

**Phần I**

**Lý thuyết đồ thị**



# Chương 1

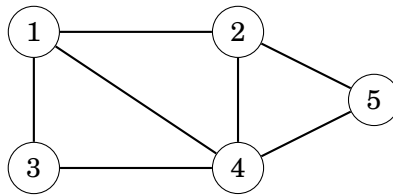
## Lý thuyết đồ thị - cơ bản

Các khái niệm về đồ thị và biểu diễn đồ thị.

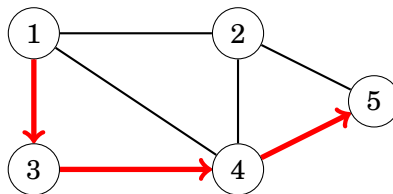
### 1.1 Các thuật ngữ đồ thị

A **đồ thị** gồm các **đỉnh** và các **cạnh**. Trong các bài toán, thường sẽ là  $n$  đỉnh,  $m$  cạnh. Các đỉnh được đánh số lần lượt là  $1, 2, \dots, n$ .

Ví dụ, đồ thị dưới đây gồm 5 đỉnh, 7 cạnh:



Một **đường đi** từ đỉnh  $a$  đến đỉnh  $b$  thông qua một số cạnh trên đồ thị. Độ dài của đường đi là số cạnh trên đường đi đó. Ví dụ, đồ thị dưới đây có một đường đi là từ đỉnh 1 đến đỉnh 5 là:  $1 \rightarrow 3 \rightarrow 4 \rightarrow 5$  với độ dài là 3.

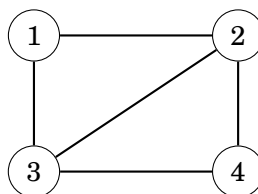


Một đường đi gọi là là **chu trình** nếu đỉnh đầu tiên và đỉnh cuối cùng trùng nhau. Ví dụ, đồ thị trên có một chu trình:  $1 \rightarrow 3 \rightarrow 4 \rightarrow 1$ . Một đường đi là **đường đi đơn** nếu mỗi nút xuất hiện nhiều nhất một lần trong đường đi.

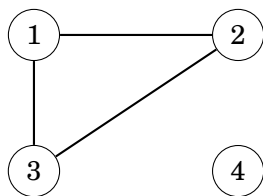
### Tính liên thông

Một đồ thị là **liên thông** nếu luôn có một đường đi giữa hai đỉnh bất kỳ trong đồ thị.

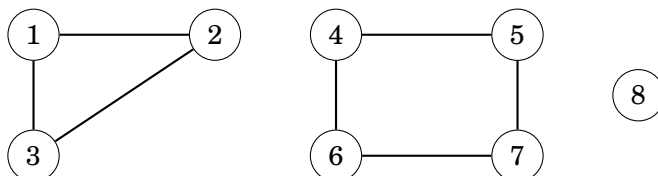
Ví dụ: đồ thị dưới đây là liên thông:



Đồ thị dưới đây không liên thông vì không có đỉnh nào được kết nối với đỉnh 4 của đồ thị:

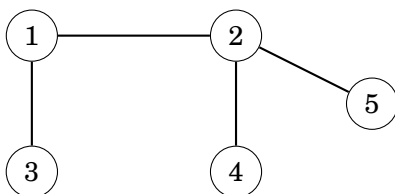


Mỗi phần liên thông trong đồ thị được gọi là **thành phần liên thông**. Ví dụ, đồ thị dưới đây gồm ba thành phần liên thông: {1, 2, 3}, {4, 5, 6, 7} và {8}.



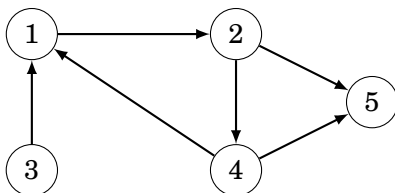
Một **cây đồ thị** là một đồ thị liên thông gồm  $n$  đỉnh,  $n - 1$  cạnh. Giữa hai đỉnh bất kỳ của cây, chỉ có một đường đi duy nhất giữa hai nút bất kỳ của cây.

