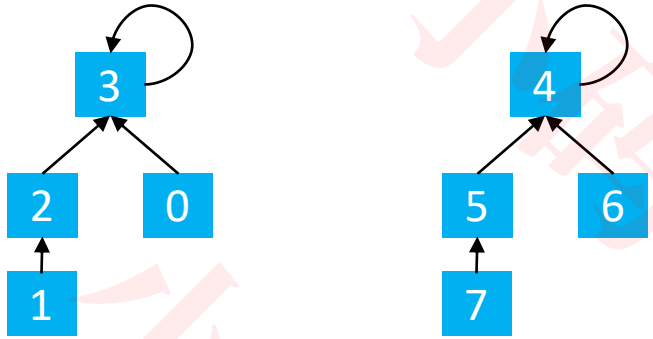
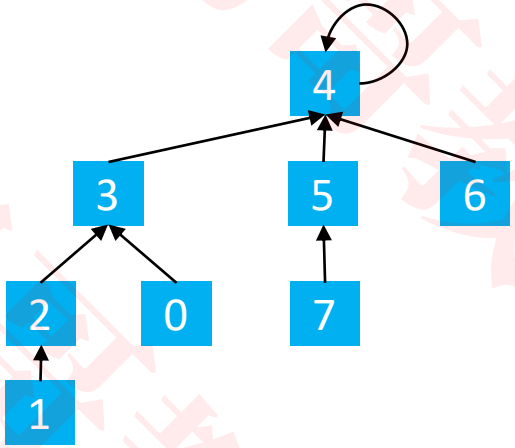
路径压缩（Path Compression）

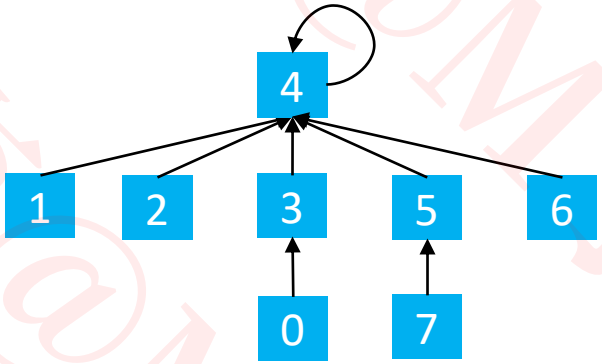


union(1, 5)

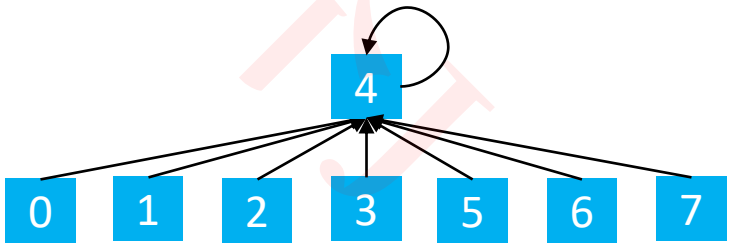


- 虽然有了基于rank的优化，树会相对平衡一点
- 但是随着Union次数的增多，树的高度依然会越来越高
  - 导致find操作变慢，尤其是底层节点（因为find是不断向上找到根节点）
- 什么是路径压缩？
  - 在find时使路径上的所有节点都指向根节点，从而降低树的高度

find(1)



find(0)、find(7)



路径压缩（Path Compression）

```
public int find(int v) {
    rangeCheck(v);
    if (parents[v] != v) {
        parents[v] = find(parents[v]);
    }
    return parents[v];
}
```

