第三章 线性表的查找

3.1 查找的概念

3.1 查找的概念

- **查找（Searching）**就是根据给定的某个值，在查找表中确定一个其关键字等于给定值的数据元素或记录。

查找表（Search Table）是由同一类型的数据元素（或记录）构成的集合。

关键字（Key）是数据元素中某个数据项的值，又称为键值，用它可以标识一个数据元素。也可以标识一个记录的某个数据项（字段），我们称为关键码。

若此关键字可以唯一地标识一个记录，则称此关键字为主关键字（Primary Key）。

那么对于那些可以识别多个数据元素（或记录）的关键字，我们称为次关键字（Secondary Key），不唯一标识一个数据元素（或记录）的关键字，它对应的数据项就是次关键码。

名称	代码	涨跌幅	最新价	涨跌额	买入/卖出价	成交量(手)
中国石油	sh601857	-0.47%	12.68	-0.06	12.68/12.69	391306
工商银行	sh601398	-2.31%	4.66	-0.11	4.65/4.66	442737
中国银行	sh601988	-1.43%	3.45	-0.05	3.45/3.46	194203
招商银行	sh600036	-1.63%	14.52	-0.24	14.52/14.54	385271
交通银行	sh601328	-1.29%	6.10	-0.08	6.09/6.10	347937
中信证券	sh600030	-2.69%	15.22	-0.42	15.22/15.23	597025
中国石化	sh600028	-1.16%	9.38	-0.11	9.37/9.38	538895
中国人寿	sh601628	-0.16%	25.63	-0.04	25.61/25.63	66666
中国平安	sh601318	+1.28%	63.29	+0.80	63.29/63.30	153700
宝钢股份	sh600019	-1.77%	7.21	-0.13	7.21/7.22	211077
中国远洋	sh601919	-2.35%	11.24	-0.27	11.22/11.24	156162
万 科A	sz000002	-1.85%	9.01	-0.17	9.00/9.01	542249

3.1 查找的概念

- 查找按照操作方式可以分为两大类：
  - 静态查找：使用静态查找表进行查找操作。

    静态查找表（Static Search Table）：只作查找操作的查找表。它的主要操作有：

    （1）查询某个“特定的”数据元素是否在查找表中。

    （2）检索某个“特定的”数据元素和各种属性。

  - 动态查找：使用动态查找表进行查找操作。

    动态查找表（Dynamic Search Table）：在查找过程中同时插入查找表中不存在的数据元素，或者从查找表中删除已经存在的某个数据元素。显然动态查找表的操作就是两个：

