

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第6章 查找 Slide. 6 - 1

第6章 查找

6.0 术语和约定

6.1 线性查找

6.2 折半查找

6.3 分块查找

6.4 二元查找树

6.5 散列法

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 2

6.0 术语和约定

- 查找表：由同一类型的数据元素（或记录）构成的集合。
- 关键字：数据元素中某个或某些数据项的值，用以标识一个数据元素（或记录）。
- 查找：根据给定的值，在查找表中确定一个关键字值等于给定值的数据元素或记录。若找到与该值匹配的关键字，则查找成功；否则，查找失败。
- 查找的分类——查找的环境
 - 静态查找：查找+提取数据元素属性信息
在静态环境下，被查找的数据集合经查找之后并不改变，就是说，既不插入新的记录，也不删除原有记录。
 - 动态查找：查找+（插入或删除元素）
在动态环境下，被查找的数据集合经查找之后可以改变，就是说，可以插入新的记录，也可以删除原有记录。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 3

- 定义在查找表上的主要操作
- **SEARCH(k, F)**：在数据集合（查找表、文件）F 中查找关键字值等于k 的数据元素（记录）。若查找成功，则返回包含 k 的记录的位置；否则，返回一个特定的值。
- **INSERT(R, F)**：在动态环境下的插入操作。在F 中查找记录 R，若查找不成功，则插入R；否则不插入R。
- **DELETE(k, F)**：在动态环境下的删除操作。在 F 中查找关键字值等于k 的数据元素（记录）。若查找成功，则删除关键字值等于k 的记录，否则不删除任何记录。
- 查找分类
 - 基于关键字比较的查找
 - 线性查找
 - 折半查找
 - 分块查找
 - 二元查找树
 - 基于关键字存储位置的查找（散列法）
 - 动态查找
 - 静态查找
 - 内查找
 - 外查找

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 4

- 查找表结点（数据元素、记录）的类型定义

```
struct records{    keytype key;    fields other;};
```
- 查找的性能指标——平均查找长度
定义 为确定记录在查找表中的位置，需要把给定值与关键字进行比较的次数的期望值称为查找算法在查找成功时的平均查找长度(Average Search Length)。
设 P_i 为查找表中第 i 个记录的概率， $\sum P_i=1$ ， C_i 为查找第 i 个记录所进行的比较次数，则
$$ASL = \sum_{i=1}^n P_i \cdot C_i$$
在等概率情况下，即 $P_i = 1/n$ 时，
$$ASL = \frac{1}{n} \sum_{i=1}^n C_i$$

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法		第6章 查找	Slide. 6 - 5
6.1 线性查找			
一、线性查找			
把数据集合或文件组织成线性表，从线性表的一端开始，依次比较其中每个数据元素或记录的关键字。			
二、顺序表上的查找——适合于静态查找			
1. 数据结构定义（类型定义）			
<pre>typedef records LIST[maxsize]; LIST F;</pre>			
2. SEARCH操作的实现			
3. INSERT操作的实现			
4. DELETE操作的实现			
不适合			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第6章 查找	Slide. 6 - 6
<pre>int SEARCH(keytype k, int last, LIST F) /* 在F[1]...F[last]中查找关键字为k的记录，若 找到，则返回该记录所在的下标，否则返回 0 */ { int i; F[0].key = k; /*F[0]为伪记录或哨兵*/ i = last; while (F[i].key != k) i = i - 1; return i; } /* ASL=(n+1)/2, 时间复杂性 O(n) */</pre>			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第6章 查找	Slide. 6 - 7
三、单链表上的查找——既适合于静态也适合于动态查找			
1. 数据结构定义（类型定义）			
<pre>struct celltype { records data ; celltype * next ; }; typedef celltype * LIST ;</pre>			
2. 查找操作的实现			
3. INSERT操作的实现			
4. DELETE操作的实现			
<pre>LIST SEARCH(keytype k, LIST F) /*在不带表头的单向链表中查找关 键字为k的记录，返回其指针*/ { LIST p ; p=F; while (p!=NULL) if (p->data.key == k) return p ; else p = p->next ; return p ; }/*ASL=(n+1)/2, 时间复杂性 O(n)*/</pre>			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第6章 查找	Slide. 6 - 8																						
6.2 折半查找——有序表上的查找																									
一、折半查找（也称二分查找）的要求																									
■查找表（被查找的数据集合）必须采用顺序式存储结构； ■查找表中的数据元素（记录）必须按关键字有序。																									
 <table border="1"><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td></tr><tr><td>05</td><td>13</td><td>19</td><td>21</td><td>37</td><td>56</td><td>64</td><td>75</td><td>80</td><td>88</td><td>92</td></tr></table> F[0].key ≤ F[1].key ≤ F[2].key ≤ ... ≤ F[last].key 或 F[0].key ≥ F[1].key ≥ F[2].key ≥ ... ≥ F[last].key				0	1	2	3	4	5	6	7	8	9	10	05	13	19	21	37	56	64	75	80	88	92
0	1	2	3	4	5	6	7	8	9	10															
05	13	19	21	37	56	64	75	80	88	92															
注意：折半查找只适合于静态查找！																									
哈尔滨工业大学 计算机科学与技术学院 张岩																									

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第6章 查找 Slide. 6 - 9

6.2 折半查找——有序表上的查找

二、折半查找算法的要点

- 初态：令 low , up 分别表示查找范围的上、下界，初始时 $low = 0$, $up = last$;
- 折半：令 $mid = (low+up)/2$, 取查找范围中间位置元素下标;
- 判断：
 - 若 $F[mid].key == k$, 查找成功, 返回 mid ;
 - 若 $F[mid].key > k$, low 不变, 调整 $up = mid - 1$, 查找范围缩小一半;
 - 若 $F[mid].key < k$, 调整 $low = mid + 1$, up 不变, 查找范围缩小一半;
- 重复 2~3 步。当 $low > up$ 时, 查找失败, 返回 -1。

