TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Chương trình Đào tạo Quốc tế SIE



**Báo cáo môn học**

Môn: Project I

Đề tài: Project Euler

Giáo viên: Lê Bá Vui

Sinh Viên

Nguyễn Đình Tiến

MSSV: 20168533

Lớp: VUWIT15

Ngôn ngữ Lập trình: Python

*Hà Nội, tháng 12 năm* *2018*

# MỤC LỤC

1. [TUẦN 1 (1|2|3) 3](#_Toc529319818)
2. [TUẦN 2 (4|5|6) 6](#_Toc529319818)
3. [TUẦN 3 (7|8|9) 10](#_Toc529319818)
4. [TUẦN 4 (10|11|16) 14](#_Toc529319818)
5. [TUẦN 5 (13|14|20) 18](#_Toc529319818)
6. [TUẦN 6 (21|22|25) 21](#_Toc529319818)
7. [TUẦN 7 (28|29|35) 25](#_Toc529319818)
8. [TUẦN 8 (36|40|42) 29](#_Toc529319818)
9. [TUẦN 9 (43|44|45) 32](#_Toc529319818)
10. [TUẦN 10 (48|50|52) 36](#_Toc529319818)
11. [TUẦN 11 (53|55|56) 40](#_Toc529319818)
12. [TUẦN 12 (57|58|62) 44](#_Toc529319818)­
13. [TUẦN 13 (67|18|97) 48](#_Toc529319818)
14. [TUẦN 14 (12|23|24) 52](#_Toc529319818)
15. [TUẦN 15 (27|33|39) 56](#_Toc529319818)

[TÀI LIỆU THAM KHẢO 60](#_Toc529319818)

1. *Tuần 1 (1|2|3)*

Problem 1:

1.Đề bài:

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000.

\* Tính tổng tất cả các số nguyên nhỏ hơn 1000 chia hết cho 3 hoặc chia hết cho 5

2.Thuật toán:

- Dùng 1 biến sum = 0 ban đầu để cộng tổng

- Dùng vòng lặp for để duyệt các giá trị từ 0 đến 1000, khi gặp số chia hết cho 3 hoặc 5 thì cộng thêm vào tổng

3.Đáp án: 233168

4.Mã nguồn:

*# Multiples of 3 and 5*

sum = 0

for i in range(1000):

if i % 3 == 0 or i % 5 == 0:

sum += i

print(sum)

Problem 2:

1.Đề bài: Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

\* Tính tổng tất cả số fibonacci chẵn không lớn hơn 4.000.000

2.Thuật toán:

- Dùng vòng for thêm vào list fibonacci những số fibonacci (List có sẵn 2 phần tử đầu để tính toán các phần tử sau)

- Dùng vòng for thêm vào list total tất cả số chẵn trong list fibonacci

- Tính tổng các item trong list total

3.Đáp án: 4613732

4.Mã nguồn:

*# Even Fibonacci numbers*

fibonacci = [1,2]

total = []

for i in range(4000001):

if i == fibonacci[-1] + fibonacci[-2]:

fibonacci.append(i)

for j in fibonacci:

if j % 2 ==0:

total.append(j)

print(sum(total))

Problem 3:

1.Đề bài:

The prime factors of 13195 are 5, 7, 13 and 29.

What is the largest prime factor of the number 600851475143 ?

\* Tìm ước số nguyên tố lớn nhất của số 600851475143

2.Thuật toán:

- Phân tích số đã cho thành các thừa số nguyên tố (Lưu ý chỉ check đến số có giá trị bằng căn bậc 2 của số đã cho là đủ) (Dùng vòng while để lặp i tới giá trị căn bậc 2 của số đã cho)

- Dùng tiếp vòng while để phân tích n thành các thừa số nguyên tố và gán luôn n với giá trị lớn nhất (vì i chạy từ nhỏ đến lớn nên n sẽ được phân tích và gán nhỏ dần)

- Đến khi n không phân tích được nữa thì 2 vòng while sẽ dừng lại ở thừa số nguyên tố n có giá trị lớn nhất

3.Đáp án: 6857

4.Mã nguồn:

*# Largest prime factor*

n = 600851475143

i = 2

while i \*\* 2 < n:

while n % i == 0:

n = n / i

i += 1

print(n)

1. *Tuần 2 (4|5|6)*

Problem 4:

1.Đề bài: A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is 9009 = 91 × 99.

Find the largest palindrome made from the product of two 3-digit numbers.

\* Tìm số đối xứng lớn nhất bằng tích của 2 số có 3 chữ số

2.Thuật toán:

- Viết hàm kiểm tra số đối xứng, duyệt các số được nhân từ 999 và 999 đổ xuống

- Nếu là số đối xứng và là tích của 2 số có 3 chữ số thì trả về kết quả cần tìm

3.Đáp án: 906609

4.Mã nguồn:

*# Largest palindrome product*

**def** isPalindromic(n):

if str(n) == str(n)[::-1]: return True

return False

for i in range(999\*999, 10000, -1):

j = 100

isFound = False

while j < 1000 and isPalindromic(i):

if i % j != 0: j += 1

else:

k = i / j

if 99 < k < 1000:

print(i)

isFound = True

break

else:

j += 1

if isFound:

break

Problem 5:

1.Đề bài:

2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder.

What is the smallest positive number that is evenly divisible by all of the numbers from 1 to 20?

\* Tìm số nguyên nhỏ nhất chia hết cho tất cả các số từ 1 đến 20

2.Thuật toán:

- Phân tích các số trong dãy thành thừa số nguyên tố, lấy tất cả thừa số với số mũ lớn nhất nhân lại với nhau (Chi tiết Comment trên mã nguồn)

4.Đáp án: 232792560

5.Mã nguồn:

*# Smallest multiple*

*# Nhập vào dãy số*

sequence = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

*# Tạo ra các List và Dictionary để check và add các ước và số mũ*

*# List chứa các dictionary, mỗi dictionary có key và value là thừa số nguyên tố và số mũ của nó khi phân tích từng số trong dãy*

list\_number = []

dict\_number = {}

list\_key = []

*# Thuật toán thêm vào list\_number các dictionary thỏa mãn miêu tả trên, ta sẽ có các thừa số nguyên tố và số mũ khi phân tích từng số trong dãy*

for number in sequence:

dict\_number = {}

i = 1

while i <= number:

if number % i == 0:

if i not in list\_key:

list\_key.append(i)

dict\_number[i] = 1

else:

ex\_number = 0

for key, value in dict\_number.items():

if key == i:

ex\_number = value

dict\_number[i] = ex\_number + 1

if i == 1:

i += 1

else:

number = number / i

else:

i += 1

list\_number.append(dict\_number)

*# Tạo ra 1 dictionary (list\_result) chứa key là các thừa số nguyên tố, value là số mũ lớn nhất sau khi tổng hợp từ list\_number*

list\_result = {}

for item in list\_key:

list\_result[item] = 1

*# Thuật toán lấy ra giá trị lớn nhất của số mũ để gán vào value của key là 1 thừa số nguyên tố*

for key, value in list\_result.items():

for index1, item1 in enumerate(list\_number):

for key1, value1 in item1.items():

if key1 == key and value1 > value:

list\_result[key] = value1

value = value1

*# Thuật toán tính tích của các thừa số nguyên tố với số mũ lớn nhất*

scale = 1

for key2, value2 in list\_result.items():

scale \*= key2 \*\* value2

*# Kết quả chính là giá trị cần tìm*

print(scale)

Problem 6:

1.Đề bài: The sum of the squares of the first ten natural numbers is,

12 + 22 + ... + 102 = 385

The square of the sum of the first ten natural numbers is,

(1 + 2 + ... + 10)2 = 552 = 3025

Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is 3025 − 385 = 2640.

Find the difference between the sum of the squares of the first one hundred natural numbers and the square of the sum.

\* Tính hiệu bình phương của tổng và tổng cuả bình phương và các số từ 1 đến 100

2.Thuật toán:

- Dùng vòng for duyệt từ 1 đến số 100, tính tổng các bình phương từng số một và tính tổng tất cả các số, khi đã có tổng tất cả các số thì ta bình phương lên

- Lấy bình phương tổng các số trừ đi tổng các số bình phương, ta được giá trị cần tìm

4.Đáp án: 25164150

5.Mã nguồn:

*# Sum square difference*

*# Bình phương của tổng*

square\_of\_summ = 0

*# Tổng các bình phương*

summ\_of\_square = 0

for i in range(1, 101):

summ\_of\_square += i\*\*2

square\_of\_summ += i

*# Tính ra bình phương của tổng các số đã tính ra ở trên*

square\_of\_summ = square\_of\_summ\*\*2

print(square\_of\_summ - summ\_of\_square)

1. *Tuần 3(7|8|9)*

Problem 7:

1.Đề bài:

By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6th prime is 13.

What is the 10 001st prime number?

