class DisjointSet {

private:

    vector<int> parent;  // 用于存储每个节点的父节点

    vector<int> rank;    // 用于存储每个集合的秩（树的高度）

    int count;           // 记录当前的集合数量

public:

    // 构造函数，初始化 n 个元素

    DisjointSet(int n) : parent(n), rank(n, 0), count(n) {

        for (int i = 0; i < n; i++) {

            parent[i] = i;  // 每个元素的父节点初始化为自己

        }

    }

    // 查找操作，带路径压缩

    int find(int x) {

        if (parent[x] != x) {

            parent[x] = find(parent[x]);  // 递归查找根节点，并路径压缩

        }

        return parent[x];

    }

    // 合并两个集合，按秩合并

    void unionSets(int x, int y) {

        int rootx = find(x);  // 查找 x 的根节点

        int rooty = find(y);  // 查找 y 的根节点

        if (rootx != rooty) {  // 如果根节点不同，说明两个集合不相同

            // 按秩合并

            if (rank[rootx] > rank[rooty]) {

                parent[rooty] = rootx;  // 将秩小的树根连接到秩大的树根上

            } else if (rank[rootx] < rank[rooty]) {

                parent[rootx] = rooty;

            } else {

                parent[rooty] = rootx;  // 秩相同，任意合并，并增加 rootx 的秩

                rank[rootx]++;

            }

            count--;  // 合并后，集合数量减少

        }

    }

    // 获取当前集合的数量

    int getCount() const {

        return count;

    }

    // 判断两个元素是否属于同一个集合

    bool connected(int x, int y) {

        return find(x) == find(y);

    }

};