Data Structures

Ch7

查找
Searching

2023 年 12 月 5 日

学而不厌 誨人不倦

Chapter 7 查找

- 7.1 概述
- 7.2 线性表查找技术
- 7.3 树表的查找技术
- 7.4 散列表查找技术
- 7.5 各种查找方法的比较
- 7.6 扩展与提高

本章的重点就是研究查找表的存储方法以及在此基础上的查找方法。

7.4 散列表的查找技术

7-4-1　散列查找的基本思想

7.4 散列表的查找技术

1. 回顾查找技术

待查值k ⇨ 确定 k 在存储结构中的位置

(1) 顺序查找

==、!=　　$O(n)$

(2) 折半查找

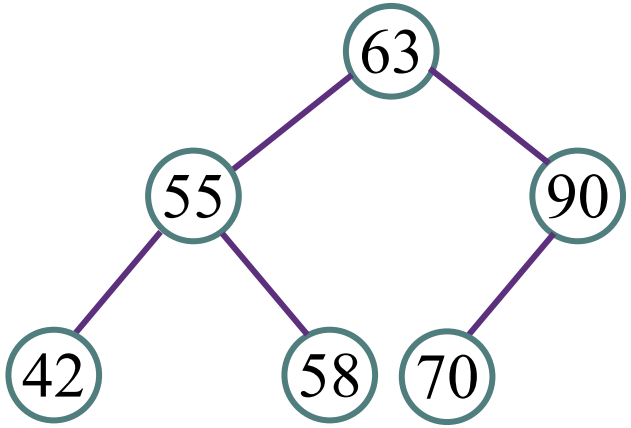
<、==、>　　$O(\log_2 n)$

(3) 二叉排序树查找

$O(n) \sim O(\log_2 n)$

0	1	2	3	4	5	6	7	8	9
