# A,Đồ thị vô hướng

## 1, Chu trình Euler và chu trình Hamilton

-Sử dụng code C++ (bai1.cpp) để chạy test ta thấy rằng cả 3 đồ thị nếu không có chu trình euler và chu trình Hamilton

## 2, Đếm đồ thị

Có bao nhiêu đồ thị vô hướng khác nhau có V đỉnh và E cạnh (không có cạnh song song)?

Số lượng đồ thị vô hướng khác nhau có V đỉnh và E cạnh (không có cạnh song song) có thể được tính bằng công thức "C(V, E) \* 2^(E - V + 1)", trong đó C(V, E) là hệ số nhị thức và có giá trị là tổ hợp chập V lấy E.

Để tính toán số lượng này, chúng ta cần xác định E cạnh nào sẽ được chọn từ tổng số C (V, E) cách chọn cạnh từ V đỉnh. Sau đó, mỗi cạnh có thể có hai trạng thái: có hoặc không có trong đồ thị. Do không có cạnh song song, nên số lượng cạnh tối đa là V - 1.

Vì vậy, tổng số đồ thị vô hướng khác nhau có V đỉnh và E cạnh (không có cạnh song song) là

C (V, E) \* 2 ^ (E - V + 1).

## 3, Phát hiện cạnh song song

Thuật toán sau đây sử dụng một loạt các vòng lặp để duyệt qua đỉnh và cạnh của đồ thị, và áp dụng các điều kiện để xác định xem cặp cạnh nào là cặp cạnh song song.

Hàm đếm số cạnh song song có thể được thực hiện theo các bước sau

1. Duyệt qua từng đỉnh u của đồ thị:

for (int u = 0; u < soDinh; ++u) {

Bắt đầu vòng lặp, duyệt qua từng đỉnh của đồ thị.

2. Duyệt qua từng đỉnh kề v của đỉnh u:

for (int v: graph[u]) {

Với mỗi đỉnh u, duyệt qua danh sách kề của nó để xem có cạnh nào kề với nó hay không.

3. Kiểm tra điều kiện u < v:

if (u < v) {

Điều này đảm bảo rằng mỗi cặp cạnh chỉ được kiểm tra một lần. Ngăn chặn việc kiểm tra cặp cạnh hai lần với thứ tự ngược lại (v, u).

4. Tạo một set chứa các đỉnh kề của `u` (`dinhKeU`):

unordered\_set<int> dinhKeU(graph[u].begin(), graph[u].end());

Sử dụng `unordered\_set` để dễ dàng kiểm tra sự hiện diện của một đỉnh trong danh sách kề của `u`.

5. Duyệt qua từng đỉnh kề w của đỉnh v:

for (int w: graph[v]) {

Với mỗi đỉnh kề của v, kiểm tra xem cạnh (u, v) và (v, w) có phải là cạnh song song không.

6. Kiểm tra có cạnh song song không:

if (w != u && dinhKeU.find(w) == dinhKeU.end()) {

- w != u: Đảm bảo đỉnh w không trùng với u.

- dinhKeU.find(w) == dinhKeU.end(): Kiểm tra xem w có kề với u hay không. Nếu không, thì cạnh (u, v) và (v, w) là cạnh song song.

7. Tăng biến đếm nếu cạnh song song được tìm thấy:

soCanhSongSong++;

8. Kết quả:

return soCanhSongSong;

Trả về số cạnh song song được đếm trong đồ thị.

## 4, Chu trình lẻ

Để chứng minh rằng một đồ thị là đồ thị hai mầu (bipartite) khi và chỉ khi nó không chứa chu trình độ dài lẻ, ta sẽ sử dụng phương pháp phản chứng như gợi ý.

Giả sử rằng có một đồ thị G có chu trình độ dài lẻ và vẫn được tô màu theo hai mầu sao cho không có cạnh nối giữa hai đỉnh cùng mầu. Chúng ta sẽ chứng minh rằng G không thể là một đồ thị hai mầu.

