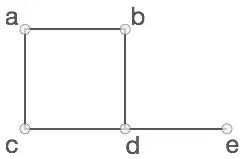
**GRAPH**

1. **Các khái niệm cơ bản.**
2. Các định nghĩa của đồ thị :
3. Đồ thị là một dạng biểu diễn hình ảnh của một tập các đối tượng, trong đó các cặp đối tượng được kết nối bởi các link.

Các đối tượng được nối liền nhau được biểu diễn bởi các điểm được gọi là các đỉnh (vertices), và các link mà kết nối các đỉnh với nhau được gọi là các cạnh (edges).

VD:



-1 đồ thị có dạng: G(V,E) trong đó: V là số đỉnh,E là số cạnh.

1. Phân loại : Đồ thị có thể chia ra thành:

- Đồ thị vô hướng và đồ thị có hướng.

|  |  |
| --- | --- |
| Đồ thị vô hướng | Đồ thị có hướng |
| +Là đồ thị mà các cạnh là các đoạn thẳng. | +Là đồ thị mà các cạnh là các vector đi từ đỉnh này qua đỉnh kia. |
| VD:  IMG_256 | VD:  IMG_256 |

-Đơn đồ thị và đa đồ thị

|  |  |
| --- | --- |
| Đơn đồ thị | Đa đồ thị |
| -Là đồ thị có nhiều nhất 1 cạnh nối giữa 2 đỉnh. | -Là đồ thị có thể có >1 cạnh nối giữa 2 đỉnh. |
| VD: | VD: |

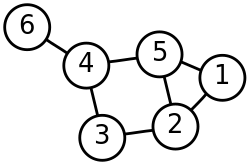
1. Các khái niệm khác:

-Cạnh liên thuộc: Là cạnh nối giữa 2 đỉnh kề.

-Khuyên: Cạnh có 2 đầu trùng nhau(chung 1 đỉnh).

-Đỉnh kề:Là 2 đỉnh mà giữa chúng có cạnh nối.

*-Bậc: bậc* của đỉnh *v* trong đồ thị *G*, ký hiệu *dG*(*v*), là số cạnh liên thuộc với *v*

*VD:*

-Đỉnh 4 có bậc = 3.Cạnh 4-5 là cạnh liên thuộc.Đỉnh kề 4 là 5.

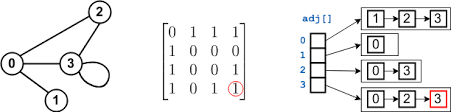
1. **Biểu diễn đồ thị .**
2. Ma trận kề.

+Đồ thị N đỉnh, ta sử dụng 1 ma trận N\*N để lưu, a[I][j] =1 hoặc 0.

A[I][j] =1 khi có cạnh nối từ đỉnh I đến j.

A[I][j] =0 khi không có cạnh nối từ I đến j.

VD:



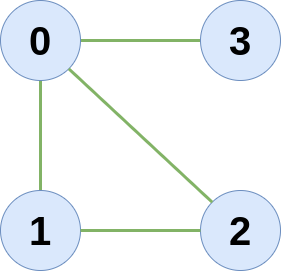
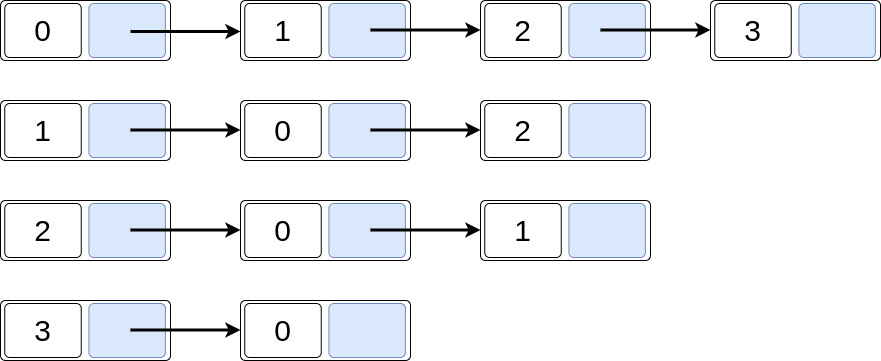
-Tính chất:+Nếu là 1 ma trận vô hướng: Ma trận kề là 1 ma trận đối xứng.

