# **Android源码分析--ArrayMap优化**

ArrayMap是Android系统独有封装的，我们要在4.4以前运用，要使用v4的包兼容来获取ArrayMap。

要了解ArrayMap，需要大家先去了解HashMap。

HashMap基于哈希表的Map接口的实现。此实现提供所有可选的映射操作，并允许使用null值和null键。（除了不同步和允许使用 null 之外，HashMap 类与Hashtable大致相同）此类不保证映射的顺序，特别是它不保证该顺序恒久不变。

值得注意的是HashMap不是线程安全的，如果想要线程安全的HashMap，可以通过Collections类的静态方法synchronizedMap获得线程安全的HashMap。

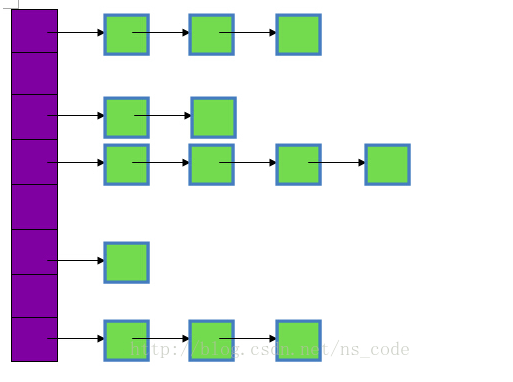
```

Map map= Collections.synchronizedMap(newHashMap());

```

二、HashMap数据结构

HashMap的底层主要是基于数组和链表来实现的，它之所以有相当快的查询速度主要是因为它是通过计算散列码来决定存储的位置。HashMap中主要是通过key的hashCode来计算hash值的，只要hashCode相同，计算出来的hash值就一样。如果存储的对象对多了，就有可能不同的对象所算出来的hash值是相同的，这就出现了所谓的hash冲突。学过数据结构的同学都知道，解决hash冲突的方法有很多，HashMap底层是通过链表来解决hash冲突的。



图中，紫色部分即代表哈希表，也称为哈希数组，数组的每个元素都是一个单链表的头节点，链表是用来解决冲突的，如果不同的key映射到了数组的同一位置处，就将其放入单链表中。

HashMap的基础构造器HashMap(int initialCapacity, float loadFactor)带有两个参数，它们是初始容量initialCapacity和加载因子loadFactor。

initialCapacity：HashMap的最大容量，即为底层数组的长度。

loadFactor：负载因子loadFactor定义为：散列表的实际元素数目(n)/ 散列表的容量(m)。

负载因子衡量的是一个散列表的空间的使用程度，负载因子越大表示散列表的装填程度越高，反之愈小。对于使用链表法的散列表来说，查找一个元素的平均时间是O(1+a)，因此如果负载因子越大，对空间的利用更充分，然而后果是查找效率的降低；如果负载因子太小，那么散列表的数据将过于稀疏，对空间造成严重浪费。

一个HashMap  初始容量为 16，负载因子为 0.75

只要一满足扩容条件，HashMap的空间将会以2倍的规律进行增大。

HashMap基础的结构介绍到这里。

为何要使用ArrayMap？

我们可以很清晰看到ArrayMap是继承于SimpleArrayMap，SimpleArrayMap才是其真正的实现，而通过Map提供的接口包装暴露方法

IMG_257

其真正实现类是SimpleArrayMap。

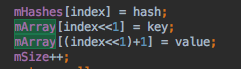
1、存储方式不同

HashMap内部有一个HashMapEntry[]对象，每一个键值对都存储在这个对象里，当使用put方法添加键值对时，就会new一个HashMapEntry对象

ArrayMap的存储中没有Entry这个东西，他是由两个数组来维护的

IMG_258

mHashes数组中保存的是每一项的HashCode值，mArray中就是键值对，每两个元素代表一个键值对，前面保存key，后面的保存value。



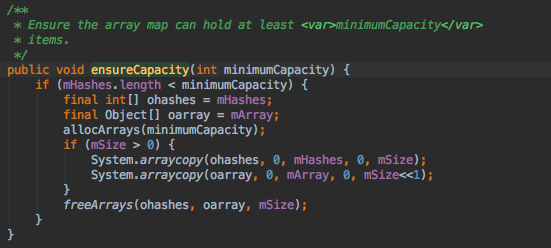
从代码查看put的方法可以清晰看到，mHashes保存的是hash值，而mArray保存的相邻两个值保存的是key和value。

