教学内容---第九章

DS/SS

1. 绪论
2. 线性表
3. 栈、队列和串
4. 数组
5. 广义表
6. 树和二叉树
7. 图
8. 动态存储管理
9. 查找
10. 内部排序
11. 外部排序
12. 文件

9.1查找的概念和思路

DS/SS

基本概念和术语

- 查找表(Search Table)：同一类型的数据元素组成的集合。
- 静态查找表(Static Search Table)：若对查找表只作“查找”操作（查询某个“特定的”数据元素是否在表中并检查该数据元素各种属性），则称此类查找表为静态查找表。
- 动态查找表(Dynamic Search Table)：若对查找表不仅作“查找”操作，而且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，则称此类查找表为动态查找表。
- 查找(Searching)：在一个含有众多数据元素的查找表中找出某个“特定的”数据元素。（以关键字为依据）
- 关键字(Key)：数据元素中某个数据项的值，用它可以标识一个数据元素。
- 主关键字(Primary Key)：若关键字可以唯一标识一个数据元素，则称此关键字为主关键字。
- 次关键字(Secondary Key)：用以标识若干数据元素的关键字

9.1查找的概念和思路（续）

DS/SS

基本概念和术语

- 查找成功(Searching Success)： 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。若表中存在这样的一个元素，则称查找成功，查找结果为整个元素信息或者指示该元素在查找表中的位置。
- 查找不成功(Searching Failed)： 一次查找过程中，若查找表中不存在关键字等于给定值的元素，则称查找不成功，查找结果为一个空元素或者一个空指针。
- 查找成功时平均查找长度： 为确定元素在查找表中的位置，需和给定值进行比较的关键字个数的期望值称为查找算法在查找成功时平均查找长度。
- 查找不成功平均查找长度： 查找不成功时和给定值进行比较的关键字个数的期望值称为在查找不成功时平均查找长度。
- 平均查找长度(Average Search Length)： 查找算法在查找成功时平均查找长度和查找不成功时平均查找长度之和。

9.1查找的概念和思路（续）

DS/SS

基本思路

- 查找表中元素关系松散，直接在原始查找表中查找的效率很低。
- 查找一般思路：向查找表上依附其他数据结构，使查找表中数据元素之间关系约束增加，从而利于查找过程的进行。
- 根据依附的数据结构不同，可以有不同的查找过程、查找特点和查找性能。
- 依附数据结构的选择取决于问题域特点和查找表本身特性。

9.2静态查找表

DS/SS

抽象数据类型定义

ADT StaticSearchTable {

数据对象：V={具有相同性质的数据元素,各个数据元素均含有类型相同，可唯一标识数据元素的关键字}

数据关系：R: 集合关系

基本操作：P:

Create(&ST,n);

Destroy(&ST);

Search(ST,key);

Traverse(ST,Visit())

}ADT StaticSearchTable

9.2静态查找表（续）

DS/SS

顺序表的查找

顺序表查找=查找表+线性表+顺序存储

顺序查找

```
//-------------静态查找表存储表示---------------
 typedef struct {
     ElemType       *elem;    //0号单元留空
     int    length;
}SSTable;
```

9.2静态查找表（续）

DS/SS

顺序表的查找

顺序查找(Sequential Search)过程：

从表中最后一个元素开始，逐个进行元素的关键字和给定值的比较，若某个元素的关键字和给定值相等，则查找成功，找到所查元素；反之，若直至第一个记录，其关键字和给定值比较都不等，则表明表中没有所查元素，查找不成功。

```
int Search_Seq(SSTable ST, KeyType key ){
  //若找到，含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; !EQ(ST.elem[i].key, key);  --i );  //从后往前找
    return  i ;          //找不到，i为0（这也是哨兵的作用所在）
} Search_Seq
```

性能分析：

n = ST.length; $P_i = 1/n$，　$q_i = 1/n$　（假设的前提）

$ASL_{ss成功} = nP_1+(n-1)P_2+\ldots+2P_{n-1}+P_n =(n+1)/2$

$ASL_{ss不成功} =$　$(n+1)\ (q_1+q_2+\ldots+q_{n-1}+q_n)=n+1$

顺序查找的优点：

*算法简单、适用面广；

*对线性关系没有其他要求；

*对存储结构没有要求

顺序查找的缺点：

*平均查找长度大；

9.2静态查找表（续）

DS/SS

有序表的查找

有序表查找=查找表+线性表+有序性+顺序存储

```
//------------静态查找表存储表示--------------
 typedef struct {
     ElemType      *elem;   //0号单元留空
     int    length;
}SSTable;
```

