第8章 查找算法

知识要点

(1) 理解查找的概念；
(2) 掌握静态查找表的方法；
(3) 熟悉并能运用散列技术解决实际问题；
(4) 熟悉线性索引以及树形索引。

薄钧戈
2021年5月29日

上课安排

- 回顾：排序算法
  - 简单排序：冒泡、简单插入、简单选择
  - 高级排序：希尔、快速、归并、锦标赛、堆排序
  - 桶排序、计数排序、基数排序
- 查找
  - 顺序、折半查找
  - 哈希查找
  - 索引

各种排序算法的比较

表 7-1 各种排序方法的比较

排序方法	时间复杂度			辅助存储	稳定性
	最好情况	平均情况	最坏情况		
直接插入排序	O(n)	O(n2)	O(n2)	O(1)	√
希尔排序	O(n1.5)			O(1)	×
冒泡排序	O(n)	O(n2)	O(n2)	O(1)	√
快速排序	O(nlog2n)	O(nlog2n)	O(n2)	O(log2n)	×
简单选择排序	O(n2)	O(n2)	O(n2)	O(1)	×
堆排序	O(nlog2n)	O(nlog2n)	O(nlog2n)	O(1)	×
归并排序	O(nlog2n)	O(nlog2n)	O(nlog2n)	O(n)	×
基数排序	O(d(n+rd))	O(d(n+rd))	O(d(n+rd))	O(rd)	√

结论

（1）从平均时间性能而言，一般认为快速排序最佳，但快速排序在最坏情况下的时间性能不如堆排序和归并排序。对于n值很大而关键字位数较小的序列，基数排序的速度最快。

（2）从稳定性而言，一般简单排序算法是稳定的，高级排序算法不稳定，但归并排序是稳定的。

快速排序-讨论

- 思考：
  - 快速排序在平均情况下的时间复杂度是多少：
    - 也就是首元素划分后每个位置概率都相等，$C(N)= \Theta(NlogN)$
    - 如何计算？
  - 如果是均衡化分的时间复杂度是多少？
    - 也就是子问题的规模比不变，比如1：3
    - $C(N)=C(N/3)+ C(2N/3)+N=\Theta(NlogN)$
    - 提示：可用递归树求解