Xét một chu trình độ dài lẻ bất kỳ trong G và xác định hai đỉnh bất kỳ trên chu trình đó. Gọi hai đỉnh này là u và v. Vì G là đồ thị hai mầu, ta gán màu khác nhau cho u và v.

Trên chu trình đó, xét hai đỉnh kề nhau, gọi chúng là x và y. Theo giả thiết, hai đỉnh x và y cùng thuộc cùng một mầu. Nhưng khi đó, hai đỉnh x và y được nối bởi một cạnh, điều này mâu thuẫn với giả thiết rằng không có cạnh nối giữa hai đỉnh cùng mầu.

Do đó, giả thiết ban đầu là sai và ta có thể kết luận rằng nếu một đồ thị không chứa chu trình độ dài lẻ, thì nó là một đồ thị hai mầu.

Tương tự, chúng ta có thể chứng minh ngược lại rằng nếu một đồ thị là đồ thị hai mầu, thì nó không chứa chu trình độ dài lẻ.

Vì vậy, ta có thể kết luận rằng một đồ thị là đồ thị hai mầu khi và chỉ khi nó không chứa chu trình độ dài lẻ.

## 5, Biconnected

Một đồ thị được gọi là biconnected nếu mỗi cặp đỉnh đều được nối với nhau bởi hai đường đi không giao nhau. Trong đồ thị liên thông, điểm articulation là đỉnh mà khi xóa nó và các cạnh kề sẽ làm đồ thị mất tính liên thông. Hãy chứng minh rằng một đồ thị bất kì mà không có điểm articulation là đồ thị biconnected.

Chứng minh:

Để chứng minh rằng một đồ thị bất kỳ mà không có điểm articulation là đồ thị biconnected, ta sẽ sử dụng phản chứng.

Giả sử rằng đồ thị G không có điểm articulation, nhưng không phải là đồ thị biconnected. Điều này có nghĩa là tồn tại một cặp đỉnh s và t trong G sao cho không có đường đi nào nối giữa chúng mà không đi qua một điểm articulation.

Giả sử ta có một đường đi P nối giữa s và t trong G. Vì G không biconnected, ta có thể chia đường đi P thành hai phần, gọi chúng là P1 và P2, sao cho không có đỉnh nào thuộc P1 nối với đỉnh nào thuộc P2 (nghĩa là không có cạnh nào cắt qua P1 và P2).

Giả sử x là một đỉnh thuộc P1 và y là một đỉnh thuộc P2. Khi đó, ta có đường đi từ s đến x qua P1, sau đó từ y đến t qua P2. Nhưng không có cạnh nối giữa các đỉnh thuộc P1 và P2, điều này mâu thuẫn với giả thiết rằng không có đường đi nào nối giữa s và t mà không đi qua một điểm articulation.

Do đó, giả thiết ban đầu là sai và ta có thể kết luận rằng một đồ thị bất kỳ mà không có điểm articulation là đồ thị biconnected.

Vì vậy, ta có thể chứng minh rằng một đồ thị không có điểm articulation là đồ thị biconnected.

Hoặc ta có thể nhận thấy các đỉnh nằm cùng trên 1 chu trình đơn sẽ cùng thuộc 1 thành phần song liên thông (biconnected) vì cho dù có xóa cạnh nào đi nữa thì vẫn còn 1 đường đi giữa 2 cặp đỉnh bất kì trong 1 chu trình đơn do trong 1 chu trình đơn của 1 đồ thị vô hướng, luôn tồn tại nhiều hơn 1 đường đi giữa 2 cặp đỉnh bất kì. Do đó, nếu 1 đồ thị là đồ thị biconnected, thì cả đồ thị đó sẽ là 1 chu trình đơn. Theo như đề bài, nếu không có điểm articulation (đồ thị có xóa 1 cạnh bất kì nào đi nữa thì cũng vẫn liên thông) => Đồ thị là 1 chu trình đơn. Mà do đồ thị là 1 chu trình đơn nên nó sẽ là 1 đồ thị Biconnected => Điều phải chứng minh.