	10	15	24	6	12	35	40	98	55

0	1	2	3	4	5	6	7	8	9
	10	15	20	25	30	35	40	45	50



通过一系列的给定值与关键码的比较，查找效率依赖于查找过程中进行的给定值与关键码的比较次数

1. 回顾查找技术

能否不用比较，通过关键码能够直接确定存储位置？

在存储位置和关键码之间建立一个确定的对应关系

设关键码*key*在存储结构中的位置是*addr*，则有 $addr = H(key)$。

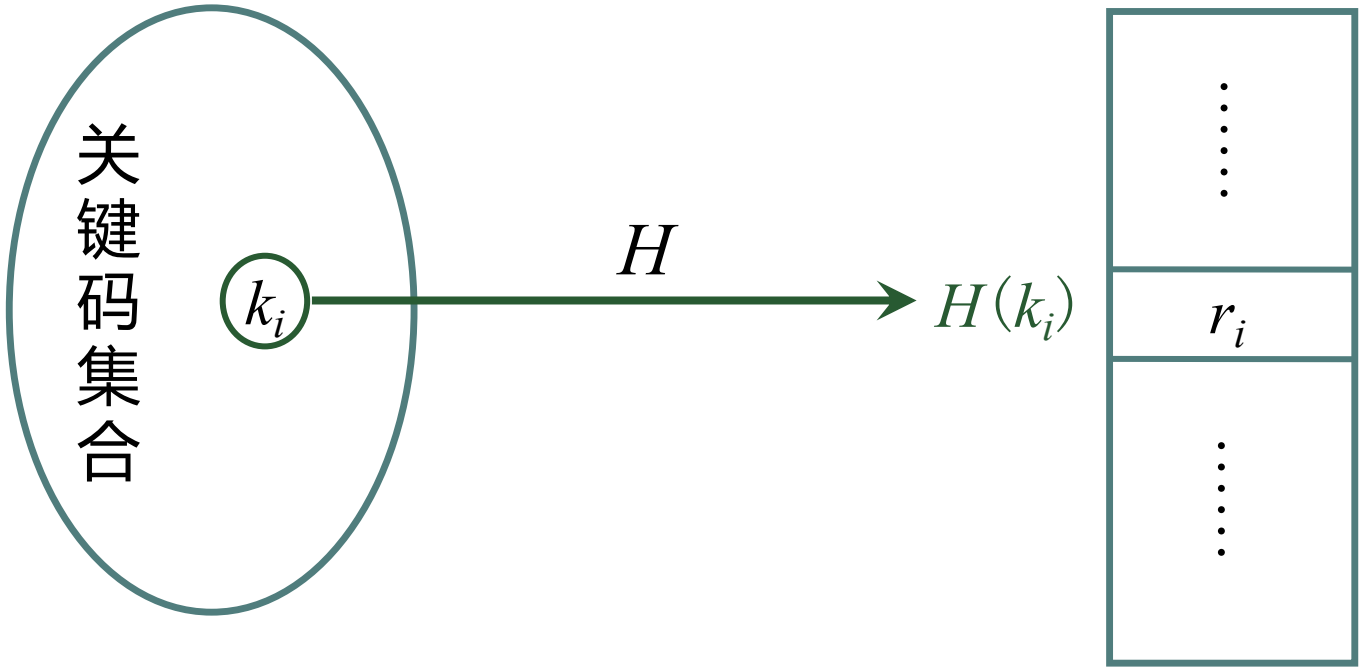
查找技术
- 比较式查找
- 计算式查找

→ Hash哈希查找技术

→ 散列表查找技术

2. 散列的基本思想

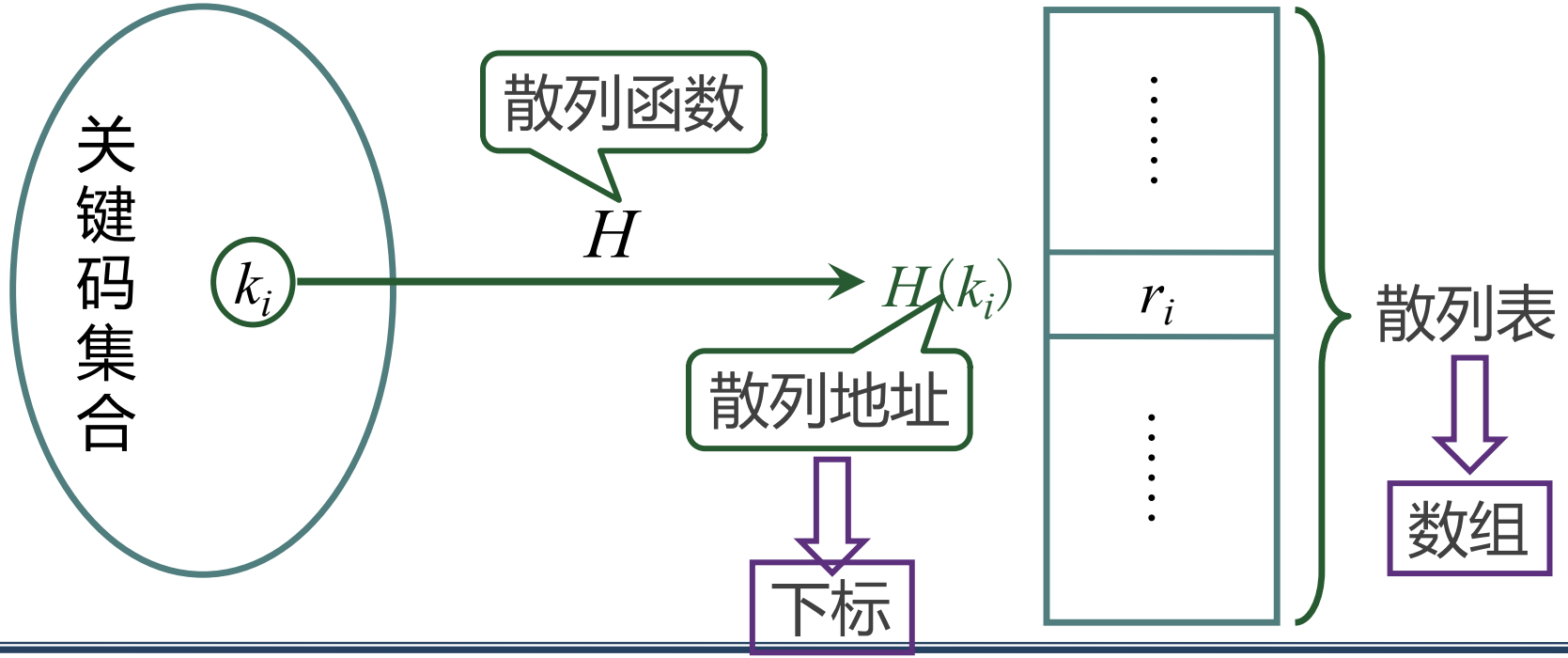
散列的基本思想：在记录的关键码和存储地址之间建立一个确定的对应关系，通过计算得到待查记录的地址。