单选题 1分

设置

以下关于冒泡和选择排序算法的叙述何者正确？

A 平均时间复杂度上，选择排序的复杂度较低

B 平均时间复杂度上，冒泡排序的复杂度较低

C 其它选项皆不正确。

D 空间复杂度上，冒泡排序的复杂度较低

提交

单选题 1分

设置

设一组初始记录关键字序列(5，2，6，3，8)，利用冒泡排序进行升序排序，则第一趟冒泡排序的结果为以下何者？

A 2，5，6，3，8

B 2，3，6，5，8

C 2，5，3，6，8

D 2，3，5，6，8

提交

单选题 1分

设置

设一组初始记录关键字序列(5，2，6，3，8)，利用插入排序进行升序排序，则第二次插入排序的结果为以下何者？

A 2，3，5，6，8

B 5，2，3，6，8

C 2，5，6，3，8

D 2，5，3，6，8

提交

多选题 1分

设置

所谓排序算法的稳定性是指：排序前2个相等的数，其在序列的前后位置顺序和排序后它们两个的前后位置顺序相同。以下哪些排序算法是稳定的？

A 希尔排序

B 选择排序

C 插入排序

D 冒泡排序

提交

多选题 1分

设置

以下哪些排序方式，其最坏情况的时间复杂度O(N^2)的？

A 希尔排序

B 选择排序

C 插入排序

D 冒泡排序

提交

单选题 1分

设置

已知输入数据1，2，8，5，9，11，34，56，51，77，请分析数据，采用（　　）排序算法效率最低

A 简单插入排序

B 选择排序

C 冒泡排序

D 三者取中法作为枢纽的快速排序

提交

单选题 1分

设置

已知输入数据13，24，7，1，8，9，11，56，34，51，2，77，5，请采用快速归排序算法进行排序，其中枢纽采用3者取中法，2趟排序后的结果为（　　）

A （（2，5，1），7，（8，9）），11，（（13，34，24），51，（77，56））

B （（2，5，1），（7，8），9）），11，（（13，34，24），51，（77，56））

C （（1，2），5，（7，8，9，11）），13，（（24），34，（56，77，51））

D （（1，2），5，（13，8，9，7）），11，（（24），34，（51，77，56））

提交

单选题 1分

设置

将两个各有n个元素的有序表归并成一个有序表，其最少的比较次数是（ ）

A n-1

B n

C 2n

D n/2

提交

单选题 1分

设置

已知输入数据13，24，7，1，8，9，11，56，34，51，2，77，5，请采用2路归并递归排序算法进行排序，2趟排序后的结果为（　　）

A 13，7，24，1，8，9，11，56，34，51，2，5，77

B 13，24，1，7，8，9，11，34，56，51，2，77，5

C 1，7，13，24，8，9，11，34，51，56，2，5，77

D 7，13，24，1，8，9，11，34，51，56，2，5，77

提交

第8章 查找算法

知识要点

(1) 理解查找的概念；
(2) 掌握静态查找表的方法；
(3) 熟悉并能运用散列技术解决实际问题；
(4) 熟悉线性索引以及树形索引。

薄钧戈
2021年5月29日

8.1 查找的基本概念

- 查找又称为检索，确定一个已经给出的数据是否出现在某个数据元素集合中。
  - 一是数据如何组织——查找表
  - 二是在查找表上如何查找——查找方法。

8.1 查找的基本概念

- 有关查找表的相关概念：
  - （1）键是数据元素中的某个项或者组合项的值，用来标识一个数据元素。能够唯一确定一个数据元素的键，称为主键。而不能唯一确定一个数据元素的键，称为次键。
  - （2）查找表指由具有同一类型（属性）的数据元素组成的集合。
    - 静态查找表只允许进行查找，不能改变表中的数据元素。
    - 动态查找表不仅可以实现查找，还允许向表中插入或删除数据元素。
  - （3）查找是指在查找表中查找键值与待查元素值相等的过程。
    - 若找到，表示查找成功，返回该记录的信息或该记录在表中的位置；
    - 找不到，表示查找失败，返回相关的指示信息

8.1 查找的基本概念

- 查找方法的性能指标
  - 查找运算时间主要花费在关键字比较上，通常把查找过程中执行的关键字平均比较次数（也称为平均查找长度）作为衡量一个查找算法效率优劣的标准。
  - 对n个对象记录进行查找时，平均查找长度ASL(Average Search Length)为：

$$ASL=\sum_{i=0}^{n-1} P_i C_i$$

  - 其中，n是查找表中记录的个数。pi是查找第i个记录的概率，一般地，认为每个记录的查找概率相等，即pi=1/n（1≤i≤n），ci是找到第i个记录所需进行的比较次数。

8.1 查找的基本概念

- 平均查找长度分为
  - 成功情况下的平均查找长度
  - 不成功情况（失败）下的平均查找长度

8.2 静态查找表

- 静态查找表大部分情况下采用顺序存储结构，有时在概率不等的复杂情况下也可以采用链表结构。

算法8.1：定义记录的结构

```
1.  const int MAXSIZE=100;      //静态表的最大长度
2.  typedef int KeyType;        //关键字的类型
3.  typedef struct {            //记录(数据元素)的结构
4.  KeyType key;                //关键字域
5.  char ch;                    //其他属性域
6.  } datanode;
```

8.2 静态查找表

- **算法8.2：顺序表类的定义**

```
1.  class Sqtable{
2.  private:
3.  datanode r[MAXSIZE];        //数组，查找表
4.  int n;                      //数据元素个数，表长
5.  public:
6.  Sqtable();  //构造函数
7.  void create();
8.  void output();
9.  int sq_search(KeyType k);      //顺序查找
10. int binary_search(KeyType k);  //折半查找
11.  };
```

8.2.1 顺序表的查找

- 顺序查找又称线性查找，是最基本的查找方法之一。将待查找的元素存放于数组或链表中。
- 基本思想：
  - 按照元素存放位置顺序遍历所有元素，从而找到待查找的元素。
  - 从表中的第一个记录开始，逐个进行记录的关键字与给定值的比较，如果某个记录的关键字与给定值比较相等，则查找成功；
  - 反之，若直至最后一个记录，其关键字与给定值比较都不等，则表明表中没有所查记录，查找不成功。

