

华东师范大学计算机科学技术系作业

	华东师范大学计算机科学技术系作业	
课程名称：编程导论Python	年级：2018级	作业成绩：
指导教师：杨燕	姓名：吴子靖	提交作业日期：2018年12月26日
专业：计算机系	学号：10185102141	作业编号： 11

一、要如何改写#<程序：递归实现二分查找>，使得若 k 大于 $L[\text{len}(L)//2]$ ，调用 $\text{BinSearch}(L[\text{len}(L)//2 + 1 :], k)$ 。注意,return 的索引 $\text{len}(L)//2 + index$ 要改动。另外，可否去掉 if $\text{len}(L)==1$ 的检查。
(15分)

可以去掉 if $\text{len}(L) == 1$ 的检查，因为当改写程序之后, $k ==, <, > \text{len}(L//2)$ 已经具备了对应不同L长度的情况

```
In [13]: def BinSearch(L, k):
        if L == []:
            return False
        if k == L[len(L)//2]:
            return True, len(L)//2
        if k < L[len(L)//2]:
            return BinSearch(L[:len(L)//2], k)
        else:
            k += len(L)//2
            return BinSearch(L[len(L)//2+1:], k)
L = list(map(int, input("请输入一串有序数列:").split()))
k = int(input("请输入上述数列中的一个元素"))
print("通过内置index函数查找的索引为:", L.index(k))
print("通过BinSearch函数查找的索引为:", BinSearch(L, k))
```

请输入一串有序数列:12 43 78 134 568 1234

请输入上述数列中的一个元素134

通过内置index函数查找的索引为: 3

通过BinSearch函数查找的索引为: (True, 3)

二、非递归实现二分法求解问题中的程序#<程序：二分法递归查找插入位置>。
(10分)

```
In [29]: L = list(map(int, input("请输入一串有序数列:").split()))
k = int(input("请输入一个整数:"))
L1 = L[:]; index = 0
while len(L1) > 1:
    if k < L1[len(L1)//2]:
        L1 = L1[:len(L1)//2]
    elif k > L1[len(L1)//2]:
        index += len(L1)//2
        L1 = L1[len(L1)//2:]
    else:
        index += len(L1)//2
        break
if len(L1) == 1:
    if k > L1[0]:
        index += 1
L2 = L[:index] + [k] + L[index:]
print("原列表为:", L)
print("插入位置索引为:", index)
print("插入k后的列表是:", L2)
```

请输入一串有序数列:1 5 8 45 78 97 13456 345678

请输入一个整数:1024

原列表为: [1, 5, 8, 45, 78, 97, 13456, 345678]

插入位置索引为: 6

插入k后的列表是: [1, 5, 8, 45, 78, 97, 1024, 13456, 345678]

三、<程序：二分法递归查找插入位置>中，假如去掉“if k<L[0]: return index_min”，程序是否还正确？
(10分)

正确，因为去掉“if k<L[0]: return index_min”，无非是增加了递归次数，当L的长度被分片到1时，L[mid]也就等于L[0]了，if k<L[0]: return index_min还是改善程序速度，并不影响程序对错

```
In [27]: def binary_r0_Insert(L, k):
        def r0_Insert(L, index_min):
            if len(L) == 0:
                return index_min
            if k > L[len(L)-1]:
                return index_min + len(L)
            mid = len(L)//2
            if L[mid] > k:
                x = r0_Insert(L[:mid], index_min)
            elif L[mid] < k:
                x = r0_Insert(L[mid+1:], index_min+mid+1)
            else:
                x = index_min + mid + 1
            return x
        return r0_Insert(L, 0)
L = [1, 2, 4, 5]
k = 9
print(binary_r0_Insert(L, k))
```

