数据结构

（C语言版）（第2版）

查找

线性表的查找

主讲教师：汪红松

教学目标

01 OPTION 熟练掌握顺序表和有序表（折半查找）的查找算法及其性能分析方法；

02 OPTION 熟练掌握二叉排序树的构造和查找算法及其性能分析方法；

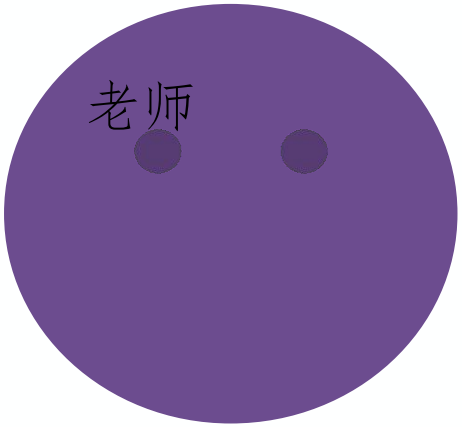
03 OPTION 掌握二叉排序树的插入算法，掌握二叉排序树的删除方法；

04 OPTION 熟练掌握哈希函数的构造

05 OPTION 熟练掌握哈希函数解决冲突的方法及其特点

教 学 内 容 Contents

一、顺序查找
二、折半查找
三、分块查找



查找的基本概念

- 查找表：由同一类型的数据元素（或记录）构成的集合
- 静态查找表：查找的同时对查找表不做修改操作（如插入和删除）
- 动态查找表：查找的同时对查找表具有修改操作
- 关键字：记录中某个数据项的值，可用来识别一个记录
- 主关键字：唯一标识数据元素
- 次关键字：可以标识若干个数据元素

查找算法的评价指标

关键字的平均比较次数，也称平均搜索长度*ASL*(*Average Search Length*)

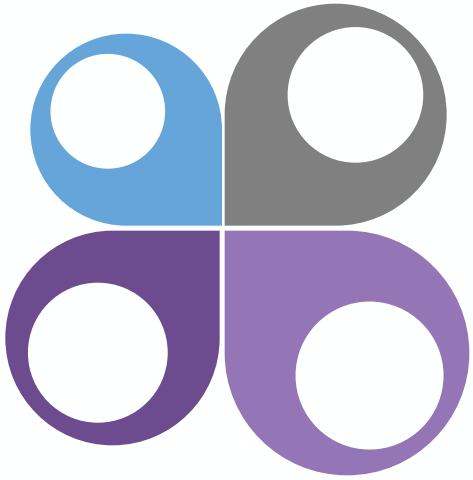
$$ASL = \sum_{i=1}^{n} p_i c_i$$

n：记录的个数

pi：查找第i个记录的概率 (通常认为pi =1/n)

ci：找到第i个记录所需的比较次数

线性表的查找



一、顺序查找（线性查找）

二、折半查找（二分或对分查找）

三、分块查找

一、顺序查找

应用范围：

顺序表或线性链表表示的静态查找表

表内元素之间无序

顺序表的表示

```
typedef struct
{
    ElemType   *R; //表基址
    int        length;    //表长
}SSTable;
```

一、顺序查找

1.在顺序表L中查找值为e的数据元素

```
int LocateELem(SqList L,ElemType e)
{
    for (i=0;i< L.length;i++)
        if (L.elem[i]==e) return i+1;
    return 0;
}
```

改进：把待查关键字key存入表头（“哨兵”），从后向前逐个比较，可免去查找过程中每一步都要检测是否查找完毕，加快速度。

一、顺序查找

1.在顺序表L中查找值为e的数据元素

```
int Search_Seq( SSTable  ST , KeyType  key )

{
   //若成功返回其位置信息，否则返回0
  ST.R[0].key =key;
 for( i=ST.length; ST.R[ i ].key!=key;  - - i  );
 //不用for(i=n; i>0; - -i) 或 for(i=1; i<=n; i++)
   return i;
}
```