四种查找方法：

*折半查找　　*顺序查找

*斐波那契查找　　*插值查找

9.2静态查找表（续）

DS/SS

有序表的折半查找

折半查找(Binary Search)过程：

先确定待查元素所在范围（区间），然后逐步缩小范围直到找到或找不到该元素为止。具体而言，折半查找是以处于区间中间位置记录的关键字和给定值比较，若相等，则查找成功，若不等，则缩小范围，直至新的区间中间位置记录的关键字等于给定值或者查找区间的大小小于零时（表明查找不成功）为止。

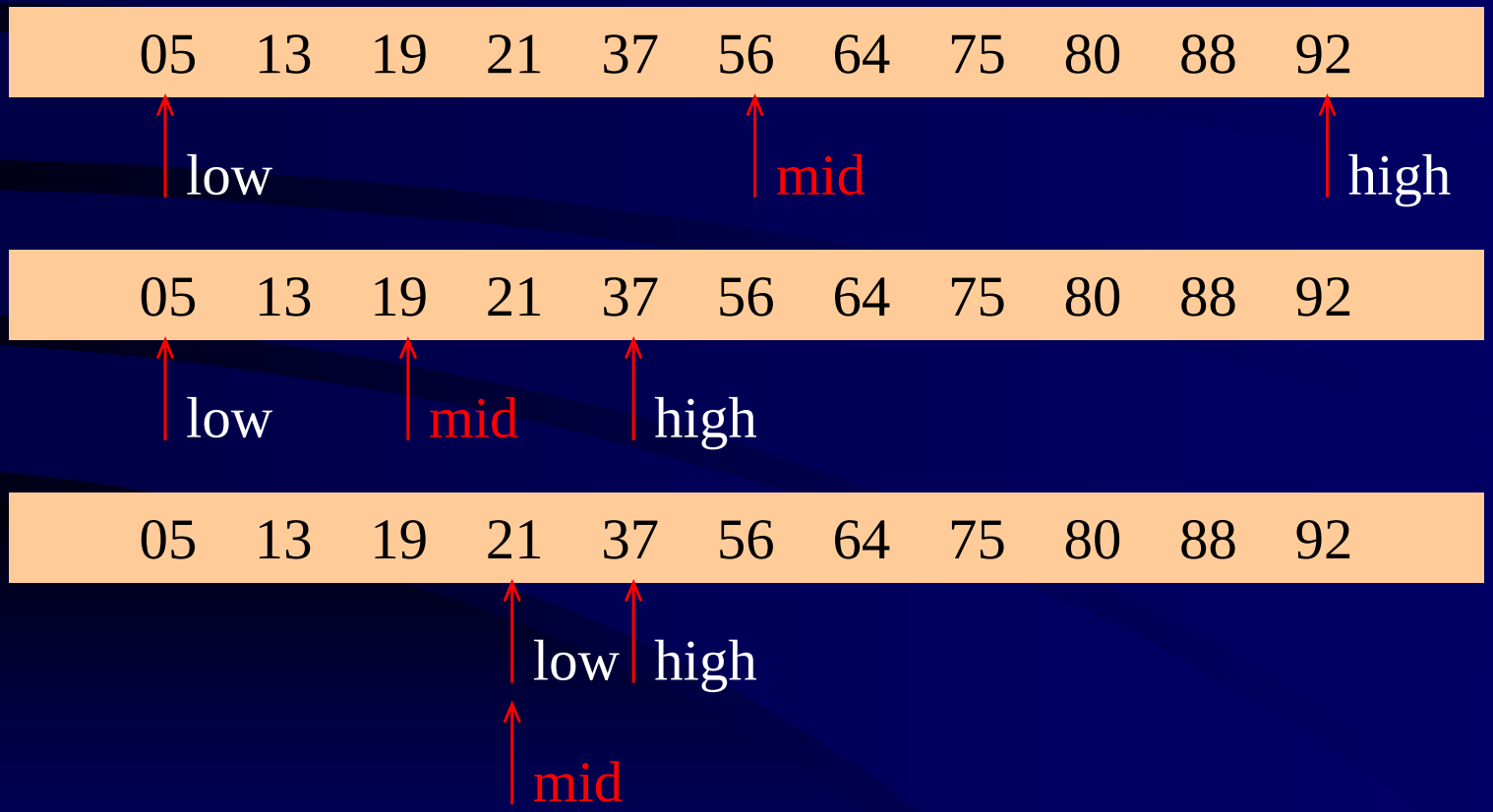
9.2静态查找表（续）

```
int Search_Bin(SSTable ST, KeyType key ){
 //若找到，含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;     //查找成功，返回位置
    else if  LT(key, ST.elem[mid].key)  high = mid – 1;  //继续在前半区间
         else  low = mid +1;                    //继续在后半区间查找
   }
    return  0;              //查找失败          } Search_Bin
```

