9.1 静 态 查 找 表

抽象数据定义：

ADT StaticSearchTable {

数据对象D： D是具有相同特性的数据元素的集合。每个数据元素含有类型相同的关键字，可唯一标识数据元素。

数据关系R： 数据元素同属一个集合。

基本操作 P： Create(&ST, n); Destroy(&ST);

Search(ST, key); Traverse(ST, Visit());

} ADT StaticSearchTable

Create(&ST, n);

操作结果：构造一个含n个数据元素的静态查找表ST。

Destroy(&ST);

初始条件： 静态查找表ST存在；

操作结果： 销毁表ST。

Search(ST, key);

初始条件: 静态查找表ST存在，key 为和查找表中元素的关键字类型相同的给定值；

操作结果: 若 ST 中存在其关键字等于key 的数据元素，则函数值为该元素的值或在表中的位置，否则为“空”。

Traverse(ST, Visit());

初始条件： 静态查找表ST存在，Visit是对元素操作的应用函数

操作结果： 按某种次序对ST的每个元素调用函数Visit()一次且仅一次，一旦Visit()失败，则操作失败。

静态表的查找有：

9.1.1 顺序表的查找

9.1.2 有序表的查找

9.1.4 索引顺序表的查找

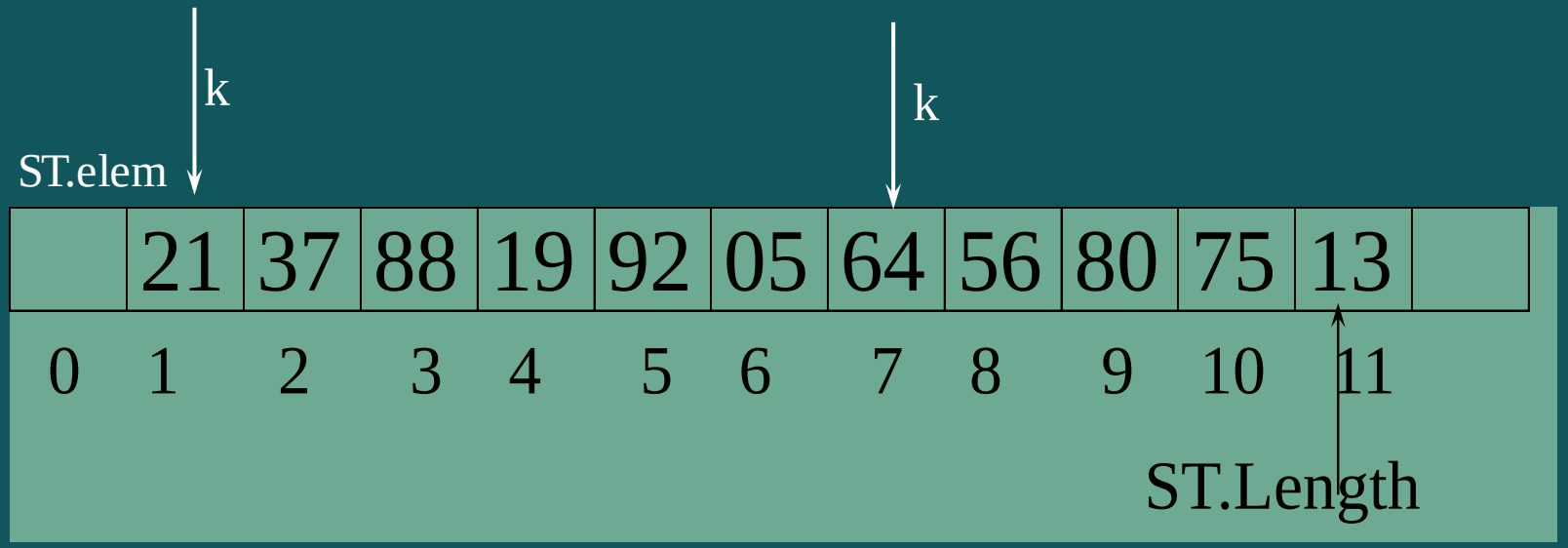
9.1.1 顺序表的查找

以顺序表或线性链表表示静态查找表。则Search函数可用顺序查找来实现。

假设静态查找表的顺序存储结构为

```
typedef  struct {
   ElemType *elem;  // 数据元素存储空间基址，建
                    //表时按实际长度分配，0号单元留空
   int       length;    // 表的长度
} SSTable;
```

回顾顺序表的查找过程：



假设给定值 e=64,

要求 ST.elem[k] = e, 问: k = ?

