数据结构与算法

第八章　查找

顺序查找

折半查找

分块查找

哈希查找

第八章 查找

- 查找——也叫检索，是根据给定的某个值，在表中确定一个关键字等于给定值的记录或数据元素
- 关键字——是数据元素中某个数据项的值，它可以标识一个数据元素
- 查找方法评价
  - 查找速度
  - 占用存储空间多少
  - 算法本身复杂程度
  - 平均查找长度ASL(Average Search Length)：为确定记录在表中的位置，需和给定值进行比较的关键字的个数的期望值叫查找算法的~

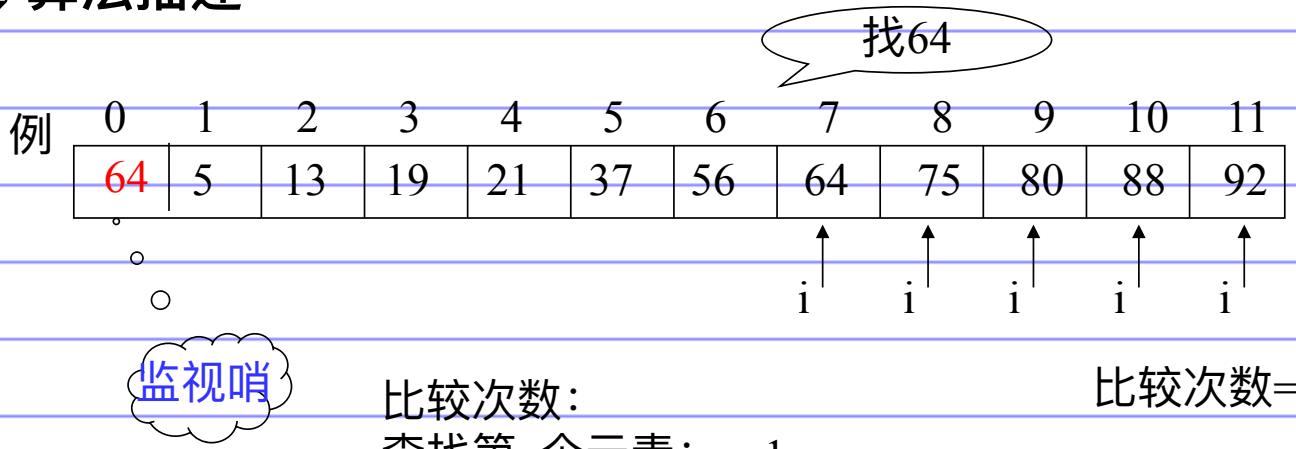
对含有n个记录的表，$ASL=\sum_{i=1}^{n} p_i c_i$

其中：p_i为查找表中第i个元素的概率，$\sum_{i=1}^{n} p_i = 1$

c_i为找到表中第i个元素所需比较次数

