算法与数据结构

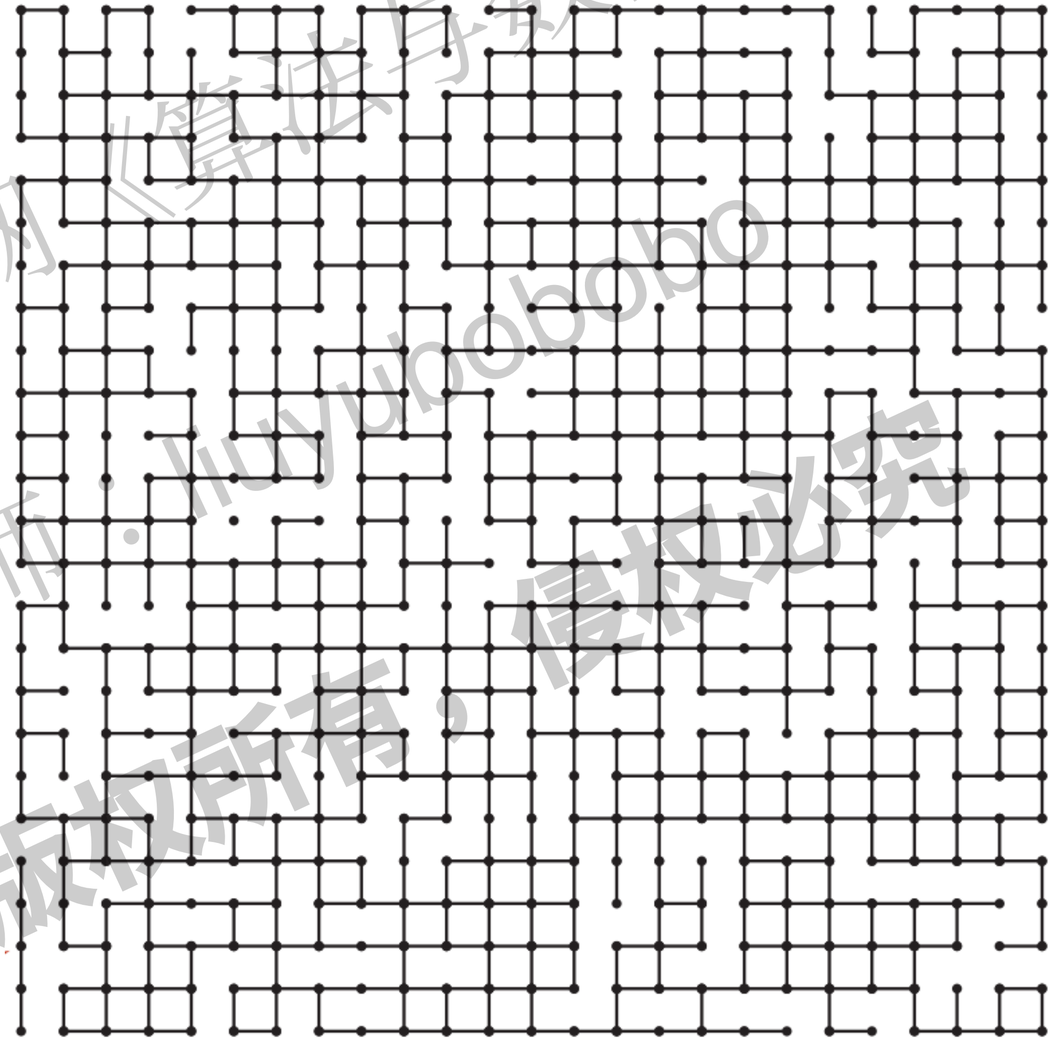
liuyubobobo

并查集 Union Find

一种很不一样的树形结构

连接问题 Connectivity Problem

连接问题



连接问题

网络中节点间的连接状态

- 网络是个抽象的概念：用户之间形成的网络

数学中的集合类实现

连接问题和路径问题

比路径问题要回答的问题少

- 和二分查找作比较
- 和select作比较
- 和堆作比较

并查集 Union Find

对于一组数据，主要支持两个动作：

- union(p , q)
- find(p)

用来回答一个问题

- isConnected(p , q)

并查集的基本数据表示

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1

并查集的基本数据表示

id	0	1	2	3	4	5	6	7	8	9
	0	1	0	1	0	1	0	1	0	1

操作：并查集的基本数据表示

Quick Find

id	0	1	2	3	4	5	6	7	8	9
	0	1	0	1	0	1	0	1	0	1

Quick Find 时间复杂度 O(1)

Quick Find 下的 Union

	0	1	2	3	4	5	6	7	8	9
id	0	1	0	1	0	1	0	1	0	1

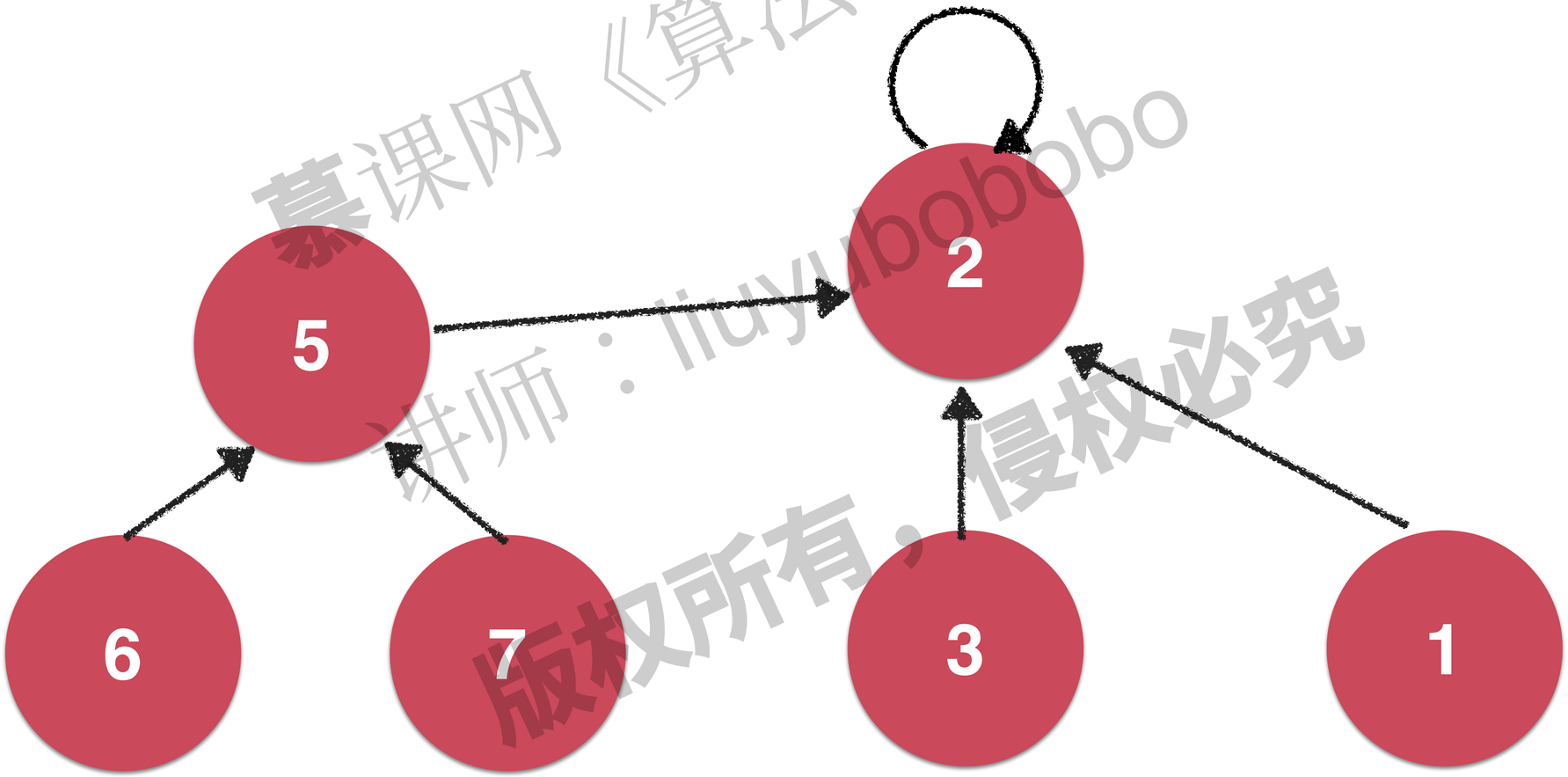
Quick Find 下的 Union 时间复杂度 O(n)

操作：Quick Find 和相应的 Union

操作：测试并查集的时间效率

并查集的另外一种实现思路

将每一个元素，看做是一个节点



Quick Union

Quick Union 下的数据表示

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	3	4	5	6	7	8	9

Quick Union

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	3	4	5	6	7	8	9

union 4,3

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	3	4	5	6	7	8	9

union 4,3

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	3	4	5	6	7	8	9

union 4,3

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	3	3	5	6	7	8	9

union 3,8

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	3	3	5	6	7	8	9

union 3, 8

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	3	3	5	6	7	8	9

union 3, 8

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	6	7	8	9

union 6,5

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	6	7	8	9

union 6,5

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	6	7	8	9

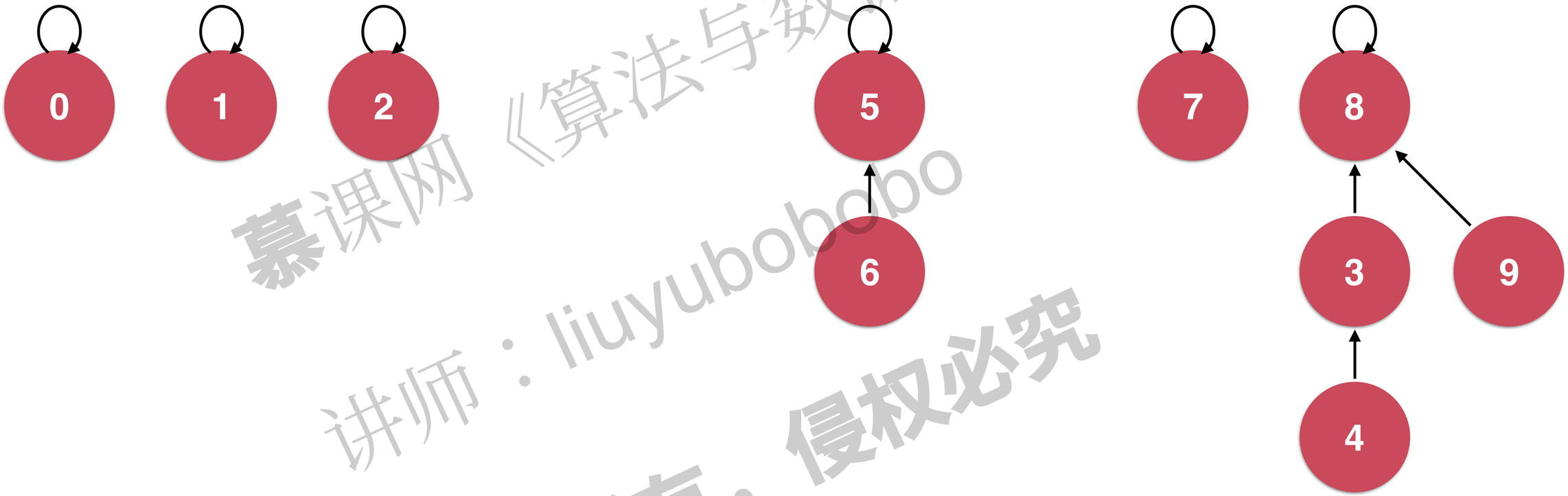
union 6,5

	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	9

union 9,4

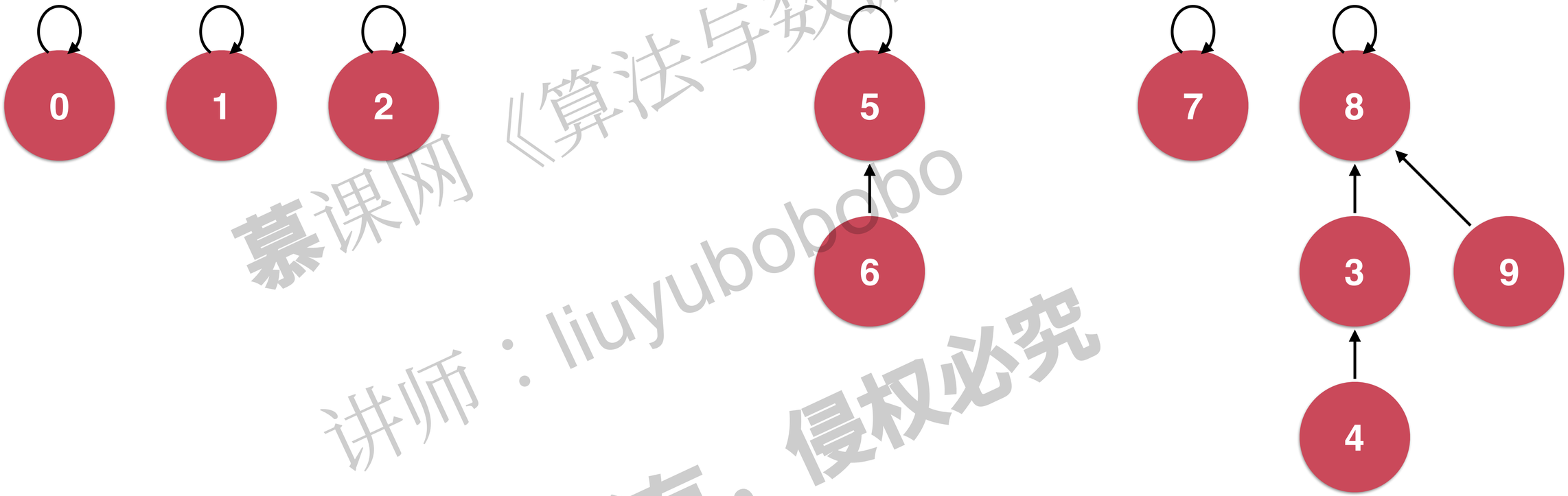
	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	9

union 9,4



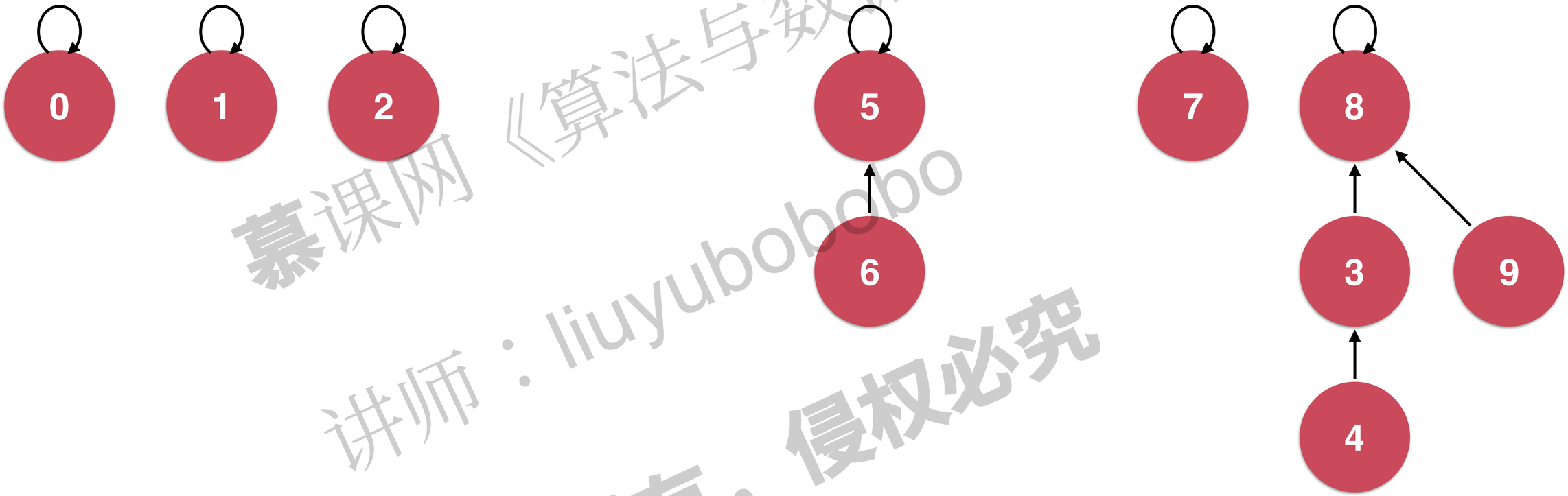
	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	9