    （1）查找时插入数据元素。

    （2）查找时删除数据元素。

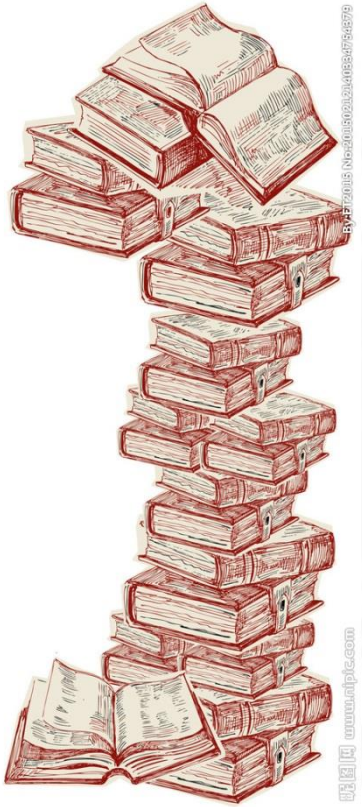
本章内容

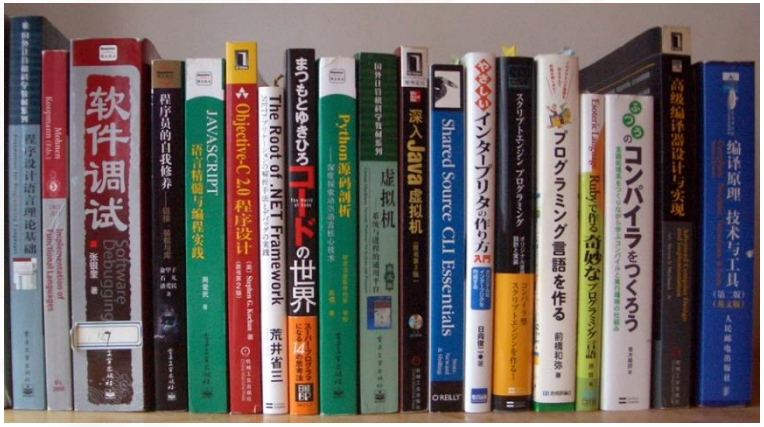
- 查找的概念
- 静态查找
  - 顺序查找
  - 有序查找
    - 折半查找
    - 斐波那契查找
    - 插值查找
  - 线性索引查找
- 动态查找

