Big-O Blue

LECTURE 16

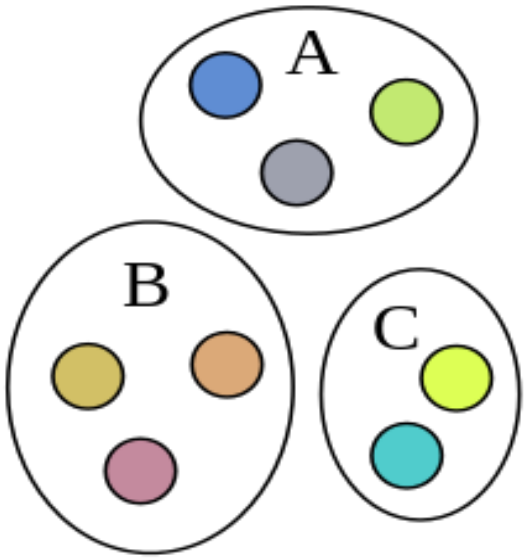
DISJOINT SET UNION

Phạm Nguyễn Sơn Tùng

Email: sontungtn@gmail.com

Disjoint Set Union

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 hợp các phần tử lại với nhau thành những tập hợp lớn.



Các thao tác trên DSU

Cấu trúc này cho phép thực hiện 3 thao tác cơ bản:

- **makeSet( ):** Tạo ra tập hợp cho mỗi phần tử ban đầu.
- **findSet(u):** Tìm phần tử đại diện của tập hợp chứa u.
- **unionSet(u, v):** Hợp tập hợp chứa u và tập hợp chứa v thành một tập hợp lớn. Nếu 2 phần tử u và v thuộc cùng một tập hợp thì thao tác này sẽ không thực hiện.

Time complexity: Mỗi thao tác sẽ có độ phức tạp O(N).

Bài toán minh họa

Cho đồ thị vô hướng gồm N đỉnh được đánh số từ 1 đến N ($1 \le N \le 10.000$). Ở trạng thái ban đầu tất cả các đỉnh đều không có cạnh nối. Để kết nối các đỉnh trong đồ thị với nhau bạn cần một số truy vấn. Cho bạn danh sách các truy vấn thuộc một trong 2 loại như sau:

- u v 1: Union(u, v) yêu cầu nối 2 đỉnh u và v lại với nhau (loại 1).
- u v 2: Find(u), Find(v) Kiểm tra liệu hai đỉnh u, v có kết nối (tức thuộc cùng một thành phần liên thông) hay không (loại 2).

Lưu ý: *2 đỉnh được xem là thuộc cùng một thành phần liên thông nếu tồn tại ít nhất một đường đi từ đỉnh này đến đỉnh kia bằng cạnh nối trực tiếp hoặc thông qua 1 số đỉnh khác.*

Bài toán minh họa

Input:

- Dòng đầu tiên là số nguyên Q là số truy vấn ($1 \le Q \le 50.000$).
- Q dòng tiếp theo là u, v, q với **u, v** là 2 đỉnh của đồ thị, **q** là loại truy vấn (loại 1 hoặc loại 2).

Output:

- Với mỗi yêu cầu dạng u v 2 (với q = 2) bạn cần in ra trên một dòng là "YES" hoặc "NO".
    - YES: Tại thời điểm hiện tại 2 đỉnh u và v có kết nối với nhau.
    - NO: Tại thời điểm hiện tại 2 đỉnh u và v không có kết nối với nhau.

Bài toán minh họa

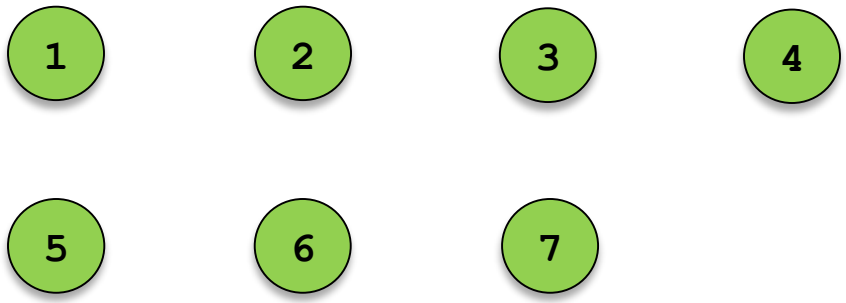
Input	Output
9 1 2 1 2 3 1 4 5 1 5 6 1 2 6 2 6 7 1 7 3 1 6 2 2 7 1 2	NO YES YES

Bước 0: Chuẩn bị dữ liệu

Xác định xem đồ thị có bao nhiêu đỉnh, nếu **không** xác định được thì có thể chọn hết tối đa số đỉnh trong bài toán. Dùng thao tác **makeSet( )** xem mỗi đỉnh thuộc một tập hợp riêng biệt, 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

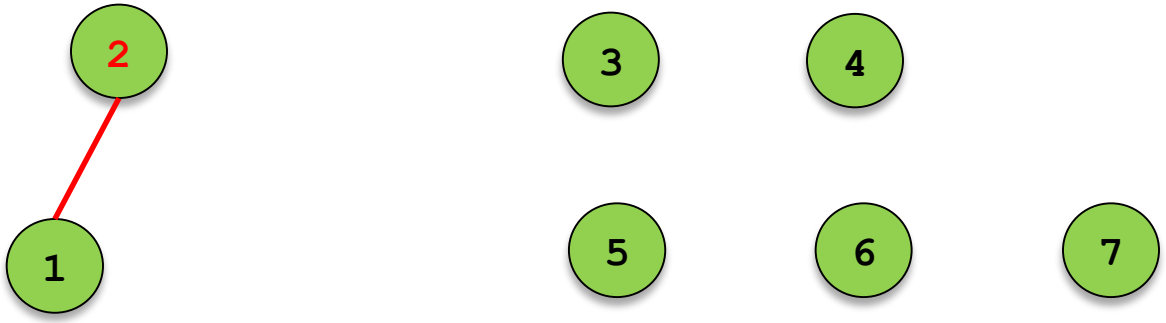


Bước 1: Chạy truy vấn 1

Truy vấn: 1 2 1

Mảng chứa đỉnh cha parent.

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

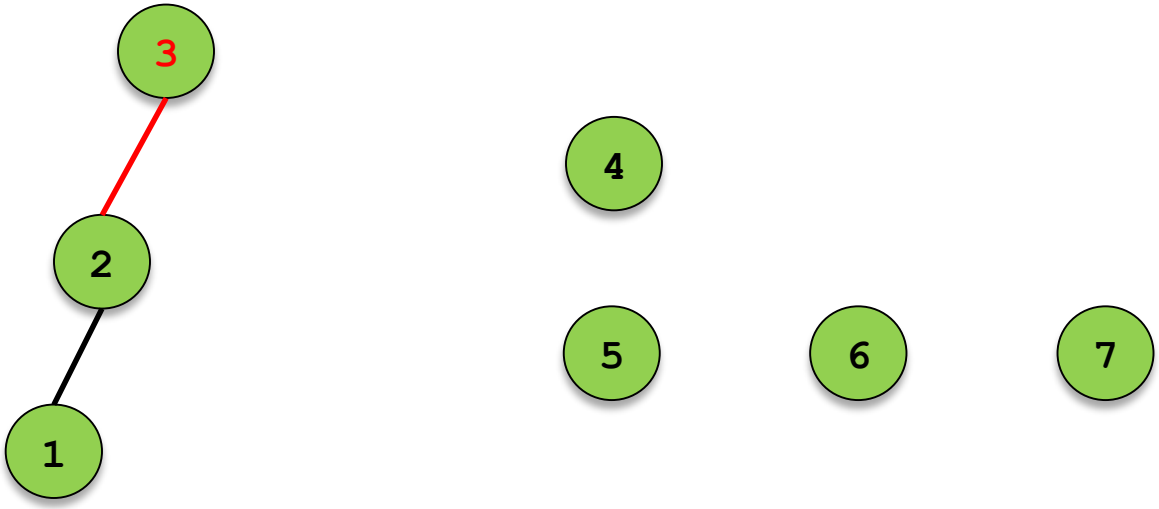


Bước 2: Chạy truy vấn 2

Truy vấn: 2 3 1

Mảng chứa đỉnh cha parent.

