

什么是散列

陈斌 北京大学 gischen@pku.edu.cn

散列: Hashing

- ◇前面我们利用数据集中关于数据项之间排列关系的知识,来将查找算法进行了提升
- ❖ 如果数据项之间是按照大小排好序的话, 就可以利用二分查找来降低算法复杂度。
- ◇ 现在我们进一步来构造一个新的数据结构 ,能使得查找算法的复杂度降到O(1),这 种概念称为"散列Hashing"

散列: Hashing

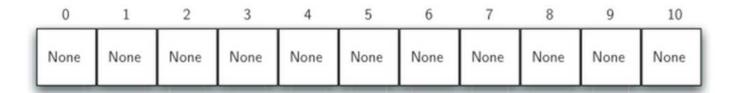
- ❖能够使得查找的次数降低到常数级别,我们对数据项所处的位置就必须有更多的先验知识。
- ❖如果我们事先能知道要找的数据项应该出现在数据集中的什么位置,就可以直接到那个位置看看数据项是否存在即可。
- ❖ 由数据项的值来确定其存放位置,如何能做到这一点呢?

散列:基本概念

- ❖ 散列表 (hash table,又称哈希表)是一种数据集,其中数据项的存储方式尤其有利于将来快速的查找定位。
- ◇散列表中的每一个存储位置,称为槽(slot),可以用来保存数据项,每个槽有一个唯一的名称。

散列:基本概念

- ❖ 例如: 一个包含11个槽的散列表, 槽的名称分别为0~10
- ❖在插入数据项之前,每个槽的值都是 None,表示空槽



散列:基本概念

- ◇实现从数据项到存储槽名称的转换的,称为散列函数 (hash function)
- ❖下面示例中,散列函数接受数据项作为参数,返回整数值0~10,表示数据项存储的槽号(名称)

❖ 为了将数据项保存到散列表中,我们设计第一个散列函数

数据项: 54, 26, 93, 17, 77, 31

❖有一种常用的散列方法是"求余数",将数据项除以散列表的大小,得到的余数作为槽号。

实际上"求余数"方法会以不同形式出现在所有散列函数里

因为散列函数返回的槽号必须在散列表大小范围 之内,所以一般会对散列表大小求余

❖本例中我们的散列函数是 最简单的求余:

h(item) = item % 11

❖按照散列函数h(item), 为每个数据项计算出存放 的位置之后,就可以将数 据项存入相应的槽中

Item	Hash Value
54	10
26	4
93	5
17	6
77	0
31	9

❖ 例子中的6个数据项插入后,占据了散列表 11个槽中的6个。

槽被数据项占据的比例称为散列表的"负载因子", 这里负载因子为6/11

- ❖ 数据项都保存到散列表后, 查找就无比简单
- ❖ 要查找某个数据项是否存在于表中,我们只需要使用同一个散列函数,对查找项进行计算,测试下返回的槽号所对应的槽中是否有数据项即可

实现了0(1)时间复杂度的查找算法。

- ❖ 不过,你可能也看出这个方案的问题所在,这组数据相当凑巧,各自占据了不同槽
- ◇假如还要保存44, h(44)=0, 它跟77被分配到同一个0#槽中, 这种情况称为"冲突collision", 我们后面会讨论到这个问题的解决方案。

0	1	2	3	4	5	6	7	8	9	10
77	None	None	None	26	93	17	None	None	31	54