0	1	2	3	4	5	6	7	8	9	10	
F	05	13	19	21	37	56	64	75	80	88	92

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 10

6.2 折半查找——有序表上的查找

举例

0	1	2	3	4	5	6	7	8	9	10	
k = 21	05	13	19	21	37	56	64	75	80	88	92

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 11

6.2 折半查找——有序表上的查找

三、算法的实现

```
typedef records LIST[maxsize];
int binary_search(keytype k, LIST F)
{
    int low, up, mid;
    low = 0; up = last;
    while (low <= up) {
        mid = (low + up) / 2;
        if (F[mid].key == k)
            return mid;
        else if (F[mid].key > k)
            up = mid - 1;
        else
            low = mid + 1;
    }
    return -1;
} /* F必须是顺序式有序表（此处为增序）*/

```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 12

6.2 折半查找——有序表上的查找

四、性能分析

当 n 很大时, $ASL_{bs} = \log_2(n+1)-1$ 作为查找成功时的平均查找长度。

在查找不成功和最坏情况下查找成功所需关键字的比较次数都不超过判定树的高度。因为判定树的中度小于 2 的结点只能出现在下面两层上, 所以 n 个结点的判定树高度和 n 个结点的完全二元树的高度相同, 即 $\lceil \log_2(n+1) \rceil$ 。由此可见, 折半查找的最坏性能与平均性能相当接近。时间复杂性: $O(\log_2 n)$

当 n 很大时, $ASL_{bs} = \log_2(n+1)-1$ 作为查找成功时的平均查找长度。
 $n=2^h-1$ ($h=\log_2(n+1)$)
 $ASL_{bs} = \sum_{i=1}^n P_i C_i$ /* $P_i=1/n$ */
 $= 1/n \cdot \sum_{j=1}^h j \cdot 2^{j-1}$ /* $S=2S - S$ */
 $= (n+1)/n \cdot \log_2(n+1)-1$

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法

第6章 查找

Slide. 6 - 13

6.3 分块查找——线性查找+折半查找

一、基本思想

- 首先将表中的元素均匀地分成若干块，每一块中的元素的任意排列，而各块之间要按顺序排列。

若按从小到大的顺序排列，则第一块中的所有元素的关键字都小于第二块中的所有元素的关键字，第二块中的所有元素的关键字都小于第三块中的所有元素的关键字，如此等等

- 然后再建一个线性表，用以存放每块中最大（或最小）的关键字，此线性表成为索引表。是一个有序表

IX	0	1	2	
	22	44	74	

索引表

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
F	22	12	13	9	8	33	42	44	38	24	48	60	58	74	47

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法

第6章 查找

Slide. 6 - 14

6.3 分块查找——线性查找+折半查找

二、算法要点

在带索引的线性表中查找已知关键字为k的记录，

- 首先查找索引表，确定k可能出现的块号；
- 然后到此块中进行顺序查找。

	0	1	2												
IX	22	44	74	索引表											
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
F	22	12	13	9	80	33	42	44	38	24	48	60	58	74	47

三、算法实现

```
typedef records LIST[maxsize] /* 线性表—主表 */  
typedef keytype INDEX[maxblock] /* 线性表—索引表 */
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法

第6章 查找

Slide. 6 - 15

```
int index_search( k, last, blocks, ix, F, L )
keytype k; int last, blocks ; INDEX ix ,LIST F; int L ;
{ int i=0, j ;
  while ( (k > ix[i])&&( i < blocks) ) /*查索引表，确定k所在块i*/
    i++ ;
  if( i<blocks ) {
    j = i*L; /* 第i块的起始下标 */
    while( ( k != F[j].key )&&( j <= (i+1)*L-1 )&&( j < last ) )
      j = j + 1 ;
    if ( k == F[ j ].key ) return j ; /* 查找成功 */
  }
  return -1 ; /* 查找失败 */
}
```

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法

第6章 查找

Slide. 6-10

6.3 分块查找——线性查找+折半查找

四、性能分析

1、平均查找长度

设长度为 n 的表分成 b 块，每块长度为 L ，则 $b = \lceil n / L \rceil$ ，又设表中每个元素的查找概率相等，则每块查找的概率为 $1/b$ ，块中每个元素的查找概率为 $1/L$ 。于是，

索引表的查找平均长度为： $ASL_{ix} = \sum_{i=1}^b P_i \cdot C_i = \frac{1}{b} \sum_{i=1}^b i$

块内的查找平均长度为： $ASL_{blk} = \sum_{j=1}^L P_j \cdot C_j = \frac{1}{L} \sum_{j=1}^L j$

所以分块查找平均长度为：

$$ASL(L) = ASL_{ix} + ASL_{blk} = (b + 1)/2 + (L + 1)/2 = (n/L + L)/2 + 1$$

令 $ASL'(L) = 0$ ，知当 $L = \sqrt{n}$ 时， $ASL(L) = \sqrt{n} + 1$ （最小值）局限性和改进

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 17

6.3 分块查找——线性查找+折半查找

四、性能分析

- 2、局限性和改进
 - 只适合静态查找：
 - 改进：1) 在索引表中保存各块的下标范围，此时不必均匀分块；
2) 各块存放在不同的向量（一维数组）中；

五、动态分块的实现

元素组织成一个链表。

IX

1、带索引表的链表 2、算法的实现

数据结构定义
三个算法的实现

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 18

6.4 二元查找树——在树上的查找

一、二元查找树——二元搜索树、二元分类（排序）树

二元查找树或者是空树，或者是满足下列性质的二元树：

- 若它的左子树不空，则左子树上所有结点的关键字的值都小于根结点关键字的值；
- 若它的右子树不空，则右子树上所有结点的关键字的值都大于根结点关键字的值；
- 它的左、右子树本身又是一个二元查找树。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 19

6.4 二元查找树——在树上的查找

- 二元查找中任意一个结点X的关键字，都大于(小于)其左(右)子树中任意结点Y的关键字，因此各结点的关键字互不相同；
- 按中序遍历二元查找树所得的中序序列是一个递增的有序序列；
- 同一个数据集合，可按关键字表示成不同的二元查找树，即同一数据集合的二元查找树不唯一；但中序序列相同。
- 每个结点X的右子树的最左结点Y，称为X的继承结点。有如下性质：(1)在此右子树中，其关键字值最小，但大于X的关键字；

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 20

6.4 二元查找树——在树上的查找

二、二元查找树的存储结构

