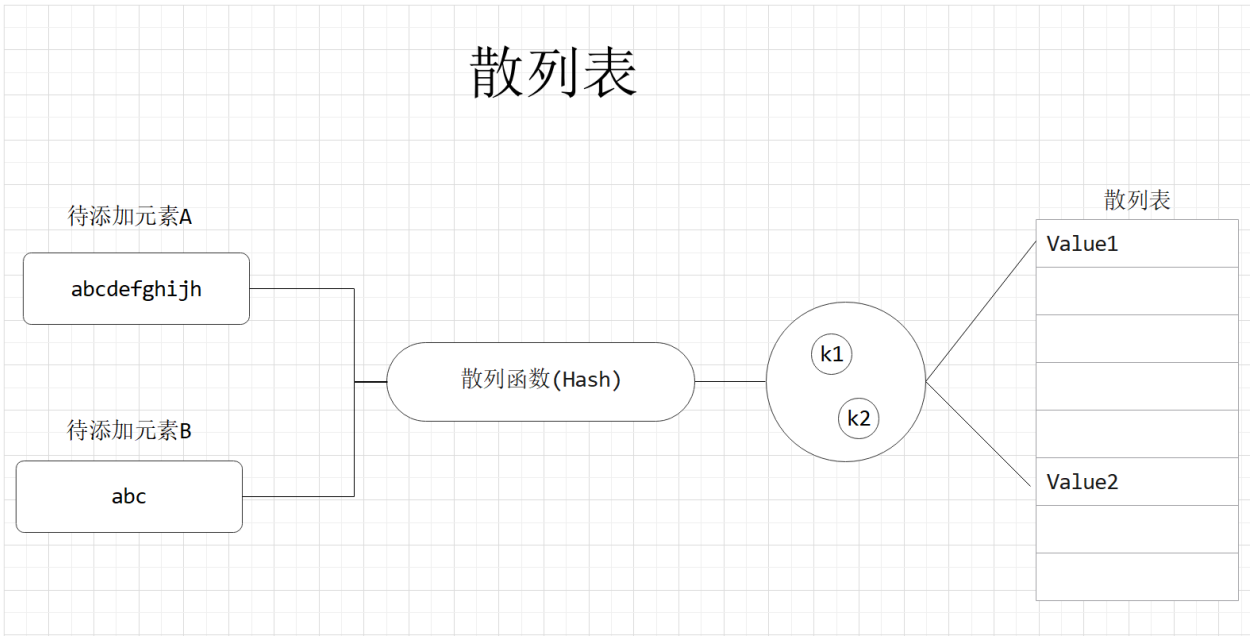
在说明HashMap之前要先说明什么是HashTable,因为HashTable(哈希表)是HashMap的底层最直接的实现方式.

Hash表也称之为散列表,是一种根据关键字值(Key-Value)而直接进行访问的数据结构,它通过把关键码值映射到表中的一个位置来访问记录,以此加快访问速度.

在链表/数组[ArrayList/LinkedList]中查找某个关键字通常需要遍历整个数据结构,也就是O(N)的时间级,对于Hash表而言,只需要O(1)的时间级别.

映射的过程中使用到的映射函数称之为散列函数,存放最终记录数据的数组称为散列表.是把任意长度的输入通过散列算法变换成固定长度的输出,该输出就是散列值。



实际上,在现实生活中汉语字典就是哈希表的典型体现.

汉语字典的优点就是可以通过前面的拼音目录快速定位要查找的汉字,当给定一个汉字,大脑会自动将汉字转化为拼音,这个转换的过程可以看成一个散列函数,再根据拼音找到该字所在的页码.

哈希表有两个问题需要思考:

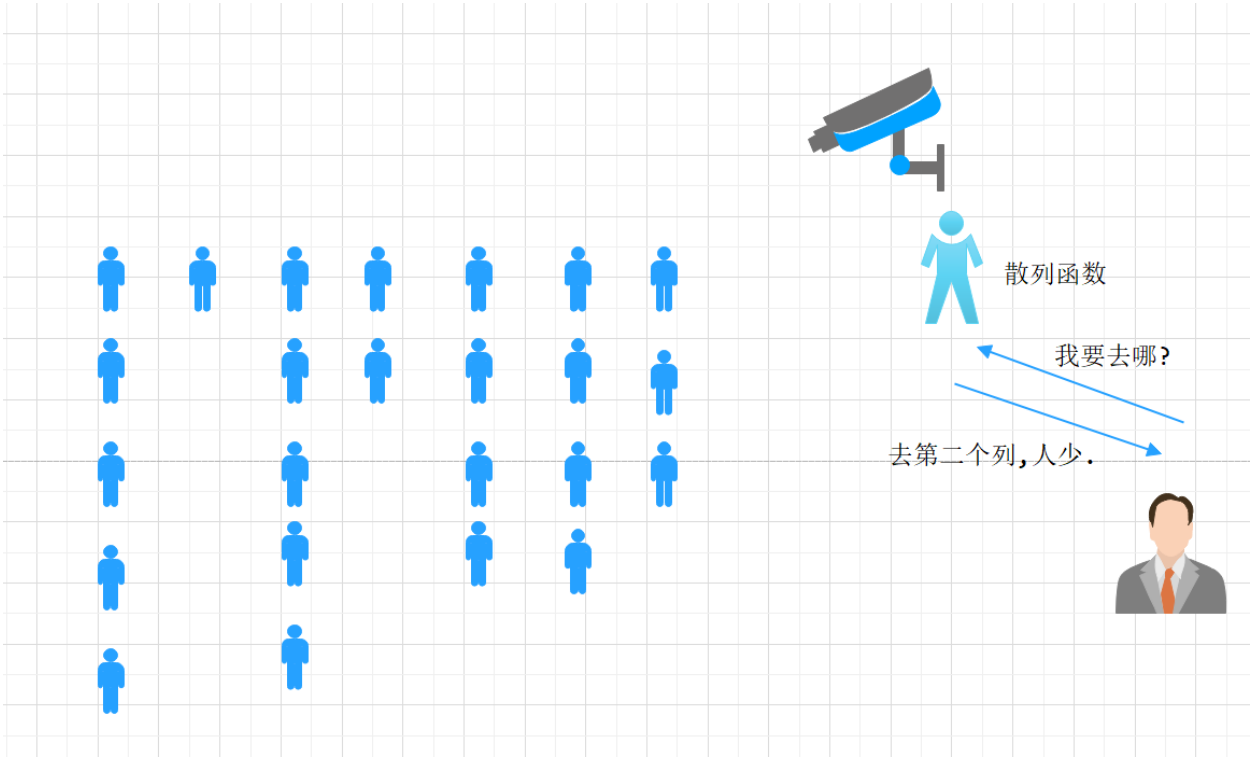
1.为什么要有散列函数?

2.多个Key通过散列函数会得到相同的散列值,这时候应该如何处理?

答:

1.散列函数的存在可以快速定位到一个Key在散列表中的位置,和汉语词典一致,如果没有拼音与汉字的转换规则,那么除了翻一遍整个词典,没有更好的方法.

也可以通过这个场景进行理解,散列函数可以帮你监控到所有队列的长度,如果没有散列函数,你需要自己一列一列的看哪一列人少,相当于遍历了一遍,但是如有散列函数,会告诉你直接去哪个位置.



2.多个Key通过散列函数获取到了相同的结果,这种情况称之为哈希冲突,也叫作哈希碰撞.

比如字典中的同音字,获取到的拼音是相同的,那么如何在字典中/哈希表中保存这些数据呢?

- 开放地址法

当通过散列函数计算遇到了冲突,再通过另外一个函数计算获取到新的映射关系.

例:有一个汉字(里),拼音是li,通过散列函数计算,位置在100页,又来了一个汉字(理),通过散列函数计算,位置也在100页,但是100页已经有了里字,这时候再通过另外一种函数计算新的位置,比如说计算另外一种函数的计算方式是获取到的值+1,那么理字就可以放到101的位置.

缺点:对于开放地址法,可能会遇到二次冲突,或者三次冲突,所以需要良好的散列函数,分布的越均匀越好,优点:每一个元素都有单独的位置.

- 链地址法

可以将字典中的每一页都看成是一个子数组或者子链表,当遇到冲突了,直接向当前页码中的子数组填充即可,遇到同音字可能会遍历子数组或者子链表.

HashMap默认采取的就是链地址法,所以哈希表的本质上就是数组+链表,通过散列函数确定元素在数组中存储的索引,再将元素放到每一个索引对应的链表中去.

缺点:如果一次冲突过多,会造成子数组/子链表过长,遍历的时间也会变长,优点:添加元素的时候不会冲突.