Big-O Blue

LECTURE 18

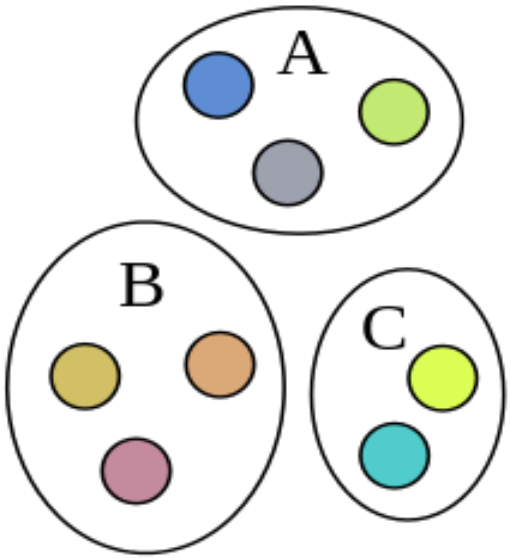
DISJOINT SET UNION

Phạm Nguyễn Sơn Tùng

Email: sontungtn@gmail.com

Định nghĩa DSU

Disjoint set union (DSU) hay còn được gọi với những tên gọi khác như Disjoint set, Union-Find là cấu trúc dữ liệu dùng để tập hợp các phần tử lại với nhau thành một tập lớn.



Các thao tác của DSU

Cấu trúc này bao gồm 2 thao tác cơ bản như sau:

- **Find (u):** Là thao tác sẽ thực hiện và trả về phần tử đại diện của tập hợp.
- **Union (u, v):** Ghép 1 phần tử vào tập hợp lớn hoặc ghép 2 tập hợp lại với nhau. Nếu 2 phần tử đã cùng tập hợp thì thao tác này sẽ không thực hiện.

Độ phức tạp: Mỗi thao tác **Find** và **Union** sẽ có chi phí O(N).

Bài toán minh họa

Cho 1 đồ thị vô hướng gồm N đỉnh được đánh số từ 1 đến N ($1 \leq N \leq 10000$), giữa 2 đỉnh bất kỳ đều có thể nối hoặc không nối với nhau. Ở trạng thái ban đầu tất cả các đỉnh đều không có cạnh nối. Cho bạn danh sách các truy vấn thuộc 2 dạng sau:

- X Y 1: Union(x, y) có ý nghĩa là bạn cần nối 2 đỉnh X và Y.
- X Y 2: Find(x), Find(y) có ý nghĩa là bạn cần cho biết với trạng thái như hiện tại thì 2 đỉnh X và Y có thuộc cùng một thành phần liên thông hay không? Hai đỉnh được coi là thuộc cùng một thành phần liên thông, nếu có đường đi từ đỉnh này đến đỉnh kia, thông qua 1 số đỉnh khác và 2 đỉnh liên tiếp trên đường đi đều có cạnh nối.

Bài toán minh họa

Input:

- Dòng đầu tên là số Q là số truy vấn ($1 \leq Q \leq 50000$).
- Q dòng tiếp theo là X, Y Z với **X, Y** là 2 đỉnh của đồ thị, **Z** là truy vấn đó sẽ thuộc loại gì (1 hoặc 2).

Output:

- Với mỗi yêu cầu dạng X Y 2 (với Z = 2) bạn cần ghi ra số 0 hoặc 1 trên 1 dòng tùy thuộc 2 đỉnh X và Y không thuộc hoặc thuộc cùng một thành phần liên thông.