7.1 顺序查找

- 查找过程：从表的一端开始逐个进行记录的关键字和给定值的比较
- 算法描述



比较次数：
查找第n个元素： 1
查找第n-1个元素： 2
……
查找第1个元素： n
查找第i个元素： n+1-i
查找失败： n+1

比较次数=5

■ 顺序查找方法的ASL

对含有n个记录的表，$ASL=\sum_{i=1}^{n} p_i c_i$

设表中每个元素的查找概率相等$p_i=\frac{1}{n}$（因为未找到的概率为0或不好决定，故不考虑）

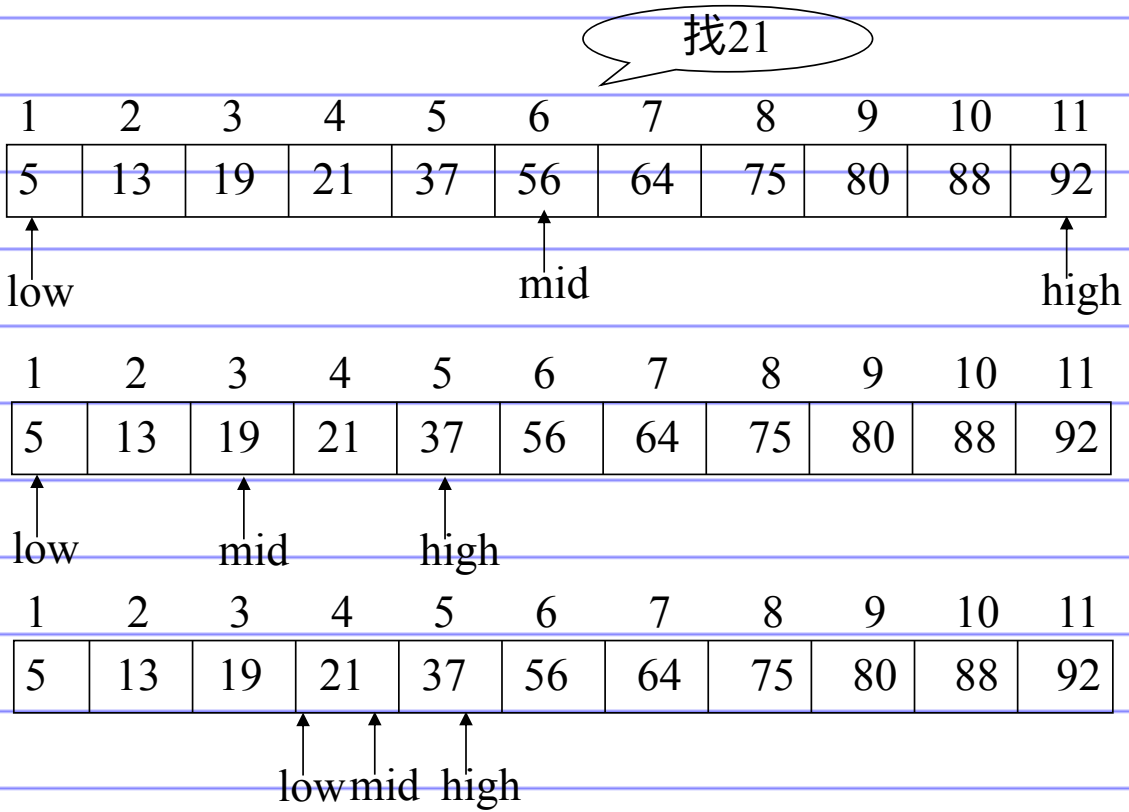
则$ASL=\sum_{i=1}^{n} p_i c_i=\frac{1}{n}\sum_{i=1}^{n} i=\frac{1}{n}\cdot\frac{n(n+1)}{2}=\frac{n+1}{2}$

