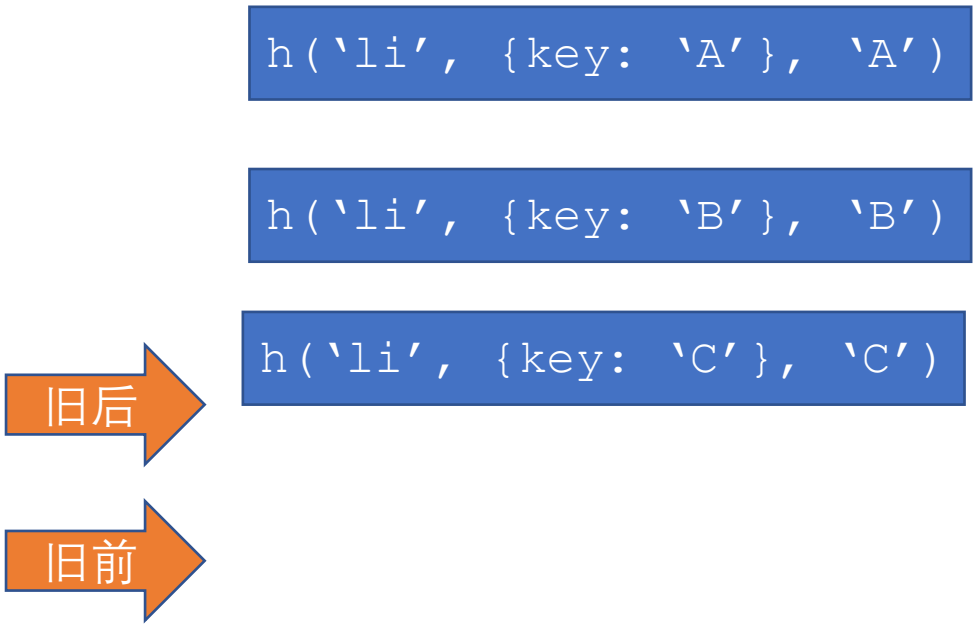
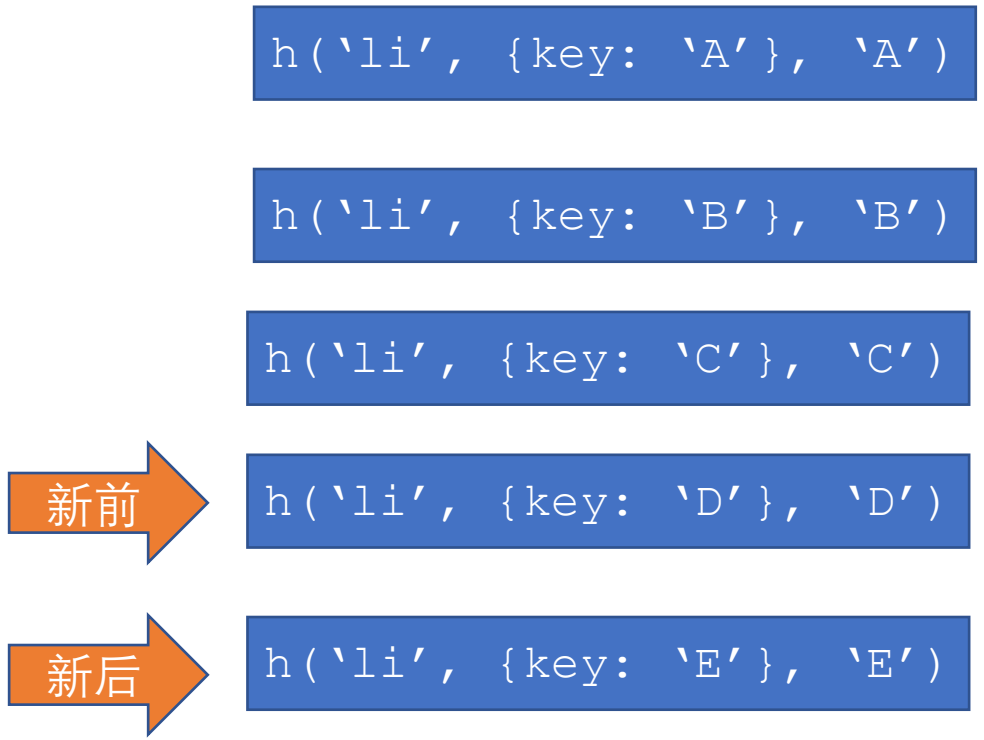
新增的情况　　旧子节点　　新子节点