8.2.1 顺序表的查找

- 已知表r，关键字数据K，表长n：
- 算法8.3：顺序查找算法

```
1.  int Sqtable::sq_search(KeyType K){
2.  r[n].key=K;        //设置监视哨
3.  int i=0;
4.  while(r[i].key!=K)  i++;
5.  if(i<n) return i;  //查找成功，该记录下标为i
6.  else return -1;    //查找失败
7.  }
```

8.2.1 顺序表的查找

顺序表的性能分析：

- 假设顺序表中有n个记录，每个记录r_i（i=0,1,…,n-1）被查找的概率为P_i，查找r_i所用比较次数为C_i，查找到第i个元素需要比较i+1次，因此C_i=i+1。
- 假设每个记录的查找机会都相等，即P_i=1/n，则在等概率情形下顺序查找的平均查找长度为：

$$ASL=\sum_{i=0}^{n-1} P_i C_i = \frac{1}{n}\sum_{i=0}^{n-1} i = \frac{1}{n} * \frac{n(n+1)}{2} = \frac{n+1}{2}$$

  - 查找成功时的平均比较次数约为表长的一半。

- 当查找不成功时需要和表中所有元素都比较一次，平均查找长度为n，故算法的时间复杂度为O(n)。

8.2.1 顺序表的查找-改进

- 对顺序表的查找有如下三种改进策略（自组织表）：
  - ① 每次成功查找将被查元素移动到查找表的表头；
  - ② 每次成功查找将被查元素向前移动一个位置；
  - ③ 给每个元素增加一个频率域，每次成功查找时，更新该元素的频率并且移动元素的位置，以保持查找表中的元素按频率从大到小存储。
- 根据“二八规律”，80%的查找都会在前20%部分成功。