一、顺序查找

2.顺序查找的性能分析

（1）空间复杂度：一个辅助空间。

（2）时间复杂度：

1) 查找成功时的平均查找长度

设表中各记录查找概率相等

$ASLs(n)=(1+2+ \dots +n)/n =(n+1)/2$

2)查找不成功时的平均查找长度 $ASLf =n+1$

3.顺序查找算法优缺点

算法简单，对表结构无任何要求（顺序和链式）；

n很大时查找效率较低；

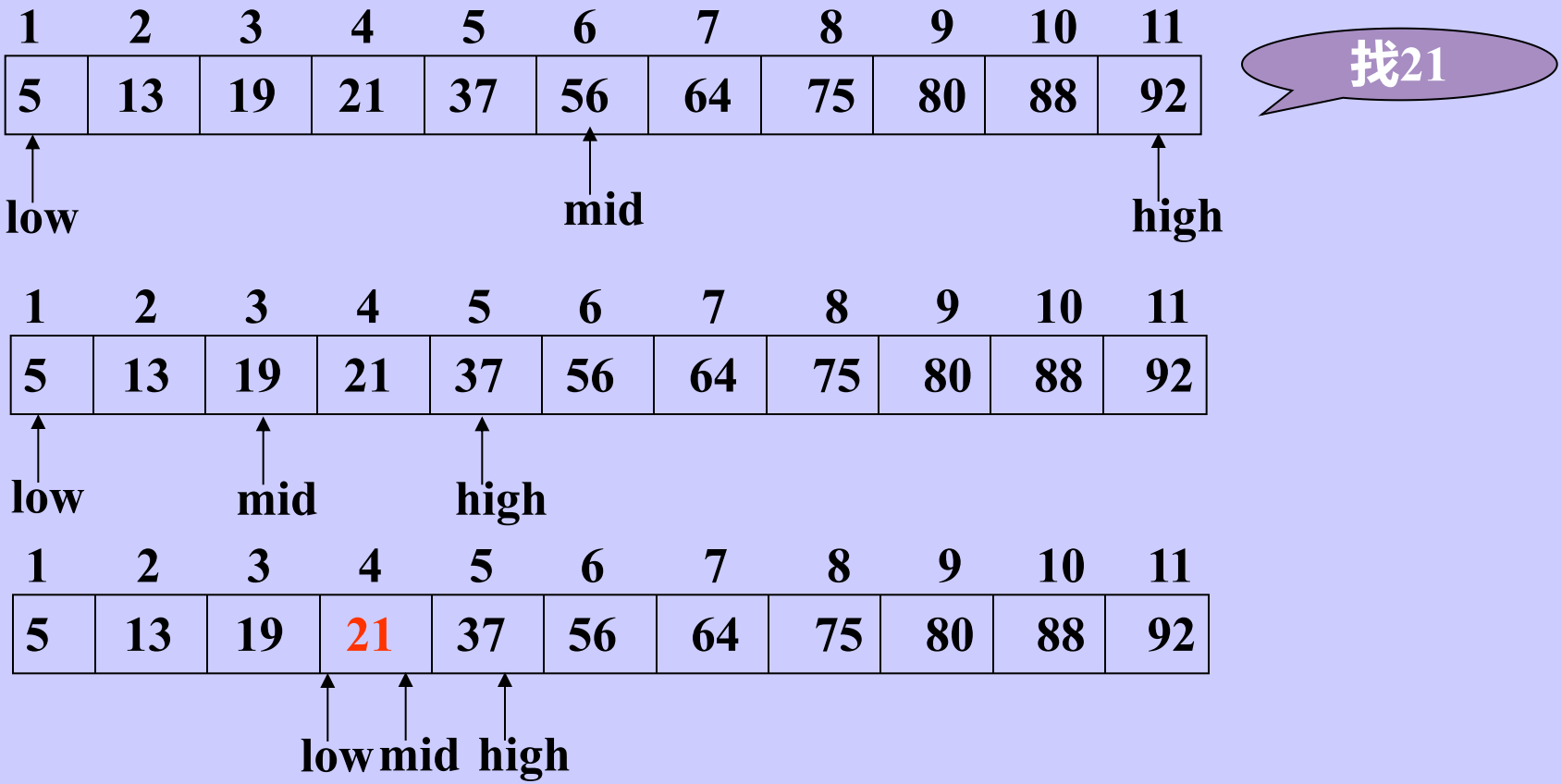
改进措施：非等概率查找时，可按照查找概率进行排序。

二、折半查找