```
typedef struct celltype {
    records data;
    struct celltype *lchild,*rchild;
} BSTNode;
typedef BSTNode * BST;
```

三、SEARCH查找操作的实现

算法要点：设F为二元查找树的根结点的指针，k为已知的关键字。在F中查找关键字为k的记录如下：

1. 若F = NULL，则查找失败；否则，
2. k == F->data.key，则查找成功；否则，
3. k < F->data.key，则查找F所指结点的左子树；否则，
4. k > F->data.key，则查找F所指结点的右子树。

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第6章 查找 Slide. 6 - 21

6.4 二元查找树——在树上的查找

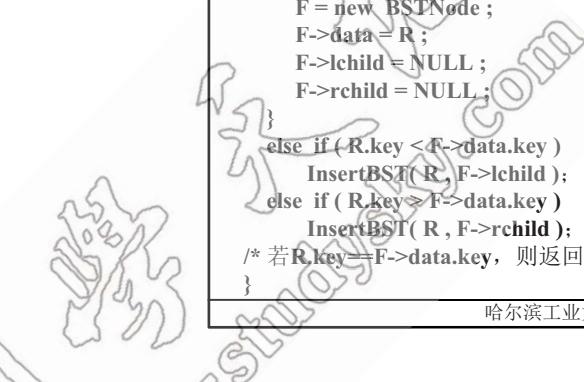
```

typedef struct celldata {
    records data;
    struct celldata *lchild,*rchild ;
} BSTNode;
typedef BSTNode * BST ;

BSTNode * SearchBST( keytype k, BST F )
{
    BSTNode * p = F;
    if ( p == NULL || k == p->data.key ) /* 递归终止条件 */
        return p;
    if ( k < p->data.key )
        return ( SearchBST ( k, p->lchild ) ); /* 查找左子树 */
    else
        return ( SearchBST ( k, p->rchild ) ); /* 查找右子树 */
}

```

哈尔滨工业大学 计算机科学与技术学院 张岩



数据结构与算法 第6章 查找 Slide. 6 - 22

三、INSERT操作的实现

```

/*二元查找树F 的类型定义 */
void InsertBST( records R, BST F )
/* 在二元查找树F 中插入新记录R */
{
    if ( F == NULL )
    {
        F = new BSTNode ;
        F->data = R ;
        F->lchild = NULL ;
        F->rchild = NULL ;
    }
    else if ( R.key < F->data.key )
        InsertBST( R, F->lchild );
    else if ( R.key > F->data.key )
        InsertBST( R, F->rchild );
    /* 若R.key==F->data.key, 则返回 */
}

```

- 若二叉排序树为空树，则新插入的结点为根结点；
- 否则，新插入的结点必为一个新的叶子结点。

新插入的结点一定是查找不成功时，查找路径上最后一个结点的左儿子或右儿子。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 23

四、二元查找树的建立

```

typedef int keytype;
typedef struct celldata {
    records data;
    struct celldata *lchild,*rchild ;
} BSTNode;
typedef BSTNode * BST ;