9.2静态查找表（续）

有序表的折半查找

设给定值key = 21,则查找过程为：



9.2静态查找表（续）

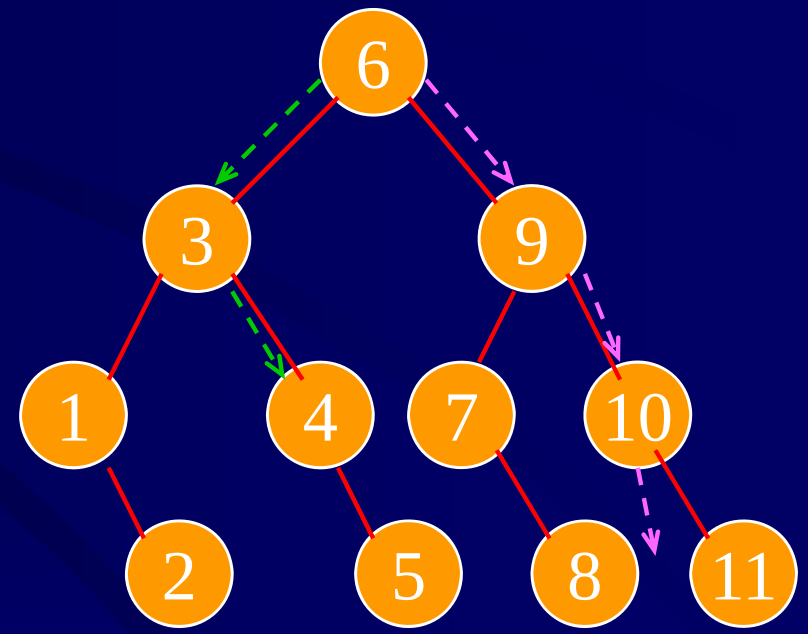
有序表的折半查找

一个位置查找成功或者失败的查找长度只与元素位置有关。

05	13	19	21	37	56	64	75	80	88	92

设给定值key = 21,则查找路径为：

设给定值key = 85,则查找路径为：



9.2静态查找表（续）

折半查找过程中，找到有序表中任何一个元素的过程就是走了一条根结点到与该元素相应的结点的路径，和给定值比较的关键字个数恰为该结点在判定树上的层次数。

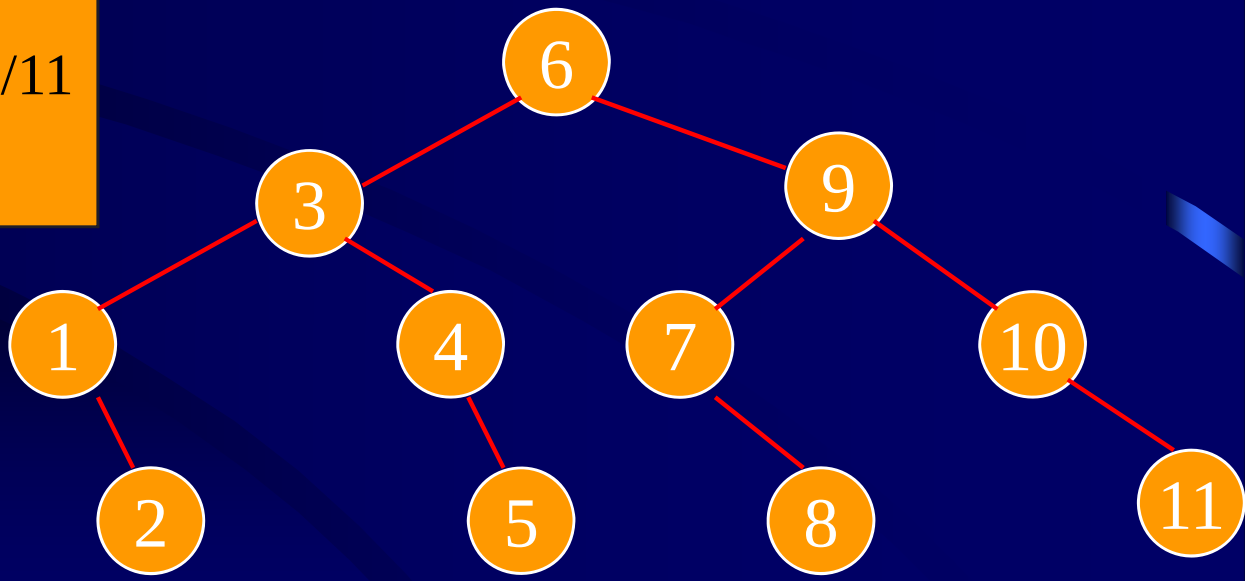
性能分析：假定有序表长度$n = 2^h-1$, $P_i = 1/n$

$$ASL_{bs成功} = (1/n)*(1*1+2*2+\ldots+h*(2^{h-1})) =((n+1)/n)*\log_2(n+1)-1$$

当n=11时：

$ASL_{bs成功}$ =(1*1+2*2+3*4+4*4)/11

=3



9.2静态查找表（续）

如下图：方形结点为构造的判定树的外部结点。

折半查找查找不成功的过程就是走了一条根结点到外部结点的路径，和给定值进行比较的关键字个数等于该路径上内部结点的个数。

当n=11时：　$ASL_{bs不成功}$ =(3*4+4*8)/12 =11/3

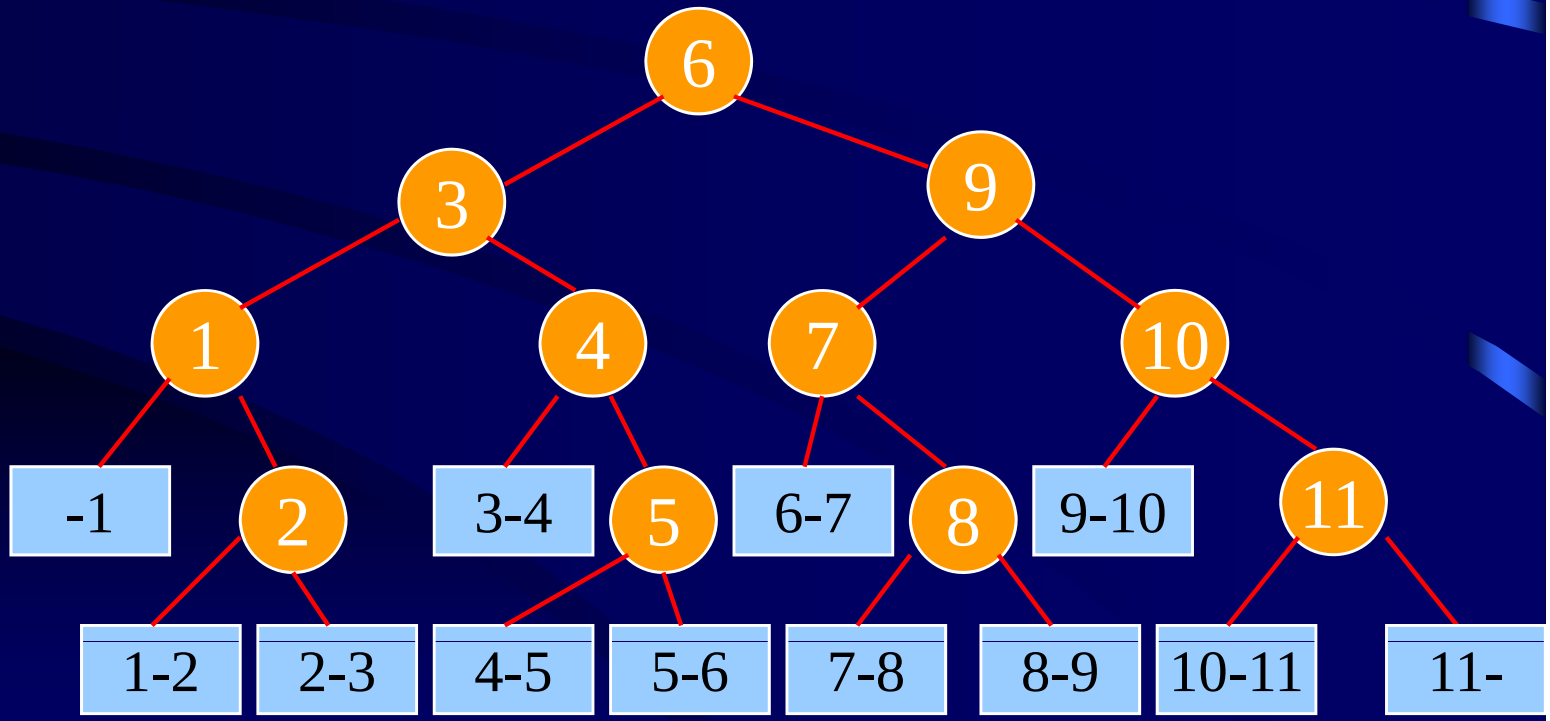
折半查找的优点：

*效率高；

折半查找的缺点：

*只适用于有序表

*限于顺序存储



