<https://www.hackerearth.com/challenges/competitive/code-monk-graph-theory-i/algorithm/monks-birthday-treat/>

# Tiệc sinh nhật của người keo kiệt (party)

Anh Kiệt là một người keo kiệt. Anh ta không muốn chi tiền cho bất kỳ thứ, dù tốt hay xấu. Nhưng lần này bạn bè của anh ấy gợi ý một bữa tiệc sinh nhật, nếu anh ta không tổ chức thì họ sẽ cắt đứt quan hệ bạn bè với anh ta.

Kiệt nhận ra rằng không có cách nào để cứu các quan hệ bạn bè của mình. Vì vậy, anh ta quyết định mở tiệc chiêu đãi bạn bè. Nhưng anh ta vẫn cho rằng có thể bỏ qua việc mời những người bạn mà mình không có bất kỳ ràng buộc nào. Nói thêm về ràng buộc: Giả sử Kiệt có *N* người bạn và một người bạn *a* sẽ không đến bữa tiệc nếu người *b* không được mời, nhưng không nhất thiết điều ngược lại cũng đúng.

Kiệt muốn mời số lượng bạn bè tối thiểu tới dự bữa tiệc sinh nhật của mình để tiết kiệm tiền. Hãy giúp Kiệt tìm ra số bạn bè tối thiểu mà anh ta có thể mời mà đáp ứng nhu cầu của họ. Anh ấy không thể tổ chức một bữa tiệc mà không có bất kỳ người bạn nào, vì vậy, anh ấy đã mời một người bạn đi cùng, ít nhất là để chúc mừng sinh nhật của anh ấy.

### Đầu vào

* Dòng đầu tiên sẽ chứa hai số nguyên *N* và *M*, trong đó *N* biểu thị số lượng bạn bè của Kiệt, *M* biểu thị số lượng quan hệ phụ thuộc.
* Mỗi dòng trong *M* dòng tiếp theo chứa hai số nguyên *a,* *b* đại diện cho quan hệ ràng buộc là *a* chỉ đến dự bữa tiệc khi và chỉ khi *b* cũng được mời.

### Đầu ra

In ra số lượng bạn bè tối thiểu mà Kiệt có thể mời dự sinh nhật.

### Ràng buộc

* 1 ≤ *N, M* ≤ 1000
* 1 ≤ *a* ≠ *b* ≤ *N*

### Ví dụ

|  |  |
| --- | --- |
| Đầu vào | Đầu ra |
| 3 3  1 2  2 3  3 1 | 3 |

### Solution

Phát biểu bài toán: ta có một tập *M* cặp nút biểu thị sự phụ thuộc của 2 người bạn. Gọi một cặp (*a, b*) biểu thị rằng người bạn *a* chỉ đến dự tiệc cho đến khi và chỉ khi *b* cũng được mời. Nó ngụ ý rằng nếu *a* dự tiệc thì *b* cũng dự.

#### Giải pháp 1: BFS/DFS.

Hoạt động 1: Tạo một mảng toàn cục *set[N]* gồm *N* nút với mỗi nút là một SET

Hoạt động 2: Với mỗi cặp (*a, b*), ta thêm một cạnh từ *a* đến *b*, nghĩa là chèn *b* vào *set[a]*. Do đó, nếu *a* được chọn, thì *b* tự động được chọn.

Hoạt động 3: Bây giờ với mỗi nút *i* và duyệt qua các cạnh cho đến tất cả các nút lá và đếm tổng số nút được kết nối trực tiếp / gián tiếp với nút *i*. Trước khi duyệt từ i, ta dùng một mảng *visit[N]* để kiểm tra xem một nút đã được thăm hay chưa. Ban đầu *visit[N] = {false}*.

Ta sẽ tìm được thành phần liên thông tối thiểu từ mỗi nút *i*.

Trong hàm DFS, chúng ta đếm các nút bằng cách duyệt qua từng nút, sử dụng hàng đợi và mảng *visit*.

#### Giải pháp 2: DSU

Bài toán cũng có thể được giải quyết bằng DSU có trọng số. Gợi ý: Tìm trọng số tối thiểu trong tất cả các trọng số.

*Lưu ý*: Chúng tôi sử dụng kỹ thuật Fast IO để nhập và xuất nhanh hơn.