union 9,4



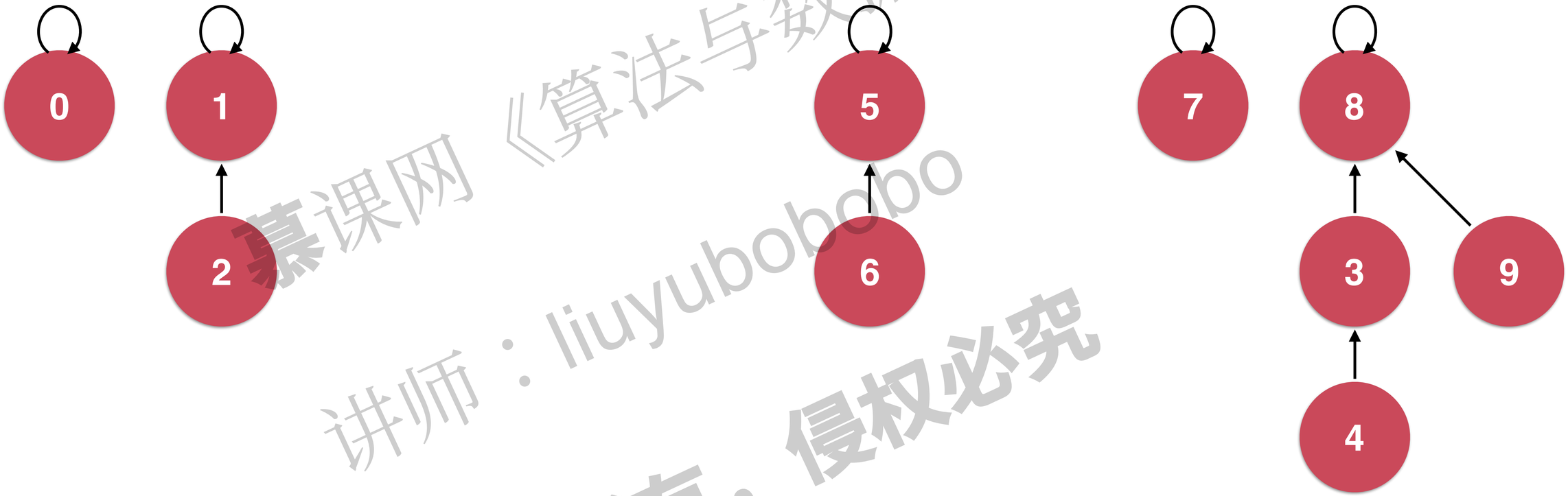
	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	8

union 2,1



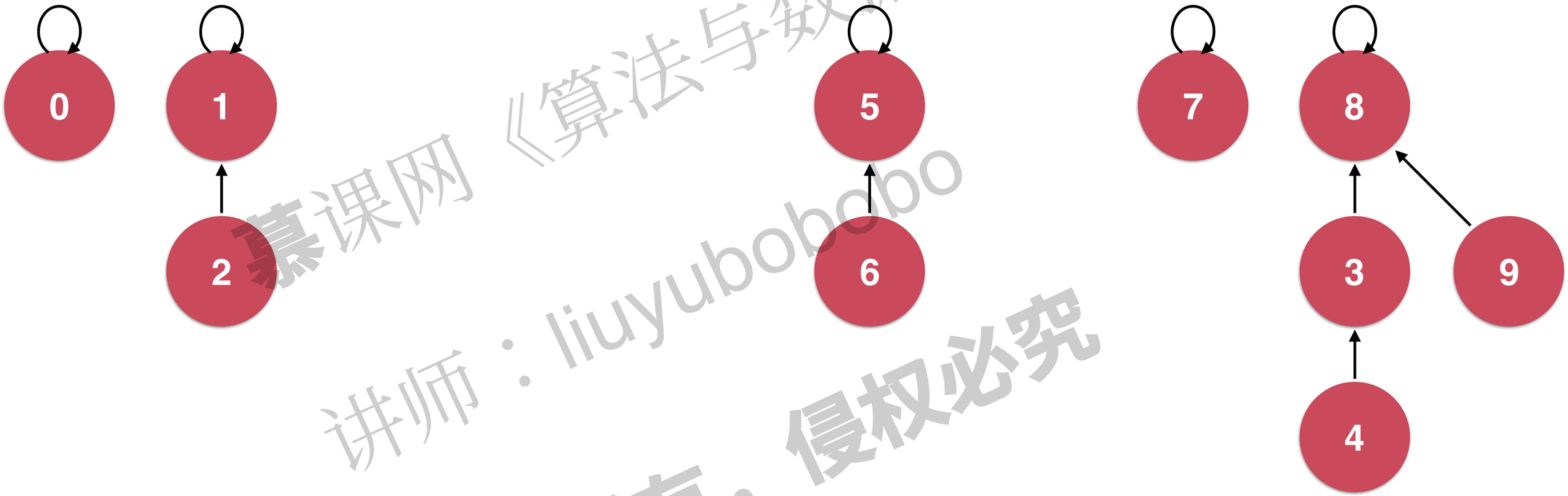
	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	8

union 2,1



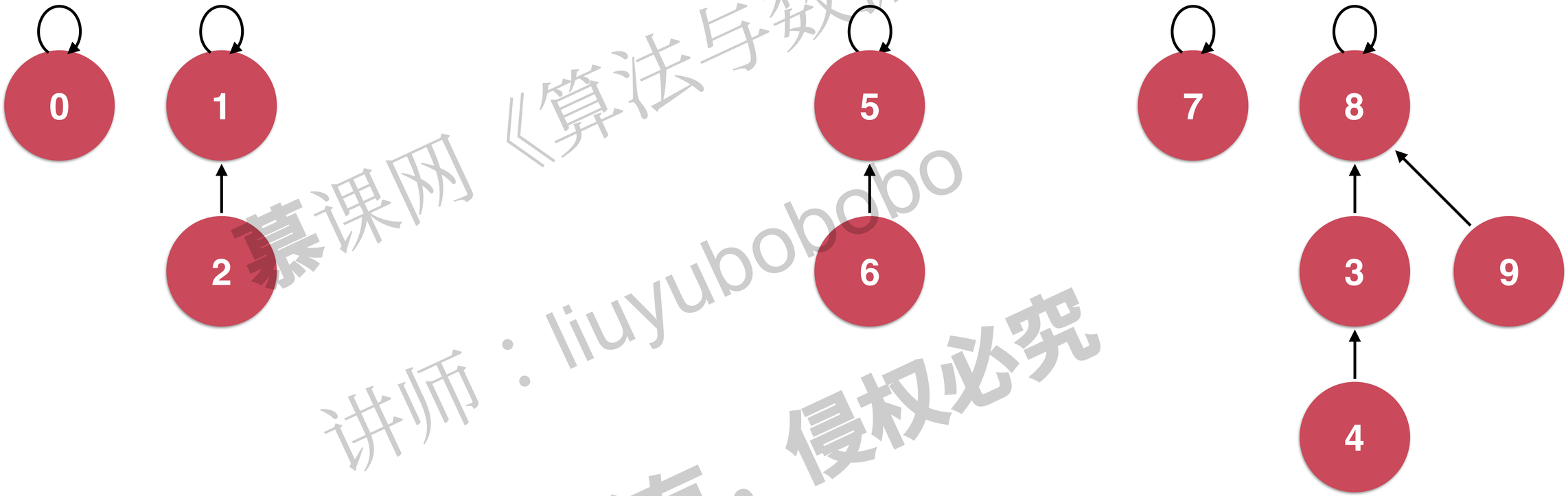
	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	8

union 2,1



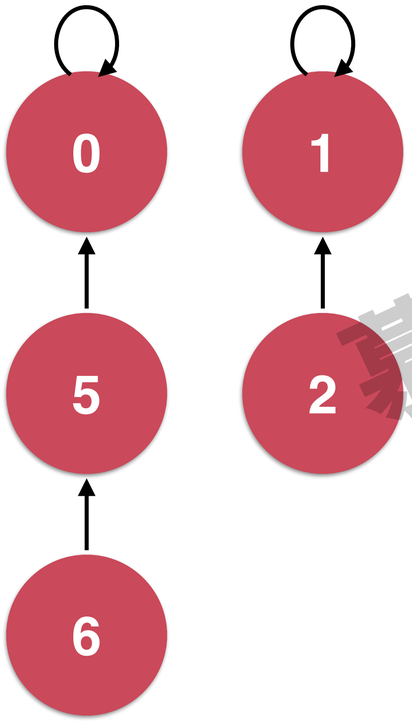
	0	1	2	3	4	5	6	7	8	9
parent	0	1	1	8	3	5	5	7	8	8