3.2 静态查找

3.2.1 顺序查找（Sequential Search）

- 从头（尾）记录开始，逐个比较关键字是否等于给定值。

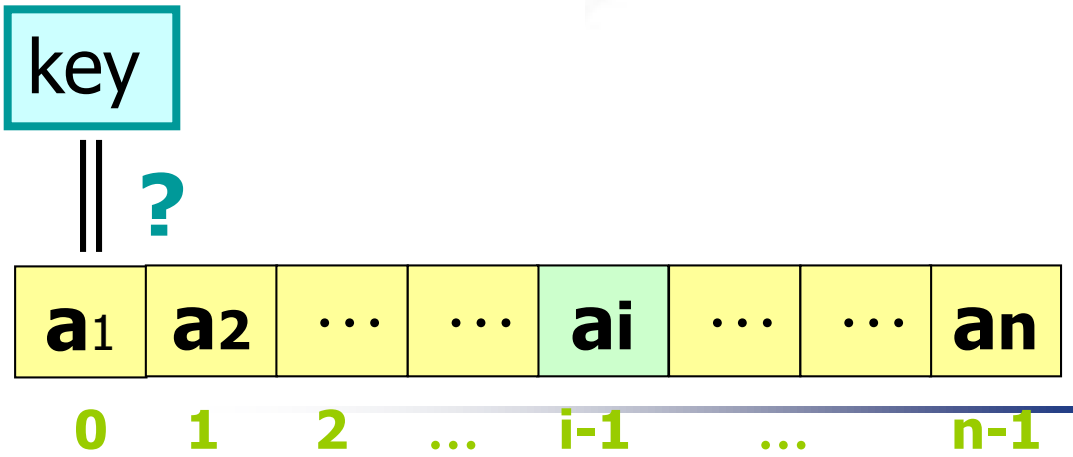




3.2.1 顺序查找

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

     return i+1;
}
```

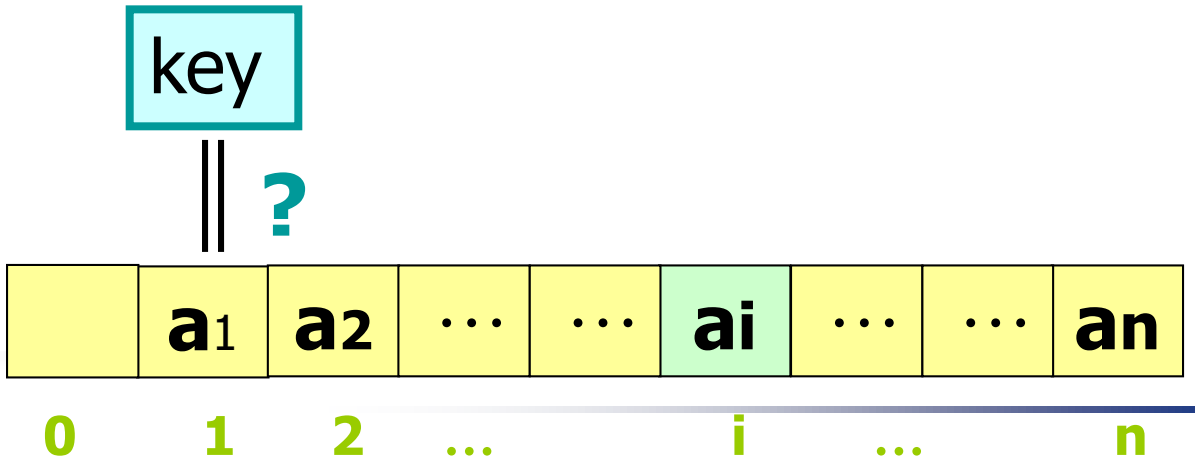
关键字key值



3.2.1 顺序查找

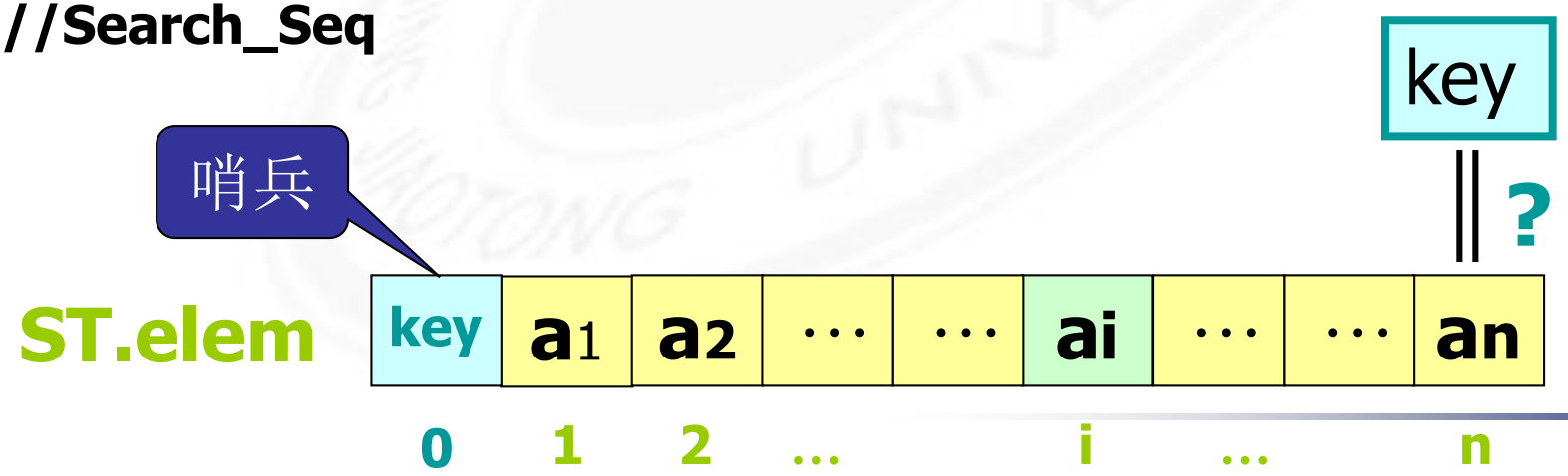
```
int LocateElem_Sq (SqList L, ElemType e) {
    int i=0;
    for(i=0; i<=L.length; i++){
        if(L.elem[i]==e)
            break;
    }
    if(i>L.length)
        return ERROR;

