Euler Project

@GaZ3ll3[1]

March 29, 2015

[1] www.github.com/gaz3ll3

Contents

1

LEVEL 0 @Algorithms

"An algorithm must be seen to be believed."

– Donald Knuth

2

LEVEL 1 @Problem 1 - 25

"Not brute force but only persuasion and faith are the kings of this world."

– Thomas Carlyle

2.1 Problem 001 - Multiples of 3 and 5

Problem 1.
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.

Solution. Brute force is simple.

```
#include <stdio.h>

void main(){
    int s = 0;
    for (int i = 1; i < 1000; i++){
        if (!(i % 3) || !(i % 5)) {
             s += i;
        }
    }
    printf("%d\n", s);
}
```

If using *Including-Excluding Principle*, then denote $S_{k_1,k_2,\ldots,k_n}$ as the sum of common multiples of $k_1, k_2, \ldots, k_n$.

$$S = S_3 + S_5 - S_{3,5} = S_3 + S_5 - S_{15} \tag{2.1}$$

where S_k can be calculated with $k\lfloor \frac{n}{k} \rfloor (1 + \lfloor \frac{n}{k} \rfloor)/2$.

2.2 Problem 002 - Even Fibonacci numbers

Problem 2.
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:

$$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

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

Solution. It is simple to observe that the F_{3k} is even. And we are looking for the summation $\sum_k^n F_{3k}$ over k such that $F_{3n} \leq N$. However, the following code computes the even numbers sequentially, time complexity $O(n)$, if there are n elements before the loop ends.

```
#include <stdio.h>

void main(){

    int s = 0;
    int x = 1, y = 1, z = 2;
    while (z <= 4000000){
        s +=z;
        x = y + z;
        y = z + x;
        z = x + y;
    }
    printf("%d\n", s);

}
```

To make this process faster, we can use the formula

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right) \tag{2.2}$$

Which will be at cost of $O(\log n)$ complexity.

2.3 Problem 003 - Largest prime factor

Problem 3.
The prime factors of 13195 are 5, 7, 13 and 29. What is the largest prime factor of the number 600851475143 ?

Solution. Brute force is easy to implement, since the number overflows `int`, we use `long long int`.

```
#include <stdio.h>
#include <stdbool.h>

/*
 * verify prime number
 */
bool isprime(long long n){
    if (n == 2) {
        return true;
```

```
        }

        if (n > 3) {
            if (!(n % 2)) {
                return false;
            }
        }

        for (long long j = 3; j * j <= n; j += 2){
            if (!(n % j)) return false;
        }
        return true;
    }

    void main(){

        long long m = 600851475143;
        long long n = 0;
        long long k = 0;
        /*
         * after locating one prime factor p, we continue with n/p.
         */
        for (long long j = 3; j * j <= m; j += 2){
            if (!(m % j)) {
                k = m / j;
                if (isprime(j) && n < j) {n = j; m /= j;}
                if (isprime(k) && n < k) {n = k; m /= k;}
            }
        }
        printf("%lld\n", n);

    }
```

2.4 Problem 004 - Largest palindrome product

Problem 4.
A palindromic number reads the same both ways. The largest palindrome made from the product of two 2-digit numbers is $9009 = 91 \times 99$. *Find the largest palindrome made from the product of two 3-digit numbers.*

Solution. The palindrome should be written as

$$\overline{abccba} = 100001a + 10010b + 1100c = 11(9091a + 910b + 100c) \tag{2.3}$$

Now we can brute force the problem with one number divisible by 11.

```
#include <stdio.h>
#include <stdbool.h>

bool ispalindrome(int n){
    int m = n;
    int r = 0;
    int d = 0;
    while (m > 0) {
```

```
        d = m % 10;
        r = r * 10 + d;
        m = m / 10;
    }

    if (r == n) return true;
    return false;
}


void main(){
    int r = 0;
    int max = 0;
    for (int p = 999; p > 99; p--){
        for (int q = 990; q > 99; q -= 11){
            r = p * q;
            if (ispalindrome(r)) {
                if (r > max) {
                    max  =r;
                }
            }
        }
    }
    printf("%d\n", max);
}
```

2.5 Problem 005 - Smallest multiple

Problem 5.
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?

Solution. LCM(least common multiple) is as easy as GCD(greatest common divider) by

$$(a, b)[a, b] = ab \tag{2.4}$$

We just iteratively calculate the LCM from 1 to 20.

```
#include <stdio.h>

typedef long long int64;

int64 gcd(int64 a, int64 b){
    int64 r = a;
    int64 s = b;
    int64 tmp;
    while (s != 0) {
        tmp = r;
        r = s;
        s =  tmp % s;
    }
    return r;
}
```

```
int64 lcm(int64 a, int64 b){
    return a * b / (gcd(a, b));
}

void main() {
    int64 n = 1;
    for (int64 i = 2; i < 21; i++){
        n = lcm(n , i);
    }
    printf("%lld\n", n);
}
```

2.6 Problem 006 - Sum square difference

Problem 6.
The sum of the squares of the first ten natural numbers is,

$$1^2 + 2^2 + ... + 10^2 = 385$$

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.*

Solution. Observe that

$$(\sum_{i=1}^{n} i)^2 - \sum_{i=1}^{n} i^2 = \frac{1}{4}n^2(n+1)^2 - \frac{1}{6}n(n+1)(2n+1) = \frac{n(n+1)(3n^2-n-2)}{12}$$

2.7 Problem 007 - 10001st prime

Problem 7.
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?

