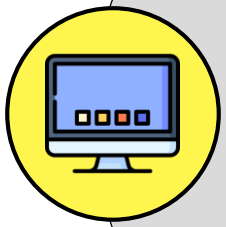
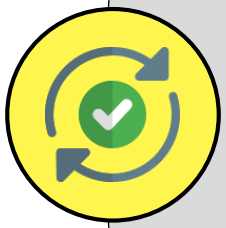
DISJOINT SET UNION

Giới thiệu:



Cấu trúc các tập hợp rời nhau (DSU) hay Union Find là một cấu trúc dữ liệu cực kì quan trọng và hiệu quả.

DSU giúp giải quyết bài toán: Cho một tập hợp các phần tử, ban đầu mỗi phần tử là một tập hợp riêng biệt. DSU cho phép gộp 2 tập hợp với nhau hoặc chỉ ra phần tử bất kì đang thuộc tập hợp nào.



Trong tài liệu này, mình sử dụng mỗi phần tử là một đỉnh của đồ thị cho dễ tiếp cận, tuy nhiên phạm vi của DSU không chỉ áp dụng trong đồ thị.

Ban đầu đồ thị chứa N đỉnh, mỗi đỉnh sẽ là một tập hợp riêng biệt và có đại diện (cha) là chính đỉnh đó. Mỗi lần bạn gộp 2 đỉnh lại với nhau, tương tự như có một cạnh nối giữa 2 đỉnh này.

1. Các thao tác của DSU:

a. Thao tác khởi tạo:

Hàm khởi tạo này sẽ gán đại diện cho các đỉnh là chính nó

```
int n; // số đỉnh của đồ thị
int maxn = 1005;
int parent[maxn];

void init(){
    cin >> n;
    for(int i=1; i<=n; i++){
        parent[i] = i;
    }
}
```

1. Các thao tác của DSU:

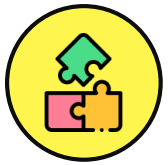
b. Thao tác tìm đại diện:

Hàm này sẽ giúp bạn tìm đại diện của tập hợp mà đỉnh cần tìm đang thuộc về. Đỉnh x đại diện cho tập hợp chứa nó nếu: x = parent[x].

```
int Find(int u){
    while(u != parent[u]){
        u = parent[u];
    }
    return u;
}
```

1. Các thao tác của DSU:

c. Thao tác gộp:



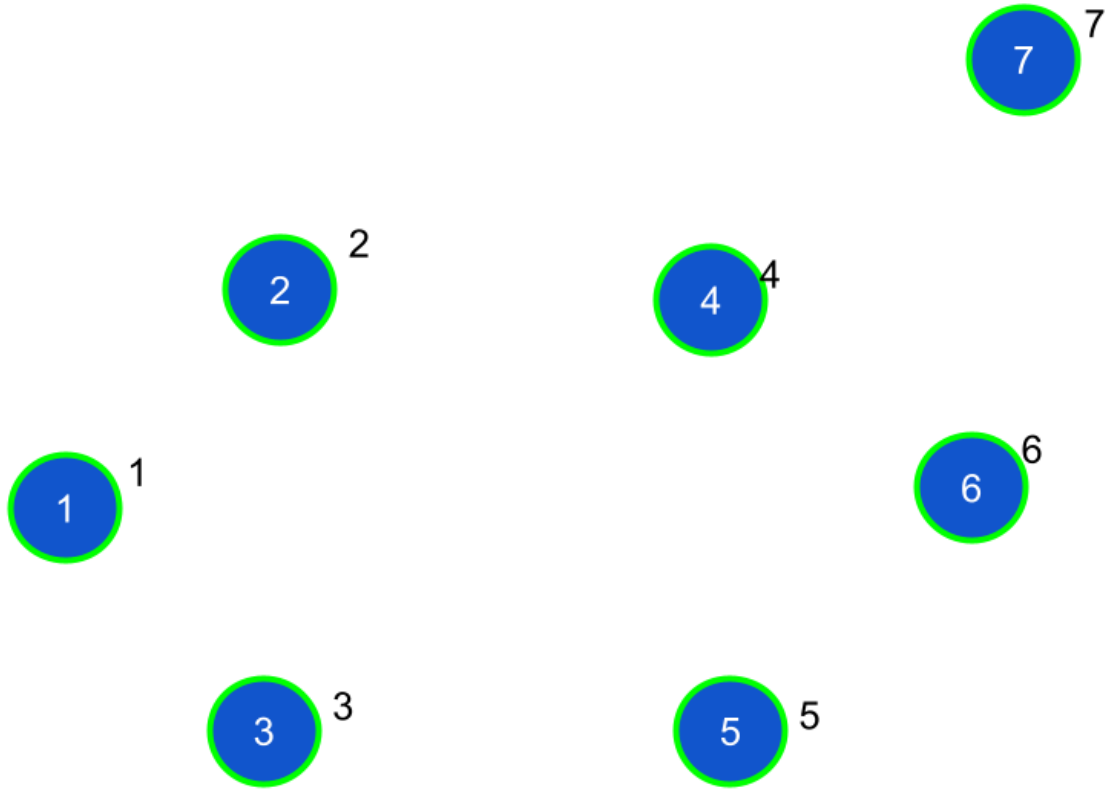
Mỗi khi bạn muốn gộp 2 đỉnh u, v vào làm một hay có một cạnh nối giữa 2 đỉnh u, v đầu tiên bạn phải xác định xem u và v có đại diện là gì.