    return i;
}
```

```
    return i+1;
}
```



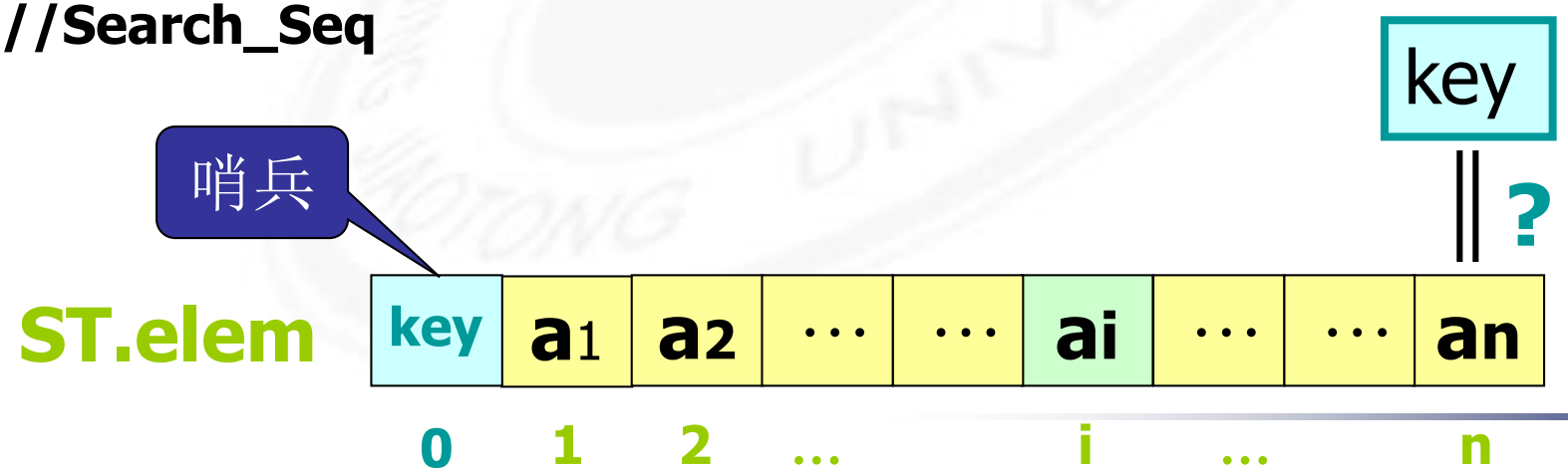
9.2.1 顺序查找

```
int Search_Seq(SSTable L, ElemType e)
{ // 在顺序表L中顺序查找其关键字等于e的数据元素。若找到，则
  //函数值为该元素在表中的位置，否则为0。
  int i;
  L.elem[0]=e;  // 哨兵
  for(i=L.length; L.elem[i]!=e; --i);  //从后往前找
  return i; // 找不到时，i为0
}//Search_Seq
```



9.2.1 顺序查找

```
int Search_Seq(SSTable L, ElemType e)
{ // 在顺序表L中顺序查找其关键字等于e的数据元素。若找到，则
  //函数值为该元素在表中的位置，否则为0。
  int i;
  L.elem[0]=e;  // 哨兵
  for(i=L.length; !EQ(L.elem[i], e); --i);  //从后往前找
  return i; // 找不到时，i为0
}//Search_Seq
```



3.2.2 有序查找

- 如果查找表中的元素按照一定顺序排列，则查找效率会显著提高。
  - 刚才整理书架时，如果按照书名拼音排序，就形成一种有序排序。
- 折半查找

折半查找

假设我们现在有这样一个有序表数组{0,1,16,24,35,47,59,62,73,88,99}[16]，除 0 下标外共 10 个数字。对它进行查找是否存在 62 这个数。我们来看折半查找的算法是如何工作的。

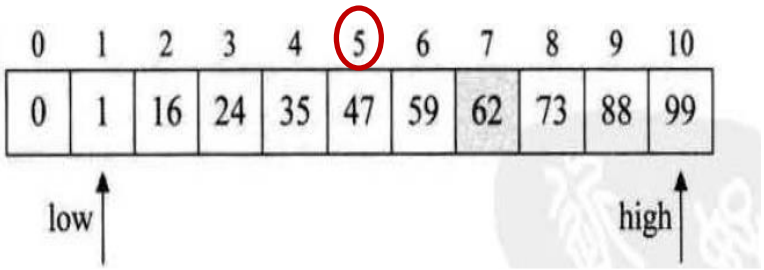
```
int Binary_Search(SqList L,ElemType key){
    int high,low,mid;
    ElemType curKey;
    low = 1;
    high = L.length;
    while (low<=high)
    {
        mid = (low+high)/2;
        GetElem(L,mid,&curKey);
        if(key<curKey)
            high=mid;
        else if(key>curKey)
            low=mid;
        else
            return mid;
    }
```