2、添加数据时扩容时的处理不一样

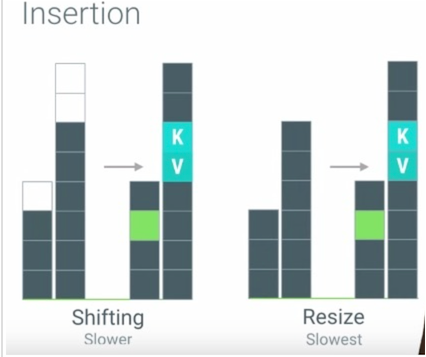
HashMap使用New的方式申请空间，并返回一个新的对象，开销会比较大

ArrayMap用的是System.arrayCopy数据，所以效率相对要高。

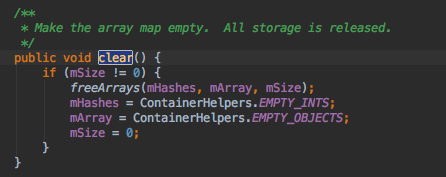
初始化计算是否满足最小需要的容量，然后去扩容，可以看出ArrayMap是使用System.arrayCopy来移动数组的。



3、ArrayMap提供了数组收缩的功能，只要判断过判断容量尺寸，例如clear，put，remove等方法，只要通过判断size大小触发到freeArrays或者allocArrays方法，会重新收缩数组，是否空间。



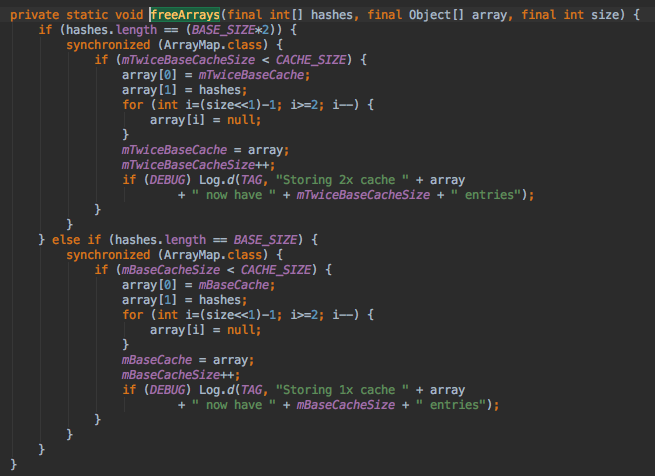
扩容过程



申请数组，对于BASE\_SIZE\*2和BASE\_SIZE两种尺寸的数组在这里它并没有对它们进行释放，而是把它们缓存起来，这样我们在分配的时候，如果需要分配这两种大小的数组，就可以直接从缓存中取得，否则，就直接new两个数组，第二个数组存放的是键值对，所以大小是size的两倍，size<<1左移一位操作就相当于乘以2

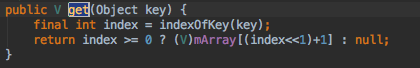


释放数组

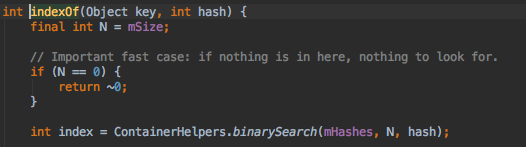


4、ArrayMap相比传统的HashMap速度要慢，因为查找方法是二分法，并且当你删除或者添加数据时，会对空间重新调整，在使用大量数据时，效率低于50%。可以说ArrayMap是牺牲了时间换区空间。但在写手机app时，适时的使用ArrayMap，会给内存使用带来可观的提升。ArrayMap内部还是按照正序排列的，这时因为ArrayMap在检索数据的时候使用的是二分查找，所以每次插入新数据的时候ArrayMap都需要重新排序，逆序是最差情况；

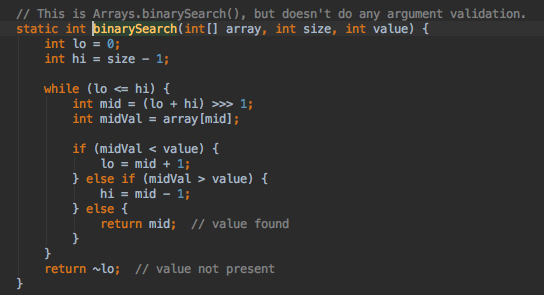
get的方法会调用indexOfKey



indexOfKey实际是调用indexOf方法，然后再调用ContainerHelpers里面的二分法查找的方法



通过二分法查找到index



如果使用HashMap，当事件个数不断加大，那么更加会产生大量空余的内存。

在内存和计算速度的取舍，在移动端来说，内存比较金贵。