**3.1 符号表**

符号表最主要的目的就是将一个键和一个值联系起来。

符号表是一种存储键值对的数据结构，支持两种操作：插入和查找。

**3.1.1 API**

每个键只对应一个值（表中不允许存在重复的键）

删除的实现有两种方法：延时删除，也就是将键对应的值置为空，然后在某个时候删去所有值为空的键，或是即时删除，立刻从表中删除指定的键。

**3.1.2 有序符号表**

典型的应用程序中，键都是Comparable的对象，因此可以使用a.compareTo(b)来比较a和b两个键。

**3.1.3 用例举例**

**3.1.4 无序链表中的顺序查找**

符号表中使用的数据结构的一个简单选择是链表，每个节点存储一个键值对。

get的实现即为遍历链表，put的实现也需要遍历链表。

编程：顺序查找（基于无序链表）

向一个空表中插入N个不同的键需要~N2/2次比较。

**3.1.5 有序数组中的二分查找**

它使用的数据结构是一对平行的数组，一个存储键一个存储值。

编程：二分查找（基于有序数组）

**3.1.6 对二分查找的分析**

二分查找减少了比较的次数但无法减少运行所需时间，在键是随机排列的情况下，构造一个基于有序数组的符号表所需要访问数组的次数是数组长度的平方级别。

向一个大小为N的有序数组中插入一个新的元素在最坏情况下需要访问~2N次数组，因此向一个空符号表中插入N个元素在最坏情况下需要访问~N2次数组。

**3.1.7 预览**

二分查找平均情况下的成本，查找为lgN，插入为N。

参考文档：符号表的各种实现的优缺点