Solution. We can use *Sieve algorithm* to calculate the prime numbers below a given number N, and the estimated $p_n \sim n \log n$, thus $p_{10,001} \sim 115141$, we take $N = 200000$ for calculation.

```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

#define N 200000L

typedef long long int64;
```

```
bool isprime[N];

void sieve(){
    for (int64 k = 2; k < sqrt(N); k++){
        if (isprime[k]){
            for (int64 j = k * k; j <= N; j += k){
                isprime[j] = false;
            }
        }
    }
}

void main(){
    for (int64 i = 0; i < N; i++) {
        isprime[i] = true;
    }
    sieve();
    int l = 0;
    for (int64 j = 2; j < N; j++){
        if (isprime[j]) l = l + 1;
        if (l == 10001) {
            printf("%lld\n", j);
            return;
        }
    }
}
```

2.8 Problem 008 - Largest product in a series

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

```
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?

Solution. Brute force.

```
#include <stdio.h>

typedef long long int64;

void main(){

    char digits[1000];
    sprintf(digits,
    "7316717653133062491922511967442657474235534919493496"
    "9698352031277450632623957831801698480186947885184385"
    "8586156078911294949545950173795833195285320880551115"
    "1254069874715852386305071569329096329522744304355766"
    "6689664895044524452316173185640309871112172238311354"
    "6222989342338030813533627661428280644448664523874904"
    "3035890729629049156044077239071381051585930796086670"
    "7017242712188399879790879227492190169972088809377665"
    "6572733300105336788122023542180975125454059475224352"
    "5258490771167055601360483958644670632441572215539753"
    "5369781797784617406495514929086256932197846862248290"
    "8397224137565705605749026140797296865241453510047482"
    "8216637048440319989000889524345065854122758866688116"
    "1642717147992444292823086346567481391912316282458617"
    "1786645835912456652947654568284891288314260769004224"
    "2421902267105562632111110937054421750694165896040807"
    "0719840385096245544436298123098787992724428490918819"
    "8458015616609791913387549920052406368991256071760605"
    "0588611646710940507754100225698315520005593572972571"
    "6163626956188267042825248360082325753042075296345050");

    int64 max = 0, n;
    for (int64 i = 0; i < 988; i++){
        n = 1;
        for (int64 j = 0; j < 13; j++) {
            n *= (digits[i + j] - '0');
        }
        if (n > max) max = n;
    }
    printf("%lld\n", max);

}
```

2.9 Problem 009 - Special Pythagorean triplet

Problem 9.
A Pythagorean triplet is a set of three natural numbers, $a < b < c$, for which,

$$a^2 + b^2 = c^2$$

For example, $3^2 + 4^2 = 9 + 16 = 25 = 5^2$.

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

Solution. It is known that Pythagorean triplet takes form of

$$(m^2 - n^2, 2mn, m^2 + n^2)$$

therefore, $2m^2 + 2mn = 1000$ only has one solution such that $m > n$.

$$m^2 + m < m(m + n) = 500 < 2m^2 \tag{2.5}$$

then $23 > m > 15$, there is only one solution $m = 20, n = 5$.

2.10 Problem 010 - Summation of primes

Problem 10.
The sum of the primes below 10 is 2 + 3 + 5 + 7 = 17. Find the sum of all the primes below two million.

Solution. Brute force. Use sieve algorithm to find all the prime numbers, and sum them up.

```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

#define N 2000000L

typedef long long int64;

bool isprime[N];

void sieve(){
    for (int64 k = 2; k < sqrt(N); k++){
        if (isprime[k]){
            for (int64 j = k * k; j <= N; j += k){
                isprime[j] = false;
            }
        }
    }
}

void main(){
    for (int64 i = 0; i < N; i++) {
        isprime[i] = true;
    }
    sieve();
    int64 l = 0;
    for (int64 j = 2; j < N; j++){
        if (isprime[j]) l += j;
    }

    printf("%lld\n", l);
}
```

2.11 Problem 011 - Largest product in a grid

Problem 11.
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 $26 \times 63 \times 78 \times 14 = 1788696$.

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

Solution. Brute force.

```
#include <stdio.h>

void main(){

        int A[20][20];

        FILE* file = fopen("../data/011.txt", "r");

        for (int i = 0; i < 20; i++) {
                for (int j = 0; j < 20; j++){
                        fscanf(file, "%d", &A[i][j]);
                }
        }

    int res = 0;
    int V = 0, H= 0, L = 0, R= 0;
    for (int i = 0; i < 20; i++) {
        for (int j = 0; j < 20; j++) {
            V = 0; H = 0; L = 0; R = 0;
            if (i <= 16) V = A[i][j]*A[i+1][j]*A[i+2][j]*A[i+3][j];
            if (j <= 16) H = A[i][j]*A[i][j+1]*A[i][j+2]*A[i][j+3];
```

```
                        if (i<=16 && j <=16) {
                                L = A[i][j]*A[i+1][j+1]*A[i+2][j
                                    +2]*A[i+3][j+3];
                                R = A[i+3][16 - j]*A[i+2][17 - j] *
                                     A[i+1][18 - j]*A[i][19 - j];
                        }
                        if (V > res) {res = V;}
                        if (H > res) {res = H;}
                        if (L > res) {res = L;}
                        if (R > res) {res = R;}
            }
        }
            printf("%d\n", res);
    }
```

2.12 Problem 012 - Highly divisible triangular number

Problem 12.

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?

Solution. Brute force.