单选题 1分

设置

静态查找表和动态查找表的区别是（ ）。

A 所包含的数据元素的类型不同

B 施加其上的操作不同

C 它们的逻辑结构相同

D 其他选项都不对

提交

单选题 1分

设置

顺序查找法适合于存储结构为（ ）的线性表。

A 哈希存储

B 索引存储

C 顺序存储或链式存储

D 压缩存储

提交

单选题 1分

设置

采用顺序查找方法查找长度为n的顺序表时，在等概率时成功查找的平均查找长度为（ ）。

A n/2

B (n-1)/2

C n

D (n+1)/2

提交

单选题 1分

设置

采用顺序查找方法查找长度为n的顺序表时，在等概率时不成功查找的平均查找长度为（ ）。

A n/2

B (n-1)/2

C n

D (n+1)/2

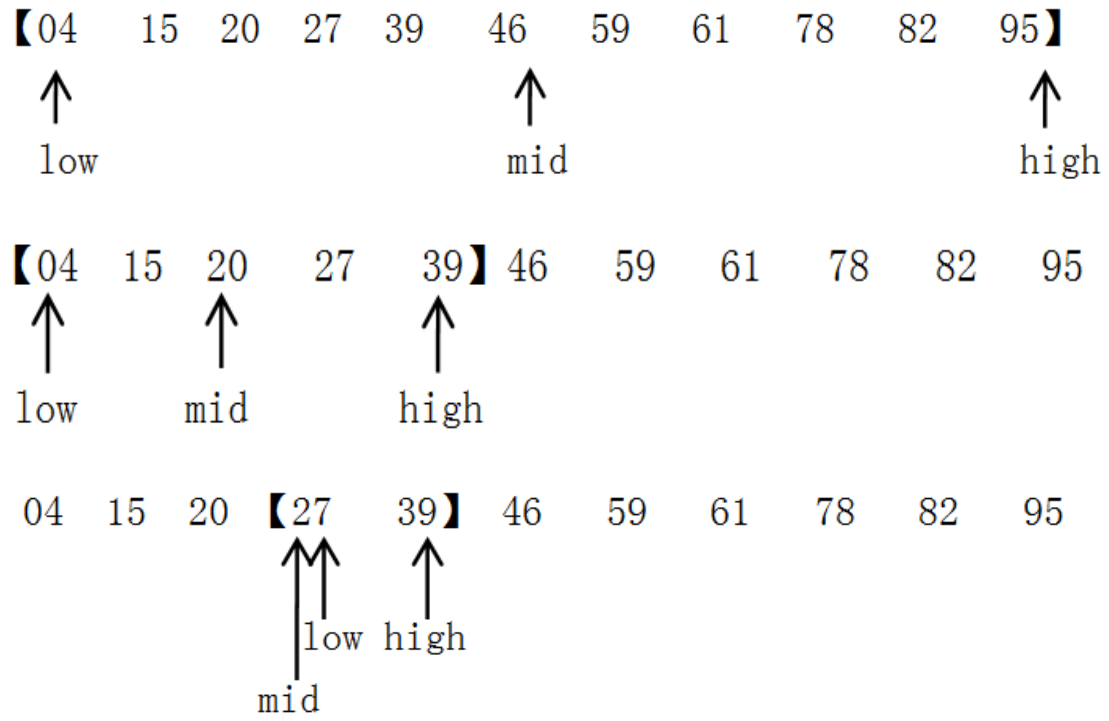
提交

8.2.2 折半查找

- 折半查找又称二分法查找：