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

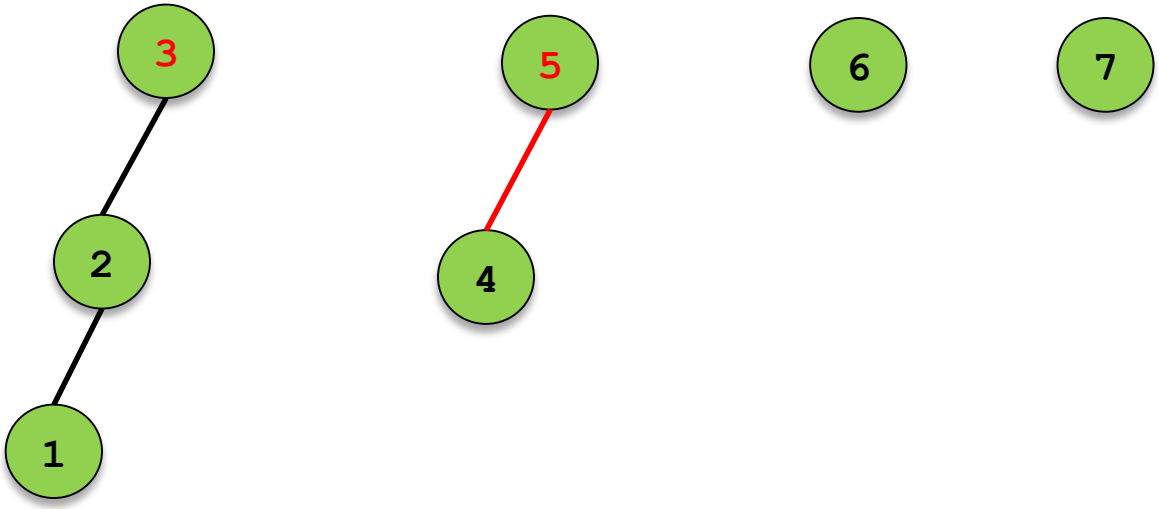


Bước 3: Chạy truy vấn 3

Truy vấn: 4 5 1

Mảng chứa đỉnh cha parent.

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

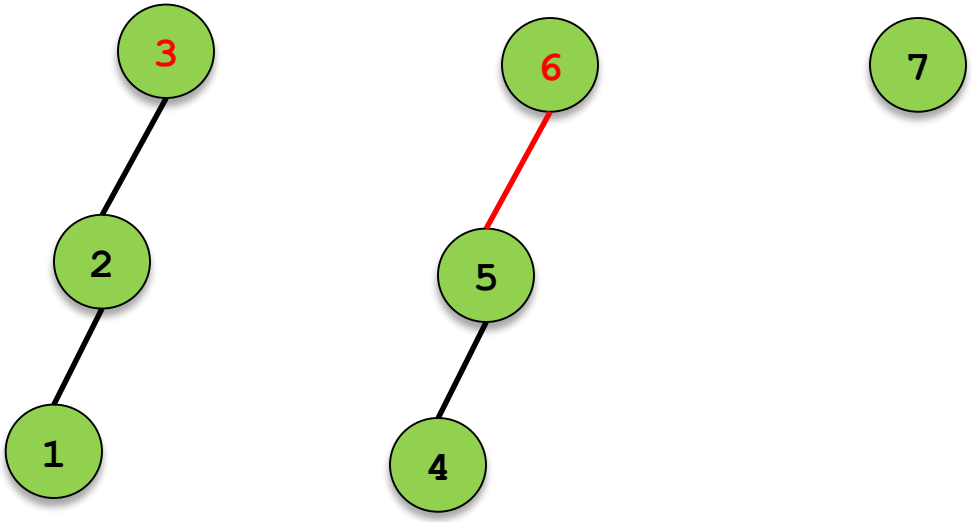


Bước 4: Chạy truy vấn 4

Truy vấn: 5 6 1

Mảng chứa đỉnh cha parent.

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

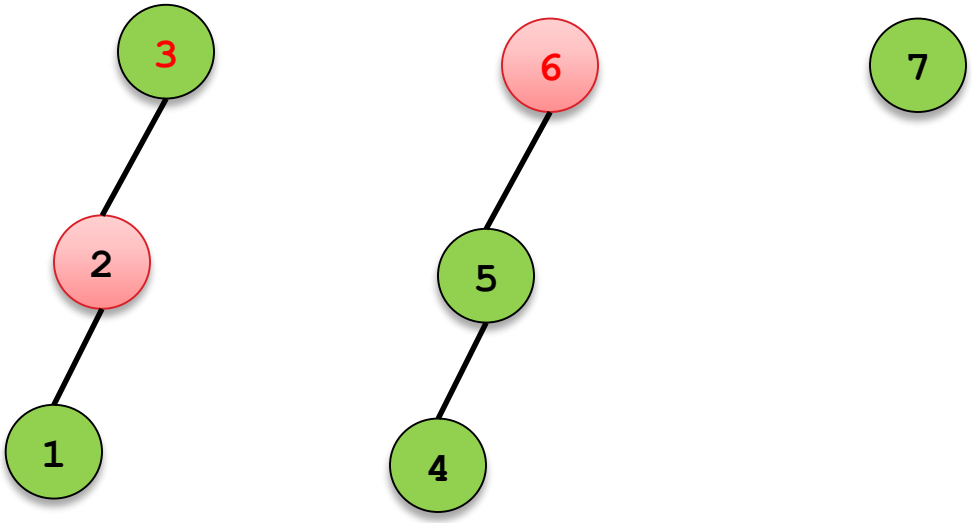


Bước 5: Chạy truy vấn 5

Truy vấn: 2 6 2

Mảng chứa đỉnh cha parent.