- 优点： 查找速度极快O(1),查找效率与元素个数n无关

3. 散列的基本概念

- 散列表：采用散列技术存储查找集合的连续存储空间。
- 散列函数：将关键码映射为散列表中适当存储位置的函数。
- 散列地址：由散列函数所得的存储地址。



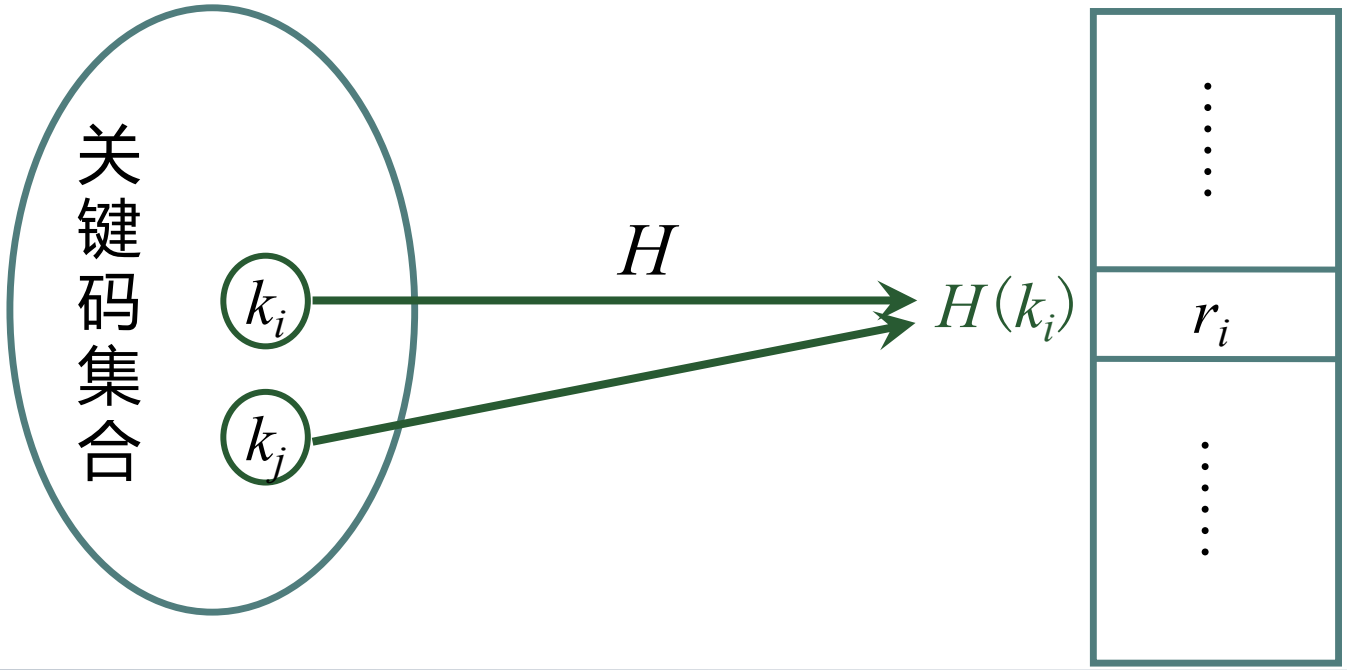
4. 散列的关键问题

散列表是基于散列函数建立的一种查找表。

如何设计散列函数？ **如何解决冲突？**

冲突：对于两个不同关键码$k_i \neq k_j$，有$H(k_i) = H(k_j)$。

同义词：k_i和k_j相对于H称做同义词。



散列通过关键码定位记录，无法表达记录之间的**逻辑关系**，所以，散列主要是**面向查找**的存储结构。

散列技术最适合回答的问题是：

如果有的话，哪个记录的关键码等于待查值？ 散列不能进行范围查找

7.4 散列表的查找技术

7-4-2　散列函数设计

设计原则

如何设计散列函数?

（1）计算简单。散列函数不应该有很大的计算量，否则会降低查找效率。

（2）地址均匀。函数值要尽量均匀散布在地址空间，保证存储空间的有效利用并减少冲突。

对数字的关键字可以有多种哈希函数的构造方法；

若是非数字关键字，则需先对其进行数字化处理。

1、直接定址法
2、数字分析法
3、平方取中法
4、折叠法
5、除留余数法
6、随机数法

1. 直接定址法

取关键字或关键字的某个线性函数值为散列地址。

$$H(key) = key \quad \text{(i)}$$

$$H(key) = a \cdot key + b \quad \text{(ii)}$$

例：统计解放后出生人口，以出生年份作为关键字

$$H(key) = key - 1949 + 1$$

地址	01	02	···	23	···
出生年份	1949	1950	···	1971	···

例： {100，300，500，700，800，900}，

散列函数： Hash(key)=key/100

0	1	2	3	4	5	6	7	8	9
	100		300		500		700	800	900

仅限于：地址集合的大小=关键字集合的大小

2. 数字分析法

对关键字进行按“位”分析，取重复度小的若干位组合成散列地址。

假设关键字集合中的每个关键字都是s位数字组成(k1,k2,……kn)，分析关键字集合中的全体，并从中提取分布均匀的若干位或它们的组合作为地址。

例，多条记录，关键字为 8 位十进制数，要求取两位作为散列地址。

8	1	3	4	6	5	3	7
7	1	3	7	2	2	4	7
8	1	3	8	7	4	2	2
8	2	3	0	1	3	6	7
8	1	4	2	2	8	1	7
8	1	3	3	8	9	6	7

取 4、5、6、7 位中的任意两位即可。

仅限于：能预先估计出全体关键字的每一位上各种数字出现的频度。

3. 平方取中法

对关键码平方后，按散列表大小，取中间的若干位作为散列地址。

例：散列地址为 2 位，设计平方取中法的散列函数。

$$(1234)^2 = 1522756$$

$$(1235)^2 = 1525225$$

平方扩大了相近数之间的差别

例：

X	X^2	取4位
2 0 0 5 2 4	4020987 4576	2098
2 0 0 5 0 2	4020105 2004	2010
0 1 2 0 0 5	0014412 0025	1441
0 2 2 0 0 5	0048422 0025	4842
0 3 2 0 0 5	0102432 0025	0243

