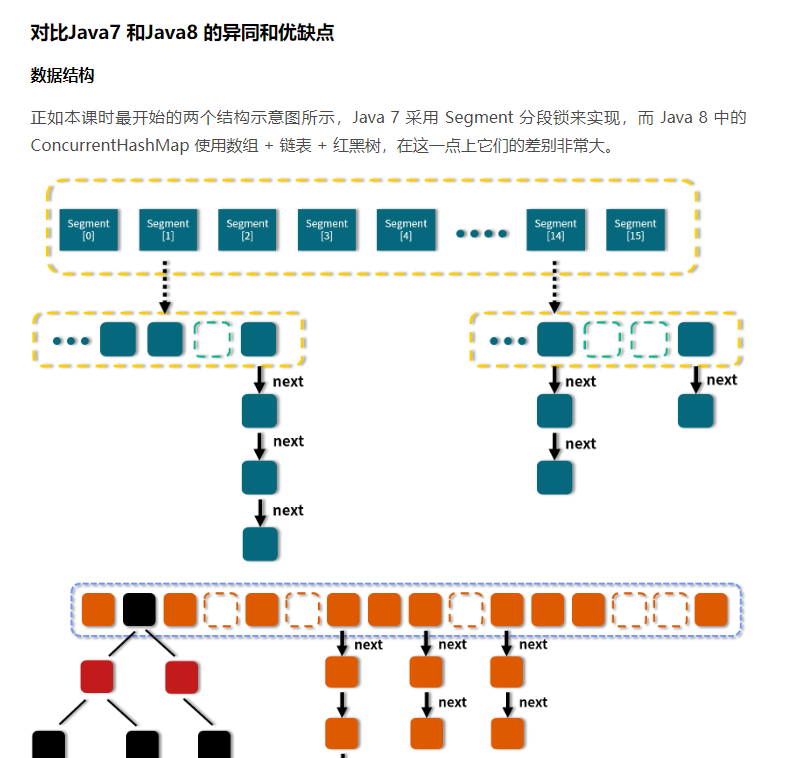
1. **HashMap 为什么是线程不安全的？**



1. **ConcurrentHashMap 在 Java7 和 8 有何不同？**

如图所示：





1. **为什么map桶中超过8个才转为红黑树？**

事实上，链表长度超过 8 就转为红黑树的设计，更多的是为了防止用户自己实现了不好的哈希算法时导致链表过长，从而导致查询效率低，而此时转为红黑树更多的是一种保底策略，用来保证极端情况下查询的效率。

通常如果 hash 算法正常的话，那么链表的长度也不会很长，那么红黑树也不会带来明显的查询时间上的优势，反而会增加空间负担。所以通常情况下，并没有必要转为红黑树，所以就选择了概率非常小，小于千万分之一概率，也就是长度为 8 的概率，把长度 8 作为转化的默认阈值。

所以如果平时开发中发现 HashMap 或是 ConcurrentHashMap 内部出现了红黑树的结构，这个时候往往就说明我们的哈希算法出了问题，需要留意是不是我们实现了效果不好的 hashCode 方法，并对此进行改进，以便减少冲突。

1. **CopyOnWriteArrayList特点**

* 将当前容器进行 Copy，复制出一个新的容器，然后修改新的容器，完成修改之后，再将原容器的引用指向新的容器
* 迭代期间允许修改集合内容

读写规则：读读共享、其他都互斥

缺点：内存占用问题、在元素较多或者复杂的情况下，复制的开销很大、数据一致性问题

适用场景：读操作可以尽可能的快，而写即使慢一些也没关系，读多写少