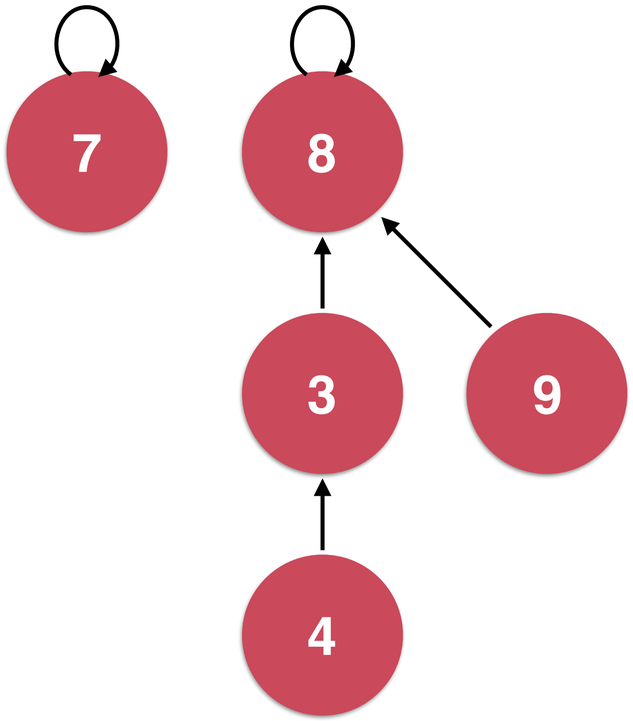
union 5,0



	0	1	2	3	4	5	6	7	8	9
parent	0	1	1	8	3	5	5	7	8	8

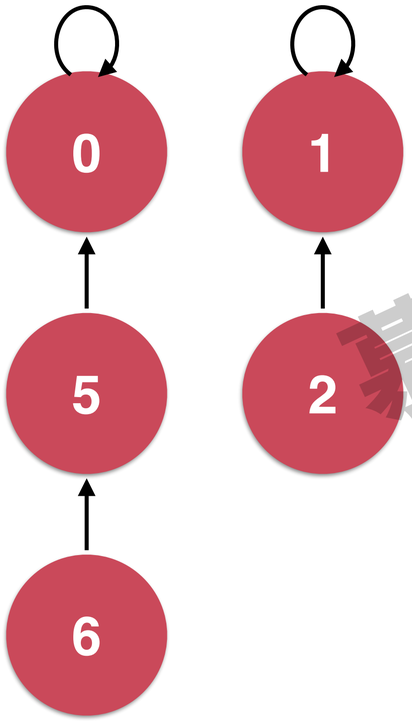
union 5,0

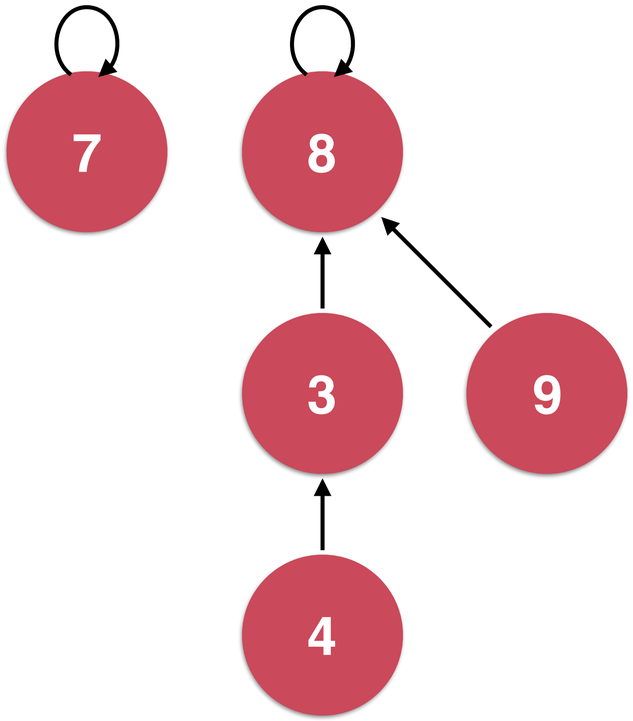




	0	1	2	3	4	5	6	7	8	9
parent	0	1	1	8	3	5	5	7	8	8

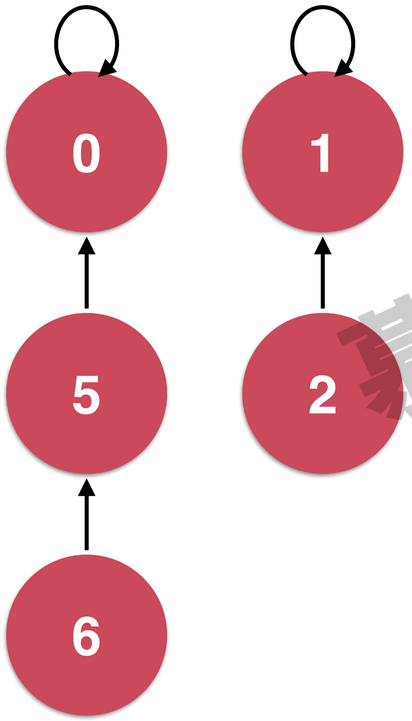
union 5,0

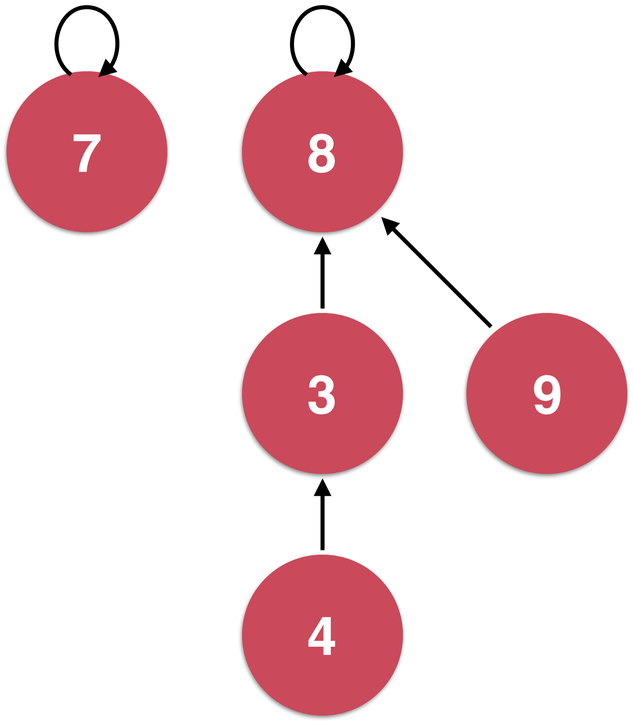




	0	1	2	3	4	5	6	7	8	9
parent	0	1	1	8	3	0	5	7	8	8

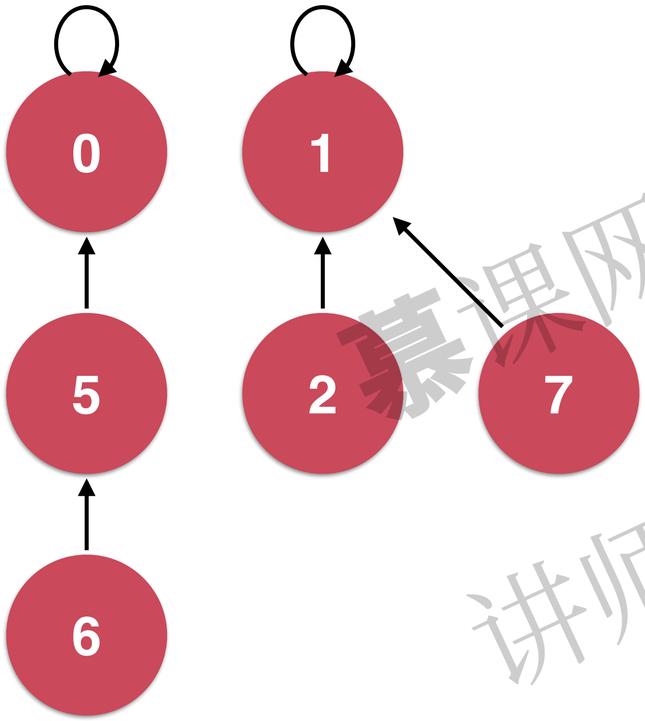
union 7,2

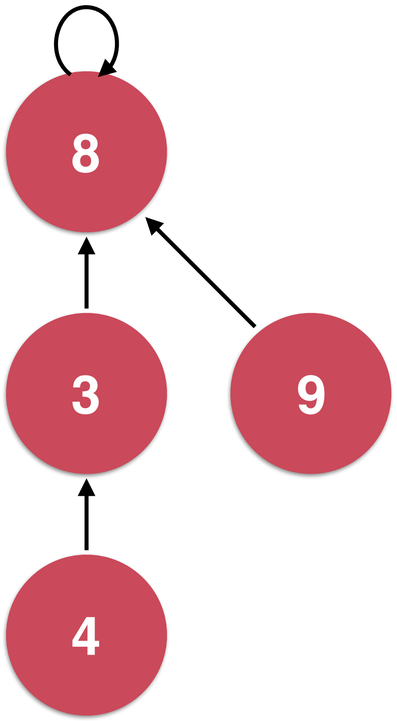




	0	1	2	3	4	5	6	7	8	9
parent	0	1	1	8	3	0	5	7	8	8

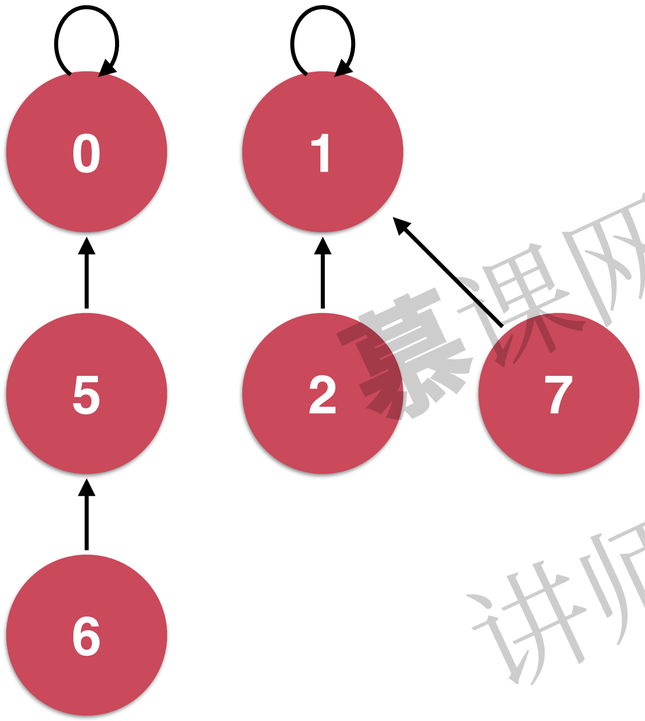
union 7,2

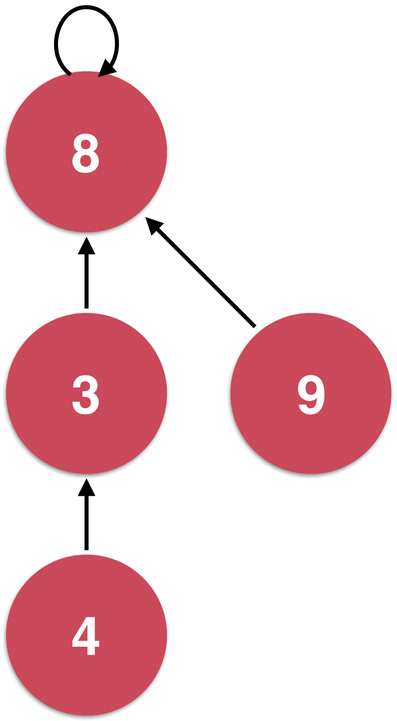




	0	1	2	3	4	5	6	7	8	9
parent	0	1	1	8	3	0	5	7	8	8

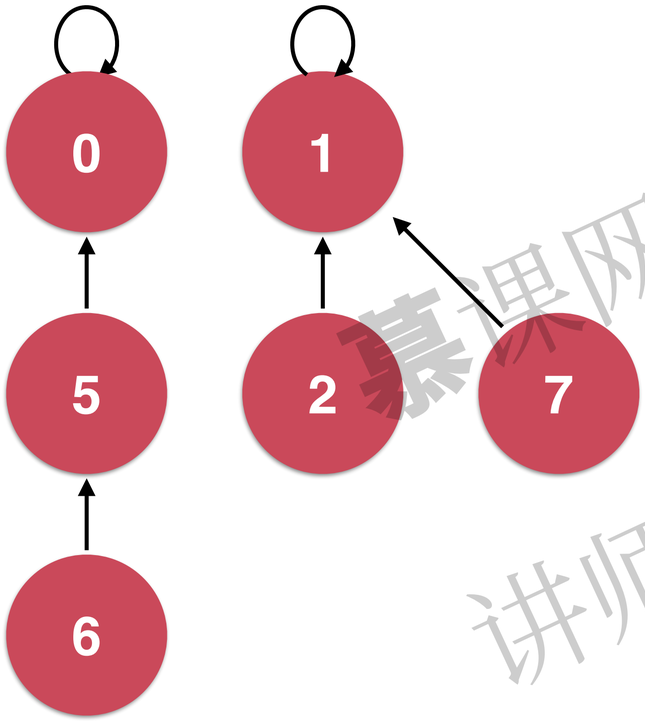
union 7,2

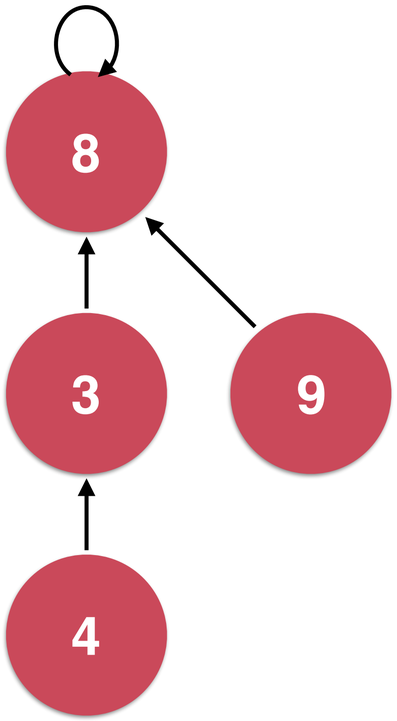




