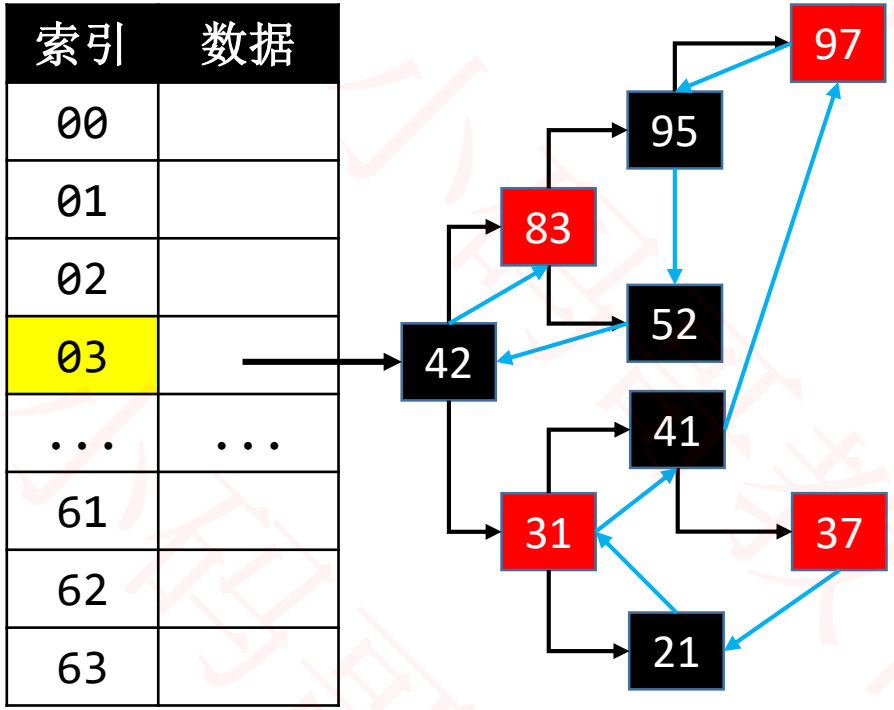
LinkedHashMap

- 在HashMap的基础上维护元素的添加顺序，使得遍历的结果是遵从添加顺序的



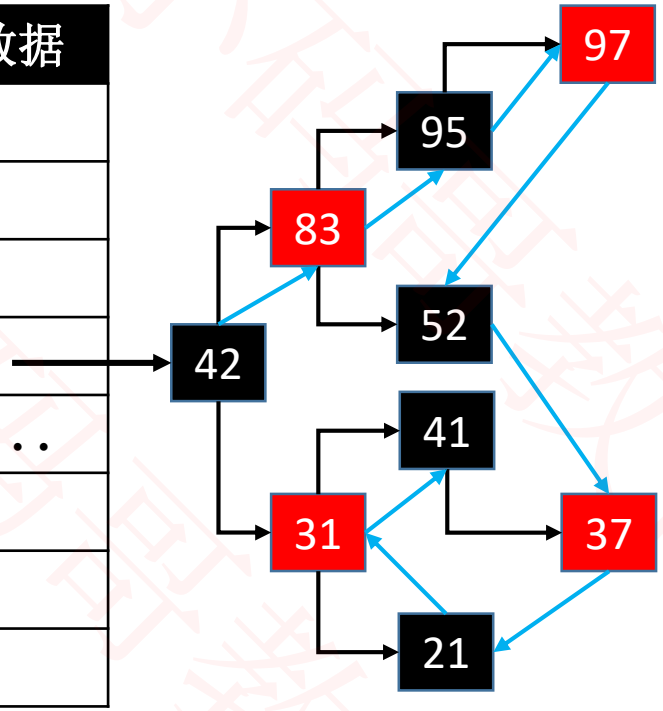
- 假设添加顺序是
  - 37、21、31、41、97、95、52、42、83