4.折叠法

若关键字的位数特别多，则可将其分割成几部分，然后取他们的叠加和作为地址。**移位叠加、间界叠加**

将关键字从低到高分割成位数相同的几部分，然后取各部分的叠加和(舍去进位)作为哈希函数。

例，　图书编号　0 – 442 – 20586 – 4

0 4 | 4 2 2 0 | 5 8 6 4

```
   5 8 6 4
   4 2 2 0
+)       0 4
-----------
   0 0 8 8
```

最终取 0 0 8 8 作为哈希地址。

5.除留余数法

除留余数法

$$H(key)=key \mod p$$

如何选取合适的 p，才能产生较少的同义词?

例如：$p=21=3\times7$

关键码	7	14	21	28	35	42	49	56
散列地址	7	14	0	7	14	0	7	14

小于等于表长（最好接近表长）的最小素数或不包含小于20质因子

5.除留余数法

除留余数法

$$H(key)=key \mod p$$

例 2：散列表长为15，设计除留余数法的散列函数。

$$H(key)= key \mod 13$$

散列技术名称的演变过程

Hash ⇨ 哈希 ⇨ 杂凑 ⇨ 散列

适用于：最简单、最常用，不要求事先知道关键码的分布

6.随机数法

取关键字的随机函数值作为散列地址。

$$H(key) = Random(key)$$

总结:

实际造表时，采用何种构造哈希函数的方法取决于建表的关键字集合的情况（包括关键字的范围和形态），总的原则是使产生冲突的可能性降到尽可能地小。

7.4 散列表的查找技术

7-4-3　处理冲突的方法

开放定址法　双散列探测　拉链法/链地址法

散列/哈希冲突

对于不同的关键字可能得到同一散列地址，即 key1≠ key2 ，而 f(key1) = f(key2)，这种现象称为**散列冲突**。

处理冲突的实际含义是：为产生冲突的地址寻找下一个散列地址。

例，数字分析法中

8	1	3	4	6	5	3	7
7	1	3	7	2	2	4	7
8	1	3	8	7	4	2	2
8	2	3	0	1	3	6	7
8	1	4	2	2	8	1	7
8	1	3	3	8	9	6	7

例，除留余数法中

关键字	28	35	63	77	105
散列地址	7	14	0	14	0

$p = 21$

A. 主观设计不当

B. 客观存在

哈希地址是有限的，而记录是无限的。

开放定址法　双散列探测　拉链法

1. 开放定址法

为产生冲突的地址H(key)求得一个地址序列：

H0, H1, H2,…Hs　　1≤s ≤ m-1

其中：$H0 = (key) \bmod m$

$Hi = (key + d_i) \bmod m \quad i = 1,2,\ldots s$

$H(key)$　散列函数

m　　散列表长

d_i　　增量序列

即：在 key mod m 的基础上，若发现冲突，则使用增量 d_i 进行新的探测，直至无冲突出现为止。

关键是如何设计 d_i

线性探测法:

$d_i = c \times i$ 最简单的情况下 c=1;

二次探测法:

平方探测再散列。

随机探测法:

di是一组伪随机数列，

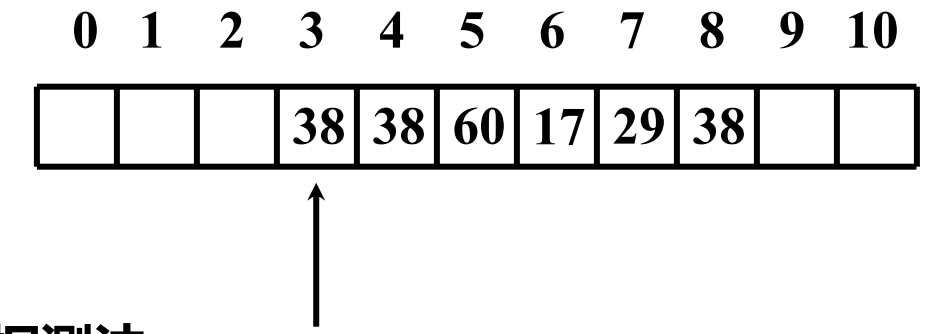
或者 di =i×H(key)

1. 开放定址法

线性探测法	$d_i = 1, 2, 3, \ldots, m-1$
平方探测法	$1^2, -1^2, 2^2, -2^2, \ldots, q^2, -q^2$ $(q \leqslant m/2)$
随机探测法	d_i = 随机数

例，关键字为 (17 , 60 , 29 , 38)， 散列表长 11 ，

初始，



线性探测法 d = 9 无冲突

平方探测法

$H0 = (\ key\ )\ \ mod\ \ m$

$Hi = (\ key + d_i\ )\ \ mod\ \ m$

随机探测法 不妨设第一次随机数为 9

注意：增量di应具有“完备性”

