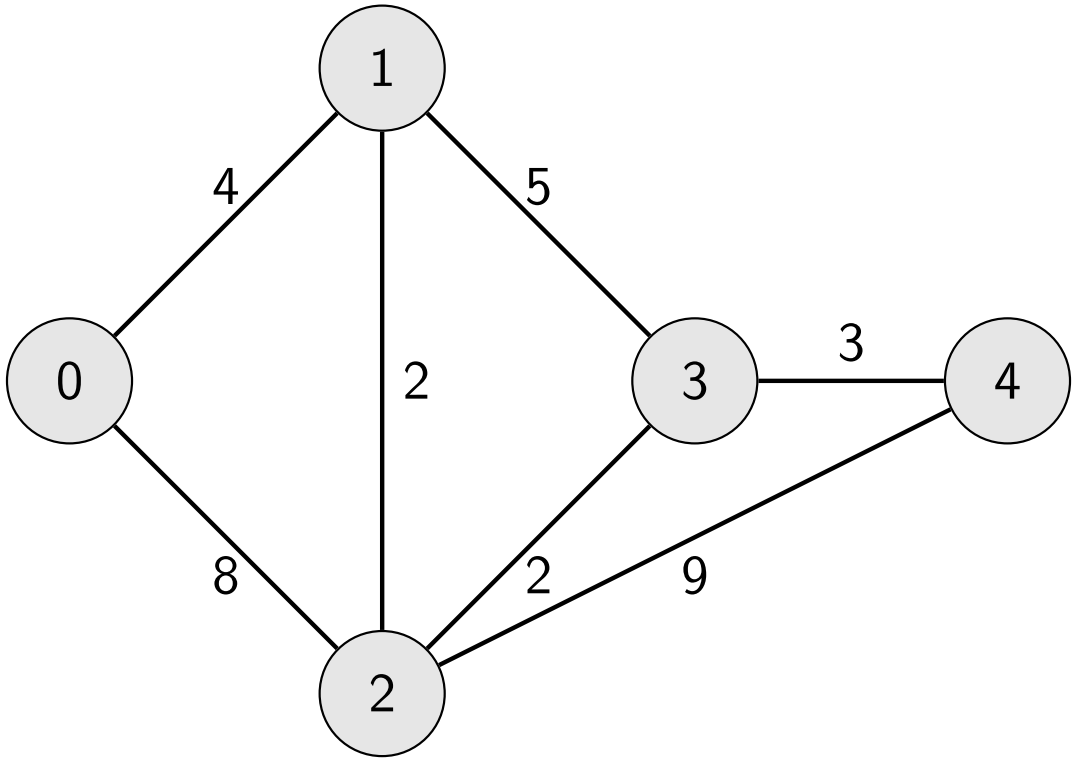
Thuật toán Dijkstra

Các bước thực hiện

1. **Khởi tạo**:
   - Đặt $d[s] = 0$, $d[v] = \infty$ (với $v \neq s$).
   - Mảng*trace* để lưu đỉnh cha của mỗi đỉnh trên đường đi ngắn nhất (dùng để truy vết đường đi).
   - Mảng *flag* để đánh dấu đỉnh đã xử lý
   - Tạo một hàng đợi ưu tiên để lưu các đỉnh chưa được xử lý, sắp xếp theo khoảng cách $d[v]$. Ban đầu, thêm tất cả các đỉnh vào hàng đợi.
2. **Lặp lại cho đến khi hàng đợi rỗng**:
   - Lấy đỉnh u chưa được xử lý $flag[u] = false$ có khoảng cách nhỏ nhất $d[u]$ từ hàng đợi.
   - Đặt $flag[u] = true$.
   - Với mỗi v kề với u mà $flag[v] = false$: nếu $d[u] + w(u, v) < d[v]$, thì:
     - Cập nhật $d[v] = d[u] + w(u, v)$.
     - Cập nhật cha của v: $trace[v] = u$.
     - Cập nhật lại hàng đợi ưu tiên (vì $d[v]$ đã thay đổi).
3. **Kết thúc**:
   - Khi hàng đợi rỗng, thuật toán kết thúc.

Thuật toán Dijkstra

Ví dụ



Hình 3: Đồ thị ví dụ với 5 đỉnh và các cạnh có trọng số.

Thuật toán Dijkstra

Ví dụ

1. **Khởi tạo**:
   - $d[0] = 0$, $d[1] = \infty$, $d[2] = \infty$, $d[3] = \infty$, $d[4] = \infty$.
   - $trace[]$: Tất cả là -1.
   - Hàng đợi ưu tiên: $\{(0, 0), (1, \infty), (2, \infty), (3, \infty), (4, \infty)\}$.
2. **Lần 1**:
   - Lấy đỉnh 0 (khoảng cách nhỏ nhất: 0).
   - $flag[0] = true$
   - Tính khoảng cách tới các đỉnh từ 0:
     - $(0, 1)$: $d[1] = d[0] + 4 = 4$, $trace[1] = 0$.
     - $(0, 2)$: $d[2] = d[0] + 8 = 8$, $trace[2] = 0$.
   - Hàng đợi: $\{(1, 4), (2, 8), (3, \infty), (4, \infty)\}$.

Thuật toán Dijkstra

Ví dụ

1. **Lần 2**:
   - Lấy đỉnh 1 (khoảng cách nhỏ nhất: 4).
   - $flag[1] = true$
   - Cập nhật khoảng cách từ 1
     - $(1, 2)$: $d[2] = d[1] + 2 = 4 + 2 = 6$, $trace[2] = 1$ (cải thiện từ 8 xuống 6).
     - $(1, 3)$: $d[3] = d[1] + 5 = 4 + 5 = 9$, $trace[3] = 1$.
   - Hàng đợi: $\{(2, 6), (3, 9), (4, \infty)\}$.
2. **Lần 3**:
   - Lấy đỉnh 2 (khoảng cách nhỏ nhất: 6).
   - $flag[2] = true$
   - Cập nhật khoảng cách từ 2
     - $(2, 3)$: $d[3] = d[2] + 2 = 6 + 2 = 8$, $trace[3] = 2$ (cải thiện từ 9 xuống 8).
     - $(2, 4)$: $d[4] = d[2] + 9 = 6 + 9 = 15$, $trace[4] = 2$.
   - Hàng đợi: $\{(3, 8), (4, 15)\}$.

Thuật toán Dijkstra

Ví dụ

1. **Lần 4**:
   - Lấy đỉnh 3 (khoảng cách nhỏ nhất: 8).
   - $flag[3] = true$
   - Cập nhật khoảng cách từ 3:
     - $(3, 4)$: $d[4] = d[3] + 3 = 8 + 3 = 11$, $trace[4] = 3$ (cải thiện từ 15 xuống 11).
   - Hàng đợi: $\{(4, 11)\}$.
2. **Lần 5**:
   - Lấy đỉnh 4 (khoảng cách nhỏ nhất: 11).
   - $flag[4] = true$.
   - Không cập nhật được
   - Hàng đợi: Rỗng.

Thuật toán Dijkstra

Ví dụ

- Khoảng cách ngắn nhất từ đỉnh 0:
  - $d[0] = 0$
  - $d[1] = 4$
  - $d[2] = 6$
  - $d[3] = 8$
  - $d[4] = 11$
- Đường đi ngắn nhất (truy ngược từ *trace*):
  - Từ 0 đến 4: $0 \to 1 \to 2 \to 3 \to 4$.