\* Tìm số nguyên tố thứ 10001

2.Thuật toán:

- Xét các số từ 1 đến n cho trước, sau khi xét xong 1 số sẽ loại bỏ hết các số nhỏ hơn n và là là bội của số hiện tại ra khỏi tập các số cần xét, như vậy các số ở lại đều là số nguyên tố không có một ước nào

- Vòng lặp dừng lại khi đủ 10001 số nguyên tố, hiển thị ra số cuối trong list là kết quả cần tìm

3.Đáp án: 104743

4.Mã nguồn:

*# 10001st prime*

from math import ceil

list\_primes = []

n = 1000000

primes = [True] \* n

primes[0] = False

primes[1] = False

roundUp = **lambda** n, prime: int(ceil(n / prime))

for currentPrime in range(2, n):

if not primes[currentPrime]: continue

list\_primes.append(currentPrime)

for i in range(2, roundUp(n, currentPrime)):

primes[i \* currentPrime] = False

if len(list\_primes) == 10001: break

print(list\_primes[-1])

Problem 8:

1.Đề bài:

The four adjacent digits in the 1000-digit number that have the greatest product are 9 × 9 × 8 × 9 = 5832.

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

Find the thirteen adjacent digits in the 1000-digit number that have the greatest product. What is the value of this product?

\* Tính tích lớn nhất của 13 số liền nhau trong chuỗi số trên

2.Thuật toán:

- Đưa số có 1000 chữ số vào list chứa 1000 chữ số đó

- Xét từng phần tử trong list nhân với 12 phần tử trước nó và cho các giá trị nhân ra vào list\_value

- Lấy ra giá trị lớn nhất trong list\_value, ta được tích lớn nhất của 13 chữ số liền nhau.

3.Đáp án: 23514624000

4.Mã nguồn:

*# Largest product in a series*

number = str(7316717653133....7530420752963450) *# Tất cả các số*

list\_digits = list(number)

list\_value = []

for index, item in enumerate(list\_digits):

value = 1

for i in range (0,13):

value \*= int(list\_digits[index-i])

list\_value.append(value)

print(max(list\_value))

Problem 9:

1.Đề bài:

A Pythagorean triplet is a set of three natural numbers, *a* < *b* < *c*, for which,

*a*2 + *b*2 = *c*2

For example, 32 + 42 = 9 + 16 = 25 = 52.

There exists exactly one Pythagorean triplet for which *a* + *b* + *c* = 1000.  
Find the product *abc*

\* Tính tích 3 số a, b, c thỏa mãn *a* + *b* + *c* = 1000 và *a*2 + *b*2 = *c*2

2.Thuật toán:

- Gía trị lớn nhất của a chỉ có thể là 332 nên ta duyệt a từ 1 đến 332

- Trong đó duyệt tiếp b từ 1 đến 1000 – a

- Trong từng cặp giá trị khi duyệt, kiểm tra xem giá trị c = 1000 – a – b có thỏa mãn *a*2 + *b*2 = *c*2 hay không, nếu có ta nhân 3 số a, b, c với nhau và được giá trị cần tìm

3.Đáp án: 31875000

4.Mã nguồn:

*# Special Pythagorean triplet*

for a in range(1, 332):

for b in range(a, 1000 - a):

c = 1000 - a - b

if a \*\* 2 + b \*\* 2 == c \*\* 2:

print(a, b, c)

print("Product: {}".format(a \* b \* c))

1. *Tuần 4 (10|11|16)*

Problem 10:

1.Đề bài:

The sum of the primes below 10 is 2 + 3 + 5 + 7 = 17.

Find the sum of all the primes below two million.

\* Tính tổng tất cả số nguyên tố nhỏ hơn 2.000.000

2.Thuật toán:

- Sử dụng thuật toán tìm số nguyên tố giống Problem007 với n = 2.000.000, ta được 1 list các số nguyên tố nhỏ hơn 2.000.000

- Tình tổng của list, ta được kết quả cần tìm

3.Đáp án: 142913828922

4.Mã nguồn:

*# Summation of primes*

from math import ceil

list\_primes = []

n = 2000000

primes = [True] \* n

primes[0] = False

primes[1] = False

roundUp = **lambda** n, prime: int(ceil(n / prime))

for currentPrime in range(2, n):

if not primes[currentPrime]:

continue

list\_primes.append(currentPrime)

for i in range(2, roundUp(n, currentPrime)):

primes[i \* currentPrime] = False

print(sum(list\_primes))

Problem 11:

1.Đề bài:

In the 20×20 grid below, four numbers along a diagonal line have been marked in red.

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

The product of these numbers is 26 × 63 × 78 × 14 = 1788696.

What is the greatest product of four adjacent numbers in the same direction (up, down, left, right, or diagonally) in the 20×20 grid?

\* Tìm tích lớn nhất của 4 số liền nhau trong mảng số trên theo tất cả các hướng (trên dưới trái phải)

2.Thuật toán:

- Đưa toàn bộ số vào List

- Xét từng phần tử trong list nhân với 3 phần tử theo hàng trên, trái, chéo trên bên phải, chéo trên bên trái của nó và cho các giá trị nhân được vào list\_value

- Lấy ra giá trị lớn nhất trong list\_value, ta được tích lớn nhất của 4 chữ số liền nhau theo tất cả các hướng có thể (8 hướng nhưng vì duyệt tất cả các số nên xét nhân theo 4 hướng là đủ)

3.Đáp án: 70600674

4.Mã nguồn:

*# Largest product in a grid*

number\_input = "08 02 22 97 38 15 ... 52 01 89 19 67 48" *# Tất cả các số*

list\_number = number\_input.split(" ")

list\_value = []

for index, item in enumerate(list\_number):

value2 = 1 *# Up*

value1 = 1 *# Right*

value3 = 1 *# Up-Right*

value4 = 1 *# Up-Left*

for i in range(0, 80, 20):

value1 \*= int(list\_number[index-i])

for i in range(0, 4):

value2 \*= int(list\_number[index-i])

for i in range(0, 76, 19):

value3 \*= int(list\_number[index-i])

for i in range(0, 84, 21):

value4 \*= int(list\_number[index-i])

list\_value.append(value1)

list\_value.append(value2)

list\_value.append(value3)

list\_value.append(value4)

print(max(list\_value))

Problem 16:

1.Đề bài:

215 = 32768 and the sum of its digits is 3 + 2 + 7 + 6 + 8 = 26.

What is the sum of the digits of the number 21000?

\* Tính tổng tất cả các kí tự của 21000

2.Thuật toán:

- Tính giá trị 21000, ép kiểu thành String và tách thành tất cả kí tự đưa vào 1 list

- Dùng vòng for xét list đó và cộng tổng tất cả các kí tự

3.Đáp án: 1366

4.Mã nguồn:

*# Power digit sum*

n = 2 \*\* 1000

sum = 0

digits = list(str(n))

for index, item in enumerate(digits):

sum += int(item)

print(sum)

1. *Tuần 5 (13|14|20)*

Problem 13:

1.Đề bài:

Work out the first ten digits of the sum of the following one-hundred 50-digit numbers.

37107287533902102798797998220837590246510135740250  
46376937677490009712648124896970078050417018260538  
............................   
20849603980134001723930671666823555245252804609722  
53503534226472524250874054075591789781264330331690

\* Tìm 10 chữ số đầu tiên của tổng của 100 số có 50 chữ số trên

2.Thuật toán:

- Tính tổng 100 số có 50 chữ số

- Tách kết quả thành chuỗi và cho vào List, lấy ra 10 số đầu tiên

3.Đáp án: 5537376230

4.Mã nguồn:

*# Work out the first ten digits of the sum of the following one-hundred 50-digit numbers.*

sequence = '''37107287533902102798797998220837590246510135740250

.......

53503534226472524250874054075591789781264330331690''' *# Tất cả các số*

list\_sequence = sorted(sequence.split('\n'))

total = 0

for index, item in enumerate(list\_sequence):

total += int(item)

list\_digits = list(str(total))

ten\_digits = ""

for i in range(0,10):

ten\_digits += str(list\_digits[i])

print(ten\_digits)

Problem 14:

1.Đề bài:

The following iterative sequence is defined for the set of positive integers:

*n* → *n*/2 (*n* is even)  
*n* → 3*n* + 1 (*n* is odd)

Using the rule above and starting with 13, we generate the following sequence:

13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1

It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1.

Which starting number, under one million, produces the longest chain?