即：产生的Hi均不相同，且所产生的s个Hi值能覆盖散列表中所有的地址。

要求：

平方探测时的表长m必为4j+3的质数。

随机探测时的m和di没有公因子。

1. 开放定址法　线性探测再散列

线性探测法：从冲突位置的下一个位置起，依次寻找空的散列地址。

设散列表的长度为m，对于键值key，发生冲突时，寻找空散列地址的公式为：

$$H_i=(H(key)+d_i)\ \%\ m \quad (d_i=1，2，\ldots，m-1)$$

例 1：设关键码集合为 {47, 7, 29, 11, 16, 92, 22, 8, 3}，散列表表长为11，散列函数为$H(key)=key \bmod 11$，用线性探测法处理冲突，散列表的构造过程如下：

0	1	2	3	4	5	6	7	8	9	10
11	22		47	92	16	3	7	29	8	
22			3	3	3		29	8		

堆积：非同义词对同一个散列地址争夺的现象

1. 开放定址法　二次探测法

二次探测法：以冲突位置为中心，跳跃式寻找空的散列地址。

设散列表的长度为m，对于键值key，发生冲突时，寻找空散列地址的公式为：

$$H_i=(H(key)+d_i)\ \%\ m \quad (d_i=1^2,-1^2,2^2,-2^2,\ldots,q^2,-q^2(q\leqslant m/2))$$

例 2：设关键码集合为 {47, 7, 29, 11, 16, 92, 22, 8, 3}，散列表表长为11，散列函数为$H(key)=key$ mod 11，用二次探测法处理冲突，散列表的构造过程如下：

0	1	2	3	4	5	6	7	8	9	10
11	22	3	47	92	16		7	29	8	
22			3	3			29	8		

1. 开放定址法

二次探测法

$$H_i=(H(key)+d_i)\ \%\ m \quad (d_i=1^2,-1^2,2^2,-2^2,\ldots,q^2,-q^2(q\leqslant m/2))$$

相对于线性探测法，**二次探测法**能够在一定程度上减少堆积

设$H(k_i)=6$，$H(k_j)=5$，对于线性探测法：

k_i的探测序列：6, 7, 8, 9, …

k_j的探测序列：5, 6, 7, 8, …

重合之后再不分开

设$H(k_i)=6$，$H(k_j)=5$，对于二次探测法：

k_i的探测序列：6, 7, 5, 10, 2, …

k_j的探测序列：5, 6, 4, 9, 1, …

重合之后很快分开

2. 闭散列表的类定义

闭散列表：用开放定址法处理冲突得到的散列表

```
const int MaxSize = 100;
class HashTable1
{
public:
    HashTable1( );
    ~HashTable1( );
    int Insert(int k);
    int Delete(int k);
    int Search(int k);
private:
    int H(int k );
    int ht[MaxSize];
};
```

```
HashTable1 :: HashTable1( )
{
    for (int i = 0; i < MaxSize; i++)
        ht[i] = 0;
}

HashTable1 :: ~HashTable1( )
{

}
```

3. 散列表查找伪代码

算法：Search

输入：闭散列表ht[ ]，待查值k

输出：如果查找成功，则返回记录的存储位置，否则返回查找失败的标志-1

1. 计算散列地址 j；
2. 探测下标i初始化：i = j；
3. 执行下述操作，直到 ht[i] 为空：
   3.1 若 ht[i] 等于 k，则查找成功，返回记录在散列表中的下标；
   3.2 否则，i 指向下一单元；
4. 查找失败，返回失败标志-1；

4. 散列表查找的实现

```
int HashTable1 :: Search(int k)
{
    int i, j = H(k);                          //计算散列地址
    i = j;                                    //设置比较的起始位置
    while (ht[i] != 0)
    {
        if (ht[i] == k) return i;                    //查找成功
        else i = (i + 1) % MaxSize;                  //向后探测一个位置
    }
    return -1;                                //查找失败
}
```