	0	1	2	3	4	5	6	7	8	9
parent	0	1	1	8	3	0	5	1	8	8

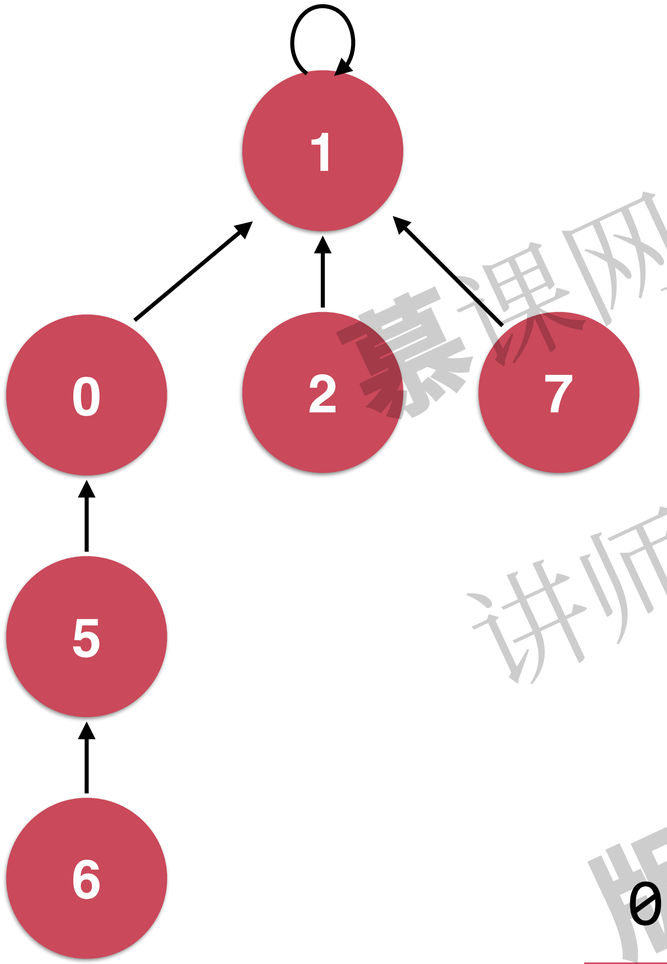
union 6,2

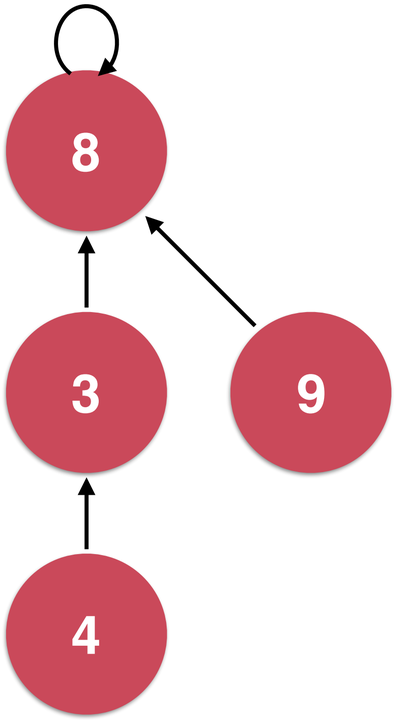




	0	1	2	3	4	5	6	7	8	9
parent	0	1	1	8	3	0	5	1	8	8

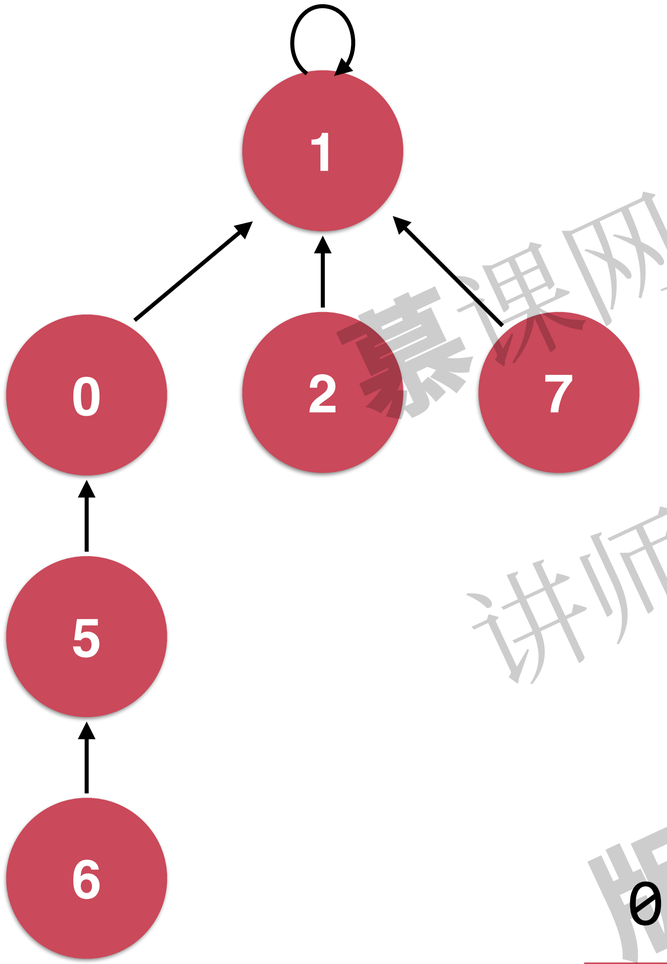
union 6,2

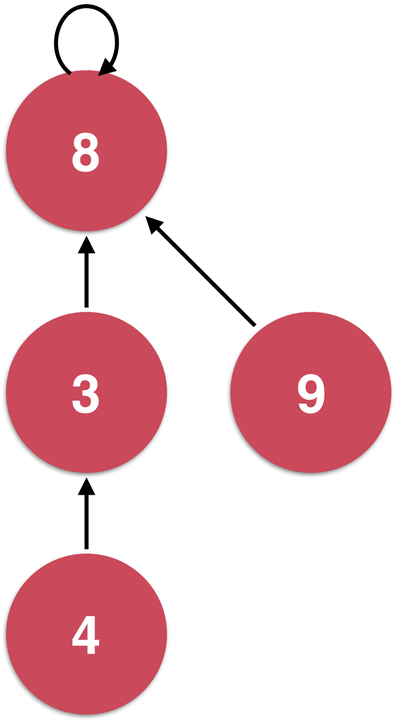




	0	1	2	3	4	5	6	7	8	9
parent	0	1	1	8	3	0	5	1	8	8

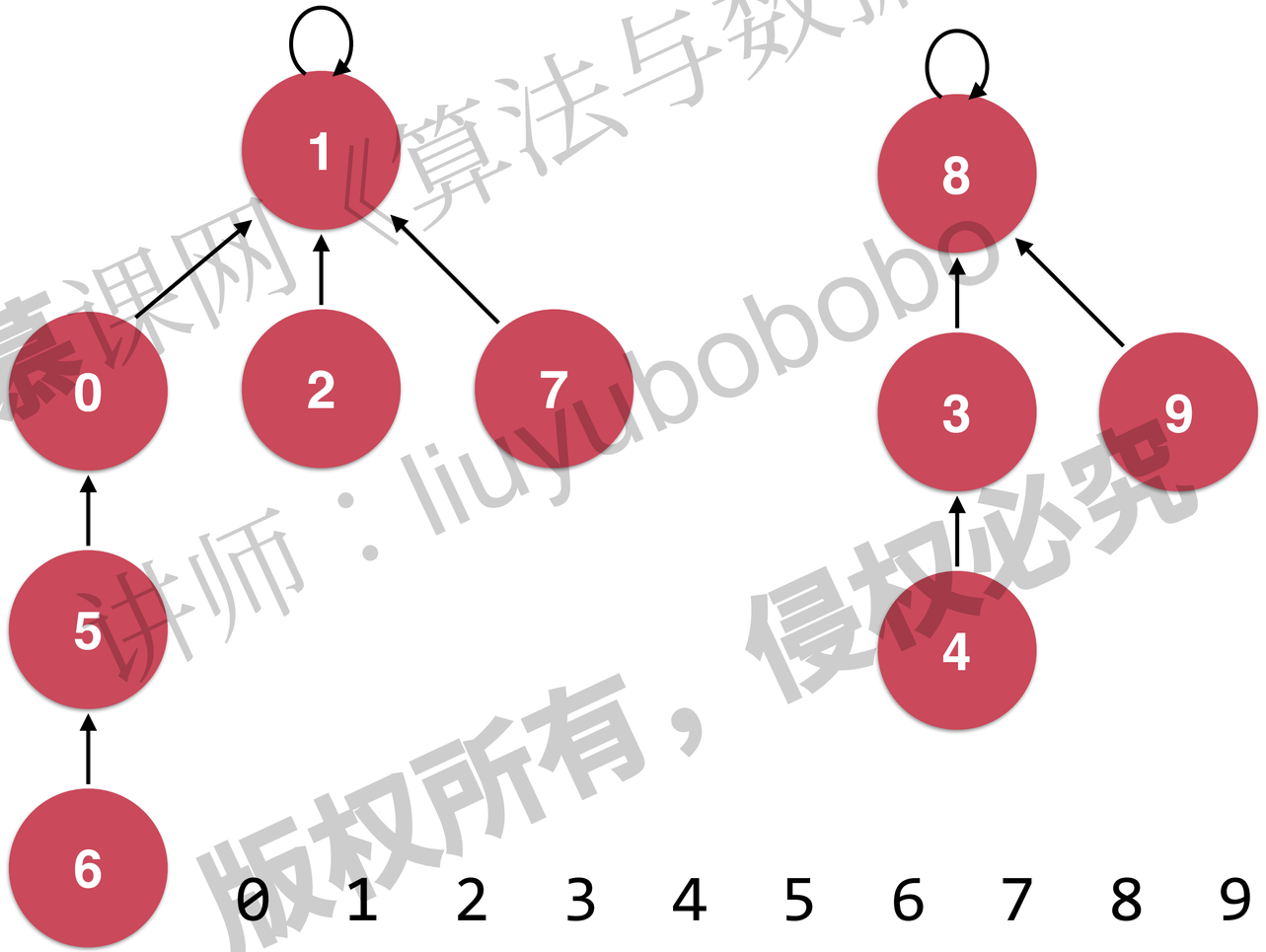
union 6,2





	0	1	2	3	4	5	6	7	8	9
parent	1	1	1	8	3	0	5	1	8	8

并查集



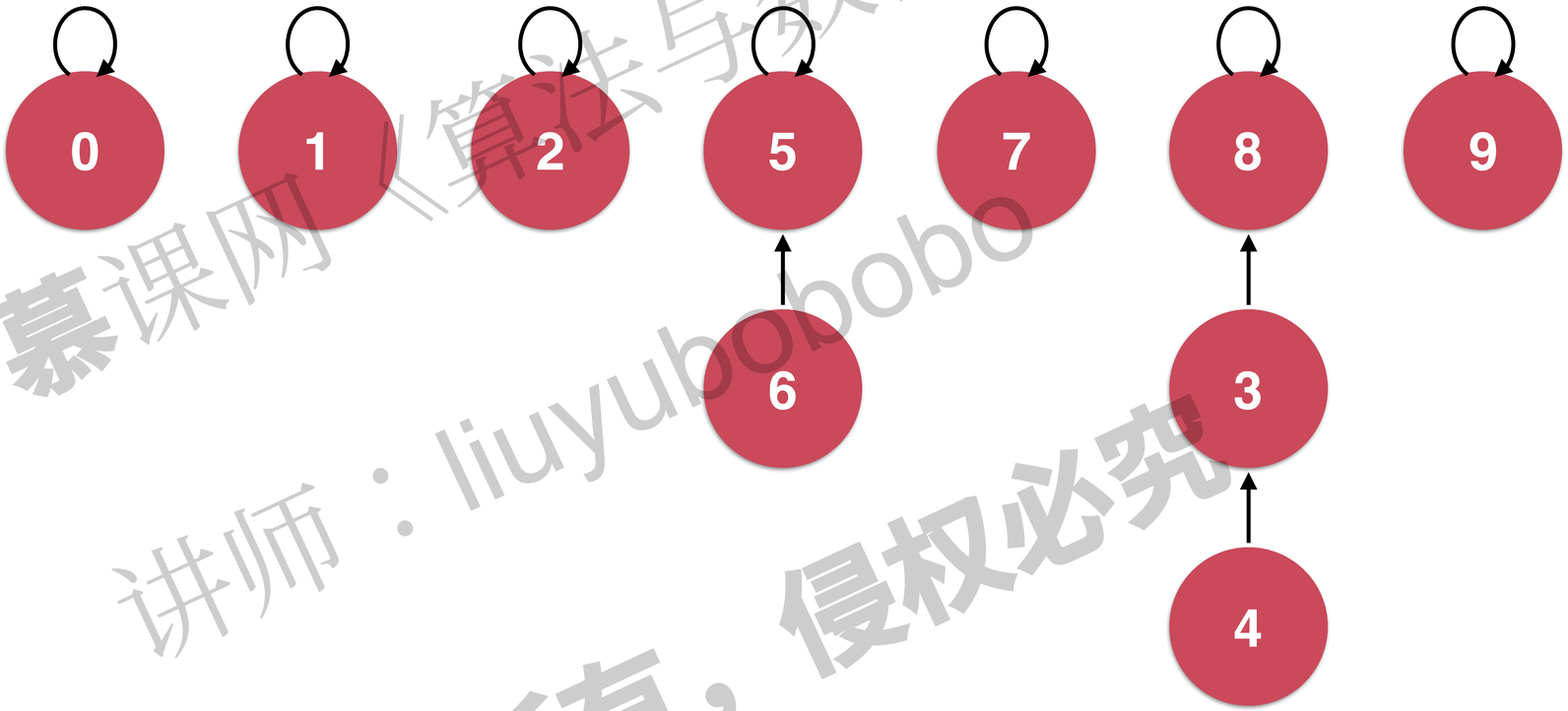
	0	1	2	3	4	5	6	7	8	9
parent	1	1	1	8	3	0	5	1	8	8

操作：Quick Union 的并查集实现

操作：Quick Union 和 Quick Find 的时间效率比较

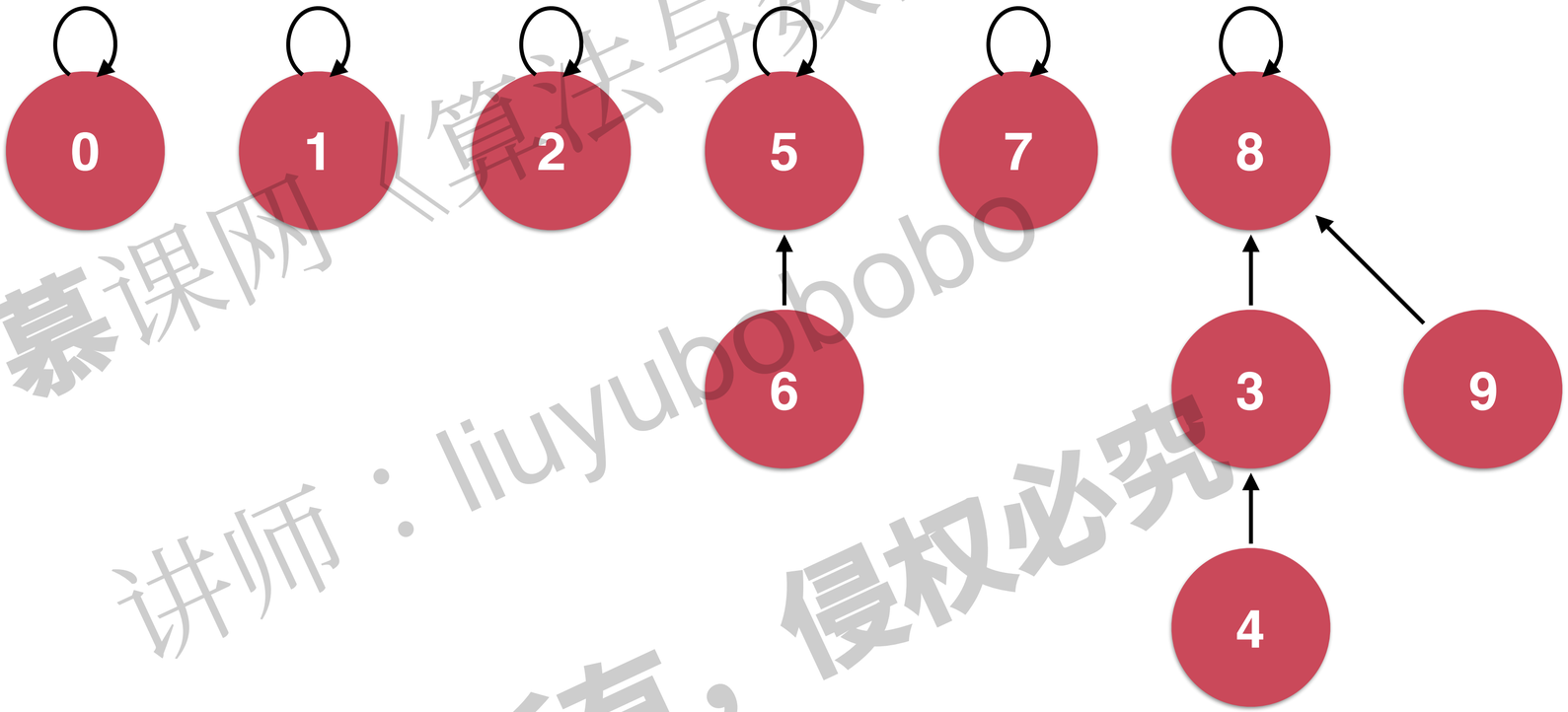
并查集的优化

union 9,4



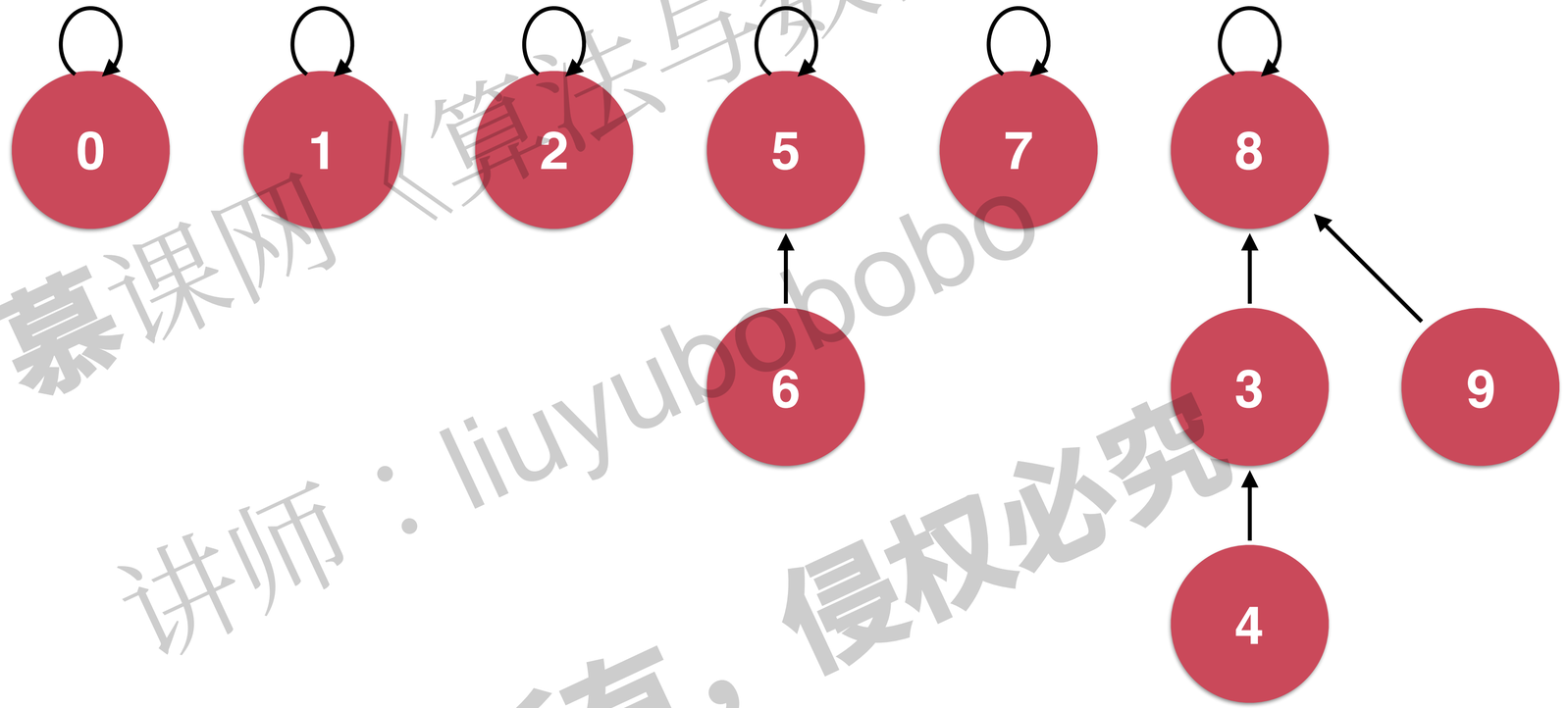
	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	9

