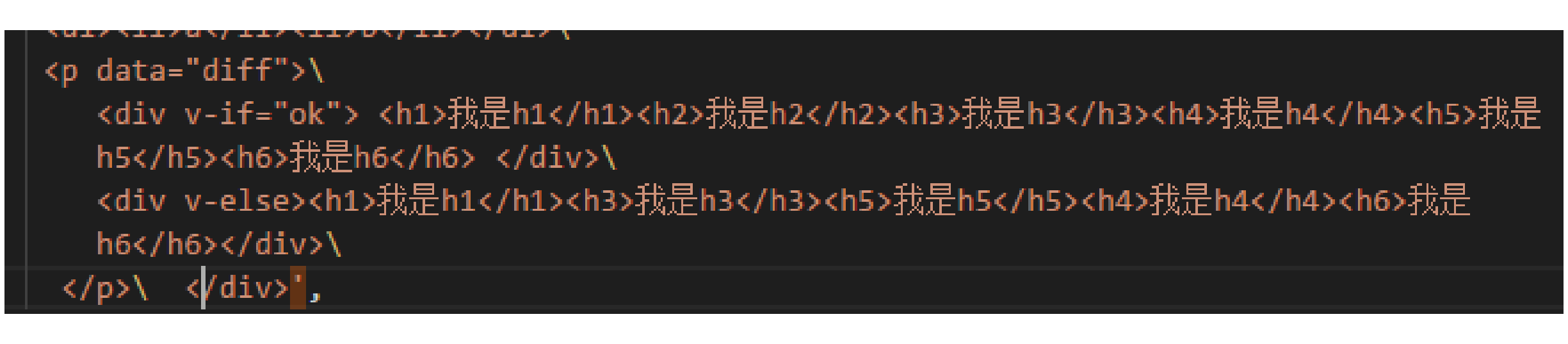
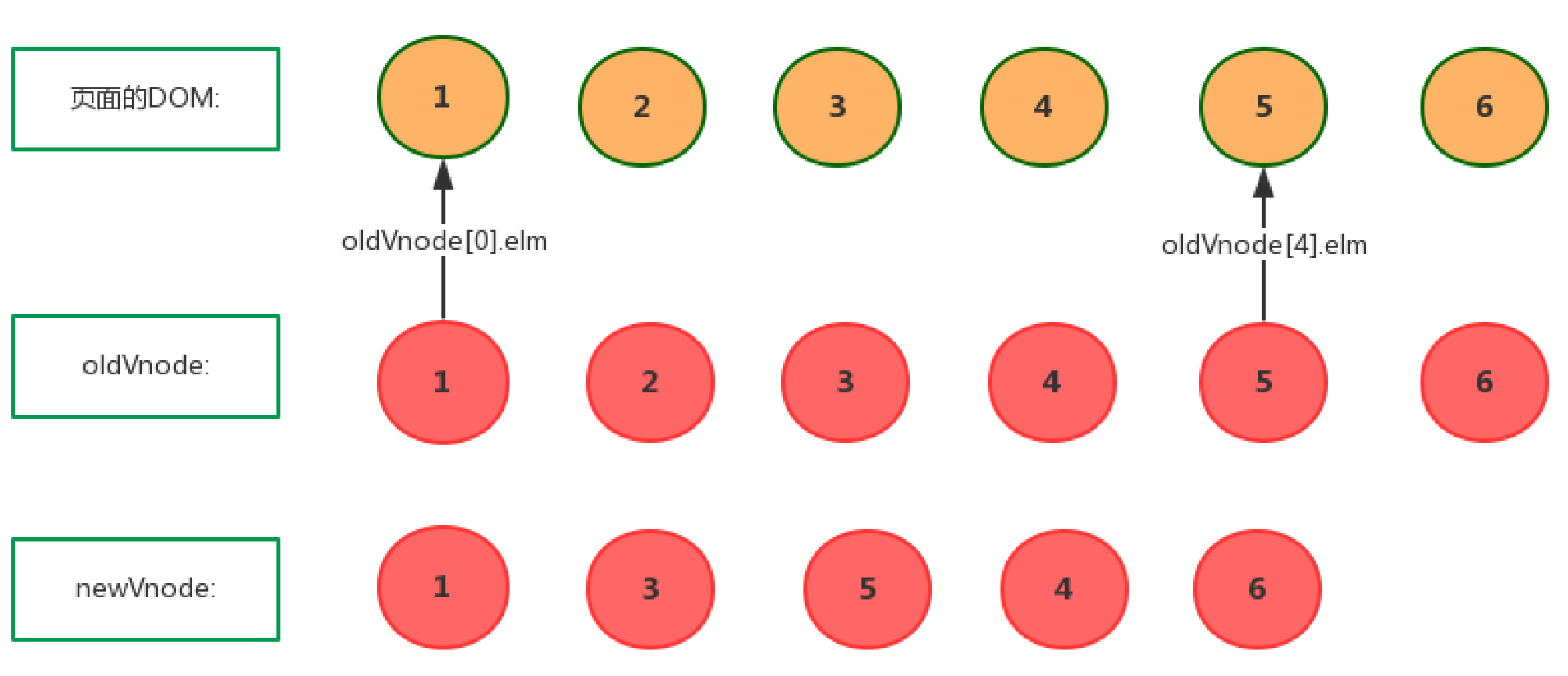