7.2 折半查找

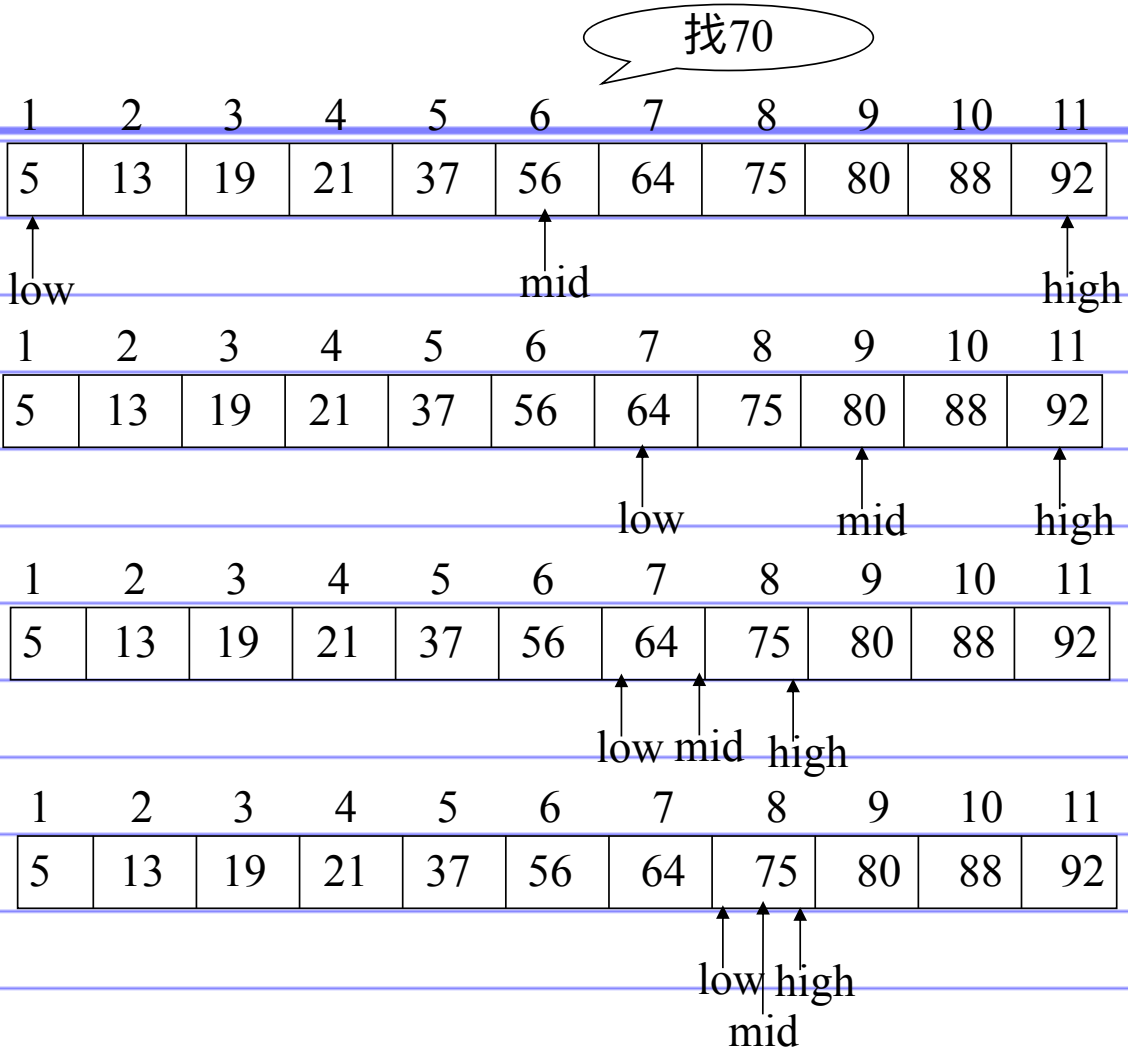
- 查找过程：每次将待查记录所在区间缩小一半
- 适用条件：采用顺序存储结构的有序表
- 算法实现
  - 设表长为n，low、high和mid分别指向待查元素所在区间的上界、下界和中点,k为给定值
  - 初始时，令low=1,high=n,mid=⌊(low+high)/2⌋
  - 让k与mid指向的记录比较
    - ✧若k==r[mid].key，查找成功
    - ✧若k<r[mid].key，则high=mid-1
    - ✧若k>r[mid].key，则low=mid+1
  - 重复上述操作，直至low>high时，查找失败

■ 算法描述

例


找21
1 2 3 4 5 6 7 8 9 10 11
5 13 19 21 37 56 64 75 80 88 92
low
mid
high
1 2 3 4 5 6 7 8 9 10 11
5 13 19 21 37 56 64 75 80 88 92
low
mid
high
1 2 3 4 5 6 7 8 9 10 11
5 13 19 21 37 56 64 75 80 88 92
low
mid
high

例

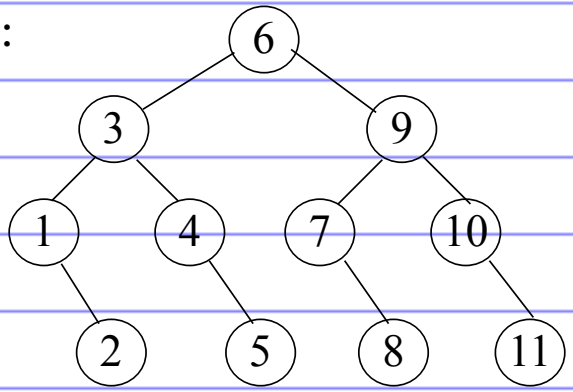

找70
1 2 3 4 5 6 7 8 9 10 11
5 13 19 21 37 56 64 75 80 88 92
low
mid
high
1 2 3 4 5 6 7 8 9 10 11
5 13 19 21 37 56 64 75 80 88 92
low
mid
high
1 2 3 4 5 6 7 8 9 10 11
5 13 19 21 37 56 64 75 80 88 92
low mid
high
1 2 3 4 5 6 7 8 9 10 11
5 13 19 21 37 56 64 75 80 88 92
low high
mid