Bài toán minh họa

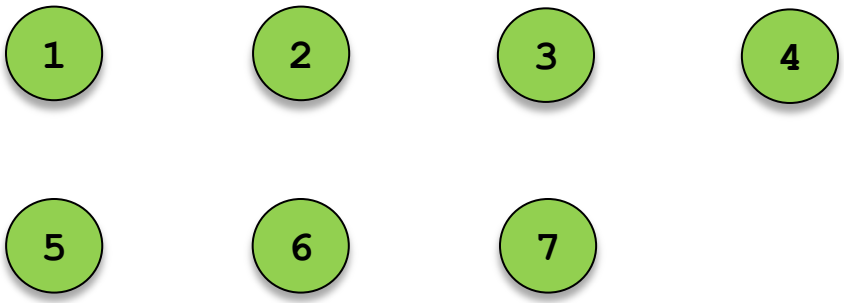
7 1 2 1 2 3 1 5 6 1 2 6 2 6 7 1 7 3 1 6 2 2	0 1

Bước 0: Chuẩn bị dữ liệu

Xác định xem mình có bao nhiêu đỉnh trên đồ thị, nếu chưa xác định được thì có thể chọn hết tối đa số đỉnh. Gán đỉnh cha của mỗi đỉnh là chính nó.

Mảng chứa đỉnh cha parent.

Đỉnh	0	1	2	3	4	5	6	7
Đỉnh cha	0	1	2	3	4	5	6	7

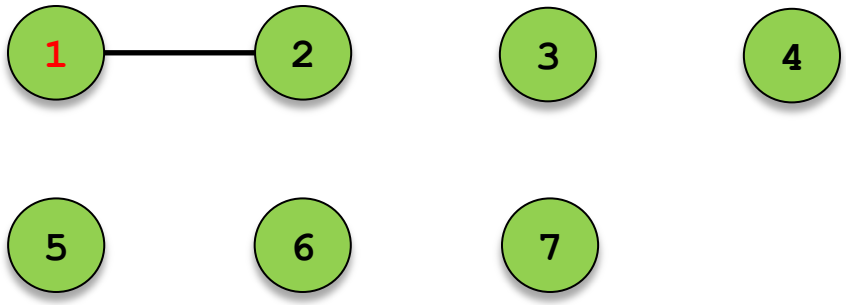


Bước 1: Chạy truy vấn 1

Truy vấn: 1 2 1

Mảng chứa đỉnh cha parent.

Đỉnh	0	1	2	3	4	5	6	7
Đỉnh cha	0	1	1	3	4	5	6	7

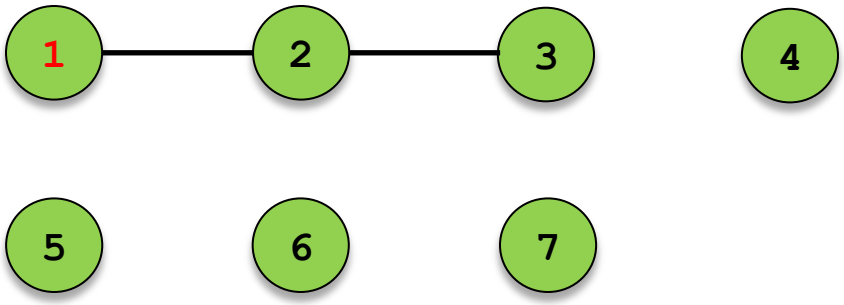


Bước 2: Chạy truy vấn 2

Truy vấn: 2 3 1

Mảng chứa đỉnh cha parent.

Đỉnh	0	1	2	3	4	5	6	7
Đỉnh cha	0	1	1	1	4	5	6	7

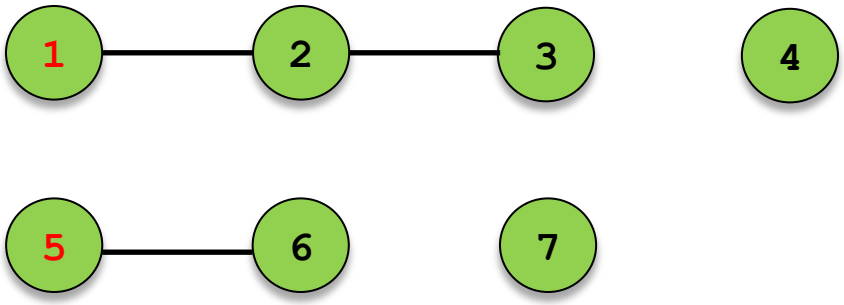


Bước 3: Chạy truy vấn 3

Truy vấn: 5 6 1

Mảng chứa đỉnh cha parent.

