查　　找

-散列查找

学习目标

理解查找的基本概念

掌握常用查找算法及其性能分析

掌握线性表查找（顺序、折半查找及其判定树）

掌握树表查找（二叉排序树、平衡二叉树）

掌握散列表的构造和查找

讲什么？

- 散列技术的基本思想
- 散列函数——设计原则、几种常见的散列函数
- 冲突的处理方法——开放定址法
- 冲突的处理方法——拉链法
- 散列查找的性能分析
- 闭散列表和开散列表的比较

回顾查找技术

查找操作要完成什么任务?

待查值k ⇨ 确定 k 在存储结构中的位置

前面学过哪些查找技术?这些查找技术有什么共性呢?

(1) 顺序查找

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

(2) 折半查找

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

回顾查找技术

查找操作要完成什么任务？

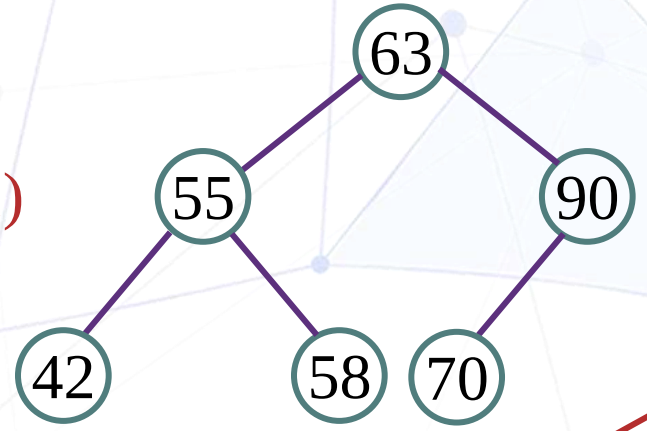
待查值k ⇨ 确定 k 在存储结构中的位置

有哪些查找技术？这些查找技术有什么共性呢？

（1）顺序查找　==、!=　$O(n)$

（2）折半查找　<、==、>　$O(\log_2 n)$

（3）二叉排序树查找　$O(n) \sim (\log_2 n)$