BST CreateBST ( void )
{
    BST F = NULL; /* 初始化时F为空*/
    keytype key;
    cin>>key>>其他字段; /*读入一个记录*/
    while( key ){/*假设key=0是输入结束标志*/
        InsertBST( R, F ); /* 插入记录R */
        cin>>key>>其他字段 /*读入下一个记录*/
    }
    return F; /*返回建立的二元查找树的根*/
}

```

注意：在建立二元查找树时，若按关键字有序顺序输入各记录，则产生退化的二元查找树—单链表。
如何防止？

- 随机输入各结点
- 在建立、插入和删除各结点过程中平衡相关结点的左、右子树。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 24

五、DELETE操作的实现

在二元查找树中删除结点，不能破坏二元查找树的结构性质，可按如下三种情况分别处理：

- 如果删除的是叶结点，则直接删除，不会破坏二元查找树的结构性质；
- 如果删除的结点只有一株左子树或右子树，则直接继承：将该子树移到被删结点位置；
- 如果删除的结点有两株子树，则用继承结点代替被删结点，这相当于删除继承结点——按 1) 或 2) 处理继承结点。

删除之前

删除14之后

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法		第 6 章 查找	Slide. 6 - 25
在二元查找树中删除结点		<pre>records deletemin(BST &F) { records tmp ; BST p ; if (F->lchild == NULL) { p = F ; tmp = F->data ; F = F->rchild ; delete p ; return tmp ; } else return(deletemin(F->lchild) ; }</pre>	<pre>void DeleteB(keytype k, BST &F) { if (F != NULL) if (k < F->data.key) DeleteB(k, f->lchild) ; else if (k > F->data.key) DeleteB(k, f->rchild) ; else if (F->rchild == NULL) F = F->lchild ; else if (F->lchild == NULL) F = F->rchild ; else F->data =deletemin(F->rchild); }</pre>

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法		第 6 章 查找	Slide. 6 - 26
在二元查找树的性能评价			<ul style="list-style-type: none"> ■二元查找树上进行查找的平均长度和二元树的形态有关。 ■在最坏情况下，二元查找树是通过把有序表的n个结点依次插入而生成的，此时所得到的二元查找树退化为一株高度为n的单支树，它的平均查找长度和单链表上的顺序查找相同，$(n+1)/2$。 ■在最好情况下，二元查找树的形态比较均匀，最终得到一株形态与折半查找的判定树相似，此时的平均查找长度约为$\log_2 n$。 ■二元查找树的平均高度为$O(\log_2 n)$。因此平均情况下，三种操作的平均时间复杂性为$O(\log_2 n)$。 ■就平均性能而言，二元查找树上的查找和二分查找差不多。 ■就维护表的有序性而言，二元查找树更有效。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法		第 6 章 查找	Slide. 6 - 27		
6.5 散列法——哈希 (Hash) 查找			<h3>6.5 散列法——哈希 (Hash) 查找</h3> <ul style="list-style-type: none"> 6.5.1 散列技术的基本思想和相关概念 6.5.2 散列函数的构造方法 6.5.3 散列表上处理冲突的方法 6.5.4 散列表上的查找、插入和删除 6.5.5 散列表的性能分析 <p>Hash查找的关键问题</p> <table border="0"> <tr> <td style="vertical-align: top;"> ①构造Hash函数 ②解决冲突的方法 </td> <td style="vertical-align: top;"> 研究和学习内容 散列表上的查找、插入和删除 </td> </tr> </table>	①构造Hash函数 ②解决冲突的方法	研究和学习内容 散列表上的查找、插入和删除
①构造Hash函数 ②解决冲突的方法	研究和学习内容 散列表上的查找、插入和删除				

哈尔滨工业大学 计算机科学与技术学院 张岩

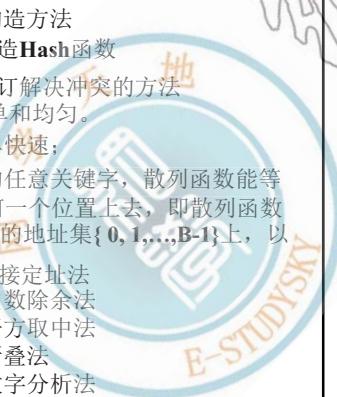
数据结构与算法		第 6 章 查找	Slide. 6 - 28
6.5.1 散列技术的基本思想和相关概念		<h4>6.5.1 散列技术的基本思想和相关概念</h4> <h5>一、散列技术的基本思想</h5> <ul style="list-style-type: none"> ■基于比较关键字的查找，其平均时间为$O(\log n) \sim O(n)$。 ■实际上用判定树可以证明，基于比较关键字的查找的平均和最坏情况下的比较次数的下界是$\log n + O(1)$，即$\Omega(\log n)$。 ■要向突破此下界，就不能仅依赖于比较来进行查找。 ■散列技术的基本思想是：把记录（元素）在表中的存储位置和该记录的关键字之间建立一种映射关系，关键字的值在这种映射关系下的像，就是相应记录在表中的存储位置。 ■散列技术在理想情况下，无需任何比较就可以找到待查的关键字，其查找的期望时间为$O(1)$。 	<h4>6.5.1 散列技术的基本思想和相关概念</h4> <h5>一、散列技术的基本思想</h5> <ul style="list-style-type: none"> ■基于比较关键字的查找，其平均时间为$O(\log n) \sim O(n)$。 ■实际上用判定树可以证明，基于比较关键字的查找的平均和最坏情况下的比较次数的下界是$\log n + O(1)$，即$\Omega(\log n)$。 ■要向突破此下界，就不能仅依赖于比较来进行查找。 ■散列技术的基本思想是：把记录（元素）在表中的存储位置和该记录的关键字之间建立一种映射关系，关键字的值在这种映射关系下的像，就是相应记录在表中的存储位置。 ■散列技术在理想情况下，无需任何比较就可以找到待查的关键字，其查找的期望时间为$O(1)$。

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法		第6章 查找	Slide. 6 - 29
6.5.1 散列技术的基本思想和相关概念			
二、相关概念			
设 U 表示所有可能出现的关键字集合， K 表示实际出现（实际存储）的关键字集合，即 $K \subseteq U$ ， $F[B-1]$ 是一个数组，其中 $B = O(K)$ 。则，			
<ul style="list-style-type: none"> ▪ 从 U 到表 $F[B-1]$ 下标集合上的一个映射 $h: U \rightarrow \{0, 1, 2, \dots, B-1\}$ 称为散列函数（哈希函数，杂凑函数） ▪ 数组 F 称为散列表（Hash表，杂凑表）。数组 F 中的每个单元称为桶(bucket)。 ▪ 对于任意关键字 $K_i \in U$，函数值 $h(K_i)$ 称为 K_i 的散列地址（Hash地址，散列值，存储地址，桶号） ▪ 将结点（记录）按其关键字的散列地址存储到散列表中的过程称为散列。 (collision) ▪ 不同的关键字具有相同散列地址的现象称为散列冲突（碰撞）。而发生冲突的两个关键字称为同义词(synonym)。 			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第6章 查找	Slide. 6 - 30																																				
6.5.1 散列技术的基本思想和相关概念																																							
例：全国30个地区的人口统计，每个地区为一个记录，内容如下：																																							
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>编号</td> <td>地区名</td> <td>总人口</td> <td>汉族</td> <td>回族</td> <td>...</td> </tr> </table>				编号	地区名	总人口	汉族	回族	...																														
编号	地区名	总人口	汉族	回族	...																																		
$h_1(key)$: 取关键字第一个字母在字母表中的序号 $h_2(key)$: 求第一和最后字母在字母表中序号之和，然后取30的余数 $h_3(key)$: 将第一个字母的八进制看成十进制，再取30的余数																																							
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Key</th> <th>BEIJING (北京)</th> <th>TIANJIN (天津)</th> <th>HEBEI (河北)</th> <th>SHANXI (山西)</th> <th>SHANGHAI (上海)</th> <th>SHANDONG (山东)</th> <th>HENAN (河南)</th> <th>SICHUAN (四川)</th> </tr> </thead> <tbody> <tr> <td>$h_1(key)$</td> <td>02</td> <td>20</td> <td>08</td> <td>19</td> <td>19</td> <td>19</td> <td>08</td> <td>19</td> </tr> <tr> <td>$h_2(key)$</td> <td>09</td> <td>04</td> <td>17</td> <td>28</td> <td>28</td> <td>26</td> <td>22</td> <td>03</td> </tr> <tr> <td>$h_3(key)$</td> <td>04</td> <td>26</td> <td>02</td> <td>13</td> <td>23</td> <td>17</td> <td>16</td> <td>16</td> </tr> </tbody> </table>				Key	BEIJING (北京)	TIANJIN (天津)	HEBEI (河北)	SHANXI (山西)	SHANGHAI (上海)	SHANDONG (山东)	HENAN (河南)	SICHUAN (四川)	$h_1(key)$	02	20	08	19	19	19	08	19	$h_2(key)$	09	04	17	28	28	26	22	03	$h_3(key)$	04	26	02	13	23	17	16	16