Ví dụ, đồ thị dưới đây là một cây:



### Cạnh có hướng

Một đồ thị gọi là **có hướng** nếu các cạnh của nó chỉ được duyệt theo một hướng chỉ định. Ví dụ, đồ thị dưới đây là một đồ thị có hướng:

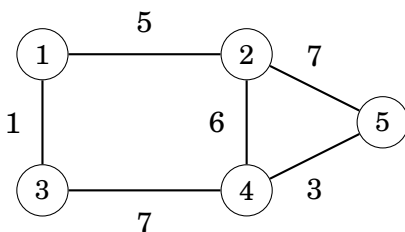


Đồ thị ở trên tồn tại đường đi từ đỉnh 3 tới đỉnh 5:  $3 \rightarrow 1 \rightarrow 2 \rightarrow 5$  nhưng không có đường đi nào từ đỉnh 5 tới đỉnh 3.

### Cạnh có trọng số

Trong một đồ thị **có trọng số** mỗi cạnh sẽ được gán một giá trị gọi là **trọng số**. Các trọng số thường được hiểu là độ dài cạnh.

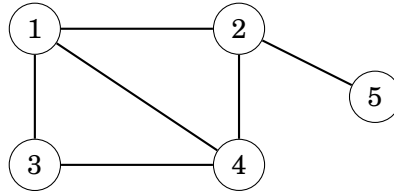
Ví dụ: đồ thị dưới đây là đồ thị có trọng số:



Độ dài của một đường đi trong đồ thị có trọng số là tổng trọng số tất cả các cạnh trên đường đi đó. Ví dụ với đồ thị ở trên, đường đi  $1 \rightarrow 2 \rightarrow 5$  có độ dài là 12, còn độ dài của đường đi  $1 \rightarrow 3 \rightarrow 4 \rightarrow 5$  là 11, đây cũng là đường đi **ngắn nhất** từ đỉnh 1 đến đỉnh 5.

### Đỉnh kề và bậc của đỉnh

Hai đỉnh gọi là **liền kề** nếu tồn tại một cạnh nối trực tiếp giữa chúng. **Bậc** của một đỉnh là số lượng đỉnh kề với đỉnh đó. Ví dụ, đồ thị trên đỉnh 2 có 3 đỉnh liền kề là 1, 4, 5 do đó nó có bậc là 3.


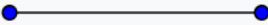
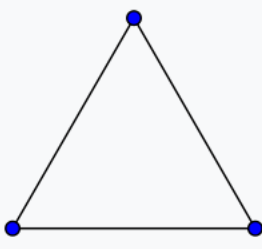
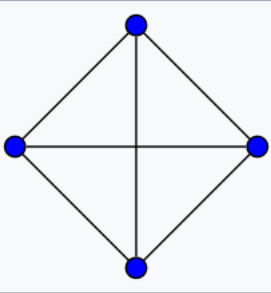
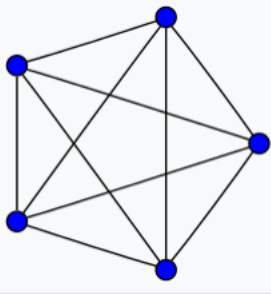
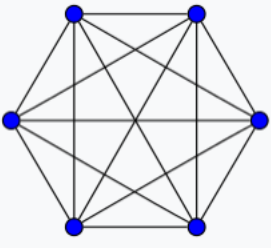
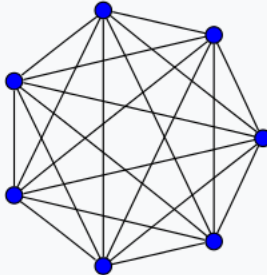
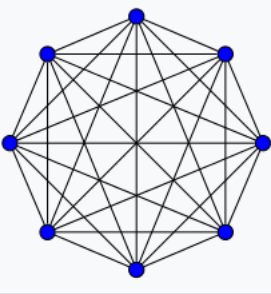
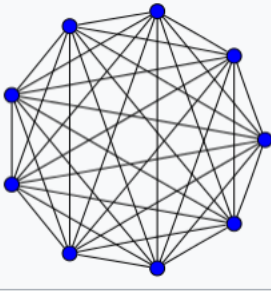
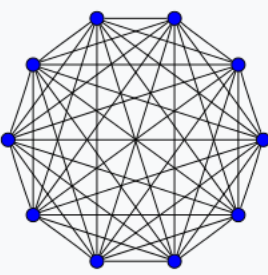
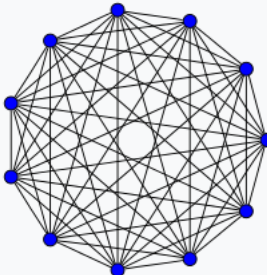
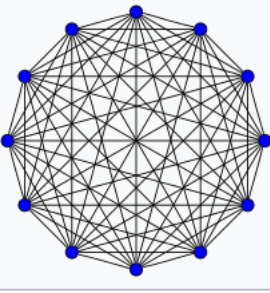


