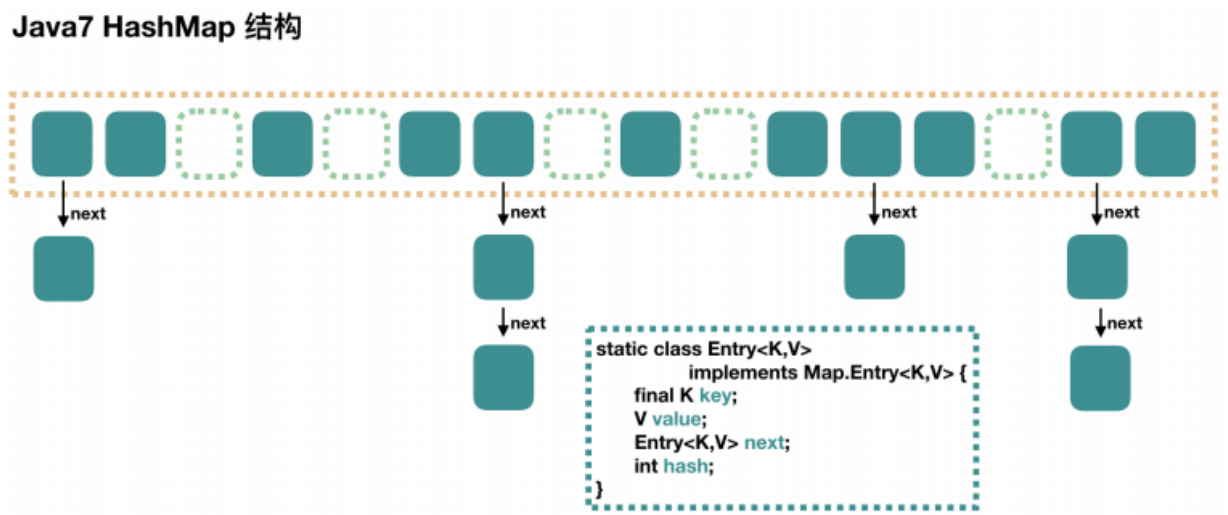
HashMap 根据键的 hashCode 值存储数据，大多数情况下可以直接定位到它的值，因而具有很快
的访问速度，但遍历顺序却是不确定的。 HashMap 最多只允许一条记录的键为 null，允许多条记
录的值为 null。HashMap 非线程安全，即任一时刻可以有多个线程同时写 HashMap，可能会导
致数据的不一致。如果需要满足线程安全，可以用 Collections 的 synchronizedMap 方法使
HashMap 具有线程安全的能力，或者使用 ConcurrentHashMap。我们用下面这张图来介绍
HashMap 的结构。



大方向上，HashMap 里面是一个数组，然后数组中每个元素是一个单向链表。上图中，每个绿色
的实体是嵌套类 Entry 的实例，Entry 包含四个属性：key, value, hash 值和用于单向链表的 next。