1	2	3	4	5	6	7	8	9	10	11
5	13	19	21	37	56	64	75	80	88	92

high low

1	2	3	4	5	6	7	8	9	10	11
5	13	19	21	37	56	64	75	80	88	92

判定树:



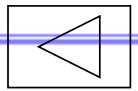
■ 算法评价

- ✧判定树：描述查找过程的二叉树叫~
- ✧有n个结点的判定树的深度为$\lfloor \log_2 n \rfloor + 1$
- ✧折半查找法在查找过程中进行的比较次数最多不超过其判定树的深度
- ✧折半查找的ASL

设表长$n = 2^h - 1, h = \log_2(n+1)$,即判定树是深度为$h$的满二叉树

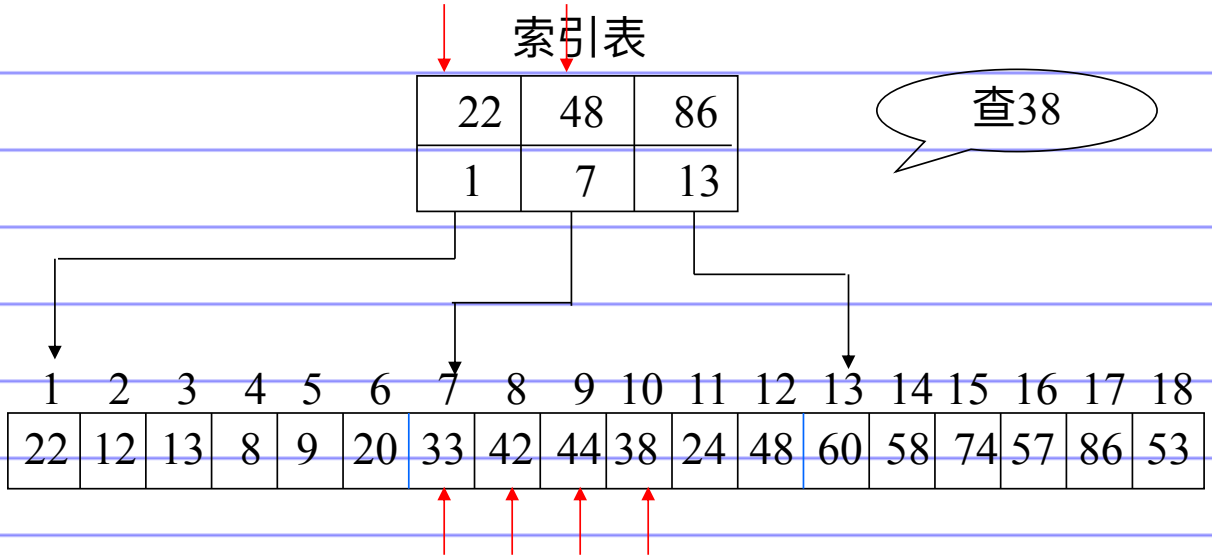
设表中每个记录的查找概率相等$p_i = \frac{1}{n}$

则：$ASL = \sum_{i=1}^{n} p_i c_i = \frac{1}{n}\sum_{i=1}^{n} c_i = \frac{1}{n}\sum_{j=1}^{h} j \cdot 2^{j-1} = \frac{n+1}{n}\log_2(n+1) - 1 \approx \log_2(n+1) - 1$



7.3 分块查找

- 查找过程：将表分成几块，块内无序，块间有序；先确定待查记录所在块，再在块内查找
- 适用条件：分块有序表
- 算法实现
  - 用数组存放待查记录,每个数据元素至少含有关键字域
  - 建立索引表，每个索引表结点含有最大关键字域和指向本块第一个结点的指针
- 算法描述


索引表
22
48
86
1
7
13
查38
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
22 12 13 8 9 20 33 42 44 38 24 48 60 58 74 57 86 53