**NOTE:** Once the chain starts the terms are allowed to go above one million.

\* Tìm số nhỏ hơn 1 triệu thỏa mãn khi tạo ra chuỗi số từ số đó thì được chuỗi dài nhất

2.Thuật toán:

- Xét tất cả các số dưới 1.000.000

- Phân tích từng số theo quy tắc cho đến khi bằng 1 thì dừng lại

- Mỗi số trong chuỗi phân tích sẽ được thêm vào 1 list, nếu độ dài list lớn hơn độ dài list lớn nhất thì cập nhật

3.Đáp án: 837799

4.Mã nguồn:

*# Which starting number, under one million, produces the longest chain?*

list\_longest = []

number = 2

while number < 1000000:

list\_check = [number]

number\_check = number

while number\_check != 1:

if number\_check % 2 == 0:

number\_check = number\_check / 2

list\_check.append(number\_check)

elif number\_check % 2 == 1:

number\_check = number\_check \* 3 + 1

list\_check.append(number\_check)

if len(list\_check) > len(list\_longest):

list\_longest = list\_check

number += 1

print(list\_longest[0])

Problem 20:

1.Đề bài:

*n*! means *n* × (*n* − 1) × ... × 3 × 2 × 1

For example, 10! = 10 × 9 × ... × 3 × 2 × 1 = 3628800,  
and the sum of the digits in the number 10! is 3 + 6 + 2 + 8 + 8 + 0 + 0 = 27.

Find the sum of the digits in the number 100!

\*Tính tổng tất cả các chữ số của 100!

2.Thuật toán:

- Tính số 100 giai thừa, ép thành chuỗi và cho vào list, xét list và tính tổng từng chữ số một

3.Đáp án: 648

4.Mã nguồn:

*# Factorial digit sum*

import math

n = math.factorial(100)

sum = 0

digits = list(str(n))

for index, item in enumerate(digits): sum += int(item)

print(sum)

1. *Tuần 6 (21|22|25)*

Problem 21:

1.Đề bài:

Let d(*n*) be defined as the sum of proper divisors of *n* (numbers less than *n* which divide evenly into *n*).  
If d(*a*) = *b* and d(*b*) = *a*, where *a* ≠ *b*, then *a* and *b* are an amicable pair and each of *a* and *b* are called amicable numbers.

For example, the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 and 110; therefore d(220) = 284. The proper divisors of 284 are 1, 2, 4, 71 and 142; so d(284) = 220.

Evaluate the sum of all the amicable numbers under 10000.

\* Tính tổng tất cả số amicable dưới 10000

2.Thuật toán:

- Duyệt các số từ 1 đến 10000, thêm các ước của số đang xét vào 1 list và tính tổng các số trong list

- Xét các ước của giá trị tổng đó, giống như trên, nếu tổng các ước đó lại bằng số đang xét bên trên thì ta được 1 cặp (2 số) thỏa mãn

3.Đáp án: 31626

4.Mã nguồn:

*# Amicable numbers*

list\_amicable\_number = []

for n in range(1,10000):

if n not in list\_amicable\_number:

list\_divisor\_1 = []

list\_divisor\_2 = []

for i in range(1, int(n/2)+1):

if n % i == 0 and i not in list\_divisor\_1:

list\_divisor\_1.append(int(i))

if n/i not in list\_divisor\_1 and i != 1:

list\_divisor\_1.append(int(n/i))

pair = sum(list\_divisor\_1)

if n != pair:

for j in range(1, pair):

if pair % j == 0 and j not in list\_divisor\_2:

list\_divisor\_2.append(int(j))

if n/i not in list\_divisor\_2 and j != 1:

list\_divisor\_2.append(int(pair/j))

if sum(list\_divisor\_2) == n:

list\_amicable\_number.append(n)

list\_amicable\_number.append(pair)

print(sum(list\_amicable\_number))

Problem 22:

1.Đề bài:

Using [names.txt](https://projecteuler.net/project/resources/p022_names.txt) (right click and 'Save Link/Target As...'), a 46K text file containing over five-thousand first names, begin by sorting it into alphabetical order. Then working out the alphabetical value for each name, multiply this value by its alphabetical position in the list to obtain a name score.

For example, when the list is sorted into alphabetical order, COLIN, which is worth 3 + 15 + 12 + 9 + 14 = 53, is the 938th name in the list. So, COLIN would obtain a score of 938 × 53 = 49714.

What is the total of all the name scores in the file?

Link file names.txt: <https://projecteuler.net/project/resources/p022_names.txt>

\* Tính tổng name scores dựa vào STT từng chữ cái trong bảng chữ cái

2. Thuật toán:

- Đọc file và đưa toàn bộ tên vào list

- Xét từng tên trong list, đưa các kí tự của tên vào 1 list và tính giá trị value của kí tự đó dựa vào dictionary alphabet, sau đó update tên đó trong list ban đầu bằng giá trị, sau khi xét toàn bộ tên thì t được list mới đã trở thành list các giá trị của từng tên cũ

- Xét list ban đầu từng phần tử và nhân nó với index + 1, ta được giá trị sum cần tìm

3.Đáp án: 871198282

4.Mã nguồn:

*# Names scores*

file\_names = open("problem022\_names.txt","r")

total\_names = sorted(file\_names.read().replace('"','').split(','))

alphabet = {

"a": 1, "b": 2, "c": 3, "d": 4, "e": 5, "f": 6, "g": 7, "h": 8, "i": 9,

"j": 10, "k": 11, "l": 12, "m": 13, "n": 14, “o": 15, "p": 16, "q": 17,

"r": 18, "s": 19, "t": 20, "u": 21, "v": 22, "w": 23, "x": 24, "y": 25,

"z": 26 }

for index, item in enumerate(total\_names):

value = 0

for index1, item1 in enumerate(list(item)):

value += alphabet[item1.lower()]

total\_names[index] = value

sum = 0

for index, item in enumerate(total\_names):

sum += item \* (index+1)

print(sum)

Problem 25:

1.Đề bài:

The Fibonacci sequence is defined by the recurrence relation:

F*n* = F*n*−1 + F*n*−2, where F1 = 1 and F2 = 1.

Hence the first 12 terms will be:

F1 = 1  
F2 = 1  
F3 = 2  
F4 = 3  
F5 = 5  
F6 = 8  
F7 = 13  
F8 = 21  
F9 = 34  
F10 = 55  
F11 = 89  
F12 = 144

The 12th term, F12, is the first term to contain three digits.

What is the index of the first term in the Fibonacci sequence to contain 1000 digits?

\* Tìm số thứ tự của số Fibonacci đầu tiên chứa 1000 chữ số

2.Thuật toán:

- Thêm các số fibonacci vào dãy

- Đến khi số có giá trị vượt quá 10 mũ 999 thì dừng lại

- Số phần tử của dãy chính là số thứ tự của số fibonacci đầu tiên cần tìm

3.Đáp án: 4782

4.Mã nguồn:

*# 1000-digit Fibonacci number*

fibonacci = [1,1]

loop = True

while loop:

number = fibonacci[-1] + fibonacci[-2]

fibonacci.append(number)

if number >= 10 \*\* 999:

loop = False

print(len(fibonacci))

1. *Tuần 7 (28|29|35)*

Problem 28:

1.Đề bài:

Starting with the number 1 and moving to the right in a clockwise direction a 5 by 5 spiral is formed as follows:

**21** 22 23 24 **25**  
20  **7**  8  **9** 10  
19  6  **1**  2 11  
18  **5**  4  **3** 12  
**17** 16 15 14 **13**

It can be verified that the sum of the numbers on the diagonals is 101.

What is the sum of the numbers on the diagonals in a 1001 by 1001 spiral formed in the same way?

\* Tính tổng các số nằm trên 2 đường chéo của mảng số vuông có cạnh dài 1001 số

2.Thuật toán:

- Dùng vòng for để tính tổng tất cả các số nằm trên đường chéo, ta thấy các số sau bằng số trước đó cộng thêm độ dài cạnh trừ đi 1

3.Đáp án: 669171001

4.Mã nguồn:

*# What is the sum of the numbers on the diagonals in a 1001 by 1001 spiral formed in the same way?*

sum = 25 *# side = 3*

last\_number = 9

for i in range(3, 1000, 2):

for j in range(4):

last\_number += i + 1

sum += last\_number

print(sum)

Problem 29:

1.Đề bài:

Consider all integer combinations of *ab* for 2 ≤ *a* ≤ 5 and 2 ≤ *b* ≤ 5:

22=4, 23=8, 24=16, 25=32  
32=9, 33=27, 34=81, 35=243  
42=16, 43=64, 44=256, 45=1024  
52=25, 53=125, 54=625, 55=3125

If they are then placed in numerical order, with any repeats removed, we get the following sequence of 15 distinct terms:

4, 8, 9, 16, 25, 27, 32, 64, 81, 125, 243, 256, 625, 1024, 3125

How many distinct terms are in the sequence generated by *ab* for 2 ≤ *a* ≤ 100 and 2 ≤ *b* ≤ 100?

\* Tìm tất cả các giá trị a^b với a và b nằm trong đoạn [2, 100]

2. Thuật toán:

- Tạo ra 1 list là tập giá trị

- Dùng vòng for duyệt các giá trị a và b trong khoảng đã cho và kiểm tra nếu giá trị a^b chưa có trong tập giá trị thì ta thêm vào list sau đó hiển thị ra độ dài list ta được kết quả cần tìm

3.Đáp án: 9183

4.Mã nguồn:

*# Distinct powers*

list\_terms = []

for a in range(2, 101):

for b in range(2,101):

if a \*\* b not in list\_terms:

list\_terms.append(a\*\*b)

print(len(list\_terms))

Problem 35:

1.Đề bài:

The number, 197, is called a circular prime because all rotations of the digits: 197, 971, and 719, are themselves prime.

