Java

```java
class UnionFind {
    private Vector<Integer> p, rank;

    public UnionFind(int N) {
        p = new Vector<Integer>(N);
        rank = new Vector<Integer>(N);
        for (int i = 0; i < N; i++) {
            p.add(i);
            rank.add(0);
        }
    }

    public int findSet(int i) {
        if (p.get(i) == i) return i;
        else {
            int ret = findSet(p.get(i));
            p.set(i, ret);
            return ret;
        }
    }

    public Boolean isSameSet(int i, int j) {
        return findSet(i) == findSet(j);
    }

    public void unionSet(int i, int j) {
        if (!isSameSet(i, j)) {
            int x = findSet(i), y = findSet(j);
            // rank is used to keep the tree short
            if (rank.get(x) > rank.get(y)) {
                p.set(y, x);
            }
            else                              {
                p.set(x, y);
                if (rank.get(x) == rank.get(y)) rank.set(y, rank.get(y) + 1);
            }
        }
    }
}
```

C++

```cpp
class UnionFind {
private:
    vector<int> p, rank;

public:
    UnionFind(int N) {
        rank.assign(N, 0);
        p.assign(N, 0);
        for (int i = 0; i < N; i++)
            p[i] = i;
    }
    int findSet(int i) {
        return (p[i] == i) ? i : (p[i] = findSet(p[i]));
    }
    bool isSameSet(int i, int j) {
        return findSet(i) == findSet(j);
    }
    void unionSet(int i, int j) {
        if (!isSameSet(i, j)) {
            int x = findSet(i), y = findSet(j);
            // rank is used to keep the tree short
            if (rank[x] > rank[y]) {
                p[y] = x;
            }
            else  {
                p[x] = y;
                if (rank[x] == rank[y])
                    rank[y]++;
            }
        }
    }
}
```