欧拉工程解题方法分析

1. 3 或 5 的倍数

如果我们将小于 10 的所有是 3 或 5 倍数的自然数列出来，我们得到 3，5，6 和 9，它们的和是 23。与之类似，计算 1000 以下所有是 3 或 5 的倍数的自然数的和。

分析：此题解法较为直接，将 1000 以下所有 3 或 5 的倍数列出再求即可，以下代码使用了 python 的列表推导式以简化代码结构。

```
print sum([x for x in range(1,1000) if x%3==0 or x%5==0])
```

2. 偶数斐波那契数

斐波那契序列中的数都是由前两项加总得出，假设第一与第二项为 1 与 2，则前十项分别为：

$$1，2，3，5，8，13，21，34，55，89\cdots$$

考虑不超过四百万的斐波那契数，计算其中偶数斐波那契数的和。

分析：首先编写一个函数，这个函数计算小于等于给定 N 的斐波那契数，然后筛选出其中的偶数并求和。

```
def fibs_below(n,a=1,b=2):
    fib = [a,b]
    while True:
        a,b = b,a+b
        if b <= n:
            fib.append(b)
        else:
            return fib

print sum([x for x in fibs_below(4e6) if x%2==0])
```

3. 最大质因数

13195 的质因数分别为 5，7，13 与 29，600851475143 最大的质因数是多少？

分析：求解质因数的方法较多，最常用的方法为短除法，即对于任意大于 2 的自然数 N，先用 N 除以 2，再用所得之商即(N/2)再除以 2，直到商不能为 2 所整除，此时将被除数加一并比较其平方是否小于被除数，如果小于则再用商除以 3，如不能整除，则除以 5(因为所有 2 的倍数即偶数因数已在第一轮不断除以 2 时被排除了，所以之后都要加二)。这样循环下去直到除数的平方不再小于被除数，则退出循环，最后得到的 N 即为最大的质因数。实际执行中，要分两层循环，外层循环判断除数的平方是否小于被除数，内层循环判断被除数是否可以整除除数。

```
def max_prime_factor(n):
    i = 2
    while i * i < n:
        while n%i == 0:
            n /= i
        i += 2 if i>2 else 1
    return n

print max_prime_factor(600851475143)
```

4. 最大回文数乘积

回文数即从正反两边读都是一样的数，两个二位数的乘积中最大的回文数为$9009 = 91 \times 99$，寻找两个三位数乘积中最大的回文数。

分析：编写一个函数判断给定的数是否为回文数，即首先将数转化为字符串，再看该字符串与其逆转是否同一，如果同一则为回文数。然后在 999 至 100 的三位数做两两守相乘，从中筛选回文数并求最大值。

```
r = range(999,99,-1)
is_palindrome = lambda x : str(x) == str(x)[::-1]
print max([i*j for i in r for j in r if is_palindrome(i*j)])
```

5. 最小公倍数

2520 是可以被从一到十所有自然数整除的最小的数，即为从一到十的自然数的最小公倍数，求从一到二十所有自然数的最小公倍数。

分析：根据欧几里德公式，我们可以两个数最大公约数推出两个数的最小公倍数，即对于任意自然数a,b，设两个数的最大公约数为$gcd(a,b)$，则两个数的最小公倍数

$$lcm\ a,b\ = \frac{ab}{gcd\ a,b}$$

据此，我们可以先求出两个数的最小公倍数，再用这个最小公倍数与第三个数求最小公倍数，从而迭代得出N个数的最小公倍数。代码中从外部库中导入了 python 中求最大公约数的函数，并利用 `reduce()`函数迭代求出多个数的最小公倍数。

```
from fractions import gcd

def lcm(numbers):
    def lcm(a, b):
        return (a * b) // gcd(a, b)
    return reduce(lcm, numbers)

print lcm(range(1,21))
```

6. 和的平方与平方的和之间的差值

前十个自然数的平方的和为：

$$1^2+2^2+3^2+\cdots 10^2=385$$

而前十个自然数和的平方为：

$$1+2+3\cdots+10\ ^2=55^2=3025$$

两者的差为$3025-385=2640$。因此求前一百个自然数的和的平方与平方的和之间的差值。

分析：此题至少有两种解法：第一种解法为求各公式法，因为自然数的和以及平方的和都有相应的求和公式，可以根据公式直接推出两者差值的计算公式；第二种方式即为直接计算并求差值，因为计算量不大，这里使用第二种方法。

```
arr = [x**2 for x in range(1,101)]
res = (sum(range(1,101)))**2 - sum(arr)
print res
```