9.2静态查找表（续）

DS/SS

有序表的顺序查找

有序表顺序查找过程：

从表中第一个元素开始，逐个进行元素的关键字和给定值的比较，若某个元素的关键字和给定值相等，则查找成功，找到所查元素；反之，由于查找表中元素本来有序（设为从小到大有序），如给定值比某个元素关键字小，则表明表中没有所查元素，查找不成功。

性能分析：

n = ST.length; $P_i = 1/n$, $q_i = 1/(n+1)$ （假设的前提）

$ASL_{ss成功} = nP_1+(n-1)P_2+\ldots+2P_{n-1}+P_n =(n+1)/2$

$ASL_{ss不成功} = 1q_1+2q_2+\ldots+(n-1)q_{n-1}+nq_n +(n+1)q_{n+1} =(n+2)/2$

9.2静态查找表（续）

DS/SS

有序表的斐波那契查找

斐波那契序列：$F_0=0$，$F_1=1$，$F_i=F_{i-1}+F_{i-2}$，$i>=2$

斐波那契查找过程：

根据斐波那契序列对表进行分割。假设开始时表中元素个数比某个斐波那契数小1，即$n=F_u-1$，然后将给定值key和ST.elem[F_{u-1}].key进行比较，若相等，则查找成功；若key< ST.elem[F_{u-1}].key，则继续在自ST.elem[1]至ST.elem[F_{u-1}-1]的子表中进行查找，否则继续在自ST.elem[F_{u-1}+1]至 ST.elem[F_u-1]的子表中进行查找，后一子表的长度为$F_{u-2}-1$。

平均性能比折半查找好，但最坏情况下性能比折半查找差。

9.2静态查找表（续）

有序表的插值查找

查找思想：

