**HashMap分析**

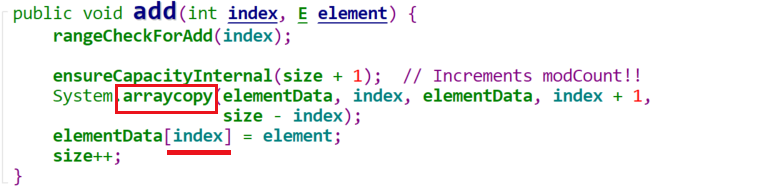
数组是在**物理内存**上一片连续的、逻辑上也连续、容量确定的**存储空间**。如int[10]。

数组不能动态的添加或删除其中的元素。所以产生了线性表。

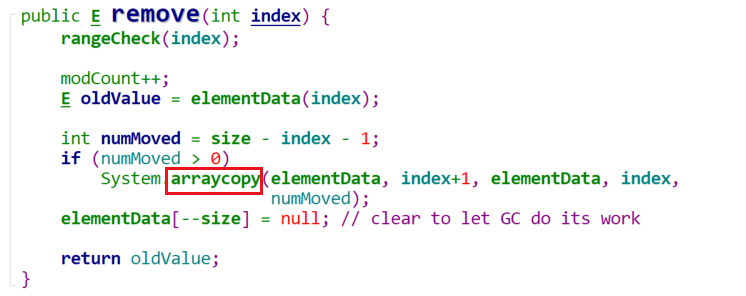
线性表（ArrayList）就是**物理上连续、逻辑上连续、**大小可**动态增加**的**数组**。

ArrayList添加或删除节点，要涉及到大多数节点的位移，耗费内存和时间非常多，造成代码性能降低，所以它的增删慢，但是可以使用链表来解决这个问题。

（1）如果要添加一个元素，就需要先复制原始数据，然后在对应位置插入这个元素。



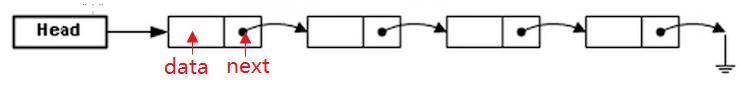
（2）同样，如果要删除一个元素，就需要先复制原始数据，然后在对应位置插入这个元素。



线性表中增删元素，都要涉及到整个数组的位移等操作，效率比较低。

由于链表的效率低，所以要考虑增删效率高的结构。就是链表结构LinkedList。

LinkedList是链表结构，他是**物理上不连续、逻辑上连续，**可以动态增加和删除节点。



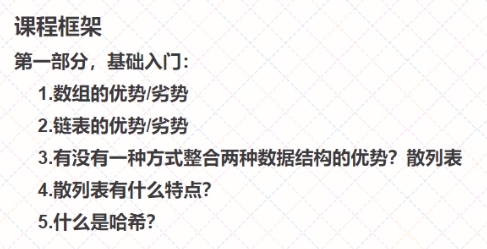
说明：data是当前元素，next是下一个元素的地址，所以他是物理上不连续、逻辑上连续的数据结构。

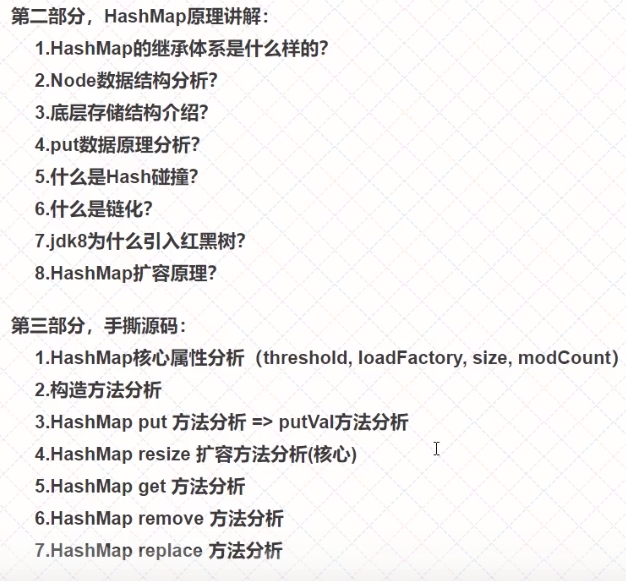
LinkedList添加或删除节点，只要改变相关节点的link.previous或者link.next值，只是赋值过程，所以添删节点效率高。而它的查询需要轮寻查找，所以它的查询慢。