There are thirteen such primes below 100: 2, 3, 5, 7, 11, 13, 17, 31, 37, 71, 73, 79, and 97.

How many circular primes are there below one million?

\* Tìm tất cả các số nguyên tố circular (Khi đảo các chữ số thì vẫn là số nguyên tố) dưới 1.000.000

2.Thuật toán:

- Tìm tất cả các số nguyên tố dưới 1.000.000 (Sử dụng thuật toán tìm số nguyên tố giống Problem7 với n = 1.000.000)

- Kiểm tra xem số đó khi đảo có còn là số nguyên tố hay không

- Xóa các số không thỏa mãn ra khỏi list và hiển thị ra độ dài list khi duyệt xong, ta được kết quả cần tìm

(Chi tiết Comment trên mã nguồn)

3.Đáp án: 55

4.Mã nguồn:

*# How many circular primes are there below one million?*

from math import ceil

list\_primes = []

*# Tìm tất cả số nguyên tố dưới 1 triệu*

n = 1000000

primes = [True] \* n

primes[0] = False

primes[1] = False

roundUp = **lambda** n, prime: int(ceil(n / prime))

for currentPrime in range(2, n):

if not primes[currentPrime]:

continue

list\_primes.append(currentPrime)

for i in range(2, roundUp(n, currentPrime)):

primes[i \* currentPrime] = False

*# Kiểm tra tất cả các số nguyên tố có đảo không là số nguyên tố và thêm vào 1 list*

list\_remove = []

for number\_check in list\_primes:

if ("2" in str(number\_check)) or ("4" in str(number\_check)) or ("6" in str(number\_check)) or ("8" in str(number\_check)) or ("0" in str(number\_check)) or ("5" in str(number\_check)):

list\_remove.append(number\_check)

continue

number = str(number\_check)

for i in range(0, len(number)):

rotatedNumber = number[i:len(number)] + number[0:i]

if int(rotatedNumber) not in list\_primes:

list\_remove.append(number\_check)

*# Remove tất cả số nguyên tố đảo đi thì không là số nguyên tố ra khỏi list tất cả số nguyên tố*

for remove\_number in list\_remove:

if remove\_number in list\_primes:

list\_primes.remove(remove\_number)

*# Trong quá trình lọc thì lọc cả số 2 và 5 thỏa mãn nên ta thêm lại*

list\_primes.append(2)

list\_primes.append(5)

print(len(list\_primes))

1. *Tuần 8 (36|40|42)*

Problem 36:

1.Đề bài:

The decimal number, 585 = 10010010012 (binary), is palindromic in both bases.

Find the sum of all numbers, less than one million, which are palindromic in base 10 and base 2.

(Please note that the palindromic number, in either base, may not include leading zeros.)

\* Đếm tất cả số nhỏ hơn 1.000.000 đối xứng trên cả hệ thập phân và hệ nhị phân

2.Thuật toán:

- Kiểm tra một số có là số đối xứng hay không, nếu có thì tiếp tục kiểm tra xem số đó ở dạng nhị phân có đối xứng hay không, nếu thỏa mãn thì đếm.

3.Đáp án: 872187

4.Mã nguồn:

*# Double-base palindromes*

n = 0

for i in range(1,1000000):

    if str(i) == str(i)[::-1]:

        if bin(i)[2:] == bin(i)[2:][::-1]:

            n += i

print(n)

Problem 40:

1.Đề bài:

An irrational decimal fraction is created by concatenating the positive integers:

0.123456789101112131415161718192021...

It can be seen that the 12th digit of the fractional part is 1.

If *dn* represents the *n*th digit of the fractional part, find the value of the following expression.

*d*1 × *d*10 × *d*100 × *d*1000 × *d*10000 × *d*100000 × *d*1000000

\* Tính tích của các chữ số nằm ở các vị trí yêu cầu khi tăng dần phần thập phân

2. Thuật toán:

- Dùng vòng while để lặp thêm dần vào chuỗi các số lần lượt từ 1 tăng dần, kiểm tra đến khi nào số kí tự của chuỗi vượt quá 1000001 thì dừng lại

- Đưa chuỗi vào list và lấy ra các kí tự ở vị trí 1, 10, 100, ... theo yêu cầu và nhân lại, ta được kết quả

3.Đáp án: 210

4.Mã nguồn:

*# Champernowne's constant*

sequence = " "

loop = True

i = 1

while loop:

sequence += str(i)

i += 1

list\_check = list(sequence)

if len(list\_check) >= 1000001:

break

value = int(list\_check[1]) \* int(list\_check[100]) \* int(list\_check[1000]) \* int(list\_check[10000]) \* int(list\_check[100000]) \* int(list\_check[1000000])

print(value)

Problem 42:

1.Đề bài:

The *n*th term of the sequence of triangle numbers is given by, *tn* = ½*n*(*n*+1); so the first ten triangle numbers are:

1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...

By converting each letter in a word to a number corresponding to its alphabetical position and adding these values we form a word value. For example, the word value for SKY is 19 + 11 + 25 = 55 = *t*10. If the word value is a triangle number then we shall call the word a triangle word.

Using [words.txt](https://projecteuler.net/project/resources/p042_words.txt) (right click and 'Save Link/Target As...'), a 16K text file containing nearly two-thousand common English words, how many are triangle words?

\* Đếm tất cả các tên trong file cho sẵn, thỏa mãn giá trị của nó là 1 số triangle

2.Thuật toán:

- Xét từng tên trong file, tính giá trị tương ứng của tên đó dựa trên alphabets, kiểm tra nếu giá trị đó là triangle numbers thì đếm

3.Đáp án: 162

4.Mã nguồn:

*# Coded triangle numbers*

import math

file\_names = open("problem042\_names.txt","r")

total\_names = sorted(file\_names.read().replace('"','').split(','))

alphabet = {"a": 1, "b": 2, "c": 3, "d": 4, "e": 5, "f": 6, "g": 7, "h": 8,

"i": 9, "j": 10, "k": 11, "l": 12, "m": 13, "n": 14, "o": 15, "p": 16,

"q": 17, "r": 18, "s": 19, "t": 20, "u": 21, "v": 22, "w": 23, "x": 24,

"y": 25, "z": 26}

count = 0

for index, item in enumerate(total\_names):

value = 0

for index1, item1 in enumerate(list(item)): value += alphabet[item1.lower()]

if value \* 2 == int(math.sqrt(value \* 2)) \* (int(math.sqrt(value \* 2)) + 1):

count += 1

print(count)

1. *Tuần 9 (43|44|45)*

Problem 43:

1.Đề bài:

The number, 1406357289, is a 0 to 9 pandigital number because it is made up of each of the digits 0 to 9 in some order, but it also has a rather interesting sub-string divisibility property.

Let *d*1 be the 1st digit, *d*2 be the 2nd digit, and so on. In this way, we note the following:

* *d*2*d*3*d*4=406 is divisible by 2
* *d*3*d*4*d*5=063 is divisible by 3
* *d*4*d*5*d*6=635 is divisible by 5
* *d*5*d*6*d*7=357 is divisible by 7
* *d*6*d*7*d*8=572 is divisible by 11
* *d*7*d*8*d*9=728 is divisible by 13
* *d*8*d*9*d*10=289 is divisible by 17

Find the sum of all 0 to 9 pandigital numbers with this property.

\* Tính tổng tất cả các số là hoán vị của 10 chữ số 0-9 thỏa mãn 7 tính chất yêu cầu

2.Thuật toán:

- Đưa tất cả số hoán vị của 10 chữ số từ 0 đến 9 vào 1 list

- Duyệt từng số hoán vị, đưa vào 1 list để xét từng cụm chữ số có thỏa mãn chia hết được yêu cầu

- Nếu thỏa mãn thì cộng vào tổng, sau khi kiểm tra hết ta được giá trị cần tìm

3.Đáp án: 16695334890

4.Mã nguồn:

*# Sub-string divisibility*

from itertools import permutations

p = permutations('0123456789')

sum = 0

for i in list(p):

if (int(str(i[1])+str(i[2])+str(i[3])) % 2 == 0) and (int(str(i[2])+str(i[3])+str(i[4])) % 3 == 0) and (int(str(i[3])+str(i[4])+str(i[5])) % 5 == 0) and (int(str(i[4])+str(i[5])+str(i[6])) % 7 == 0) and (int(str(i[5])+str(i[6])+str(i[7])) % 11 == 0) and (int(str(i[6])+str(i[7])+str(i[8])) % 13 == 0) and (int(str(i[7])+str(i[8])+str(i[9])) % 17 == 0):

sum += int(''.join(i))

print(sum)

Problem 44:

1.Đề bài:

Pentagonal numbers are generated by the formula, P*n*=*n*(3*n*−1)/2. The first ten pentagonal numbers are:

1, 5, 12, 22, 35, 51, 70, 92, 117, 145, ...