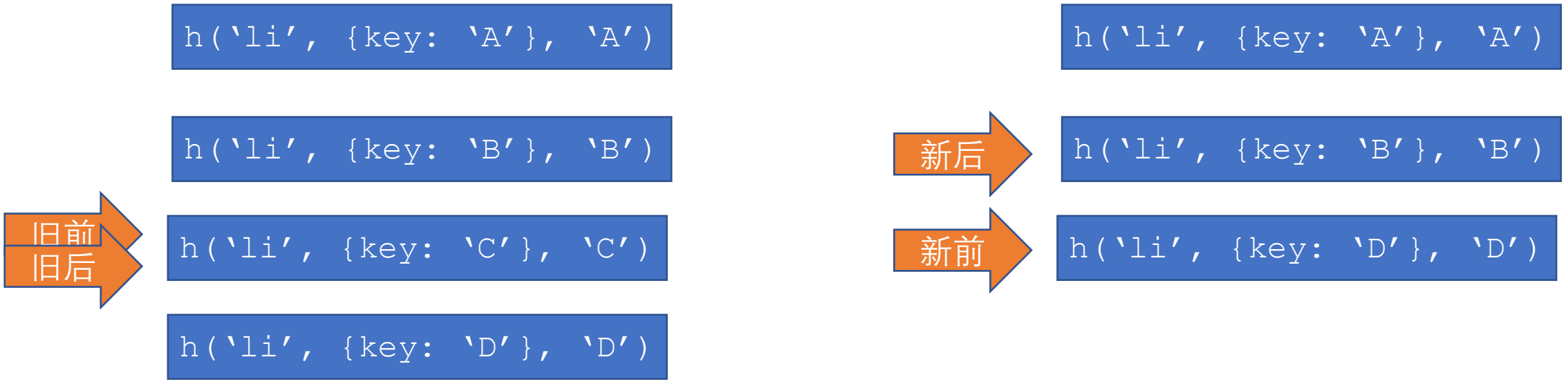




While(新前<=新后 && 旧前<=旧后)

如果旧的节点先循环完毕，说明新节点中有要插入的节点

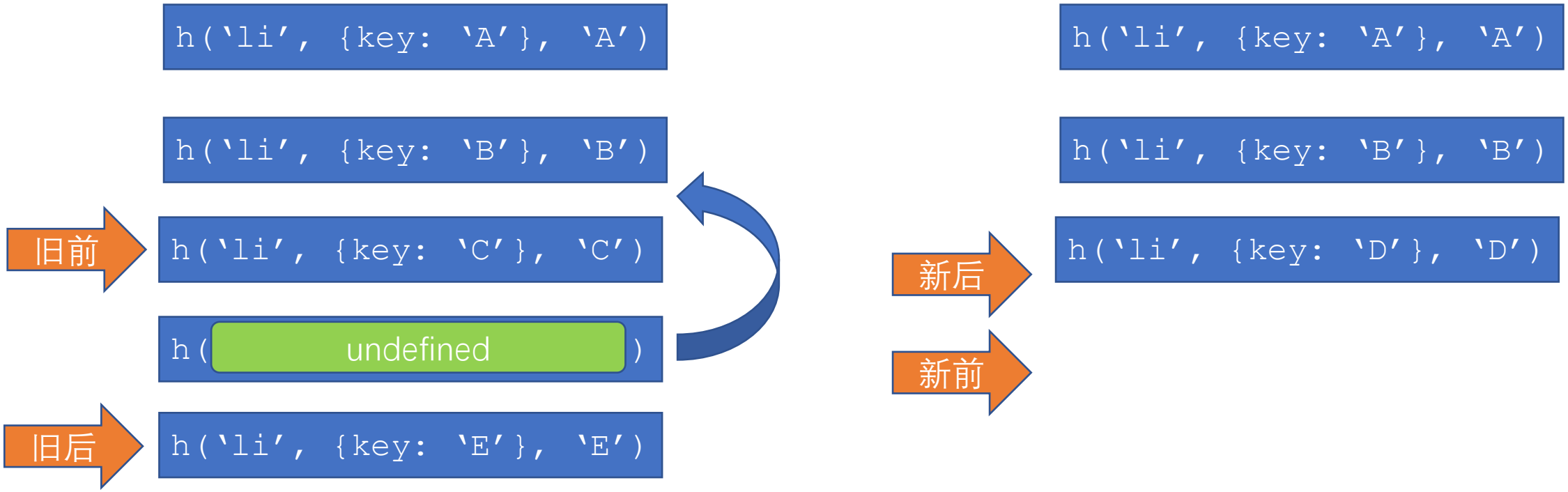
删除的情况　　旧子节点　　新子节点



新前 – 旧前
新后 – 旧后
新后 – 旧前
新前 – 旧后

如果新节点先循环完毕，如果老节点中还有剩余节点，就说明它们是要被删除的节点

多删除的情况　　旧子节点　　新子节点



新前 – 旧前
新后 – 旧后
新后 – 旧前
新前 – 旧后

如果都没有命中，循环来寻找（循环旧节点）
如果找到了对应节点，说明该节点发生了位置移动，真实DOM会移动到旧前之前，虚拟DOM会打个undefined标记

如果新节点先循环完毕，如果老节点中还有剩余节点（旧前与旧后中间的节点），就说明它们是要被删除的节点

复杂的情况

旧子节点

新子节点

E

C

M

旧前

h('li', {key: 'A'}, 'A')

h('li', {key: 'B'}, 'B')

h('li', {key: 'C'}, 'C')