$\Omega(\log_2 n)$

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

查找技术的分类

能否不用比较，通过关键码能够直接确定存储位置？

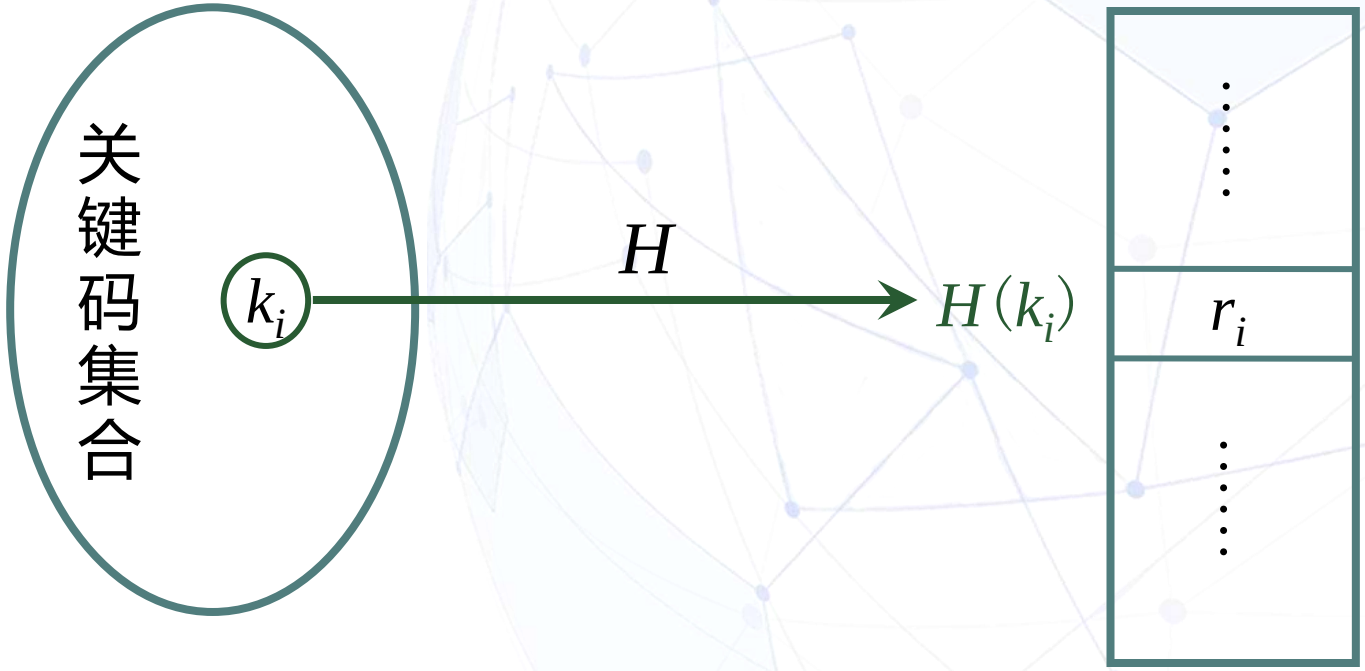
在存储位置和关键码之间建立一个确定的对应关系

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

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。

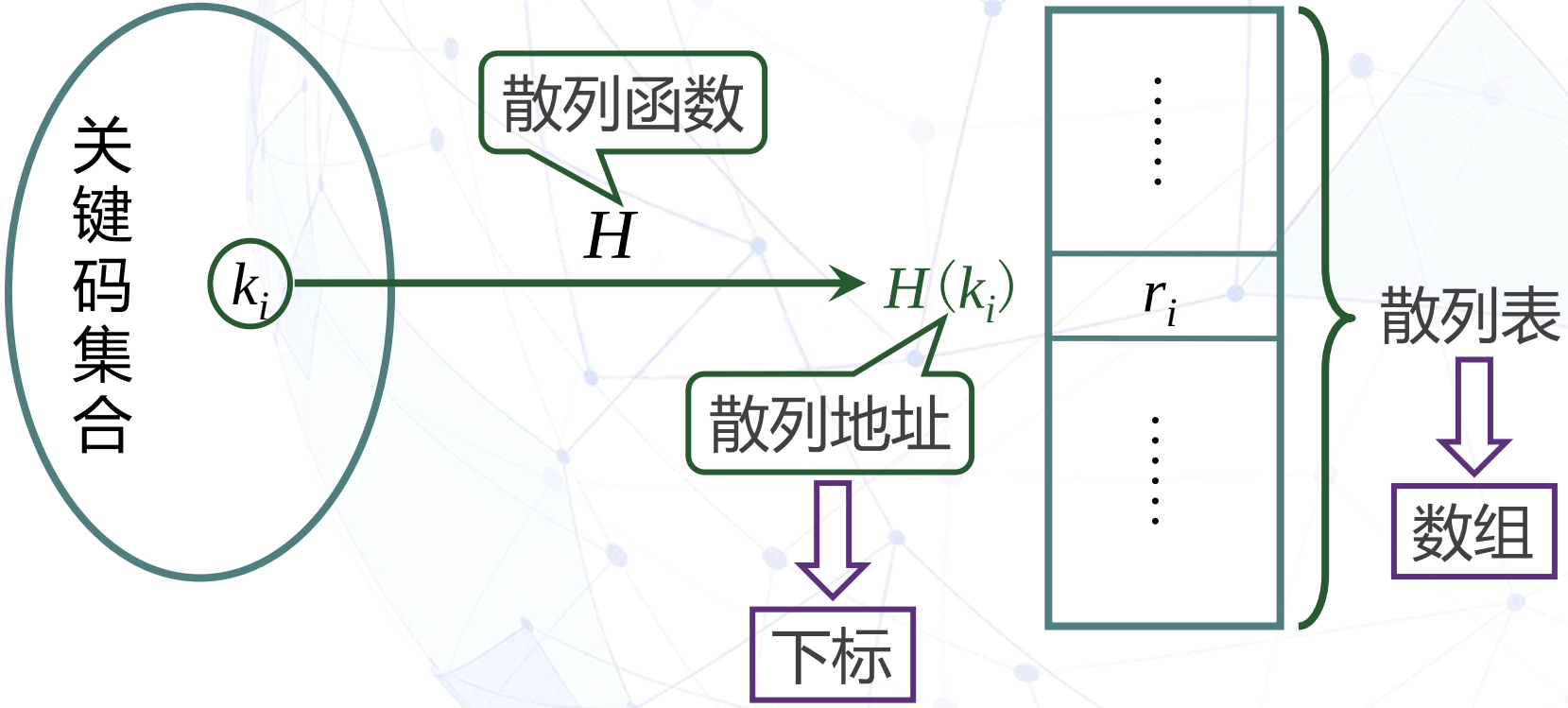


散列技术名称

Hash ⇨ 哈希 ⇨ 杂凑 ⇨ 散列

散列的基本概念

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



散列的关键问题

如何设计散列函数?

数据元素序列（21，23，39，9，25，11），若规定每个元素K的存储地址为H(K)=k，则其存储结构如下：

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
									9		11										21		23		25															39