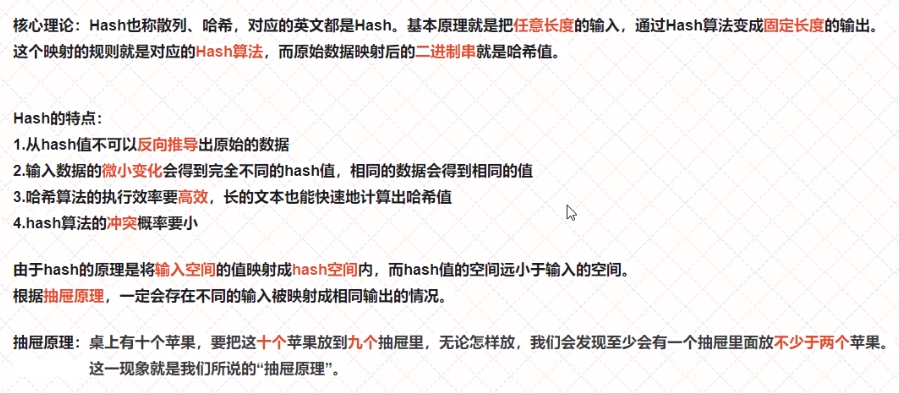
一句话都是因为他是链表特点，物理不连续，逻辑连续的造成的。

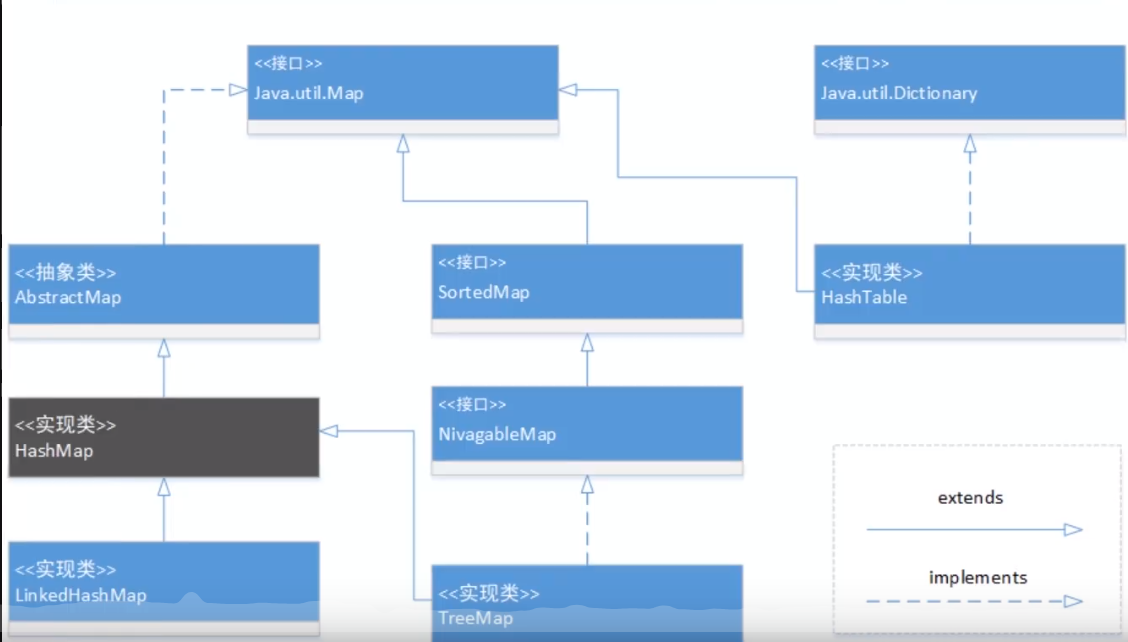
有没有结合两者的优点（查询和增删都快）的数据结构？

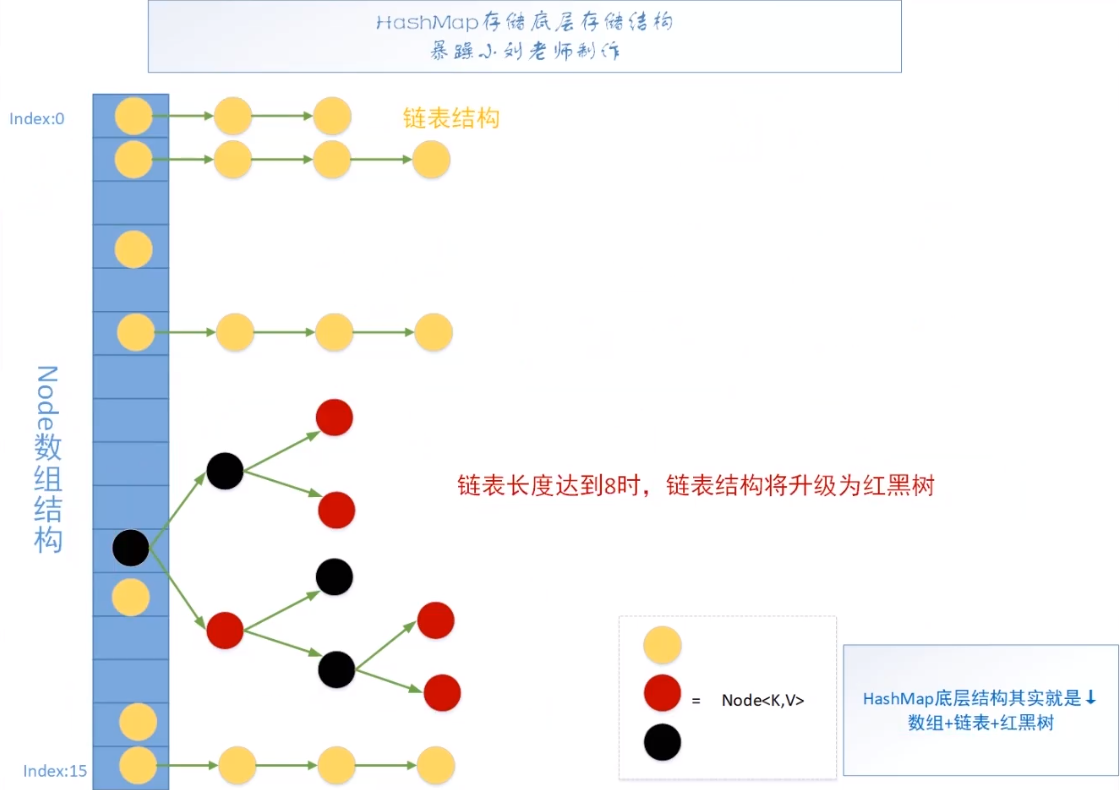
有，散列表结构，Hash表。

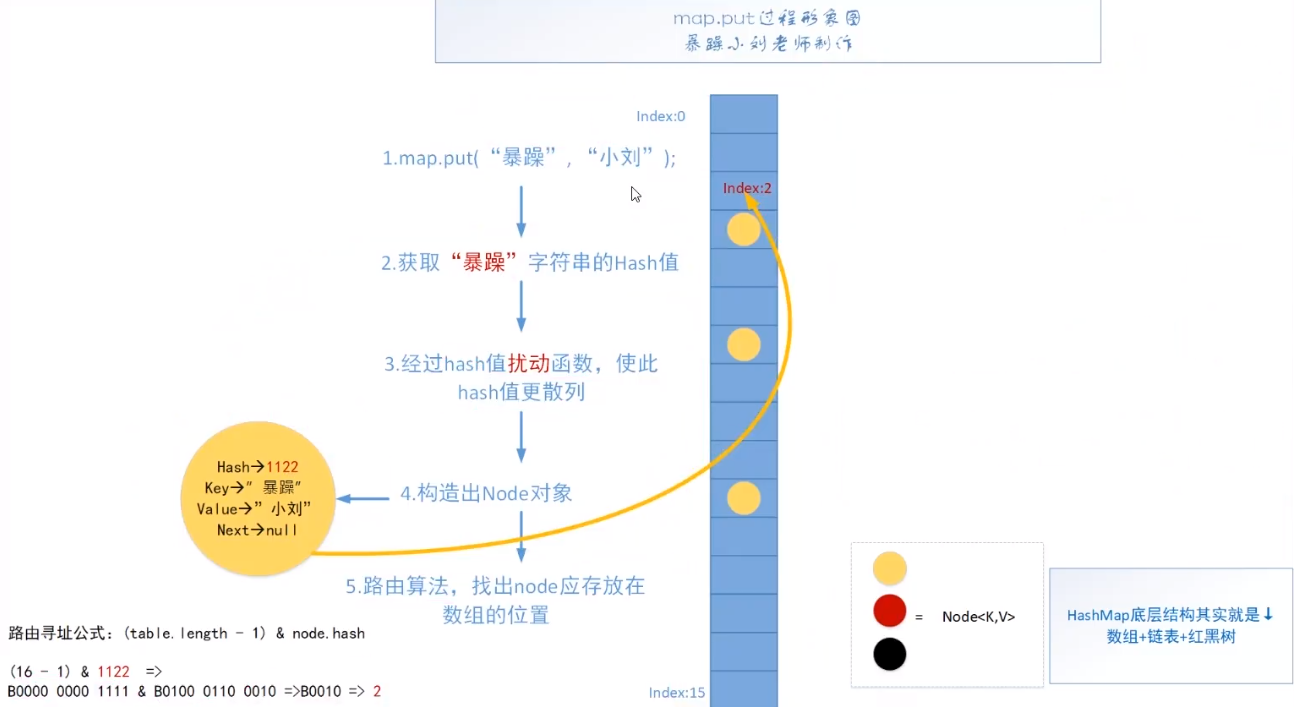


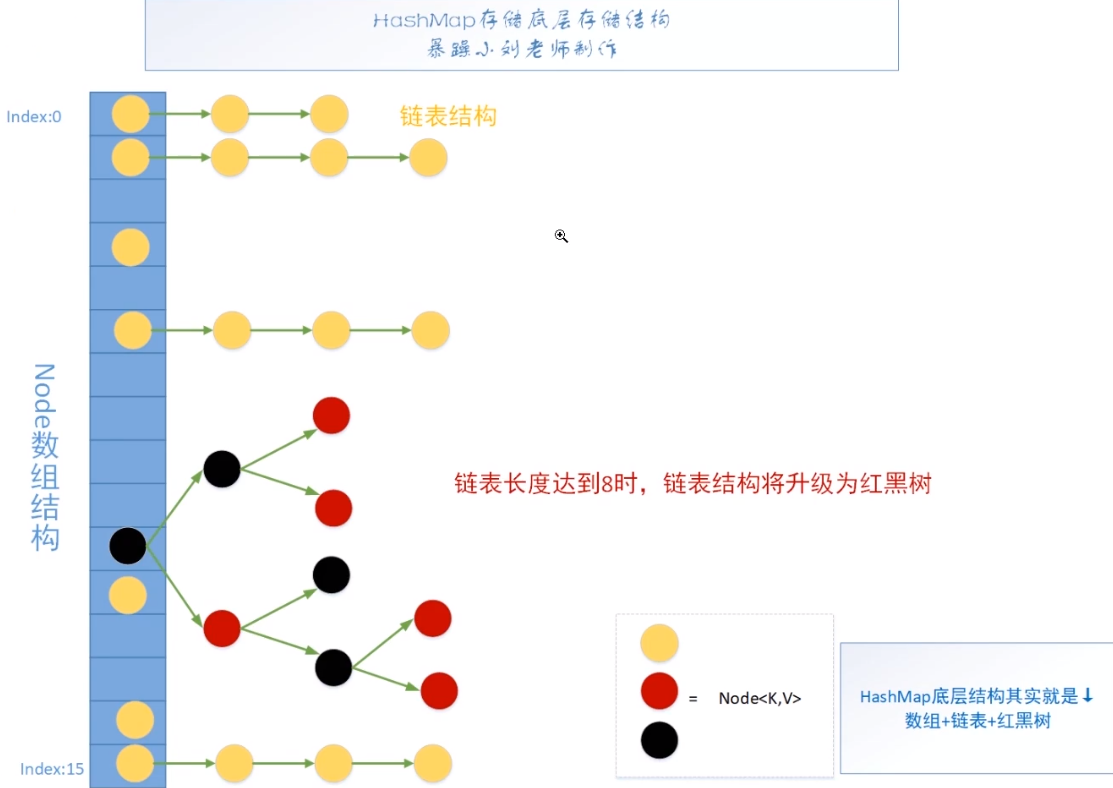


