Project Euler Solution

(No145, No173, No179)

학 과	컴퓨터공학과
학 번	201211704
이 름	김기홍
작성일	2015.10.03
제출일	2015.11.26

부경대학교

Problem 145

- How many reversible numbers are there below one-billion?

Some positive integers have the property that the sum [n + reverse(n)] consists entirly of odd (decimal) digits.
For instance, 36 + 63 = 99 and 409 + 904 = 1313.
We will call such numbers reversible;
so 36, 63, 409, and 904 are reversible.
Leading zeroes are not allowed in either n or reverse(n).

There are 120 reversible numbers below one-thousand.

How many reversible numbers are there below one-billion
(0)?

Solution)

case by the number of digits!

if a case of 1-digit

$\square$

<1~9>

can't be reversible. Becasue n is the same as reverse(n), the sum of them is always even.

if a case of 2-digits

$\square$ $\square$

<1~9> <1~9>

Think each digit of n as a and b, so reverse(n) is b and a.
n + reverse(n) is (a+b) (b+a).
n can be reversible if a+b is odd and a+b<10.

if a+b is greater than or equal to 10, there occurs a carry so that can't be reversible.

```
for(i=1; i<10; i++)
   for(j=1; j<10; j++)
       if((i+j)%2 == 1 && (i+j) < 10)
           count++;
```

count = 20.

if a case of 3-digits

<1~9><0~4><1~9>

Same as a case of 2-digts, each digit of n as a, b and c, so reverse(n) is c,b and a.

a b c
\+ c b a

so (a+c) (b+b) (c+a), b+b is always even without a carry. And b is greater than 4, n cannot be reversible because of a useless carry.

so, a+b is odd and a+b>10 and b is less than 5.

count = 20 * 5 = 100.

Let's define a count case by case. (by using grammar for(;;))

a+b odd and less than 10 : 20(a and b are a digit of the end of n else 30) - odd_less

a+b odd and greater than 10 : 20 - odd_greater

a+b even and less than 10 : 16(a and b are a digit of the end of n else 25) - even_less

if a case of 4-digits

odd_less odd_less odd_less odd_less -> *count* = 20 * 30 = 600.

if a case of 5-digits

can't be reversible. Because no case that makes all of digit odd exists.

if a case of 6-digits

odd_less " " " " "

-> *count* = 20 * 30 * 30 = 18000.

if a case of 7-digits

odd_greater even_less odd_greater <0~4>

-> *count* = 20 * 25 * 20 * 5 = 50000.

if a case of 8-digits

odd_less " " " " " " "

-> *count* = 20 * 30 * 30 * 30 = 540000.

if a case of 9-digits

no case.

The number of digits is

2n -> *count* = $0*30^{-1}$

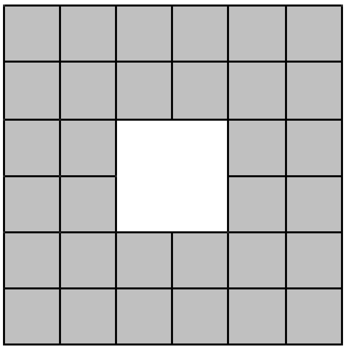
4n-1 -> *count* = $5*20^{n}*25^{n-1}$

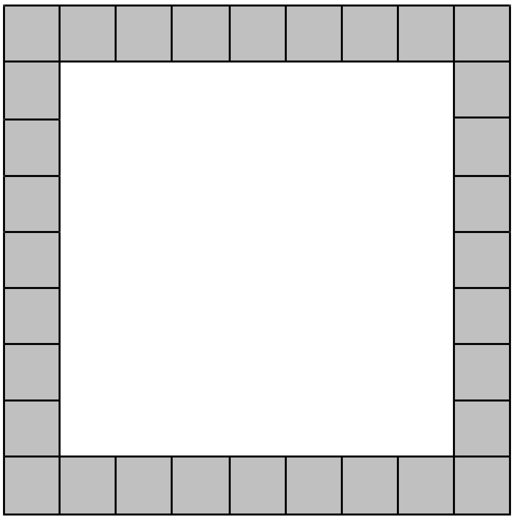
4n-3 -> *count* = X

The answer is 608720.

Problem 173
- Using up to one million tiles how many different "hollow" square laminae can be formed?

We shall define a square lamina to be a square outline with a square "hole" so that the shape possesses vertical and horzontal symmetry. For example, using exactly thirty-two square tiles we can form two different square laminae:





With one-hunderd tiles, and not necessarily using all of the tiles at one time, it is possible to form forty-one different square laminae.
Using up to one million tiles how many different square laminae can be formed?

Solution)

Find the rule!