  - 折半查找只适用于对有序、顺序表进行查找的情况。
- 算法步骤：
  - 给定一个待查找关键值K，在存储表rec[low..high]中查找。
  - 每次与mid=(low+high)/2的位置元素比较，根据比较的结果可分为以下三种情况：
    - （1）如果K与r[mid]的值相等，则r[mid]的值为查找的记录；
    - （2）如果K小于r[mid]的值，则在r[low,mid-1]中进行查找；
    - （3）如果K大于r[mid]的值，则在r[mid+1,high]中进行查找。

8.2.2 折半查找-例

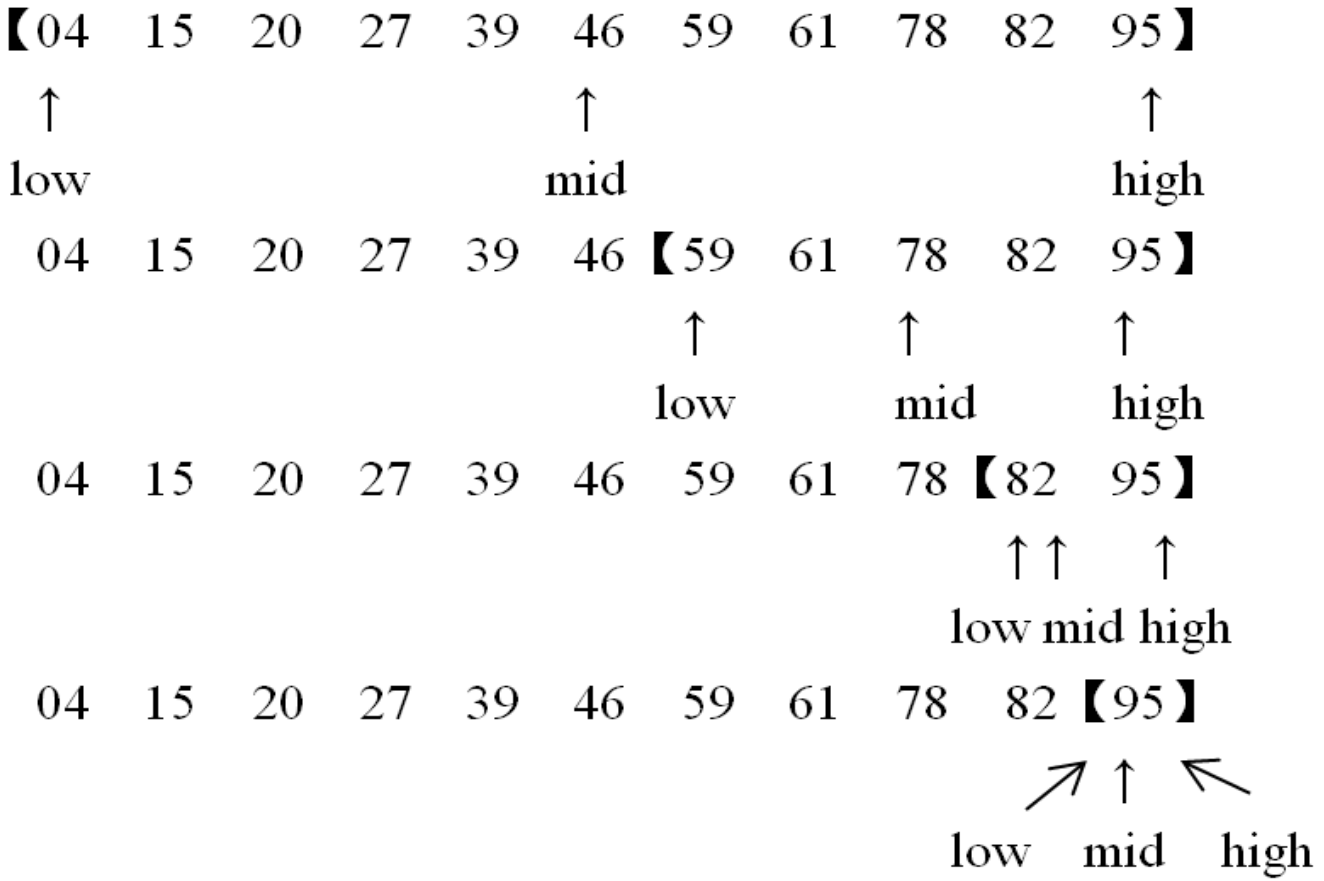
- 【例8.1】已知一个有11个数据元素的有序表：（04，15，20，27，39，46，59，61，78，82，95），以折半查找方法查找关键字为27和86的数据元素。
  - （1）如查找给定值key=27的元素



