loj6053-简单函数-min25

题目

$$积性函数f(p^c)=p\oplus c,求\sum_{i=1}^{n}f(i)$$

分析

$$易知f(p)=(p-1)+[p=2]*2,则:$$

$$\sum_{p>2}f(p)=\sum_{p>2}p+\sum_{p>2}1=\sum_{i=1}^{n}i*[i\in P]+\sum_{i=1}^{n}[i\in P]=g(n,|P|)+h(n,|P|)$$

特殊处理包含2的时候：在处理S的时候, 如果$mfp=1$(即$j=1$), 说明此时有2, 直接$ans+=2$即可

代码

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int N = 1e6+5;
const int mod = 1e9+7;
int Sqr,zhi[N],pri[N],sp[N],tot,m,id1[N],id2[N],g[N],h[N];
ll n,w[N];
void Sieve(int n){
    zhi[1]=1;
    for (int i=2;i<=n;++i){
        if (!zhi[i]) pri[++tot]=i,sp[tot]=(sp[tot-1]+i)%mod;
        for (int j=1;i*pri[j]<=n;++j){
            zhi[i*pri[j]]=1;
            if (i%pri[j]==0) break;
        }
    }
}
int S(ll x,int y){
    if (x<=1||pri[y]>x) return 0;
    int k=(x<=Sqr)?id1[x]:id2[n/x];
    int res=(1ll*g[k]-h[k]-sp[y-1]+y-1)%mod;res=(res+mod)%mod;
    if (y==1) res+=2;
    for (int i=y;i<=tot&&1ll*pri[i]*pri[i]<=x;++i){
        ll p1=pri[i],p2=1ll*pri[i]*pri[i];
        for (int e=1;p2<=x;++e,p1=p2,p2*=pri[i])
            (res+=(1ll*S(x/p1,i+1)*(pri[i]^e)%mod+(pri[i]^(e+1)))%mod)%=mod;
    }
    return res;
}
int main(){
    scanf("%lld",&n);
    Sqr=sqrt(n);Sieve(Sqr);
    for (ll i=1,j;i<=n;i=j+1){
        j=n/(n/i);w[++m]=n/i;
        if (w[m]<=Sqr) id1[w[m]]=m;
        else id2[n/w[m]]=m;
```

```
        h[m]=(w[m]-1)%mod;
        g[m]=((w[m]+2)%mod)*((w[m]-1)%mod)%mod;
        if (g[m]&1) g[m]+=mod;g[m]/=2;
    }
    for (int j=1;j<=tot;++j)
        for (int i=1;i<=m&&1ll*pri[j]*pri[j]<=w[i];++i){
            int k=(w[i]/pri[j]<=Sqr)?id1[w[i]/pri[j]]:id2[n/(w[i]/pri[j])];
            g[i]=(g[i]-1ll*pri[j]*(g[k]-sp[j-1])%mod)%mod;g[i]=
(g[i]+mod)%mod;
            h[i]=(h[i]-h[k]+j-1)%mod;h[i]=(h[i]+mod)%mod;
        }
    printf("%d\n",S(n,1)+1);
    return 0;
}
```

oiwiki上的代码:

```
/* 「LOJ #6053」简单的函数 */
#include <algorithm>
#include <cmath>
#include <cstdio>

using i64 = long long;

constexpr int maxs = 200000;  // 2sqrt(n)
constexpr int mod = 1000000007;

template <typename x_t, typename y_t>
inline void inc(x_t &x, const y_t &y) {
  x += y;
  (mod <= x) && (x -= mod);
}
template <typename x_t, typename y_t>
inline void dec(x_t &x, const y_t &y) {
  x -= y;
  (x < 0) && (x += mod);
}
template <typename x_t, typename y_t>
inline int sum(const x_t &x, const y_t &y) {
  return x + y < mod ? x + y : (x + y - mod);
}
template <typename x_t, typename y_t>
inline int sub(const x_t &x, const y_t &y) {
  return x < y ? x - y + mod : (x - y);
}
template <typename _Tp>
inline int div2(const _Tp &x) {
  return ((x & 1) ? x + mod : x) >> 1;
}
template <typename _Tp>
inline i64 sqrll(const _Tp &x) {
  return (i64)x * x;
}

int pri[maxs / 7], lpf[maxs + 1], spri[maxs + 1], pcnt;

inline void sieve(const int &n) {
```

```
  for (int i = 2; i <= n; ++i) {
    if (lpf[i] == 0)
      pri[lpf[i] = ++pcnt] = i, spri[pcnt] = sum(spri[pcnt - 1], i);
    for (int j = 1, v; j <= lpf[i] && (v = i * pri[j]) <= n; ++j) lpf[v] =
j;
  }
}

i64 global_n;
int lim;
int le[maxs + 1],  // x \le \sqrt{n}
    ge[maxs + 1];  // x > \sqrt{n}
#define idx(v) (v <= lim ? le[v] : ge[global_n / v])

int G[maxs + 1][2], Fprime[maxs + 1];
i64 lis[maxs + 1];
int cnt;

inline void init(const i64 &n) {
  for (i64 i = 1, j, v; i <= n; i = n / j + 1) {
    j = n / i;
    v = j % mod;
    lis[++cnt] = j;
    idx(j) = cnt;
    G[cnt][0] = sub(v, 1ll);
    G[cnt][1] = div2((i64)(v + 2ll) * (v - 1ll) % mod);
  }
}

inline void calcFprime() {
  for (int k = 1; k <= pcnt; ++k) {
    const int p = pri[k];
    const i64 sqrp = sqrll(p);
    for (int i = 1; lis[i] >= sqrp; ++i) {
      const i64 v = lis[i] / p;
      const int id = idx(v);
      dec(G[i][0], sub(G[id][0], k - 1));
      dec(G[i][1], (i64)p * sub(G[id][1], spri[k - 1]) % mod);
    }
  }
  /* F_prime = G_1 - G_0 */
  for (int i = 1; i <= cnt; ++i) Fprime[i] = sub(G[i][1], G[i][0]);
}

inline int f_p(const int &p, const int &c) {
  /* f(p^{c}) = p xor c */
  return p xor c;
}

int F(const int &k, const i64 &n) {
  if (n < pri[k] || n <= 1) return 0;
  const int id = idx(n);
  i64 ans = Fprime[id] - (spri[k - 1] - (k - 1));
  if (k == 1) ans += 2;
  for (int i = k; i <= pcnt && sqrll(pri[i]) <= n; ++i) {
    i64 pw = pri[i], pw2 = sqrll(pw);
    for (int c = 1; pw2 <= n; ++c, pw = pw2, pw2 *= pri[i])
      ans +=
```

```
          ((i64)f_p(pri[i], c) * F(i + 1, n / pw) + f_p(pri[i], c + 1)) % mod;
  }
  return ans % mod;
}

int main() {
  scanf("%lld", &global_n);
  lim = sqrt(global_n);

  sieve(lim + 1000);
  init(global_n);
  calcFprime();
  printf("%lld\n", (F(1, global_n) + 1ll + mod) % mod);

  return 0;
}
```