查找k=25，根据H(K)一下可找到

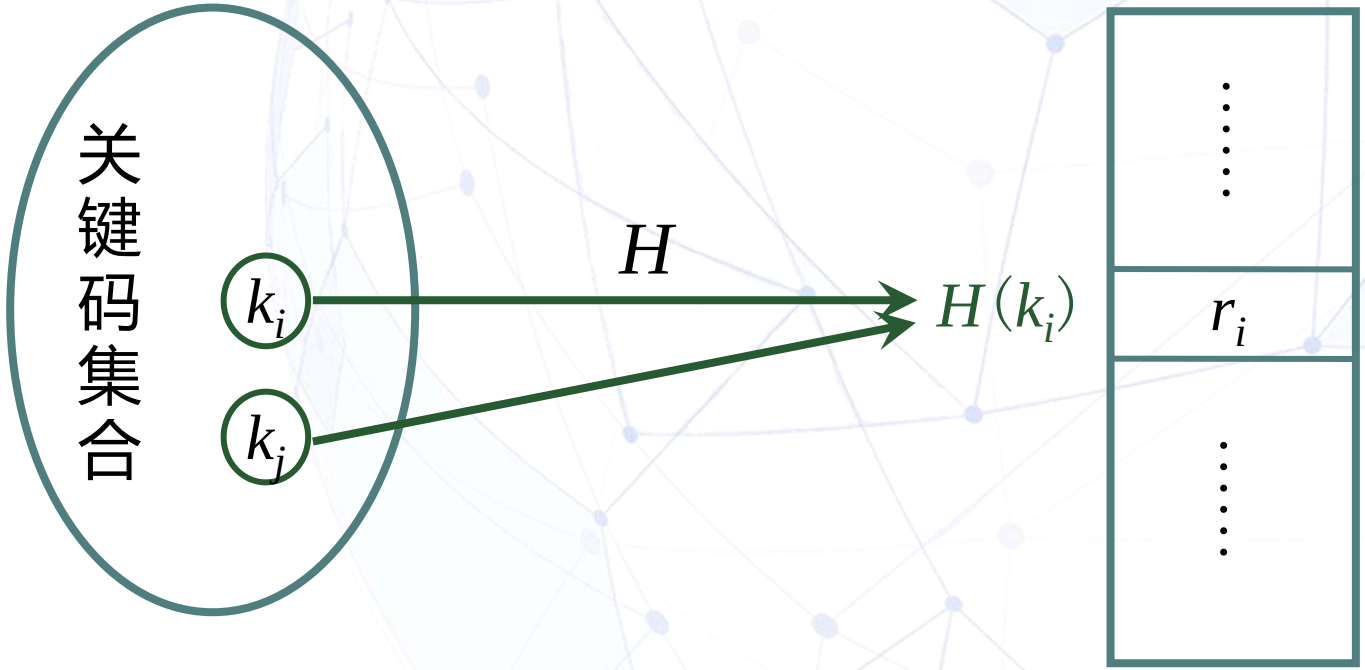
- 优点：查找快速
- 缺点：空间效率低

设计更合理的散列函数

散列的关键问题

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

同义词(synonym)：k_i和k_j相对于H称做同义词。



例如：对于如下 9 个关键字

{Zhao, Qian, Sun, Li, Wu, Chen, Han, Ye, Dei}

字符的次序函数

设 哈希函数 H(key) = ⌊(Ord(第一个字母)/2⌋

0	1	2	3	4	5	6	7	8	9	10	11	12	13
	Chen	Dei		Han		Li		Qian	Sun		Wu	Ye	Zhao

问题：若添加关键字 Zhou，怎么办？

如何解决冲突?

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。

(3) 哈希函数对于某一个关键字，每次计算都应得到相同的结果。

(4)通常对关键字进行分割，混合，累加等多种运算。

不失一般性，以关键字是int类型进行讲解

常见的散列函数

直接定址法

散列函数是关键码的线性函数，即：

$$H(key) = a \times key + b \quad (a，b为常数)$$

例 1：关键码集合为{10, 30, 50, 70, 80, 90}，选取的散列函数为$H(key)=key/10$，散列表构造过程如下：

0	1	2	3	4	5	6	7	8	9
	10		30		50		70	80	90

适用于：事先知道关键码，关键码集合不很大且连续性较好

常见的散列函数

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 2：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

适用于：事先不知道关键码的分布且关键码的位数不是很多

常见的散列函数

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

所有含质因子 7 的关键字均映射到“7 的倍数”的地址上，从而增加了“冲突”的可能。

小于等于表长（最好接近表长）的最小素数或不包含小于20质因子的合数，一般可取(4*K+3)

常见的散列函数

除留余数法

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

例 3：散列表长为15，设计除留余数法的散列函数。

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

适用于：最简单、最常用，不要求事先知道关键码的分布


开放定址法
散列技术关键--
处理冲突的方法
拉链法

开放地址法

- 开放定址法的基本思想
- 线性探测法
- 二次探测法

开放定址法

开放定址法(Open addressing)如何处理冲突?

造表时，对于给定的关键码*key*执行下述操作：

（1）计算散列地址：$j = H(key)$

（2）如果地址 j 的存储单元没有存储记录，则存储*key*对应的记录；

（3）如果在地址 j 发生冲突，则寻找一个空的散列地址，存储*key*对应的记录；

闭散列表：用开放定址法处理冲突得到的散列表

如何寻找一个空的散列地址呢?

（1）线性探测法；（2）二次探测法；（3）随机探测法……

闭散列表的类定义

```
const int MaxSize = 11;
class HashTable1
{
public:
    HashTable1( );
    ~HashTable1( );
    int Insert(int k);
    int Delete(int k);
    int Search(int k);
    void Print( );
private:
    int H(int k);
    int ht[MaxSize];
};
```

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


}
void HashTable1 :: Print( )
{
    for (int i = 0; i < MaxSize; i++)
        cout << ht[i] << "\t";
}
```

线性探测法

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Clustering（聚集）

伪代码

约定： (1)静态查找 (2)闭散列表不会发生上溢

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；

算法描述

```
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)          //构造函数中初始化赋值了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
}
```

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$H(key)=key \bmod 11$，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		

二次探测法

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

处理冲突的方法——拉链法

拉链法

拉链法如何处理冲突?

对于给定的关键码*key*执行下述操作:

（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;

同义词子表：所有散列地址相同的记录构成的单链表。

开散列表：用拉链法处理冲突得到的散列表。

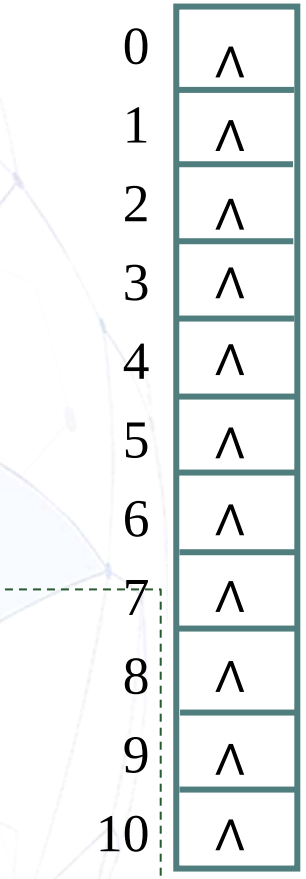
开散列表中存储同义词子表的头指针，开散列表不会出现堆积现象

开散列表的类定义