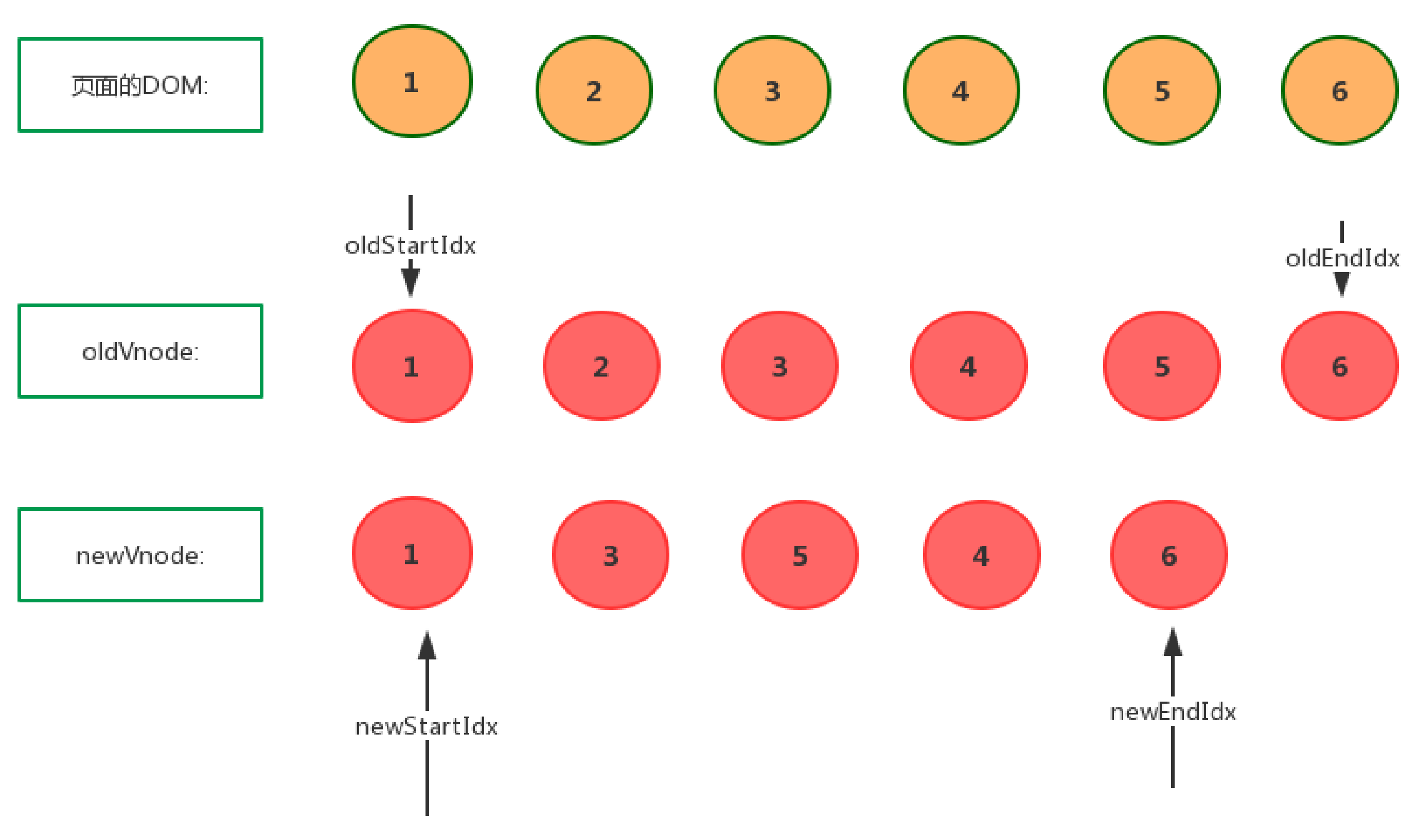
```
<p data="diff">\
   <div v-if="ok"> <h1>我是h1</h1><h2>我是h2</h2><h3>我是h3</h3><h4>我是h4</h4><h5>我是
   h5</h5><h6>我是h6</h6> </div>\
   <div v-else><h1>我是h1</h1><h3>我是h3</h3><h5>我是h5</h5><h4>我是h4</h4><h6>我是
   h6</h6></div>\
 </p>\  </div>',
```