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



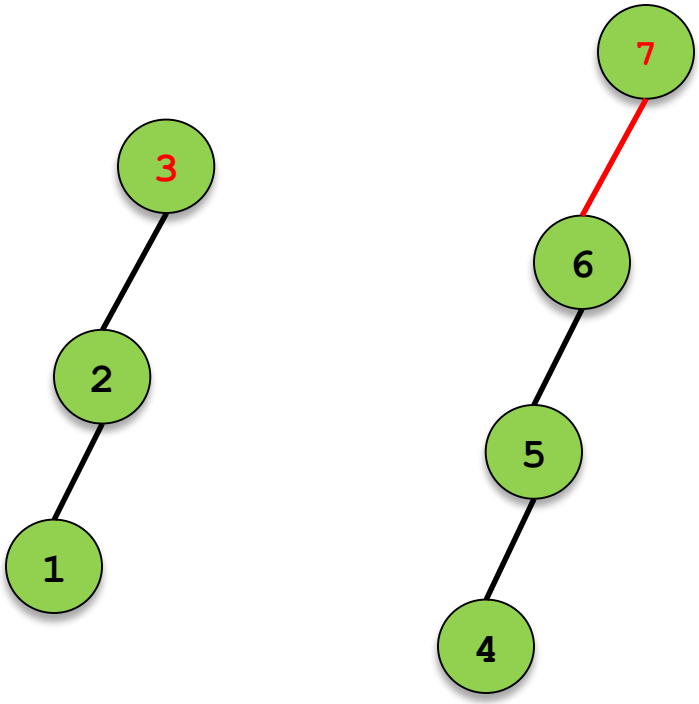
NO

Bước 6: Chạy truy vấn 6

Truy vấn: 6 7 1

Mảng chứa đỉnh cha parent.

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

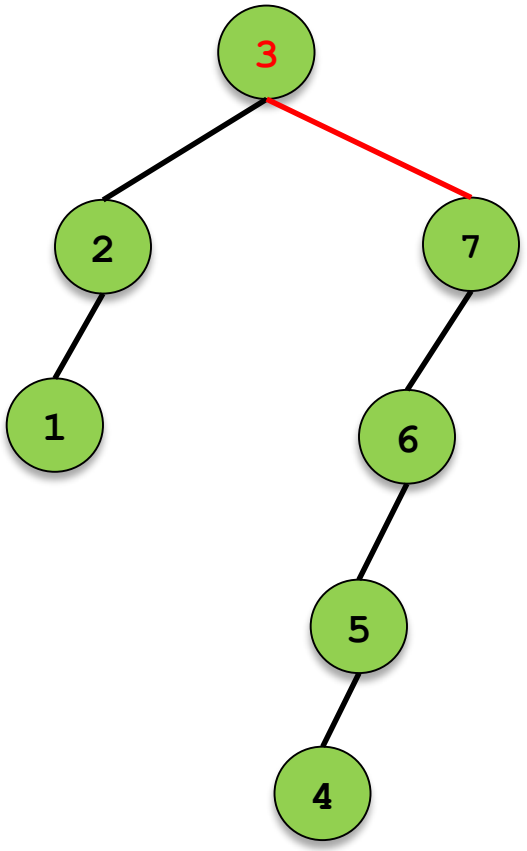


Bước 7: Chạy truy vấn 7

Truy vấn: 7 3 1

Mảng chứa đỉnh cha parent.

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

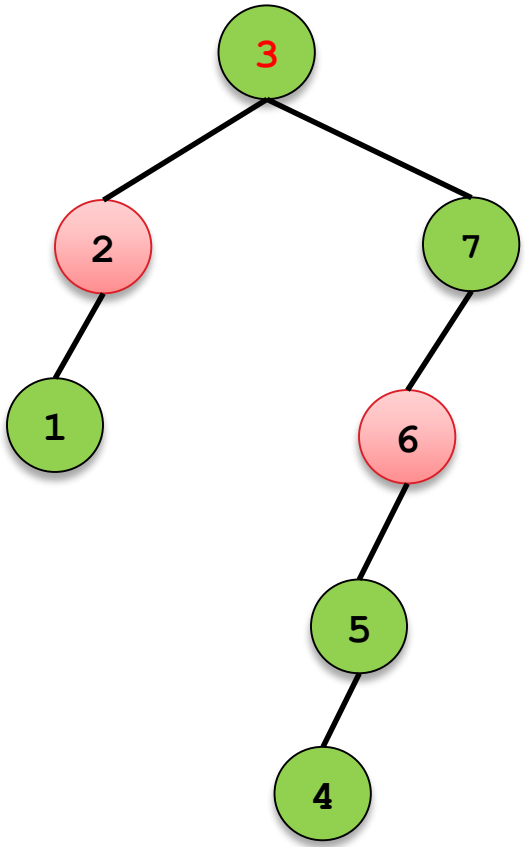


Bước 8: Chạy truy vấn 8

Truy vấn: 6 2 2

Mảng chứa đỉnh cha parent.

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



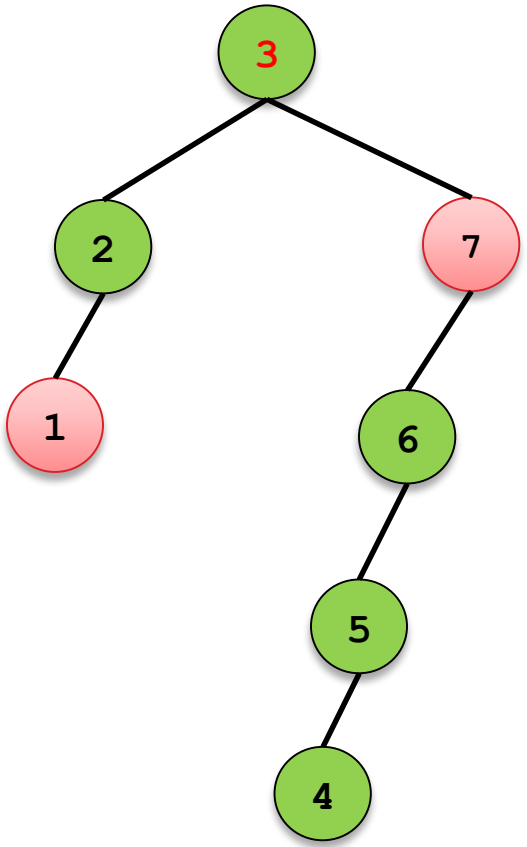
YES

Bước 9: Chạy truy vấn 9

Truy vấn: 7 1 2

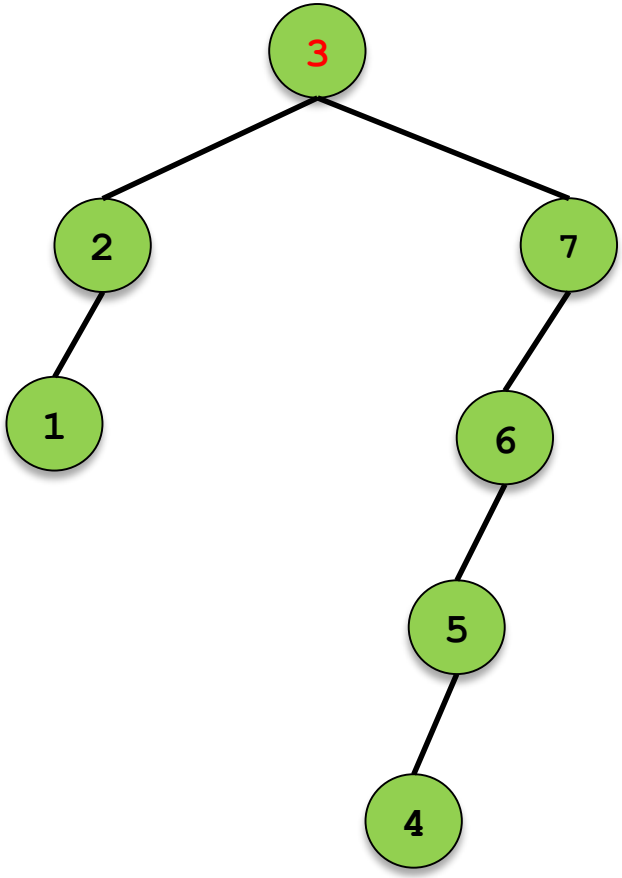
Mảng chứa đỉnh cha parent.