```
const int MaxSize = 11;
class HashTable2
{
public:
    HashTable2( );
    ~HashTable2( );
    int Insert(int k);
    int Delete(int k);
    Node<int> * Search(int k);
    void Print( );
private:
    int H(int k);
    Node<int> * ht[MaxSize];
};
```

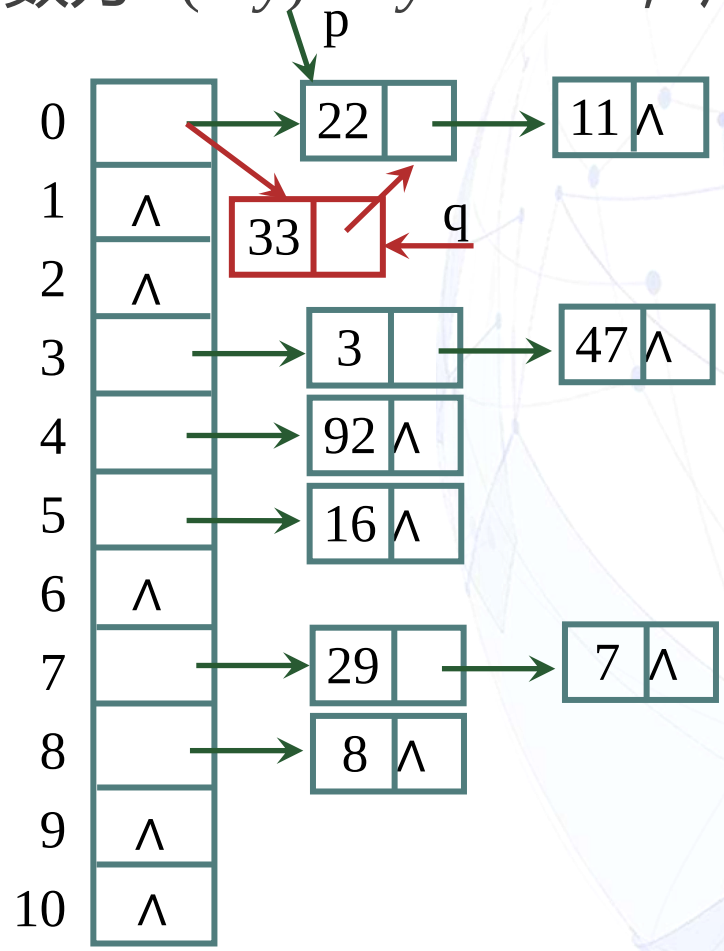
- 单链表的结点结构?
- 单链表类的析构函数?