union 9,4



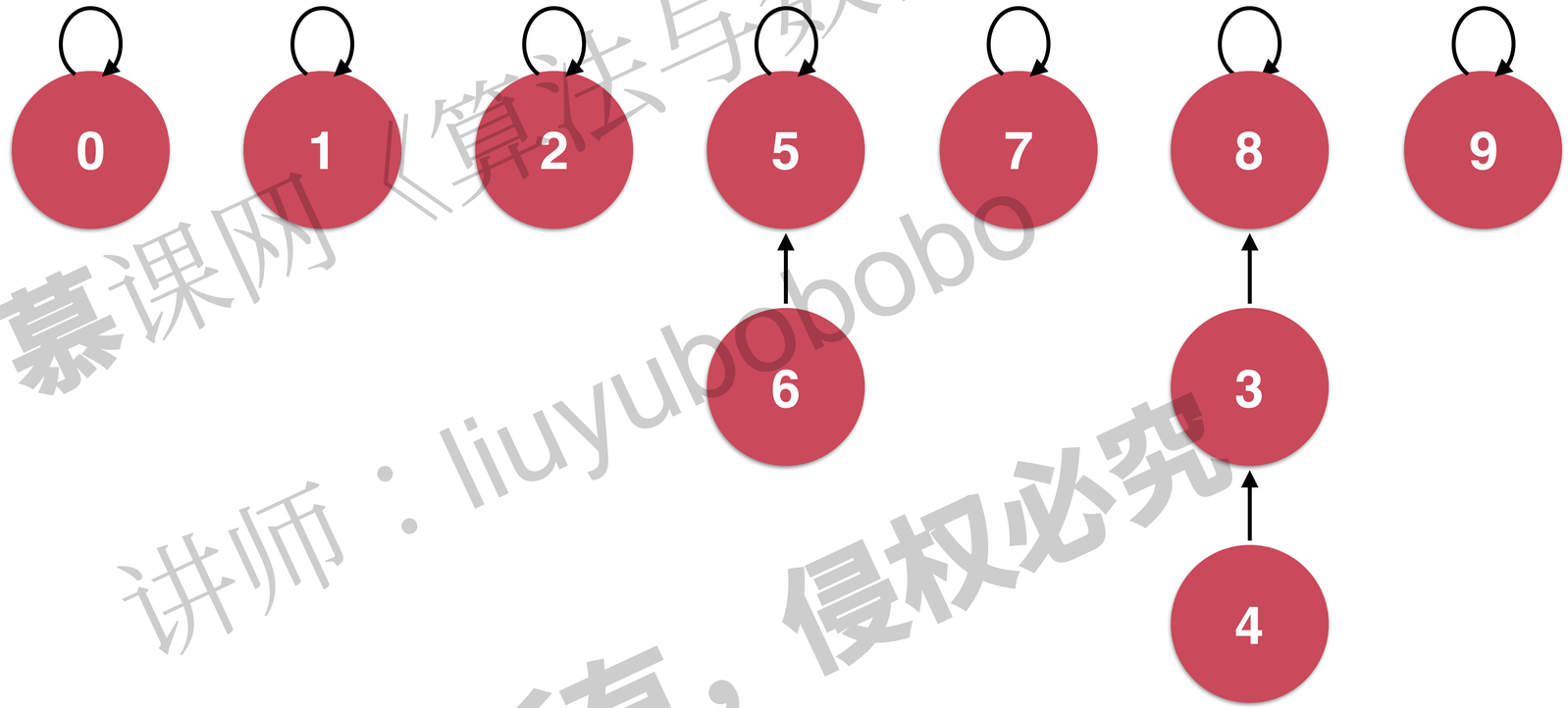
	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	9

union 9,4



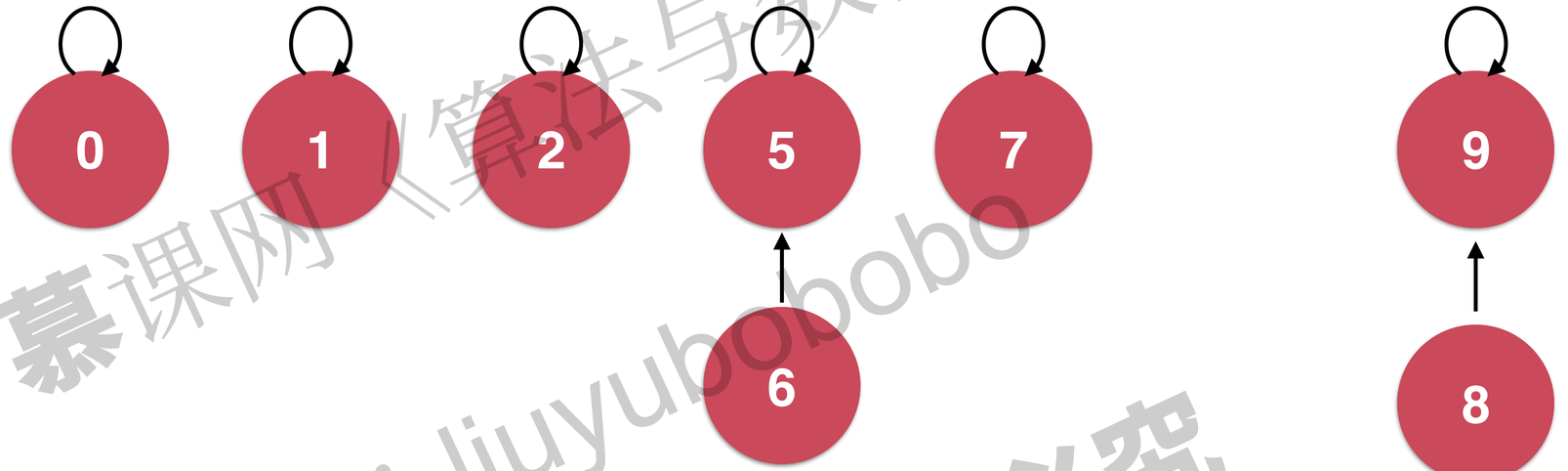
	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	8

