Hash Table

哈希表

描述：

哈希表是一种高效的key-value存储结构，时间复杂度为，通过散列函数对key值计算，得到value在哈希表数组中应该存储的位置。散列函数/哈希函数是哈希表的核心，一般形式为，具有速度快的优势，设计好的散列函数的碰撞率低、数据分布平均的优点，可以将任意情况下的数据都分散在哈希表内部的数组中。

哈希表的一般形式如图所示：



将散列函数hash计算出key的哈希值作为这个key在哈希表中的存储位置，即数组下标index。很多哈希函数的结果都在0~2,147,483,647之间，远远大于哈希表中数组的长度，因此一般实际应用时需要再对数组长度n取模，即。

当哈希表中存储的数据量count超过n时，显然某些key会存储在相同的下标中，即碰撞情况，设计良好的散列函数可以让key均匀的分布在数组中，保证时间复杂度近似为。当数据量count过多时，可以将数组n的长度扩充，然后重新哈希所有key，找出在新数组中的存储位置。

解法：

有序序列s可以是升序或降序的，即从小到大或从大到小。本问题中假设s是升序的。

在这样的序列s中想要找出某个元素x是否存在，首先检查序列s最小、最大和中间三个位置处的元素low、high和mid，其中。

若x与mid相等则mid即为所求，算法结束；若x小于mid，则x在mid左边的某一位置；若x大于mid，则x在mid右边的某一位置。然后将序列的检查范围缩小，继续递归的重复上面的过程，直到找到x等于mid。

对于下面这个长度为10的序列s，查找，初始时设置low、high和mid三个值，分为作为序列s的最左、最右和中间的下标值，即，，：



1. 时，，因此x在mid与high之间，并且不包括mid，设置，；
2. 时，，x仍然在mid与high之间，设置；
3. ，返回x所在下标值8，算法结束。如果查找的x为49，则在第(3)步中仍然有，则仍然设置，，有，因此设置，此时不再成立，算法结束，x没有找到。

对于长度为n的序列s，每次计算mid的时间看作。在最好情况下，一次查找就可以找到；在最坏情况下需要次才能找到x；在随机情况下该算法的时间复杂度为。