+Có thể dùng để lưu trọng số(Nếu có)

1. Danh sách kề.

-Là 1 list lưu các đỉnh kề với đỉnh đang xét

Vd:

 => 

-Ưu điểm:

+Tiết kiệm bộ nhớ hơn ma trận kề.

+Duyệt tới các đỉnh liền kề tới 1 đỉnh đang xét sẽ nhanh chóng,dễ dàng hơn.

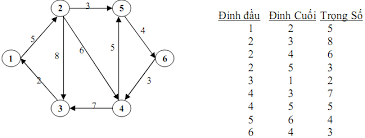
-Nhược điểm: Khi muốn kiểm tra 2 đỉnh có đi đến được nhau không phải duyệt tuần tự.

-Cài đặt: Sử dụng 1 mảng lưu vector.

1. Danh sách cạnh.

-Là 1 danh sách lưu đầu cuối của 1 cạnh.

VD:



-Ưu điểm:

+Tiết kiệm bộ nhớ hơn ma trận kề.

+Duyệt tới các đỉnh liền kề tới 1 đỉnh đang xét sẽ nhanh chóng,dễ dàng hơn.

-Nhược điểm: Khi muốn kiểm tra 2 đỉnh có đi đến được nhau không phải duyệt tuần tự => Rất mất thời gian.

-Cách cài đặt: sử dụng 1 mảng lưu struct chứa đỉnh đầu,đỉnh cuối,trọng số.

1. **Thuật toán tìm kiếm trên đồ thị.**
2. Các khái niệm cơ bản.

-Đường đi: là 1 dãy các đỉnh,và cạnh đi từ đỉnh A->B.

-Chu trình: Là 1 đường đi đơn mà có đỉnh đầu trùng đỉnh cuối.

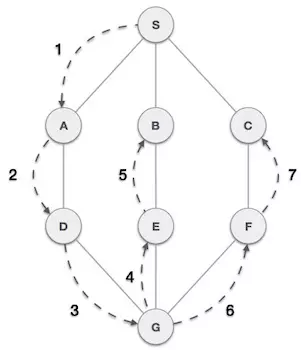
-Đường đi ,chu trình đơn: Là đường đi các đỉnh chỉ xuất hiện 1 lần.

1. DFS.Thuật toán tìm kiếm theo chiều sâu.
2. Ý tưởng:

Giải thuật tìm kiếm theo chiều sâu còn được gọi là giải thuật tìm kiếm ưu tiên chiều sâu, là giải thuật duyệt hoặc tìm kiếm trên một cây hoặc một đồ thị và sử dụng stack (ngăn xếp) để ghi nhớ đỉnh liền kề để bắt đầu việc tìm kiếm khi không gặp được đỉnh liền kề trong bất kỳ vòng lặp nào.

Thuật toán này sẽ duyệt cho tới khi gặp đỉnh ko có đỉnh nào kề nữa hoặc tới khi không còn đỉnh nào chưa đi qua nữa.

VD Mô tả:



b)Quy tắc:

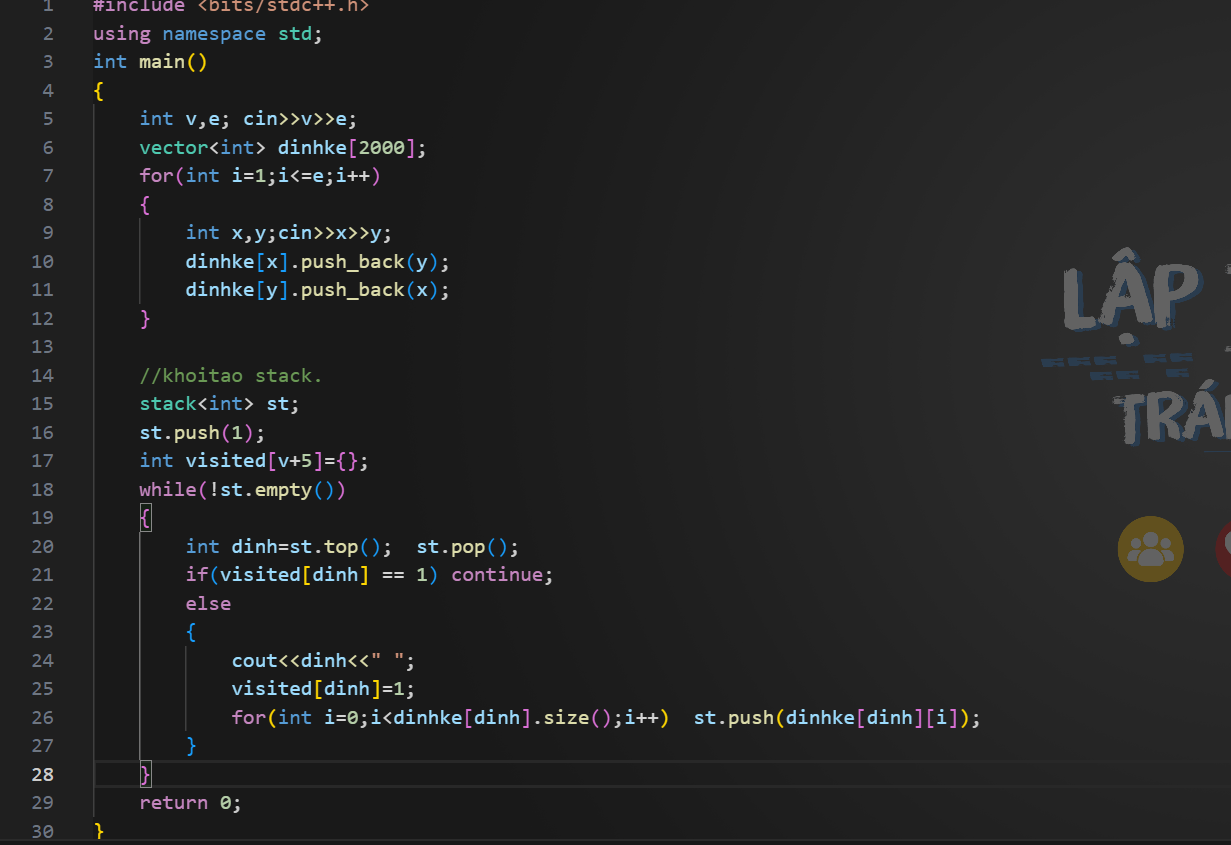
Duyệt tiếp tới đỉnh liền kề mà chưa được duyệt. Đánh dấu đỉnh mà đã được duyệt. Hiển thị đỉnh đó và đẩy vào trong một ngăn xếp (stack).

Nếu không tìm thấy đỉnh liền kề, thì lấy một đỉnh từ trong ngăn xếp (thao tác pop up). (Giải thuật sẽ lấy tất cả các đỉnh từ trong ngăn xếp mà không có các đỉnh liền kề nào).