对应的算法为：

```
int location( SqList L, ElemType& e,
      Status (*compare)(ElemType, ElemType)) {
  k = 1;
  p = L.elem;
  while ( k<=L.length &&
            !(*compare)(*p++,e))) k++;
  if ( k<= L.length)  return k;
  else  return 0;
} //location
```

顺序查找的查找过程：


i
i
ST.elem
64 21 37 88 19 92 05 64 56 80 75 13
0 1 2 3 4 5 6 7 8 9 10 11
key=64
ST.Length
i
i
ST.elem
60 21 37 88 19 92 05 64 56 80 75 13
0 1 2 3 4 5 6 7 8 9 10 11
key=60
ST.Length

从表中最后一个记录开始，逐个进行比较，若某个记录的关键字和给定值相等，则查找成功，找到所查记录；若直至第一个记录，其关键字和给定值都不相等，则查找不成功。

```
int Search_Seq(SSTable ST, KeyType key) {
  // 在顺序表ST中顺序查找其关键字等于key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。
  ST.elem[0].key = key;      //  “哨兵”
  for (i=ST.length; ST.elem[i].key!=key;  --i);
                          // 从后往前找，
  return i;              // 找不到时，i为0
} // Search_Seq
```

算法9.1

分析顺序查找的时间性能:

定义： 查找算法的平均查找长度 ASL

(Average Search Length)

