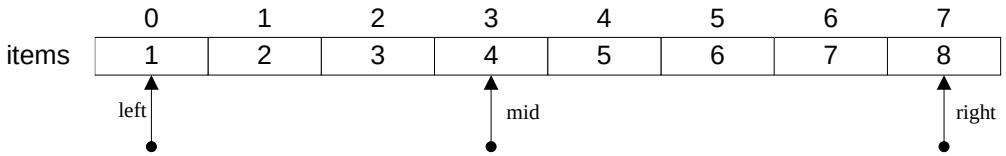
习题 10 部分参考答案

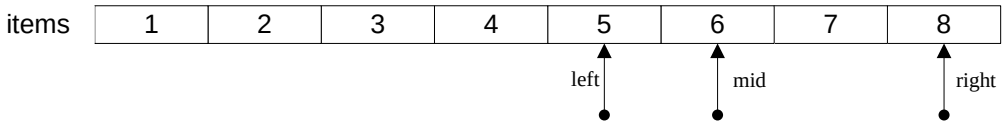
10.1 试分别画出在有序表{1, 2, 3, 4, 5, 6, 7, 8}中查找 6 和 10 的折半查找过程。

解答：

查找 6 的折半查找过程

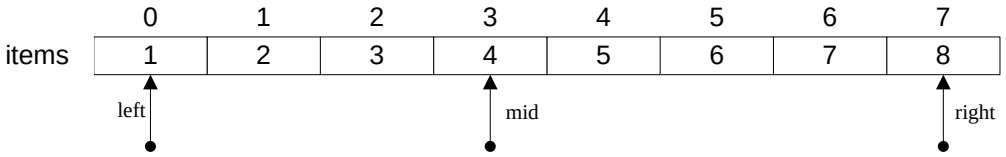


(a) k=6，k>items[mid]，left=mid+1

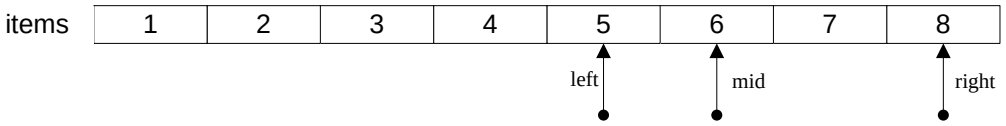


(b) k==items[mid]，查找成功，返回mid

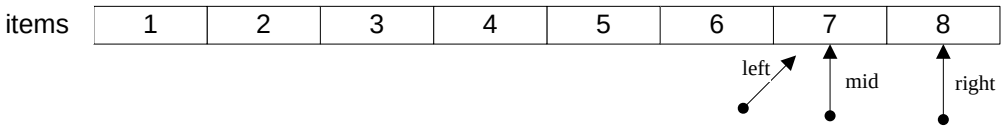
查找 10 的折半查找过程



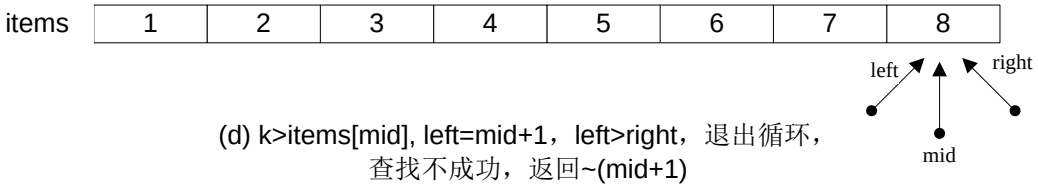
(a) k=10，k>items[mid]，left=mid+1



(b) k>items[mid]，left=mid+1



(c) k>items[mid], left=mid+1



(d) k>items[mid], left=mid+1，left>right，退出循环，
查找不成功，返回~(mid+1)