7. 第 10001 个质数

列出前六个质数，我们可以发现第六个质数为 13，那么第 10001 个质数是多少？

分析：一般而言，对于第n个素数p_n，满足以下不等式(参见素数计数函数)：

$$ln\ n\ +ln\ ln\ n\ \ -1<\frac{p_n}{n}<ln\ n\ +ln(ln(n)$$

在这里我们只关注上界，则可以得到$p_n<nln\ n\ +nln(ln\ n\ )$，我们先筛选出从 2 到上界的所有质数，再取其中的第 10001 个质数，即得到结果。

```
from sympy import sieve
from math import log,ceil
```

```
def nth_prime(n):
    up_bound = ceil((log(n)+log(log(n))) * n)
    primes = list(sieve.primerange(1,up_bound))
    return primes[n-1]

print nth_prime(10001)
```

8. 序列中的最大乘积

在下面 1000 位数中，连续四个数的最大乘积为 9 × 9 × 8 × 9 = 5832。寻找连续十三数的最大乘积，这个乘积是多少？

73167176531330624919225119674426574742355349194934
96983520312774506326239578318016984801869478851843
85861560789112949495459501737958331952853208805511
12540698747158523863050715693290963295227443043557
66896648950445244523161731856403098711121722383113
62229893423380308135336276614282806444486645238749
30358907296290491560440772390713810515859307960866
70172427121883998797908792274921901699720888093776
65727333001053367881220235421809751254540594752243
52584907711670556013604839586446706324415722155397
53697817977846174064955149290862569321978468622482
83972241375657056057490261407972968652414535100474
82166370484403199890008895243450658541227588666881
16427171479924442928230863465674813919123162824586
17866458359124566529476545682848912883142607690042
24219022671055626321111109370544217506941658960408
07198403850962455444362981230987879927244284909188
84580156166097919133875499200524063689912560717606
05886116467109405077541002256983155200055935729725
71636269561882670428252483600823257530420752963450

分析：此题解题思路较为直接，分为以下步骤：1)将以上数字存入到 TXT 文件中，用 python 导入并存入到一个 LIST 中；2)从 LIST 中第 0 位开始直到第 987 位，依次求连续十三位数的乘积并存入到结果 LIST 中；3)求结果 LIST 的最大值得到结果。

```
data = ''
f = open("euler/ep08.txt")
for line in f.readlines():
    data = data + line.strip()

res = []
for i in range(988):
```

```
    sub = [int(x) for x in data[i:i+13]]
    prod = reduce(lambda x,y:x*y, sub)
    res.append(prod)
print max(res)
```

9. 特殊的毕达哥拉斯三元数

毕达哥拉斯三元数是指一类三个自然数的集合，其中$a < b < c$ 且 $a^2 + b^2 = c^2$。例如$3^2 + 4^2 = 25 = 5^2$。仅存在一组毕达哥拉斯三元数使得$a + b + c = 1000$，求abc。

分析：根据欧几里德公式，对于$m > n > 0$，$a = m^2 - n^2, b = 2mn, c = m^2 + n^2$构成毕达哥拉斯三元数，则有$a + b + c = m^2 - n^2 + 2mn + m^2 + n^2 = 2m^2 + 2mn = 2m\ m + n\ = 1000$，即$m\ m + n\ = 500$。可以看出$m < m + n$且$m, n$都为 500 的因数，则$m < \sqrt{500} = 22.36$，则可以从 22 依次递减一进行尝试，寻找 500 的因数，到 20 时寻找结束，得到$n = 5$，则跳出循环。最后，根据以下欧几里德公式计算abc。

```
from math import sqrt

def find_pytha():
    upBoundary = int(sqrt(500)) + 1
    for i in range(upBoundary,1,-1):
        if 500%i == 0:
            n = 500/i - i
            if i > n:
                return i,n

i,n = find_pytha()
a = i**2 + n**2
b = 2*i*n
c = i**2 - n**2
print a*b*c
```

10.质数的和

10 以下的质数的和为 2 + 3 + 5 + 7 = 17，求所有两百万以下的质数的和。

分析：首先筛选出两百万以下的质数，然后求和即可。

```
from sympy import sieve
print sum(list(sieve.primerange(1,2e6)))
```