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



YES

Kết quả bài toán




```
NO
YES
YES
```

Source Code DSU

Khai báo biến toàn cục và hàm khởi tạo tập hợp.

```cpp
#include <iostream>
using namespace std;
#define MAX 20
int parent[MAX + 5];

void makeSet()
{
    for (int i = 1; i <= MAX; i++)
        parent[i] = i;
}
```

Source Code DSU

```
10. int findSet(int u)
11. {
12.     while (u != parent[u])
13.         u = parent[u];
14.     return u;
15. }
16. void unionSet(int u, int v)
17. {
18.     int up = findSet(u);
19.     int vp = findSet(v);
20.     parent[up] = vp;
21. }
```

Source Code DSU

```
22. int main()
23. {
24.     int Q, u, v, q;
25.     cin >> Q;
26.     makeSet();
27.     for (int i = 0; i < Q; i++)
28.     {
29.         cin >> u >> v >> q;
30.         if (q == 1)
31.             unionSet(u, v);
32.         if (q == 2) {
33.             int parentU = findSet(u);
34.             int parentV = findSet(v);
35.             if (parentU == parentV)
36.                 cout << "YES" << endl;
37.             else
38.                 cout << "NO" << endl;
39.         }
40.     }
41.     return 0;
42. }
```

Source Code DSU

Khai báo biến toàn cục và hàm khởi tạo tập hợp.

```python
MAX = 20
parent = []

def makeSet():
    global parent
    parent = [i for i in range(MAX + 5)]
```

Source Code DSU

```
6.  def findSet(u):
7.      while u != parent[u]:
8.          u = parent[u]
9.      return u

10. def unionSet(u, v):
11.     up = findSet(u)
12.     vp = findSet(v)
13.     parent[up] = vp
```

Source Code DSU

```
14. if __name__ == '__main__':
15.     Q = int(input())
16.     makeSet()
17.     for i in range(Q):
18.         u, v, q = map(int, input().split())
19.         if q == 1:
20.             unionSet(u, v)
21.         if q == 2:
22.             parentU = findSet(u)
23.             parentV = findSet(v)
24.             if (parentU == parentV):
25.                 print("YES")
26.             else:
27.                 print("YES")
```

Source Code DSU

Khai báo biến toàn cục mà hàm khởi tạo tập hợp.

```java
import java.util.Scanner;

// khai báo biến trong class Main
private static final int MAX = 20;
private static int[] parent = new int[MAX + 5];

private static void makeSet() {
    for (int i = 1; i <= MAX; i++) {
        parent[i] = i;
    }
}
```

Source Code DSU

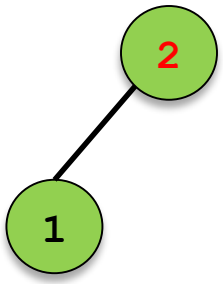
```
10. private static int findSet(int u) {
11.     while (u != parent[u]) {
12.         u = parent[u];
13.     }
14.     return u;
15. }

16. private static void unionSet(int u, int v) {
17.     int up = findSet(u);
18.     int vp = findSet(v);
19.     parent[up] = vp;
20. }
```

Source Code DSU

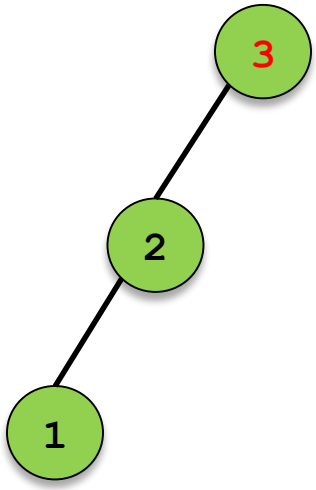
```
21. public static void main (String[] args) {
22.     Scanner sc = new Scanner(System.in);
23.     int Q = sc.nextInt();
24.     makeSet();
25.     for (int i = 0; i < Q; i++) {
26.         int u = sc.nextInt();
27.         int v = sc.nextInt();
28.         int q = sc.nextInt();
29.         if (q == 1)
30.             unionSet(u, v);
31.         if (q == 2) {
32.             int parentU = findSet(u);
33.             int parentV = findSet(v);
34.             if (parentU == parentV)
35.                 System.out.println("YES");
36.             else
37.                 System.out.println("NO");
38.         }
39.     }
40. }
```

Trường hợp đặc biệt

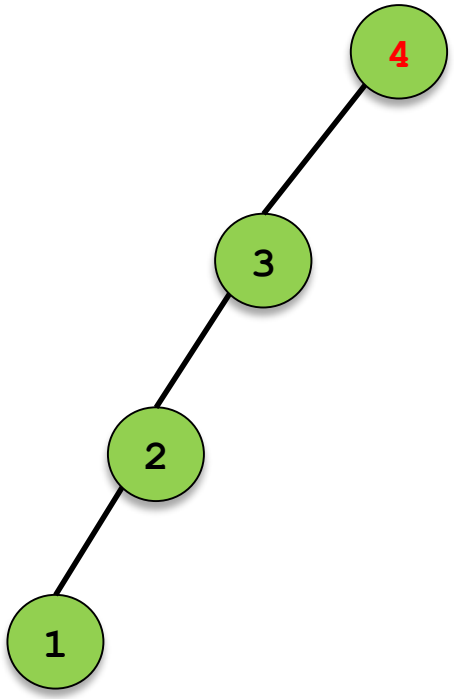
Truy vấn: 1 2 1



Truy vấn: 2 3 1



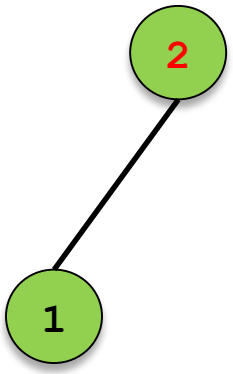
Truy vấn: 3 4 1



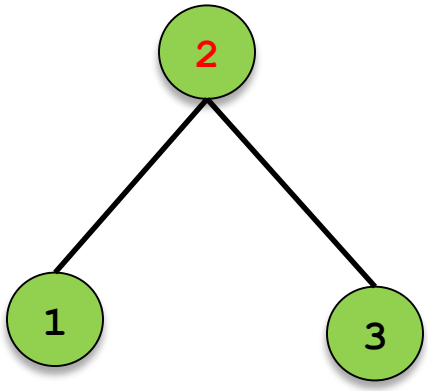
Union By Rank and Path Compression

Union By Rank and Path Compression: Là phương pháp Union các node lại theo rank của node và nén đường đi để việc truy vết đỉnh cha được nhanh hơn.

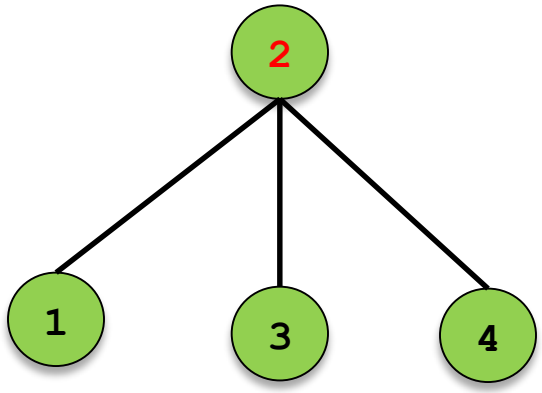
Truy vấn: 1 2 1



Truy vấn: 2 3 1



Truy vấn: 3 4 1



Time Complexity: Mỗi thao tác sẽ có độ phức tạp O(log(N)).