❑ 分块查找方法评价

$ASL_{bs} = L_b + L_w$

其中：L_b ——查找索引表确定所在块的平均查找长度

L_w ——在块中查找元素的平均查找长度

若将表长为n的表平均分成b块，每块含s个记录，并设表中每个记录的查找概率相等，则：

(1)用顺序查找确定所在块：$ASL_{bs} = \frac{1}{b}\sum_{j=1}^{b} j + \frac{1}{s}\sum_{i=1}^{s} i = \frac{b+1}{2} + \frac{s+1}{2} = \frac{1}{2}(\frac{n}{s} + s) + 1$

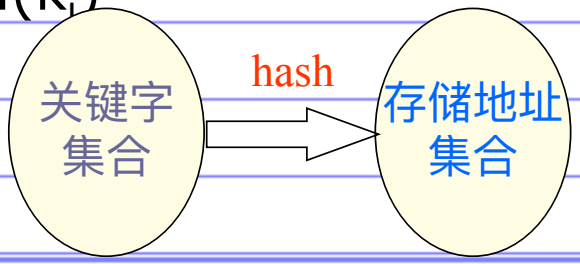
(2)用折半查找确定所在块：$ASL_{bs} \approx \log_2(\frac{n}{s} + 1) + \frac{s}{2}$

查找方法比较

	顺序查找	折半查找	分块查找
ASL	最大 $\frac{n+1}{2}$	最小 $\log_2(n+1)-1$	两者之间 $\log_2(\frac{n}{s}+1)+\frac{s}{2}$
表结构	有序表、无序表	有序表	分块有序表
存储结构	顺序存储结构 线性链表	顺序存储结构	顺序存储结构 线性链表

7.4 哈希查找

- 基本思想：在记录的存储地址和它的关键字之间建立一个确定的对应关系；这样，不经过比较，一次存取就能得到所查元素的查找方法
- 定义
  - 哈希函数——在记录的关键字与记录的存储地址之间建立的一种对应关系叫~
  - 哈希函数是一种映象，是从关键字空间到存储地址空间的一种映象
  - 哈希函数可写成：$addr(a_i)=H(k_i)$
    - a_i是表中的一个元素
    - $addr(a_i)$是a_i的存储地址
    - k_i是a_i的关键字



✧哈希表——应用哈希函数，由记录的关键字确定记录在表中的地址，并将记录放入此地址，这样构成的表叫~

✧哈希查找——又叫散列查找，利用哈希函数进行查找的过程叫~

例 30个地区的各民族人口统计表

编号	地区别	总人口	汉族	回族……
1	北京			
2	上海			
⋮	⋮			

以编号作关键字，
构造哈希函数：H(key)=key
H(1)=1
H(2)=2

以地区别作关键字，取地区名称第一个拼音字母的序号作哈希函数：H(Beijing)=2
H(Shanghai)=19
H(Shenyang)=19

从例子可见：

- 哈希函数只是一种映象，所以哈希函数的设定很灵活，只要使任何关键字的哈希函数值都落在表长允许的范围之内即可
- 冲突：key1≠key2，但H(key1)=H(key2)的现象
- 同义词：具有相同函数值的两个关键字，叫该哈希函数的
- 哈希函数通常是一种压缩映象，所以冲突不可避免，只能尽量减少；同时，冲突发生后，应该有处理冲突的方法

- 哈希函数的构造方法
- 直接定址法
  - 构造：取关键字或关键字的某个线性函数作哈希地址，即H(key)=key　或　H(key)=a·key+b
  - 特点
    - 直接定址法所得地址集合与关键字集合大小相等，不会发生冲突
    - 实际中能用这种哈希函数的情况很少

■ 数字分析法

✧ 构造：对关键字进行分析，取关键字的若干位或其组合作哈希地址

✧ 适于关键字位数比哈希地址位数大，且可能出现的关键字事先知道的情况

例 有80个记录，关键字为8位十进制数，哈希地址为2位十进制数

①② ③ ④⑤⑥ ⑦⑧

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

分析：①只取8
②只取1
③只取3、4
⑧只取2、7、5
④⑤⑥⑦数字分布近乎随机

所以：取④⑤⑥⑦任意两位或两位与另两位的叠加作哈希地址

- 平方取中法
  - 构造：取关键字平方后中间几位作哈希地址
  - 适于不知道全部关键字情况
- 折叠法
  - 构造：将关键字分割成位数相同的几部分，然后取这几部分的叠加和（舍去进位）做哈希地址
  - 种类
    - 移位叠加：将分割后的几部分低位对齐相加
    - 间界叠加：从一端沿分割界来回折送，然后对齐相加
  - 适于关键字位数很多，且每一位上数字分布大致均匀情况