11.网格中的最大乘积

在下面这个 20×20 的网格中，对角线上相邻的四个元素已经用红色标记出来了，这四个数的乘积为 26 ×63 ×78 ×14 = 1788696。求在这个网格中，同一方向(上、下、左、右以及对角线)相邻四个元素最大乘积。

08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 **26** 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 **63** 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 **78** 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 **14** 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48

分析：此题似乎没有什么讨巧的简便算法，先将数据存入到一个嵌套列表当中，然后依次遍历每个元素，求它每个方向上相邻四个元素(如果有的话，需要条件判断)的乘积，存入到结果列表中，然后求最大值。

```
with open('euler/ep11.txt','r') as f:
    data = [map(int, s.split()) for s in f.readlines()]

def grid_largest_prod(data):
    import numpy as np
    length = len(data)
    res = np.zeros((length,length))
    right = down = down_right = left_down = 0
    for i in range(length):
        for j in range(length):
            right_exist = all([x<length for x in [i,i+1,i+2,i+3]])
            down_exist = all([x<length for x in [j,j+1,j+2,j+3]])
            left_exist = all([x>=0 for x in [i,i-1,i-2,i-3]])
            if right_exist:
                right = data[i][j]*data[i+1][j]*data[i+2][j]*data[i+3][j]
```

```
            if down_exist:
                down = data[i][j]*data[i][j+1]*data[i][j+2]*data[i][j+3]
            if right_exist and down_exist:
                down_right = data[i][j]*data[i+1][j+2]*data[i+2][j+2]*data[i+3][j+3]
            if left_exist and down_exist:
                left_down = data[i][j]*data[i-1][j+1]*data[i-2][j+2]*data[i-3][j+3]
            res[i][j] = max(right,down,down_right,left_down)
    max_res = np.max(res)
    return max_res

print grid_largest_prod(data)
```

12.高度可除的三角数

三角数即由依次排列的自然数的和构成，所以第 7 个三角数是 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28，前十个三角数是：1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...。让我们列出前七个三角数的因子：

1: 1
3: 1,3
6: 1,2,3,6
10: 1,2,5,10
15: 1,3,5,15
21: 1,3,7,21
28: 1,2,4,7,14,28

可以看出 28 是第一个因子超过 5 的三角数，求第一个因子超过五百万的三角数。

分析：首先编写一个函数计算某给定自然数的因子数，每个自然的因子对称分布在平方根的两边，因此只需要找出小于平方根的因子数再乘以二便可以得到总的因子数。需要注意的，这样计算出来的平方数的因子会多一，因此需要从总数中减去一。接着，依次对每个三角数求其因子数，只到找到第一个因子数超过五百万的三角数，然后返回。

```
from math import sqrt

def num_of_divisor(n):
    if n == 1:
        return 1
    else:
        up_bound = int(sqrt(n)) + 1
        num = 0
        for i in range(1,up_bound):
            if n%i == 0:
                num = num + 1
```

```
        return (2*num) - (1 if (up_bound-1)**2 == n else 0)

def numd_of_triangle():
    n = 7
    while True:
        triangle = n*(n+1)//2
        num_d = num_of_divisor(triangle)
        if num_d <= 500:
            n = n + 1
        else:
            return triangle

print numd_of_triangle()
```

13. 大数之和

计算如下一百个 50 位数的和(见数据文件 ep13.txt)，求这个和的前十位数。

分析：将这些数存入到 TXT 文件中，用 python 依次读取并加总，然后将该和转化为字符串并加前十位。

```
large_sum = 0
f = open("euler/ep13.txt")
for line in f.readlines():
    large_sum += int(line)
print str(large_sum)[:10]
```

14. 最长的考拉兹序列

对于所有正整数，考虑如下迭代序列：

$$n \to \frac{n}{2}\ if\ n\ is\ even, n \to 3n+1\ if\ n\ is\ odd$$

从 13 开始并使用以上迭代规则，我们可以得到以下序列：

$$13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

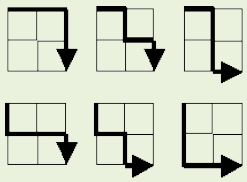
可以看出这个序列从 13 开始并到 1 结束总共包含 10 个数。考拉兹猜想指出使用以上迭代规则，所有正整数都会最终回到一，虽然这个猜想仍未得到证明。求在一百万以下，那个超始数可以产生最长的考拉兹序列？(注意：序列中包含的数的个数可以超过一百万。)