```
int Binary_Search(SqList L,ElemType key){
    int high,low,mid;
    ElemType curKey;
    low = 1;
    high = L.length;
    while (low<=high)
    {
        mid = (low+high)/2;
        GetElem(L,mid,&curKey);
        if(key<curKey)
            high=mid-1;
        else if(key>curKey)
            low=mid+1;
        else
            return mid;
    }
```

key = 62

```
low = 1;
high = L.length;
while (low<=high)
{
    mid = (low+high)/2;
    GetElem(L,mid,&curKey);
    if(key<curKey)
        high=mid-1;
    else if(key>curKey)
        low=mid+1;
    else
        return mid;
}
```

mid = (1+10)/2 -> 5



mid = (6+10)/2 -> 8

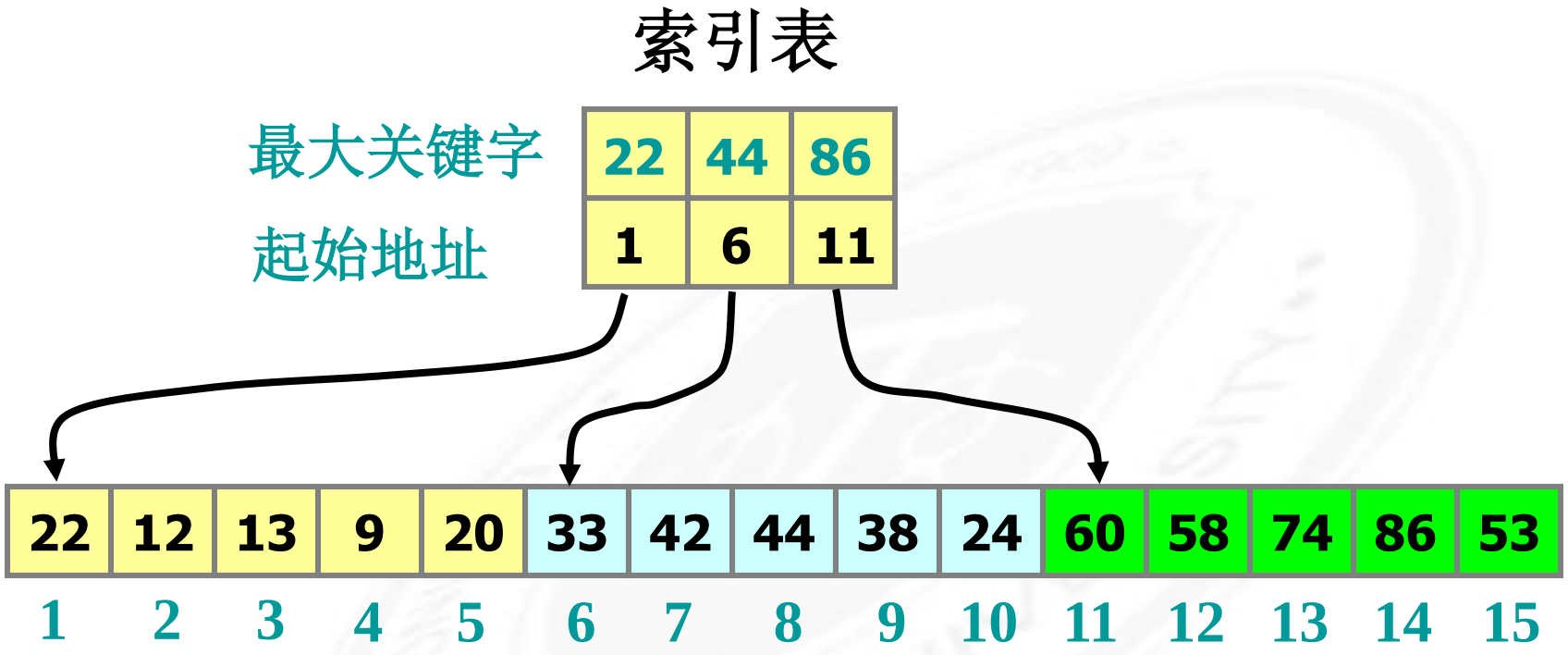
mid = (6+7)/2 -> 6

mid = (7+7)/2 -> 7

9.2.3 线性索引查找

- 索引：把关键字与对应的记录相关联的过程。一个索引由若干个索引项构成，每个索引项至少应包含关键字和其对应的记录在存储器中的位置等信息。
- 索引按照结构可以分为线性索引、树形索引和多级索引。
- 线性索引：将索引项集合组织为线性结构。
  - 重点介绍分块索引

9.2.3 索引顺序表的查找



块查找：顺序查找或折半查找

块内查找：顺序查找