例　关键字为：0442205864，哈希地址位数为4

5 8 6 4
4 2 2 0
0 4
1 0 0 8 8

H(key)=0088

移位叠加

5 8 6 4
0 2 2 4
0 4
6 0 9 2

H(key)=6092

间界叠加

- 除留余数法
  - 构造：取关键字被某个不大于哈希表表长m的数p除后所得余数作哈希地址，即H(key)=key MOD p，p≤m
  - 特点
    - 简单、常用，可与上述几种方法结合使用
    - p的选取很重要；p选的不好，容易产生同义词
- 随机数法
  - 构造：取关键字的随机函数值作哈希地址，即H(key)=random(key)
  - 适于关键字长度不等的情况

- 选取哈希函数，考虑以下因素：
  - 计算哈希函数所需时间
  - 关键字长度
  - 哈希表长度（哈希地址范围）
  - 关键字分布情况
  - 记录的查找频率

■ 处理冲突的方法

✧ 开放定址法

? 方法：当冲突发生时，形成一个探查序列；沿此序列逐个地址探查，直到找到一个空位置（开放的地址），将发生冲突的记录放到该地址中，即

$H_i=(H(key)+d_i)MOD\ m$，i=1,2,……k(k≤m-1)

其中：H(key)——哈希函数

m——哈希表表长

d_i——增量序列

- 分类
  - 线性探测再散列：$d_i=1,2,3,\ldots\ldots m-1$
  - 二次探测再散列：$d_i=1^2,-1^2,2^2,-2^2,3^2,\ldots\ldots\pm k^2(k\le m/2)$
  - 伪随机探测再散列：d_i=伪随机数序列

例 表长为11的哈希表中已填有关键字为17，60，29的记录，
H(key)=key MOD 11,现有第4个记录，其关键字为38，
按三种处理冲突的方法，将它填入表中

0	1	2	3	4	5	6	7	8	9	10
			38	38	60	17	29	38		

(1) H(38)=38 MOD 11=5 冲突
H_1=(5+1) MOD 11=6 冲突
H_2=(5+2) MOD 11=7 冲突
H_3=(5+3) MOD 11=8 不冲突

(2) H(38)=38 MOD 11=5 冲突
H_1=(5+1^2) MOD 11=6 冲突
H_2=(5-1^2) MOD 11=4 不冲突

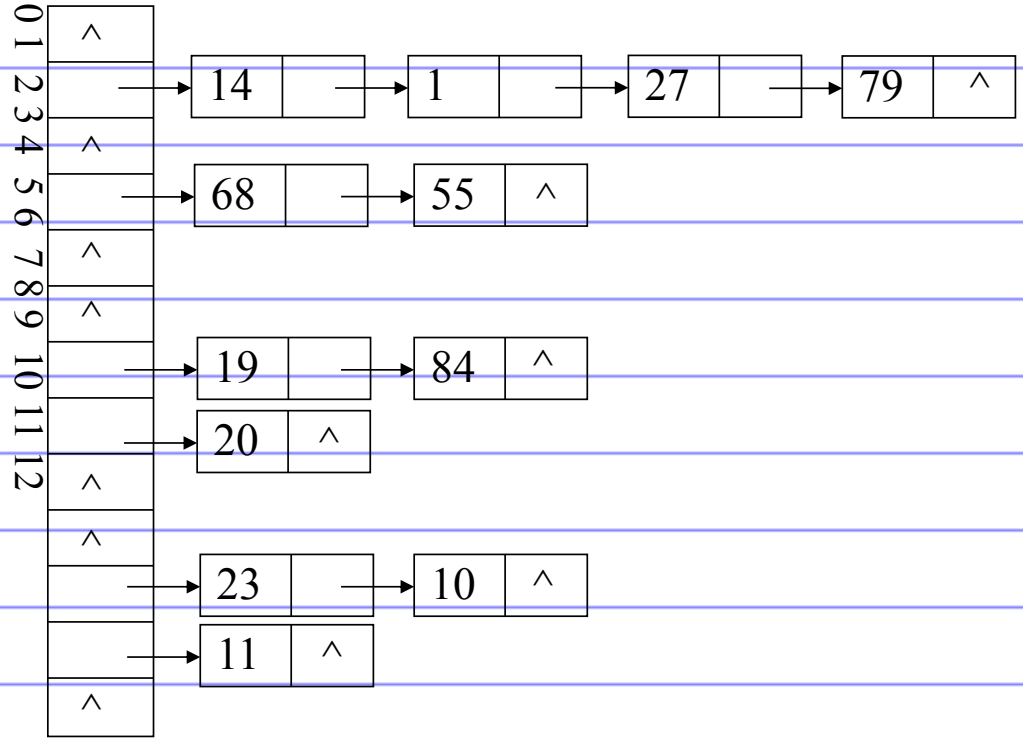
(3) H(38)=38 MOD 11=5 冲突
设伪随机数序列为9，则：
H_1=(5+9) MOD 11=3 不冲突