Key	BEIJING (北京)	TIANJIN (天津)	HEBEI (河北)	SHANXI (山西)	SHANGHAI (上海)	SHANDONG (山东)	HENAN (河南)	SICHUAN (四川)																															
$h_1(key)$	02	20	08	19	19	19	08	19																															
$h_2(key)$	09	04	17	28	28	26	22	03																															
$h_3(key)$	04	26	02	13	23	17	16	16																															
被查找元素的存储地址 = Hash (Key)																																							
散列法通常用于实际出现的关键字的数目远小于关键字所有可能取值的数量。																																							
哈尔滨工业大学 计算机科学与技术学院 张岩																																							

数据结构与算法		第6章 查找	Slide. 6 - 31
6.5.2 散列函数的构造方法			
Hash查找的关键问题 <ul style="list-style-type: none"> ① 构造Hash函数 ② 制订解决冲突的方法 			
■ 散列函数的选择两条标准：简单和均匀。			
简单指散列函数的计算简单快速； 均匀指对于关键字集合中的任意关键字，散列函数能等概率地将其映射到表空间的任何一个位置上去，即散列函数能将子集 K 随机均匀地分布在表的地址集 $\{0, 1, \dots, B-1\}$ 上，以使冲突最小化。			
构造散列函数的几种方法：  <ul style="list-style-type: none"> ➢ 直接定址法 ➢ 质数除余法 ➢ 平方取中法 ➢ 折叠法 ➢ 数字分析法 ➢ 随机数法 			
为简单起见，以下假定关键字是定义在自然数集合上，...。			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法		第6章 查找	Slide. 6 - 32																																																
构造散列函数的几种方法																																																			
1. 直接定址法： $Hash(key) = key$ ；或 $Hash(key) = a \cdot key + b$ ；																																																			
例一：1岁到100岁的人口统计表，年龄是关键字 $Hash(key) = key$																																																			
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>地址</td> <td>01</td> <td>02</td> <td>03</td> <td>04</td> <td>05</td> <td>...</td> <td>25</td> <td>26</td> <td>27</td> <td>...</td> <td>100</td> </tr> <tr> <td>年龄</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>...</td> <td>25</td> <td>26</td> <td>27</td> <td>...</td> <td>100</td> </tr> <tr> <td>人数</td> <td>3000</td> <td>2000</td> <td>5000</td> <td>...</td> <td>6000</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> </tr> <tr> <td>:</td> <td></td> </tr> </table>				地址	01	02	03	04	05	...	25	26	27	...	100	年龄	1	2	3	4	5	...	25	26	27	...	100	人数	3000	2000	5000	...	6000	:											
地址	01	02	03	04	05	...	25	26	27	...	100																																								
年龄	1	2	3	4	5	...	25	26	27	...	100																																								
人数	3000	2000	5000	...	6000																																								
:																																																			
例二：解放后出生的人口统计表，关键字是年份 $Hash(key) = key + (-1949) + 1$																																																			
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>地址</td> <td>01</td> <td>02</td> <td>03</td> <td>04</td> <td>05</td> <td>...</td> <td>25</td> <td>26</td> <td>27</td> <td>...</td> <td>100</td> </tr> <tr> <td>年份</td> <td>1949</td> <td>1950</td> <td>1951</td> <td>...</td> <td>1973</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> </tr> <tr> <td>人数</td> <td>3000</td> <td>2000</td> <td>5000</td> <td>...</td> <td>6000</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> <td>...</td> </tr> <tr> <td>:</td> <td></td> </tr> </table>				地址	01	02	03	04	05	...	25	26	27	...	100	年份	1949	1950	1951	...	1973	人数	3000	2000	5000	...	6000	:											
地址	01	02	03	04	05	...	25	26	27	...	100																																								
年份	1949	1950	1951	...	1973																																								
人数	3000	2000	5000	...	6000																																								
:																																																			
哈尔滨工业大学 计算机科学与技术学院 张岩																																																			

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法			
第6章 查找			
Slide. 6 - 33			
2. 质数除余法			
$\text{Hash}(\text{key}) = \text{key \% m}$			
设桶数B，取质数 $m \leq B$ (m 为小于等于B的最大质数)			
注：若m是关键字的基数的幂次，则就是选择关键字最后若干位为地址，而与高位无关。高位不同低位相同的均为同义词。特别地， m 为偶数，....			
记录 key key^2 Hash			
A	0100	0 010 000	010
I	1100	1 210 000	210
J	1200	1 440 000	440
3. 平方取中法			
I0	1160	1 370 400	370
P1	2061	4 310 541	310
P2	2062	4 314 704	314
Q1	2161	4 734 741	734
Q2	2162	4 741 304	741
Q3	2163	4 745 651	745
散列地址的位数要根据B来确定，有时要设一个比例因子，限制地址越界。			
平方取中法			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法			
第6章 查找			
Slide. 6 - 34			
4. 折叠法			
若关键字段数较多，可根据B的位数将关键字分割成位数相同的若干段（最后一段位数可以不同），然后将各段叠加和（设去进位）作为散列地址。分两种：			
例如，图书编号：0-442-20586-4 B<1000			
5864	5864	左：移位叠加	
4220	0224	Hash(key) = 0088	
+ 04	+ 04	右：间界叠加	
10088	6092	Hash(key) = 6092	
8 1 3 4 6 5 3 2			
8 1 3 7 2 2 4 2			
8 1 3 8 7 4 2 2			
若事先知道关键字集合，且关键字的位数比散列表的地址位数多，则可选分布			
8 1 3 0 1 3 6 7			
8 1 3 2 2 8 1 7			
8 1 3 3 8 9 6 7			
8 1 3 5 4 1 5 7			
如右图所示，假设散列表长为 100_{10} ，可取中间4位中的两位为散列地址。			
(1) (2) (3) (4) (5) (6) (7) (8)			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法			
第6章 查找			
Slide. 6 - 35			
6. 随机数法			
选择一个随机函数，取关键字的随机函数值作为散列地址，即			
$\text{Hash}(\text{key}) = \text{random}(\text{key})$			
其中 random 是某个伪随机函数，且函数值在 $0, \dots, B-1$ 之间。			
通常，当关键字长度不等时采用此法较恰当。			
小结：构造 Hash 函数应注意以下几个问题：			
o 计算 Hash 函数所需时间 计算简单			
o 关键字的长度			
o 散列表的大小			
o 关键字的分布情况			
o 记录的查找频率			
分布均匀			
哈尔滨工业大学 计算机科学与技术学院 张岩			