如上图，点击change，改变 ok=false，进入v-else，就会触发 v-if 节点列表 和 v-else 列表的diff

代码转换成图片 精简如下

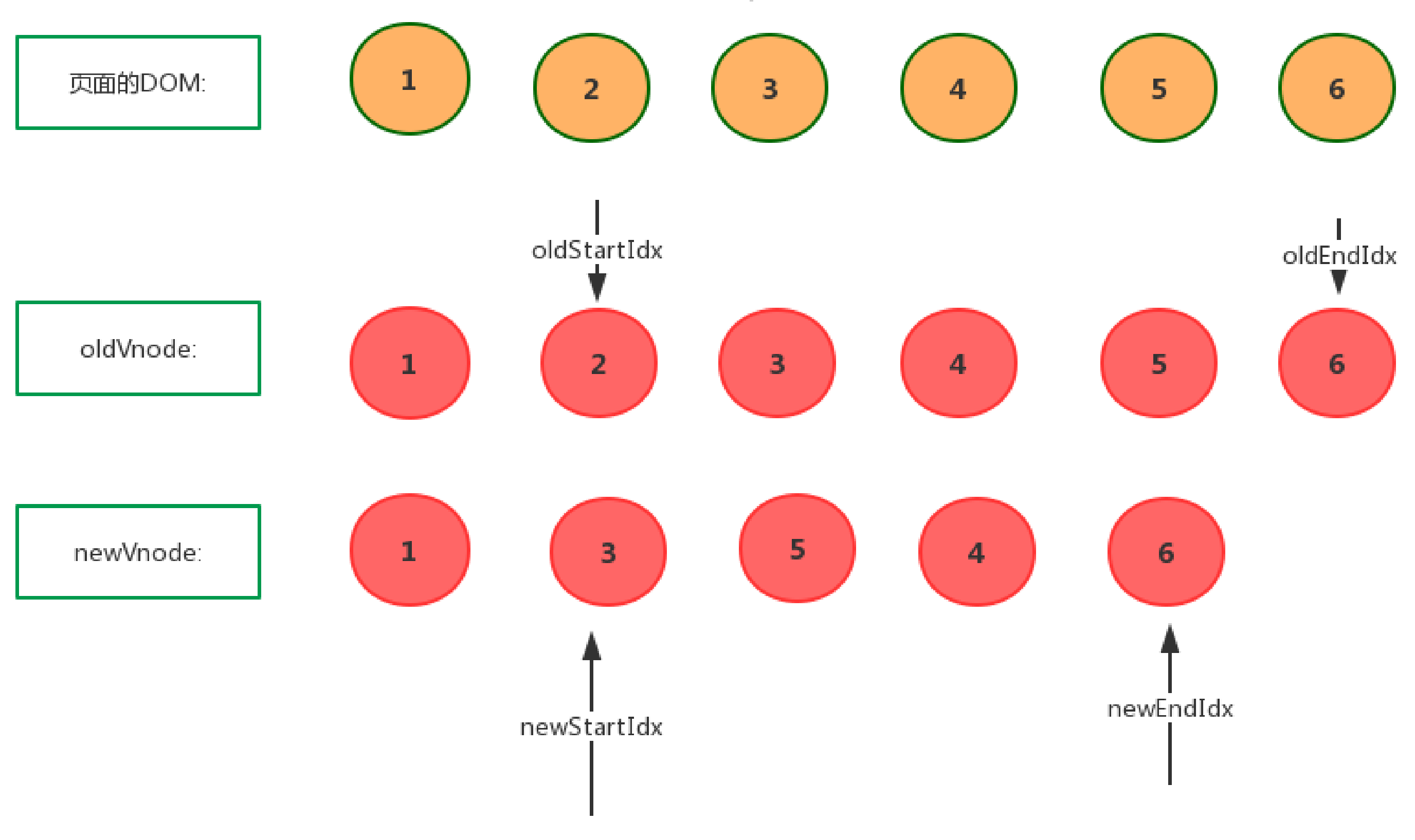


第一次循环