- 再哈希法
  - 方法：构造若干个哈希函数，当发生冲突时，计算下一个哈希地址，即：Hi=Rhi(key)

    i=1,2,……k

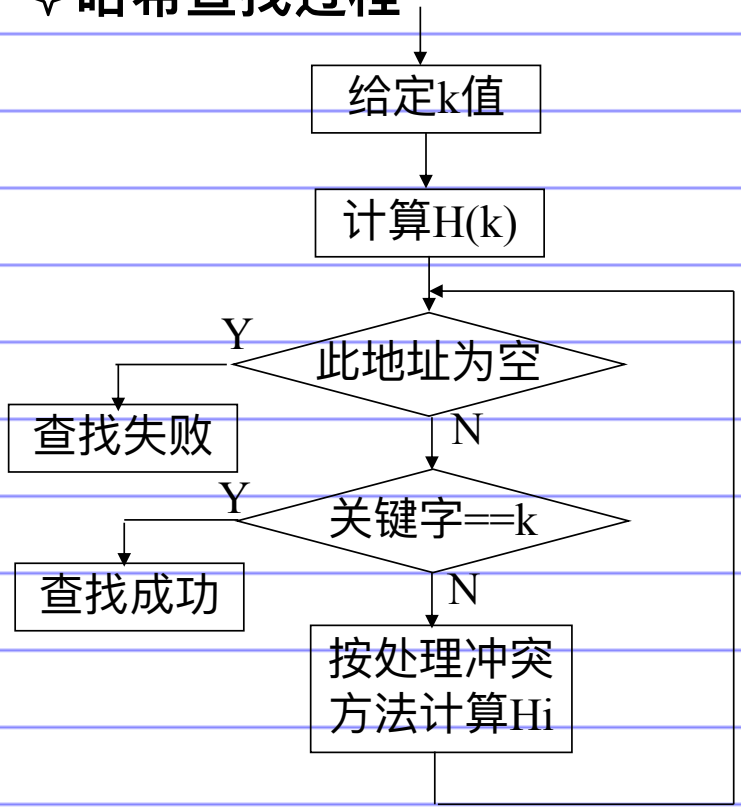
    其中：Rhi——不同的哈希函数
  - 特点：计算时间增加
- 链地址法
  - 方法：将所有关键字为同义词的记录存储在一个单链表中，并用一维数组存放头指针

例　已知一组关键字(19,14,23,1,68,20,84,27,55,11,10,79)
哈希函数为：H(key)=key MOD 13,
用链地址法处理冲突



■ 哈希查找过程及分析

✧ 哈希查找过程



- 哈希查找分析
  - 哈希查找过程仍是一个给定值与关键字进行比较的过程
  - 评价哈希查找效率仍要用ASL
  - 哈希查找过程与给定值进行比较的关键字的个数取决于：
    - 哈希函数
    - 处理冲突的方法
    - 哈希表的填满因子α=表中填入的记录数/哈希表长度

例　已知一组关键字(19,14,23,1,68,20,84,27,55,11,10,79)
哈希函数为：H(key)=key MOD 13, 哈希表长为m=16，
设每个记录的查找概率相等