hashMap初始化容量默认为16

1.变量size，它记录HashMap的底层数组中已用槽的数量

2. capacity：当前数组容量，始终保持 2^n，可以扩容，扩容后数组大小为当前的 2 倍。

3. loadFactor：负载因子，默认为 0.75。

4. threshold：扩容的阈值，等于 capacity * loadFactor

5. HashMap扩容的条件是：当size大于threshold时，对HashMap进行扩容

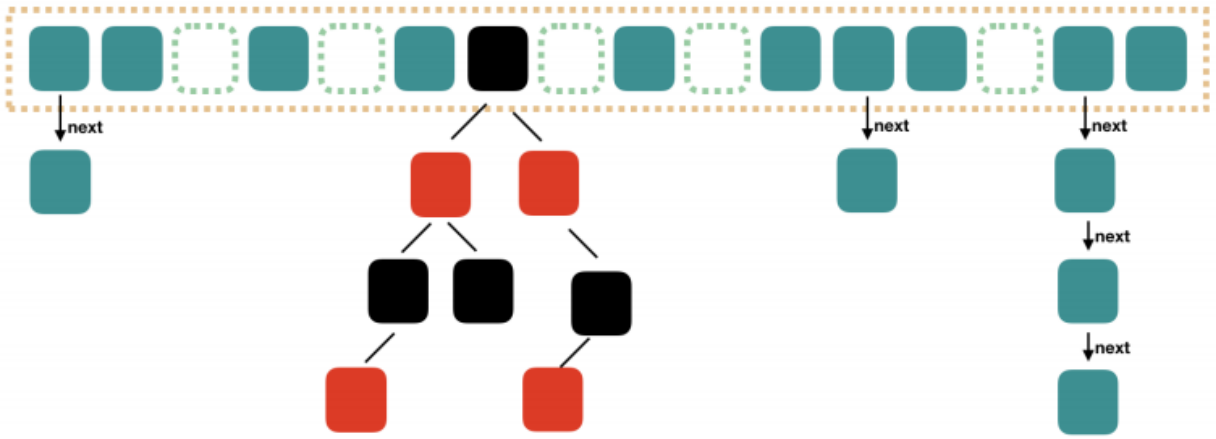
3.4.1.2.

JAVA8 实现

Java8 对 HashMap 进行了一些修改，最大的不同就是利用了红黑树，所以其由 数组+链表+红黑
树 组成。

根据 Java7 HashMap 的介绍，我们知道，查找的时候，根据 hash 值我们能够快速定位到数组的
具体下标，但是之后的话，需要顺着链表一个个比较下去才能找到我们需要的，时间复杂度取决
于链表的长度，为 O(n)。为了降低这部分的开销，在 Java8 中，当链表中的元素超过了 8 个以后，
会将链表转换为红黑树，在这些位置进行查找的时候可以降低时间复杂度为 O(logN)。

Java8 HashMap 结构



put() 方法
i.如果 K 的 hash 值在 HashMap 中不存在，则执行插入，若存在，则发生碰撞；
ii.如果 K 的 hash 值在 HashMap 中存在，且它们两者 equals 返回 true，则更新键值对；
iii. 如果 K 的 hash 值在 HashMap 中存在，且它们两者 equals 返回 false，则插入链表的尾部（尾插法）或者红黑树中（树的添加方式）。

1.定义(他是什么):
HashMap由数组+链表组成的，数组是HashMap的主体，链表则是主要为了解决哈希冲突而存在的，如果定位到的数组位置不含链表（当前entry的next指向null）,那么对于查找，添加等操作很快，仅需一次寻址即可；如果定位到的数组包含链表，对于添加操作，首先遍历链表，存在即覆盖，否则新增；对于查找操作来讲，仍需遍历链表，然后通过key对象的equals方法逐一比对查找。所以， HashMap中的链表出现越少，性能才会越高。

Q：HashMap 的工作原理？

A：HashMap 底层是 hash 数组和单向链表实现，数组中的每个元素都是链表，由 Node 内部类（实现 Map.Entry<K,V>接口）实现，HashMap 通过 put & get 方法存储和获取。

存储对象时，将 K/V 键值传给 put() 方法：①、调用 hash(K) 方法计算 K 的 hash 值，然后结合数组长度，计算得数组下标；②、调整数组大小（当容器中的元素个数大于 capacity * loadfactor 时，容器会进行扩容resize 为 2n）；

③、i.如果 K 的 hash 值在 HashMap 中不存在，则执行插入，若存在，则发生碰撞；

ii.如果 K 的 hash 值在 HashMap 中存在，且它们两者 equals 返回 true，则更新键值对；

iii. 如果 K 的 hash 值在 HashMap 中存在，且它们两者 equals 返回 false，则插入链表的尾部（尾插法）或者红黑树中（树的添加方式）。

（JDK 1.7 之前使用头插法、JDK 1.8 使用尾插法）

（注意：当碰撞导致链表大于 TREEIFY_THRESHOLD = 8 时，就把链表转换成红黑树）

获取对象时，将 K 传给 get() 方法：①、调用 hash(K) 方法（计算 K 的 hash 值）从而获取该键值所在链表的数组下标；②、顺序遍历链表，equals()方法查找相同 Node 链表中 K 值对应的 V 值。

hashCode 是定位的，存储位置；equals是定性的，比较两者是否相等