```
#include <stdio.h>
#include <limits.h>
#define LIMIT 500

typedef long long int64;

int num_of_factor(int64 n){
    int64 k;
    int count = 2;
    for (int64 k = 2; k < n ; k++){
        if (n % k == 0){
            count++;
        }
    }
    return count;
}

void main(){
    int64 n = 0;
    int count = 0;

    for (int64 i = 1; i < LLONG_MAX - 1; i++){
        n += i;
        if (i & 1) {
            count = num_of_factor(i) * num_of_factor((i + 1)/2);
        }
        else{
            count = num_of_factor(i/2) * num_of_factor(i + 1);
        }

        if (count > LIMIT) {
            printf("%lld\n", n);
```

```
                return;
            }
        }
    }
```

2.13 Problem 013 - Large sum

Problem 13.
Work out the first ten digits of the sum of the following one-hundred 50-digit numbers. See data here.

Solution. The quick way is to use double type to save the numbers as $\overline{a.bcdefgh\ldots}$, though machine error will only give limited accuracy around 10^{-15}, but it is more than enough, when converting back into integer, be careful of possible overflow.

```
#include <stdio.h>

double mapping(char* c){

    double res = 0.;
    double d = 1.;
    for (int i = 0; i < 50; i++){
        res += d * (c[i] - '0');
        d /= 10.;
    }
    return res;
}

void main(){

    char digits[51];
    double res = 0.;
    FILE* file = fopen("../data/013.txt", "r");

    while (fgets(digits, 52, file)) {
        res += mapping(digits);
    }

    printf("%lld\n", (long long)(res * 10000000));

}
```

2.14 Problem 014 - Longest Collatz sequence

Problem 14.

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

$n \to n/2$ (n is even)
$n \to 3n + 1$ (n is odd)

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

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

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.

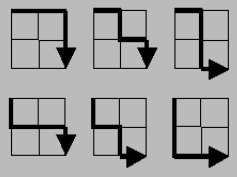
Solution. Brute force. Calculate the length of sequence for each start.

```
#include <stdio.h>

void main() {
    int longest = 0, terms = 0, i;
    unsigned long j;

    for (i = 1; i <= 1000000; i++) {
        j = i; int this_terms = 1;
        while (j != 1) {
            this_terms++;
            if (this_terms > terms) {
                terms = this_terms; longest = i;
            }
            if (j % 2 == 0) {
                j = j / 2;
            } else {
                j = 3 * j + 1;
            }
        }
    }
    printf("%d\n", longest);
}
```

2.15 Problem 015 - Lattice paths

Problem 15.

Starting in the top left corner of a 2×2 grid, and only being able to move to the right and down, there are exactly 6 routes to the bottom right corner.



How many such routes are there through a 20×20 grid?

Solution. Suppose $f(m,n)$ represents the number of path for lattice m $timesn$, then it is easy to observe:

$$f(m,n) = f(m-1,n) + f(m,n-1)$$

and $f(m,0) = 1$, $f(0,n) = 1$. On the other hand, we only have m steps moving right, and n steps down. Thus it is equivalent to say we choose m steps out of $(m+n)$ steps to move right, the rest are for moving down, which gives $\binom{m+n}{m}$.

For this problem

$$\binom{40}{20} = \frac{40!}{20!20!} = \frac{40 \times 39 \times \ldots 21}{20 \times 19 \times \ldots 1}$$

Using `long long` type should be enough to calculate.

2.16 Problem 016 - Power digit sum

Problem 16.

2^{15} = 32768 and the sum of its digits is 3 + 2 + 7 + 6 + 8 = 26.

What is the sum of the digits of the number 2^{1000}?

Solution. Brute force using `BigInt` is extremely easy, since it can display all digits (around 300) of 2^{1000}, otherwise string addition is required.

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
```

```
#define L 400

void multiply_2(char * res, int N){
    int carry = 0;
    for (int j = N - 1; j >=0; j--){
        if (carry + 2 * (res[j] - '0') >= 10) {
            res[j] = carry + 2 * (res[j]-'0') - 10 + '0';
            carry = 1;
        }
        else {
            res[j] = carry + 2 * (res[j] - '0') + '0';
            carry = 0;
        }
    }
}

int digitsum(char * res, int N){
    int n = 0;
    for (int j = 0 ; j < N; j++){
        n += res[j] - '0';
    }
    return n;
}

void main(){
    char res[L];
    for (int j = 0; j < L; j++){
        res[j] = '0';
    }
    res[L - 1] = '1';
    for (int j = 0; j < 1000; j++) {
        multiply_2(res, L);
    }
    printf("%d\n", digitsum(res, L));
}
```

2.17 Problem 017 - Number letter counts

Problem 17.