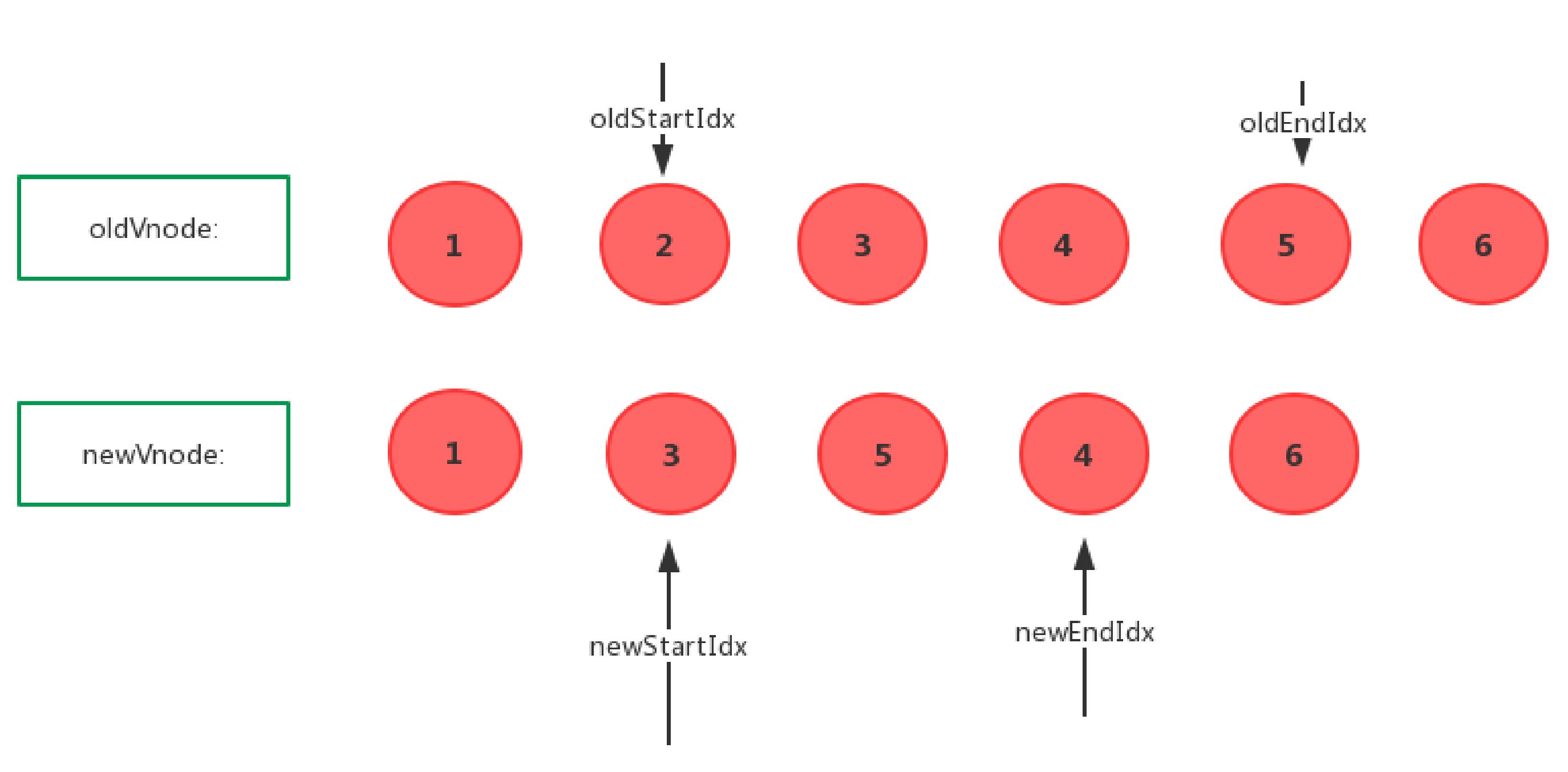
开始定义了两对 指针 分别指向新旧节点 首尾
进入第一次对比，发现 新旧列表的第一个节点相同，都是 h1
则，不需要操作dom，oldStartIdx++; newStartIdx++; 都向右移一位

