Homework 4

Exercise 1

二分查找

题目：

编写一个程序使用二分法查找给定的包含若干整数的列表 `s` 中是否存在给定的整数 `k` 。若存在则输 出 `k` 的索引值, 否则输出 `-1` 。4.19已列出了部分代码, 需要实现函数 `is_sorted` 和递归函数 `binary_search` 。 `binary_search` 在列表 `s` 的索引值属于闭区间 `[low,high]` 的数据中查找 `k` , 若找到则返回 `k` 的索引值, 否则返回 `-1` 。 `binary_search` 的设计方案是：首先判断 `low` 是否大于 `high` ; 若是则返回 `1` ; 否则计算 `low` 和 `high` 的平均值 `mid` ; 若 `k` 等于 `s[mid]` , 则返回 `mid` ; 若 `k` 大于 `s[mid]` 或小于 `s[mid]` , 则根据具体情况确定合适的 `low` 和 `high` 值作为实参递归调用 `binary_search` 。

源代码：

```
import math
print("This is a python code for homework4-1: Binary search")


def is_sorted(fs):
    flag = True
    for j in range(len(fs)-1):
        if fs[j] > fs[j+1]:
            flag = False
    return flag


def qsort(fs):
    if len(fs) <= 1:
        return fs
    s1 = [i for i in fs if i < fs[0]]
    s2 = [i for i in fs if i > fs[0]]
    s0 = [i for i in fs if i == fs[0]]
    return qsort(s1) + s0 + qsort(s2)


def binary_search(f_s, f_low, f_high, f_k):
    if f_low > f_high:
        return -1
    f_mid = math.ceil(0.5*f_low+0.5*f_high)
    if f_s[f_mid] == f_k:
        return f_mid
    if f_s[f_mid] > f_k:
        if f_s[f_mid-1] == f_k:
            return f_mid-1
```

```
        elif f_mid - 1 == f_low:
            return -1
        return binary_search(f_s, f_low, f_mid-1, f_k)
    if f_s[f_mid] < f_k:
        return binary_search(f_s, f_mid, f_high, f_k)


s = [5, 6, 21, 32, 51, 60, 67, 73, 77, 99]
if not is_sorted(s):
    s = qsort(s)
print(binary_search(s, 0, len(s) - 1, 5))
print(binary_search(s, 0, len(s) - 1, 31))
print(binary_search(s, 0, len(s) - 1, 99))
print(binary_search(s, 0, len(s) - 1, 64))
print(binary_search(s, 0, len(s) - 1, 51))
```

运行结果：

```
/usr/local/bin/python3.9 "/Users/wangyijie/Library/Mobile
Documents/com~apple~CloudDocs/Study_in_USTC/杂事/python科学计算/hw4/hw4_1.py"

This is a python code for homework4-1: Binary search
0
-1
9
-1
4

Process finished with exit code 0
```

Exercise 2

有理数四则运算

题目：

有理数的一般形式是 a/b, 其中 a 是整数, b 是正整数, 并且当 a 非 0 时 $|a|$ 和 b 的最大公约数是 1 。编写一个模块`rational.py`实现有理数的四则运算。4.20已列出了部分代码, 需要实现标注了 `to be implemented` 的函数。程序中用一个列表 `[n, d]` 表示有理数, 其中`n`表示分子,`d`表示分母。函数`reduce`调用函数`gcd`进行约分。函数`add`、`sub`、`mul`和`div`分别进行加减乘除运算, 运算的结果都需要约分, 并且分母不出现负号。函数`test_allfunctions`使用已知答案的数据对这些运算进行测试。函数output按照示例的格式输出有理数, 例如`[-13,12]` 表示的有理数的输出结果是字符串 `"-13/12"`。用 户在命令行输入三个命名参数。`"-op"` 表示运算符, 可以是`"add"` (加法)、`"sub"` (减法)、`" mul"` (乘法)或`"div"` (除法)。`"-x"` 和 `"-y"` 表示进行计算的两个有理数。有理数以字符串的形式输入， 必须用圆括号括起, 分子和分母之间用`"/"`分隔。例如有理数 `"-20/3"`对应的输入形式是`"(-20/3)"`。函数`get_rational`从表示有理数的字符串中得到列表`[-20,3]`, 例如从字符串 `"(-20/3)"` 得到 `[-20,3]`。