5. 双散列探测

再哈希法 （双散列探测）

定义双重哈希函数。

$H(key)$

$R(key)$

若$H(key)$出现冲突，则再使用$R(key)$求取哈希地址。

6. 拉链法/链地址法

开散列表：用拉链法处理冲突得到的散列表。

思想：将具有同一散列地址的记录存储在一条线性链表（**同义词子表**）中。

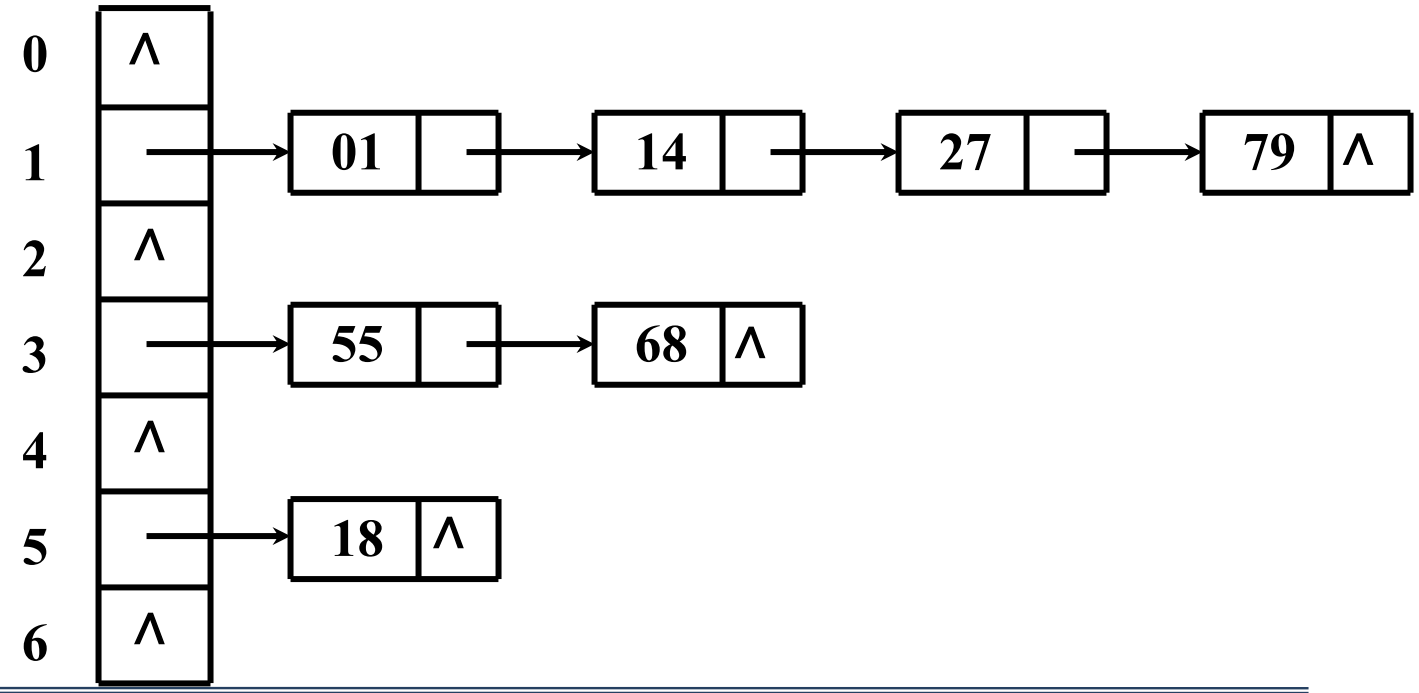
（1）计算散列地址：$j = H(key)$

（2）将*key*对应的记录插入到同义词子表 j 中；

例，除留余数法中，　$p = 13$

设关键字为 (18 , 14 , 01 , 68 , 27 , 55 , 79)

哈希地址为 (5 , 1 , 1 , 3 , 1 , 3 , 1)



7.4 散列表的查找技术

7-4-4　散列查找性能分析

1. 衡量方法

决定散列表查找的ASL的因素：

1、选用的**散列函数**

2、选用的**处理冲突的方法**

3、散列表饱和的程度，**装载因子α=n/m值的大小。**

n: 表中填入的记录数

m: 散列表的长度

一般情况下，可以认为选用的散列函数是“均匀”的，则在讨论ASL时可以不考虑它的因素。

散列表的ASL是处理冲突方法和装载因子的函数。

2. 查找性能分析

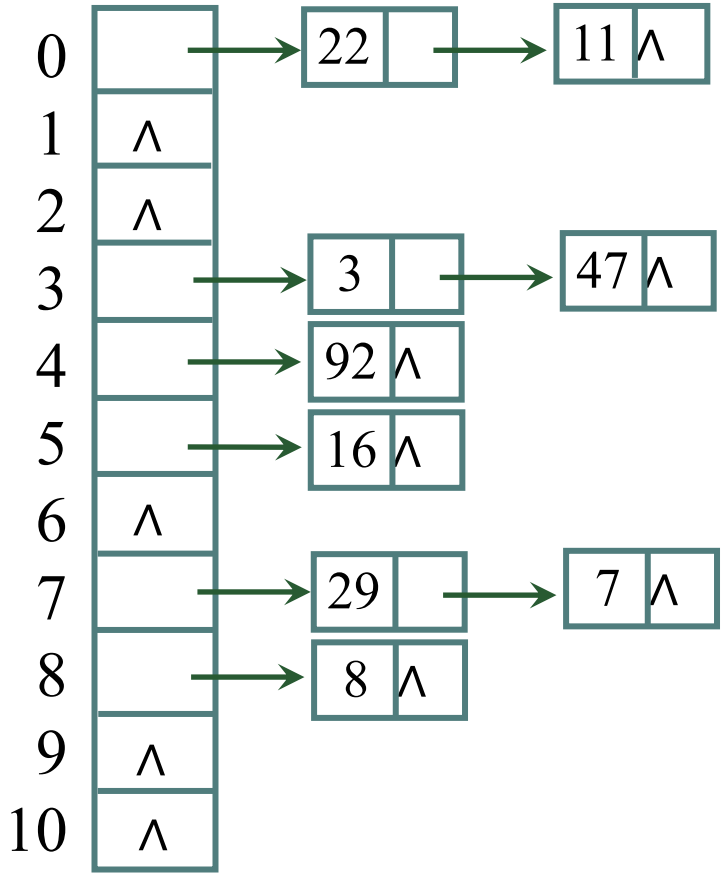
例 1：设关键码集合为 {47, 7, 29, 11, 16, 92, 22, 8, 3}，散列表表长为11，散列函数为$H(key)=key \bmod 11$，用线性探测法和拉链法处理冲突，分析查找性能。