变成下图



进入第 二 次对比，发现 新旧列表的 最后一个 节点相同，都是 h6
则，不需要操作dom，oldEndIdx++; newEndIdx++; 都向左移一位，向中间靠拢

变成下图



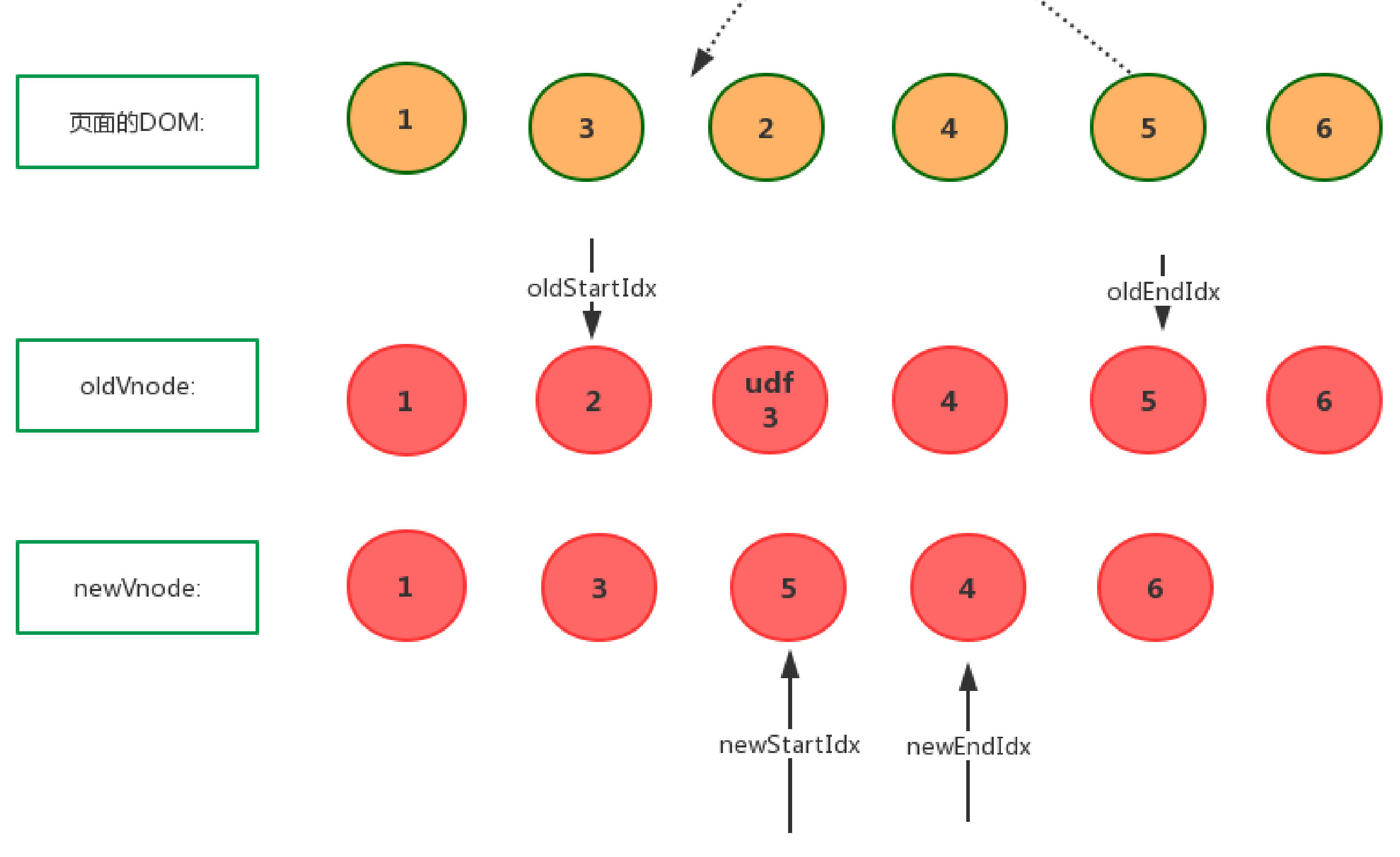
进入第 三 次对比， 首尾都不相同，只能 看oldVnode有没有3这个节点，

发现 oldStartIdx 在 2 的位置有 存在 节点 3， 把 oldVnode[2].elm insertbefor 到

oldVnode[oldStartIdx] 的前面， 也就是 elm[2] 放在 elm[1]前面