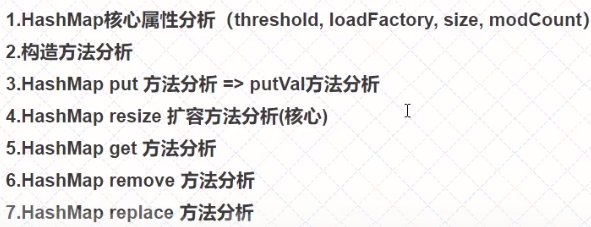












源码分析：

左移运算：十进制算法：左移n位相当于乘以2的n次方

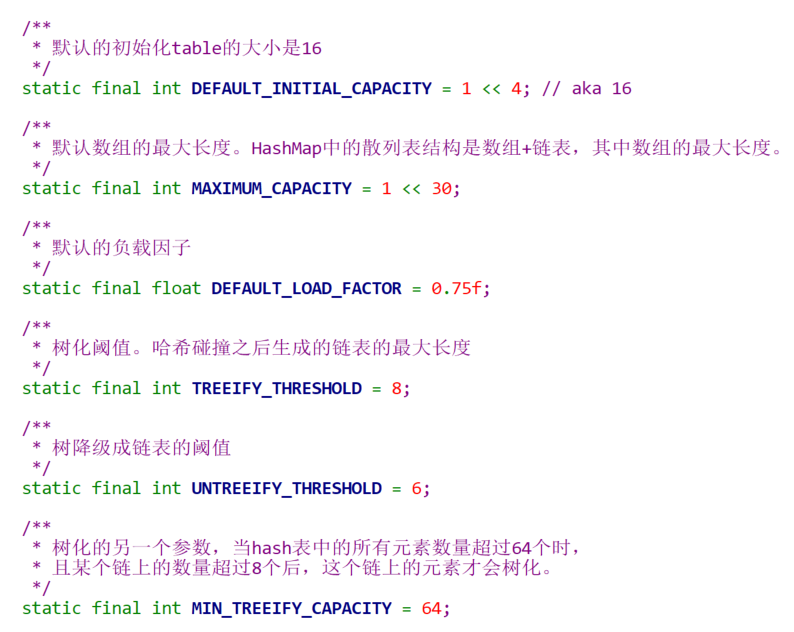
3<<2 3\*2\*2=12

1<<4 1\*2\*2\*2\*2=16

右移运算： 十进制算法：右移n位相当于除以2的n次方，结果取余数。

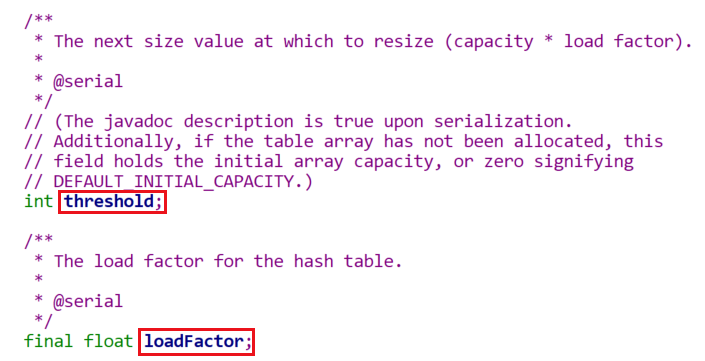
1. >2 11/2/2=2

35>>2 35/2/2=8



threshold扩容阈值。作用:当哈希表中的元素超过这个阈值时会触发扩容，如果超过这个阈值时而不去扩容，会导致查找性能会很差。扩容是因空间换时间，可以提高查找效率。

loadFactor 负载因子。作用：可以计算出threshold。Threshold = capacity（数组大小16） \* loadFactor （0.75）



HashMap(int initialCapacity, float loadFactor)

方法：作用是设置初始化值

当initialCapacity=7 tableSizeFor=8

当initialCapacity=8 tableSizeFor=8

当initialCapacity=9 tableSizeFor=16

也就是initialCapacity通过tableSizeFor计算后，threshold值始终大于或等于initialCapacity的值。

threshold的扩容规律是2的n次方，如：threshold =2、4、8、16...，n由tableSizeFor算法来保证。

tableSizeFor 作用：返回一个大于或者等于当前capacity的数字，返回的是2的次方值。目的是给数组准备扩容阈值threshold。