If the numbers 1 to 5 are written out in words: one, two, three, four, five, then there are 3 + 3 + 5 + 4 + 4 = 19 letters used in total.

If all the numbers from 1 to 1000 (one thousand) inclusive were written out in words, how many letters would be used?

NOTE: Do not count spaces or hyphens. For example, 342 (three hundred and forty-two) contains 23 letters and 115 (one hundred and fifteen) contains 20 letters. The use of "and" when writing out numbers is in compliance with British usage.

Solution. Brute force, no other methods.

2.18 Problem 018 - Maximum path sum I

Problem 18.

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, is the same challenge with a triangle containing one-hundred rows; it cannot be solved by brute force, and requires a clever method! ;o)

Solution. It is a DP problem from bottom to top.

```
triangle =
[[75],
[95, 64],
[17, 47, 82],
[18, 35, 87, 10],
[20,  4, 82, 47, 65],
[19,  1, 23, 75,  3, 34],
[88,  2, 77, 73,  7, 63, 67],
[99, 65,  4, 28,  6, 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,  4, 68, 89, 53, 67, 30, 73, 16, 69, 87, 40, 31],
[ 4, 62, 98, 27, 23,  9, 70, 98, 73, 93, 38, 53, 60,  4, 23]]

for row in reversed(range(1, len(triangle))):
    for i in range(0, len(triangle[row]) - 1):
        triangle[row - 1][i] += max(triangle[row][i], triangle[row
            ][i + 1])
print triangle[0][0]
```

2.19 Problem 019 - Counting Sundays

Problem 19.

You are given the following information, but you may prefer to do some research for yourself.

- 1 Jan 1900 was a Monday.
- Thirty days has September,
  April, June and November.
  All the rest have thirty-one,
  Saving February alone,
  Which has twenty-eight, rain or shine.
  And on leap years, twenty-nine.
- A leap year occurs on any year evenly divisible by 4, but not on a century unless it is divisible by 400.

How many Sundays fell on the first of the month during the twentieth century (1 Jan 1901 to 31 Dec 2000)?

Solution. Brute force with library `datetime` from `Python` is straightforward.

```
import datetime
count = 0
for y in range(1901,2001):
    for m in range(1,13):
        if datetime.datetime(y,m,1).weekday() == 6:
            count += 1
print count
```

2.20 Problem 020 - Factorial digit sum

Problem 20.

$n!$ means $n \times (n - 1) \times \ldots \times 3 \times 2 \times 1$

For example, $10! = 10 \times 9 \times \ldots \times 3 \times 2 \times 1 = 3628800$,
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 $100!$

Solution. Brute force.

```
print reduce(lambda x, y: x + y, [int(i) for i in str(reduce(lambda
    x, y: x * y, range(1, 101)))])
```

2.21 Problem 021 - Amicable numbers

Problem 21.

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 $a \neq b$, 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.

Solution. Brute force, check all numbers and their sum of divisors.

```
import math

def f(n):
    return  sum([i + n/i for i in range(1, int(math.sqrt(n)) + 1)
        if n % i == 0]) - n

print reduce(lambda x, y: x+y, [i for i in range(1, 10000) if i ==
    f(f(i))!= f(i)])
```

2.22 Problem 022 - Names scores

Problem 22.

Using 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?

Solution. Brute force, using `qsort`.

```
names = open('../data/022.txt').readlines()[0][1:-1].split('","')
names.sort()
print reduce(lambda x, y : x + y, [reduce(lambda x, y: x + y, [(j +
     1) * (ord(i) - 64) for i in names[j]]) for j in xrange(len(
    names))])
```

2.23 Problem 023 - Non-abundant sums

Problem 23.

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.

Solution. Brute force. Nested loop to check whether a number can be written as a sum of two abundant number.

```
#include <stdio.h>
#include <math.h>
#include <stdbool.h>

#define LIMIT 28123

int abundant[LIMIT];

int is_abundant(int n){
    int d, s;
    if (abundant[n] > -1) {
        return abundant[n];
    }
    else {
        s = 1;
        for (d = 2; d * d < n ; d++) {
            if (n % d == 0) {
                s += d + n/ d;
            }
        }
        if (d * d == n) s += d;
    }
    abundant[n] = (s > n) ? 1:0;
    return abundant[n];
}

bool is_a_sum(int n) {
    for (int m = 12; m <= n/2; m++){

        if (is_abundant(m) && is_abundant(n - m)) return true;
    }
    return false;
}

void main(){
    int res = 0;
```

```
        for (int n = 0; n < LIMIT; n++) {
            abundant[n] = -1;
        }

        for (int n = 1; n < LIMIT; n++) {
            if (!is_a_sum(n)) res += n;
        }
        printf("%d\n", res);
    }
```

2.24 Problem 024 - Lexicographic permutations

Problem 24.

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?

Solution. Not necessary to call `next_permutation` million times.