We can use a variable which is *hole* and *border*.
hole means the number of a center square's row(column).
border means the number of a outline square's row(column).

hole \ border	1	2	3	n
1*1	-1^2	5^2-1^2	7^2-1^2	$(1+2n)^2-1^2$
2*2	4^2-2^2	6^2-2^2	8^2-2^2	$(2+2n)^2-2^2$
3*3	5^2-3^2	7^2-3^2	9^2-3^2	$(3+2n)^2-3^2$
n*n	$(n+2)^2-n^2$	$(n+4)^2-n^2$	$(n+6)^2-n^2$	$8n^2$

We can get the answer

```
hole = 1; border = hole+2;
while(border*border - hole*hole <= MAXNUM)
{                                            the number of whole tile
    count++;
    border += 2;
    while(border*border - hole*hole <= MAXNUM)
    {
        count++;
        border += 2;
    }
    hole++;
    border = hole+2;
}
```

The procedure is

first - Find all different "hollow" square laminae with the hole size 1*1 which can be made with one million tiles.

and then - the same routine except the hole size(2*2,3*3,...).

The answer is 1572729.

no173.c + (~/Desktop/SYSProg) - VIM

```c
#include <stdio.h>
#define MAXNUM 1000000              /* the number of whole tile */

int get_num_hollow()
{
    int num=0;
    int hole, border;

    hole=1; border=hole+2;

    while(border*border - hole*hole <= MAXNUM)
    {
        num++;
        border += 2;
        while(border*border - hole*hole <= MAXNUM)      /* hole size is constant but only border is increasing  */
        {
            num++;
            border += 2;
        }

        hole++;                                     /* hole size is increasing  */
        border = hole+2;
    }

    return num;
}

int main()
{
    int answer = get_num_hollow();

    printf("The number of different \"hollow\" with %d tiles is %d \n\n", MAXNUM, answer);

    return 0;
}
```

```
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $ no173
The number of different "hollow" with 1000000 tiles is 1572729

mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
```

Problem 179
- Consecutive positive divisors

Find the number of integers $< n < 10$, for which n and $n+1$ have the same number of positive divisors. For example, 14 has the positive divisors 1, 2, 7 ,14 while 15 has 1, 3, 5, 15.

Solution)

Use an similar algorithm with *Sieve of Erastosthenes*

This algorithm uses a value of an array's index.
The array's index matches an integer.
For example,

array					
index	0	1	2	3	4
integer	0	1	2	3	4

And the value of array is the number of divisors.

array	0	1	2	2	3
index	0	1	2	3	4
integer	0	1	2	3	4

We can get the value like following method.

```
index = 2; multiple; table[MAXNUM]; // initialized as 2
while(index*index < MAXNUM)
{
    multiple = index*index;
    table[multiple]++;
    multiple += index;
    while(multiple < MAXNUM)
    {
        table[multiple] += 2;
        multiple += index;
    }
    index++;
}
```

array	2	2	2	2	2	2	2
index	2	3	4	5	6	7	8
integer	2	3	4	5	6	7	8

<table_1>

	index		multiple				
array	2	2	3	2	2	2	2
index	2	3	4	5	6	7	8
integer	2	3	4	5	6	7	8

<table_2>

	index				multiple		
array	2	2	3	2	4	2	2
index	2	3	4	5	6	7	8
integer	2	3	4	5	6	7	8

<table_3>

	index						multiple
array	2	2	3	2	4	2	4
index	2	3	4	5	6	7	8
integer	2	3	4	5	6	7	8

<table_4>

And then we get the table.
To solve this problem finally, we should compare index with index+1. If they are same, we are counting.
Increasing the index, continue comparing.
After the loop is over, the variable count is the answer.

The answer is 986262.

no179.c + (~/Desktop/SYSProg) - VIM

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

#define MAXNUM 10000000

void make_table(int *table)         /* make a table which has the integer's postive divisors as the value  */
{
    int index=2;
    int multiple;

    while(index*index < MAXNUM)
    {
        multiple = index*index;
        table[multiple]++;            /* power of index has been added 1(index)  */

        multiple += index;

        while(multiple < MAXNUM)      /* all the value of multiple by index has been added 2(index and integer/index)  */
        {
            table[multiple] += 2;
            multiple += index;
        }

        index++;
    }
}

int compare()
{
    int *table = (int *)malloc(sizeof(int)*MAXNUM);
    int i, count=0;

    table[0]=0; table[1]=1;

    for(i=2; i<MAXNUM; i++)           /* intialized as 2(1 and integer)  */
        table[i]=2;

    make_table(table);

    for(i=2; i<MAXNUM-1; i++)       /* compare index with index+1  */
        if(table[i] == table[i+1])
            count++;

    return count;
}

int main()
{
    int answer = compare();

    printf("The number of consecutive integers 1 < n < %d which have the same number of divisors is %d \n\n", MAXNUM, answer);

    return 0;
}
```

```
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $ no179
The number of consecutive integers 1 < n < 10000000 which have the same number of divisors is 986262

mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
mint@mint ~/Desktop/SYSProg $
```