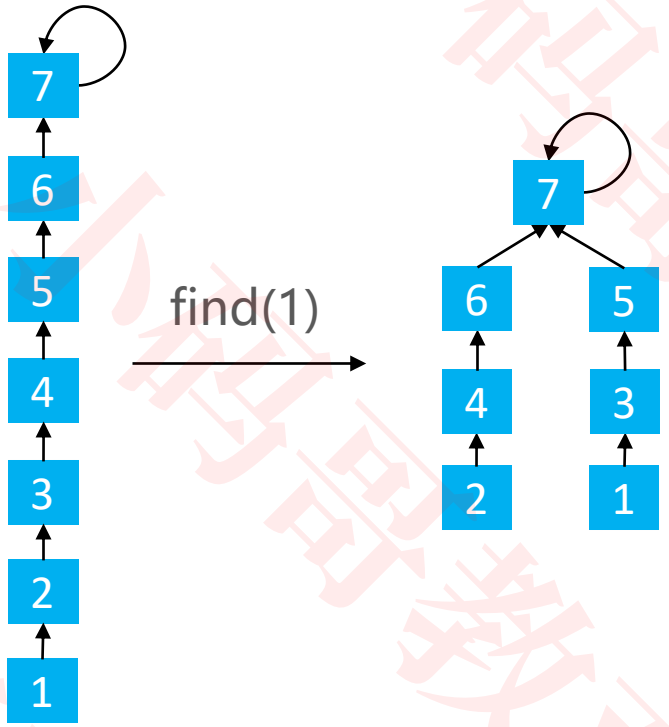
- 路径压缩使路径上的所有节点都指向根节点，所以实现成本稍高

- 还有2种更优的做法，不但能降低树高，实现成本也比路径压缩低
  - 路径分裂（Path Spliting）
  - 路径减半（Path Halving）

- 路径分裂、路径减半的效率差不多，但都比路径压缩要好

路径分裂（Path Spliting）

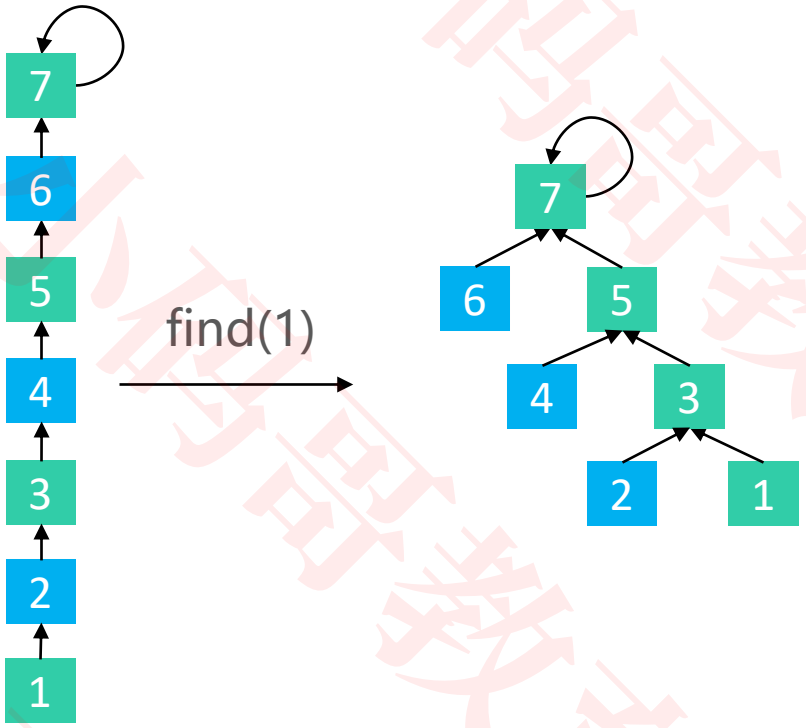
- 路径分裂：使路径上的每个节点都指向其祖父节点（parent的parent）




```
public int find(int v) {
    rangeCheck(v);
    while (v != parents[v]) {
        int parent = parents[v];
        parents[v] = parents[parent];
        v = parent;
    }
    return v;
}
```

路径减半（Path Halving）

- 路径减半：使路径上每隔一个节点就指向其祖父节点（parent的parent）




```
public int find(int v) {
    rangeCheck(v);
    while (v != parents[v]) {
        parents[v] = parents[parents[v]];
        v = parents[v];
    }
    return v;
}
```