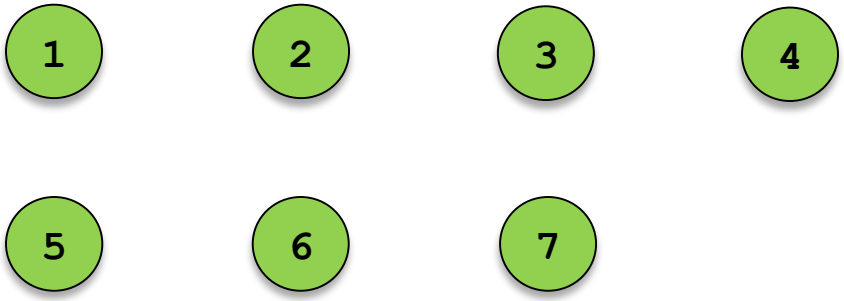
Bước 0: Chuẩn bị dữ liệu

Mảng chứa đỉnh cha parent.

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

Mảng ranks.

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



Bước 1: Chạy truy vấn 1

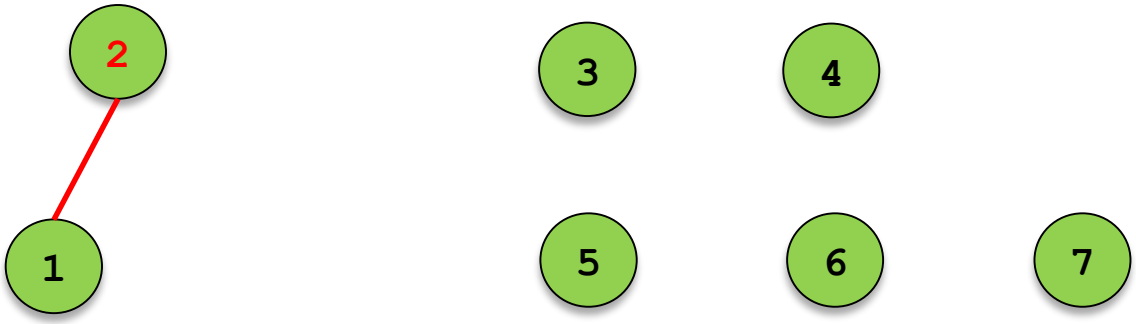
Truy vấn: 1 2 1

Mảng chứa đỉnh cha parent.

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

Mảng ranks.

Đỉnh	0	1	**2**	3	4	5	6	7
rank	0	0	**0 → 1**	0	0	0	0	0



Bước 2: Chạy truy vấn 2

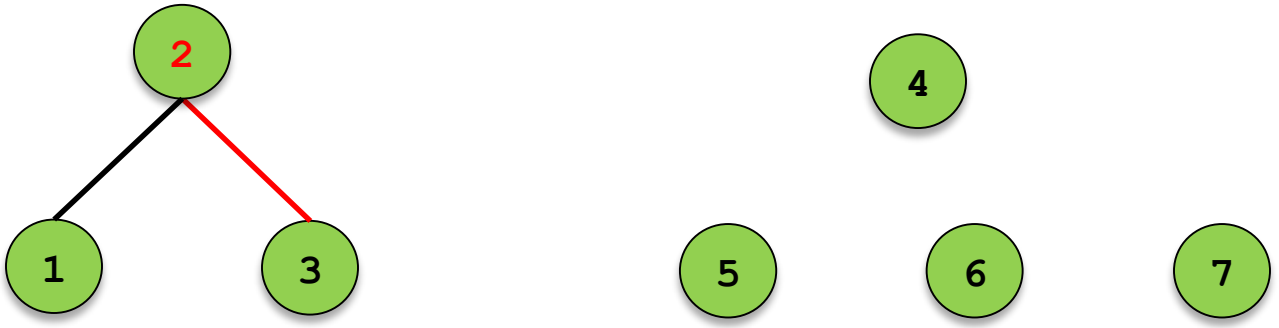
Truy vấn: 2 3 1

Mảng chứa đỉnh cha parent.

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

Mảng ranks.

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



Bước 3: Chạy truy vấn 3

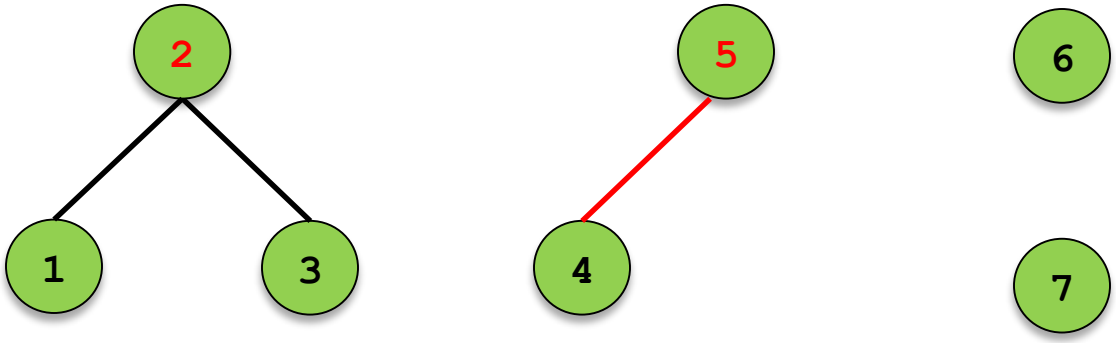
Truy vấn: 4 5 1

Mảng chứa đỉnh cha parent.

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

Mảng ranks.

Đỉnh	0	1	2	3	4	5	6	7
rank	0	0	1	0	0	0 → 1	0	0



Bước 4: Chạy truy vấn 4

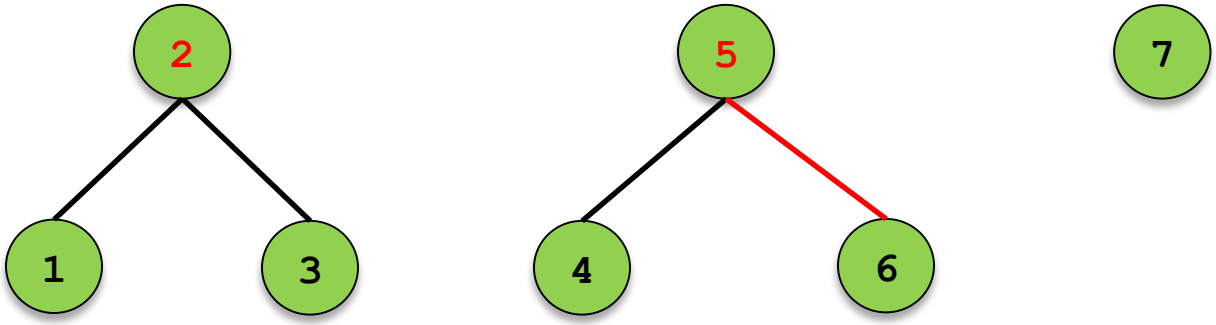
Truy vấn: 5 6 1

Mảng chứa đỉnh cha parent.

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

Mảng ranks.

