二分查找

一.什么是二分查找?

二分查找又称折半查找，优点是比较次数少，查找速度快，平均性能好；其缺点是要求待查表为有序表，且插入删除困难.

例：

　　如果规定某一科目成绩分数范围：0 – 100，现在小明知道自己的成绩，他让你猜他的成绩，如果猜的高了或者低了都会告诉你；用最少的次数猜出他的成绩，你会如何设定方案?(排除运气成分和你对小明平时成绩的了解程度)

1.笨方法：从0开始猜，一直猜到100分；考虑这样来猜的最少次数:1(运气最好)，100(运气最差).

2.好方法：分数均分. 在我们根本不知道对方水平的条件下，我们每一次的猜测都想尽量将不需要猜的部分去除掉，所以我们考虑将分数均分，将分数区间一分为2，我们第一次猜的分数将会是50，当回答是低了的时候，我们将其分数区域从0-100确定到51-100；当回答高了的时候，我们将分数区域确定到0-49，这样一下子就减少了多余的50次猜想. 那么我们假设当猜完50分之后答案是低了，那么我们需要在51-100分的区间内继续猜小明的分数，同理我们继续折半，第二次我们将猜75分，当回答是低了的时候，我们将其分数区域从51-100确定到76-100；当回答高了的时候，我们将分数区域确定到51-74.这样一下子就减少了多余的猜想. 就此继续下去，直到回复是正确为止，这样考虑显然是最优的.

二.算法的复杂度

最多将会操作7次，其实因为每一次我们都抛掉当前确定的区间的一半的区间作为不可能解部分，那么相当于求最多操作次数，就是在区间内最多将有多少个一半可以抛去，那么就是将100一直除以2，直到不能除为止. 那么这个运算过程，其实就是相当于求了一个log2 (100) = 7.

常数阶O(1),对数阶O( ),线性阶O(n),
线性对数阶O(nlog2n),平方阶O(n^2)，立方阶O(n^3),...,
k次方阶O(n^k),指数阶O(2^n)。随着问题规模n的不断增大，上述时间复杂度不断增大，算法的执行效率越低.

三.查找过程

首先假设表中元素是按升序排列，将表中间位置记录的关键字与查找关键字比较，如果两者相等，则查找成功；否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于查找关键字，则进一步查找前一子表，否则进一步查找后一子表. 重复以上过程，直到找到满足条件的记录，使查找成功或直到子表不存在为止，此时查找不成功.

算法要求

1.必须采用顺序存储结构.

2.必须按关键字大小有序排列.

In [1]:
```python
#示例代码:

def f1(lists, val):
    if lists:
        listLen = len(lists)
        middle = int(listLen / 2)
        ltList = lists[0:middle]
        gtList = lists[middle + 1:]
        #print(ltList, gtList)
        if val  == lists[middle]:
            return True
        elif val < lists[middle]:  #向左找
            se = f1(ltList, val)
            return se
        elif val > lists[middle]:  #向右找
            se = f1(gtList, val)
            return se
    else:
        return False

if __name__ == '__main__':
    l = list(range(1, 101))
    x = f1(l, 40)
    print(x)
```

```
True
```