Tổng bậc trong đồ thị luôn là  $2 \times m$ , với  $m$  là số cạnh của đồ thị, bởi vì mỗi cạnh tăng bậc hai đỉnh của nó mỗi nút thêm một. Do đó, tổng bậc của đồ thị luôn chẵn.

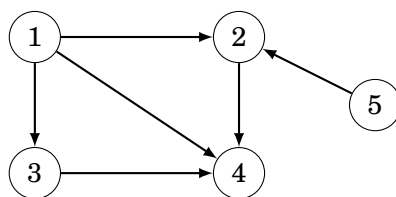
Một đồ thị gọi là **chính quy** nếu các đỉnh đều có cùng bậc. Một đồ thị chính quy với đỉnh có bậc  $k$  được gọi là **đồ thị chính quy bậc  $k$** . Đồ thị chính quy bậc 0 gồm các đỉnh không có cạnh chung. Đồ thị chính quy bậc 1 gồm tập các cạnh không nối với nhau, đồ thị chính quy bậc 2 gồm các chu trình không nối với nhau.

Một đồ thị gọi là **đầy đủ** mỗi đỉnh đều có bậc là  $n - 1$ . Trong đồ thị đầy đủ, luôn có cạnh nối trực tiếp hai đỉnh bất kỳ. Đồ thị đầy đủ là đồ thị đơn có nhiều cạnh nhất và là đồ thị chính quy bậc  $n - 1$ .

Dưới đây là hình minh họa một số đồ thị đầy đủ với số đỉnh từ 1 đến 12:

$K_1 : 0$	$K_2 : 1$	$K_3 : 3$	$K_4 : 6$
			
$K_5 : 10$	$K_6 : 15$	$K_7 : 21$	$K_8 : 28$
			
$K_9 : 36$	$K_{10} : 45$	$K_{11} : 55$	$K_{12} : 66$
			

Trong đồ thị có hướng, **bậc trong** của một đỉnh là số cạnh kết thúc tại đỉnh đó và **bậc ngoài** của một đỉnh là số cạnh bắt đầu tại đỉnh đó. Ví dụ, trong biểu đồ sau, bậc trong của nút 2 là 2 và bậc ngoài của nút 2 là 1.

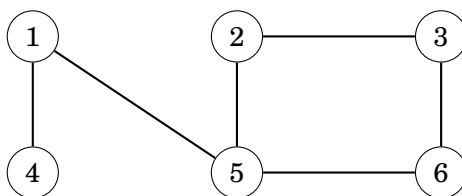


## Tô màu

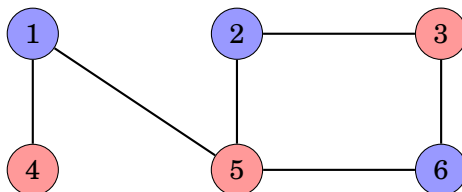
Khi **tô màu** cho một đồ thị, mỗi đỉnh sẽ được gán cho một màu sao cho không có hai đỉnh kề nhau nào được tô cùng một màu.

Một đồ thị gọi là **đồ thị hai phía** nếu có thể tô màu nó bằng hai màu. Như vậy, một đồ thị hai phía sẽ không có chu trình với số cạnh lẻ.