(1) 用线性探测再散列处理冲突，即Hi=(H(key)+di) MOD m

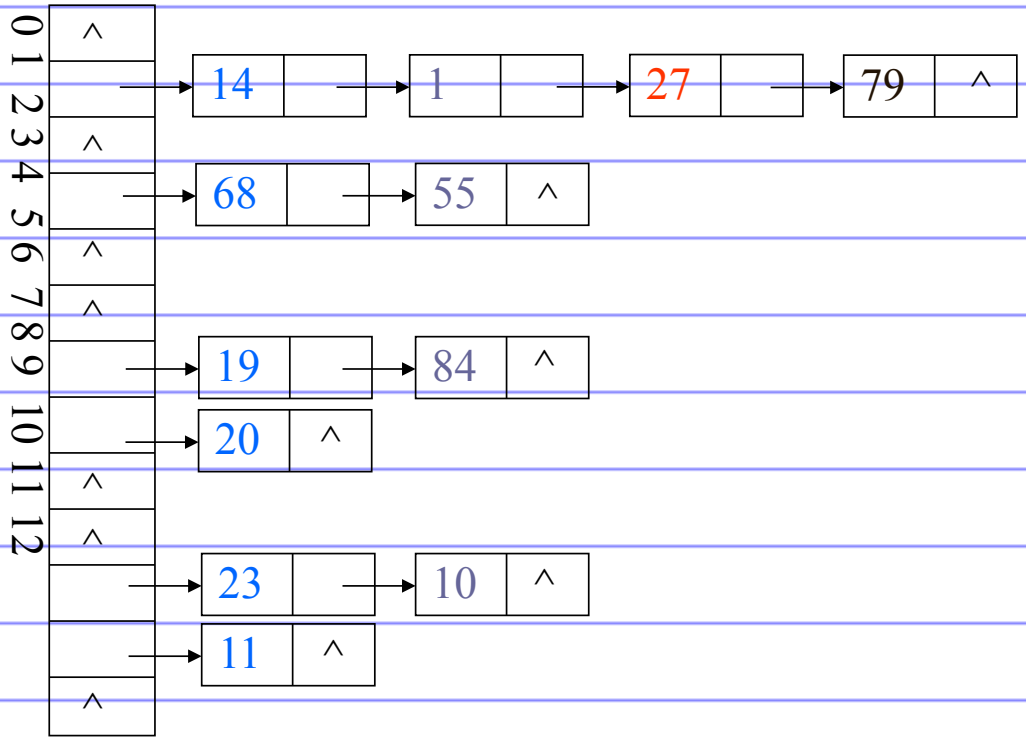
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	14	1	68	27	55	19	20	84	79	23	11	10			

H(19)=6
H(14)=1
H(23)=10
H(1)=1　冲突，H1=(1+1) MOD16=2
H(68)=3
H(20)=7
H(84)=6　冲突，H1=(6+1)MOD16=7
　　冲突，H2=(6+2)MOD16=8
H(27)=1　冲突，H1=(1+1)MOD16=2
　　冲突，H2=(1+2)MOD16=3
　　冲突，H3=(1+3)MOD16=4
H(55)=3　冲突，H1=(3+1)MOD16=4
　　冲突，H2=(3+2)MOD16=5
H(11)=11
H(10)=10　冲突，H1=(10+1)MOD16=11
　　冲突，H2=(10+2)MOD16=12
H(79)=1　冲突，H1=(1+1)MOD16=2
　　冲突，H2=(1+2)MOD16=3
　　冲突，H3=(1+3)MOD16=4
　　冲突，H4=(1+4)MOD16=5
　　冲突，H5=(1+5)MOD16=6
　　冲突，H6=(1+6)MOD16=7
　　冲突，H7=(1+7)MOD16=8
　　冲突，H8=(1+8)MOD16=9

ASL=(1*6+2+3*3+4+9)/12=2.5

关键字(19,14,23,1,68,20,84,27,55,11,10,79)

(2) 用链地址法处理冲突



ASL=(1*6+2*4+3+4)/12=1.75

- 哈希查找算法实现
  - 用线性探测再散列法处理冲突
    - 实现
      - 查找过程：同前
      - 删除：只能作标记，不能真正删除
      - 插入：遇到空位置或有删除标记的位置就可以插入
    - 算法描述：
  - 用外链表处理冲突算法