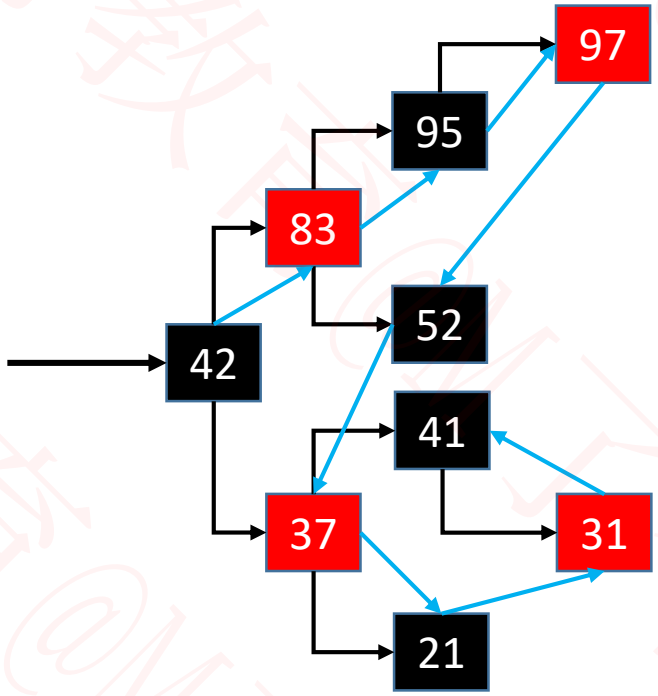
- 删除度为2的节点node时
  - 需要注意更换 node 与 前驱\后继节点 的连接位置

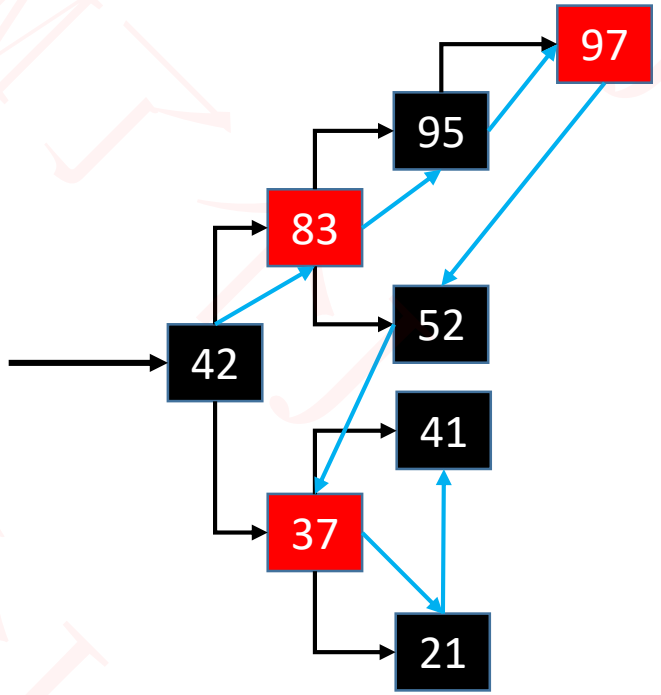
LinkedHashMap – 删除注意点

- 删除度为2的节点node时（比如删除31）
  - 需要注意更换 node 与 前驱\后继节点 的连接位置

索引	数据
00	
01	
02	
03	
...	...
61	
62	
63	







LinkedHashMap – 更换节点的连接位置


```
// 交换prev
LinkedNode<K, V> tmp = node1.prev;
node1.prev = node2.prev;
node2.prev = tmp;
if (node1.prev != null) {
    node1.prev.next = node1;
} else {
    first = node1;
}
if (node2.prev != null) {
    node2.prev.next = node2;
} else {
    first = node2;
}
```

```
// 交换next
tmp = node1.next;
node1.next = node2.next;
node2.next = tmp;
if (node1.next != null) {
    node1.next.prev = node1;
} else {
    last = node1;
}
if (node2.next != null) {
    node2.next.prev = node2;
} else {
    last = node2;
}
```