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

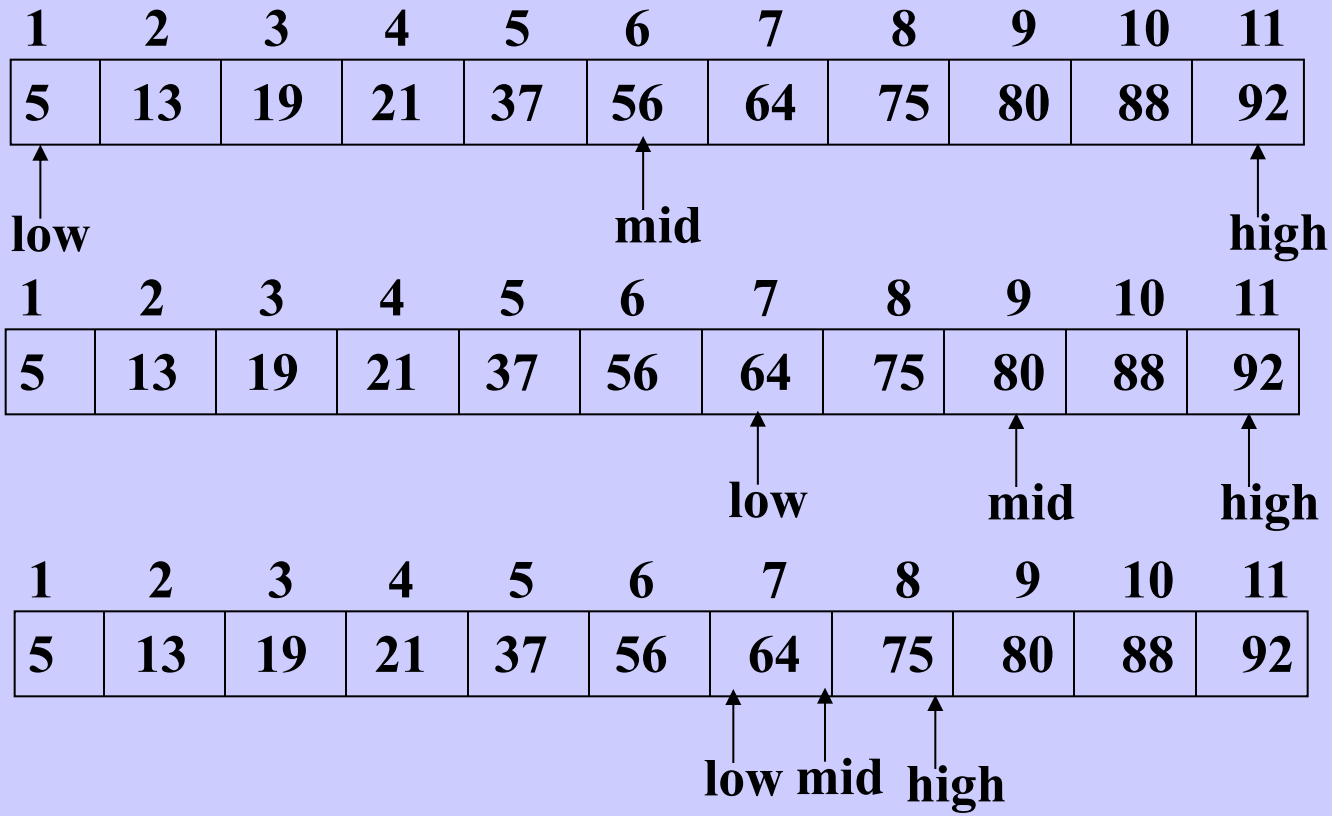


二、折半查找

若k==R[mid].key，查找成功

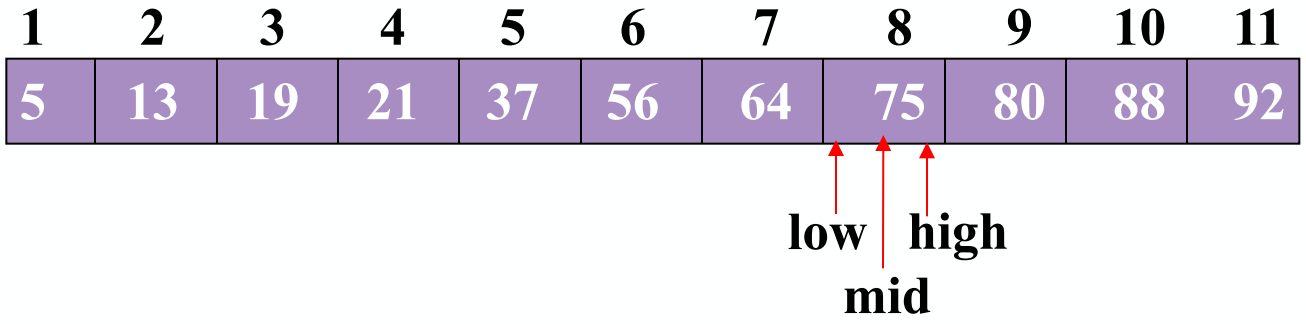
若k<R[mid].key，则high=mid-1

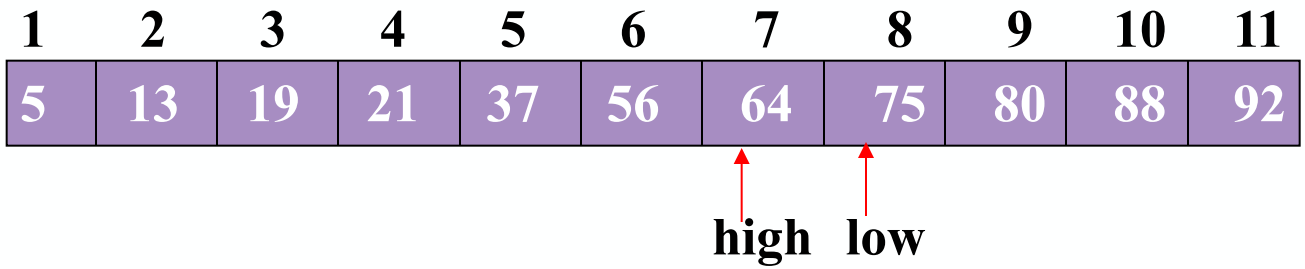
若k>R[mid].key，则low=mid+1



找70

二、折半查找


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