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
res = []

n = 999999
m = 362880 # 9!

for i in range(9, 0, -1):
    d = n / m
    n = n - d * m
    res.append(numbers[d])
    numbers.pop(d)
    m /= i
res.append(numbers[0])
print ''.join(map(str, res))
```

2.25 Problem 025 - 1000-digit Fibonacci number

Problem 25.

The Fibonacci sequence is defined by the recurrence relation:

$F_n = F_{n-1} + F_{n-2}$, where $F_1 = 1$ and $F_2 = 1$.

Hence the first 12 terms will be:

$F_1 = 1$
$F_2 = 1$
$F_3 = 2$
$F_4 = 3$
$F_5 = 5$
$F_6 = 8$
$F_7 = 13$
$F_8 = 21$
$F_9 = 34$
$F_{10} = 55$
$F_{11} = 89$
$F_{12} = 144$

The 12th term, F_{12}, is the first term to contain three digits.

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

Solution. Brute force, just use $O(\log n)$ time complexity to get the n^{th} Fibonacci number. Or better Since we know the n^{th} is rounding following expression

$$F_n \sim \frac{1}{\sqrt{5}}(\phi^n) \geq 10^{999} \tag{2.6}$$

then

$$n \geq \frac{\log(5)/2 + 999 \log 10}{\log \phi} \sim 4781.8$$

3

LEVEL 2 @Problem 26 - 50

"God used beautiful mathematics in creating the world."

– Paul Dirac

3.1 Problem 026 - Reciprocal cycles

Problem 26.

A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10 are given:

$1/2 = 0.5$
$1/3 = 0.(3)$
$1/4 = 0.25$
$1/5 = 0.2$
$1/6 = 0.1(6)$
$1/7 = 0.(142857)$
$1/8 = 0.125$
$1/9 = 0.(1)$
$1/10 = 0.1$

Where 0.1(6) means 0.166666..., and has a 1-digit recurring cycle. It can be seen that $1/7$ has a 6-digit recurring cycle.

Find the value of $d < 1000$ for which $1/d$ contains the longest recurring cycle in its decimal fraction part.

Solution. If t is the period for number p, then

$$10^{s+t} \equiv 10^s \mod p \tag{3.1}$$

If p is relative prime with $2, 5$, then the above expression can be reduced to

$$10^t \equiv 1 \mod p \tag{3.2}$$

Then brute force.

```
max = 0;
for i in range(2, 1001):
    while (i % 2 == 0):
        i /= 2;
    while (i % 5 == 0):
        i /= 5;

    if (i > 1):
        r, n = 10, 1
        while (r % i != 1):
            r *= 10
            r %= i
            n += 1
        if (n > max):
            max = i
print max
```

3.2 Problem 027 - Quadratic primes

Problem 27.

Euler discovered the remarkable quadratic formula:

$$n^2 + n + 41$$

It turns out that the formula will produce 40 primes for the consecutive values $n = 0$ to 39. However, when $n = 40$, $40^2 + 40 + 41 = 40(40 + 1) + 41$ is divisible by 41, and certainly when $n = 41$, $41^2 + 41 + 41$ is clearly divisible by 41.

The incredible formula $n^2 - 79n + 1601$ was discovered, which produces 80 primes for the consecutive values $n = 0$ to 79. The product of the coefficients, −79 and 1601, is −126479.

Considering quadratics of the form:

$n^2 + an + b$, where $|a| < 1000$ and $|b| < 1000$

where $|n|$ is the modulus/absolute value of n
e.g. $|11| = 11$ and $|-4| = 4$

Find the product of the coefficients, a and b, for the quadratic expression that produces the maximum number of primes for consecutive values of n, starting with $n = 0$.

Solution. Brute force. But we know b is a prime number between 0 to 999. Also we can see that when $n = b$, the result must be a compound, thus n only goes from 0 to at most $b - 1$.

```
#include <stdio.h>
#include <stdbool.h>

bool isprime(int n){
    if (n == 2) return true;
    for (int i = 3; i * i < n; i += 2){
        if (n % i == 0) return false;
    }
    return true;
}
```

```
void main(){

    int primes[1000];
    int max = 0, res = 0, curr = 0;

    for (int i = 2; i < 1000; i++) {
        if (isprime(i)) primes[i] = 1;
        else primes[i] = 0;
    }

    for (int a = - 999; a < 1000; a += 2){
        for (int b = 3; b < 1000; b += 2){
            if (isprime(b)){
                for (int n = 0; n < b; n++) {
                    curr = n * n + n * a + b;
                    if (isprime(curr) && curr > 0) {
                        continue;
                    }
                    else {
                        if (max < n) {max = n;res= a * b;}
                        break;
                    }
                }
            }
        }
    }

    printf("%d\n", res);