Undefined
移动位置旧前之前

旧后

h('li', {key: 'D'}, 'D')

h('li', {key: 'E'}, 'E')

undefined

新前

新后

h('li', {key: 'E'}, 'E')

h('li', {key: 'C'}, 'C')

h('li', {key: 'M'}, 'M')

查找M没有找到，插入M到旧前之前

新前 – 旧前
新后 – 旧后
新后 – 旧前
新前 – 旧后

如果都没有命中，循环来寻找
（循环旧节点）

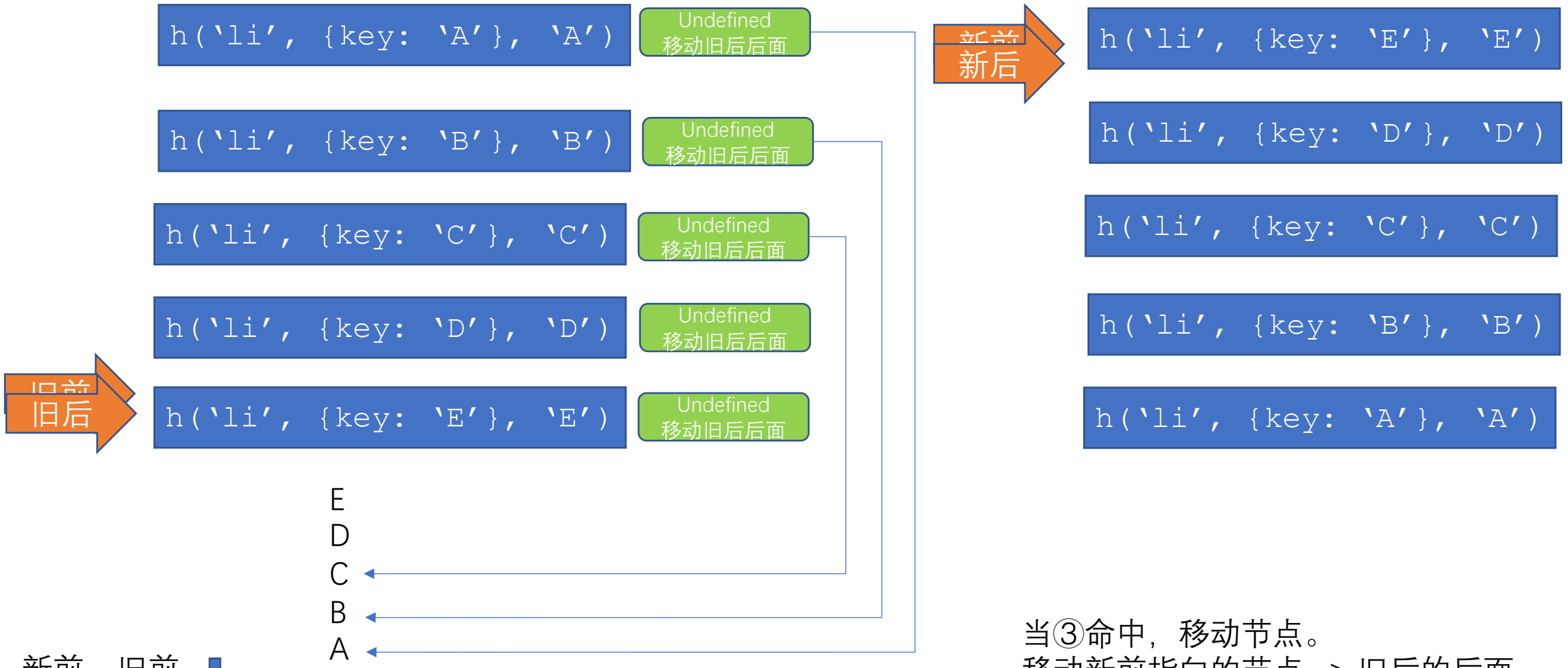
哪对命中，移动那对对应指针

当④命中，移动节点。
移动新前指向的节点 -> 旧前的前面

复杂的情况

旧子节点

新子节点



新前 – 旧前
新后 – 旧后
新后 – 旧前
新前 – 旧后

如果都没有命中，循环来寻找
（循环旧节点）

哪对命中，移动那对对应指针

当③命中，移动节点。
移动新前指向的节点 -> 旧后的后面