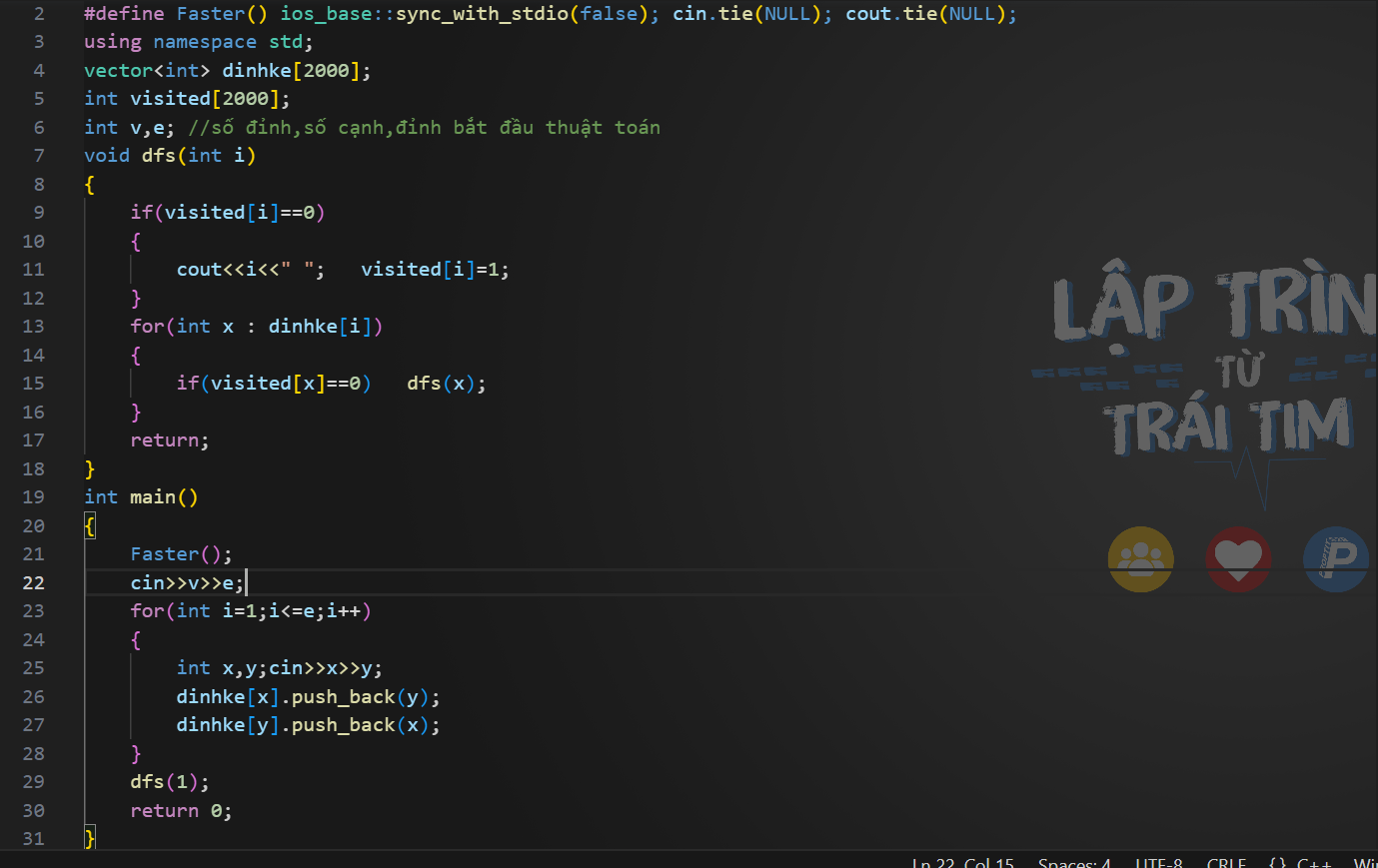
Lặp lại các qui tắc 1 và qui tắc 2 cho tới khi ngăn xếp là trống.

c) Cách làm:

C1: Sử dụng stack.

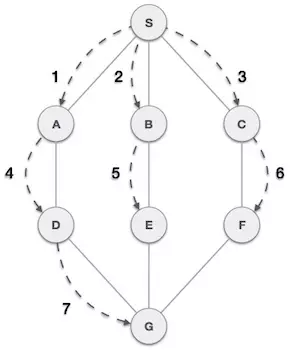


C2.Cài đặt đệ quy.