It can be seen that P4 + P7 = 22 + 70 = 92 = P8. However, their difference, 70 − 22 = 48, is not pentagonal.

Find the pair of pentagonal numbers, P*j* and P*k*, for which their sum and difference are pentagonal and D = |P*k* − P*j*| is minimised; what is the value of D?

\* Tìm số pentagonal là hiệu của 2 số pentagonal và tổng 2 số đó cũng là số pentagonal

2. Thuật toán:

- Dùng vòng while duyệt lần lượt các cặp số Pentagonal

- Nếu gặp cặp số có tổng là 1 số Pentagonal và hiệu cũng là 1 số Pentagonal thì dừng lại, hiển thị ra hiệu số là giá trị cần tìm

3.Đáp án: 5482660

4.Mã nguồn:

*# Pentagon numbers*

**def** is\_pentagonal(n):

if (1+(24\*n+1)\*\*0.5) % 6 == 0: return True

return False

loop = True

i = 1

while loop:

for j in range(1, i):

a = i\*(3\*i-1)/2

b = j\*(3\*j-1)/2

if is\_pentagonal(a+b) and is\_pentagonal(a-b):

print(a-b)

loop = False

break

i += 1

Problem 45:

1.Đề bài:

Triangle, pentagonal, and hexagonal numbers are generated by the following formulae:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Triangle |  | T*n*=*n*(*n*+1)/2 |  | 1, 3, 6, 10, 15, ... |
| Pentagonal |  | P*n*=*n*(3*n*−1)/2 |  | 1, 5, 12, 22, 35, ... |
| Hexagonal |  | H*n*=*n*(2*n*−1) |  | 1, 6, 15, 28, 45, ... |

It can be verified that T285 = P165 = H143 = 40755.

Find the next triangle number that is also pentagonal and hexagonal.

\* Tìm số kế tiếp số 40755 thỏa mãn là cả 3 số Triangle, Pentagonal, Hexagonal

2.Thuật toán:

- Dùng while duyệt từ 286, khi gặp số Triangle vừa là số Pentagonal vừa là số Hexagonal thì dừng lại và in ra giá trị ta cần tìm

3.Đáp án: 1533776805

4.Mã nguồn:

*# Triangular, pentagonal, and hexagonal*

**def** is\_pentagonal(n):

if (1+(24\*n+1)\*\*0.5) % 6 == 0: return True

return False

**def** is\_hexagonal(n):

if (1+(8\*n+1)\*\*0.5) % 4 == 0: return True

return False

loop = True

i = 286

while loop:

a = i\*(i+1)/2

if is\_pentagonal(a) and is\_hexagonal(a):

print(a)

loop = False

break

i += 1

1. *Tuần 10 (48|50|52)*

Problem 48:

1.Đề bài:

The series, 11 + 22 + 33 + ... + 1010 = 10405071317.

Find the last ten digits of the series, 11 + 22 + 33 + ... + 10001000.

\* Tìm 10 chữ số cuối cùng của giá trị 11 + 22 + 33 + ... + 10001000

2.Thuật toán:

- Tính giá trị của dãy số

- Hiển thị ra 10 kí tự cuối của giá trị đó, ta được kết quả cần tìm

3.Đáp án: 9110846700

4.Mã nguồn:

*# Self powers*

sum = 0

for i in range(1,1001):

sum += i\*\*i

print(''.join(list(str(sum))[-10:-1])+list(str(sum))[-1])

Problem 50:

1.Đề bài:

The prime 41, can be written as the sum of six consecutive primes:

41 = 2 + 3 + 5 + 7 + 11 + 13

This is the longest sum of consecutive primes that adds to a prime below one-hundred.

The longest sum of consecutive primes below one-thousand that adds to a prime, contains 21 terms, and is equal to 953.

Which prime, below one-million, can be written as the sum of the most consecutive primes?

\* Tìm số nguyên tố nhỏ hơn 1 triệu là tổng của nhiều số nguyên tố nhất

2. Thuật toán:

- Tìm tất cả số nguyên tố dưới 1.000.000 bằng thuật toán tìm số nguyên tố ở Problem7 với n = 1.000.000

- Lần lượt phân tích từng số ra các số nguyên tố nhỏ hơn bằng cách trừ dần đi các số nguyên tố từ nhỏ đến lớn

- Lấy ra số nguyên tố có thể phân tích ra tổng nhiều số nguyên tố nhất và hiển thị

3.Đáp án: 997651

4.Mã nguồn:

*# Circular primes*

from math import ceil

list\_primes = []

*# Tìm tất cả số nguyên tố dưới 1 triệu*

n = 1000000

primes = [True] \* n

primes[0] = False

primes[1] = False

roundUp = **lambda** n, prime: int(ceil(n / prime))

for currentPrime in range(2, n):

if not primes[currentPrime]: continue

list\_primes.append(currentPrime)

for i in range(2, roundUp(n, currentPrime)): primes[i \* currentPrime] = False

*# Ta được list\_primes là list các số nguyên tố dưới 1 triệu*

*# Tìm số nguyên tố bằng tổng nhiều số nguyên tố nhất*

*# list\_length: List số các số nguyên tố mà 1 số nguyên tố có thể phân tích ra*

*# list\_dictionary: List các bộ giá trị, key là số nguyên tố có thể phân tích được, value là số các số nguyên tố mà key đó có thể phân tích ra*

list\_length = []

list\_dictionary = []

for number in list\_primes:

*# list\_check: List các số nguyên tố mà 1 số có thể phân tích ra*

list\_check = []

number\_real = number

for i in list\_primes:

if i <= number:

number -= i

list\_check.append(i)

elif i > number:

for j in list\_primes:

if j < i - number:

number += j

list\_check.remove(j)

elif j == i - number:

number = 0

list\_check.append(i)

list\_check.remove(j)

break

elif j > i - number: break

break

if number == 0: break

if number == 0:

list\_length.append(len(list\_check))

list\_dictionary.append({number\_real:len(list\_check)})

*# Duyệt từng bộ giá trị trong list\_dictionary, đến key có value bằng với giá trị lớn nhất trong list\_length thì dừng lại và key là số nguyên tố cần tìm*

loop = True

for index, item in enumerate(list\_dictionary):

if loop == True:

for key, value in item.items():

if value == max(list\_length):

print(key)

loop = False

break

elif loop == False: break

Problem 52:

1.Đề bài:

It can be seen that the number, 125874, and its double, 251748, contain exactly the same digits, but in a different order. Find the smallest positive integer, *x*, such that 2*x*, 3*x*, 4*x*, 5*x*, and 6*x*, contain the same digits.

\* Tìm số nguyên nhỏ nhất mà khi nhân với 2, 3, 4, 5, 6 kết quả là hoán vị của chính nó

2.Thuật toán:

- Duyệt từng số từ 1, kiểm tra lần lượt các điều kiện: số kí tự của số đó khi nhân 2, 3, 4, 5, 6 có giữ nguyên không

- Xóa từng chữ số có trong số ban đầu với ra khỏi số sau khi nhân nếu có trong đó, dùng biến check để kiểm tra, mỗi lần xóa được hết kí tự thì cộng thêm 1, nếu check bằng 5 (5 lần kiểm tra 2, 3, 4, 5, 6) thì dừng lại và hiển thị ra giá trị cần tìm

3.Đáp án: 142857

4.Mã nguồn:

*# Permuted multiples*

i = 1

loop = True

while loop:

check = 0

for j in range(2,7):

if len(str(i)) == len(str(j\*i)):

list1 = list(str(i))

list2 = list(str(j\*i))

for index, item in enumerate(list1):

if item in list2: list2.remove(item)

if len(list2) == 0: check += 1

else: i += 1

if check == 5:

print(i)

loop = False

else: i += 1

1. *Tuần 11 (53|55|56)*

Problem 53:

1.Đề bài: There are exactly ten ways of selecting three from five, 12345:

123, 124, 125, 134, 135, 145, 234, 235, 245, and 345

In combinatorics, we use the notation, 5C3 = 10. In general,

|  |  |  |
| --- | --- | --- |
| *n*C*r* = | *n*! *r*!(*n−r*)! | ,where *r* ≤ *n*, *n*! = *n*×(*n*−1)×...×3×2×1, and 0! = 1. |

It is not until *n* = 23, that a value exceeds one-million: 23C10 = 1144066.

How many, not necessarily distinct, values of  *n*C*r*, for 1 ≤ *n* ≤ 100, are greater than one-million?