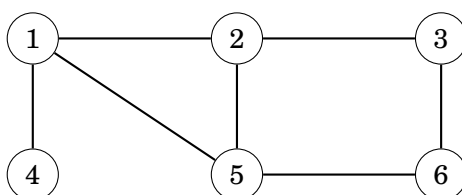
Ví dụ, đồ thị



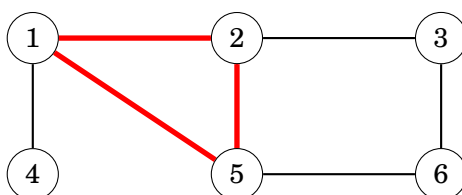
là đồ thị hai phía vì có thể tô nó bằng hai màu như sau:



Tuy nhiên, đồ thị

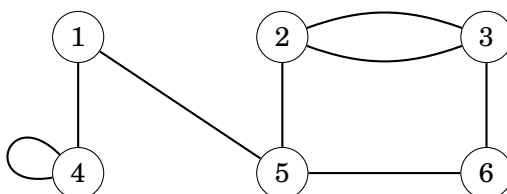


không phải là đồ thị hai phía vì không thể tô màu cho chu trình ba đỉnh bằng cách sử dụng hai màu:



### Đồ thị đơn giản

Một đồ thị gọi là **đơn giản** nếu không có cạnh nào xuất phát và kết thúc tại cùng một đỉnh và không có nhiều cạnh nối giữa hai đỉnh. Trong các bài toán, ta thường giả sử đồ thị là đơn giản. Ví dụ đồ thị dưới đây không phải là đồ thị đơn giản:



## 1.2 Biểu diễn đồ thị

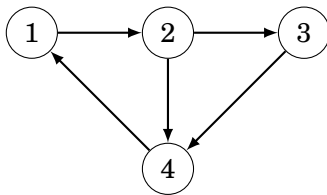
Có một số cách để biểu diễn đồ thị trong thuật toán. Việc lựa chọn cấu trúc dữ liệu phụ thuộc vào kích thước của đồ thị và cách xử lý đồ thị của thuật toán.

### Biểu diễn bằng danh sách kề

Trong biểu diễn danh sách kề, mỗi nút  $x$  trong đồ thị được gán một **danh sách kề** bao gồm các nút có cạnh nối từ đỉnh  $x$ . Danh sách kề là cách phổ biến nhất để biểu diễn đồ thị và hầu hết các thuật toán có thể được triển khai hiệu quả bằng cách sử dụng nó. Một cách thuận tiện để lưu trữ danh sách kề là khai báo một mảng vectơ như sau:

```
vector<int> adj[N];
```

Với hằng số  $N$  là số đỉnh tối đa của đồ thị. Ví dụ, đồ thị



có thể được lưu trữ như sau:

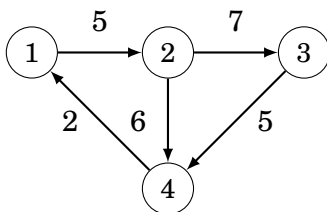
```
adj[1].push_back(2);
adj[2].push_back(3);
adj[2].push_back(4);
adj[3].push_back(4);
adj[4].push_back(1);
```

Nếu đồ thị vô hướng thì vẫn lưu trữ tương tự có điều phải đưa vào danh sách kề của cả hai đỉnh (lúc này có thể coi như là hai hướng).

Với đồ thị có trọng số thì có thể lưu như sau:

```
vector<pair<int,int>> adj[N];
```

Khi đó, danh sách các cạnh kề với đỉnh  $a$  sẽ chứa các cặp  $(b, w)$  ứng với một cạnh nối từ  $a$  đến  $b$  có trọng số là  $w$ . Ví dụ, đồ thị



Có thể lưu trữ như sau:

```
adj[1].push_back({2,5});
adj[2].push_back({3,7});
adj[2].push_back({4,6});
adj[3].push_back({4,5});
adj[4].push_back({1,2});
```

Với việc sử dụng các biểu diễn bằng danh sách kề, ta có thể tìm các đỉnh hiệu quả trong quá trình duyệt đường đi trên đồ thị. Ví dụ, vòng lặp sau sẽ duyệt qua tất cả các đỉnh kề với đỉnh  $s$ .

```
for (auto u : adj[s]) {
    // xử lý đỉnh u kề với s
}
```