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



Bước 5: Chạy truy vấn 5

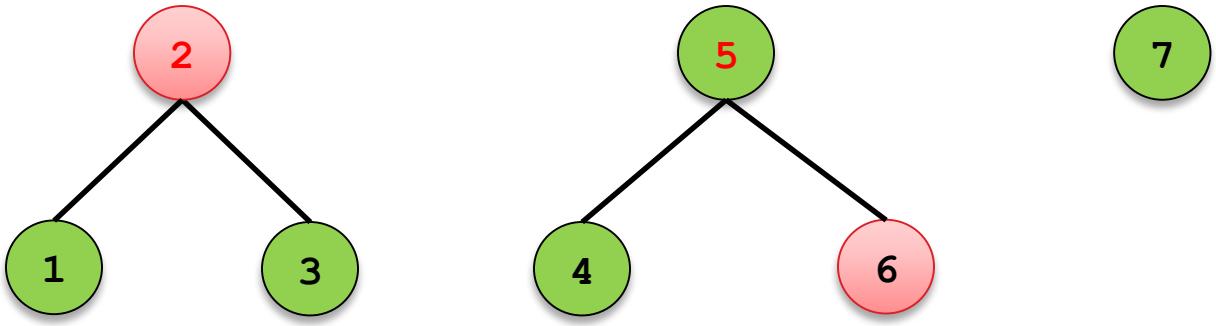
Truy vấn: 2 6 2

Mảng chứa đỉnh cha parent.

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

Mảng ranks.

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



NO

Bước 6: Chạy truy vấn 6

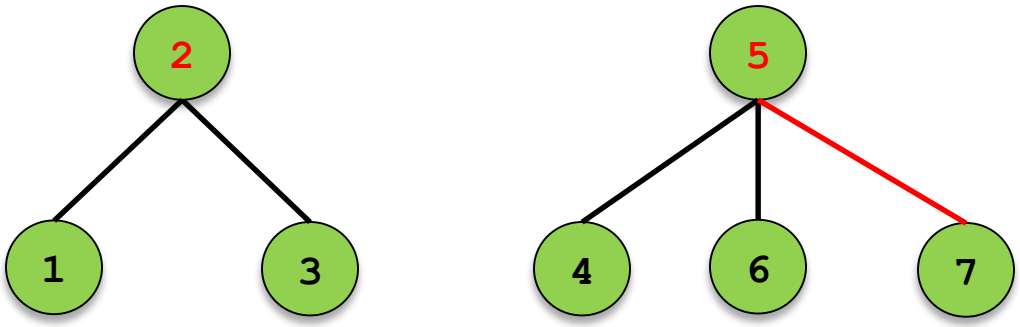
Truy vấn: 6 7 1

Mảng chứa đỉnh cha parent.

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

Mảng ranks.

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



Bước 7: Chạy truy vấn 7

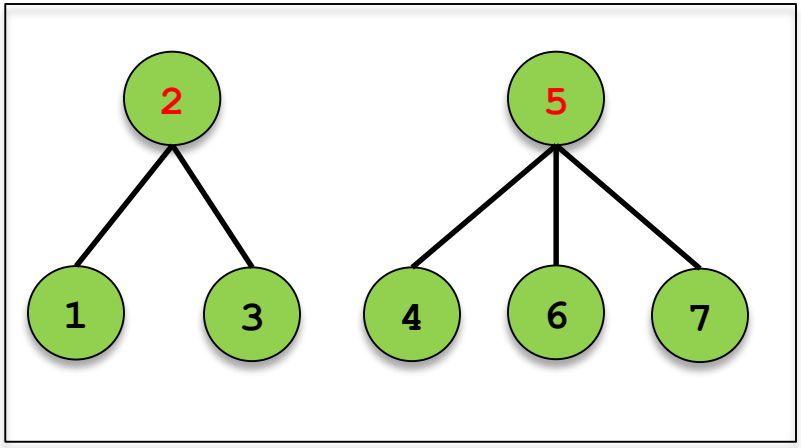
Truy vấn: 7 3 1

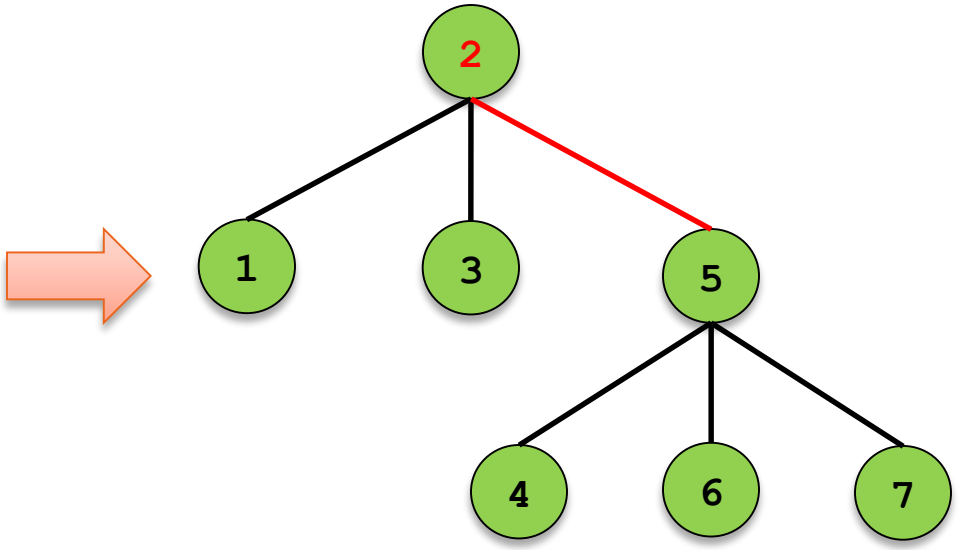
Mảng chứa đỉnh cha parent.

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

Mảng ranks.