}
```

3.3 Problem 028 - Number spiral diagonals

Problem 28.

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?

Solution. Deducing the formula directly may take time. We can prove that on each layer, the increment is linear to n, that means each layer has a sum as quadratic polynomial in n. That means the formula will take form as a cubic

polynomial $f(n)$ in n, with $f(1) = 1, f(3) = 25, f(5) = 101, f(7) = 261$.

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 81 & 27 & 3 & 1 \\ 125 & 25 & 5 & 1 \\ 343 & 49 & 7 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 1 \\ 25 \\ 101 \\ 261 \end{pmatrix} \tag{3.3}$$

gives $f(n) = \frac{1}{6}(4n^3 + 3n^2 + 8n - 9)$

3.4 Problem 029 - Distinct powers

Problem 29.

Consider all integer combinations of a^b for $2 \le a \le 5$ and $2 \le b \le 5$:

$2^2=4, 2^3=8, 2^4=16, 2^5=32$
$3^2=9, 3^3=27, 3^4=81, 3^5=243$
$4^2=16, 4^3=64, 4^4=256, 4^5=1024$
$5^2=25, 5^3=125, 5^4=625, 5^5=3125$

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 a^b for $2 \le a \le 100$ and $2 \le b \le 100$?

Solution. Brute force with `Python` one liner!

```
print(len(set(a**b for a in range(2, 101) for b in range(2, 101))))
```

3.5 Problem 030 - Digit fifth powers

Problem 30.

Surprisingly there are only three numbers that can be written as the sum of fourth powers of their digits:

$1634 = 1^4 + 6^4 + 3^4 + 4^4$
$8208 = 8^4 + 2^4 + 0^4 + 8^4$
$9474 = 9^4 + 4^4 + 7^4 + 4^4$

As $1 = 1^4$ is not a sum it is not included.

The sum of these numbers is $1634 + 8208 + 9474 = 19316$.

Find the sum of all the numbers that can be written as the sum of fifth powers of their digits.

Solution. Brute force, since the largest number will not exceed $5 \cdot 9^5$.

```
def f(i):
    return reduce(lambda x, y: x+ y, [int(d) ** 5 for d in str(i)])
print reduce(lambda x, y: x+y, [i for i in range(2, 5 * 9 **5) if f
    (i) == i])
```

3.6 Problem 031 - Coin sums

Problem 31.

In England the currency is made up of pound, £, and pence, p, and there are eight coins in general circulation:

1p, 2p, 5p, 10p, 20p, 50p, £1 (100p) and £2 (200p).

It is possible to make £2 in the following way:

1×£1 + 1×50p + 2×20p + 1×5p + 1×2p + 3×1p

How many different ways can £2 be made using any number of coins?

Solution. Consider the generating function

$$\prod_i \frac{1}{1-x^{n_i}}$$

We simply need to find out the coefficient of x^{200}. To avoid long division and observe that $n_i \mid 200$, we can use

$$\prod_i \frac{1-x^{200+n_i}}{1-x^{n_i}}$$

instead, that will simply require multiplication of polynomials.

```
def f(a, b):
    n = len(a)
    m = len(b)
    c = [0] * (n + m - 1)
    for j in range(n):
        for k in range(m):
            c[j + k] += a[j]*b[k]

    return c

coin = [1, 2, 5, 10, 20, 50, 100, 200]

print(reduce(lambda x,y:f(x,y) , [([1] + [0]*(i - 1)) * (200/i) +
    [1] for i in coin])[200])
```

3.7 Problem 032 - Pandigital products

Problem 32.

We shall say that an n-digit number is pandigital if it makes use of all the digits 1 to n exactly once; for example, the 5-digit number, 15234, is 1 through 5 pandigital.

The product 7254 is unusual, as the identity, 39 × 186 = 7254, containing multiplicand, multiplier, and product is 1 through 9 pandigital.

Find the sum of all products whose multiplicand/multiplier/product identity can be written as a 1 through 9 pandigital.

HINT: Some products can be obtained in more than one way so be sure to only include it once in your sum.

Solution. It is easy to show there are only two possible cases:

$$\overline{a} \times \overline{bcde} = \overline{fghi} \tag{3.4}$$

$$\overline{ab} \times \overline{cde} = \overline{fghi} \tag{3.5}$$

For the first case, there are 630 choices at most. For the second, there are 1260 combinations.

```
import itertools
t = '123456789'
r = set(t)
n = set()
s = list(itertools.permutations(t,5))

for i in s:
    p = (10 * int(i[0]) + int(i[1])) * (100 * int(i[2]) + 10 * int(
        i[3]) + int(i[4]))
    q = (int(i[0])) * (1000 * int(i[1]) + 100 * int(i[2]) + 10 *
        int(i[3]) + int(i[4]))
    if (set(str(p)) == r.difference(set(i)) and p < 10000):
        n.add(p)
    if (set(str(q)) == r.difference(set(i)) and q < 10000):
        n.add(q)
print sum(n)
```

3.8 Problem 033 - Digit cancelling fractions

Problem 33.

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.

Solution. `Python` provides `Fraction` to manipulate rational numbers. Here we take advantage of `Julia`'s `Rational` Type.

```
s = 1;
for i = 11:99
    for j = 11:99
        if i < j
            if (i % 10) == (int)(floor(j/10)) &&  i // j == int(
                floor(i/10)) // (j % 10)
            s *= i//j
            end
        end
    end
end
println(den(s));
```

3.9 Problem 034 - Digit factorials

Problem 34.

145 is a curious number, as 1! + 4! + 5! = 1 + 24 + 120 = 145.

Find the sum of all numbers which are equal to the sum of the factorial of their digits.

Note: as 1! = 1 and 2! = 2 are not sums they are not included.

Solution. Brute force. The upper bound can be determined by

$$9! \cdot n < 10^n$$

which gives $n \leq 7$ digits, the upper bound must be lower than $9! \cdot 7 = 2540160$.

