Euler Project

@GaZ3ll3[1]

March 23, 2015

[1]www.github.com/gaz3ll3

Contents

1

LEVEL 1 @Problem 1 - 25

"This is a quote and I don't know who said this."
– Author's name, *Source of this quote*

1.1 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 $k_1, k_2, \ldots, k_n$.

$$S = S_3 + S_5 - S_{3,5} = S_3 + S_5 - S_{15} \tag{1.1}$$

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

1.2 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

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right) \tag{1.2}$$

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

1.3 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);

    }
```

1.4 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

$$\overline{abccba} = 100001a + 10010b + 1100c = 11(9091a + 910b + 100c) \tag{1.3}$$

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);
    }
```

1.5 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

$$(a, b)[a, b] = ab \tag{1.4}$$

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);
}
```

1.6 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

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

1.7 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;
        }
    }
}
```

1.8 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"
    "7017242712188399879790879227492190169972088809377664"
    "6572733300105336788122023542180975125454059475224352"
    "5258490771167055601360483958644670632441572215539753"
    "5369781797784617406495514929086256932197846862248239"
    "8397224137565705605749026140797296865241453510047423"
    "8216637048440319989000889524345065854122758866688116"
    "1642717147992444292823086346567481391912316282458617"
    "1786645835912456652947654568284891288314260769004224"
    "2421902267105562632111110937054421750694165896040807"
    "0719840385096245544436298123098787992724428490918884"
    "8458015616609791913387549920052406368991256071760605"
    "0588611646710940507754100225698315520005593572972571"
    "6163626956188267042825248360082325753042075296345001");

    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);

}
```

1.9 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 $m > n$.

$$m^2 + m < m(m + n) = 500 < 2m^2 \tag{1.5}$$

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

1.10 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);
}
```

1.11 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);
    }
```

1.12 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;
            }
        }
    }
```

1.13 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));

}
```

1.14 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.

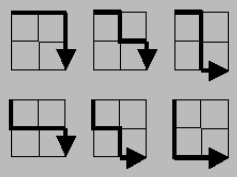
Solution. Brute force. Calculate the length of sequence for each start.

```
#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);
}
```

1.15 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.

1.16 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 `Python` is extremely easy, since it can display all digits (around 300) of 2^{1000}. Using `string` to represent the number is also fast.

1.17 Problem 017

Problem 17.

1.18 Problem 018

Problem 18.

1.19 Problem 019

Problem 19.

1.20 Problem 020

Problem 20.

1.21 Problem 021

Problem 21.

1.22 Problem 022

Problem 22.

1.23 Problem 023

Problem 23.

1.24 Problem 024

Problem 24.

1.25 Problem 025

Problem 25.

2

LEVEL 2 @Problem 26 - 50

2.1 Problem 026

Problem 26.

3

LEVEL 3 @Problem 51 - 75

3.1 Problem 051

Problem 27.

4

LEVEL 4 @Problem 76 - 100

4.1 Problem 076

Problem 28.

5

LEVEL 5 @Problem 101 - 125

5.1 Problem 101

Problem 29.

6

LEVEL 6 @Problem 126 - 150

6.1 Problem 126

Problem 30.

7

LEVEL 7 @Problem 151 - 175

7.1 Problem 051

Problem 31.

8

LEVEL 8 @Problem 176 - 200

8.1 Problem 051

Problem 32.

9

LEVEL 9 @Problem 201 - 225

9.1 Problem 051

Problem 33.

10

LEVEL 10 @Problem 226 - 250

10.1 Problem 051

Problem 34.

11

LEVEL 11 @Problem 251 - 275

11.1 Problem 051

Problem 35.

12

LEVEL 12 @Problem 276 - 300

12.1 Problem 051

Problem 36.

13

LEVEL 13 @Problem 301 - 325

13.1 Problem 301

Problem 37.

14

LEVEL 14 @Problem 326 - 350

14.1 Problem 326

Problem 38.

15

LEVEL 15 @Problem 351 - 375

15.1 Problem 351

Problem 39.

16

LEVEL 16 @Problem 376 - 400

16.1 Problem 051

Problem 40.

17

LEVEL 17 @Problem 401 - 425

17.1 Problem 401

Problem 41.

18

LEVEL 18 @Problem 426 - 450

18.1 Problem 426

Problem 42.

19

LEVEL 19 @Problem 451 - 475

19.1 Problem 451

Problem 43.

20

LEVEL 19+ @Problem 475+

20.1 Problem 476

Problem 44.