```
HashTable2 :: HashTable2( )
{
    for (int i = 0; i < MaxSize; i++)
    {
        ht[i] = nullptr;
    }
}
```



构造过程

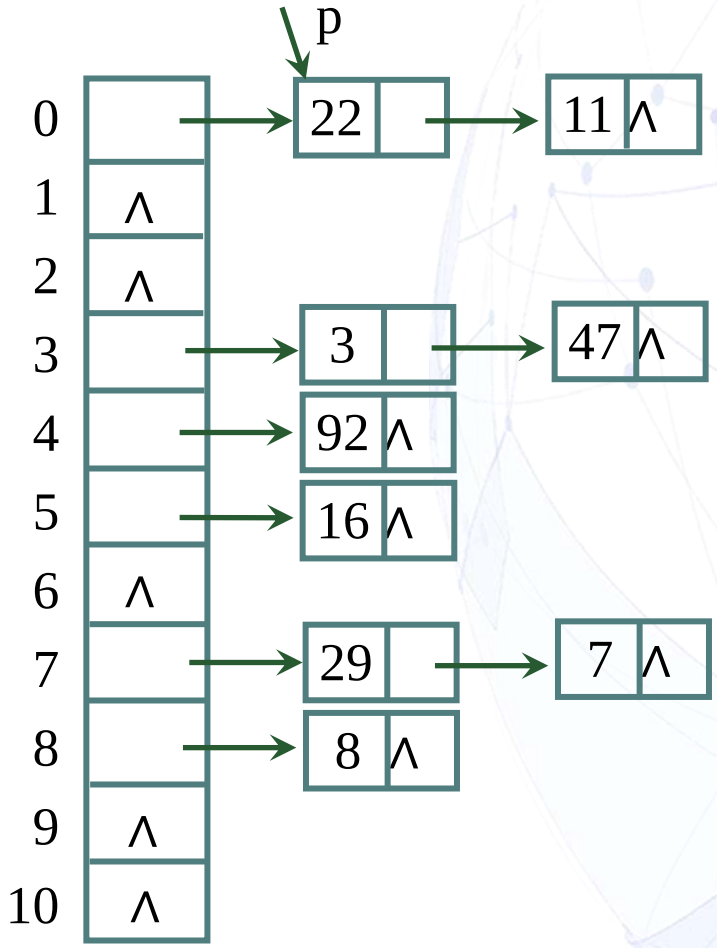
例 1：设关键码集合为 {47, 7, 29, 11, 16, 92, 22, 8, 3}，散列表表长为11，散列函数为$H(key)=key \bmod 11$，用拉链法处理冲突，散列表的构造过程如下：