Đỉnh	0	1	**2**	3	4	5	6	7
rank	0	0	**1 → 2**	0	0	1	0	0





Bước 8: Chạy truy vấn 8

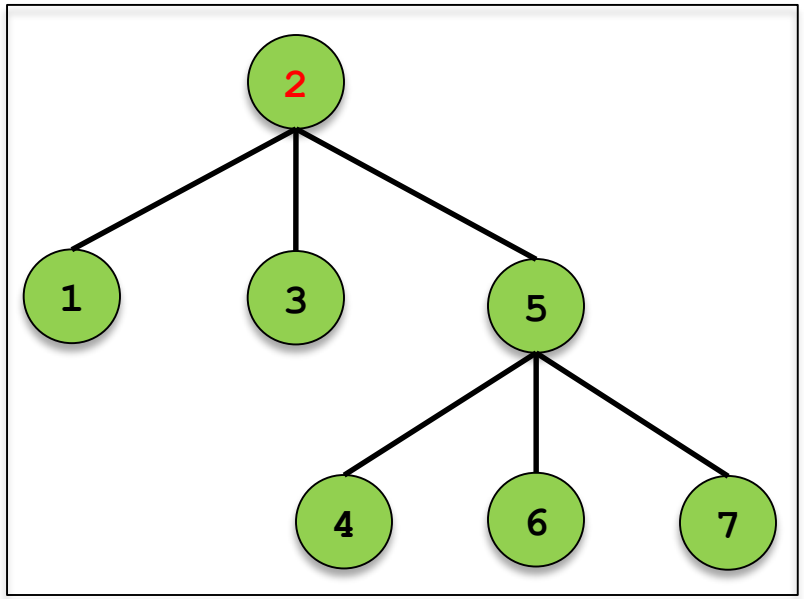
Truy vấn: 6 2 2

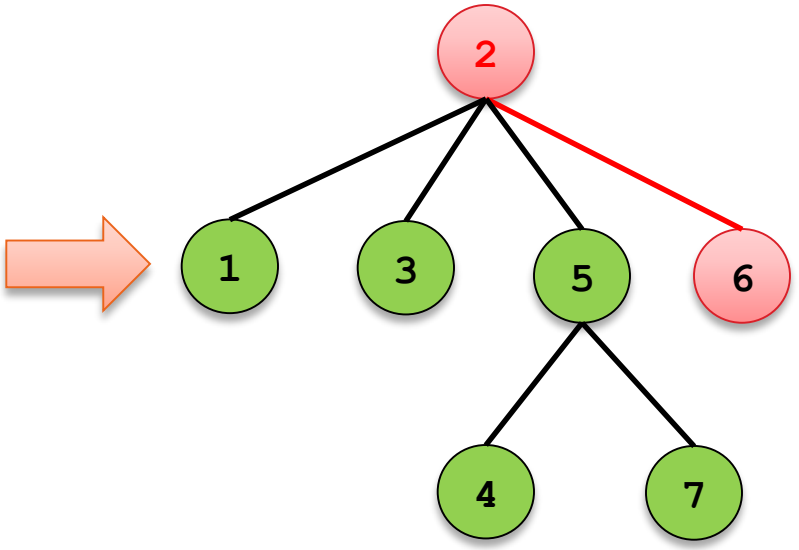
Mảng chứa đỉnh cha parent.

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

Mảng ranks.

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





YES

Bước 9: Chạy truy vấn 9

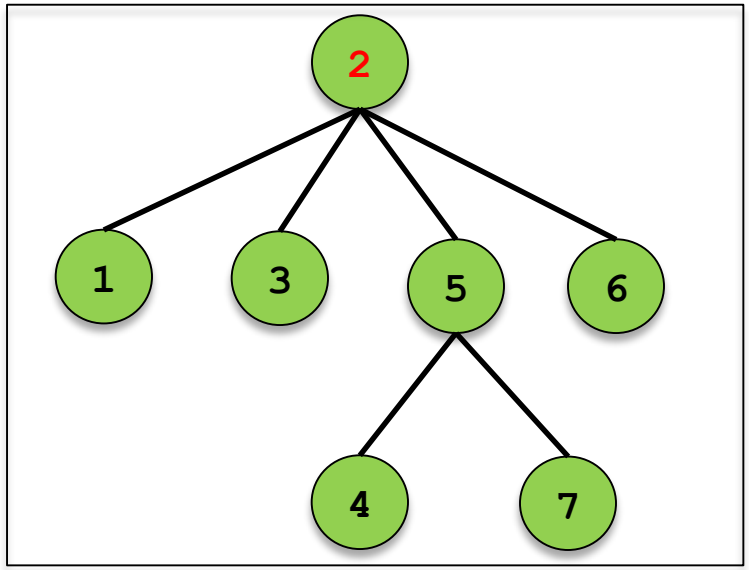
Truy vấn: 7 1 2

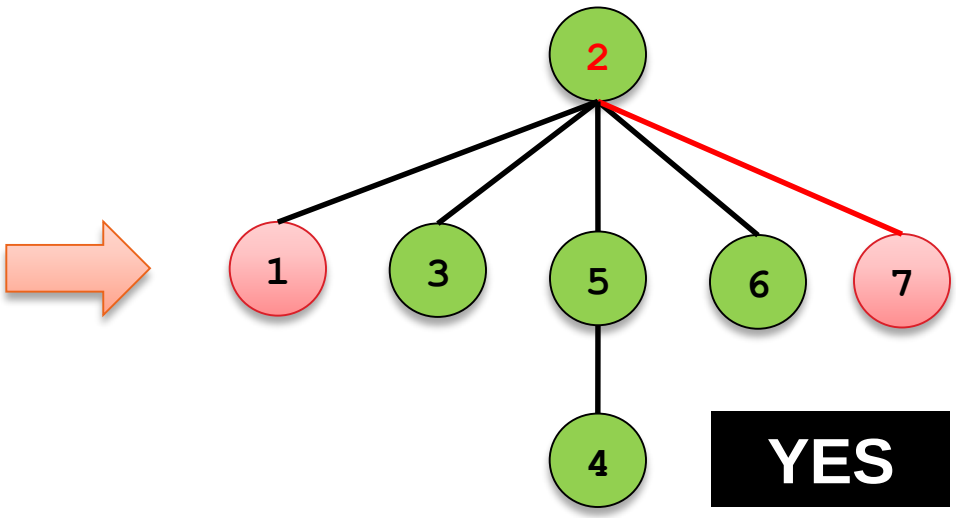
Mảng chứa đỉnh cha parent.

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

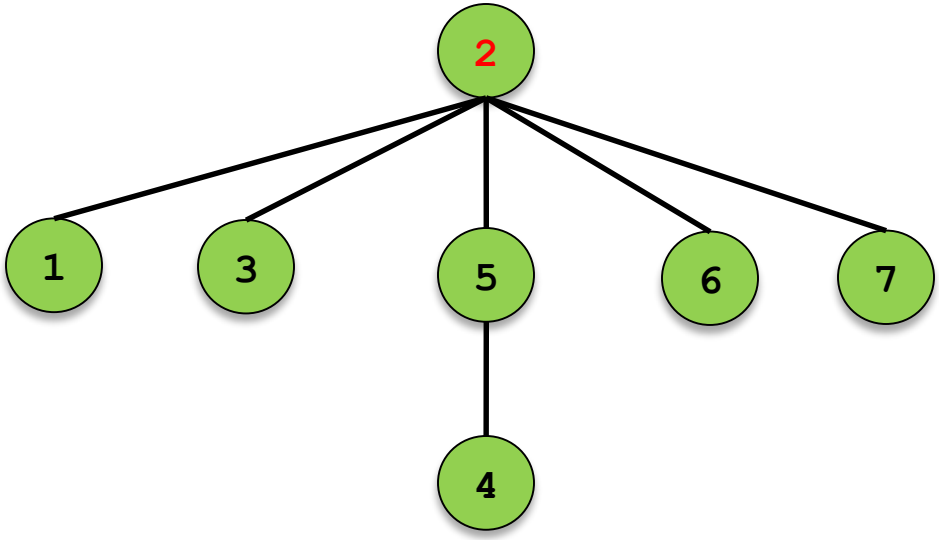
Mảng ranks.