为确定记录在查找表中的位置，需和给定值 进行比较的关键字个数的期望值。

$$ASL = \sum_{i=1}^{n} P_i C_i$$

其中: n 为表长，Pi 为查找表中查找第i个记录的概率，且 ∑Pi=1， Ci为找到该记录时，和给定值比较过的关键字的个数。

对**顺序表**而言，$C_i = n-i+1$

$$ASL = nP_1 + (n-1)P_2 + +2P_{n-1} + P_n$$

在等概率查找的情况下，$P_i = \frac{1}{n}$

在等概率情况下顺序查找的平均查找长度为：

$$ASL_{ss} = \frac{1}{n}\sum_{i=1}^{n}(n-i+1) = \frac{n+1}{2}$$

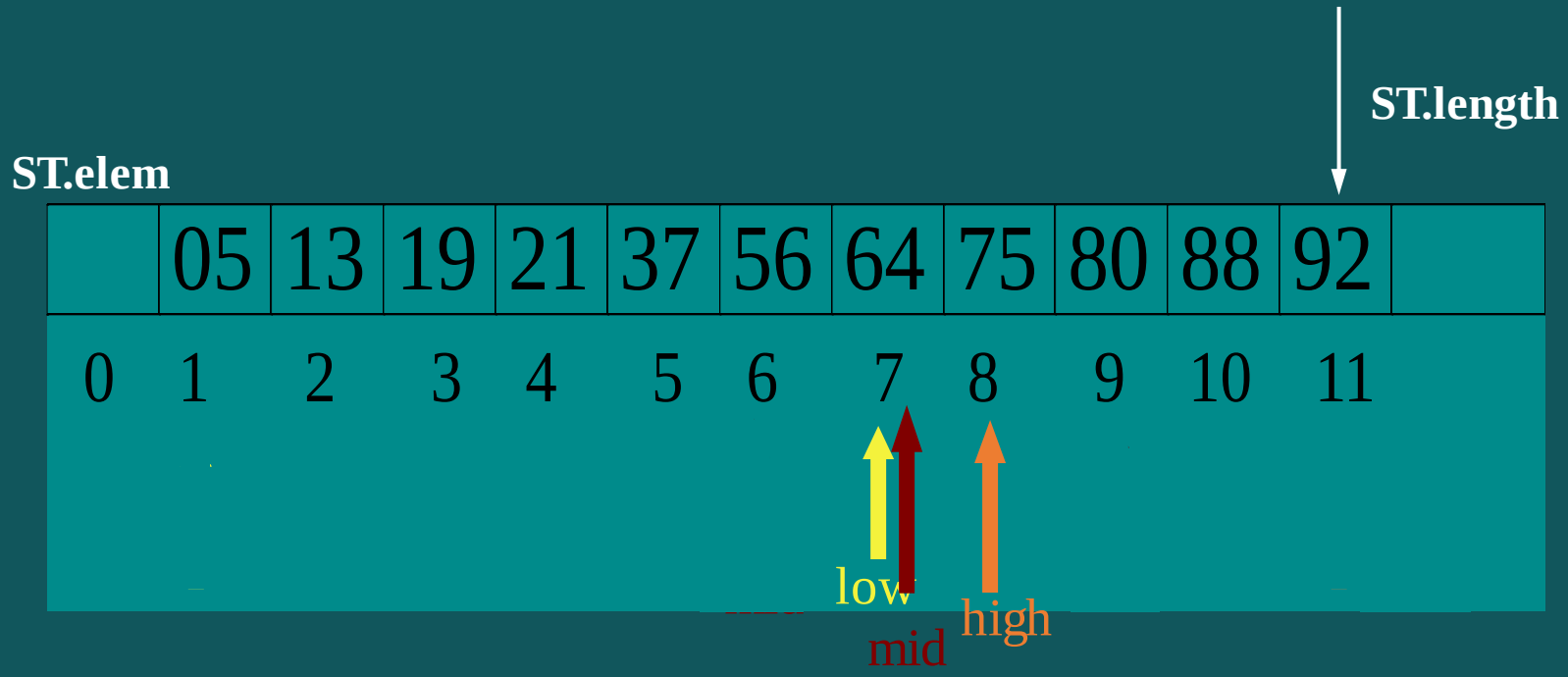
9.1.2 有序表的查找

上述顺序查找表的查找算法简单，但平均查找长度较大，特别不适用于表长较大的查找表。

若以有序表表示静态查找表，则查找过程可以基于“折半”进

。

例如: key=64 的查找过程如下：



low 指示查找区间的下界

high 指示查找区间的上界

mid = (low+high)/2

```
int Search_Bin ( SSTable ST, KeyType key ) {
//在有序表ST中折半查找其关键字等于key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。
  low = 1;  high = ST.length;     // 置区间初值
  while (low <= high) {
    mid = (low + high) / 2;
    if  ( EQ (key , ST.elem[mid].key) )
      return  mid;          // 找到待查元素
    else  if ( LT (key , ST.elem[mid].key) )
      high = mid - 1;       // 继续在前半区间进行查找
    else  low = mid + 1; // 继续在后半区间进行查找
  }
  return 0;                 // 顺序表中不存在待查元素
} // Search_Bin
```