由于Key == rec[mid]
查找成功。

8.2.2 折半查找-例

- （2）如查找给定值key=86



Low、high、mid均指着95

表明查找表中没有与key相等的关键字，即**查找不成功**。

8.2.2 折半查找-代码

- 算法8.4：折半查找的递归算法

```
1.  int Sqtable:: binary_search(KeyType k, int low, int high){
2.  if  ( low > high ) return –1;
3.  int mid = (low + high) / 2;
4.  if  ( k == r[mid].key ) return mid;
5.  else if  ( k < r[mid].key ) return binary_search(k, low, mid-1);
6.       else return binary_search(k, mid+1, high);
7.  }
```

8.2.2 折半查找-代码

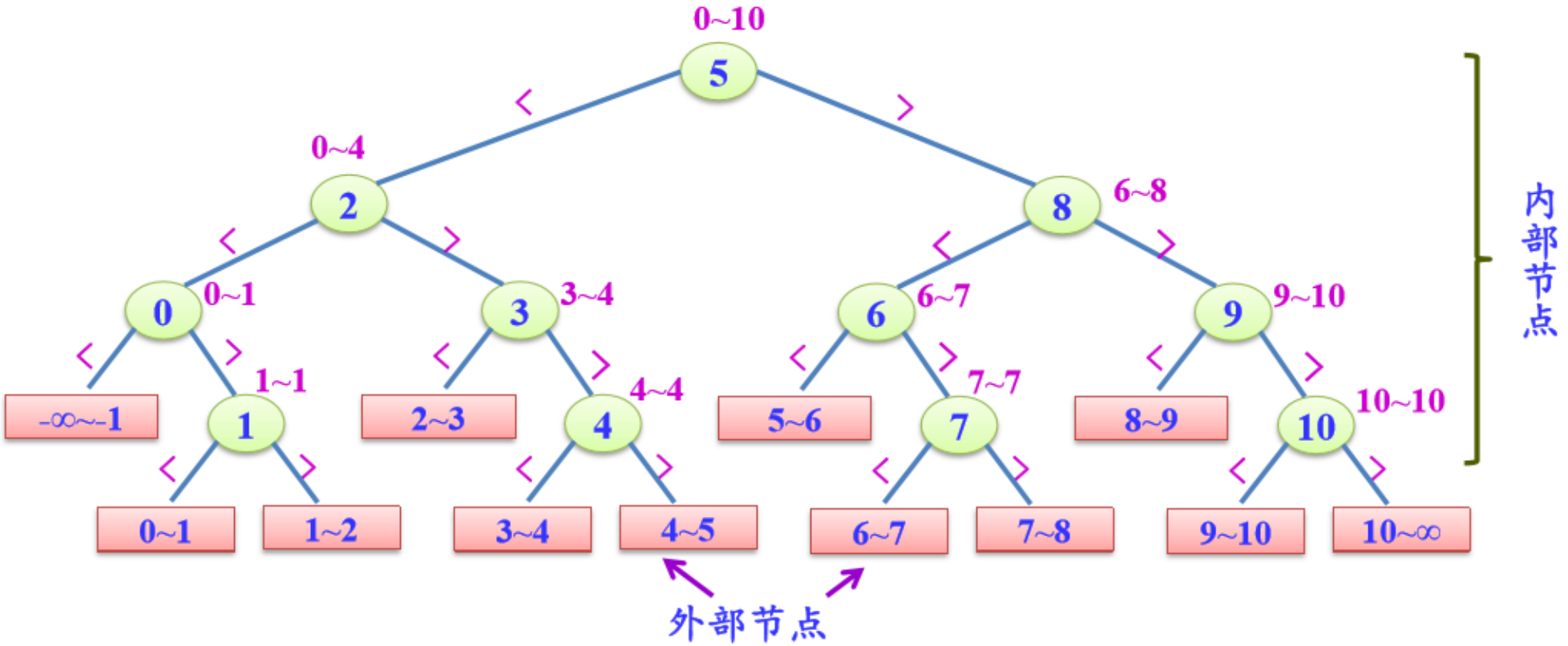
- 算法8.5：折半查找的非递归算法

```
1.  int Sqtable:: binary_search(KeyType  K){
2.  int mid, low, high;
3.  low=0; high=n-1;
4.  while( low <= high ){
5.        mid=(low+high)/2;
6.        if(K==r[mid].key) return mid;    //查找成功
7.        if(K< r[mid].key) high=mid-1;
8.        else low=mid+1;
9.  }
10.       return -1;  //查找失败
11. }
```

