实验一：递归与分治策略

学号	202018526	姓名	高树林	成绩	
友情提示	1. 算法描述及代码实现与网络或他人雷同者，均按 0 分计算； 2. 要求算法描述明确、代码清晰、格式美观； 3. 纸质版与电子版同时提交（电子版命名格式：***完整学号-姓名-实验 X-实验名称***，其中$X \in \{1,2,3,4\}$,不得省略“-”，如：2088166-乔峰-实验 1-分治与递归策略）； 4. 电子版中代码需格式化处理，方便查看（http://www.codeinword.com/）。				
实验目的	1. 掌握递归与分治策略的基本思想。 2. 通过设计求解给定问题的递归算法和分治算法学会使用递归和分治法解决问题的一般技巧。				
实验内容	1. 二分搜索问题：设$a[0:n-1]$是已排好序的数组。试改写二分搜索算法，使得当搜索元素x不在数组a中时，返回小于x的最大元素的位置i和大于x的最小元素的位置j；当搜索元素x在数组a中时，返回x在数组中的位置，此时i和j相同。 2. 假币识别问题：一个袋子里有n个硬币，其中一枚是假币，假币和真币外观一模一样，仅凭肉眼无法区分，但是已知假币比真币轻一些。试设计识别假币的分治算法。				
算法描述	1. 二分搜索问题的解题思路或算法思想： 分治思想首先要考虑特殊情况（递归基），将特殊情况考虑完后便可以将其转化为规模为原来一半规模的子问题。子问题经过递归调用最终到达递归基，就能自下而上的返回所有结果。在本题中，首先考虑要查找的数是否在给定序列中，由于序列已经定序，因此只需要判断该数是否大于右边界或者是否小于左边界，若是则返回右边界或者左边界的下标值，否则待查找数就在左右边界之间，此时又分为两种情况，一是待查找数在序列当中，此时经过层层迭代，二分的最终结果会指向这个数，返回下标 mid 即可，若待查找数不在序列中，最终的结果会是 first 和 end 为相邻关系，此时查找不到 first 会变为 end，end 变为 first，此时已经是找不到了，此时的待查找数必定小于第 mid 个元素（下标从 0 开始），必定大于第（mid-1）个元素（下标从 0 开始），因此此题可解。在整个操作中，mid 的值在不断改变，包括使用递归时，mid 的值一直都在改变，因此必须将 mid 设为全局变量，否则程序会因为没有定义 mid 而报错。				
	2. 假币识别问题的解题思路或算法思想： 相比于第一个实验，第二个显得简单许多，因为在 python 中会有一个关键字 in，来判断元素在不在序列中。本题中将假币标记为 0，真币标记为 1，采用二分迭代，递归基为 mid 下标表示的元素为 0，此时就返回 mid 值。与第一题相比，本题一定会有一个假币在序列中，因此不需判断不在范围之内的问题。先判断下标为（first+end）//2 的是否是假币，再判断假币是在左半部分还是在右半部分，依次迭代下去，最终此题可解。				
程序	1. 二分搜索 代码：				

及运行结果（附截图）

```python
def FindNum(lst, num, first, end):
    global mid
    if lst[0] > num:  # 考虑查找的数在左边界外侧的情况
        return "大于该数的最小元素位置为 0"
    if lst[-1] < num:  # 考虑查找的数在右边界外侧的情况
        return "小于该数的最大元素位置为%d" % (len(lst) - 1)
    if first > end:  # 考虑未找到，但是待查找数在中间情况
        return "大于该数的最小元素位置为%d" % mid + "小于该数的最大元素位置为%d" % (mid - 1)
    mid = (first + end) // 2
    if lst[mid] == num:  # 找到的情况
        return "已找到，该数的下标为%d" % mid
    elif lst[mid] < num:  # 查找的元素在右侧
        first = mid + 1  # 中间元素设为新的起始位置，从而达到二分的效果
        return FindNum(lst, num, first, end)
    elif lst[mid] > num:  # 查找的元素在左侧
        end = mid - 1  # 中间元素设为新的起始位置，从而达到二分的效果
        return FindNum(lst, num, first, end)


if __name__ == '__main__':
    a = [1, 2, 3, 7, 8, 9]
    num = int(input("输入你要查找的数："))
    print(FindNum(a, num, 0, len(a) - 1))
```

