Numbers Summation

Attempted by: **479** / Accuracy: **91%** / Maximum Score: **30** / 3 Votes

Tag(s): Circuits, Math, Medium, Number Theory, Simple-math

Author: Saatwik Singh Nagpal
Tester: Lewin Gan
Editorialist: Saatwik Singh Nagpal

Prerequisites:

Number Theory

Problem:

The problem asks us to evaluate the following expression for a given N:
$S = \sum_{i=1}^{N} \sum_{j=i}^{N} F(i,j)$, where $F(i,j)$ are the number of common divisors of i and j.

Small constraints:

If one writes a brute force code, it is enough to get a score of 12 points. The complexity of such a brute force is $O(N^2 * K)$, where K is the time to find common divisors of two numbers.

Medium constraints

If a contestant managed to write an $O(N)$ solution, it was enough to get a score of 27 points. But how to get an $O(N)$ solution? Let's find out. It will help us to develop the solution for the large constraints. Instead of directly evaluating the asked expression, we will evaluate a modified form of it. Let cnt_i be the number of times i comes in the above expression as a common divisor of some pairs of numbers. Now if we know the value of cnt_i for all i, our problem reduces to evaluating the following expression:
$S = \sum_{i=1}^{N} i * cnt_i$
Now, how to find the value of cnt_i? Well it's pretty easy to do. The number of numbers $\leq N$ that have i as a divisor are $\left\lfloor \frac{N}{i} \right\rfloor$. So, the number of pairs that have i as a common factor are $\binom{cnt_i}{2}$, which is equal to $\frac{\left\lfloor \frac{N}{i} \right\rfloor * (\left\lfloor \frac{N}{i} \right\rfloor - 1)}{2}$. Let's say that $f(i) = \binom{i}{2}$. So, now we just have to evaluate the following expression :

$S = \sum_{i=1}^{N} i * \frac{\left\lfloor \frac{N}{i} \right\rfloor * (\left\lfloor \frac{N}{i} \right\rfloor - 1)}{2} = \sum_{i=1}^{N} i * f\left(\left\lfloor \frac{N}{i} \right\rfloor\right)$

Large Constraints:

We propose an $O(\sqrt{N})$ to solve the problem.
Claim: When we find the values of $\left\lfloor \frac{N}{i} \right\rfloor$ for i from 1 to N, there turn out to be only $O(\sqrt{N})$ such values. Also, these values form continuous ranges where they are equal. But how ?

Short proof:

It should be clear that the that the values of $\left\lfloor \frac{N}{i} \right\rfloor$ will be non-increasing on increasing i. We will iterate over the values in a clever way and prove our claim. The iteration is in two phases:

Phase 1

Lets iterate i from 1 to $\sqrt{N}$. Here we will cover $O(\sqrt{N})$ such values and by the time we reach $i = \sqrt{N}$, the value of $\left\lfloor \frac{N}{i} \right\rfloor$ would be approximately $\sqrt{N}$. Till now, we have only found $O(\sqrt{N})$ values which range from $\sqrt{N}$ to N.

Phase 2

Till now, we have iterated i till $i = \sqrt{N}$ and the value of $\left\lfloor \frac{N}{i} \right\rfloor$ was approximately $\sqrt{N}$ at this i. Now, we know that the value of $\left\lfloor \frac{N}{i} \right\rfloor$ would further decrease from $\sqrt{N}$. Now, as there are only $\sqrt{N}$ values from 1 to $\sqrt{N}$, therefore, we found $O(\sqrt{N})$ values in phase two of our iteration.

This way we proved that there are only $O(\sqrt{N})$ distinct values of $\left\lfloor \frac{N}{i} \right\rfloor$ for i from 1 to N.

So, we will have continuous ranges which will have the same value of $f()$. Hence, we can iterate over these ranges in $O(\sqrt{N})$.

So, the complexity for solving the problem is $O(\sqrt{N})$.

IS THIS EDITORIAL HELPFUL?

Yes, it's helpful

No, it's not helpful

5 developer(s) found this editorial helpful.

Author Solution by Saatwik Singh Nagpal

```
#include <bits/stdc++.h>
using namespace std;

#define TRACE
#ifdef TRACE
#define TR(...) __f(#__VA_ARGS__, __VA_ARGS__)
template <typename Arg1>
void __f(const char* name, Arg1&& arg1){
  cerr << name << " : " << arg1 << std::endl;
}
template <typename Arg1, typename... Args>
void __f(const char* names, Arg1&& arg1, Args&&... args){
```

```
  const char* comma = strchr(names + 1, ',');cerr.write(names, comma - names) << " : " << arg1<<" | ";__f(comma+1, args...);
}
#else
#define TR(...)
#endif

typedef long long                   LL;
typedef vector < int >             VI;
typedef pair < int,int >           II;
typedef vector < II >              VII;

#define MOD                         1000000007
#define EPS                         1e-12
#define PB                          push_back
#define MP                          make_pair
#define F                           first
#define S                           second
#define ALL(v)                      v.begin(),v.end()
#define SZ(a)                       (int)a.size()
#define FILL(a,b)                   memset(a,b,sizeof(a))
#define SI(n)                       scanf("%d",&n)
#define SLL(n)                      scanf("%lld",&n)
#define PLLN(n)                     printf("%lld\n",n)
#define PIN(n)                      printf("%d\n",n)
#define REP(i,j,n)                  for(LL i=j;i<n;i++)
#define PER(i,j,n)                  for(LL i=n-1;i>=j;i--)
#define endl                        '\n'
#define fast_io                     ios_base::sync_with_stdio(false);cin.tie(NULL)

#define FILEIO(name) \
  freopen(name".in", "r", stdin); \
  freopen(name".out", "w", stdout);

inline int mult(int a , int b) { LL x = a; x *= LL(b); if(x >= MOD) x %= MOD; return x; }
inline int add(int a , int b) { return a + b >= MOD ? a + b - MOD : a + b; }
inline int sub(int a , int b) { return a - b < 0 ? MOD - b + a : a - b; }
LL powmod(LL a,LL b) { if(b==0)return 1; LL x=powmod(a,b/2); LL y=(x*x)%MOD; if(b%2) return (a*y)%MOD; return y%MOD; }

int inv;

inline int get(LL x) {
  x = x % MOD;
  return mult(mult(x , x+1),inv);
}

inline int get(LL l , LL r) {
  return sub(get(r) , get(l-1));
}

int main() {
  inv = powmod(2 , MOD - 2);
```

```
  LL n; SLL(n);
  int ans = 0;
  LL start , end , i = 1;
  while(i <= n) {
    start = i;
    LL x = n/i;
    end = n/x;
    x = get(x);
    i = end + 1;
    ans = add(ans , mult(get(start , end),x));
  }
  PIN(ans);
  return 0;
}
```


COMMENTS (0)

Start Discussion...

Cancel Post