第15课 对分查找

对分查找的概念

对分查找又称二分查找，是一种高效的查找方法。对分查找的前提是，被查找的数据序列是有序的(升序或降序)。

对分查找的基本思想

对分查找的基本思想是在有序的数列中，首先将要查找的数据与有序数列内处于中间位置的数据进行比较，如果两者相等，则查找成功；否则就根据数据的有序性，再确定该数据的范围应该在数列的前半部分还是后半部分；在新确定的缩少范围内，继续按上述方法进行查找，直到找到要查找的数据，即查找成功，如果要查找的数据不存在，即查找不成功。

对分查找的处理过程

若key为查找键，数组a存放n个已按升序排序的元素。在使用对分查找时，把查找范围[i，j]的中间位置上的数据a[m]与查找键key进行比较，结果必然是如下三种情况之一：

(1)若key<a[m]，查找键小于中点a[m]处的数据。由a中的数据的递增性，可以确定：在(m，j)内不可能存在值为key的数据，必须在新的范围(i，m－1)中继续查找；

(2)key＝a[m]，找到了需要的数据；

(3)key>a[m]，由与(1)相同的理由，必须在新的范围(m＋1，j)中继续查找。

这样，除了出现情况(2)，在通过一次比较后，新的查找范围将不超过上次查找范围的一半。

中间位置数据a[m]的下标m的计算方法：m＝ (i＋j)//2或m＝int((i＋j)/2)

对分查找的程序实现

(1)由于比较次数难以确定，所以用while语句来实现循环；

(2)在while循环体中用If语句来判断查找是否成功；

(3)若查找成功则输出查找结果，并结束循环(break)；

(4)若查找不成功，则判断查找键在数组的左半区间还是右半区间，从而缩小范围。

对分查找的程序实现

我们假设有一个列表lst = [12, 17, 23, 25, 26, 35, 47, 68, 76, 88, 96]
我们要查找元素key=25，则其对分查找的程序如下：

青少年软件编程等级考试Python标准公益课（3级）

```
lst = [12, 17, 23, 25, 26, 35, 47, 68, 76, 88, 96]
key = 25
n = len(lst)
i, j = 0, n - 1
b = -1
while i < j:
    m = (i + j) // 2
    if key == lst[m]:
        b = m          #找到了我们要找的数，赋值给b
        break          #找到key，退出循环
    elif key > lst[m]:
        i = m + 1
    else:
        j= m - 1
if b == -1:            #-1代表元素为查找到
    print("要查找的元素[" + str(key) + "]不在列表lst中。")
else:
    print("要查找的元素[" + str(key) + "]的索引是： " + str(b))
```

对分查找的查找次数的估算

对元素规模为n的列表进行对分查找时，无论是否找到，至多进行$\log_2 n+1$($\log_2 n+1$表示大于或等于 $\log_2 n$的最小整数)次查找就能得到结果，而使用顺序查找算法，在最坏的情况下(查找键在最后一个或没找到)，需要进行n次查找，最好的情况是一次查找(查找键在第一个)，平均查找$\frac{n+1}{2}$数是

小试牛刀

1．下列有关查找的说法，正确的是(　　)

A．顺序查找时，被查找的数据必须有序

B．对分查找时，被查找的数据不一定有序

C．顺序查找总能找到要查找的关键字

D．一般情况下，对分查找的效率较高

小试牛刀

2．某列表有7个元素，依次为19、28、30、35、39、42、48。若采用对分查找法在该列表中查找元素48，需要查找的次数是(　　)

A．1　B．2

C．3　D．4