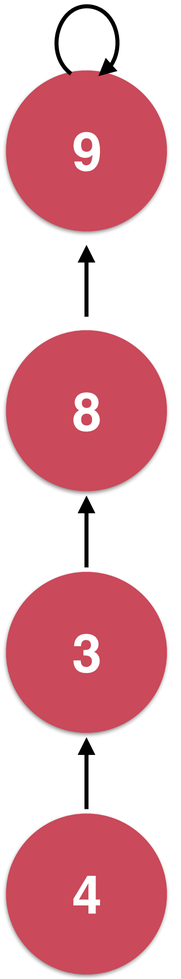
union 4，9



	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	9

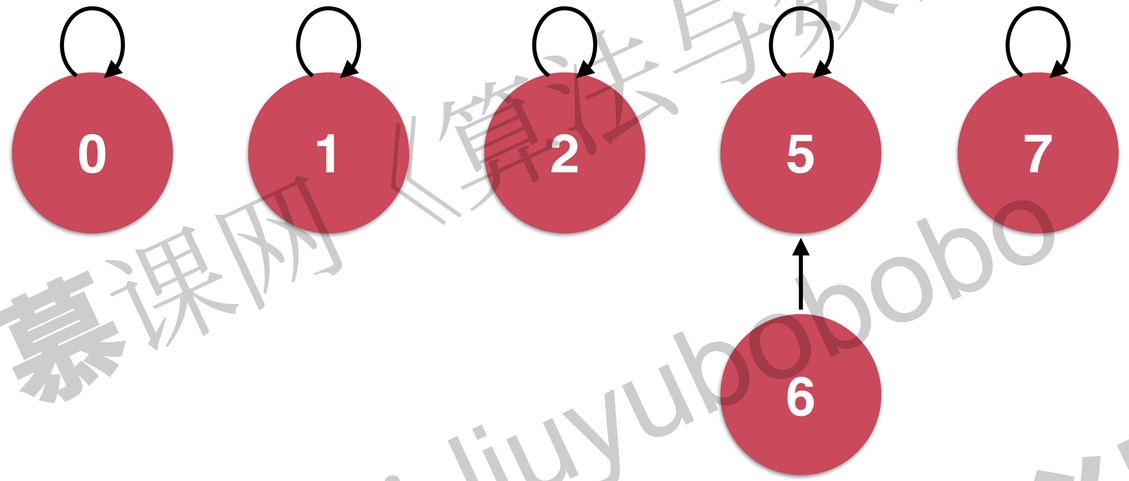
union 4，9

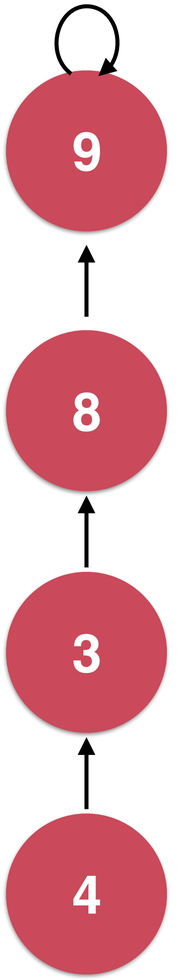




	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	8	9

union 4, 9



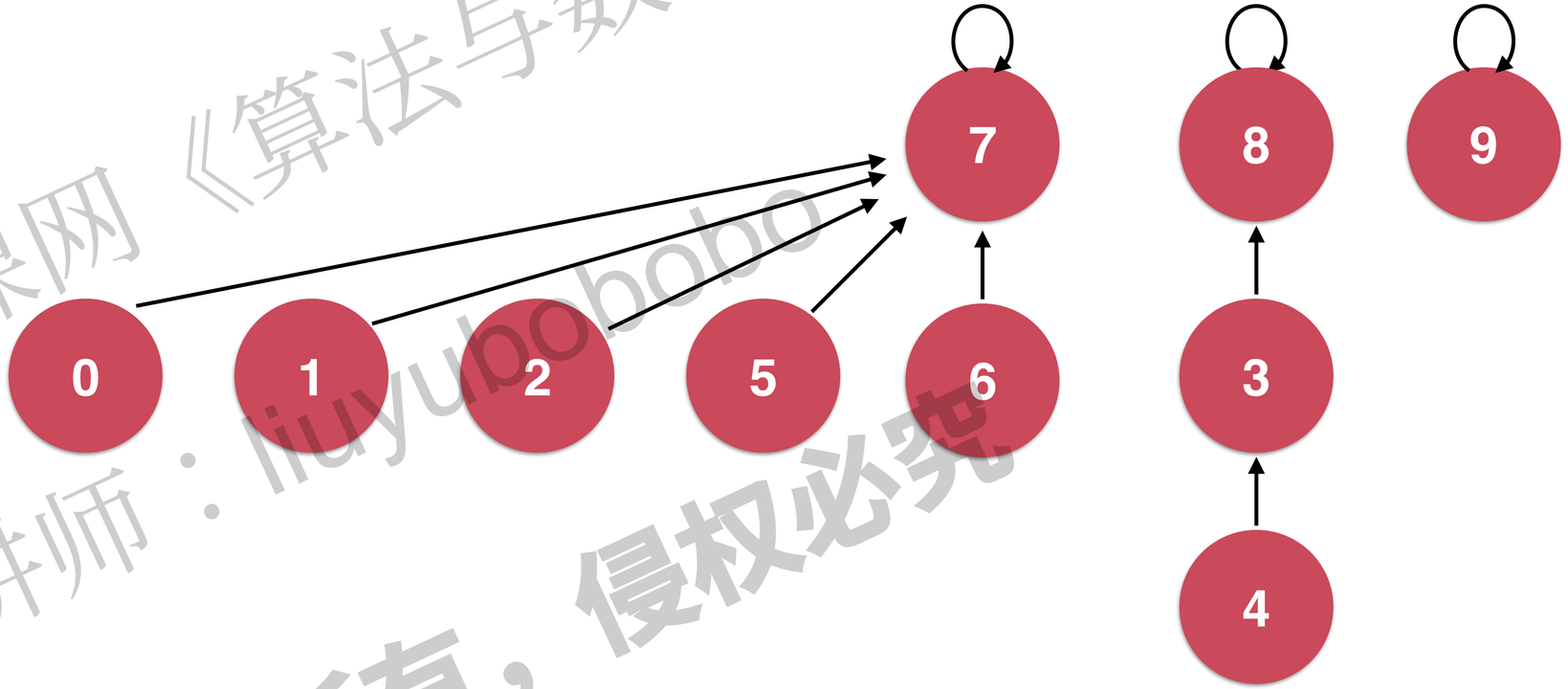


	0	1	2	3	4	5	6	7	8	9
parent	0	1	2	8	3	5	5	7	9	9

操作：基于sz的优化

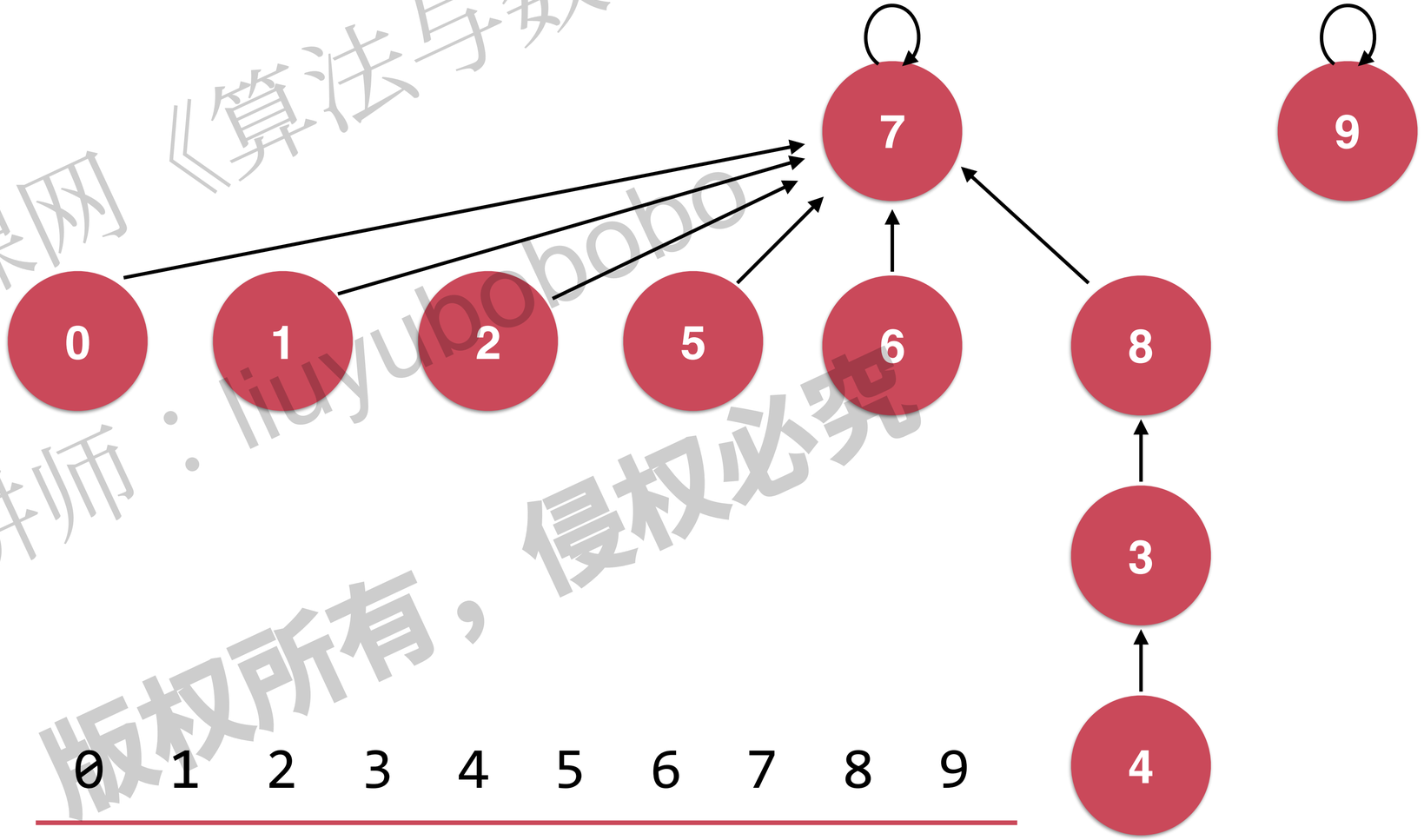
操作：Quick Union 和优化后的时间效率比较

union 4,2



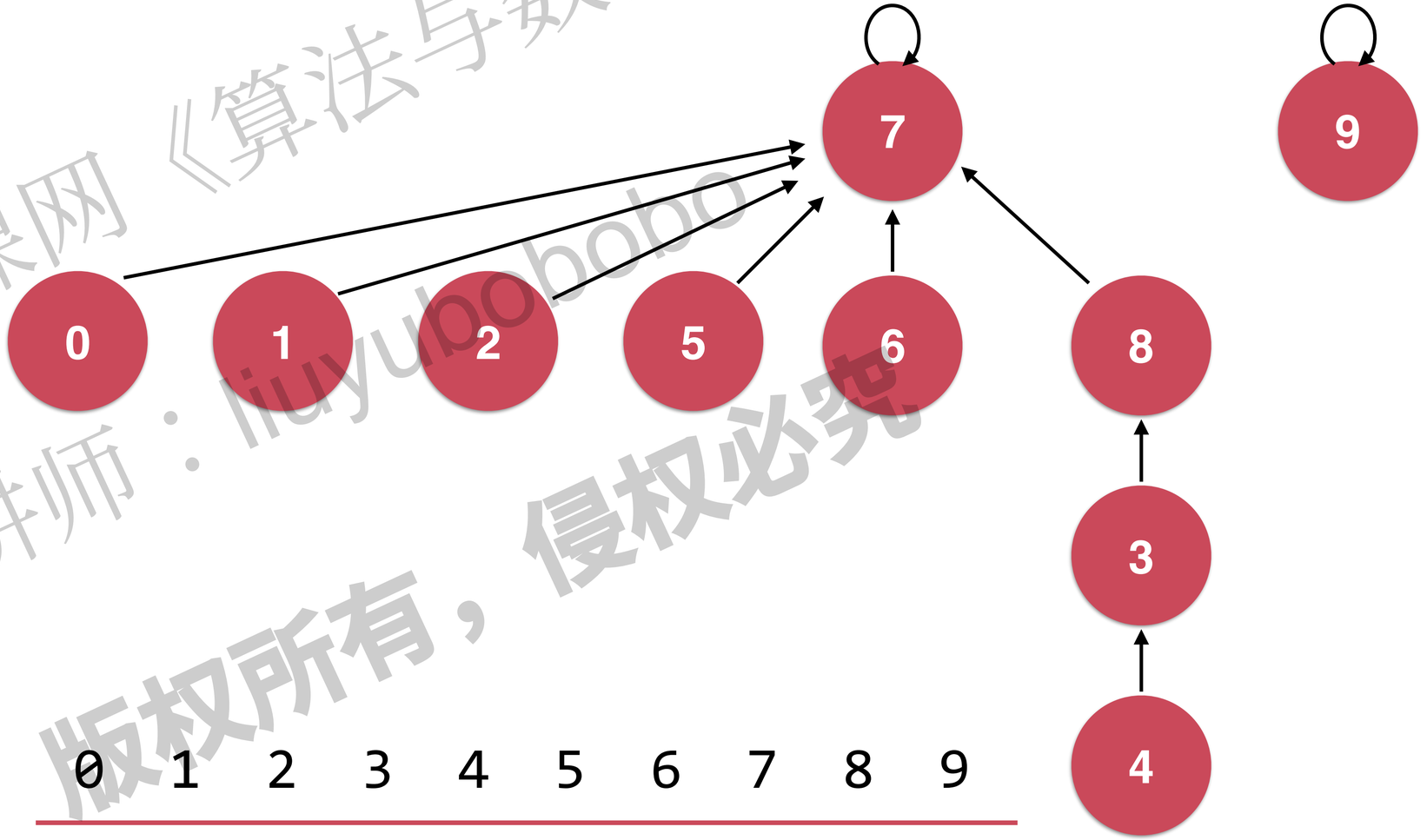
	0	1	2	3	4	5	6	7	8	9
parent	7	7	7	8	3	7	7	7	8	9

union 4,2



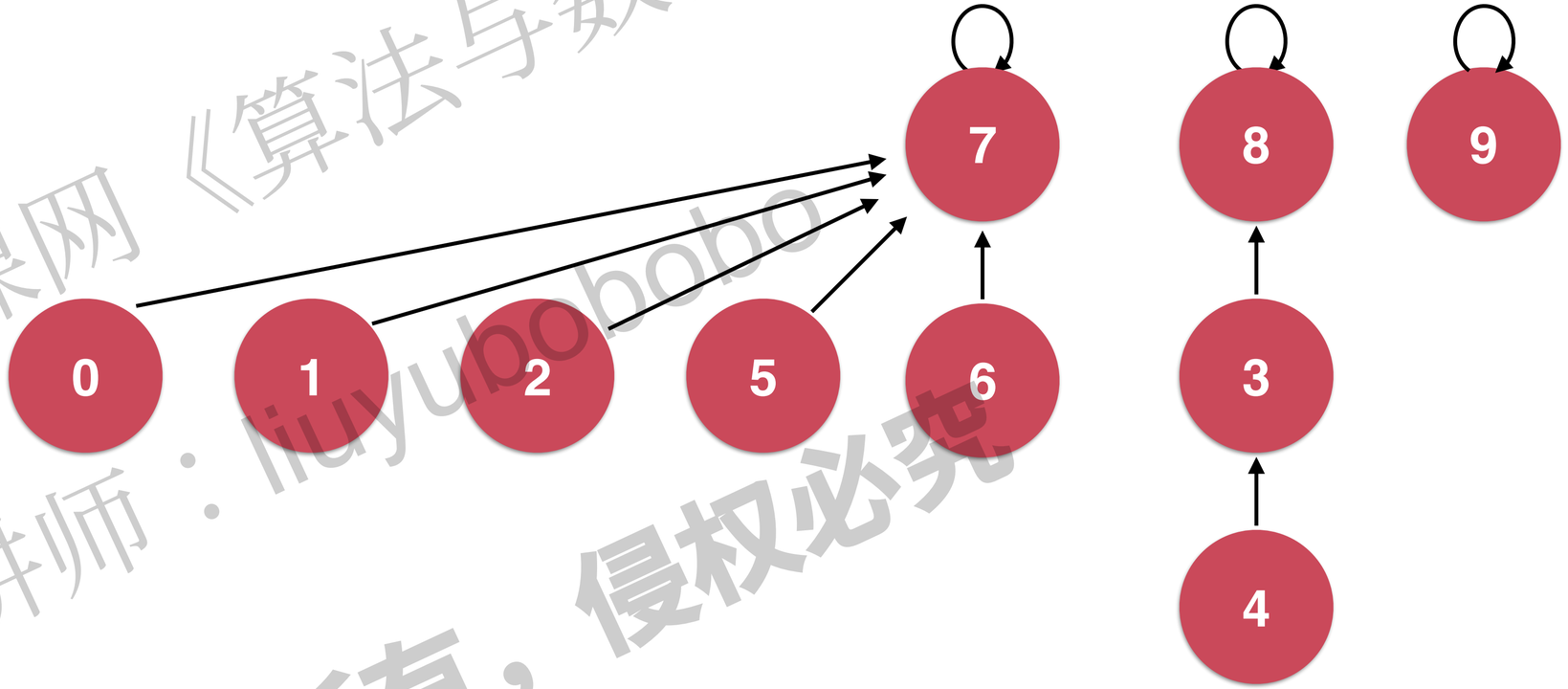
	0	1	2	3	4	5	6	7	8	9
parent	7	7	7	8	3	7	7	7	8	9

union 4,2



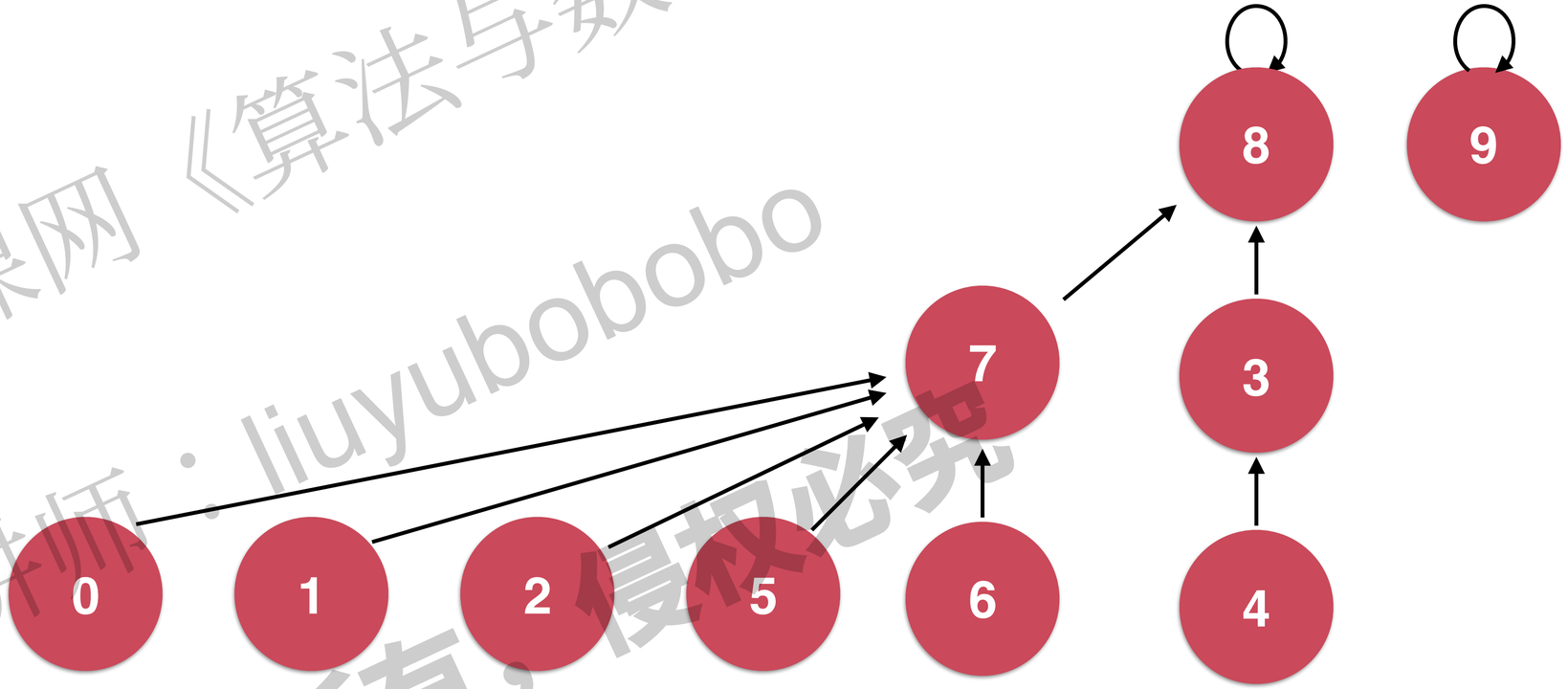
	0	1	2	3	4	5	6	7	8	9
parent	7	7	7	8	3	7	7	7	7	9

union 4,2



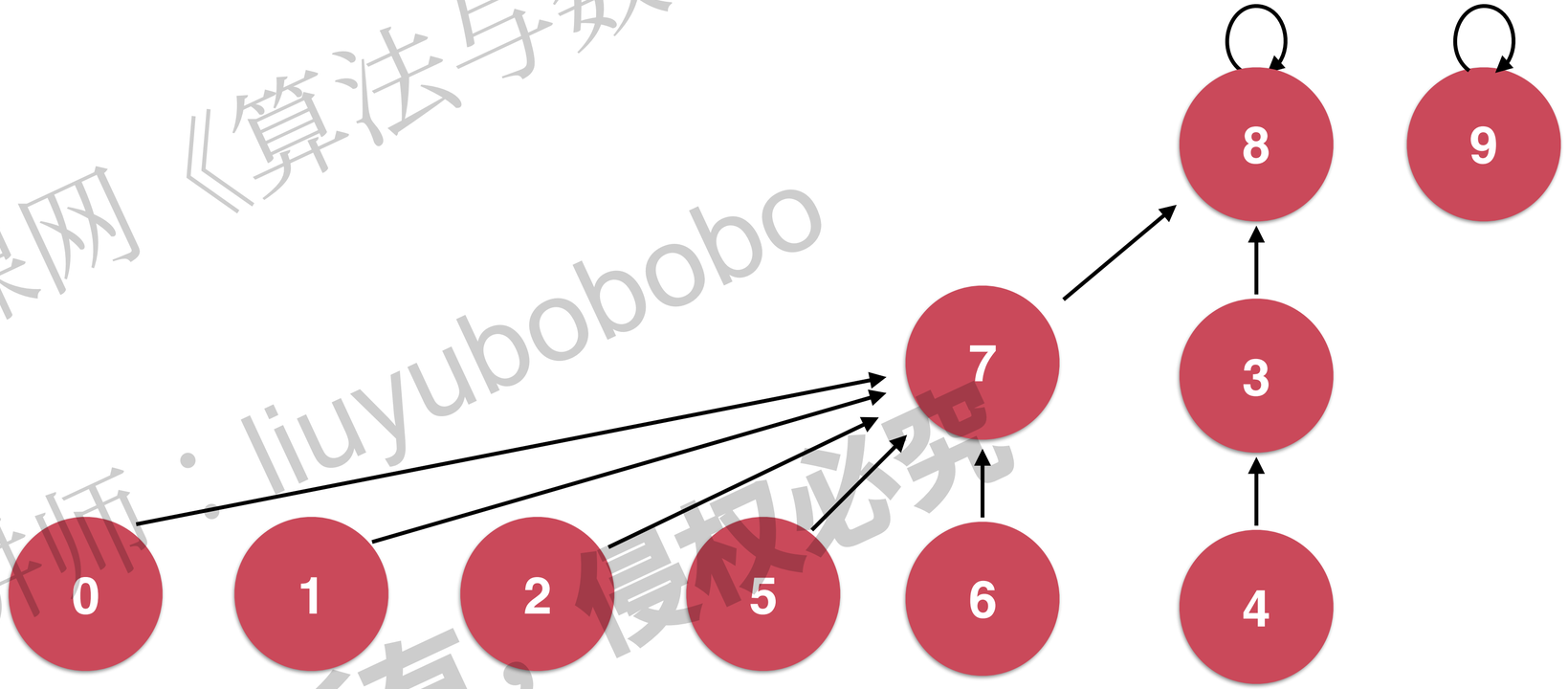
	0	1	2	3	4	5	6	7	8	9
parent	7	7	7	8	3	7	7	7	8	9