4

四、修改求解算术平方根问题中的<程序：算数平方根运算-二分法>的代码，使其可以求解一个实数c的k次方跟。（函数有c和k两个参数）（10分）

```
In [39]: def n_root(c, k):  
        i = 0; m_max = c; m_min = 0  
        g = (m_min + m_max)/2  
        while (abs(pow(g, k)-c) > 0.000000000001):  
            if (pow(g, k) < c):  
                m_min = g  
            else:  
                m_max = g  
            g = (m_min + m_max)/2  
        return g  
c = eval(input()); k = eval(input())  
print("c的k次方根为:", n_root(c, k))
```

1000

3

c的k次方根为: 10.000000000000009

五、编写代码，求解 $lgx = a$ ，精度为 0.00000000001。函数的形式为：def log(a):... return
(15分)

```

In [12]: def log(a):#10^a=x
            if a == 1:
                return 0
            elif a > 1:
                b = len(str(int(a)))
                Min = 0
                Max = b
            else:
                b = 0
                c = a
                while c < 1:
                    c = c * 10
                    b -= 1
                Max = 0
                Min = b
            g = (Min + Max)/2
            while abs(pow(10,g)-a) > 0.000000000001:
                if pow(10,g) > a:
                    Max = g
                    g = (Min + Max)/2
                else:
                    Min = g
                    g = (Min + Max)/2
            return g
x = eval(input("请输入一个大于0的数:"))
print("lg(%d)=%f"%(x), log(x))

```

请输入一个大于0的数:20

lg(20)= 1.3010299956640665

六、假如有n个钱币，其中有一个钱币是假的，已知假的钱币比较轻，你只有一个天平，如何用非递归和递归思维来找到这个假币？请写出Python程序。请先写出一个起始函数来设定一个列表，列表有100个1，代表100个钱币，再随机设定一索引值，将其改为小于1的任意数，代表是个假钱币。你的程序要输出这个假钱币的索引。

(20分)


```
In [9]: # 生成所需的列表
import random
index = random.randint(0, 99)
weight = random.uniform(0, 1)
L = [1]*100
L[index] = weight
print("所生成的列表是:", L) #展示生成的列表和索引
print("设置的假币值为:", weight)
print("事先设置的假币的位置是:", index)

# 非递归思维的程序
def find_fake_one(L):
    L1 = L[:]
    for i in range(0, len(L1)):
        if L1[i] != L1[i+1]:
            if L1[i] < L1[i+1]:
                return i
            else:
                return i+1
    return ("假币不存在!")

#递归思维解决: 二分法
i = 0
def find_fake_two(a, L):
    x = len(L)
    if x == 1:
        return a
    if x % 2 == 1:
        x = x - 1
        y = 1
    else:
        y = 0
    if sum(L[:x//2]) < sum(L[x//2:x]):
        return find_fake_two(a, L[:x//2])
    elif sum(L[:x//2]) > sum(L[x//2:x]):
        return find_fake_two(a+x//2, L[x//2:x])
    else:
        if y == 0:
```


所生成的列表是：[1, 0.8238039854007045, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

设置的假币值为：0.8238039854007045

事先设置的假币的位置是：56

用非递归所找到的假币索引为：56

用二分法递归找到的假币索引为 56

```
In [2]: def gcd(x, n):
        if x < n:
            x, n = n, x
        while x % n != 0:
            t = n
            n = x % n
            x = t
        return n
L = list(map(int, input("请输入一串数字").split()))
a = L[0]
for i in range(1, len(L)):
    a = gcd(a, L[i])
print("这些数的最大公因数是:", a)
```

请输入一串数字96 24 56 1200 256
这些数的最大公因数是：8

八、编程实现求解 $k(k \geq 3)$ 个数的最小公倍数。
(10分)

```
In [5]: def lcm(x, n):  
        if x < n:  
            x, n = n, x  
        a, b = x, n  
        while x % n != 0:  
            t = n  
            n = x % n  
            x = t  
        return a*b//n  
L = list(map(int, input("请输入一串数字").split()))  
a = L[0]  
for i in range (1, len(L)):  
    a = lcm(a, L[i])  
print("这些数的最小公倍数是:", a)
```

请输入一串数字2 8 36 32 512
这些数的最小公倍数是: 4608