```
j = H(k);
Node<int> *p = ht[j];
while (p != nullptr)
{
    if (p->data == k) break;  //插入时要判断是否已经有相同关键字
    else p = p->next;
}
if (p == null) {//无该关键字则插入
    q = new Node<int>;   q->data = k;
    q->next = ht[j]; ht[j] = q;
}
```

查找算法

约定：（1）静态查找；（2）子表无长度限制




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

性能分析

- 散列技术的初衷是什么？能完全实现吗？

  不用进行比较，通过计算得到存储地址

- 散列技术的查找性能取决于什么？

  产生冲突后，仍然是给定值与关键码进行比较

- 影响冲突产生的因素有什么？

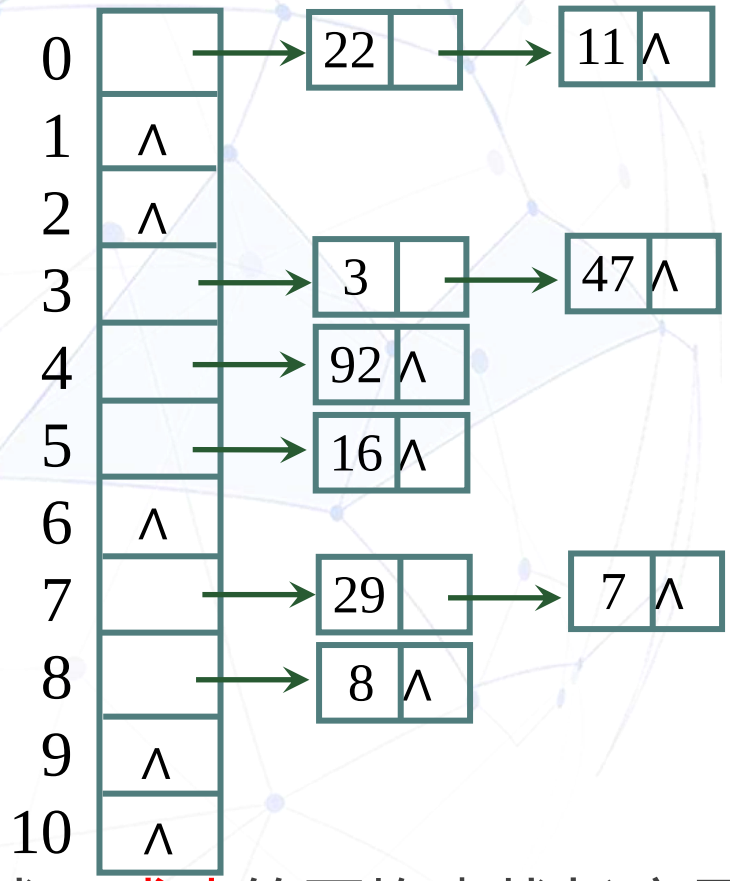
  （1）散列函数是否均匀

  （2）处理冲突的方法

性能分析

例 4：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：

$(1\times5+2\times3+4\times1)/9 = 15/9$

查找不成功的平均查找长度是：

$(3+2+1+8+7+6+5+4+3+2+1)/11 = 42/11$

查找成功的平均查找长度是：

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

查找不成功的平均查找长度是：

$(1\times3+2\times3)/11 = 9/11$