10.2 试分别写出对整型有序表数据进行折半查找的非递归与递归算法实现。

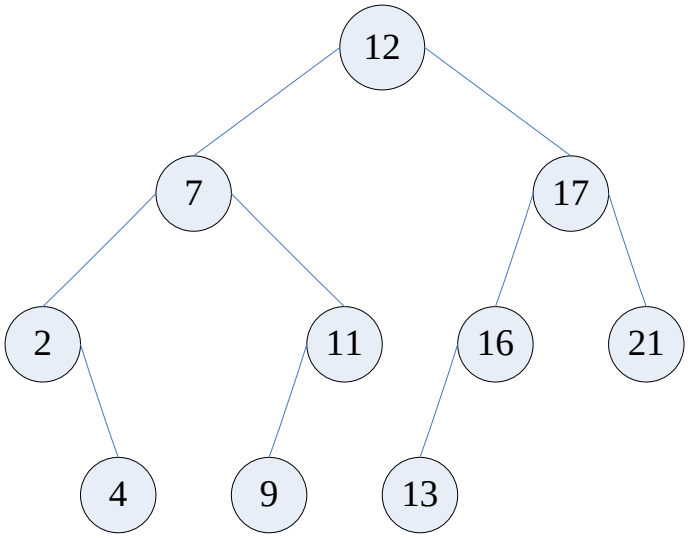
解答：

```
// 查找k值在线性表中的位置
// 查找成功时返回k值首次出现位置，否则返回应插入位置的位补码
public int BinarySearch(T k, int si, int length) {
    int mid = 0, left = si;
    int right = left + length - 1;
    while (left <= right) {
        mid = (left + right) / 2;
        if (k.CompareTo(items[mid]) == 0) {
            return mid;
        }
        else if (k.CompareTo(items[mid]) < 0)
            right = mid - 1;
        else
            left = mid + 1;
    }
    if (k.CompareTo(items[mid]) > 0)
        mid++;
    return ~mid;
}

public int BinarySearchR(T k, int left, int right) {
    int mid = (left + right) / 2;
    if (k.CompareTo(items[mid]) == 0)
        return mid;
    else if (k.CompareTo(items[mid]) < 0) {
        right = mid - 1;
        if (left > right)
            return ~mid;
        else
            return BinarySearchR(k, left, right);
    }
    else {
        left = mid + 1;
        if (left > right)
            return ~(mid+1);
        else
            return BinarySearchR(k, left, right);
    }
}
```

10.3 在一棵空的二叉查找树中依次插入关键字序列{12，7，17，11，16，2，13，9，21，4}，请画出所得到的二叉查找树。

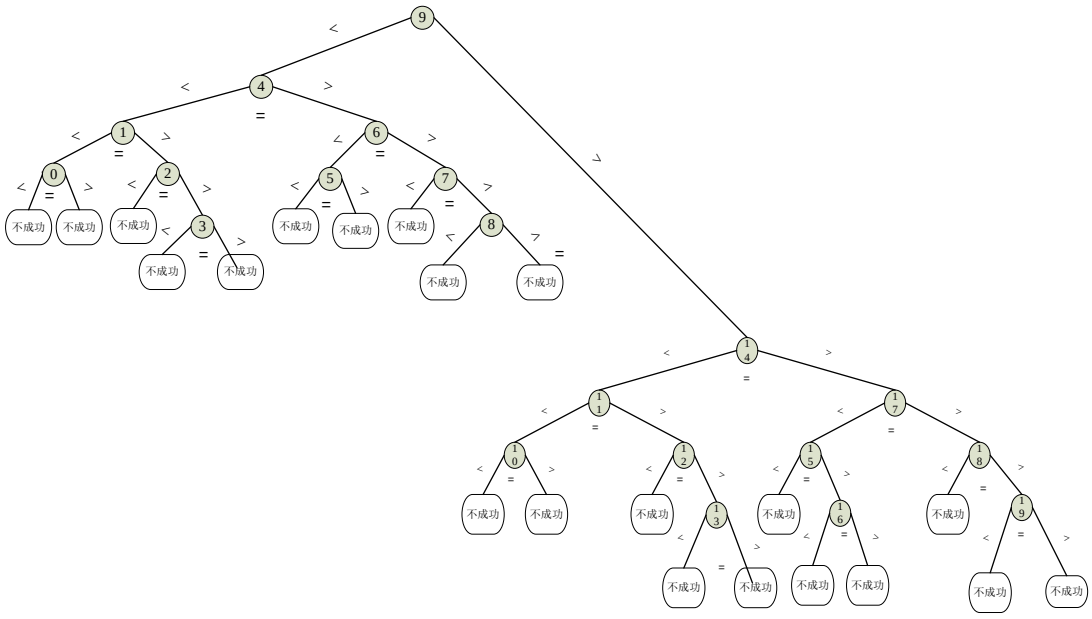
解答：



10.4 假设在有 20 个元素的有序数组 a 上进行折半查找，则比较一次查找成功的结点数为 1；比较两次查找成功的结点数为 2；比较四次查找成功的结点数为 8；在等概率的情况下查找成功的平均查找长度为 74/20。设有 100 个结点，用折半查找时，最大比较次数是 7。设有 22 个结点，当查找失败时，至少需要比较 5 次。

解答：

画判定树，可以计算各元素的比较次数：



全部元素的查找次数为=（1×1+2×2+4×3+8×4+5×5）=74；

ASL 成功=74/20=3.7

$Log_2(100) = 6.64$

$Log_2(22) = 4.46$

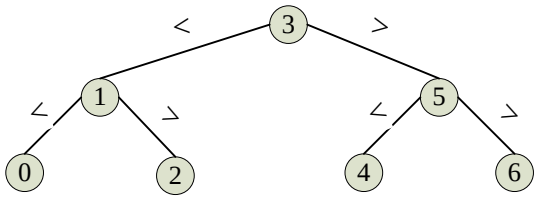
10.5 假设在有序表{2, 8, 13, 16, 27, 36, 78}中进行二分查找，请画出判定树，并分别给出查找 16 和 40 时 BinarySearch 方法的返回值。