newStartIdx 向右移动一位，oldStartIdx 位置不变，但要把 oldVnode[2] 变成 undefined

变成下图

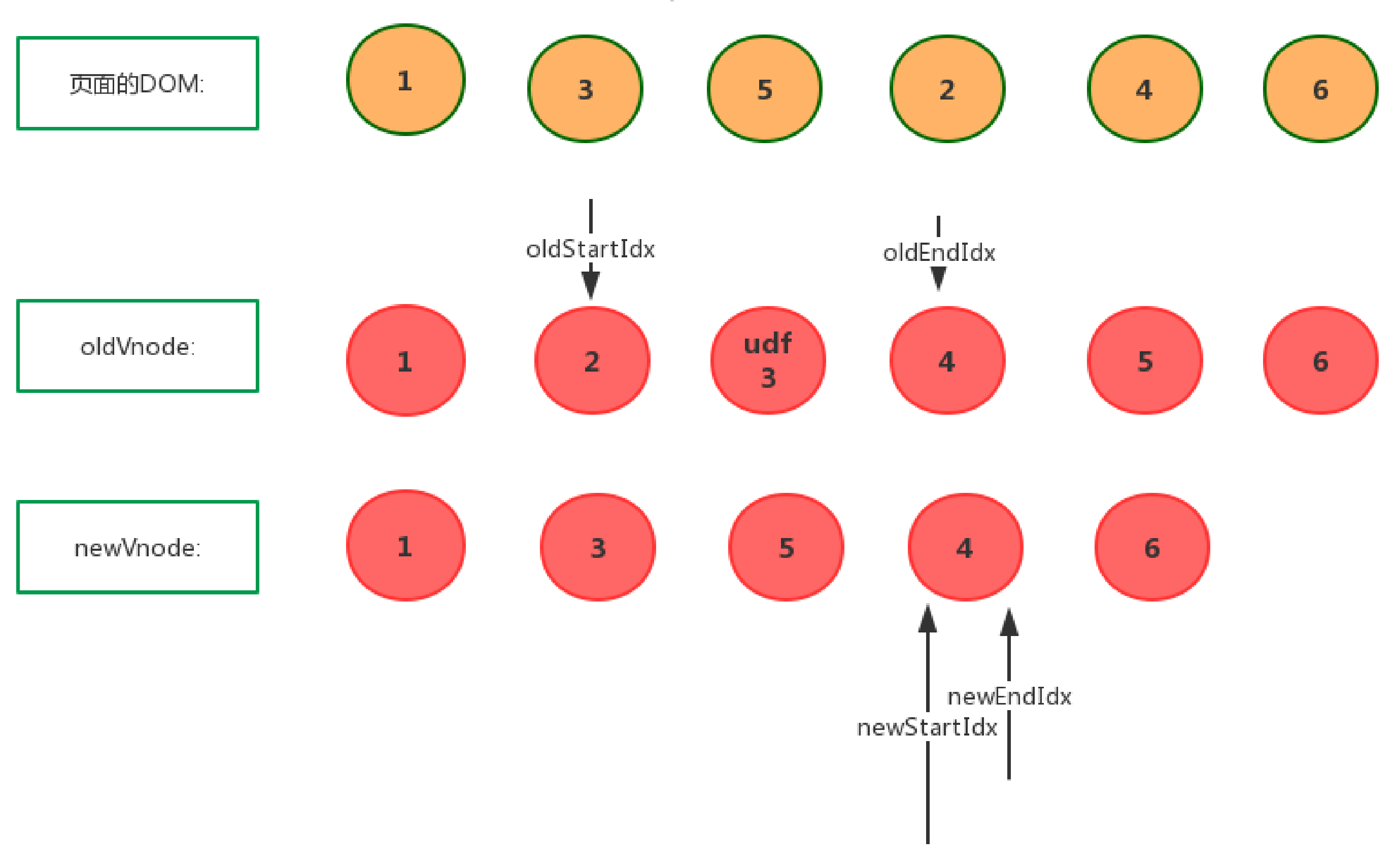


进入第 四 次对比， 尾首相同，都是h5，继续把 oldVnode[oldEndIdx].elm 移动到 oldVnode[oldStartIdx]前面，

如上图，虽然oldvnode[oldStartIdx]位置没有变，但是oldvnode[oldStartIdx].elm的位置已经变化了，不断向前

newStartIdx向右移动一位，跟 newEndIdx重合，oldEndidx相左移动一位

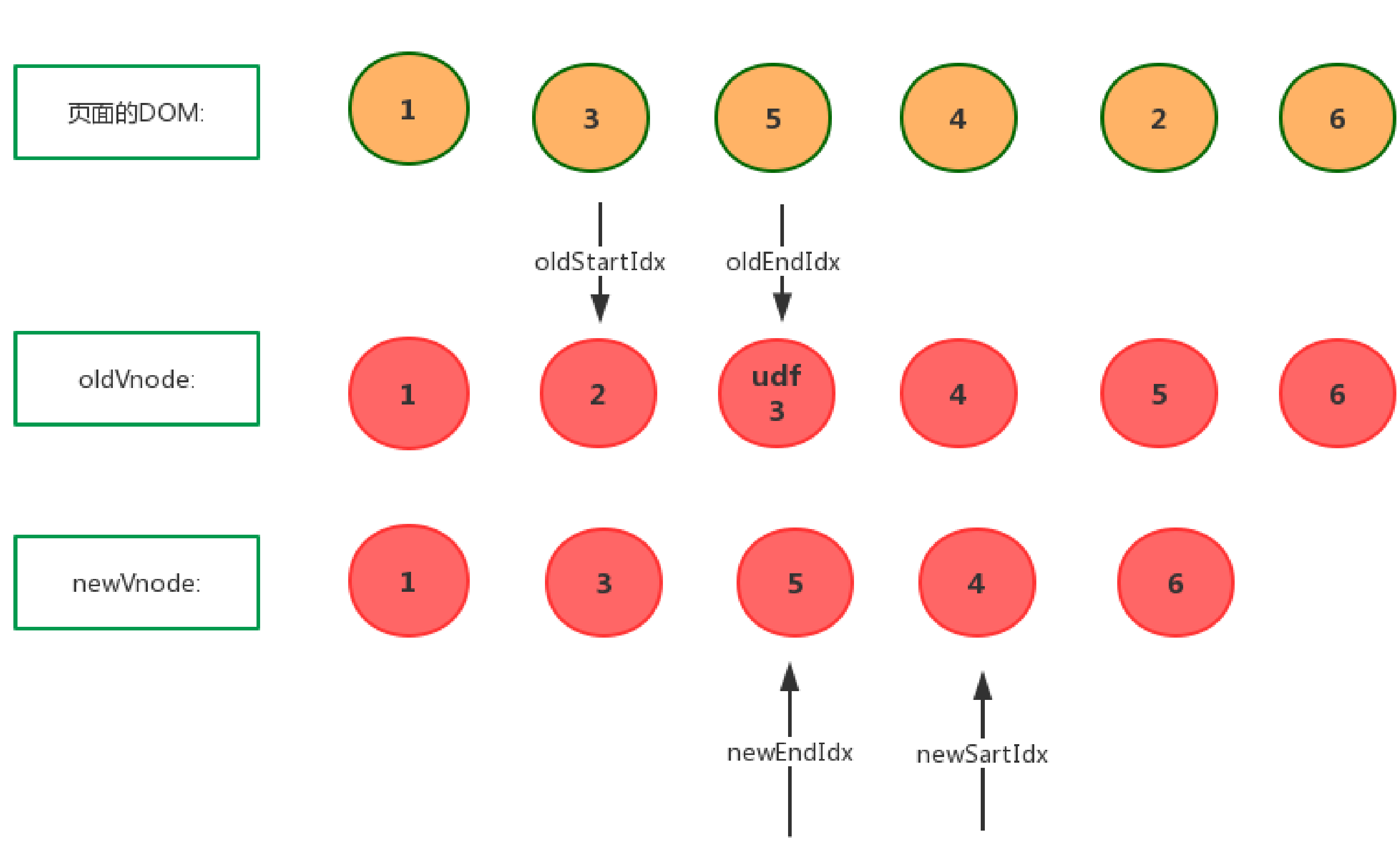
变成下图



进入第 五 次对比， 还是尾尾相同，都是h4，

newEndIdx，oldEndIdx都向左移动一位， newEndIdx 越过了 newStartIdx，--> 循环结束

变成下图



进入这个条件 else if (newStartIdx > newEndIdx) {
removeVnodes(parentElm, oldCh, oldStartIdx, oldEndIdx);
}

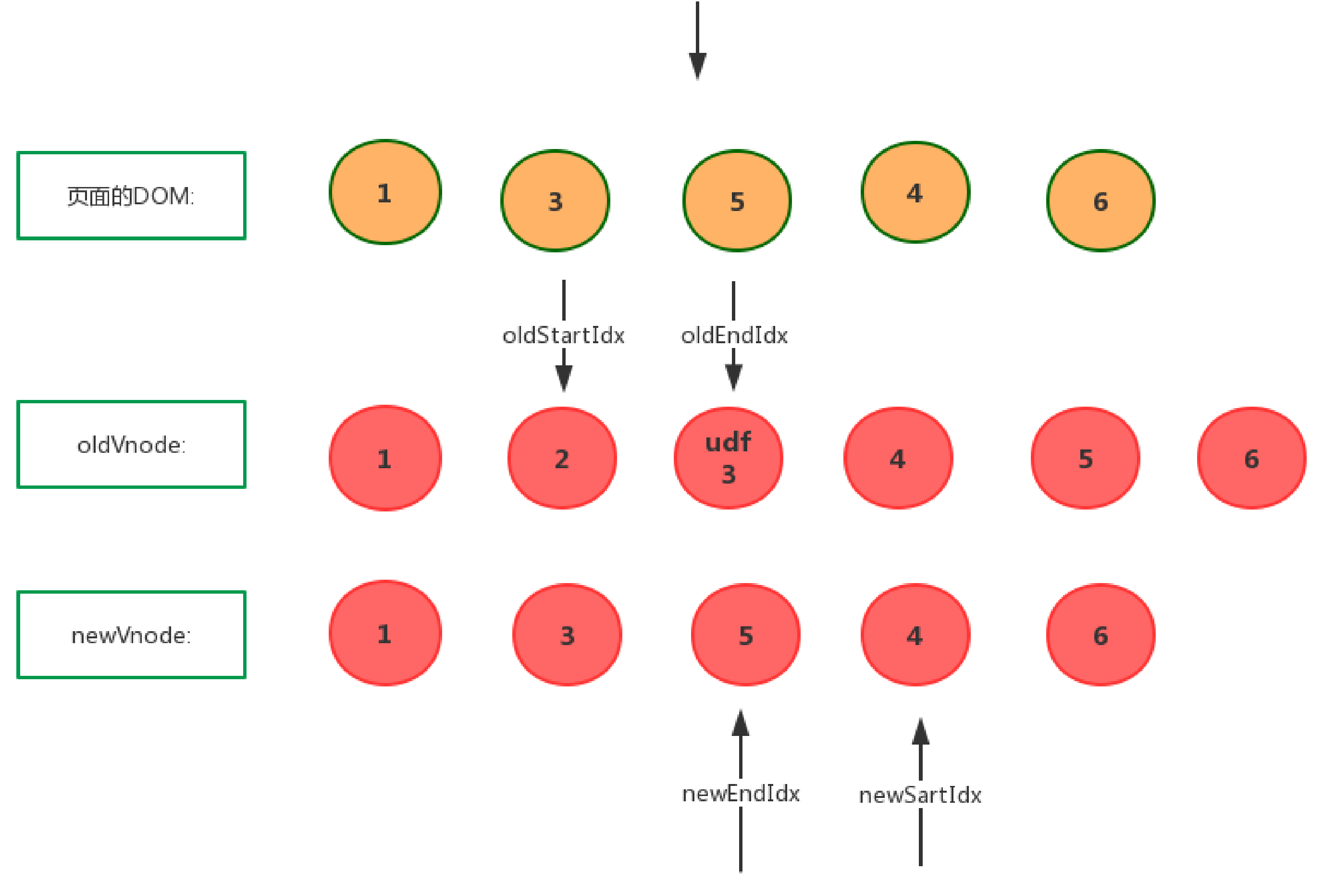
新节点都处理完成，没有新添加的节点了，这时候，只需要删除 oldStartIdx 到 oldEndIdx 直接的旧节点即可

也就是删除 oldVnode[1].elm 和 oldVnode[2].elm这两个节点，

但oldVnode[2].elm 这个节点3实际上是不能删除的，因为它已经移动到恰当的位置了，

所以 oldVnode[2] 上面被标记成 undefined， oldVnode[2].elm找不到了，也就不会被误删.

最后变成



可以看到，把oldstartIdx对应的 elm 2删除之后， 页面的dom 正好是 newVnode的节点，

而且这时候他们也是一一对应的关系了.

也就是 newVnode[0].elm 对应页面对一个节点 H1

diff完成