union 4,2



	0	1	2	3	4	5	6	7	8	9
parent	7	7	7	8	3	7	7	7	8	9

union 4,2



	0	1	2	3	4	5	6	7	8	9
parent	7	7	7	8	3	7	7	8	8	9

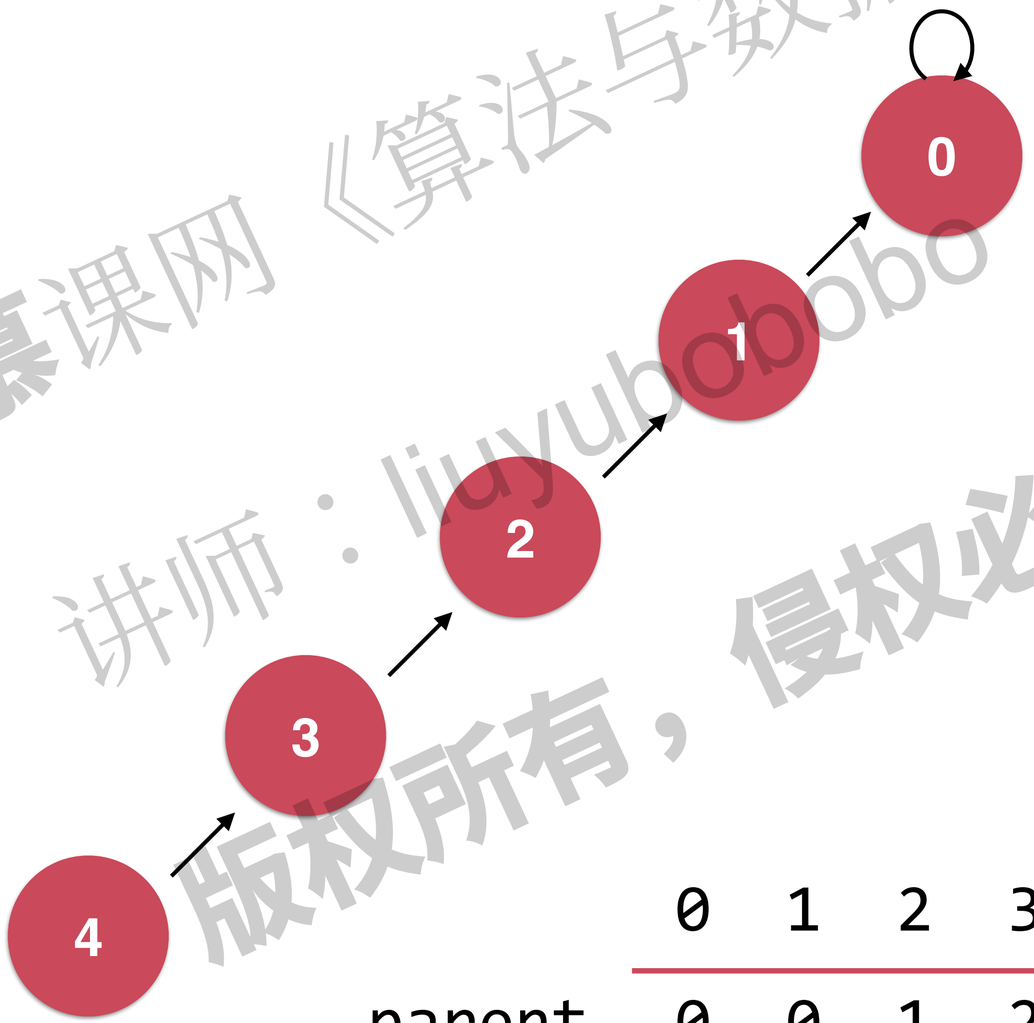
基于rank的优化

rank[i] 表示根节点为i的树的高度

操作：基于rank的优化

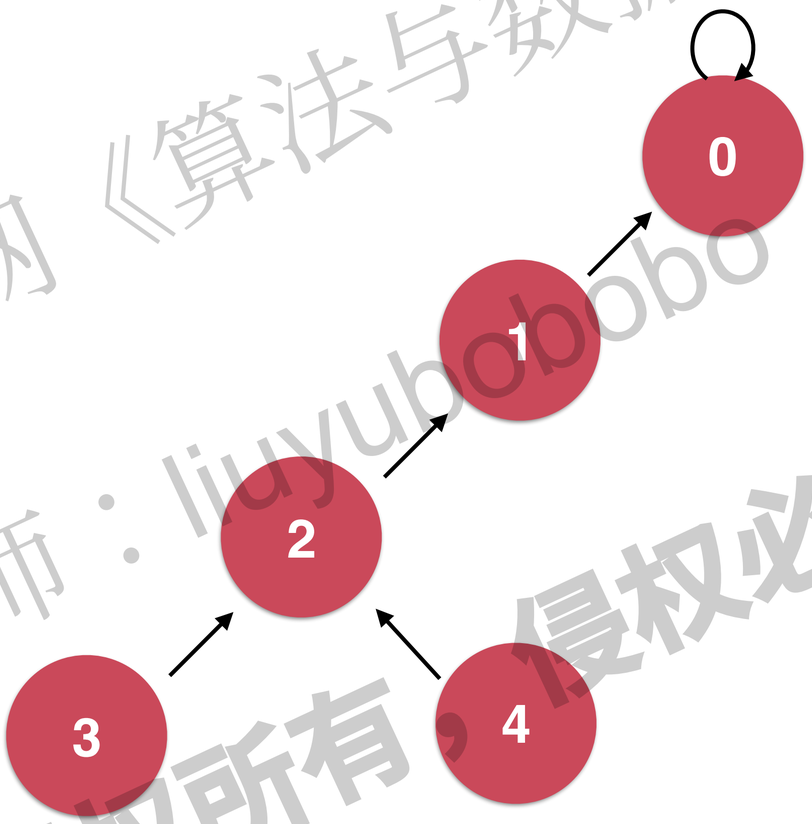
路径压缩 Path Compression

find 4



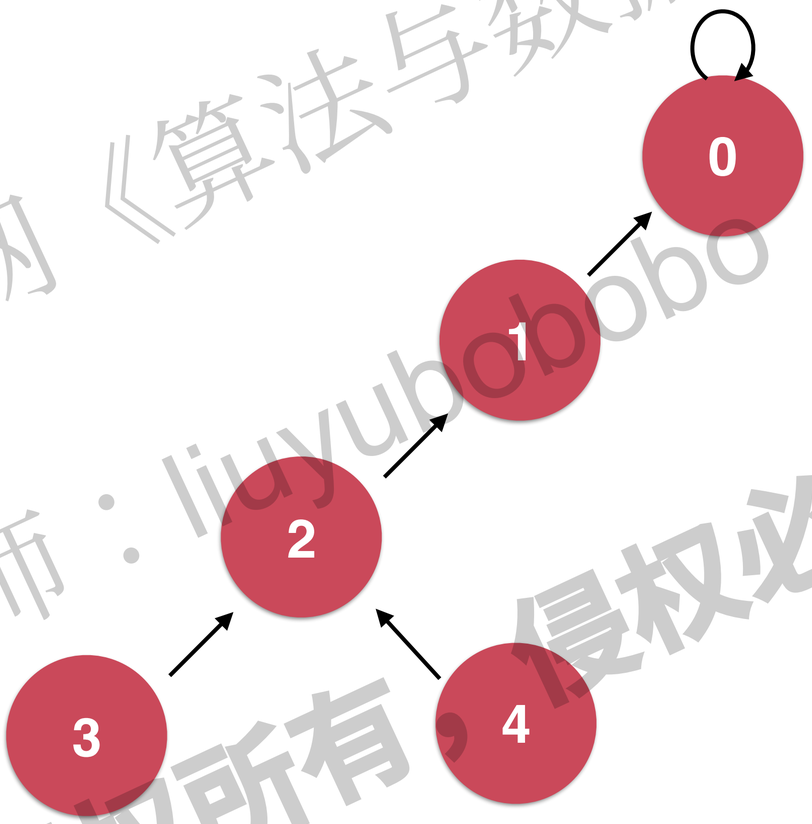
	0	1	2	3	4
parent	0	0	1	2	3

find 4



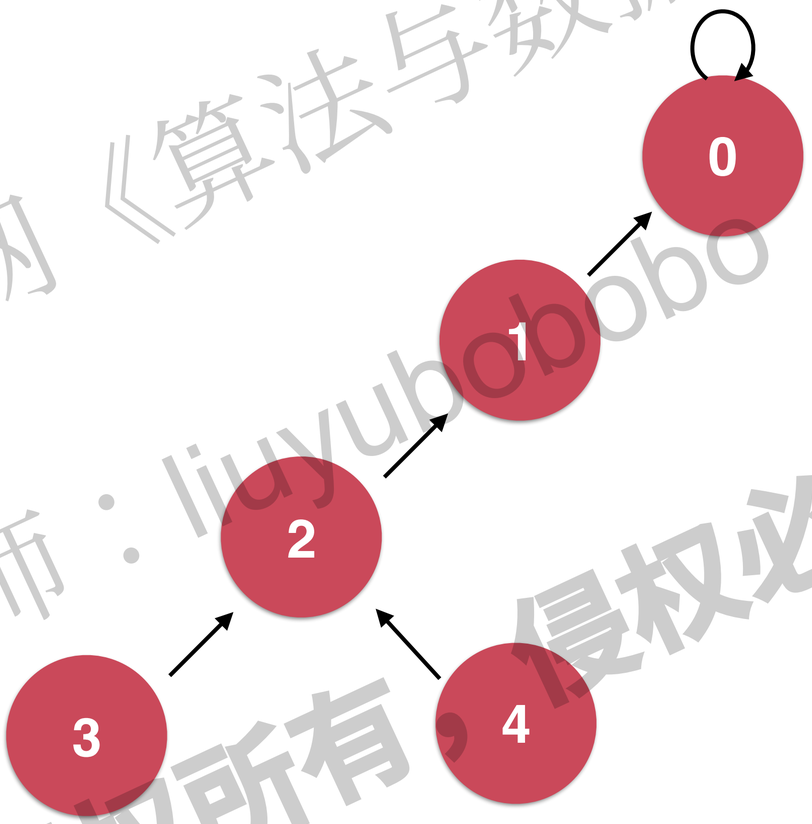
	0	1	2	3	4
parent	0	0	1	2	3

find 4



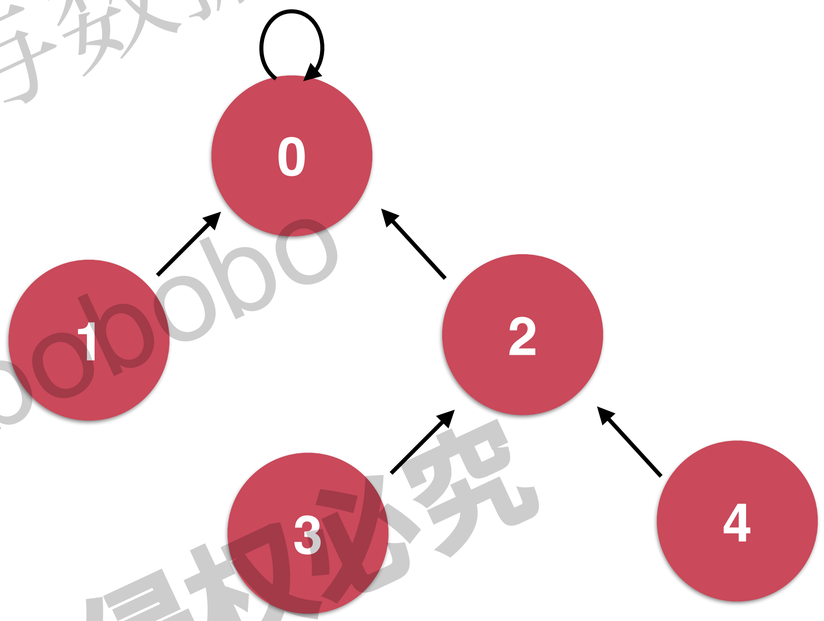
	0	1	2	3	4
parent	0	0	1	2	2

find 4



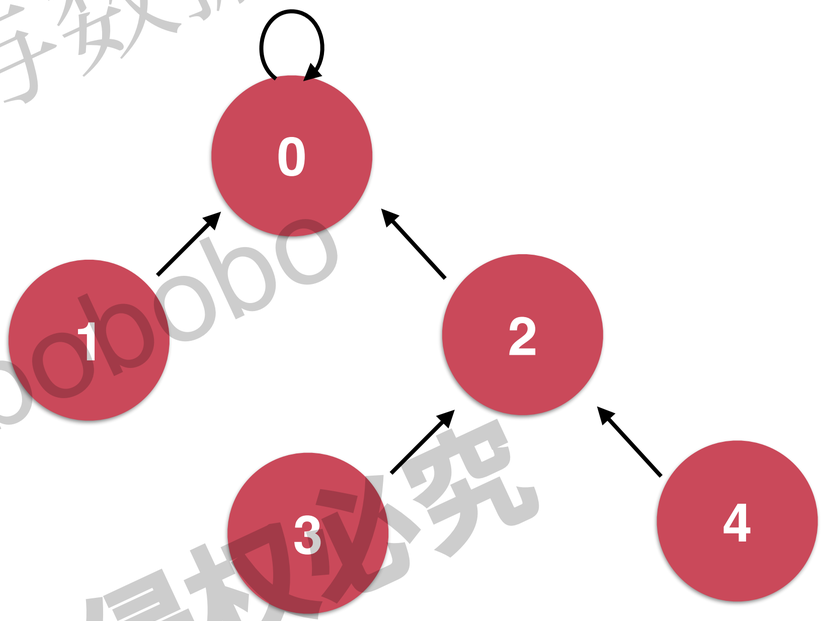
	0	1	2	3	4
parent	0	0	1	2	2

find 4