```
//索引表结点结构
typedef struct{
  KeyType key;
  int     stadr;  //起始地址
}indexItem;

//索引表结构定义
typedef struct{
  indexItem *elem;
  int          length;
}indexTable;
```

```
int Search_Idx(SSTable ST, indexTable ID, KeyType kval){
  //在顺序表ST中分块查找其关键字等于给定值kval的数据元素，ID为索引表
  //若找到，则返回该数据元素在ST中的位置，否则返回0
  low=1;high=ID.length; found=FALSE;
  while(low<=high&& !found){//折半查找索引表，确定记录查找区间
    mid=(low+high)/2;
    if(kval<ID.elem[mid].key) low=mid-1;
    else if(kval>ID.elem[mid].key) low=mid+1;
    else{found=TRUE; low=mid;}
  }//while
```

```
  s=ID.elem[low].stadr;//经索引表查找后，下一步的查找范围定位在第low块
  if(high<ID.length)  t=ID.elem[low+1].stadr-1;
  //s和t分别为在ST表中的上界和下界
  else t=ST.length;
  if(ST.elem[t]==kval) return t;
  else{
    ST.elem[0]=ST.elem[t+1];   //暂存ST.elem[t+1]
    ST.elem[t+1].key=kval;      //设置监视哨
    for(k=s; ST.elem[k].key!=kval; k++);
    ST.elem[t+1]=ST.elem[0];  //恢复暂存值
    if(k!=t+1) return k;
    else return 0;
  }//else
}//Search_Idx
```

小结

1. 基本概念: 查找，静态查找表，动态查找表

2. 顺序查找的特点，“监视哨”的作用

3. 有序表的查找
   - 折半查找的过程，算法及折半查找树的分析

4. 索引顺序表的概念及查找方法

实验4查找练习

- 在顺序表库中添加二分法查找函数，修改后的头文件如附件所示，使用附件中的主函数。完成以下功能。
- 创建包含有11个元素的有序表（例如，5,13,19,21,37,56,64,75,80,88,92）的顺序表，同时通过二分查找算法对建立的顺序表查找并插入元素，输出。要求各个操作均以函数的形式实现，在主函数中调用各个函数实现以下操作：

（1） 在顺序表中查找比数字(输入值,例如75)，查找成功，则返回该数据，若未成功，则返回失败。

（2） 在有序表中查找数据元素(输入值,例如36)，查找成功，则返回该数据，若未成功，则返回失败。