衡量方法

- 散列技术的查找性能取决于什么？

  产生冲突后，仍然是给定值与关键码进行比较

- 影响冲突产生的因素有什么？

  (1) 散列函数是否均匀

  (2) 处理冲突的方法

  (3) 散列表的装填因子$\alpha = \dfrac{\text{表中填入的记录数}}{\text{散列表的长度}}$

装填因子α标志着散列表装满的程度,表中记录越多，α越大，产生冲突的可能越大

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	$\frac{1}{2}(1+\frac{1}{1-\alpha})$	$\frac{1}{2}(1+\frac{1}{(1-\alpha)^2})$
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填因子 α 的函数，与记录个数n无关，查找性能是$O(1)$!

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：

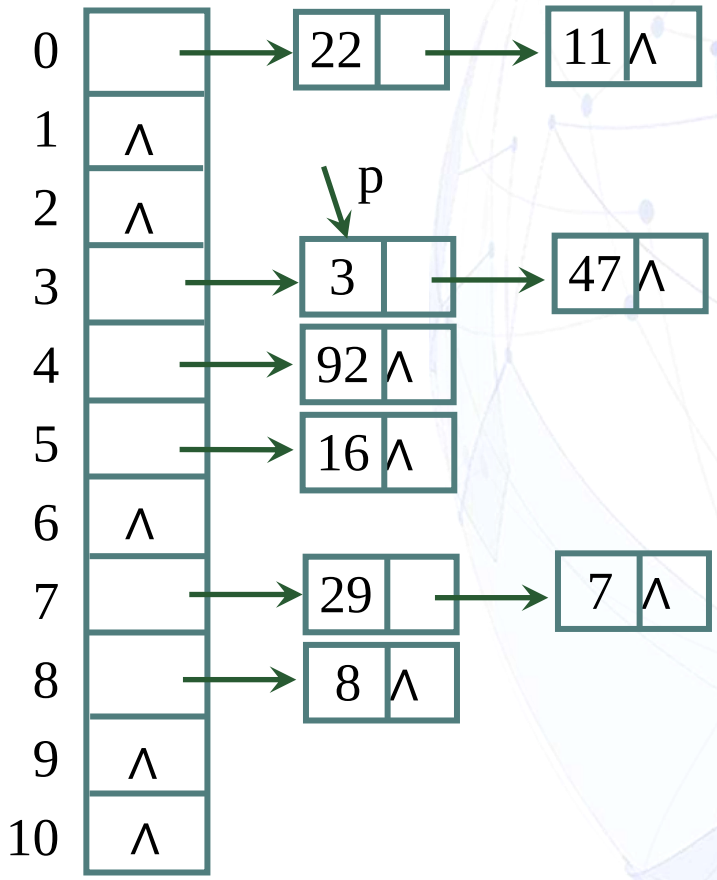
- 有堆积现象，降低查找效率
- 仅适用于静态查找

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

开散列表的删除操作

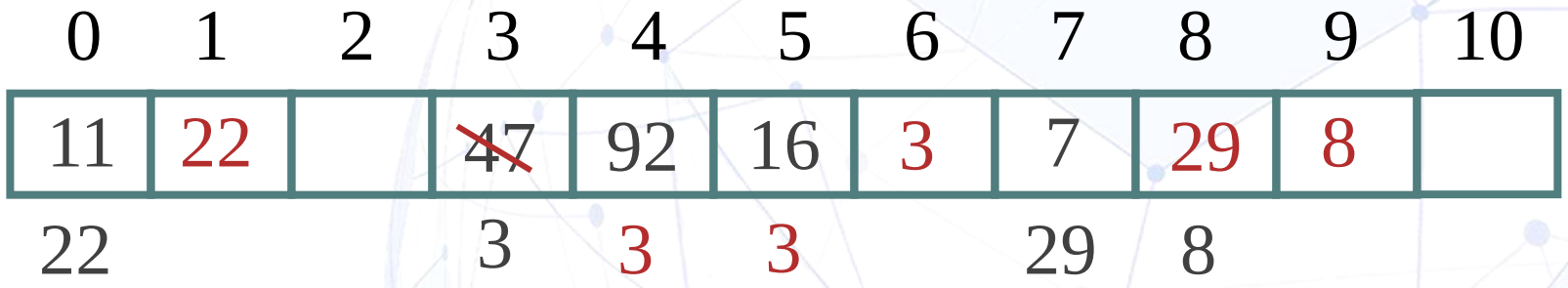
例 1：设关键码集合{47, 7, 29, 11, 16, 92, 22, 8, 3}，散列表表长为 11，散列函数为$H(key)=key \bmod 11$，用拉链法处理冲突构造开散列表，删除元素 47。