分析：首先编写一个计算考拉兹序列长度的函数，然后使用该函数对一至一百万之间的自然数求其考拉兹序列的长度，找出其中的最大值并返回初始点，即为所求。

```
import numpy as np

def num_of_terms(n):
    collatz = lambda n : 3*n+1 if n%2==1 else n//2
    num = 1
    while n !=1:
        n = collatz(n)
        num += 1
    return num

def max_start(n):
    terms_arr = np.array([num_of_terms(x) for x in range(1,n+1)])
    index = terms_arr.argmax() + 1
    return index

print max_start(1000000)
```

15.格子路径

在一个 2×2 的栅格中，从左上角出来，只能向右或向下移动，总共有 6 条路径可以到达栅格的右下角。



求在一个 20×20 栅格中，有多少条移动路径？

分析：根据维基百科上相关词条，对于一个$N \times N$的栅格，要从其 0,0 点到达其 n,k 点，总共可以行走的路线有$C(n+k,n)$条。为计算组合数，需要首先定义一个阶乘函数，然后便可求出结果。

```
def comb_num(n,k):
    fac = lambda n:reduce(lambda x,y:x*y,range(1,n+1))
    num = fac(n)/(fac(n-k)*fac(k))
    return num

def lattice_paths(n,k):
    return comb_num(n+k,k)

print lattice_paths(20,20)
```

16.指数各位数之和

$2^{15}=32768$ 且各个位数之和为 $3+2+7+6+8=26$，求2^{1000}各个位数之和。

分析：求出2^{1000}，将其转化字符串，并将字符串转化为整数再求和。

```
print sum([int(x) for x in str(2**1000)])
```

17.数词字母数量

如果把数字一到五用单词写出来，即 one, two, three, four, five，那么总共使用了$3+3+5+4+4=19$个字母。如果把从一到一千的数字用对应的单词写出来，总共需要多少个字母？
注意：不要计算空格与连字符，如 342(three hundred and forty-two)包含 23 个字母，而 115(one hundred and fifteen)包含 20 个字母，and 的使用遵守英式英语的规则。

分析：将所有需要用到的数词都存入到一个 CSV 文件中并用 pandas 导入，并将其转化为一个词典。分别考虑$[1,20]$，$[21,100]$和$[101,1000]$不同的数词对应规则，编写一个将数字转化为相应的词语的函数，但忽略空格与连字符。列出所有 1000 以下的数字，转化为对应的词语并统计总的字母数量。

```
import pandas as pd
df = pd.read_csv('euler/ep17.csv',header=None)
words_dict = df.set_index(0).to_dict()[1]

def num_to_words(n):
    if n in words_dict.keys():
        return words_dict[n]
    else:
        if 21<=n<=100:
            ten_digit = n/10*10
            digit = n%10
            return words_dict[ten_digit] + words_dict[digit]
        elif 101<=n<=1000:
            hundred_dict = n/100
            remainder = n%100
            if remainder == 0:
                return words_dict[hundred_dict] + 'hundred'
            else:
                return words_dict[hundred_dict] + 'hundredand' + num_to_words(remainder)

print sum([len(num_to_words(x)) for x in range(1,1001)])
```

18.最大和的路径

从以下这个三角形的顶部开始，向相邻的下一行的数字移动，经过之数所能得到的最大的和为 23，即：

3
7 4
2 4 6
8 5 9 3

$3+7+4+9=23$。对于以下三角形(见数据文件 ep18.txt)，求期从上到下所有路径中最大的和。【注：由于总共只有 16384 条路径，因此可以尝试每条路径并求最大值。然而，问题 67 与此问题类似但有三角形共有 100 行，从而无法用暴力枚举方法解决，这需要一个聪明的方法。】

分析：这是一道经典的动态规划问题，为求三角形从上到下的最大和，先从最下一行开始倒推，即：

$$max(8+2,5+2)=10, max(5+4,9+4)=13, max(9+6,3+6)=15$$

这样可以将最下二行合为一行，即：

3
7 4
10 13 15

依次类推，可以继续倒推出：

$$max(10+7,13+7)=20, max(13+4,15+4)=19$$

即：

3
20 19

容易得到 23 是最大值，而路径也是题目中所给出的路径。对于题目中给出的三角形，与之类似可以得出答案。

```
with open('euler/ep18.txt') as f:
    table = [map(int,s.split()) for s in f.readlines()]

for row in range(len(table)-1, 0, -1):
    for col in range(0, row):
        table[row-1][col] += max(table[row][col], table[row][col+1])

print table[0][0]
```