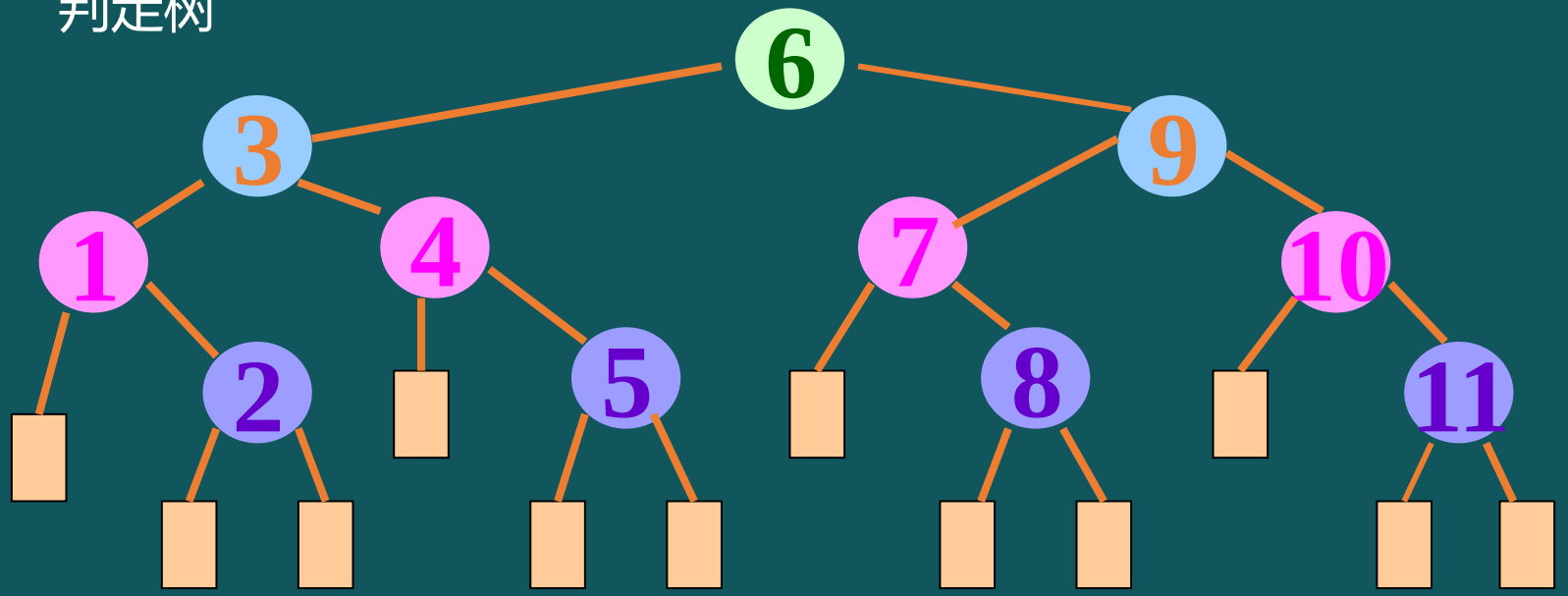
算法9.2

分析**折半查找**的平均查找长度

先看一个具体的情况，假设：n=11

i	1	2	3	4	5	6	7	8	9	10	11
Ci	3	4	2	3	4	1	3	4	2	3	4

判定树



一般情况下，表长为 n 的折半查找的判定树的深度和含有 n 个结点的完全二叉树的深度相同。

假设 $n=2^h-1$ 并且查找概率相等，则 ：

$$ASL_{bs}=\frac{1}{n}\sum_{i=1}^{n}C_i=\frac{1}{n}\left[\sum_{j=1}^{h}j*2^{j-1}\right]=\frac{n+1}{n}\log_2(n+1)-1$$

在n>50时，可得近似结果

$$ASL_{bs}\approx\log_2(n+1)-1$$

9.1.4 索引顺序表的查找

若以索引顺序表表示静态查找表，则Search函数可用分块查找来实现。

分块查找又称索引顺序查找，是顺序查找的一种改进方法。在此方法中，除表本身外，还需建立一个‘索引表’。如下图所示：

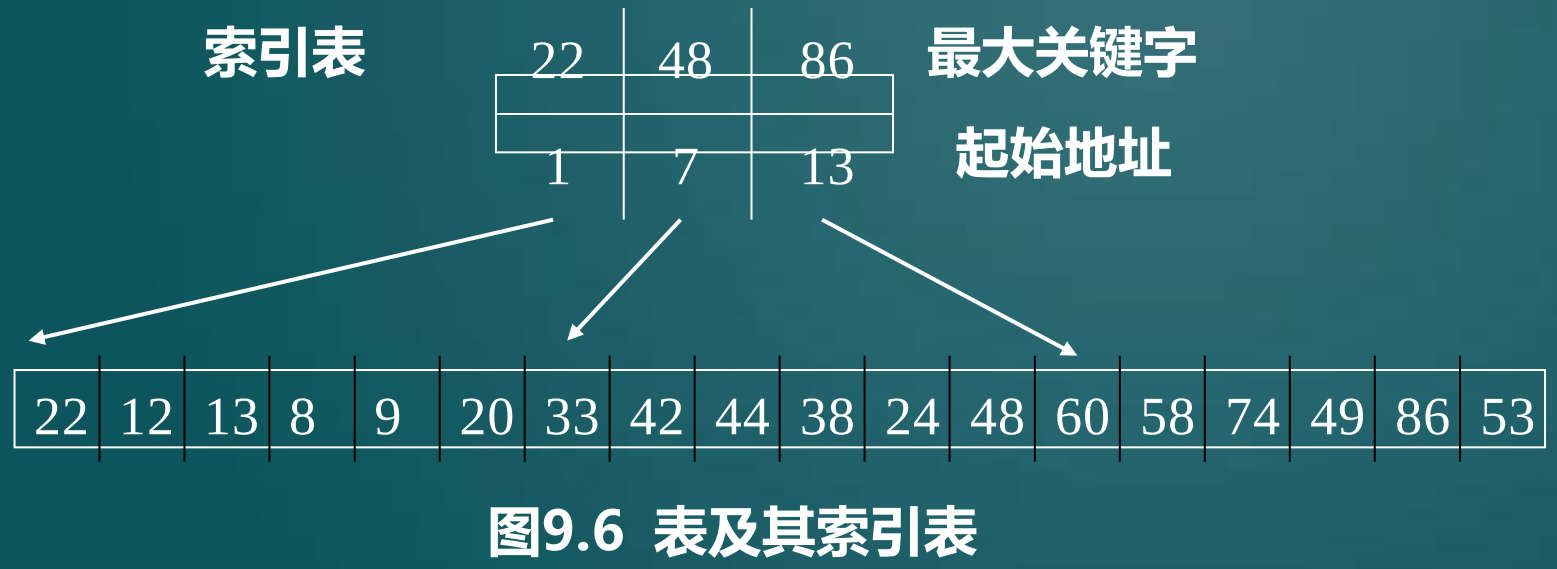


图9.6 表及其索引表

查找过程：

1）由索引确定记录所在区间；

2）在顺序表的某个区间内进行查找。

可见，**索引顺序查找**的过程也是一个**“缩小区间”**的查找过程。

注意：索引可以根据查找表的特点来构造。

索引顺序查找的平均查找长度 =

查找“索引”的平均查找长度

\+ 查找“顺序表”的平均查找长度