0	1	2	3	4	5	6	7	8	9	10
11	22		47	92	16	3	7	29	8	
22			3				29	8		

查找成功的平均查找长度是：　(1×5+2×3+4×1)/9 = 15/9

2. 查找性能分析

例 1：设关键码集合为 {47, 7, 29, 11, 16, 92, 22, 8, 3}，散列表表长为11，散列函数为$H(key)=key \bmod 11$，用线性探测法和拉链法处理冲突，分析查找性能。



查找成功的平均查找长度是：

$(1\times6+2\times3)/9 = 12/9$

2. 查找性能分析

平均查找长度 / 处理冲突的方法	查找成功时	查找不成功时
线性探测法	$\frac{1}{2}\left(1+\frac{1}{1-\alpha}\right)$	$\frac{1}{2}\left(1+\frac{1}{(1-\alpha)^2}\right)$
二次探测法	$-\frac{1}{\alpha}\ln(1+\alpha)$	$\frac{1}{1-\alpha}$
拉链法	$1+\frac{\alpha}{2}$	$\alpha+e^{-\alpha}$

散列表的平均查找长度是装载因子 α 的函数，
而不是查找集合中记录个数 n 的函数——$O(1)$！

7.4 散列表的查找技术

3. 闭散列表与开散列表比较

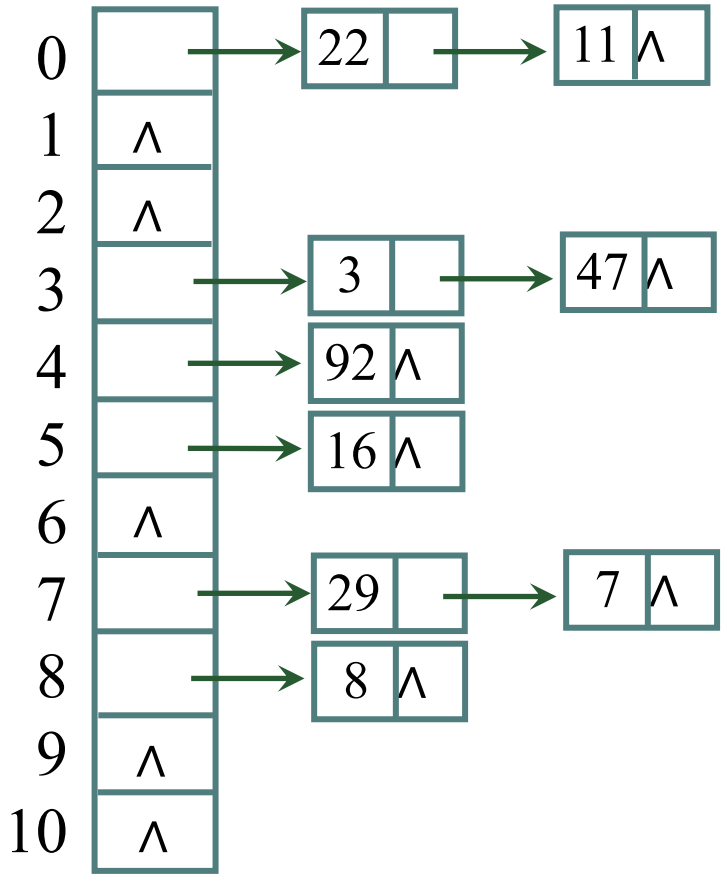
空间比较

- 闭散列表：
  - 受数组空间限制，需要考虑存储容量
  - 存储效率较高
- 开散列表：
  - 没有记录个数的限制，但子表过长会降低查找效率
  - 指针的结构性开销