二、折半查找

01 OPTION

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；

02 OPTION

初始时，令low=1,high=n,mid=⌊(low+high)/2⌋；

03 OPTION

让k与mid指向的记录比较；

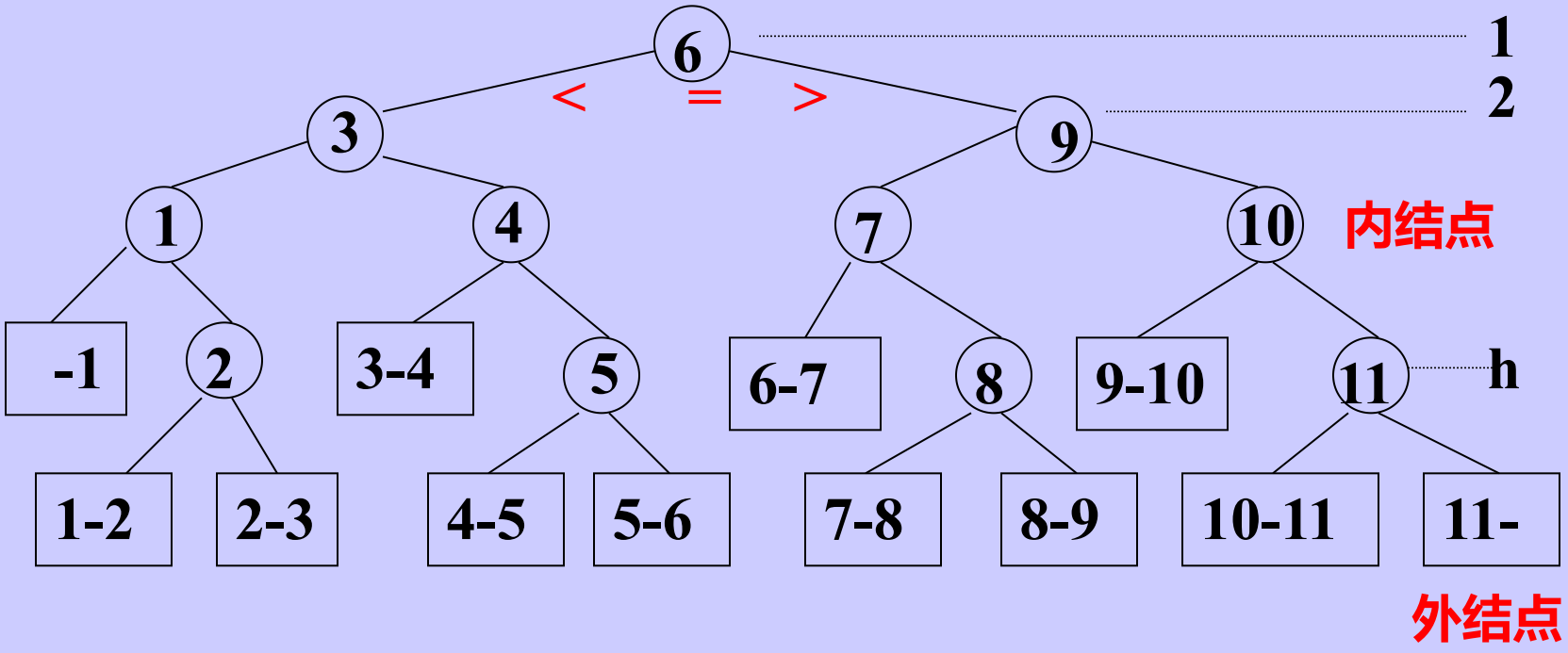
- 若k==R[mid].key，查找成功
- 若k<R[mid].key，则high=mid-1
- 若k>R[mid].key，则low=mid+1

