Number of Positive Integer Solutions

Author: Zhirong(Larry) Li

I have encountered a very good brain teaser question and its solution over the Internet. The original problem is to find the number of positive integer solutions for $\frac{1}{x}+\frac{1}{y}=\frac{1}{n!}$ where n is any natural number.

Solution:

For n is EQ 1, we can get $\frac{1}{x}=1-\frac{1}{y}=\frac{y-1}{y}$. Since $y-1$ and y are co-prime, hence the unique solution for this equation is $x=2,\ y=2$.

For generic n and $\frac{1}{x}+\frac{1}{y}=\frac{1}{n}$, since both $x\ and\ y$ are positive, they both must be greater than n. Without loss of generality, suppose $x=n+d$, then $\frac{1}{y}=\frac{1}{n}-\frac{1}{n+d}=\frac{1}{\frac{n^2}{d}+n}$. It is then obvious that d must be a divisor of n^2. Because of symmetric property of the equation, the total number of positive integer solutions must be equal to the half of the total number of distinct divisors of n^2 plus one.

When we get back to $\frac{1}{x}+\frac{1}{y}=\frac{1}{n!}$ case, the total number of positive integer solutions must be equal to the half of the total number of distinct divisors of $(n!)^2$ plus one.

Look familiar? Remember how we calculate the number of trailing zeros of 100! (The answer is 24).

We know that $n!$ can be represented as $n!=2^{e_2}\times 3^{e_3}\times\cdots\times p^{e_p}\times\cdots$. As a result, e_p can be derived from the following:

$$e_p=\left\lfloor\frac{n}{p}\right\rfloor+\left\lfloor\frac{n}{p^2}\right\rfloor+\left\lfloor\frac{n}{p^3}\right\rfloor+\cdots=\sum_{i=1}^{\lfloor\log_p N\rfloor}\left\lfloor\frac{n}{p^i}\right\rfloor$$

Finally, we can get the closed-form solution for the number of positive integer solutions for $\frac{1}{x}+\frac{1}{y}=\frac{1}{n!}$ is

$$\frac{1+\prod\limits_{p\ is\ prime,\ p\le N}(2e_p+1)}{2}$$

For $n=10$, there are 1148 distinct solutions and the value of the answer increases very fast.

20/12/2012

A snippet C++ code for computing the solution number is provided below:

```
#include <iostream>
#include <vector>
#include <math.h>
using namespace std;

int main(){

	const long N =10;
	// generate the prime table using sieve method
	bool* prime_table = new bool[N+1];
	prime_table[0] = false;
	prime_table[1] = false;
	prime_table[2] = true;
	for(long i=3;i<=N;i++)
		prime_table[i] = true;
	long upper_bound = static_cast<long>(sqrt(static_cast<double>(N)));
	for(long i=2;i<=upper_bound;i++){
		for(long j=i*i;j<=N;j+=i){
			if (prime_table[i]){
				prime_table[j] = false;
			}
		}
	}

	// compute the exponent of prime factors
	vector<int> prime_exponent;
	for(long p=2;p<=N;p++){
		if (prime_table[p]){
			int acc_sum = 0;
			int upper_bound =
static_cast<int>(floor(log(static_cast<double>(N))/log(static_cast<double>(p))));
			for (long j=1;j<=upper_bound;j++){
				acc_sum += static_cast<int>(floor(N/pow(static_cast<double>(p),j)));
			}
			prime_exponent.push_back(acc_sum);
		}
	}

	// calculate the number of positive integer solutions
	long multiplier =1;
	for(std::vector<int>::iterator it = prime_exponent.begin(); it != prime_exponent.end(); ++it)
{
		multiplier *=2*(*it)+1;
	}
	multiplier = (1+multiplier)/2;
	cout<<"The number of solutions for 1/x+1/y=1/(N!) where N="<<N<<" is: "<<multiplier<<endl;
	delete prime_table;
}
```