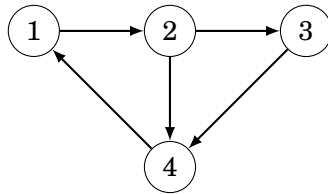
## Biểu diễn đồ thị bằng ma trận kề

**ma trận kề** là một mảng hai chiều cho biết đồ thị chứa các cạnh nào. Cách biểu diễn này cho phép tìm cạnh nối giữa hai đỉnh một cách hiệu quả. Ma trận có thể được lưu trữ dưới dạng một mảng



```
int adj[N][N];
```

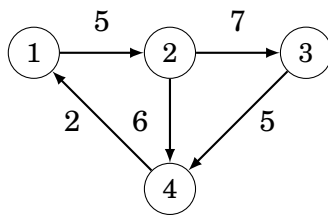
trong đó mỗi giá trị  $adj[a][b]$  cho biết đồ thị có tồn tại cạnh nối hai đỉnh  $a$  và  $b$  hay không. Nếu tồn tại thì  $adj[a][b]=1$ , còn nếu không thì  $adj[a][b]=0$ . Ví dụ, đồ thị



có thể được biểu diễn như sau:

	1	2	3	4
1	0	1	0	0
2	0	0	1	1
3	0	0	0	1
4	1	0	0	0

Nếu đồ thị có trọng số, giá trị của  $adj[a][b]$  được gán bằng trọng số của cạnh nối hai đỉnh  $a$  và  $b$ . Ví dụ, đồ thị



tương ứng với ma trận sau:

	1	2	3	4
1	0	5	0	0
2	0	0	7	6
3	0	0	0	5
4	2	0	0	0

Nhược điểm của phương pháp biểu diễn đồ thị bằng ma trận kề là ma trận gồm  $2^n$  phần tử trong đó hầu hết đều có giá trị bằng 0. Do đó, cách biểu diễn này thường không được sử dụng trong các đồ thị có kích thước lớn.

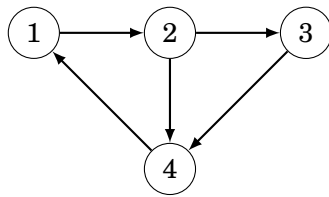
## Biểu diễn đồ thị bằng danh sách cạnh

Một **danh sách cạnh** chứa tất cả các cạnh của đồ thị. Cách biểu diễn này thuận tiện trong trường hợp thuật toán chỉ xử lý các cạnh của đồ thị mà không cần xác định cạnh bắt đầu từ một nút nào đó.

Danh sách kề có thể lưu trữ dưới dạng một vector

```
vector<pair<int,int>> edges;
```

Với mỗi cặp  $(a, b)$  xác định một cạnh nối từ đỉnh  $a$  đến đỉnh  $b$ .  
Theo đó, đồ thị



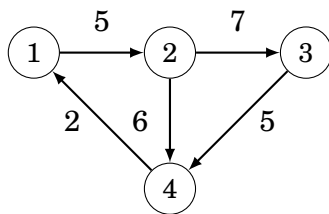
có thể biểu diễn như sau:

```
edges.push_back({1, 2});  
edges.push_back({2, 3});  
edges.push_back({2, 4});  
edges.push_back({3, 4});  
edges.push_back({4, 1});
```

Nếu đồ thị có trọng số thì có thể được lưu như sau:

```
vector<tuple<int, int, int>> edges;
```

Mỗi phần tử trong danh sách có dạng  $(a, b, w)$ , mô tả một cạnh nối từ đỉnh  $a$  đến đỉnh  $b$  có trọng số là  $w$ . Ví dụ, đồ thị



có thể biểu diễn như sau <sup>1</sup>:

```
edges.push_back({1, 2, 5});  
edges.push_back({2, 3, 7});  
edges.push_back({2, 4, 6});  
edges.push_back({3, 4, 5});  
edges.push_back({4, 1, 2});
```

---

<sup>1</sup>trong một số trình biên dịch cũ, hàm `make_tuple` phải được sử dụng thay vì dùng cặp ngoặc nhọn (ví dụ dùng `make_tuple(1, 2, 5)` thay vì dùng `{1, 2, 5}`).