```
from math import *
r = 0
m = [factorial(j) for j in range(10)]
for i in range(3, m[-1] * 7):
    s = str(i)
    t = 0
    for j in s:
        t += m[int(j)]
        if (t > i) :
            break;
    if (i == t):
        r += i
print(r)
```

3.10 Problem 035 - Circular primes

Problem 35.

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?

Solution. For each number greater than 2 which is satisfying the circular prime condition, the digits must be all odd but not 5. There are not too many prime numbers within one million which contains only $1, 3, 7, 9$. Circular prime check should be $O(1)$, if there is a prime number set.

```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#define N 1000000

bool isprime[N];

void sieve(){
    for (int k = 2; k < sqrt(N); k++){
        if (isprime[k]){
            for (int j = k * k; j <= N; j += k){
                isprime[j] = false;
            }
        }
    }
}

bool is_circular_prime(int n){
    int m = n, k = 0, l = 1;

    while (m != 0){
```

```
            m /= 10;
            l *= 10;
            k++;
        }
        l/= 10;
        for (int ii = 0; ii < k; ii++){
            n = (n - n % 10)/10 + l * (n % 10);
            if (!isprime[n]) {
                return false;
            }
        }
        return true;
    }

    void main(){
        for (int i = 0; i < N; i++) {
            isprime[i] = true;
        }
        sieve();
        int l = 0;
        for (int j = 2; j < N; j++){
            if (is_circular_prime(j)) {
                l++;
            }
        }

        printf("%d\n", l);
    }
```

3.11 Problem 036 - Double-base palindromes

Problem 36.

The decimal number, 585 = 1001001001_2 (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.)

3.12 Problem 037

Problem 37.

3.13 Problem 038

Problem 38.

A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10 are given:

$1/2 = 0.5$
$1/3 = 0.(3)$
$1/4 = 0.25$
$1/5 = 0.2$
$1/6 = 0.1(6)$
$1/7 = 0.(142857)$
$1/8 = 0.125$
$1/9 = 0.(1)$
$1/10 = 0.1$

Where 0.1(6) means 0.166666..., and has a 1-digit recurring cycle. It can be seen that $1/7$ has a 6-digit recurring cycle.

Find the value of $d < 1000$ for which $1/d$ contains the longest recurring cycle in its decimal fraction part.

A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10 are given:

$1/2 = 0.5$
$1/3 = 0.(3)$
$1/4 = 0.25$
$1/5 = 0.2$
$1/6 = 0.1(6)$
$1/7 = 0.(142857)$
$1/8 = 0.125$
$1/9 = 0.(1)$
$1/10 = 0.1$

Where 0.1(6) means 0.166666..., and has a 1-digit recurring cycle. It can be seen that $1/7$ has a 6-digit recurring cycle.

Find the value of $d < 1000$ for which $1/d$ contains the longest recurring cycle in its decimal fraction part.

3.14 Problem 039

Problem 39.

3.15 Problem 040

Problem 40.

A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10 are given:

$1/2 = 0.5$
$1/3 = 0.(3)$
$1/4 = 0.25$
$1/5 = 0.2$
$1/6 = 0.1(6)$
$1/7 = 0.(142857)$
$1/8 = 0.125$
$1/9 = 0.(1)$
$1/10 = 0.1$

Where 0.1(6) means 0.166666..., and has a 1-digit recurring cycle. It can be seen that $1/7$ has a 6-digit recurring cycle.

Find the value of $d < 1000$ for which $1/d$ contains the longest recurring cycle in its decimal fraction part.

A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10 are given:

$1/2 = 0.5$
$1/3 = 0.(3)$
$1/4 = 0.25$
$1/5 = 0.2$
$1/6 = 0.1(6)$
$1/7 = 0.(142857)$
$1/8 = 0.125$
$1/9 = 0.(1)$
$1/10 = 0.1$

Where 0.1(6) means 0.166666..., and has a 1-digit recurring cycle. It can be seen that $1/7$ has a 6-digit recurring cycle.

Find the value of $d < 1000$ for which $1/d$ contains the longest recurring cycle in its decimal fraction part.

4

LEVEL 3 @Problem 51 - 75

4.1 Problem 051

Problem 41.

5

LEVEL 4 @Problem 76 - 100

5.1 Problem 076

Problem 42.

6

LEVEL 5 @Problem 101 - 125

6.1 Problem 101

Problem 43.

7

LEVEL 6 @Problem 126 - 150

7.1 Problem 126

Problem 44.

8

LEVEL 7 @Problem 151 - 175

8.1 Problem 051

Problem 45.

9

LEVEL 8 @Problem 176 - 200

9.1 Problem 051

Problem 46.

10

LEVEL 9 @Problem 201 - 225

10.1 Problem 051

Problem 47.

11

LEVEL 10 @Problem 226 - 250

11.1 Problem 051

Problem 48.

12

LEVEL 11 @Problem 251 - 275

12.1 Problem 051

Problem 49.

13

LEVEL 12 @Problem 276 - 300

13.1 Problem 051

Problem 50.

14

LEVEL 13 @Problem 301 - 325

14.1 Problem 301

Problem 51.

15

LEVEL 14 @Problem 326 - 350

15.1 Problem 326

Problem 52.

16

LEVEL 15 @Problem 351 - 375

16.1 Problem 351

Problem 53.

17

LEVEL 16 @Problem 376 - 400

17.1 Problem 051

Problem 54.

18

LEVEL 17 @Problem 401 - 425

18.1 Problem 401

Problem 55.

19

LEVEL 18 @Problem 426 - 450

19.1 Problem 426

Problem 56.

20

LEVEL 19 @Problem 451 - 475

20.1 Problem 451

Problem 57.

21

LEVEL 19+ @Problem 475+

21.1 Problem 476

Problem 58.

21.2 Problem 479 - Roots on the Rise

Problem 59.

Let a_k, b_k, and c_k represent the three solutions (real or complex numbers) to the expression $1/x = (k/x)^2(k+x^2) - kx$.

For instance, for $k = 5$, we see that $\{a_5, b_5, c_5\}$ is approximately {5.727244, -0.363622+2.057397i, -0.363622-2.057397i}.

Let $S(n) = \Sigma\ (a_k+b_k)^p(b_k+c_k)^p(c_k+a_k)^p$ for all integers p, k such that $1 \le p$, $k \le n$.

Interestingly, $S(n)$ is always an integer. For example, $S(4) = 51160$.

Find $S(10^6)$ modulo 1 000 000 007.

Solution. By Vieta's formulas, $a_k + b_k + c_k = k$, $a_kb_k + b_kc_k + c_ka_k = 1/k$, $a_kb_kc_k = k^2$.

$$(a_k+b_k)(b_k+c_k)(c_k+a_k) = (a_k+b_k+c_k)(a_kb_k+b_kc_k+c_ka_k) - a_kb_kc_k = 1-k^2.$$

Then we just have to evaluate

$$\sum_{k=1}^{n}((1-k^2)^{n+1} - (1-k^2))/(-k^2) \mod p \tag{21.1}$$

where using `Little Fermat's Theorem`,

$$(-k^2)^{-1} \equiv (-k^2)^{p-2} \mod p \tag{21.2}$$

Using `ModPow` with `BigInt` will be efficient, in `Python`, simply use `pow` with the third argument.