8.2.2 折半查找-复杂度分析

- 二分查找过程可用二叉树来描述：
  - 把当前查找区间的中间位置上的记录作为根；
  - 左子表和右子表中的记录分别作为根的左子树和右子树。
- 这样的二叉树称为判定树或比较树

8.2.2 折半查找-复杂度分析



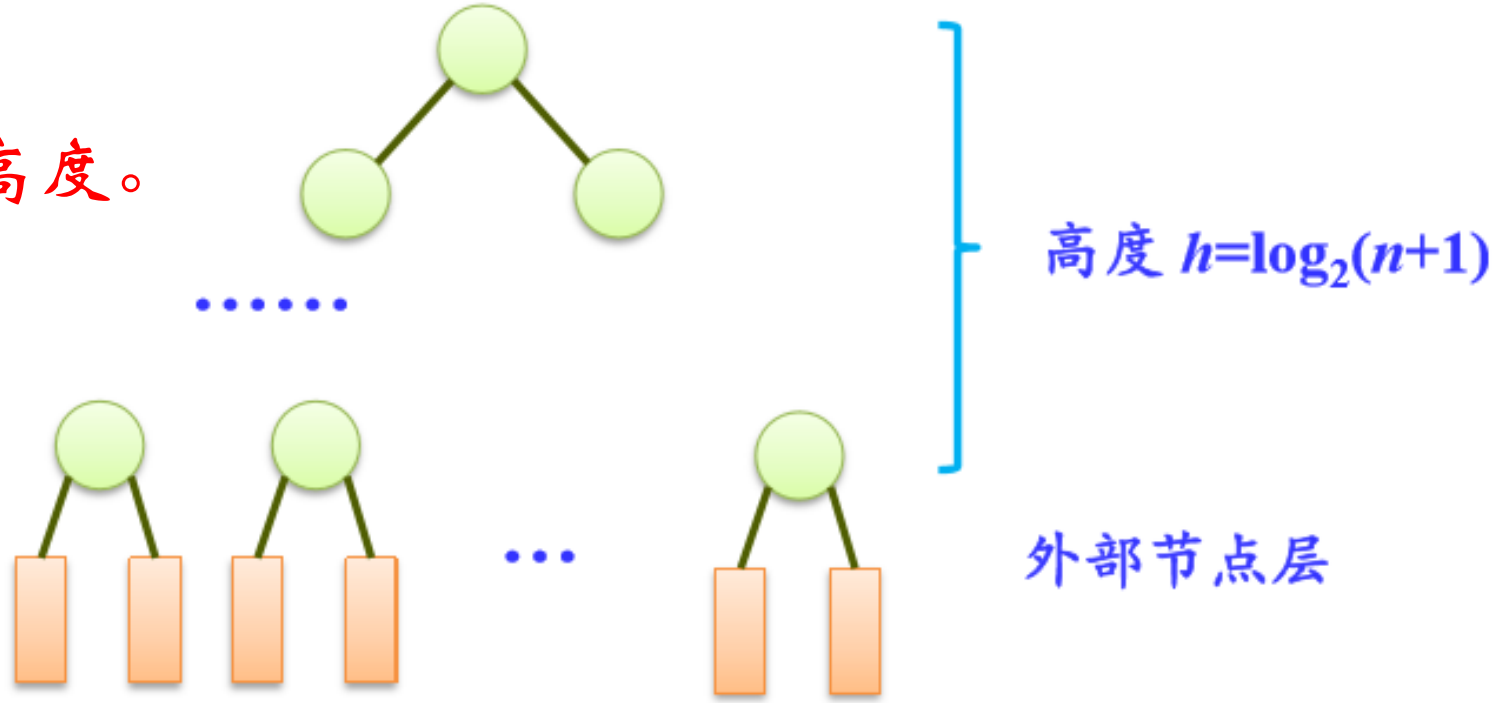
R[0..10]的二分查线的判定树（*n*=11）

- 外部节点即查找失败对应的节点，是虚拟的
- n个关键字：内部节点为n个，外部节点为n+1个

8.2.2 折半查找-复杂度分析

- 当n比较大时，将判定树看成内部节点的总数为$n=2^h-1$、高度为$h=\log_2(n+1)$的满二叉树（高度h不计外部节点）。
  - 树中第i层上的记录个数为2^{i-1}，查找该层上的每个记录需要进行i次比较

★注：折半查找中，
最大比较次数等于判定树的高度。



8.2.2 折半查找-复杂度分析

- 假定查找表中的每个记录的查找概率都相等（$P_i=\frac{1}{n}$），则查找成功时折半查找的平均查找长度为：

$$ASL_{bs}=\sum_{i=0}^{n-1}P_iC_i=\frac{1}{n}\sum_{j=1}^{h}j*2^{j-1}=\frac{n+1}{n}\log_2(n+1)-1$$

  - 即第j层最多有2^{j-1}个结点，每个结点比较次数为j。
  - 对任意的n，当n较大（n>50）时，可有近似结果$ASL_{bs}=\log_2(n+1)-1$

- *对于n个元素，二分查找，成功时最多的关键字比较次数为：$\lceil\log_2(n+1)\rceil$ 不成功时关键字比较次数为：$\lceil\log_2(n+1)\rceil$*
- 折半查找的平均时间复杂度为 $O(\log_2 n)$。

8.2.2 折半查找-小结

- 折半查找的优点与缺点：
  - 折半查找的效率高，但是首先需要将表按关键字大小排序，而排序本身又是一种非常费时的操作。即使采用高效的排序方法也要花费O(nlogn)的时间。
  - 折半查找只适用于顺序存储结构。为保持表的有序性，在顺序结构里插入和删除都必须移动大量的结点。因此，折半查找特别适用于那种一经建立就很少改动且经常需要进行查找的线性表。
  - 链式结构不适合实现二分查找算法。

单选题 1分

设置

已知在长度为n的线性表中采用顺序查找，查找成功最好的比较次数是（　　），查找成功的最坏比较次数是（　　），查找失败的比较次数是（　　）

A　0，n,n+1

B　1，n,n

C　1,n+1,n+1

D　1,n,n+1

提交

单选题 1分

设置

适合于折半查找的数据组织方式是（ ）。

A 以顺序表存储的有序线性表

B 以链表存储的线性表

C 以链表存储的有序线性表

D 以顺序表存储的线性表

提交

单选题 1分

设置

采用折半查找方法查找长度为n的线性表，当n很大时，在等概率时不成功查找的平均查找长度为（ ）。

A O(n)

B O(nlogn)

C O(log(n))

D O(n^2)

提交

单选题 1分

设置

设有100个元素的有序表，采用折半查找方法，在等概率时成功时最大的比较次数是（ ）。

A 10

B 7

C 6

D 25

提交

单选题 1分

设置

已知一个长度为16的顺序表，其元素按关键字有序排序，若采用折半查找法查找一个存在的元素，则比较的次数最多是（ ）。

A 4

B 7

C 6

D 5

提交

单选题 1分

设置

一个递增有序表为R[0..11]，采用折半查找方法进行查找，在一次不成功查找中，以下（ ）是不可能的记录比较序列。

A R[5]、R[8]、R[6]、R[7]

B R[5]、R[8]、R[6]

C R[5]、R[2]、R[3]

D R[5]、R[8]、R[10]

提交