\* Đếm tất cả tổ hợp với 1 ≤ *n* ≤ 100 có giá trị lớn hơn 1 triệu

2.Thuật toán:

- Viết hàm tính tổ hợp, xét giá trị n trong đoạn 23,100 (vì đến 23 mới có giá trị thỏa mãn)

- Tính tổ hợp cnr với từng giá trị n trong khoảng đã cho và r là giá trị trong đoạn 1,n, nếu tổ hợp lớn hơn 1 triệu thì đếm count thêm 1, sau khi duyệt xong ta được kết quả

3.Đáp án: 4075

4.Mã nguồn:

*# Combinatoric selections*

from math import factorial as f

count = 0

**def** ncr(n, r): return f(n)/(f(r)\*f(n-r))

for n in range(23, 101):

for r in range(1, n):

if ncr(n, r) > 1000000: count += 1

print(count)

Problem 55:

1.Đề bài:

If we take 47, reverse and add, 47 + 74 = 121, which is palindromic.

Not all numbers produce palindromes so quickly. For example,

349 + 943 = 1292,  
1292 + 2921 = 4213  
4213 + 3124 = 7337

That is, 349 took three iterations to arrive at a palindrome.

Although no one has proved it yet, it is thought that some numbers, like 196, never produce a palindrome. A number that never forms a palindrome through the reverse and add process is called a Lychrel number. Due to the theoretical nature of these numbers, and for the purpose of this problem, we shall assume that a number is Lychrel until proven otherwise. In addition you are given that for every number below ten-thousand, it will either (i) become a palindrome in less than fifty iterations, or, (ii) no one, with all the computing power that exists, has managed so far to map it to a palindrome. In fact, 10677 is the first number to be shown to require over fifty iterations before producing a palindrome: 4668731596684224866951378664 (53 iterations, 28-digits).

Surprisingly, there are palindromic numbers that are themselves Lychrel numbers; the first example is 4994.

How many Lychrel numbers are there below ten-thousand?

NOTE: Wording was modified slightly on 24 April 2007 to emphasise the theoretical nature of Lychrel numbers.

\* Đếm tất cả các số lychrel nhỏ hơn 10.000

2. Thuật toán:

- Viết hàm kiểm tra số lychrel (Đảo ngược 1 số cộng lại và kiểm tra kết quả có phải số đối xứng hay không, được lặp lại tối đa 50 lần nếu không thể được số đối xứng thì chính là số lycherel)

- Duyệt các số nhỏ hơn 10001, số nào là số lychrel thì count cộng thêm 1

- Sau khi duyệt xong thì count chính là số các số lychrel không lớn hơn 10000 là kết quả cần tìm

3.Đáp án: 249

4.Mã nguồn:

*# Lychrel numbers*

**def** is\_lychrel(n):

for i in range(50):

number = n + int(str(n)[::-1])

if str(number) == str(number)[::-1]:

return False

n = number

return True

count = 0

for i in range(10001):

if is\_lychrel(i):

count += 1

print(count)

Problem 56:

1.Đề bài:

A googol (10100) is a massive number: one followed by one-hundred zeros; 100100 is almost unimaginably large: one followed by two-hundred zeros. Despite their size, the sum of the digits in each number is only 1.

Considering natural numbers of the form, *ab*, where *a, b* < 100, what is the maximum digital sum?

\* Với a, b < 100, tìm số *ab* có tổng các chữ số lớn nhất

2.Thuật toán:

- Viết hàm tính tổng các chữ số của một số

- Xét a, b < 100, tính *ab* và tính tổng các chứ số của giá trị đó, nếu tổng lớn hơn giá trị largest lớn nhất hiện tại thì gán largest bằng giá trị đó

- Gía trị largest sau khi xét xong chính là kết quả cần tìm

3.Đáp án: 972

4.Mã nguồn:

*# Powerful digit sum*

**def** sum\_of\_digits(n):

sod = 0

while n != 0:

sod += n % 10

n //= 10

return sod

largest = 0

for a in range(0, 100):

for b in range(0, 100):

sod = sum\_of\_digits(a\*\*b)

if sod > largest:

largest = sod

print(largest)

1. *Tuần 12 (57|58|62)*

Problem 57:

1.Đề bài:

It is possible to show that the square root of two can be expressed as an infinite continued fraction.

√ 2 = 1 + 1/(2 + 1/(2 + 1/(2 + ... ))) = 1.414213...

By expanding this for the first four iterations, we get:

1 + 1/2 = 3/2 = 1.5  
1 + 1/(2 + 1/2) = 7/5 = 1.4  
1 + 1/(2 + 1/(2 + 1/2)) = 17/12 = 1.41666...  
1 + 1/(2 + 1/(2 + 1/(2 + 1/2))) = 41/29 = 1.41379...

The next three expansions are 99/70, 239/169, and 577/408, but the eighth expansion, 1393/985, is the first example where the number of digits in the numerator exceeds the number of digits in the denominator.

In the first one-thousand expansions, how many fractions contain a numerator with more digits than denominator?

\* Đếm tất cả phân số mở rộng theo quy tắc thỏa mãn số chữ số ở tử lớn hơn mẫu

2.Thuật toán:

- Mở rộng đến 1000 lần theo công thức, tử số số sau bằng tử số cộng mẫu số số trước, mẫu số số sau bằng tử số cộng hai lần mẫu số số trước

- Nếu số nào có số kí tự ở tử lớn hơn mẫu thì count tăng thêm 1

- Sau khi chạy mở rộng hết, count là kết quả cần tìm

3.Đáp án: 153

4.Mã nguồn:

*# Square root convergents*

p = 1

q = 1

count = 0

for i in range(1000):

a1 = p + 2\*q

b1 = p + q

if len(str(a1)) > len(str(b1)):

count += 1

p = a1

q = b1

print(count)

Problem 58:

1.Đề bài:

Starting with 1 and spiralling anticlockwise in the following way, a square spiral with side length 7 is formed.

**37** 36 35 34 33 32 **31**  
38 **17** 16 15 14 **13** 30  
39 18  **5**  4  **3** 12 29  
40 19  6  1  2 11 28  
41 20  **7**  8  9 10 27  
42 21 22 23 24 25 26  
**43** 44 45 46 47 48 49

It is interesting to note that the odd squares lie along the bottom right diagonal, but what is more interesting is that 8 out of the 13 numbers lying along both diagonals are prime; that is, a ratio of 8/13 ≈ 62%.

If one complete new layer is wrapped around the spiral above, a square spiral with side length 9 will be formed. If this process is continued, what is the side length of the square spiral for which the ratio of primes along both diagonals first falls below 10%?

\* Tìm số các số nằm trên 1 cạnh của mảng số vuông thỏa mãn tỉ lệ số các số nguyên tố nằm trên hai đường chéo xuống nhỏ hơn 10%

2. Thuật toán:

- Tìm các số nguyên tố dưới 1 triệu với thuật toán tìm số nguyên tố ở Problem7 với n = 1.000.000

- Tăng dần độ dài cạnh của bộ số, thêm tất cả các giá trị trên đường chéo vào 1 list (list\_number), nếu giá trị nào là số nguyên tố thì thêm vào 1 list khác (list\_prime)

- Đến khi tỉ lệ số nguyên tố chia tất cả số nhỏ hơn 10% thì dừng vòng lặp và hiển thị ra độ dài cạnh là kết quả cần tìm

3.Đáp án: 26241

4.Mã nguồn:

*# Spiral primes*

from math import ceil

*# Tìm tất cả số nguyên tố dưới 1 triệu*

list\_all\_primes = []

n = 1000000000

primes = [True] \* n

primes[0] = False

primes[1] = False

roundUp = **lambda** n, prime: int(ceil(n / prime))

for currentPrime in range(2, n):

if not primes[currentPrime]: continue

list\_all\_primes.append(currentPrime)

for i in range(2, roundUp(n, currentPrime)):

primes[i \* currentPrime] = False

*# Xét*

side = 3

loop = True

value = 1

list\_number = [1]

list\_prime = []

while loop:

side\_run = side - 1

for i in range(4):

for k in range(side\_run):

value += 1

list\_number.append(value)

if value in list\_all\_primes: list\_prime.append(value)

if (float(len(list\_prime))/len(list\_number)) <= 0.1:

print(side)

break

side += 2

Problem 62:

1.Đề bài:

The cube, 41063625 (3453), can be permuted to produce two other cubes: 56623104 (3843) and 66430125 (4053). In fact, 41063625 is the smallest cube which has exactly three permutations of its digits which are also cube.

Find the smallest cube for which exactly five permutations of its digits are cube.

\* Tìm số nhỏ nhất có chính xác 5 hoán vị là 1 số lập phương

2.Thuật toán:

- Lần lượt tính lập phương các số từ 1, sắp xếp các chữ số của giá trị sau khi được lập phương theo thứ tự từ nhỏ đến lớn và cho vào 1 list

- Đến khi trong list xuất hiện 5 bộ số trùng nhau (Đã có 5 hoán vị) thì in ra index của số đầu tiên đc lập phương và lập phương lên, ta được kết quả cần tìm

3.Đáp án: 127035954683

4.Mã nguồn:

*# Cubic permutations*

cubes = []

i = 0

while True:

cube = sorted(list(str(i\*\*3)))

cubes.append(cube)

if cubes.count(cube) == 5:

print((cubes.index(cube))\*\*3)

break

i += 1

1. *Tuần 13 (67|18|97)*

Problem 67:

1.Đề bài:

By starting at the top of the triangle below and moving to adjacent numbers on the row below, the maximum total from top to bottom is 23.