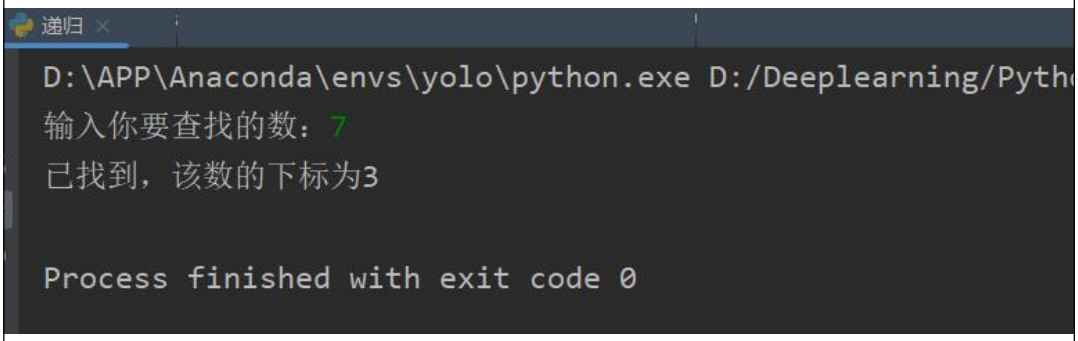
截图：

```
if __name__ == '__main__':
    a = [1, 2, 3, 7, 8, 9]
    num = int(input("输入你要查找的数："))
    print(FindNum(a, num, 0, len(a) - 1))
```

FindNum() › if lst[-1] < num

递归 ×

```
D:\APP\Anaconda\envs\yolo\python.exe D:/Deeplearning/PythonProjects/MyLibrarie
输入你要查找的数：5
该数不在序列中，大于该数的最小元素位置为3小于该数的最大元素位置为2
```



2. 假币识别

代码：

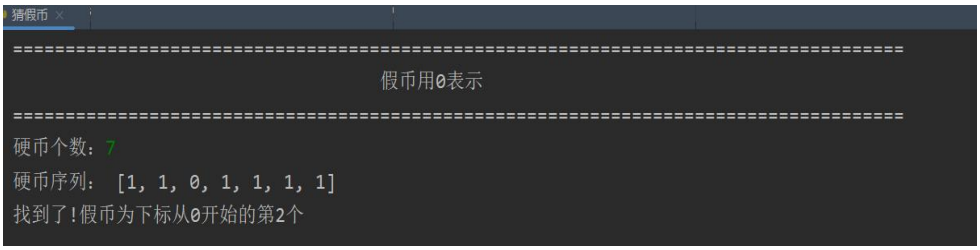
```python
import random


def find_fake_coins(coins, first, end):
    mid = (first + end) // 2
    if 0 == coins[mid]:
        return "找到了!假币为下标从 0 开始的第%d 个" % mid
    if 0 in coins[mid + 1:]:
        first = mid + 1
        mid = (first + end) // 2
        return find_fake_coins(coins, first, end)
    if 0 in coins[first:mid]:
        end = mid - 1
        mid = mid = (first + end) // 2
        return find_fake_coins(coins, first, end)
    return "No Fake Coins!"


if __name__ == "__main__":
```

```
    print("================================================================================")
    print('                                                假币用 0 表示')
    print("================================================================================")
    num = int(input("硬币个数："))
    coins = [1] * num  # 1 表示真币
    # a = random.randint(0, len(coins) - 1)
    # print("下标从 0 开始的第%d 个为假币" % a)
    # coins[a] = 0  # 0 表示假币
    coins[random.randint(0, len(coins) - 1)] = 0
    print("硬币序列：",coins)
    print(find_fake_coins(coins, 0, len(coins) - 1))
```

截图：



总结

实验心得：

通过这次实验，我明白了递归分治的思想就是分而治之，即：将一个能直接解决的大问题分割成一系列规模较小的相同问题（子问题），以便逐个击破。上述两个就是通过二分来使问题规模每次降低 50%，降低了原来的规模，从而降低时间复杂度。这让我想到了在小时候玩的猜数游戏，给出一个数，每次猜的时候对方会给反馈：大了还是小了。我总会先猜到一个很大和一个很小的数，得到反馈之后就猜中间的数，依次下去。当时并不知道这里面的原因，但是总能很快猜到，学习了分治之后我就明白了其中的道理，所谓盲猜的最大和最小值，也就是确定一个规模，猜中间值就是找中位数的过程。

这次实验给我的启示是：在学习和生活中也会遇到难题，但是某些难题可以分割成一系列的子问题，这些子问题能按照相同的套路来解决，如此递归下去，灾难的问题也会解决，但可贵的是需要找到递归基，也即先解决这个难题中最基本的问题，由此入门。