Đỉnh	0	1	2	3	4	5	6	7
Đỉnh cha	0	1	1	1	4	5	5	7

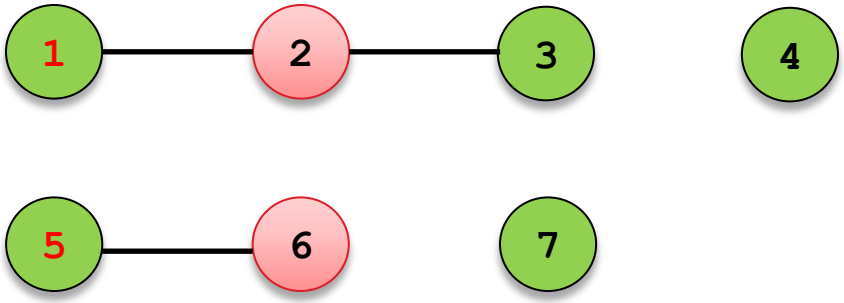


Bước 4: Chạy truy vấn 4

Truy vấn: 2 6 2

Mảng chứa đỉnh cha parent.

Đỉnh	0	1	2	3	4	5	6	7
Đỉnh cha	0	1	1	1	4	5	5	7



Kết quả

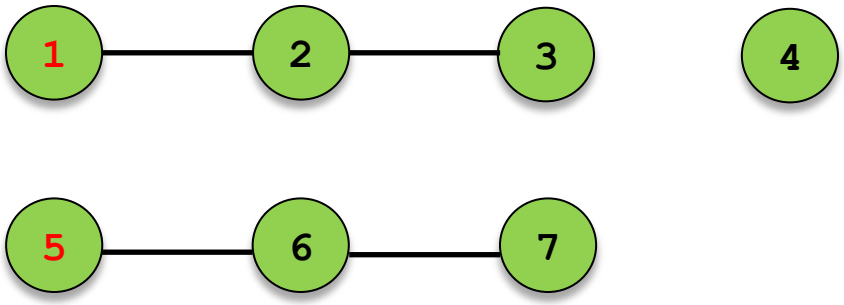
0

Bước 5: Chạy truy vấn 5

Truy vấn: 6 7 1

Mảng chứa đỉnh cha parent.

Đỉnh	0	1	2	3	4	5	6	7
Đỉnh cha	0	1	1	1	4	5	5	5

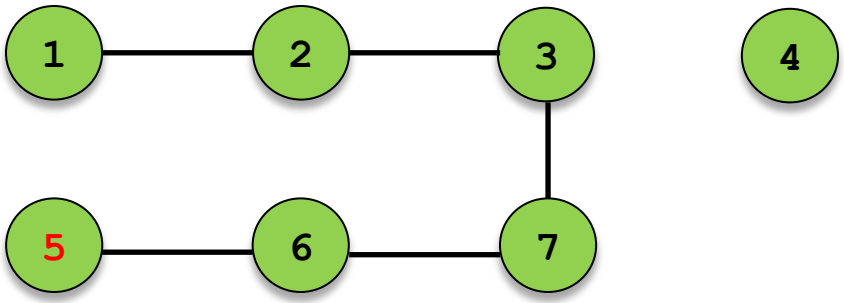


Bước 6: Chạy truy vấn 6

Truy vấn: 7 3 1

Mảng chứa đỉnh cha parent.

