Sunday • 更新于 2022-10-19

上一节 10：框架实现：... 12：前置知识：... 下一节

11：局部总结：前四种 diff 场景的总结与乱序场景

那么到目前为止，我们已经完成了 4 种 `diff` 场景的对应处理，经过前面的学习我们可以知道，对于前四种 `diff` 场景而言， `diff` 的处理本质上是比较简单的：

1. 自前向后的 `diff` 对比：主要处理从前到后的相同 `VNode` 。例如： `(a b) c` 对应 `(a b) d e`
2. 自后向前的 `diff` 对比：主要处理从后到前的相同 `VNode` 。例如： `a (b c)` 对应 `d e (b c)`
3. 新节点多余旧节点的 `diff` 对比：主要处理新增节点。
4. 旧节点多余新节点的 `diff` 对比：主要处理删除节点。

但是仅靠前四种场景的话，那么是无法满足实际开发中的所有更新逻辑的。所以我们还需要最关键的一种场景需要处理，那就是 **乱序场景**。

那么什么情况下我们需要乱序场景呢?

我们来看以下的 `diff` 场景：

> 详见： `PPT - 图谱`

那么经过以上的 `diff` 处理之后，我们就只剩下中间三个节点的对应处理，那么中间三个的 `diff` 处理逻辑就是最后一种场景 **乱序** 的处理逻辑。

那么这种乱序具体是怎么进行的呢？我们继续来往下看~~

10：框架实现：场景四：旧节点多于新节点... 上一节 下一节 12：前置知识：场景五：最长递增子序列

我要提出意见反馈

索引目录

11：局部总结：前