数据结构与算法			
第6章 查找			
Slide. 6 - 36			
6.5.3 处理冲突的方法及查找操作			
通常情况下， h 是一个压缩映像， $ K < U $ ，因此无论如何设计 h ，也不可能完全避免冲突。因此选择好的解决冲突的方法十分重要。			
解决冲突的几种方法：			
闭散列方法—开放定址法			
线性探测法			
随机探测法			
二次探测法			
双散列函数法			
桶扩充法—带溢出表的内散列表			
开散列方法—链地址法			
■ 闭散列法也称开放定址法，是将所有的结点都存放在散列表 F 中；			
■ 开(外)散列法也称链地址法，是将互为同义词的结点链接成一个单链表，而将此单链表的头指针存放在散列表 F 中。			
哈尔滨工业大学 计算机科学与技术学院 张岩			

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法 第6章 查找 Slide. 6 - 37

6.5.3 处理冲突的方法及查找操作

一、开放定址法

1. 基本思想：当冲突发生时，使用某些探测（查）技术在散列表中形成一个探测序列，沿此序列逐个单元查找，直到找到给定的关键字、或者碰到一个开放地址（即该地址单元为空、空桶）为止。插入时探测到开放地址，则将待插入的新结点插入该地址单元。查找时探测到开放地址则表明无待查的关键字，即查找失败。

注意：以发现空桶作为查找成功与否的条件，不允许删除！为了允许删除操作，要区分不曾插入过的空桶（empty）和已经把其中的元素删除而腾出的空桶（deleted）。

初始建表时，须将每个单元（关键字）标记为（empty），删除一个结点之后，要把腾出的空桶置为（deleted）

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 38

6.5.3 处理冲突的方法及查找操作

例：设 $B=8$ ，现有 7 个记录，其关键字及散列地址分别为：
 $h(a) = 3, h(b) = 0, h(c) = 4, h(d) = 3, h(e) = 4, h(f) = 5, h(x) = 3$

建表和插入操作
 ■ 起初，散列表为空；
 ■ 计算散列地址： $h = h(key)$ ；
 ■ 从 h 开始，依次查看下一个桶是否为空，第 1 次再散列的地址为 $h_1(key) = (h(key) + i) \% B$ 。若找到空桶，则插入；若查遍整个表，所有桶都不空，则表满，不能插
 痞找操作
 ■ 计算散列地址： $h = h(key)$ ；
 ■ 从 h 开始，依次查看下一个桶的关键字是否为 key ，直到找到关键字为 key 的记录或发现空桶时为止。若找到空桶，则可以确定不存在关键字为 key 的记录。
 删除操作：1. 存在的问题；2. 解决办法；3. 改进的算法

0	b
1	x
2	
3	a
4	c
5	d
6	e
7	f

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 39

6.5.3 处理冲突的方法及查找操作

一、开放定址法

2. 开放定址法一般形式： $h_i(key) = (h(key) + d_i) \% B \quad 1 \leq i < B-1 \quad (6.1)$

其中 $h(key)$ 为散列函数， d_i 为探测增量序列， B 为表长。
 $h(key)$ 为初始的探测位置，若令 (6.1) 中的 i 从 0 开始，且 $d_0 = 0$ ，则 $h_0(key) = h(key)$ 。则 (6.1) 可改写为：

$h_i(key) = (h(key) + d_i) \% B \quad 0 \leq i < B-1 \quad (6.2)$

(1) $d_i = 1, 2, 3, \dots, B-1$ ，称为线性探测再散列
 (2) $d_i = 1c, 2c, 3c, \dots, c > 1$ (c 与 B 互质)，称为线性补偿探测再散列
 (3) d_i 为伪随机序列，称为伪随机探测再散列
 (4) $d_i = 1^2, -1^2, 2^2, -2^2, \dots, k^2 (k \leq B/2)$ 称为二次探测再散列

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 40

6.5.3 处理冲突的方法及查找操作

1、线性探测法 ($c = 1$) $h_i(key) = (h(key) + i) \% B, 1 \leq i < B-1$

其基本思想是将散列表 $F[0, \dots, B-1]$ 看成一个循环数组，若初始探测的地址为 $d = h(key)$ ，则最长的探测序列为：

$d, d+1, d+2, \dots, B-1, 0, 1, \dots, d-1$

探测过程终止于三种情况：

- 若当前探测的单元为空，则表示查找失败，若是插入则插入其中；
- 若当前探测的单元含有 key ，则查找成功（对插入意味着失败）；
- 若探测到 $F[d-1]$ 时仍未发现空单元也未找到 key ，则无论查找还是插入均意味着失败。

缺点：当填满几个相继的桶后，如果在这些桶上再发生冲突，则后插入的元素将使填满的桶排列的越来越长，发生冲突的概率越来越大。因此，令 $c > 1$ 。题 2.2 线性补偿探测法

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法		第6章 查找	Slide. 6 - 41
6.5.3 处理冲突的方法及查找操作			
线性散列法($c=1$)的闭散列表的实现举例			
<pre>/* 数据结构定义 */ /* 关键字类型定义 */ typedef char keytype[10]; /* 闭散列表类型 B 为桶数 */ typedef record HASH[B]; /* 散列函数的定义 */ int h(keytype k) { int i,sum; sum=0; for(i=0;i<10;i++) sum=sum+ord(k[i]); return (sum%B); }/*仅是一个例子*/ </pre>	<pre>int Search(keytype k, HASH F) { int first,rehash,locate; first=h(k);rehash=0; locate=first; while((rehash<B)&& (F[locate].key!=empty)){ if(F[locate].key==k) return locate; else rehash=rehash+1; locate=(first+rehash)%B; } return -1; }/*Search*/ </pre>		

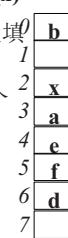
哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法		第6章 查找	Slide. 6 - 42
6.5.3 处理冲突的方法及查找操作			
线性散列法($c=1$)的闭散列表的实现			
<pre>/* 数据结构定义 */ /* 关键字类型定义 */ typedef char keytype[10]; /* 闭散列表类型 B 为桶数 */ typedef record HASH[B]; /* 散列函数的定义 */ int h(keytype k) { int i,sum; sum=0; for(i=0;i<10;i++) sum=sum+ord(k[i]); return (sum%B); }/*仅是一个例子*/ </pre>	<pre>void Insert(records R, HASH F) { int first=h(k), rehash=0; int locate=first; while((rehash<B)&& (F[locate].key!=R.key)){ locate=(first+rehash)%B; if((F[locate].key==empty) (F[locate].key==deleted)){ F[locate]=R; } else rehash+=1; } if(rehash>=B) cout<<"data base is full!"; }/*Insert*/ </pre>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法		第6章 查找	Slide. 6 - 43
6.5.3 处理冲突的方法及查找操作			
线性散列法($c=1$)的闭散列表的实现			
<pre>/* 数据结构定义 */ /* 关键字类型定义 */ typedef char keytype[10]; /* 闭散列表类型 B 为桶数 */ typedef record HASH[B]; /* 散列函数的定义 */ int h(keytype k) { int i,sum; sum=0; for(i=0;i<10;i++) sum=sum+ord(k[i]); return (sum%B); }/*仅是一个例子*/ </pre>	<pre>void Delete(keytype k,HASH F) { int locate; locate=Search(k,F); if(locate!=-1) F[locate].key=deleted; }/*Delete*/ </pre>		

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法		第6章 查找	Slide. 6 - 44
6.5.3 处理冲突的方法及查找操作			
2、线性补偿再散列 ($c > 1$)			
$h_i(\text{key}) = (h(\text{key}) + c \cdot i) \% B, 1 \leq i < B - 1$ 其中 c 和 B 互质 例：设 $B=8$ ，选择 $c=3$ ，现有 7 个记录，其关键字及散列地址分别为： $h(a)=3, h(b)=0, h(d)=3, h(e)=4, h(f)=5, h(x)$			

缺点：当发生若干次冲突之后，将出现按 c 值的间隔分段填满相邻的桶，结果发生冲突的概率越来越大。

事实上，线性法处理冲突，如果新的散列值只依赖前一个散列值，总会使填满的桶集中。

3、线性随机探测再散列

$$h_i(\text{key}) = (h(\text{key}) + d_i) \% B, 1 \leq i < B - 1$$

其中， d_1, d_2, \dots, d_{B-1} 是 $1, 2, \dots, B-1$ 的随机序列。

注意：插入、删除和查找时，要使用同一个随机序列。

此方法的关键是随机序列的产生。例 6.6(p205) 移位法

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

6.5.3 处理冲突的方法及查找操作

二、桶扩充法——带溢出表的内散列法

基本思想：扩充散列表中的每个桶，形成带溢出表的散列表。每个桶包括两部分：一部分是主表元素；另一部分或者为空或者由一个链表组成溢出表，其首结点的指针存入主表的链域。主表元素的类型与溢出表的类型相同。

特点：
 (1) 主表元素和它的溢出表元素具有相同的散列地址。
 (2) 空间利用率不高；