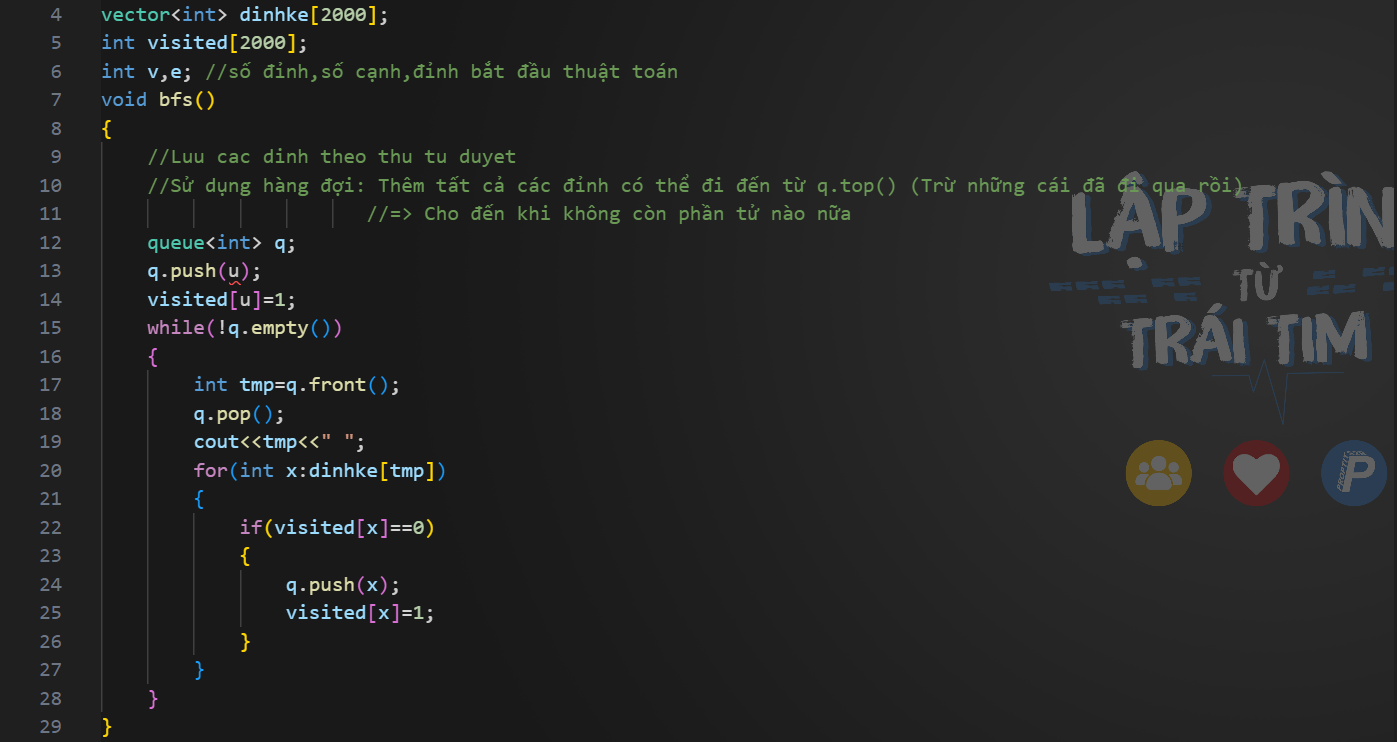
1. BFS. Thuật toán tìm kiếm theo chiều rộng.

Giải thuật tìm kiếm theo chiều rộng duyệt qua một đồ thị theo chiều rộng và sử dụng hàng đợi (queue) để ghi nhớ đỉnh liền kề để bắt đầu việc tìm kiếm khi không gặp được đỉnh liền kề trong bất kỳ vòng lặp nào.



-Ý tưởng: Duyệt qua tất cả các đỉnh kề với đỉnh đang xét,lưu vào queue trước,sau đó mới đi tiếp tới các đỉnh đằng sau.

-Cài đặt bằng queue:



-Thuật toán loang: Là 1 ứng dụng của BFS.