Trường hợp u, v cùng đại diện thì không cần phải làm gì cả, nếu u và v khác đại diện khi đó bạn cần gộp 2 tập hợp chứ u và v lại với nhau bằng cách gán đại diện của tập hợp chứa u là cha của đại diện của tập hợp chứa v hoặc ngược lại. Ở đây sau khi tìm được đại diện của u và v mình sẽ chọn đỉnh có số thứ tự nhỏ hơn làm đại diện của của tập hợp sau khi gộp.

```
bool Union(int u, int v){
    u = Find(u);
    v = Find(v);
    if(u == v) return false;
    if(u < v) parent[v] = u;
    else parent[u] = v;
    return true;
}
```

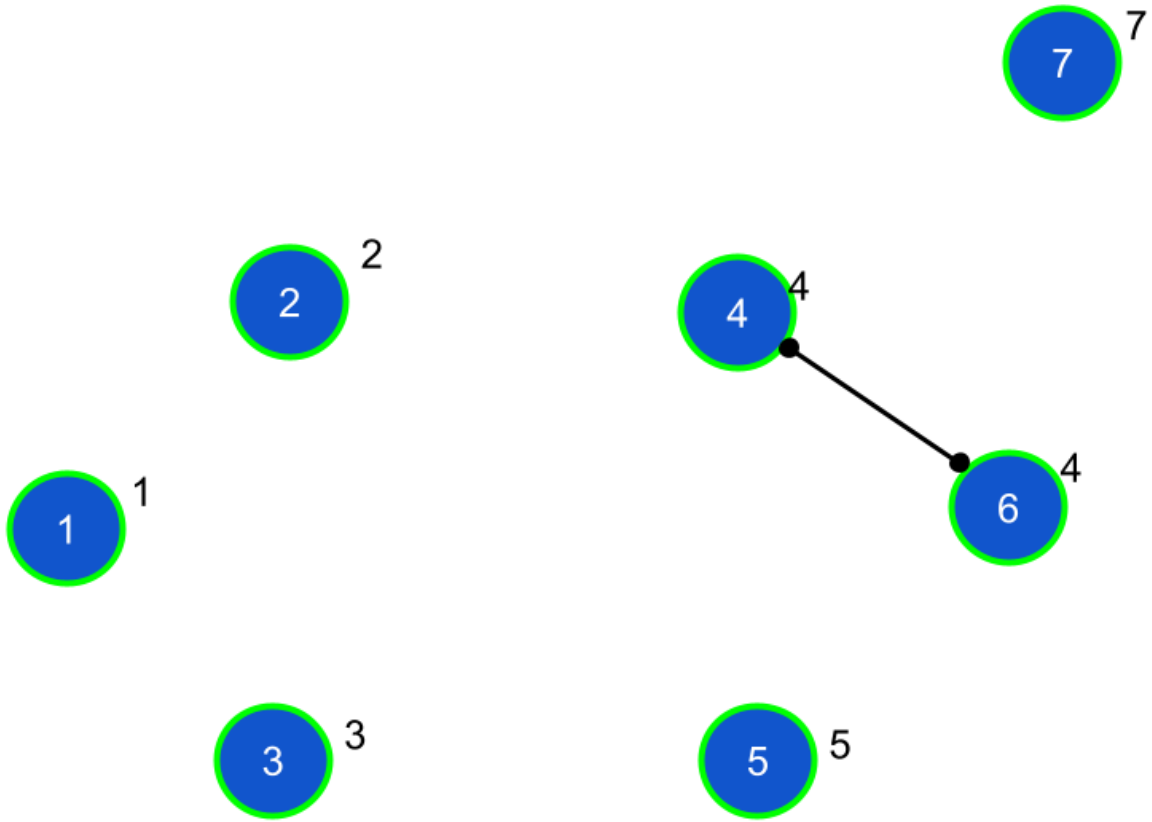
2. Ví dụ DSU:

Trong ví dụ này mình lấy N = 7. Ban đầu các đỉnh trên đồ thị đều có đại diện là chính nó.



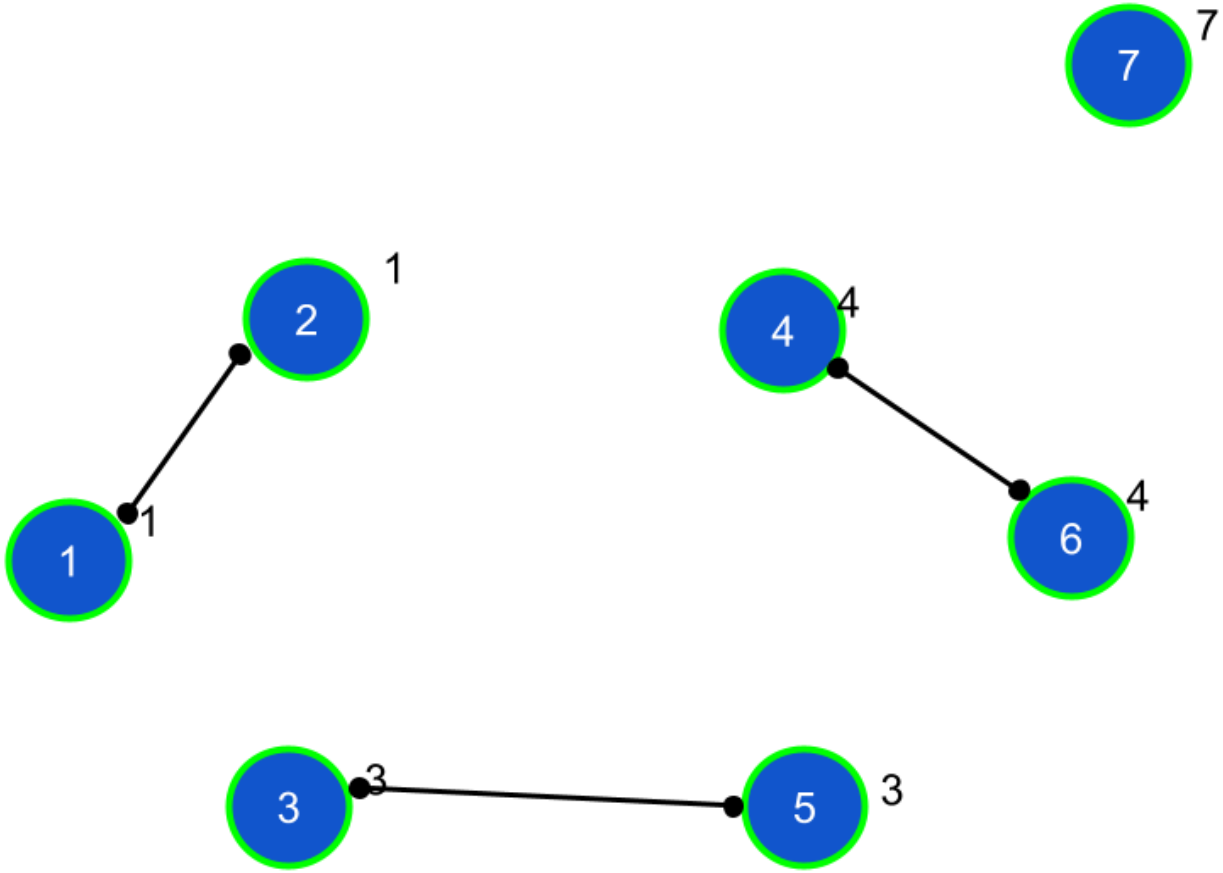
2. Ví dụ DSU:

Gộp 4 và 6, đại diện của 6 bây giờ là 4



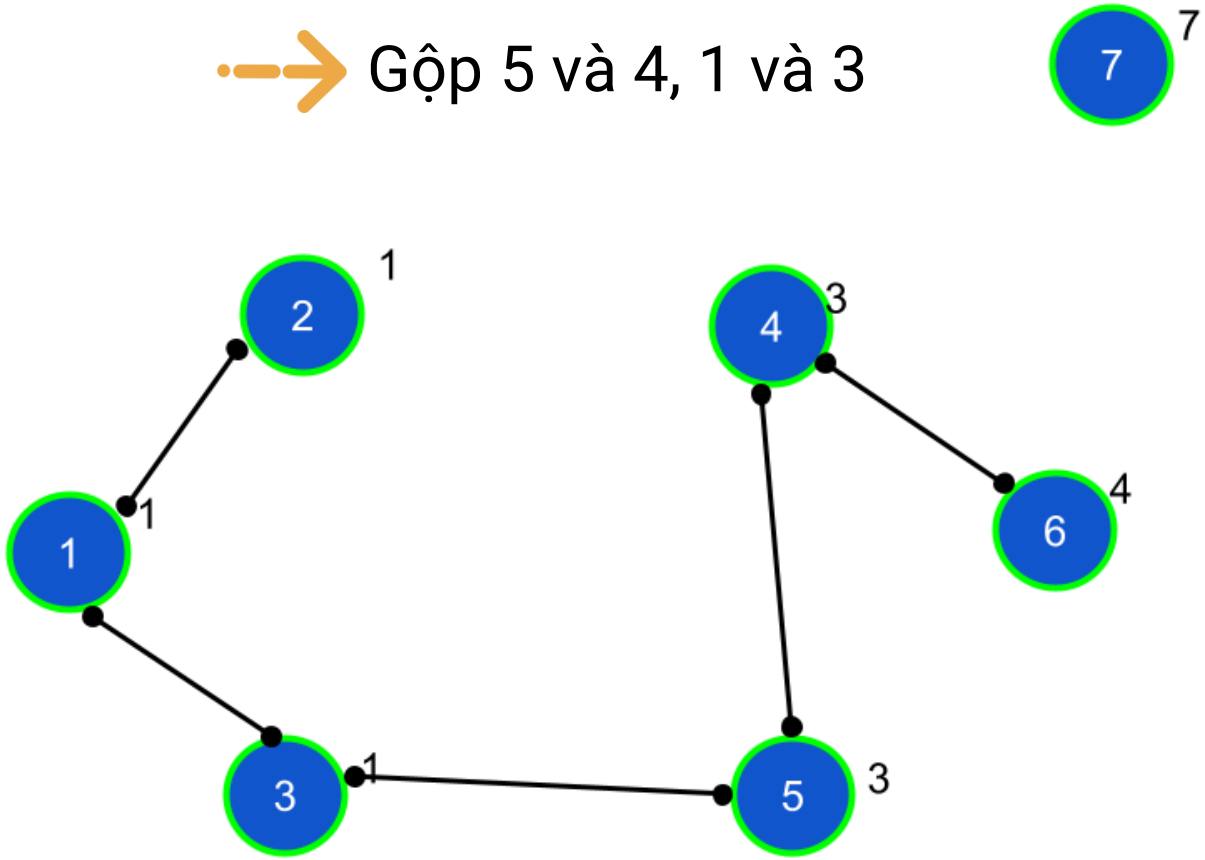
2. Ví dụ DSU:

Gộp 3 và 5, 1 và 2



2. Ví dụ DSU:

Gộp 5 và 4, 1 và 3

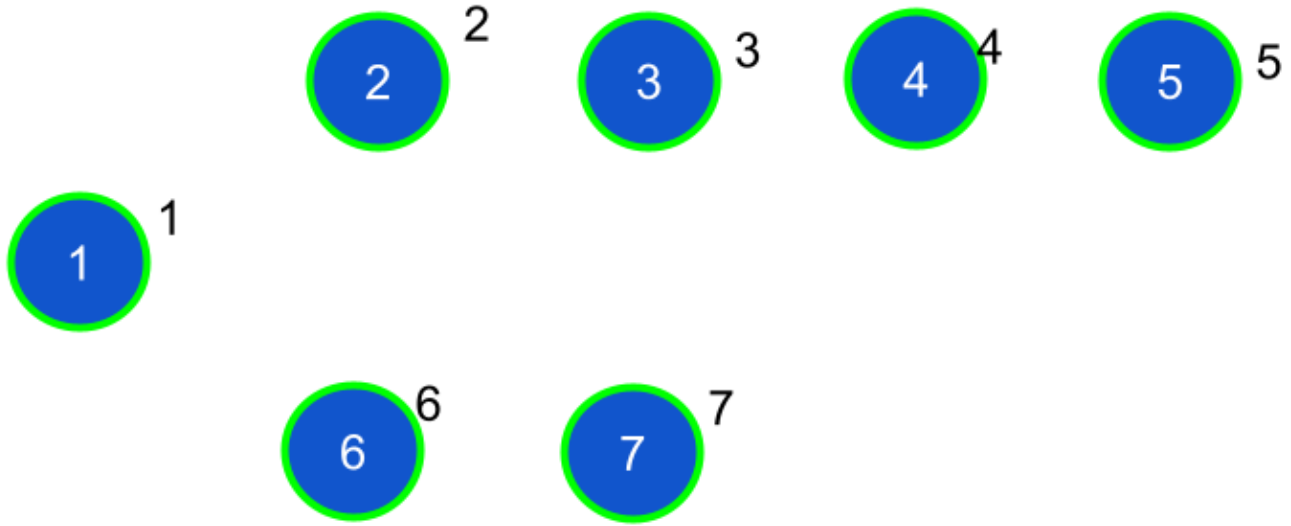


Gộp 2 và 5 : Không có gì thay đổi vì khi tìm đại diện của 2 ta tìm được 1, tìm đại diện của 5 ta cũng tìm được 1. Vì thế 2 và 5 đã thuộc cùng một tập hợp.

3. Tối ưu DSU:

a. Tối ưu nén đường (Path compression optimization):

Giả sử ban đầu bạn có đồ thị như sau :