```

typedef struct celltype{
    records data;
    celltype next;
} HASH[B];
    
```

实现：存储结构；
 操作实现：插入：当发生冲突时，把新元素插入溢出表。
 查找：要查看同一散列地址的主表和溢出表。
 删除：若被删除的是主表元素，则要遍历科学与技术学院 张岩

6.5.3 处理冲突的方法及查找操作

三、开（外）散列法——拉链法（链地址法）

解决冲突的做法是：将所有关键字为同义词的结点链接在同一个单链表中。将散列表定义为由B（表长）个单链表头指针组成的指针数组F[0, 1, ..., B-1]。

凡是散列地址为1的结点，均插入到以F[i]为头指针的单链表中。F中每个单元的初值均应为空。

例：
 $(19, 14, 23, 01, 68, 20, 84, 27, 55, 11, 10, 79)$
 $H(key) = key \bmod 13$

若选择的散列函数能使同义词的个数等于桶的平均长度：n/B(n为元素个数)，则查找的时间将是一个小常数。

6.5.3 处理冲突的方法及查找操作

三、开（外）散列法——拉链法（链地址法）

开散列表的实现

```

cellptr SEARCH(keytype k, HASH F)
{
    cellptr bptr;
    bptr=F[h(k)];
    while(bptr!=NULL)
        if(bptr->data.key==k)
            return bptr;
        else
            bptr=bptr->next;
}
/*SEARCH*/
    
```

哈尔滨工业大学 计算机科学与技术学院 张岩

6.5.3 处理冲突的方法及查找操作

三、开（外）散列法——拉链法（链地址法）

开散列表的实现

```

void INSERT(records R, HASH F)
{
    int bucket;
    struct celltype{
        records data;
        celltype *next;
    };
    cellptr oldheader;
    if(SEARCH(R.key,F)==NULL){
        bucket=h(R.key);
        oldheader=F[bucket];
        F[bucket]=new celltype;
        F[bucket]->data=R;
        F[bucket]->next=oldheader;
    }
}
/*INSERT 没找到，则插入；否则，就什么也不做*/
    