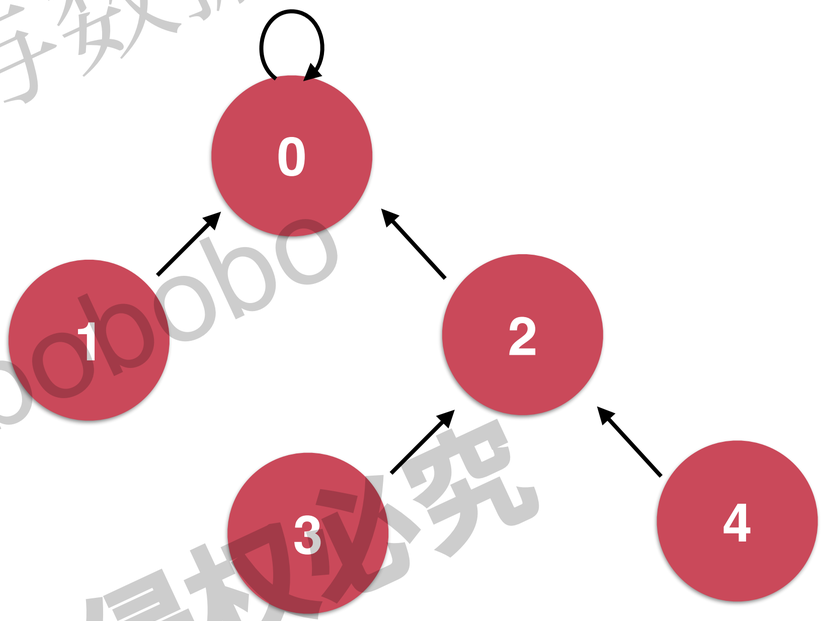
	0	1	2	3	4
parent	0	0	1	2	2

find 4



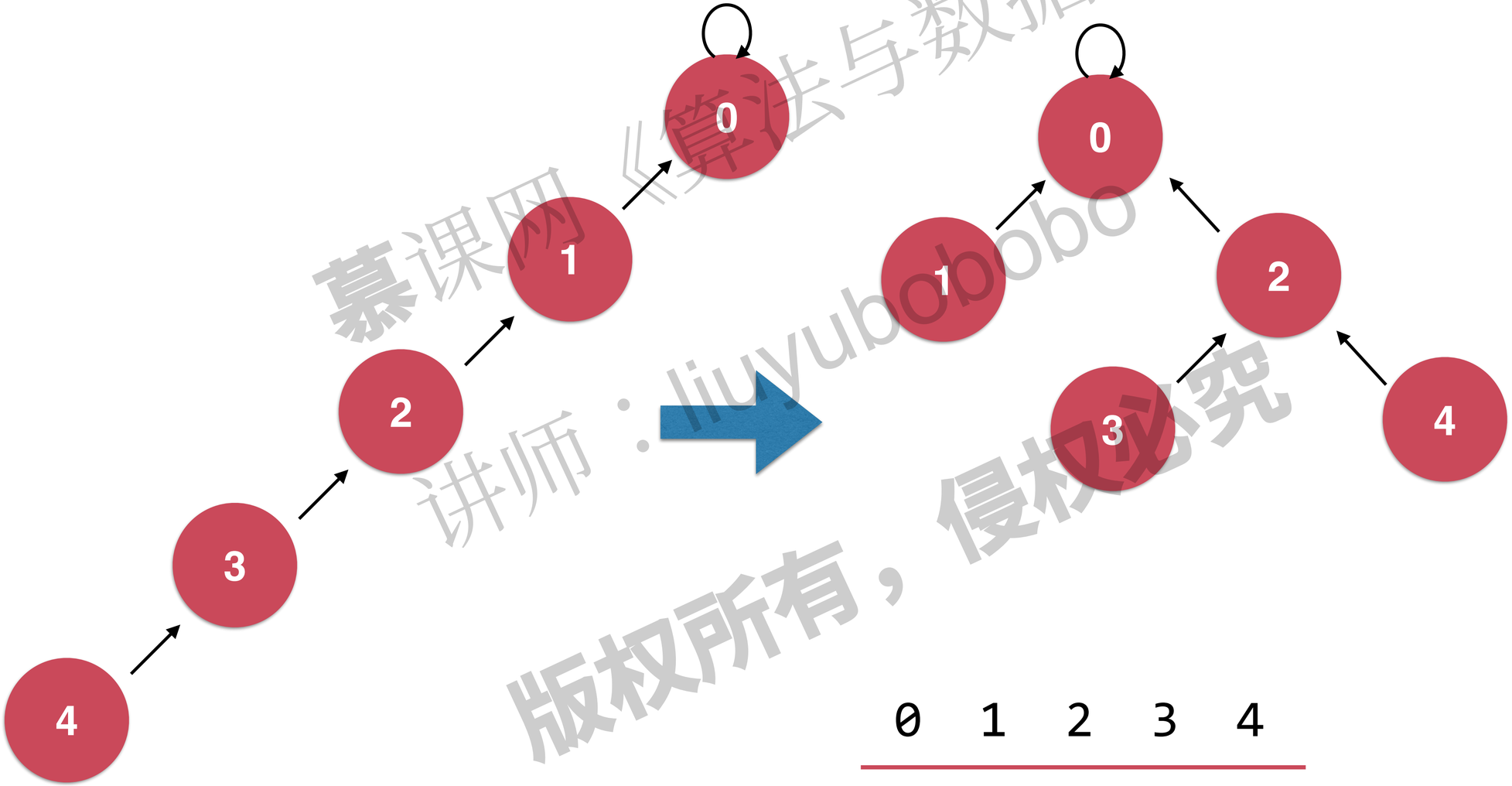
	0	1	2	3	4
parent	0	0	0	2	2

find 4



	0	1	2	3	4
parent	0	0	0	2	2

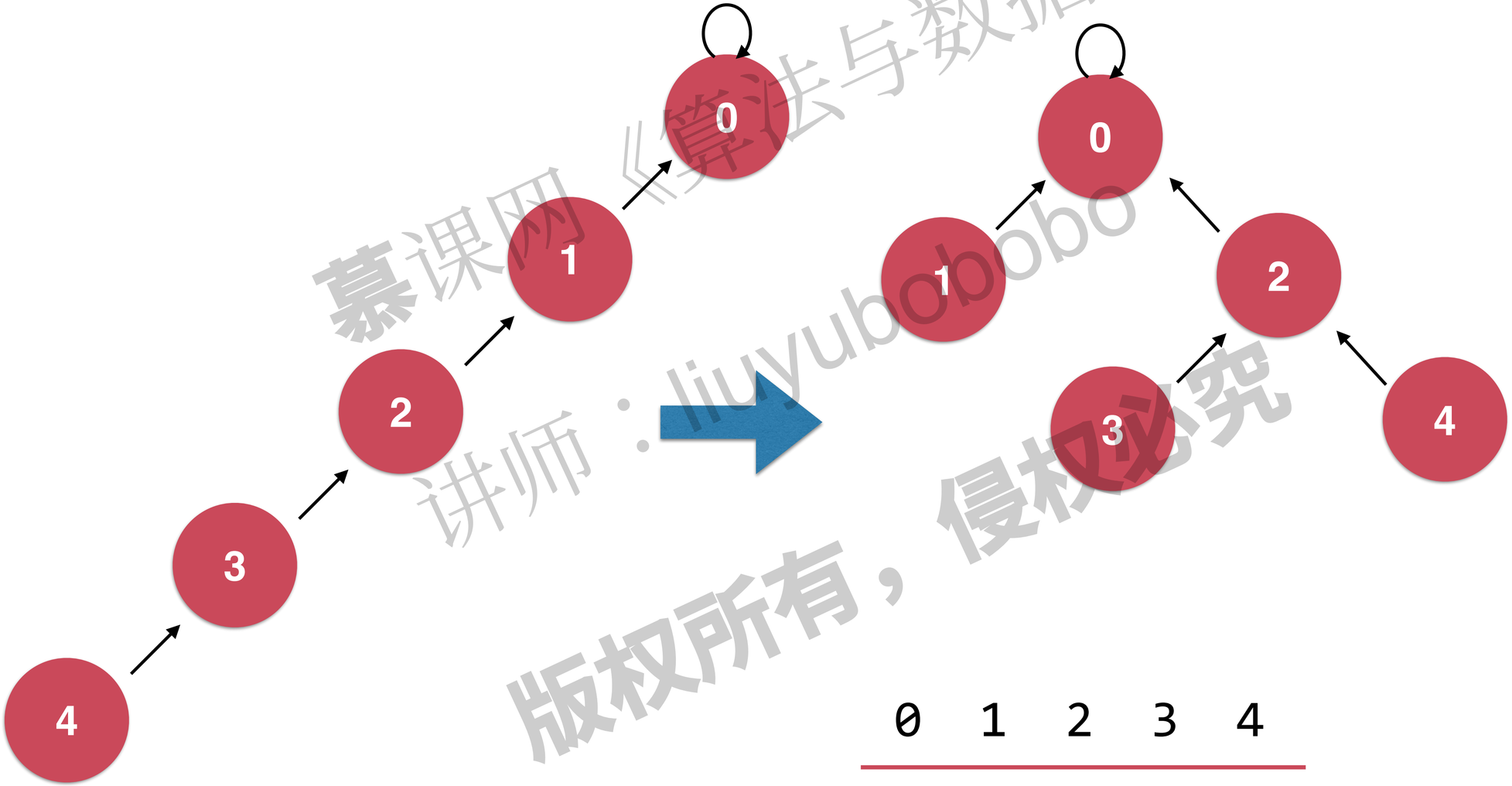
路径压缩 Path Compression



	0	1	2	3	4
parent	0	0	0	2	2

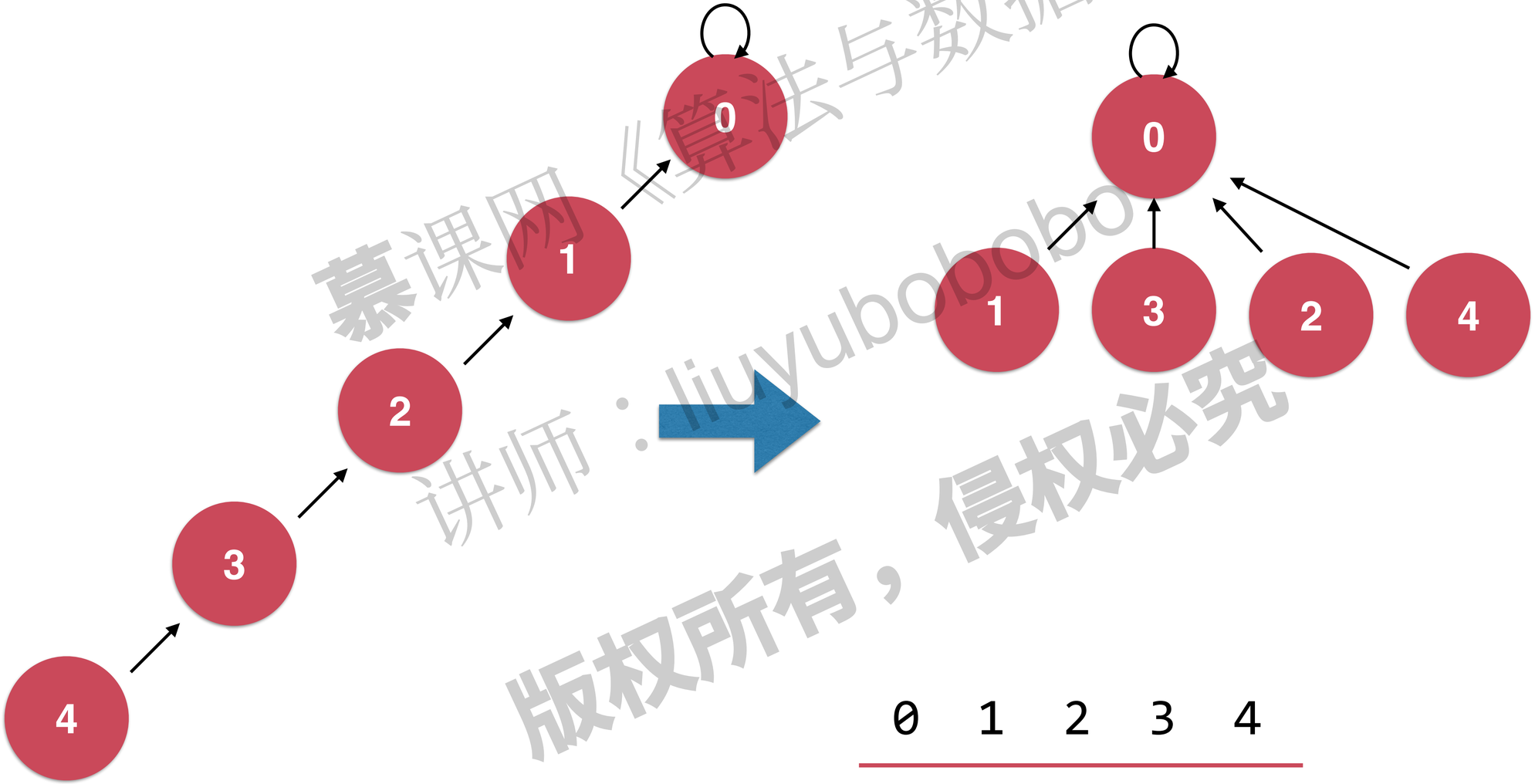
操作：路径压缩

路径压缩 Path Compression



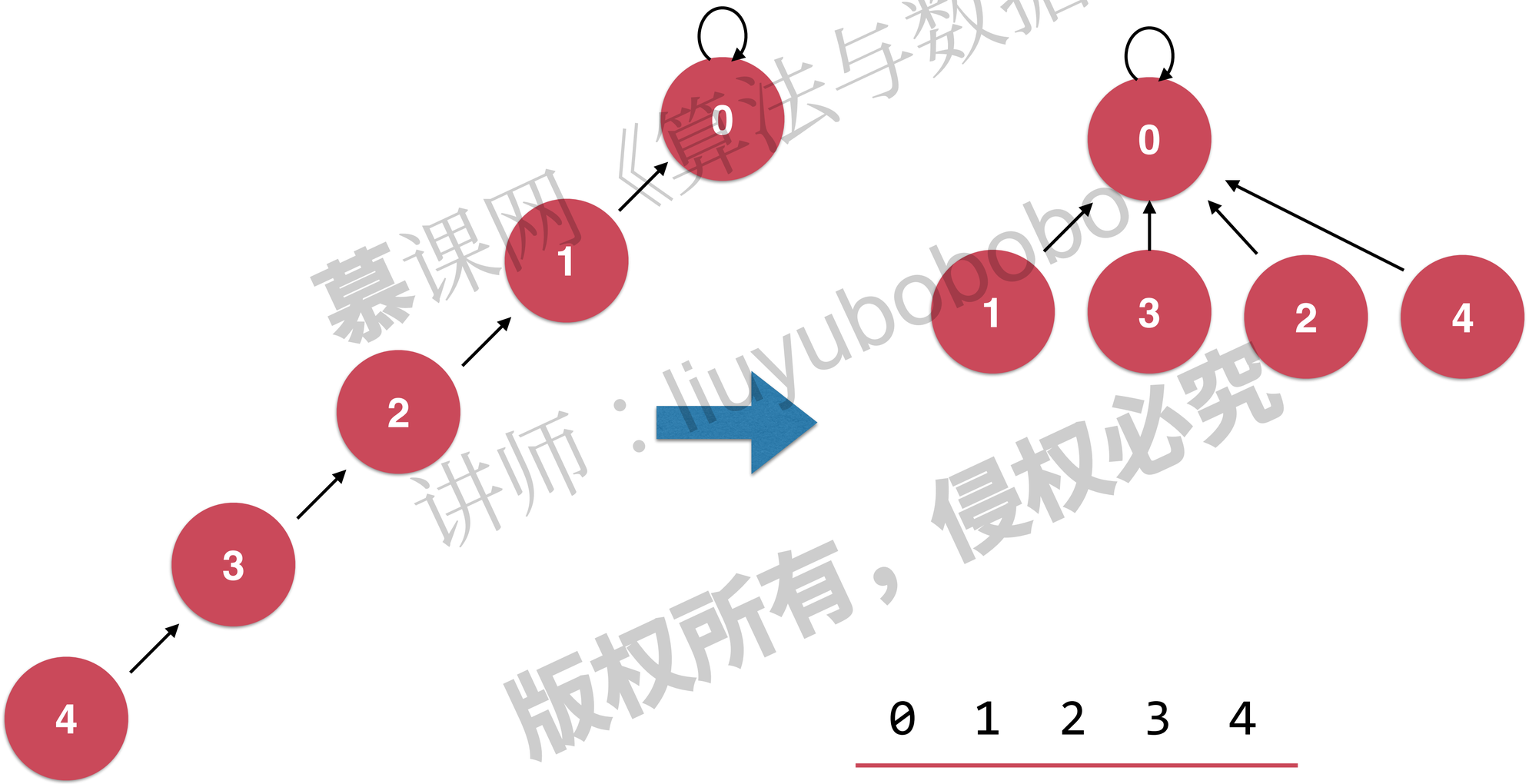
	0	1	2	3	4
parent	0	0	0	2	2

路径压缩 Path Compression



	0	1	2	3	4
parent	0	0	0	1	2

路径压缩 Path Compression



	0	1	2	3	4
parent	0	0	0	0	0

操作：递归的路径压缩

并查集的操作，时间复杂度近乎是O(1)的

其他

欢迎大家关注我的个人公众号：是不是很酷

算法与数据结构

liuyubobobo