解答：

查找 16 的返回值：3

查找 40 的返回值：-7，（即 ~6）（取负减 1）

二分查找判定树：



{2, 8, 13, 16, 27, 36, 78}
0 1 2 3 4 5 6

10.6 哈希查找的设计思想是什么？哈希技术中的关键问题有哪些？

解答：

哈希查找技术的设计思想是，由数据元素的关键字决定数据元素的存储位置，即根据数据元素的关键字值 k 计算出相应的哈希函数值 hash(k)，这个值决定该数据元素的存储位置。哈希造表和哈希查找的过程都是基于这个思想实现的。

哈希查找技术的关键问题在于以下两点：

- **避免冲突**（collision avoidance）：设计一个好的哈希函数，尽可能减少冲突。不同的关键字 k_1 和 k_2，对应相同的哈希函数值，这种现象称为**冲突**（collision）
- **解决冲突**（collision resolution）：因为冲突是不可避免的，发生冲突时，使用一种解决冲突的有效方法（collision resolution strategy）。

8.7 设哈希表的地址范围为 0～17，哈希函数为：H(k) ＝ k %16。用线性探测法处理冲突，说明对输入关键字序列{10，24，32，17，31，30，46，47，40，63，49}的哈希造表及查找过程。

解答：

（1）哈希造表过程如下：

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
item	32	17	**63**	**49**					24	**40**	10				30	31	**46**	**47**

（2）查找 63 和 60 的过程：

（a）查找 63：H(63)=63%16=15，首先要与 15 号单元内容比较，63!=31，然后顺移，与 46, 47, 32, 17, 63 比较，一共比较了 6 次，查找成功。

（b）查找 60，H(60)=60%16=12，首先要与 12 号单元内容比较，但因为 12 号单元为空（应当有空标记），所以只比较这一次即可，查找不成功。

查找过程：

查找关键字为 k 的数据元素时，首先计算哈希函数 i=hash(k) = k % 16，如果位置 i 为空，查找不成功；如果位置 i 的值等于 *k*，则查找成功；否则继续向后依次查找直至找到或到一个空位置仍没有找到，前者结果为查找成功，后者为查找不成功。

（3）假定每个关键字的查找概率相等，求查找成功时的平均查找长度。

（a）对于 17, 24, 10, 30, 31, 32 元素，各比较 1 次，共 6 次；

（b）对于 63：比较 6 次；

（c）对于 46，47 和 49：3 次；

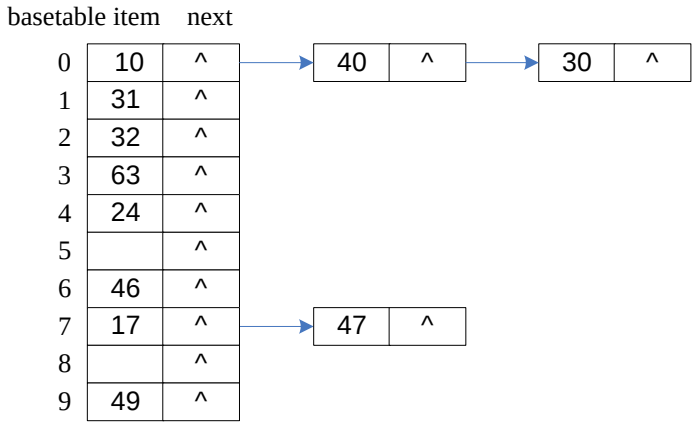
（d）对于 40：比较 2 次。

所以 $ASL_{成功}$ =（1×6 + 6×1＋2×1＋3×3）/11＝23/11

10.8 设哈希基表的地址范围为 0～9，哈希函数为：H(*k*) ＝ *k* %10。用散列链法处理冲突，说明对输入关键字序列{10，24，32，17，31，30，46，47，40，63，49}的哈希造表及查找过程。

解答：

（1）哈希造表过程如下：



（2）查找 63 和 60 的过程：

（a）查找 63：H(63)=63%10=3，首先要与 3 号单元内容比较，==，比较了 1 次，查找成功。

（b）查找 60，H(60)=60%10=0，首先要与 0 号单元内容比较，60!=10，然后在链表中查找，依次与 40, 30 比较，一共比较了 3 次，查找不成功。

散列链法的哈希查找算法可分为两步：

1） 设给定值为 *k*，计算哈希函数值*i* = hash(*k*)，若哈希基表中basetable[i]元素为空，则查找不成功。

2） 如果basetable[i]元素不为空，将它的关键字与*k*进行比较，相等则查找成功；不相等则说明产生冲突，在由basetable[i].Next 指向的哈希链表中按顺序查找。查找该链表进一步确定查找是否成功。