19.计算星期天的数量

以下信息是题目直接给予你的，当然你也可以自己做一些研究：1900 年 1 月 1 日是星期一，一月有 30 天的

月份分别为九月、四月、六月与十一月，除开二月以外其它月份都有 31 天。二月在闰年有 29 天，其它年份有 28 天。闰年是年份可以整除四的年份，除非这个年份跨世纪的年份，但在跨世纪的年份中如果该年份可以整除 400 则也为闰年。在 20 世纪(1900 年 1 月 1 日至 2000 年 12 月 31 日)每个月的首日是星期天的次数是多少？

分析：本题可以使用 python 中的 datetime 模块解决，从 1990 年 1 月 1 日开始循环至 2000 年 12 月 31 日结束，如果某天同时为一个月的第一天且是星期天，则计数加一，这样便可以统计出整个二十世纪符合条件的天数。

```
def count_sundays():
    import datetime as dt
    delta = dt.timedelta(days=1)
    start_date = s = dt.datetime(1901, 1, 1)
    end_date = dt.datetime(2000, 12, 31)
    count = 0
    while start_date <= end_date:
        if start_date.day==1 and start_date.isoweekday()==7: count+=1
        start_date += delta
    return count

print count_sundays()
```

20.计算星期天的数量

n的阶乘可以这么计算：$n!=n(n-1)(n-2)...3\cdot 2\cdot 1$。例如 10 的阶乘等于$3628800$，并且其各位数的和为$3+6+2+8+8+0+0=27$。求$100!$各位数之和。

分析：直接求出$100!$，并计算其各位数之和即可。

```
fac = lambda n:reduce(lambda x,y:x*y,range(1,n+1))
print sum([int(x) for x in str(fac(100))])
```

21. 亲和数

令$d(n)$表示自然数n所有真因子(除开数n本身的所有因子)的和，如果$d(a)=b$且$d(b)=a$，其中$a\neq b$，那么a与b便为亲和数对，其中的每个数称为亲和数。例如，220 的真因子为 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 和 110，所以$d(220)=284$；284 的真因子为 1, 2, 4, 71 与 142，所以$d(284)=220$。求10000以下所有亲和数的和。

分析：首先编写一个计算任意自然数所有因子的函数，然后编写一个计算所有真因子之和的函数。在所有小于 10000 的自然数中寻找亲和数并存入到结果列表中。满足亲和数的条件为：$d(d(a))=a$且$d(a)\neq a$。最

后对结果列表求和。

```
from math import sqrt

def divisors(n):
    divs = []
    for i in range(1,int(sqrt(n))+1):
        if n%i == 0:
            divs.append(i)
            divs.append(n/i)
    return list(set(divs))

sum_of_divs = lambda n : sum(divisors(n))-n

def amicable_num(n):
    arr = []
    for i in range(1,n+1):
        if sum_of_divs(sum_of_divs(i)) == i and sum_of_divs(i) !=i:
            arr.append(i)
    return arr

print sum(amicable_num(10000))
```

22.姓名分值

题目给定的数据文件(见数据文件 ep22.txt)包含了五千个名字，首先应将其按字母表顺序排列，计算每个名字的字母表值，然后乘以每个名字在列表中位次得到该名字的姓名分值。例如，当名字列表按字母表顺序排列，COLIN 的字母表值为$3+15+12+9+14=53$，而其在列表中的位次为938，因此 COLIN 的分值为两数相乘得到$938\times 53=49714$。求文件中所有名字的姓名分值之和。

分析：从文件中导入数据，并存入到列表中，保证每个名字为列表中的一个元素。编写一个计算每个名字字母表值的函数，即考虑名字中每个字母相对于字母 A 的位次，并加总构成名字所有字母的位次值。编写计算姓名分值的函数，首先将所有名字按字母表顺序排列计算其位次值，然后再乘以其字母表值。加总所有的姓名分值得到结果。

```
with open('euler/ep22.txt','r') as f:
    data = f.read().replace('"',"").split(',')

def alphabet(word):
    alpha = lambda s : ord(s) - ord('A') + 1
    res = sum([alpha(x) for x in word])
    return res
```

```
def sum_name_score(data):
    name_score = lambda word : alphabet(word) * (sorted(data).index(word)+1)
    res = sum([name_score(word) for word in data])
    return res
```

23.非过剩数之和