**3**  
**7** 4  
2 **4** 6  
8 5 **9** 3

That is, 3 + 7 + 4 + 9 = 23.

Find the maximum total from top to bottom in [triangle.txt](https://projecteuler.net/project/resources/p067_triangle.txt) (right click and 'Save Link/Target As...'), a 15K text file containing a triangle with one-hundred rows.

**NOTE:** This is a much more difficult version of [Problem 18](https://projecteuler.net/problem=18). It is not possible to try every route to solve this problem, as there are 299 altogether! If you could check one trillion (1012) routes every second it would take over twenty billion years to check them all. There is an efficient algorithm to solve it. ;o)

\* Xuất phát từ đỉnh, tìm đường đi tạo thành từ các số đến đáy tháp có tổng các số lớn nhất

2.Thuật toán:

- Chia mỗi dòng thành list riêng và tách từng số trong dòng thành số riêng

- Duyệt mảng từ dưới đi lên, thay thế dần mỗi số bằng tổng của nó với số bên dưới lớn hơn

- Khi duyệt xong thì mảng chỉ còn 1 số là đường đi lớn nhất cần tìm

3.Đáp án: 7273

4.Mã nguồn:

*# Maximum path sum II*

file = open("problem067.txt","r")

*#Chia mỗi dòng thành 1 chuỗi nhỏ*

number = file.read().strip().split('\n')

*#Tiếp tục tách mỗi số trong chuỗi nhỏ ra và ép kiểu thành int, chuyển các chuỗi nhỏ thành list*

for i in range(0,len(number)):

    number[i] = number[i].split(' ')

    number[i] = [int(x) for x in number[i]]

*#Tính các đường đi từ chân tháp*

for i in range(len(number)-2,-1,-1):

    for j in range(len(number[i])):

        number[i][j] = number[i][j] + max(number[i+1][j], number[i+1][j+1])

print(number[0][0])

Problem 18:

1.Đề bài:

By starting at the top of the triangle below and moving to adjacent numbers on the row below, the maximum total from top to bottom is 23.

**3**  
**7** 4  
2 **4** 6  
8 5 **9** 3

That is, 3 + 7 + 4 + 9 = 23. Find the maximum total from top to bottom of the triangle below:

75  
95 64  
17 47 82  
18 35 87 10  
20 04 82 47 65  
19 01 23 75 03 34  
88 02 77 73 07 63 67  
99 65 04 28 06 16 70 92  
41 41 26 56 83 40 80 70 33  
41 48 72 33 47 32 37 16 94 29  
53 71 44 65 25 43 91 52 97 51 14  
70 11 33 28 77 73 17 78 39 68 17 57  
91 71 52 38 17 14 91 43 58 50 27 29 48  
63 66 04 68 89 53 67 30 73 16 69 87 40 31  
04 62 98 27 23 09 70 98 73 93 38 53 60 04 23

**NOTE:** As there are only 16384 routes, it is possible to solve this problem by trying every route. However, [Problem 67](https://projecteuler.net/problem=67), is the same challenge with a triangle containing one-hundred rows; it cannot be solved by brute force, and requires a clever method! ;o)

\* Xuất phát từ đỉnh, tìm đường đi tạo thành từ các số đến đáy tháp có tổng các số lớn nhất

2. Thuật toán:

- Chia mỗi dòng thành list riêng và tách từng số trong dòng thành số riêng

- Duyệt mảng từ dưới đi lên, thay thế dần mỗi số bằng tổng của nó với số bên dưới lớn hơn

- Khi duyệt xong thì mảng chỉ còn 1 số là đường đi lớn nhất cần tìm

3.Đáp án: 1074

4.Mã nguồn:

*# Maximum path sum I*

number = '''75

95 64

17 47 82

18 35 87 10

20 04 82 47 65

19 01 23 75 03 34

88 02 77 73 07 63 67

99 65 04 28 06 16 70 92

41 41 26 56 83 40 80 70 33

41 48 72 33 47 32 37 16 94 29

53 71 44 65 25 43 91 52 97 51 14

70 11 33 28 77 73 17 78 39 68 17 57

91 71 52 38 17 14 91 43 58 50 27 29 48

63 66 04 68 89 53 67 30 73 16 69 87 40 31

04 62 98 27 23 09 70 98 73 93 38 53 60 04 23'''

*#Chia mỗi dòng thành 1 chuỗi nhỏ*

number = number.split('\n')

*#Tiếp tục tách mỗi số trong chuỗi nhỏ ra và ép kiểu thành int, chuyển các chuỗi nhỏ thành list*

for i in range(0,len(number)):

    number[i] = number[i].split(' ')

    number[i] = [int(x) for x in number[i]]

*#Tính các đường đi từ chân tháp*

for i in range(len(number)-2,-1,-1):

    for j in range(len(number[i])):

        number[i][j] = number[i][j] + max(number[i+1][j], number[i+1][j+1])

print(number[0][0])

Problem 97:

1.Đề bài:

The first known prime found to exceed one million digits was discovered in 1999, and is a Mersenne prime of the form 26972593−1; it contains exactly 2,098,960 digits. Subsequently other Mersenne primes, of the form 2*p*−1, have been found which contain more digits.

However, in 2004 there was found a massive non-Mersenne prime which contains 2,357,207 digits: 28433×27830457+1. Find the last ten digits of this prime number.

\* Tìm 10 chữ số cuối cùng của số nguyên tố chứa 2.357.207 chữ số: 28433×27830457+1.

2.Thuật toán:

- Lần lượt tính các phép lũy thừa, nhân và cộng nhưng sau mỗi phép tính đều lấy phần dư khi chia cho 10 tỉ để chỉ lấy ra 10 số cuối cùng

3.Đáp án: 8739992577

4.Mã nguồn:

*# Large non-Mersenne prime*

n = 2

for i in range(7830456): n = (2 \* n) % 10000000000

n = n \* 28433 + 1

n = n % 10000000000

print(n)

1. *Tuần 14 (12|23|24)*

Problem 12:

1.Đề bài:

The sequence of triangle numbers is generated by adding the natural numbers. So the 7th triangle number would be 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28. The first ten terms would be:

1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...

Let us list the factors of the first seven triangle numbers:

**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

We can see that 28 is the first triangle number to have over five divisors.

What is the value of the first triangle number to have over five hundred divisors?

\* Tìm số triangle đầu tiên có hơn 500 ước

2.Thuật toán:

- Viết hàm tính số ước của một số: Duyệt đến giá trị căn bậc 2 và thêm các ước vào 1 list

- Kiểm tra các số từ 1 với điều kiện là số tam giác, đến khi có số tam giác có nhiều hơn 500 ước thì dừng lại và trả ra số đó

3.Đáp án: 76576500

4.Mã nguồn:

*# Highly divisible triangular number*

import math

**def** number\_of\_divisor(n):

nod = 0

for i in range(1, math.ceil(math.sqrt(n))):

if n % i == 0:

nod += 2

elif i \*\* 2 == n:

nod -= 1

return nod

i = 1

while True:

tn = (i \* (i + 1) / 2) *# tn: triangle number*

if number\_of\_divisor(tn) > 500:

print(tn)

break

i += 1

Problem 23:

1.Đề bài:

A perfect number is a number for which the sum of its proper divisors is exactly equal to the number. For example, the sum of the proper divisors of 28 would be 1 + 2 + 4 + 7 + 14 = 28, which means that 28 is a perfect number.

A number *n* is called deficient if the sum of its proper divisors is less than *n* and it is called abundant if this sum exceeds *n*.

As 12 is the smallest abundant number, 1 + 2 + 3 + 4 + 6 = 16, the smallest number that can be written as the sum of two abundant numbers is 24. By mathematical analysis, it can be shown that all integers greater than 28123 can be written as the sum of two abundant numbers. However, this upper limit cannot be reduced any further by analysis even though it is known that the greatest number that cannot be expressed as the sum of two abundant numbers is less than this limit.

Find the sum of all the positive integers which cannot be written as the sum of two abundant numbers.