+ Giả sử ta có 1 vệt dầu đang ở đỉnh X. Từ đỉnh X có đường đi đến các đỉnh Y,Z,T. Vệt dầu sẽ loang đến cả 3 đỉnh Y,Z,T và làm tương tự X với Y,Z,T.

+Thuật toán này được ứng dụng trong bài toán tìm đường đi trên 1 đồ thị.

4.Độ phức tạp của BFS và DFS đều là O(E+V).

**IV.Tính liên thông của đồ thị.**

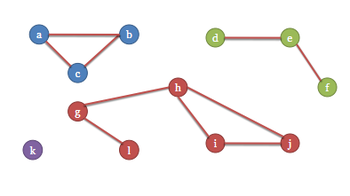
1.Các định nghĩa trong đồ thị vô hướng:

-Đồ thị liên thông là đồ thị có đường đi giữa 2 điểm bất kì

trong đồ thị.

-Thành phần liên thông là các đồ thị con liên thông trong 1 đồ thị lớn.

VD:

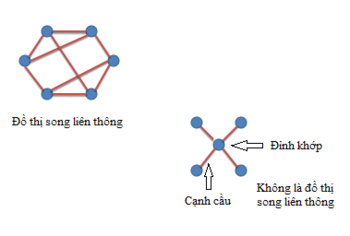


-**Đỉnh khớp** (cut vertex/ articulation point): của một đồ thị vô hướng là đỉnh mà nếu xóa đỉnh này khỏi đồ thị và các cạnh nối đến nó thì số thành phần liên thông của đồ thị sẽ tăng thêm.

**-Cạnh cầu** (bridge): của một đồ thị vô hướng là cạnh mà nếu xóa đi khỏi đồ thị thì số thành phần liên thông của đồ thị sẽ tăng thêm.

-**Đồ thị song liên thông** (biconnectivity): là đồ thị không chứa đỉnh khớp.

VD:



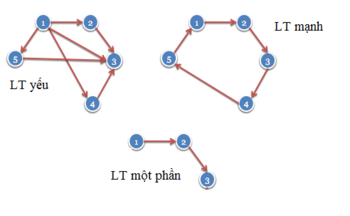
2.Các định nghĩa trong đồ thị có hướng:

-Liên thông mạnh: là 1 đồ thị mà ở mọi cặp đỉnh a-b đều có đường đi từ a-> b và từ b-> a.

*-Liên thông yếu* (weakly connected): Đồ thị có hướng gọi là liên thông yếu nếu có đường đi giữa 2 đỉnh bất kỳ của đồ thị vô hướng tương ứng với đồ thị đã cho. Tức là hủy bỏ các hướng của các cạnh trong đồ thị.

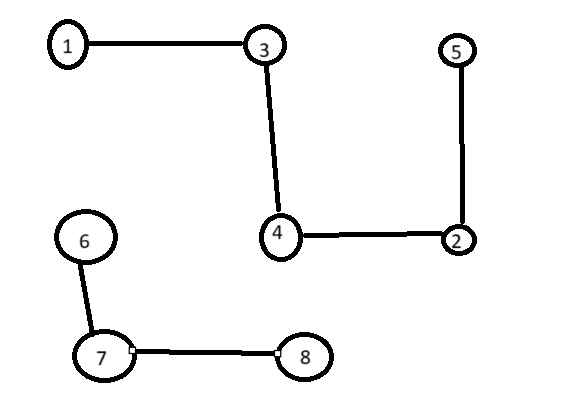
*-Liên thông một phần* (unilaterally connected): Đồ thị có hướng gọi là liên thông một phần nếu với mọi cặp đỉnh a, b bất kỳ, có ít nhất một đỉnh đến được đỉnh còn lại.

VD:



3.In ra các thành phần liên thông.

VD:



-Ý tưởng: Ta duyệt từ đỉnh 1-> V. Sử dụng DFS hoặc BFS để duyệt các đỉnh thuộc liên thông với đỉnh đó,đánh dấu nó lại.

CODE:

