性能优化方式（一）

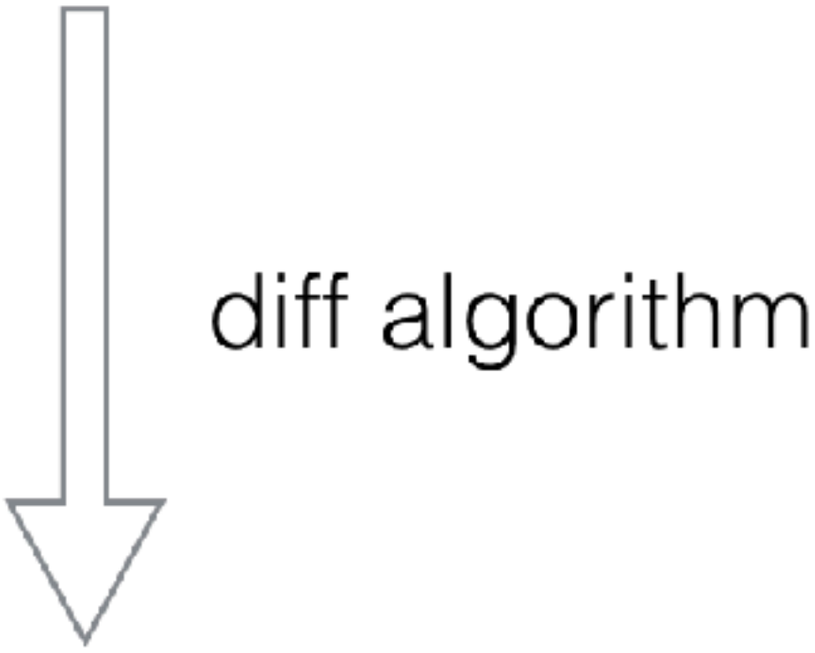
luckyadam

Nerv

virtual dom

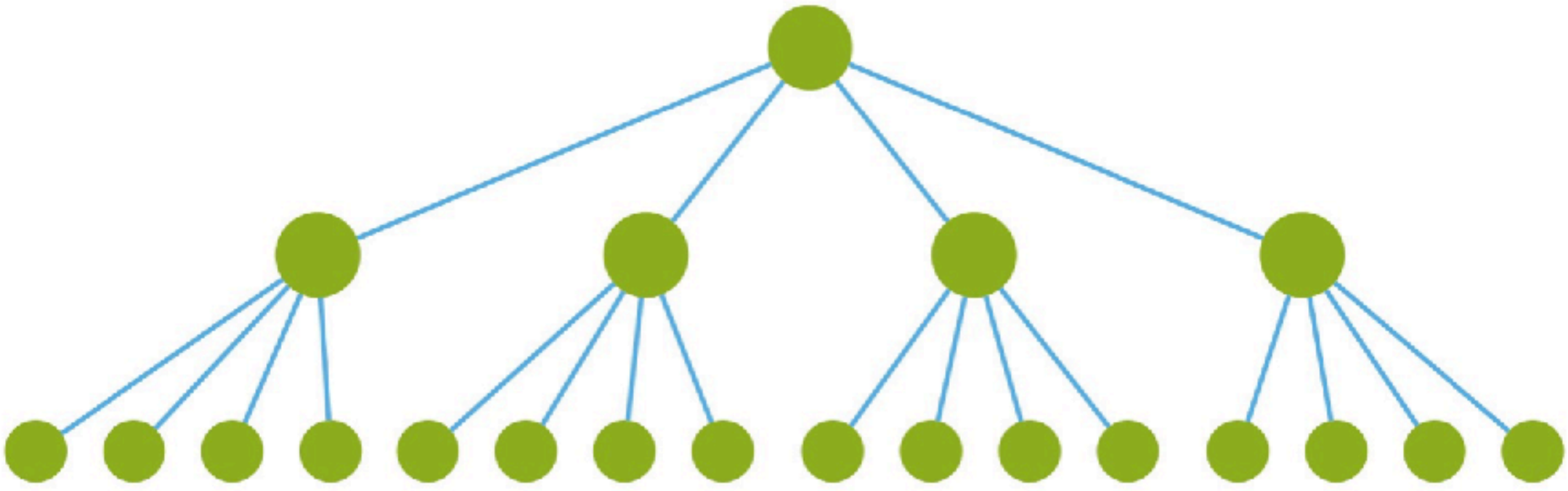
render(props, state) => dom

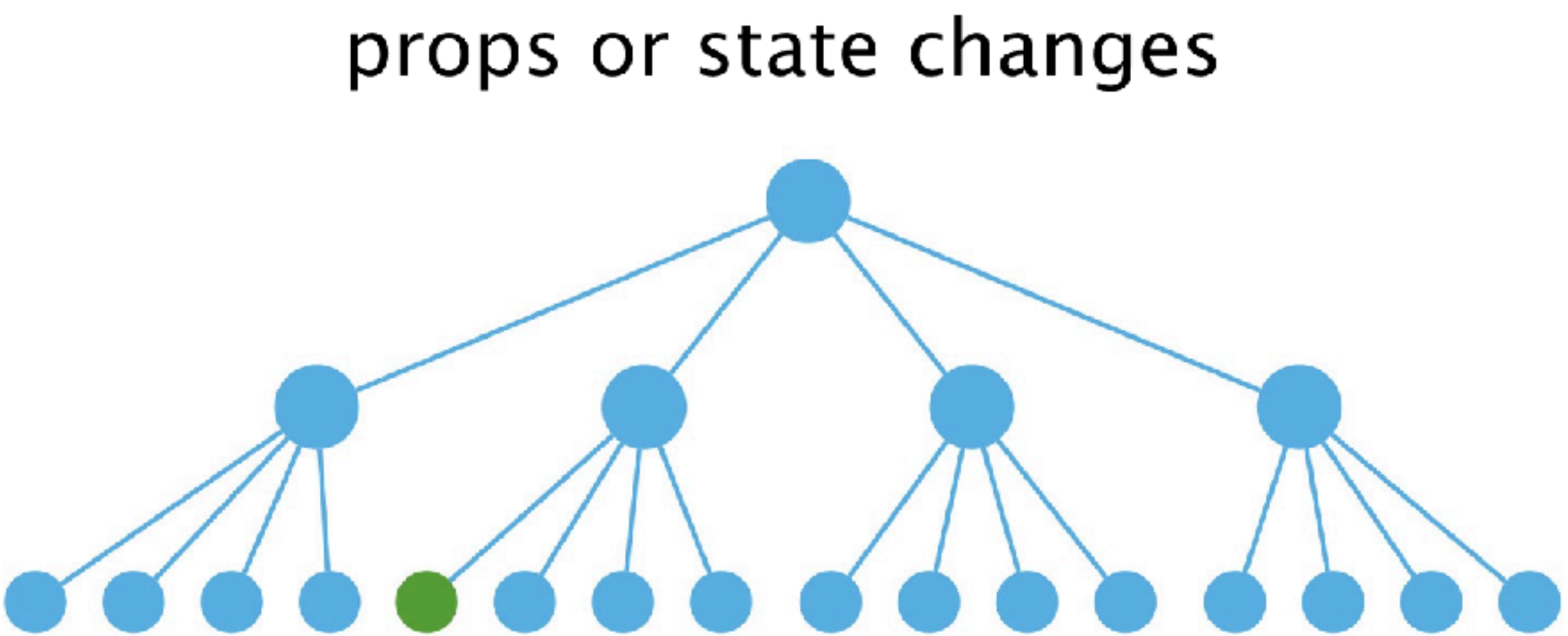
props or state changes

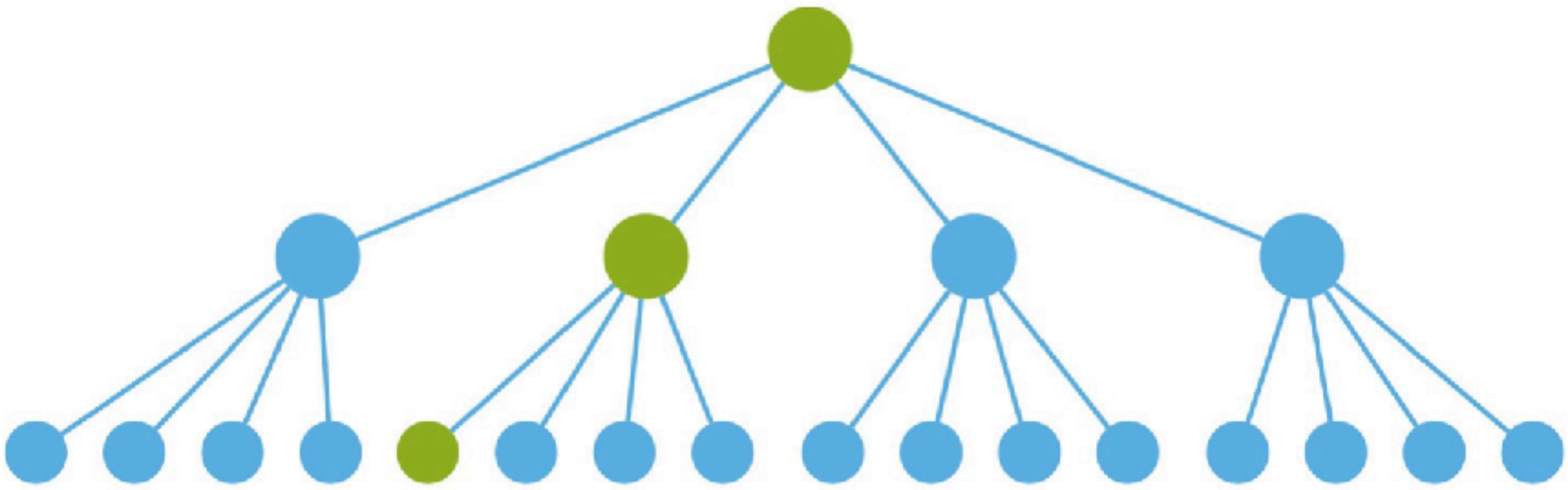


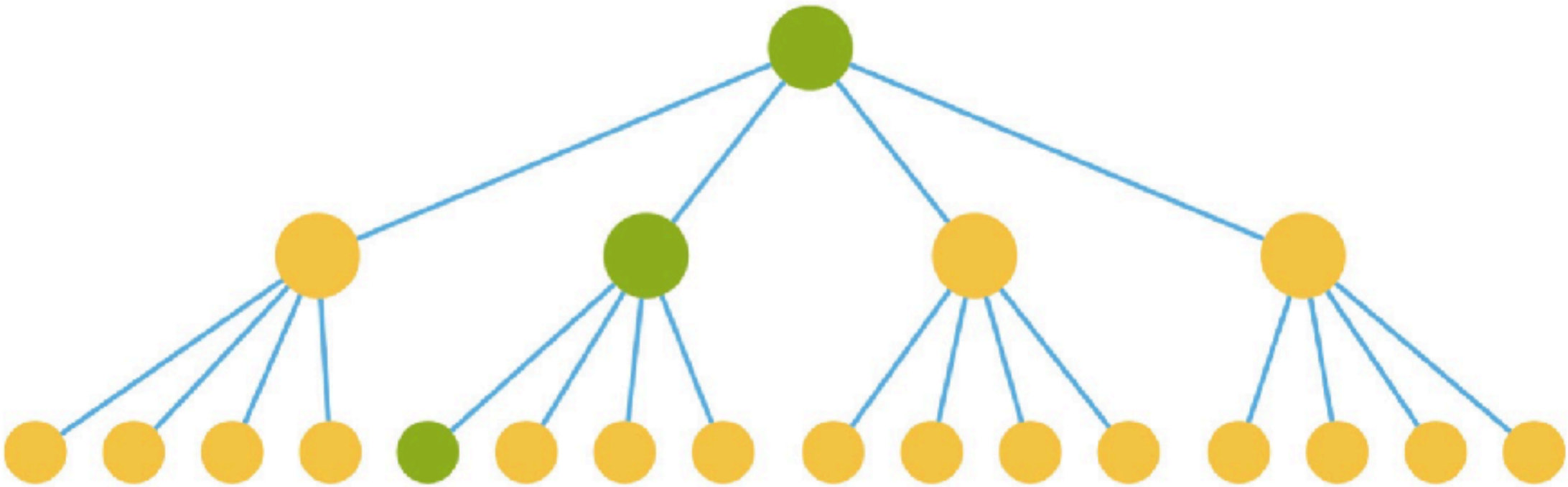
re-rendering an entire subtree of components

性能损耗！




props or state changes





如何优化?

1 shouldComponentUpdate

props or state changes

shouldComponentUpdate => true

diff algorithm

re-rendering an entire subtree of components

```
shouldComponentUpdate (nextProps, nextState) {
    if (nextProps.currentIndex !== this.props.currentIndex ||
      nextProps.count !== this.props.count) {
      return true
    }
    return false
  }
```

对于必然不变的组件

可以直接将shouldComponentUpdate返回

false

2 PureComponent

如果说一个组件的渲染结果只和props、state有关系，它就是PureComponent

- 自带实现shouldComponentUpdate
- 在shouldComponentUpdate方法中对props和state做浅比较

3 Stateless Component

无状态组件，低开销

```
const SubComponent = (props) => {
  return <div className={props.className}>{props.count}</div>
}
```

4 其他小点

- 避免直接使用{...this.props} 传递属性
- 尽量避免在事件绑定时使用 this.onChangeHandler.bind(this) 方式，可以在constructor中使用赋值的方式
- 尽量不要总是触发setState
- 列表要给元素增加key属性
- 注意组件拆分