根据给定值key来确定进行比较的关键字ST.elem[i].key。令 i= (key-ST.elem[L].key)(H-L+1)/(ST.elem[H].key-ST.elem[L].key), 其中ST.elem[L]和ST.elem[H]分别为有序表中具有最小关键字和最大关键字的元素。

适用于关键字均匀分布的表；表长较大时，平均性能比折半查找好。

9.2静态查找表（续）

索引顺序表的查找

索引顺序查找=查找表+线性表+分块有序性

索引顺序表特征：

索引顺序表包括顺序表和索引表两部分。顺序表分成若干子表（块），每个子表对应索引表中一个索引项。每个索引项包括两部分：关键字项（其值为该子表内的最大关键字）和指针项（指示该子表的第一个元素在表中位置）。索引表按关键字有序，顺序表分块有序（第i个子表中所有元素关键字均大于第i-1个子表中最大关键字，i从2至b（b为子表个数））。

分块查找过程分两步：第一步确定待查元素所在块，第二步在块中顺序查找。

9.2静态查找表（续）

DS/SS

索引顺序表的查找

性能分析：（将长度为n的表均匀分为b块，每块s个元素）

$ASL_{bs} = L_b + L_w$

其中L_b为查找索引表确定所在块地平均查找长度，L_w为在块中查找元素的平均查找长度。

若用顺序查找确定所在块，则有

$ASL_{bs} = L_b + L_w = (b+1)/2 + (s+1)/2 = (n/s+s)/2+1$

当$(n/s)=s$，即$s=\sqrt{n}$时，ASL_{bs}最小$=(\sqrt{n}+1)$

若用折半查找确定所在块，则有

$ASL_{bs} = L_b + L_w$ 约等于 $\log_2(n/s+1)+s/2$