```

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法		第6章 查找	Slide. 6 - 49
开散列表的实现		<pre>void DELETE(keytype k, HASH F) { int bucket=h(k); boolean b=FALSE; celltype bptr, p; if(F[bucket] != NULL)//可能在表中 if(F[bucket]->data.key==k){//首元素就是 bptr= F[bucket]; F[bucket]=F[bucket]->next; free(bptr); }else{//可能在中间或不存在 bptr=F[bucket]; while((bptr->next!=NULL)&&(b==FALSE)) if(bptr->next->data.key==k){ p=bptr->next; bptr->next=p->next; free(p); b=TRUE; }else bptr=bptr->next; } /* DELETE */ } 哈尔滨工业大学 计算机科学与技术学院 张岩</pre>	

数据结构与算法		第6章 查找	Slide. 6 - 50
		<p>内散列和外散列的比较</p> <ul style="list-style-type: none"> 通常，每个桶中的同义词子表都很短，设有 n 个关键码通过某一个散列函数，存放到散列表中的 B 个桶中。那么每一个桶中的同义词子表的平均长度为 n / B。以搜索平均长度为 n / B 的同义词子表代替了搜索长度为 n 的顺序表，搜索速度快得多。 应用开散列法处理冲突，需要增设链接指针，似乎增加了存储开销。事实上，由于闭散列法必须保持大量的空闲空间以确保搜索效率，如二次探查法要求装填因子 $\alpha = n / B \leq 0.5$，而表项所占空间又比指针大得多，所以使用开散列法反而比闭散列法节省存储空间。 	

数据结构与算法		第6章 查找	Slide. 6 - 51
		<h3>6.5.5 散列表的性能分析</h3> <ul style="list-style-type: none"> 散列表是一种直接计算记录存放地址的方法，它在关键码与存储位置之间直接建立了映象。 当选择的散列函数能够得到均匀的地址分布时，在搜索过程中可以不做多次探测。 由于很难避免冲突，增加了搜索时间。冲突的出现，与散列函数的选取（地址分布是否均匀），处理冲突的方法（是否产生堆积—哈希地址不相同的关键字有产生冲突）有关。 	

数据结构与算法		第6章 查找	Slide. 6 - 52
		<h3>6.5.5 散列表的性能分析</h3> <ul style="list-style-type: none"> 若设 α 是散列表的装填因子： $\alpha = \frac{\text{表中装载的记录数 } n}{\text{表的最大记录容量数 } B}$ <ul style="list-style-type: none"> 当装填因子 α 较高时，选择的散列函数不同，散列表的搜索性能差别很大。在一般情况下多选用除留余数法，其中的除数在实用上应选择不含有20以下的质因数的质数。 对散列表技术进行的实验评估表明，它具有很好的平均性能，优于一些传统的技术，如平衡树(AVL)。但散列表在最坏情况下性能很不好。如果对一个有 n 个关键码的散列表执行一次搜索或插入操作，最坏情况下需要 $O(n)$ 的时间。 Knuth对不同的冲突处理方法进行了分析。 	

数据结构与算法 第6章 查找 Slide. 6 - 53

6.5.5 散列表的性能分析

- 用地址分布均匀的散列函数Hash()计算桶号。
- S_n 是搜索一个随机选择的关键码 $x_i (1 \leq i \leq n)$ 所需的关键码比较次数的期望值，称为在 $\alpha = n / B$ 时的搜索成功的平均查找长度。
- U_n 是在长度为 B 的散列表中 n 个桶已装入表项的情况下，装入第 $n+1$ 项所需执行的关键码比较次数期望值，称为在 $\alpha = n / B$ 时的搜索不成功的平均查找长度。
- 用不同的方法溢出处理冲突时散列表的平均搜索长度如图所示。

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 54

6.5.5 散列表的性能分析

- 用不同的方法溢出处理冲突时散列表的平均搜索长度如图所示。

处理冲突的方法	平均搜索长度 ASL	
	查找成功 S_n	查找不到 U_n (插入新记录)
闭散列法	$(I+I/(1-\alpha))/2$	$(I+I/(1-\alpha)^2)/2$
线性探查法	$-\ln(1-\alpha)/\alpha$	$I/(1-\alpha)$
伪随机探查法		
三次探查法		
双散列法		
开散列法 (同义词字表法)	$I+\alpha/2$	$\alpha+e^{-\alpha} \approx \alpha$

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 55

例 求散列表大小并设计散列函数

- 设有一个含 200 个表项的散列表，用二次探查法解决冲突，按关键码查询时找到一个新表项插入位置的平均探查次数不超过 1.5 ，则散列表项应能够至少容纳多少个表项。（设查找不到的平均查找长度为 $U_n=1/(1-\alpha)$ ，其中 α 为装填因子）
- 解：设表中表项个数 $n = 200$ ，搜索不成功的平均搜索长度 $U_n=1/(1-\alpha) \leq 1.5 \Rightarrow \alpha \leq 1/3$
- $\therefore n / B = 200 / B = \alpha \leq 1/3 \quad B \geq 600$

下图给出一些实验结果

哈尔滨工业大学 计算机科学与技术学院 张岩

数据结构与算法 第6章 查找 Slide. 6 - 55

6.5.5 散列表的性能分析

$\alpha = n / B$	0.50	0.75	0.90	0.95
散列函数种类	开散列法	闭散列法	开散列法	闭散列法
平方取中	1.26	1.73	1.40	9.75
除留余数	1.19	4.52	1.31	10.20
移位折叠	1.33	21.75	1.48	65.10
分界折叠	1.39	22.97	1.57	48.70
数字分析	1.35	4.55	1.49	30.62
理论值	1.25	1.50	1.37	2.50

搜索关键码时所需对桶的平均访问次数

从图中可以看出，开散列法优于闭散列法；在散列函数中，用除留余数法作散列函数优于其它类型的散列函数，最差的是折叠法。

哈尔滨工业大学 计算机科学与技术学院 张岩

哈工大计算机考研全套视频和资料，真题、考点、典型题、命题规律独家视频讲解！
详见：网学天地（www.e-studysky.com）；咨询QQ：2696670126

数据结构与算法

第 6 章 查找

Slide. 6 - 57

本章小结

- 熟练掌握顺序表和有序表的查找方法。
- 熟练掌握二叉排序树的构造和查找方法。
- 熟练掌握哈希表的构造方法，深刻理解哈希表与其它结构的表的实质性的差别。
- 掌握描述查找过程的判定树的构造方法，以及按定义计算各种查找方法在等概率情况下查找成功时的平均查找长度。

哈尔滨工业大学 计算机科学与技术学院 张岩 [K]



网学天地
www.e-studysky.com