```
N = 1000000
m = 1000000007L
S = 0
for i in range(2, N + 1):
    S += (pow(1 - i*i, N + 1, m) - (1 - i * i))*pow((- i * i), m -
        2, m) % m
    S %= m
print S
```

21.3 Problem 485

Problem 60.

Let d(n) be the number of divisors of n.
Let M(n,k) be the maximum value of d(j) for $n \le j \le n+k-1$.
Let S(u,k) be the sum of M(n,k) for $1 \le n \le u-k+1$.

You are given that S(1000,10)=17176.

Find S(100 000 000,100 000).

Solution. Consider the problem size, $n = 10^8$, which means storing $d(n)$ for $n \le 10^8$ will not take too much memory(~ 0.8 Gb).

21.4 Problem 487 - Sums of power sums

Problem 61.

Let $f_k(n)$ be the sum of the k^{th} powers of the first n positive integers.

For example, $f_2(10) = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 + 7^2 + 8^2 + 9^2 + 10^2 = 385$.

Let $S_k(n)$ be the sum of $f_k(i)$ for $1 \le i \le n$. For example, $S_4(100) = 35375333830$.

What is $\Sigma\ (S_{10000}(10^{12}) \bmod p)$ over all primes p between $2 \cdot 10^9$ and $2 \cdot 10^9 + 2000$?

Solution. Use `Little Fermat's Theorem`, the problem size could *dramatically* be reduced.

21.5 Problem 488

21.6 Problem 491

21.7 Problem 493

21.8 Problem 498

21.9 Problem 499 - St.Petersburg Lottery

Problem 62.

A gambler decides to participate in a special lottery. In this lottery the gambler plays a series of one or more games.
Each game costs m pounds to play and starts with an initial pot of 1 pound. The gambler flips an unbiased coin. Every time a head appears, the pot is doubled and the gambler continues. When a tail appears, the game ends and the gambler collects the current value of the pot. The gambler is certain to win at least 1 pound, the starting value of the pot, at the cost of m pounds, the initial fee.

The gambler cannot continue to play if his fortune falls below m pounds.
Let $p_m(s)$ denote the probability that the gambler will never run out of money in this lottery given his initial fortune s and the cost per game m.
For example $p_2(2) \approx 0.2522$, $p_2(5) \approx 0.6873$ and $p_6(10\,000) \approx 0.9952$ (note: $p_m(s) = 0$ for $s < m$).

Find $p_{15}(10^9)$ and give your answer rounded to 7 decimal places behind the decimal point in the form 0.abcdefg.

Solution.

21.10 Problem 500

21.11 Problem 501

21.12 Problem 504

21.13 Problem 506