\* Tính tổng của tất cả các số nguyên dương không thể được viết thành tổng của 2 số abundant

2. Thuật toán:

- Viết hàm kiểm tra 1 số là số Abundant: Tổng các ước số lớn hơn chính nó

- Thêm tất cả các số Abundant nhỏ hơn 28123 vào 1 list

- Tạo ra 1 list các số trong khoảng 1 đến 28123, tính tổng 2 số abundant bất kì trong list số abundant và nếu số đó có trong list số tạo ra thì cho bằng 0

- Tính tổng list số đã tạo ra, ta được kết quả cần tìm

3.Đáp án: 4179871

4.Mã nguồn:

*# Non-abundant sums*

from math import sqrt

**def** isAbundant(n):

list\_divisors = []

for i in range(2, int(sqrt(n)) + 1):

if n % i == 0:

list\_divisors.append(i)

list\_divisors.append(n/i)

if i \*\* 2 == n:

list\_divisors.remove(i)

if sum(list\_divisors) > n:

return True

else:

return False

list\_abundant = []

for i in range(1, 28123):

if isAbundant(i):

list\_abundant.append(i)

list\_sum = [x for x in range(28123)]

for i in range(len(list\_abundant)):

for j in range(i, len(list\_abundant)):

if list\_abundant[i] + list\_abundant[j] < 28123:

list\_sum[list\_abundant[i] + list\_abundant[j]] = 0

print(sum(list\_sum))

Problem 24:

1.Đề bài:

A permutation is an ordered arrangement of objects. For example, 3124 is one possible permutation of the digits 1, 2, 3 and 4. If all of the permutations are listed numerically or alphabetically, we call it lexicographic order. The lexicographic permutations of 0, 1 and 2 are:

012   021   102   120   201   210

What is the millionth lexicographic permutation of the digits 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9?

\* Tìm hoán vị thứ 1 triệu của 10 chữ số 0 đến 9

2.Thuật toán:

- Tạo ra list chứa tất cả hoán vị của 10 chữ số từ 0 đến 9

- Lấy ra giá trị thứ 1 triệu chính là kết quả

3.Đáp án: 2783915460

4.Mã nguồn:

*# Lexicographic permutations*

import itertools

digits = ['0','1','2','3','4','5','6','7','8','9']

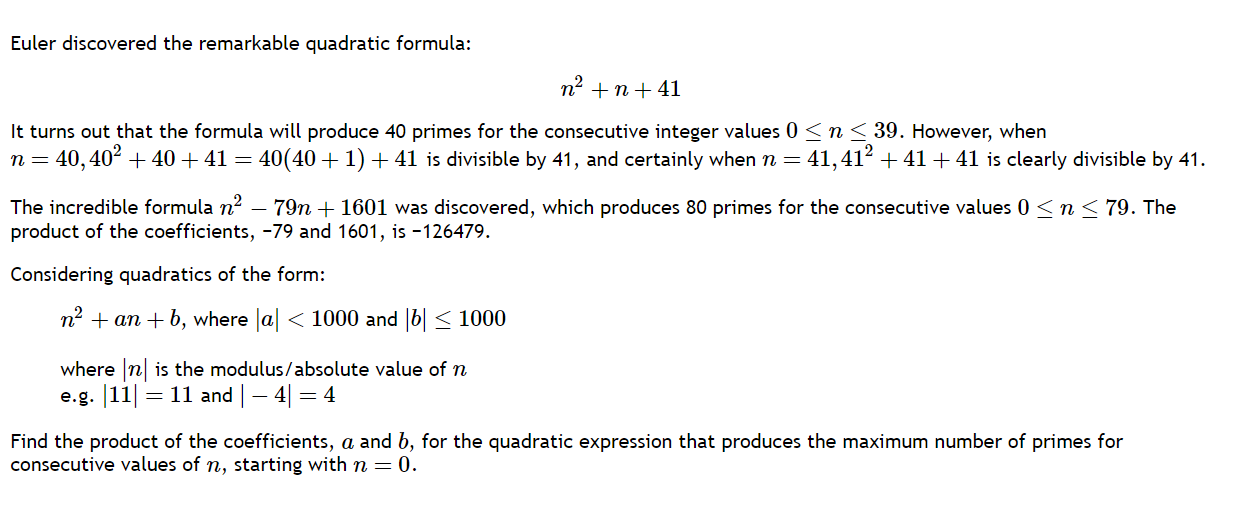
permutations = list(itertools.permutations(digits))

print (''.join(permutations[999999]))

1. *Tuần 15 (27|33|39)*

Problem 27:

1.Đề bài:



\* Tìm ra bộ số a, b thỏa mãn nằm trong khoảng và kết hợp thành 1 biểu thức tạo ra nhiều số nguyên tố liên tiếp nhất có thể, tính tích của 2 số đó

2.Thuật toán:

- Duyệt tất cả các bộ số a, b trong khoảng -1000, 1000

- Với mỗi bộ số kiểm tra xem biểu thức sẽ có giá trị nguyên tố ứng với bao nhiêu giá trị của n

- Bộ số có nhiều giá trị nguyên tố nhất là kết quả cần tìm

3.Đáp án: -59231

4.Mã nguồn:

*# Quadratic primes*

from math import sqrt

v\_a = 0

v\_b = 0

nop = 0

**def** isPrime(n):

for i in range(2, int(sqrt(n))+1):

if n % i == 0:

return False

return True

for a in range(-1000, 1001):

for b in range(-1000, 1001):

n = 0

while ((n \*\* 2 + a \* n + b) > 0) and isPrime(n \*\* 2 + a \* n + b):

n += 1

if n > nop:

nop = n

v\_a = a

v\_b = b

print(v\_a, v\_b)

print(v\_a\*v\_b)

Problem 33:

1.Đề bài:

The fraction 49/98 is a curious fraction, as an inexperienced mathematician in attempting to simplify it may incorrectly believe that 49/98 = 4/8, which is correct, is obtained by cancelling the 9s.

We shall consider fractions like, 30/50 = 3/5, to be trivial examples.

There are exactly four non-trivial examples of this type of fraction, less than one in value, and containing two digits in the numerator and denominator.

If the product of these four fractions is given in its lowest common terms, find the value of the denominator.

\* Có chính xác 4 phân số có 2 chữ số ở cả tử và mẫu, trong đó có 1 chữ số giống nhau thỏa mãn rằng khi bỏ đi chữ số giống nhau đó thì giá trị phân số không thay đổi. Hãy nhân 4 phân số với nhau, giản ước và giá trị mẫu số là bao nhiêu?

2. Thuật toán:

- Tìm tất cả các phân số thỏa mãn bằng cách duyệt tử và mẫu từ 10 đến 100, xóa 1 kí tự giống nhau ở cả tử và mẫu số và so sánh với giá trị phân số ban đầu

- Sau khi tìm ra 4 phân số thỏa mãn, nhân với nhau và được mẫu số của giá trị là kết quả cần tính

3.Đáp án: 100

4.Mã nguồn:

*# Digit cancelling fractions*

list\_result = []

for a in range(10,100):

for b in range(10,100):

if a != b and a % 10 != 0 and b % 10 != 0 and a/b < 1:

list\_a = list(str(a))

list\_b = list(str(b))

for digit in list\_a:

if digit in list\_b:

list\_a.remove(digit)

list\_b.remove(digit)

if int(list\_b[0]) != 0 and a/b == int(list\_a[0]) / int(list\_b[0]):

list\_result.append([int(list\_a[0]), int(list\_b[0])])

product = 1

for i in range(4):

product \*= list\_result[i][1]

product /= list\_result[i][0]

print(product)

Problem 39:

1.Đề bài:

If *p* is the perimeter of a right angle triangle with integral length sides, {*a*,*b*,*c*}, there are exactly three solutions for *p* = 120.

{20,48,52}, {24,45,51}, {30,40,50}

For which value of *p* ≤ 1000, is the number of solutions maximised?

\* Tìm giá trị chu vi của một tam giá vuông *p* ≤ 1000 có thể phù hợp với nhiều bộ giá trị của 3 cạnh nhất có thể

2.Thuật toán:

- Vì là tam giác vuông nên ta thay c = p – a – b vào biểu thức a^2 + b^2 = c^2 và rút ra b tính theo p và a

- Duyệt chu vi(p) trong khoảng đề bài yêu cầu(1-1000) và duyệt a với giá trị tối đa bằng nửa chu vi (Vì a < c)

- Xét biểu thức giá trị b theo p và a, nếu b là số tự nhiên thì ta được 1 bộ giá trị cạnh, nếu số bộ giá trị của 1 chu vi là lớn nhất thì hiển thị ra

3.Đáp án: 840

4.Mã nguồn:

*# Integer right triangles*

t\_max = 0

p\_max = 0

for p in range(0, 1001, 2):

t = 0

for a in range(1, p/2):

*# b = p\*(p - 2\*a)/(2\*(p - a))*

if p\*(p - 2\*a) % (2\*(p - a)) == 0:

t += 1

if t >= t\_max:

t\_max = t

p\_max = p

print(p\_max)

*TÀI LIỆU THAM KHẢO*

Tài liệu có tham khảo thuật toán của một vài trang Website:

<https://radiusofcircle.blogspot.com/>

<http://code.jasonbhill.com/>

<https://stackoverflow.com/>

<https://ramonsmathsblog.wordpress.com/>

<https://blog.dreamshire.com/>