1 BuildingRoads

Cho N thành phố được đánh số từ 1 đến N, và M con đường đã xây giữa một số cặp thành phố. Mỗi con đường nối hai thành phố a và b.

Mục tiêu là thêm vào một số tối thiểu các con đường sao cho mọi thành phố đều nằm trong cùng một thành phần liên thông.

1. Cấu trúc DSU

Mỗi thành phố ban đầu là một tập riêng biệt.

- Mảng `ds[i]` lưu thông tin về gốc của tập chứa thành phố i.
- Hàm `find(u)` trả về gốc của tập chứa u, đồng thời áp dụng *path compression*:

$$\text{find}(u) = \begin{cases} u & \text{nếu } ds[u] < 0 \\ \text{find}(ds[u]) & \text{ngược lại} \end{cases}$$

- Hàm `merge(u, v)` hợp nhất hai tập nếu u và v thuộc hai tập khác nhau:

$$\text{merge}(u, v) = \begin{cases} \text{false} & \text{nếu find}(u) = \text{find}(v) \\ \text{gộp hai tập và trả về true} & \text{ngược lại} \end{cases}$$

2. Thuật toán chính

1. Khởi tạo: `ds[i] = -1` với mọi i từ 1 đến N.
2. Với mỗi cạnh (a, b) đầu vào, gọi `merge(a, b)` để nối các thành phố đã liên thông.
3. Duyệt từ $i = 1$ đến $N - 1$:
   - Nếu `merge(i, i+1)` thành công, nghĩa là i và $i + 1$ chưa liên thông, thì thêm cạnh $(i, i + 1)$ vào danh sách kết quả.
4. In ra số đường thêm vào và danh sách các đường đó.

3. Độ phức tạp

Với kỹ thuật *path compression* và *union by size*, mỗi phép `find` hay `merge` có độ phức tạp gần $\mathcal{O}(1)$ (chính xác là $\mathcal{O}(\alpha(N))$ với α là hàm nghịch đảo Ackermann).

Tổng độ phức tạp:

$$\mathcal{O}(M \cdot \alpha(N) + N \cdot \alpha(N)) \approx \mathcal{O}(N)$$

2 CountingRooms

Mô tả bài toán

Cho một mê cung dưới dạng lưới $N \times M$ với mỗi ô là:

- `.` — ô trống có thể đi vào.
- `#` — tường không thể đi qua.

Hai ô trống được xem là **liên thông** nếu chúng kề nhau theo hướng lên, xuống, trái, hoặc phải.

Yêu cầu: Đếm số vùng liên thông các ô trống — gọi là *số phòng*.

Ý tưởng thuật toán

1. Duyệt toàn bộ ma trận.
2. Với mỗi ô chưa được thăm và là ô trống `'.'`, thực hiện thuật toán DFS để đánh dấu tất cả các ô trong vùng liên thông.
3. Mỗi lần gọi DFS tương ứng với một phòng mới.

Hàm DFS

Giả sử đang ở ô (x, y), đánh dấu là đã thăm:

$$\text{visited}[x][y] \leftarrow \text{true}$$

Sau đó, thử di chuyển sang 4 hướng:

$$(x+1, y),\ (x-1, y),\ (x, y+1),\ (x, y-1)$$

Nếu ô mới hợp lệ, không phải tường, chưa được thăm, thì tiếp tục gọi đệ quy.

Độ phức tạp thời gian

Mỗi ô được thăm nhiều nhất 1 lần, nên tổng thời gian là:

$$\mathcal{O}(N \cdot M)$$