```
int HashTable2 :: Delete(int k)
{
    int j = H(k);
    Node *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;
        delete p;  return 1;
    } else
        return 0;
}
```

闭散列表的删除操作

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$H(key)=key \bmod 11$，用线性探测法处理冲突构造闭散列表，删除元素 47。



删除 47 ⇨ 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

单选题 5分

设置

1. 散列技术无需进行比较，经过计算可直接得到关键码所在的存储地址。

A 正确

B 错误

提交

单选题 5分

设置

2. 在散列技术的应用中，冲突一定会发生，因此，查找需要进行少量比较。

A 正确

B 错误

提交

单选题 5分

设置

3. 在散列表中，一定不会出现相同关键码。

A 正确

B 错误

提交

单选题 5分

设置

4. 散列函数的设计原则是计算简单，因此一定都是初等函数。

A 正确

B 错误

提交

单选题 5分

设置

5. 设散列表长为100，散列函数为除留余数法，除数是（ ）较好。

A 100

B 97

C 91

D 89

提交

单选题 5分

设置

6. 散列技术的查找效率主要取决于散列函数和处理冲突的方法。

A 正确

B 错误

提交

单选题 5分

设置

7. 当装填因子小于1时，向散列表中存储记录时不会引起冲突。

A 正确

B 错误

提交

单选题 5分

设置

8. 在闭散列表中有可能出现堆积，开散列表一定不会出现堆积。

A 正确

B 错误

提交

主观题 10分

设置

9. 对于查找集合{16, 24, 7, 9, 13, 20}，设散列表长为8，散列函数为H(key) = key mod 7，请分别画出线性探测法和链地址法处理冲突构造的散列表。

正常使用主观题需2.0以上版本雨课堂

作答

研究报告作业

- 1、字数不少于1500字。
- 2、内容与数据结构课程相关，可以涉及但不限于：
- （1） 数据结构实际应用
- （2） 算法的比较、优化及实现
- （3） 某数据结构在C++中的内部实现
  - C++中的哈希表
  - C++中的线性表（栈、队列）
  - C++中的查找
  - C++中的排序
- （4） 问题探究
- （5） 知识拓展
- （6） 外文翻译
- 3、文后附上每位同学对文章的贡献。
- 4、文件格式：pdf或word文件