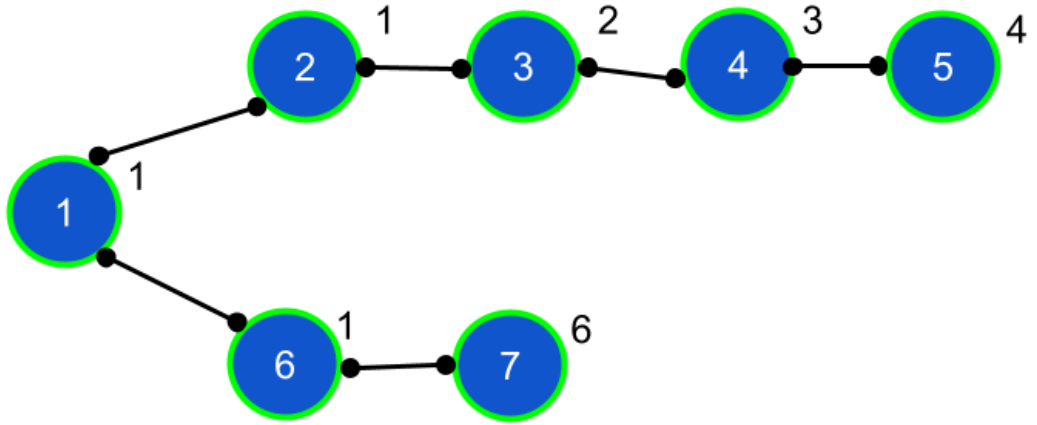


Thực hiện gộp lần lượt các cặp đỉnh theo thứ tự sau : (4, 5), (3, 4), (3, 2), (2, 1), (6, 7), (1, 6).

3. Tối ưu DSU:

a. Tối ưu nén đường (Path compression optimization):

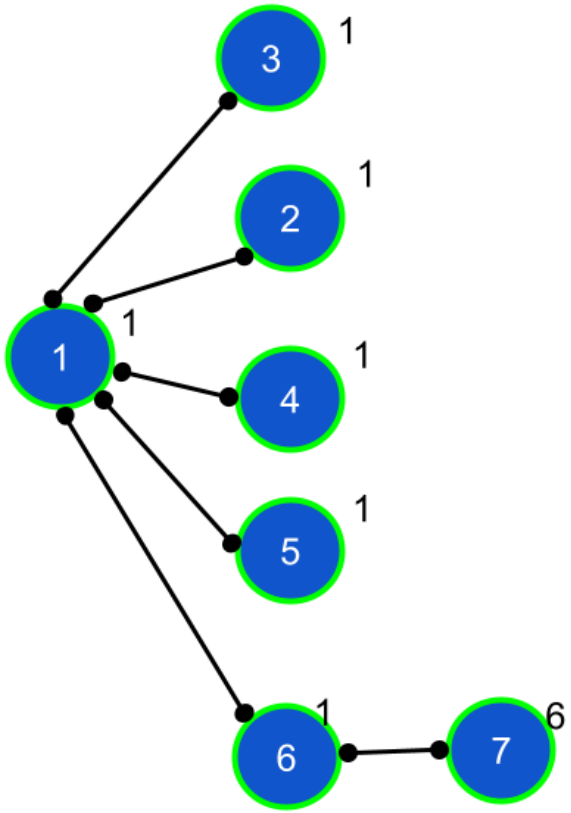
Ta quan sát thấy các đỉnh từ 1 tới 7 đều có đại diện là 1, giả sử bây giờ bạn cần gộp đỉnh 4 với 1 đỉnh nào đó thì bạn phải tìm đại diện của 1 theo thứ tự đỉnh :4 - 3 - 2 - 1 và tìm được đại diện của nó là 1. Trong quá trình tìm đại diện của mình, bạn hoàn toàn có thể gán đại diện cho các đỉnh trên đường đi từ 4 về 1 có đại diện là 1, việc này sẽ làm cho đường đi từ 1 đỉnh tới đại diện của nó ngắn hơn.



3. Tối ưu DSU:

a. Tối ưu nén đường (Path compression optimization):

→ Sau khi tối ưu nén đường:



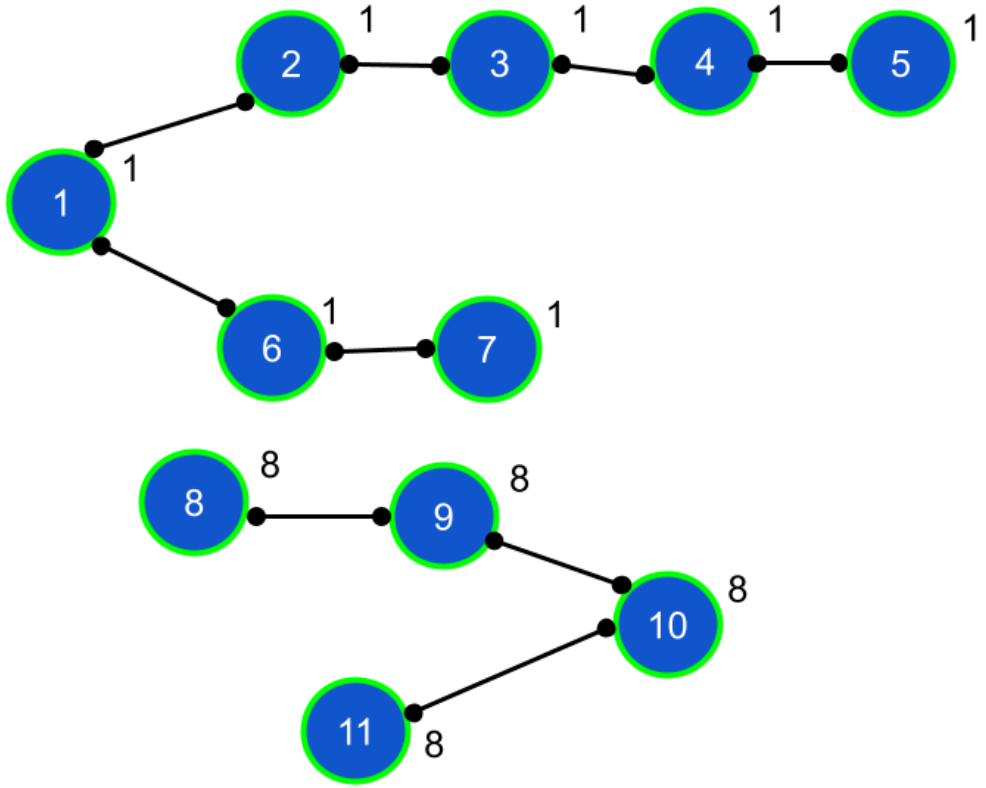
Code tối ưu nén đường

```
int Find(int u){
    if(u == parent[u])
        return u;
    else
        return parent[u] = Find(parent[u]);
}
```

3. Tối ưu DSU:

b. Tối ưu gộp:

Giả sử sau khi gộp cả đỉnh trên đồ thị bạn có đồ thị như sau:



3. Tối ưu DSU:

b. Tối ưu gộp:

Bây giờ để gộp 2 đỉnh (5, 10) bạn tìm được đại diện của 5 là 1 và đại diện của 10 là 8. Đến đây ta phải quyết định gán 8 là đại diện của 1 hay 1 là đại diện của 8.

Trường hợp 1

Khi gán đại diện của 1 là 8 tức parent[1] = 8 khi đó các đỉnh 2, 3, 4, 5, 6, 7 sẽ có đại diện mới là 8.

Trường hợp 2

Khi gán đại diện của 8 là 1 tức parent[8] = 1 khi đó các đỉnh 9, 10, 11 sẽ có đại diện mới là 1.

Để tránh việc có quá nhiều đỉnh bị gán lại đại diện, sẽ mất thêm nhiều chi phí tìm đại diện hơn trong các lần gộp sau ta cần lựa chọn đỉnh đại diện có số phần tử trong tập hợp nhiều hơn làm đại diện. Trong ví dụ trên 1 đại diện cho tập hợp có 7 đỉnh sẽ làm đại diện cho 8, 8 là đại diện của tập có 4 đỉnh.

3. Tối ưu DSU:

b. Tối ưu gộp:

Code tối ưu gộp theo kích thước

```
int n; // số đỉnh của đồ thị
int maxn = 1005;
int parent[maxn], size[maxn];

void init(){
    cin >> n;
    for(int i = 1; i <= n; i++){
        parent[i] = i;
        sz[i] = 1;
    }
}
```

```
bool Union(int u, int v){
    u = Find(u);
    v = Find(v);
    if(u == v) return false;
    if(sz[u] < sz[v]){
        swap(u, v); //u la thang co size lon hon
    }
    sz[u] += sz[v];
    parent[v] = u;
    return true;
}
```