04 OPTION

重复上述操作，直至low>high时，查找失败。

二、折半查找　　折半查找的性能分析 - 判定树

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



若所有结点的空指针域设置为一个指向一个方形结点的指针，称方形结点为判定树的外部结点；对应的，圆形结点为内部结点。

二、折半查找

折半查找的性能分析

查找过程

每次将待查记录所在区间缩小一半，比顺序查找效率高,时间复杂度O(log2 n)；

适用条件

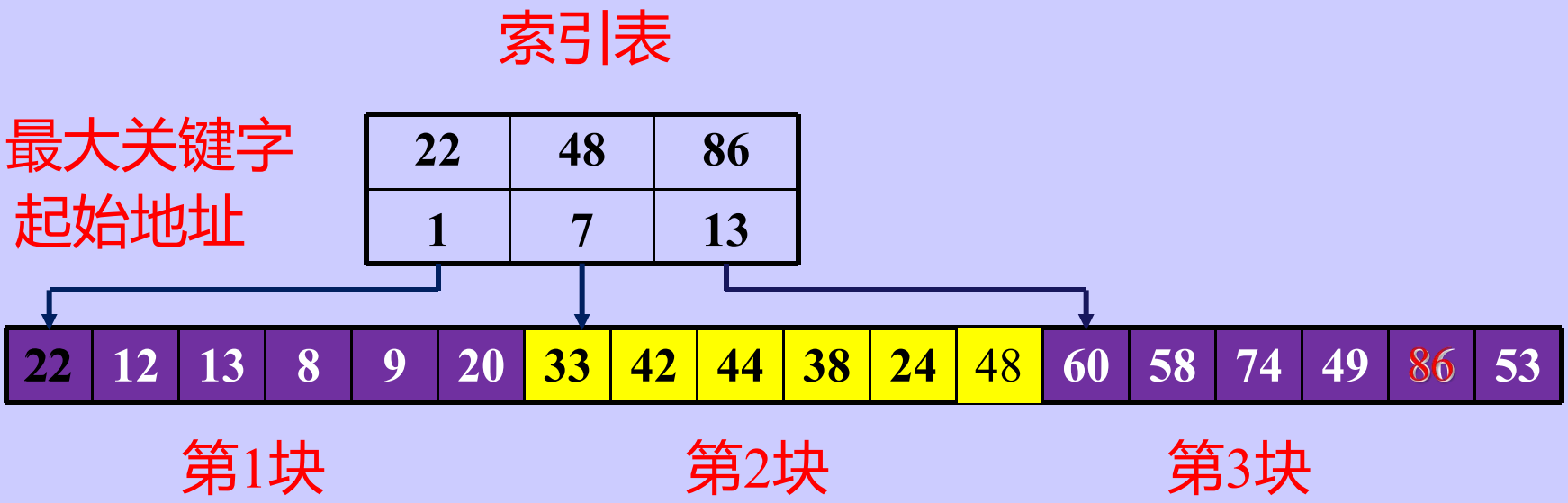
采用顺序存储结构的有序表，不宜用于链式结构。

三、分块查找

（块间有序，块内无序）

分块有序，即分成若干子表，要求每个子表中的数值都比后一块中数值小（但子表内部未必有序）。

然后将各子表中的最大关键字构成一个索引表，表中还要包含每个子表的起始地址（即头指针）。

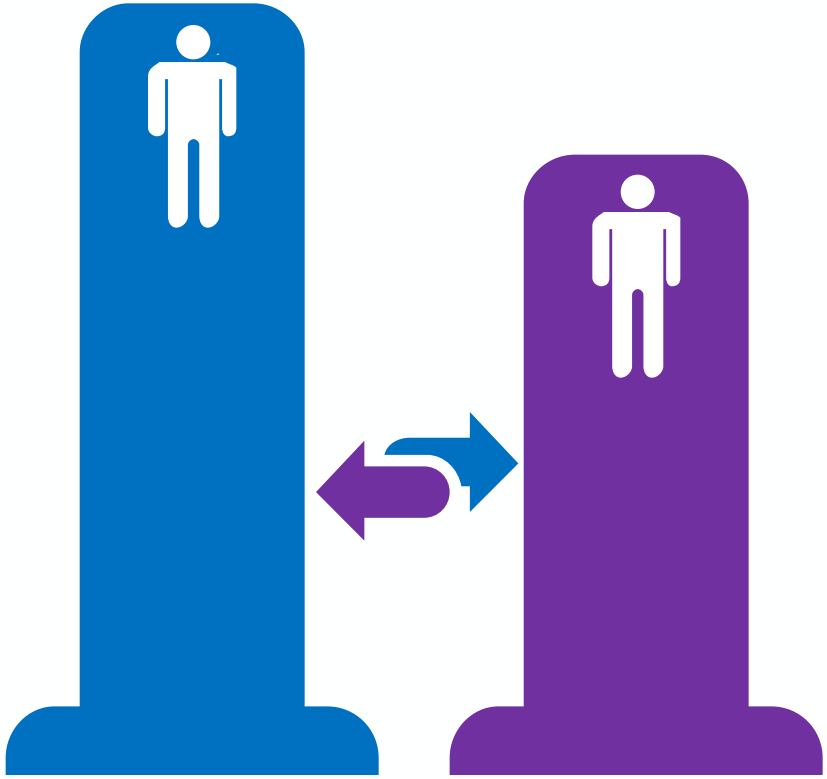


三、分块查找

1.分块查找过程

（1）对索引表使用折半查找法（因为索引表是有序表）；

（2）确定了待查关键字所在的子表后，在子表内采用顺序查找法（因为各子表内部是无序表）。



三、分块查找

2.分块查找性能分析

查找效率：$ASL=L_b+L_w$

对索引表查找的ASL

对块内查找的ASL

$$ASL_{bs} \cong \log_2(\frac{n}{s}+1)+\frac{s}{2} \qquad (\log_2 n \le ASL_{bs} \le \frac{n+1}{2})$$

s为每块内部的记录个数，n/s即块的数目

例如，当n=9，s=3时，ASL_{bs}=3.5,而折半法为3.1,顺序法为5

三、分块查找

3.分块查找优缺点

优点：插入和删除比较容易，无需进行大量移动。

缺点：要增加一个索引表的存储空间并对初始索引表进行排序运算。

适用情况：如果线性表既要快速查找又经常动态变化，则可采用分块查找。

小结

1. 查找的相关概念
2. 线性表的顺序查找算法和折半查找算法及其算法效率