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





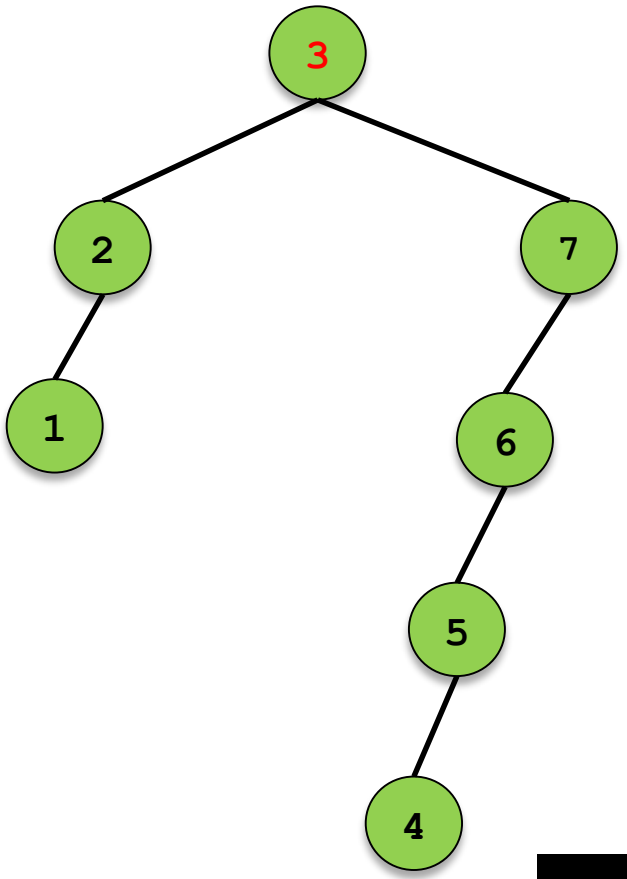
Kết quả bài toán



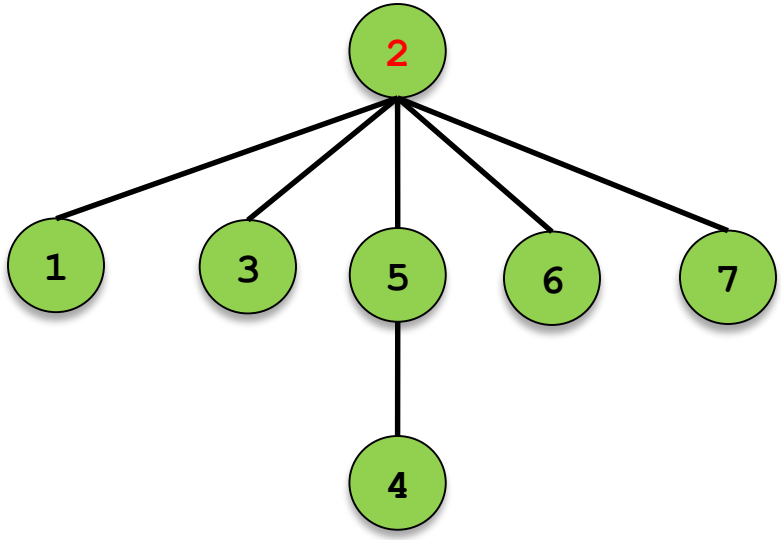

```
NO
YES
YES
```

So sánh 2 phương pháp

Disjoint Set Union cơ bản



Union By Rank & Path Compression



NO
YES
YES

Source Code DSU Improvement

Khai báo biến toàn cục mà hàm khởi tạo tập hợp.

```cpp
#include <iostream>
using namespace std;
#define MAX 20
int parent[MAX + 5];
int ranks[MAX + 5];

void makeSet()
{
    for (int i = 1; i <= MAX; i++)
    {
        parent[i] = i;
        ranks[i] = 0;
    }
}
```

Source Code DSU Improvement

Path Compression.

```
15. int findSet(int u)
16. {
17.     if (parent[u] != u)
18.         parent[u] = findSet(parent[u]);
19.     return parent[u];
20. }
```

Source Code DSU Improvement

Union By Rank.

```
21. void unionSet(int u, int v)
22. {
23.     int up = findSet(u);
24.     int vp = findSet(v);
25.     if (up == vp)
26.         return;
27.     if (ranks[up] > ranks[vp])
28.         parent[vp] = up;
29.     else if (ranks[up] < ranks[vp])
30.         parent[up] = vp;
31.     else
32.     {
33.         parent[up] = vp;
34.         ranks[vp]++;
35.     }
36. }
```

Source Code DSU Improvement

Big-O Blue


```
37.  int main()
38.  {
39.      int Q, u, v, q;
40.      cin >> Q;
41.      makeSet();
42.      for (int i = 0; i < Q; i++)
43.      {
44.          cin >> u >> v >> q;
45.          if (q == 1)
46.              unionSet(u, v);
47.          if (q == 2)
48.          {
49.              int parentU = findSet(u);
50.              int parentV = findSet(v);
51.              if (parentU == parentV)
52.                  cout << "YES" << endl;
53.              else
54.                  cout << "NO" << endl;
55.          }
56.      }
57.      return 0;
58.  }
```

Source Code DSU Improvement

Khai báo biến toàn cục mà hàm khởi tạo tập hợp.

```
1.  MAX = 20
2.  parent = []
3.  ranks = []

4.  def makeSet():
5.      global parent, ranks
6.      parent = [i for i in range(MAX + 5)]
7.      ranks = [0 for i in range(MAX + 5)]
```

Source Code DSU Improvement

Path Compression.

```
8.  def findSet(u):
9.      if parent[u] != u:
10.         parent[u] = findSet(parent[u])
11.     return parent[u]
```

Source Code DSU Improvement

Union By Rank.

```
12. def unionSet(u, v):
13.     up = findSet(u)
14.     vp = findSet(v)
15.     if up == vp:
16.         return
17.     if ranks[up] > ranks[vp]:
18.         parent[vp] = up
19.     elif ranks[up] < ranks[vp]:
20.         parent[up] = vp
21.     else:
22.         parent[up] = vp
23.         ranks[vp] += 1
```

Source Code DSU Improvement

```
24. if __name__ == '__main__':
25.     Q = int(input())
26.     makeSet()
27.     for i in range(Q):
28.         u, v, q = map(int, input().split())
29.         if q == 1:
30.             unionSet(u, v)
31.         if q == 2:
32.             parentU = findSet(u)
33.             parentV = findSet(v)
34.             if (parentU == parentV):
35.                 print("YES")
36.             else:
37.                 print("NO")
```

Source Code DSU Improvement

Khai báo biến toàn cục mà hàm khởi tạo tập hợp.


```java
import java.util.Scanner;

// khai báo biến trong class Main
private static final int MAX = 20;
private static int[] parent = new int[MAX + 5];
private static int[] ranks = new int[MAX + 5];

private static void makeSet() {
    for (int i = 1; i <= MAX; i++) {
        parent[i] = i;
        ranks[i] = 0;
    }
}
```

Source Code DSU Improvement

Path Compression.

```
12. private static int findSet(int u) {
13.     if (parent[u] != u) {
14.         parent[u] = findSet(parent[u]);
15.     }
16.     return parent[u];
17. }
```

Source Code DSU Improvement

Union By Rank.

```
18. private static void unionSet(int u, int v) {
19.     int up = findSet(u);
20.     int vp = findSet(v);
21.     if (up == vp)
22.         return;
23.     if (ranks[up] > ranks[vp])
24.         parent[vp] = up;
25.     else if (ranks[up] < ranks[vp])
26.         parent[up] = vp;
27.     else {
28.         parent[up] = vp;
29.         ranks[vp]++;
30.     }
31. }
```

Source Code DSU Improvement

```
32. public static void main (String[] args) {
33.     Scanner sc = new Scanner(System.in);
34.     int Q = sc.nextInt();
35.     makeSet();
36.     for (int i = 0; i < Q; i++) {
37.         int u = sc.nextInt();
38.         int v = sc.nextInt();
39.         int q = sc.nextInt();
40.         if (q == 1)
41.             unionSet(u, v);
42.         if (q == 2) {
43.             int parentU = findSet(u);
44.             int parentV = findSet(v);
45.             if (parentU == parentV)
46.                 System.out.println("YES");
47.             else
48.                 System.out.println("NO");
49.         }
50.     }
51. }
```

Hỏi đáp