时间比较

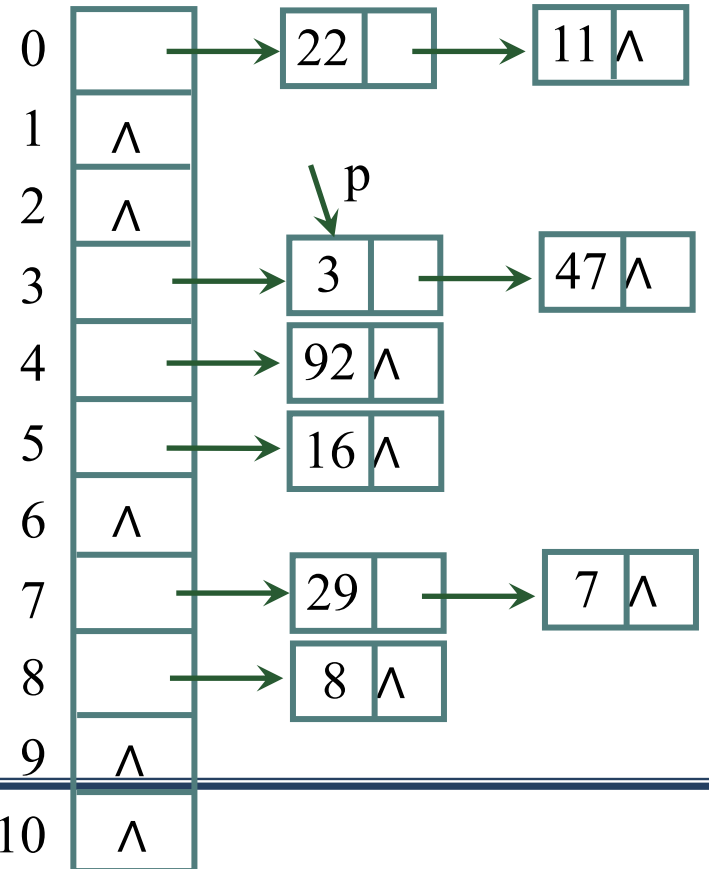
- 闭散列表：
  - 有堆积现象，降低查找效率
  - 仅适用于静态查找
- 开散列表：
  - 不会产生堆积现象，效率较高
  - 适用于静态查找和动态查找

0	1	2	3	4	5	6	7	8	9	10
11	22		47	92	16	3	7	29	8	
22			3				29	8		



4. 开散列表的删除操作

例 2：设关键码集合{47, 7, 29, 11, 16, 92, 22, 8, 3}，散列表表长为 11，散列函数为$H(key)=key \bmod 11$，用拉链法处理冲突构造开散列表，删除元素 47。




```
int HashTable2 :: Delete(int k)
{
    int j = H(k);
    Node<int> *p = ht[j], *pre = nullptr;
    while ((p != nullptr) && (p->data != k)
    {
        pre = p;  p = p->next;
    }
    if (p != nullptr) {
        if (pre == nullptr) ht[j] = p->next;
        else pre->next = p->next;
        return 1;
    } else
        return 0;
}
```

5. 闭散列表的删除操作

例 3：设关键码集合{47, 7, 29, 11, 16, 92, 22, 8, 3}，散列表表长为 11，散列函数为$H(key)=key \bmod 11$，用线性探测法处理冲突构造闭散列表，删除元素 47。

0	1	2	3	4	5	6	7	8	9	10
11	22		~~47~~	92	16	3	7	29	8	
22			3	3	3		29	8		

删除 47、92、16 ⇨ 断开探测序列 ⇨ 查找元素 3 时会得到失败信息

做删除标记，表示该位置有元素被删除，查找时遇到标记要继续进行

⇩

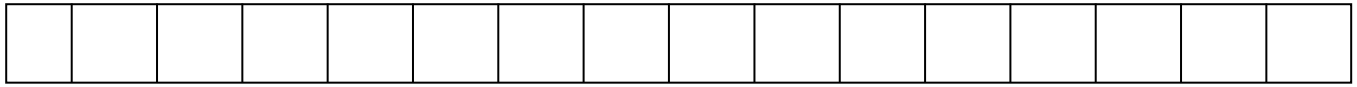
修改查找、插入算法，删除算法比较复杂

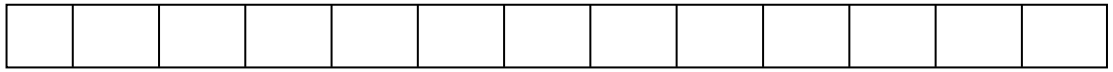
小结

1. 熟练掌握哈希表的构造方法以及处理冲突的方法
2. 深刻理解哈希表与其他结构的表的实质性的差别

作业

假设哈希函数H(key)=key MOD 13，采用开放定址法中的线性探测再散列来处理冲突，请用关键字序列(37,25,8,98,73,12,46,59,30)来构造哈希表，将各关键字按地址分别填在下面所示的两种长度不同的表中。





本章总结

1. 理解**查找**的基本概念和**算法评价方法**
2. 掌握**顺序查找**和**折半查找**实现与性能分析方法
3. 掌握**二叉排序树**的构建、查找、插入、删除原理
4. 掌握二叉排序树的实现方法和性能分析方法
5. 掌握**平衡二叉树**的相关概念和不同类型平衡调整方法
6. 了解平衡二叉树的性能分析方法
7. 掌握**B树**的定义、查找、插入和删除方法
8. 掌握**散列表的构造方法**以及**处理冲突**的方法

Thank You !

Q & A