Đỉnh	0	1	2	3	4	5	6	7
Đỉnh cha	0	5	1	1	4	5	5	5

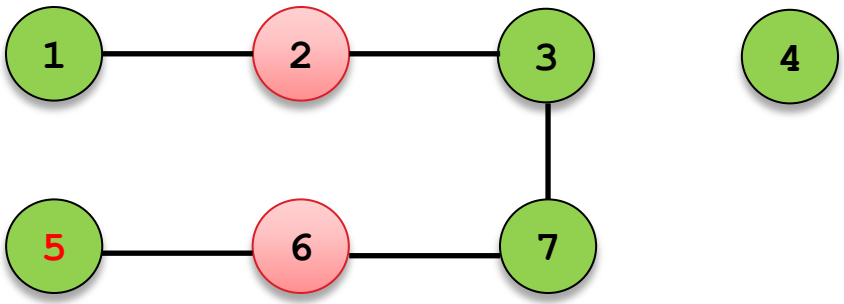


Bước 7: Chạy truy vấn 7

Truy vấn: 6 2 2

Mảng chứa đỉnh cha parent.

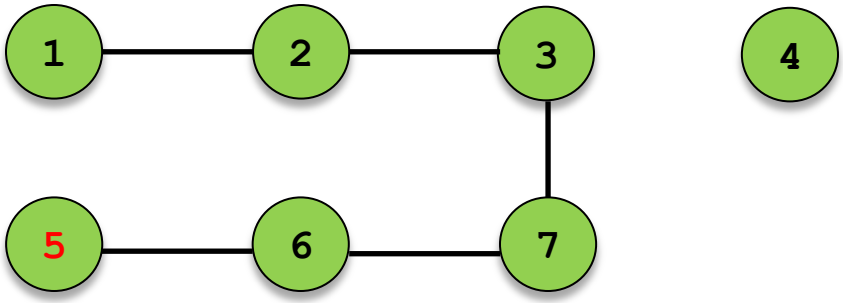
Đỉnh	0	1	2	3	4	5	6	7
Đỉnh cha	0	5	1	1	4	5	5	5



Kết quả

1

Kết quả bài toán



Kết quả

```
0
1
```

Source Code minh họa C++

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

void unionSet(int u, int v) {
    int up = findSet(u);
    int vp = findSet(v);
    parent[vp] = up;
}
```

Source Code minh họa C++

```cpp
int main() {
    int q, x, y, z;
    cin >> q;
    for (int i = 1; i <= MAX; i++)
        parent[i] = i;
    for (int i = 1; i <= q; i++) {
        cin >> x >> y >> z;
        if(z == 1) {
            unionSet(x, y);
        }
        if (z == 2) {
            int parentX = findSet(x);
            int parentY = findSet(y);
            if (parentX == parentY)
                cout << "1" <<endl;
            else
                cout << "0" << endl;
        }
    }
    return 0;
}
```

Source Code minh họa python

```
def findSet(u):
    while u != parent[u]:
        u = parent[u]
    return u


def unionSet(u, v):
    up = findSet(u)
    vp = findSet(v)
    parent[vp] = up
```

Source Code minh họa python

```
MAX = 100
q = int(input())
parent = [i for i in range(MAX + 1)]
for i in range(q):
    x, y, z = map(int, input().split())
    if z == 1:
        unionSet(x, y)
    else:
        parentX = findSet(x)
        parentY = findSet(y)
        if parentX == parentY:
            print("1")
        else:
            print("0")
```

Source Code minh họa Java

```java
public static int findSet(ArrayList <Integer> parent, int u) {
    while (u != parent.get(u)) {
        u = parent.get(u);
    }
    return u;
}

public static void unionSet(ArrayList<Integer> parent, int u, int v)
{
    int up = findSet(parent, u);
    int vp = findSet(parent, v);
    parent.set(vp,  parent.get(up));
}
```

Source Code minh họa Java

```java
static ArrayList<Integer> parent = new ArrayList<Integer>();
public static void main (String[] args) {
    for (int i = 0; i <= MAX; i++)
        parent.add(i);
    Scanner sc = new Scanner(System.in);
    int q = sc.nextInt();
    for (int i = 0; i < q; i++) {
        int x = sc.nextInt(), y = sc.nextInt(), z = sc.nextInt();
        if (z == 1)
            unionSet(x, y);
        else {
            int parentX = findSet(x);
            int parentY = findSet(y);
            if (parentX == parentY)
                System.out.println("1");
            else
                System.out.println("0");
        }
    }
}
```

Hỏi đáp