源代码：

```
import sys
import math
print("This is a python code for homework4-2: Arithmetic operations for rational
numbers.")

def gcd(a, b):
    if a == b:
        return a
    elif a > b:
        return gcd(a-b, b)
    else:
        return gcd(a, b-a)


def reduce(n, d):
    temp = gcd(math.fabs(n), math.fabs(d))
    n = int(n / temp)
    d = int(d / temp)
    if d < 0:
        n = -n
        d = -d
    return [n, d]


def add(x, y):
    ansu = x[0]*y[1]+x[1]*y[0]
    ansd = x[1]*y[1]
    return reduce(ansu, ansd)


def sub(x, y):
    ansu = x[0] * y[1] - x[1] * y[0]
    ansd = x[1] * y[1]
    return reduce(ansu, ansd)


def mul(x, y):
    ansu = x[0] * y[0]
    ansd = x[1] * y[1]
    return reduce(ansu, ansd)


def div(x, y):
    ansu = x[0] * y[1]
    ansd = x[1] * y[0]
    return reduce(ansu, ansd)
```

```
def output(x):
    print(str(x[0])+'/'+str(x[1]))


def get_rational(fs):
    temps = list(fs)
    par1 = temps.index('(')
    par2 = temps.index('/')
    par3 = temps.index(')')
    if fs[par1 + 1] == '-':
        tempu = temps[par1 + 2:par2]
        numu = -int("".join(tempu))
    else:
        tempu = temps[par1 + 1:par2]
        numu = int("".join(tempu))
    if fs[par2 + 1] == '-':
        tempd = temps[par2 + 2:par3]
        numu = -numu
        numd = int("".join(tempd))
    else:
        tempd = temps[par2 + 1:par3]
        numd = int("".join(tempd))
    return [numu, numd]


if __name__ == '__main__':
    if len(sys.argv) == 1:
        print (__doc__)
    elif len(sys.argv) == 2 and sys.argv[1] == '-h':
        print (__doc__)
    elif len(sys.argv) == 2 and sys.argv[1] == 'test':
        test_all_functions()
    else:
        import argparse
        parser = argparse.ArgumentParser()
        parser.add_argument('--op', type=str)
        parser.add_argument('--x', type=str)
        parser.add_argument('--y', type=str)
        args = parser.parse_args()
        op = args.op
        x = get_rational(args.x); y = get_rational(args.y)
        f = {'add':add, 'sub':sub, 'mul':mul, 'div':div}
        output(f[op](x, y))
```

运行结果：

此处有两点值得注意：

1. 全程序输入输出使用终端参数，故程序不自带案例及结果计算和输出，在终端输入命令验证即可。
2. macOS终端下运行程序命令有所不同，老师在课件中命令为：

   ```
   run FILENAME.py --op div --x (-6/19) --y (-114/-28)
   ```

   但是在macOS系统下，如果希望在终端正常运行，应该输入：

   ```
   python3 hw4_2.py --op mul --x "(-6/19)" --y "(-114/18)"
   ```

   可见有两点不同，第一点是`run`命令更换为`python3`（注意，`python`命令也不能生效，因为macOS系统自带`python2`而不是`python3`）。第二点是，需要在输入的字符串参数加上`""`，否则会有如下报错：

   ```
   fish: Unknown command: -6/19
   in command substitution
   fish: Unknown command
   python3 hw4_2.py --op div --x (-6/19) --y (-114/-28)
                                     ^
   ```

以上均操作正确以后，在终端得到的输入输出如下：

```
wangyijie@wangyijiedeMacBook-Pro-10 ~/L/M/c/S/杂/p/hw4> python3 hw4_2.py --op add --x "
(2/3)" --y "(-70/40)"
This is a python code for homework4-2: Arithmetic operations for rational numbers.
-13/12
wangyijie@wangyijiedeMacBook-Pro-10 ~/L/M/c/S/杂/p/hw4> python3 hw4_2.py --op sub --x "
(-20/3)" --y "(120/470)"
This is a python code for homework4-2: Arithmetic operations for rational numbers.
-976/141
wangyijie@wangyijiedeMacBook-Pro-10 ~/L/M/c/S/杂/p/hw4> python3 hw4_2.py --op mul --x "
(-6/19)" --y "(-114/18)"
This is a python code for homework4-2: Arithmetic operations for rational numbers.
2/1
wangyijie@wangyijiedeMacBook-Pro-10 ~/L/M/c/S/杂/p/hw4> python3 hw4_2.py --op div --x "
(-6/19)" --y "(-114/-28)"
This is a python code for homework4-2: Arithmetic operations for rational numbers.
-28/361
```