签到题 题解

Posted on 2019-04-13

Description：

我们设 $f[i]$ 表示 i 个点的环，每个点染 k 种颜色之一，旋转同构算一种方案的不同的染色方案数，求：

$$\sum_{i=1}^{n} i \times f[i]$$

Solution：

对于 20% 的数据：$1 \leqslant n \leqslant 5, 1 \leqslant k \leqslant 5$

直接爆搜即可。

对于 20% 的数据：$1 \leqslant n \leqslant 50000, 1 \leqslant k \leqslant 10^9$

~~这部分是给根号算法留的，但是由于出题人没有仔细想怎么做，所以就当送分好了。~~

对于 20% 的数据：$1 \leqslant n \leqslant 10^6, 1 \leqslant k \leqslant 10^9$

根据 pólya 定理我们可以知道：

$$f[n] = \frac{1}{n} \sum_{i=1}^{n} k^{\gcd(i,n)}$$

经过简单的推导可以得到：

$$
\begin{aligned}
\sum_{i=1}^{n} i\times f[i] &= \sum_{i=1}^{n}\sum_{d=1}^{i} k^{\gcd(d,i)} && (1)\\
&= \sum_{i=1}^{n}\sum_{d|i} k^{d}\sum_{x=1}^{i}[\gcd(x,i)=d] && (2)\\
&= \sum_{i=1}^{n}\sum_{d|i} k^{d}\sum_{x=1}^{\frac{i}{d}}[\gcd(x,\frac{i}{d})=1] && (3)\\
&= \sum_{i=1}^{n}\sum_{d|i} k^{d}\varphi(\frac{i}{d}) = \sum_{d=1}^{n} k^{d}\sum_{d|i}^{n}\varphi(\frac{i}{d}) && (4)\\
&= \sum_{d=1}^{n} k^{d}\sum_{i=1}^{\lfloor\frac{n}{d}\rfloor}\varphi(i) && (5)
\end{aligned}
$$

根据这个式子就可以线性筛 φ 并预处理前缀和计算，时间复杂度 $O(n)$ 。

```cpp
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstdio>
#include<cmath>
#include<map>
#include<cctype>
#include<cstring>
using namespace std;
inline int rd()
{
    register int res = 0,f = 1;register char c = getchar();
    while(!isdigit(c)){if(c == '-')f = -1;c = getchar();}
    while(isdigit(c))res = (res << 1) + (res << 3) + c - '0',c = getchar();
    return res * f;
}
int n,k;
#define MAXN 1000010
#define MOD 1000000007
bool isprime[MAXN];
int prime[MAXN],tot = 0;
int phi[MAXN],sumphi[MAXN];
int powk[MAXN];
int main()
{
    scanf("%d%d",&n,&k);
    for(int i = 2;i <= n;++i)isprime[i] = true;
    phi[1] = 1;
    for(int i = 2;i <= n;++i)
    {
        if(isprime[i])
        {
            prime[++tot] = i;
            phi[i] = i - 1;
        }
        for(int j = 1;j <= tot && i * prime[j] <= n;++j)
        {
            int k = i * prime[j];
            isprime[k] = false;
            if(i % prime[j] == 0)
            {
                phi[k] = phi[i] * prime[j];
                break;
            }
            else
            {
                phi[k] = phi[i] * phi[prime[j]];
            }
        }
    }
```

```
    for(int i = 1;i <= n;++i)sumphi[i] = (sumphi[i - 1] + phi[i]) % MOD;
    pow k[0] = 1;
    int ans = 0;
    for(int i = 1;i <= n;++i)
    {
        powk[i] = 1ll * powk[i - 1] * k % MOD;
        ans = (ans + 1ll * powk[i] * sumphi[n / i] % MOD) % MOD;
    }
    cout << ans << endl;
    return 0;
}
```

对于 20% 的数据：$1 \leqslant n \leqslant 10^9, 1 \leqslant k \leqslant 10^9$

后面是一个等比数列求和，于是我们只要解决括号内的前半部分即可。

后面那部分可以莫比乌斯反演，得到：

设：

那么：

如果我们用除法分块求g的话，会发现我们要用到的μ的前缀和的位置一定可以表示成$\left\lfloor \frac{n}{d} \right\rfloor$的形式，这些位置的前缀和可以用杜教筛$O(n^{\frac{2}{3}})$一次性预处理出来，那么$g[n]$可以$O(\sqrt{n})$计算，外面再套一个除法分块复杂度是$O(n^{\frac{3}{4}})$的，这个复杂度的证明可以看：这里

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstdio>
#include<cmath>
#include<map>
#include<cctype>
#include<cstring>
using namespace std;
typedef long long ll;
ll n;
int k;
#define MAXN 3000010
#define N 3000000
bool isprime[MAXN];
int prime[MAXN],tot = 0;
int mu[MAXN],summu[MAXN];
#define MOD 1000000007
int power(int a,ll b)
{
    int res = 1;
    while(b > 0)
    {
        if(b & 1)res = 1ll * res * a % MOD;
        a = 1ll * a * a % MOD;
        b = b >> 1;
    }
    return res;
}
int sum(int q,ll n)
{
    if(q == 1)return n % MOD;
    else return 1ll * (power(q,n + 1) - 1) * power(q - 1,MOD - 2) % MOD;
}
#define I inline
#define R register
struct table
{
    #define MO 19260817
    int head[MO];
    ll st[4000];
    int val[4000],nxt[4000];
    int cntnum;
    I int& operator [] (ll x)
    {
        R int modx = x % MO;
        for(R int i = head[modx];i != 0;i = nxt[i])if(st[i] == x)return val[i];
        ++cntnum;st[cntnum] = x;val[cntnum] = 0;nxt[cntnum] = head[modx];head[modx] =
cntnum;
        return val[cntnum];
```

```
    }
    I bool find(ll x)
    {
        R int modx = x % MO;
        for(R int i = head[modx];i != 0;i = nxt[i])if(st[i] == x)return true;
        return false;
    }
}mu_;
int calc(ll n)
{
    if(n <= N)return summu[n];
    if(mu_.find(n))return mu_[n];
    R int ans = 1;
    for(R ll l = 2,r;l <= n;l = r + 1)
    {
        r = n / (n / l);
        ans = (ans - (r - l + 1) % MOD * calc(n / l) % MOD + MOD) % MOD;
    }
    mu_[n] = ans;
    return ans;
}
I int g(ll n)
{
    R int res = 0;
    for(R ll l = 1,r;l <= n;l = r + 1)
    {
        r = n / (n / l);
        res = (res + (calc(r) - calc(l - 1) + MOD) * ((n / l) % MOD) % MOD * ((n / l)
% MOD) % MOD) % MOD;
    }
    return res;
}
int main()
{
    scanf("%lld%d",&n,&k);
    for(R int i = 2;i <= N;++i)isprime[i] = true;
    mu[1] = 1;
    for(R int i = 2;i <= N;++i)
    {
        if(isprime[i])prime[++tot] = i,mu[i] = -1;
        for(R int j = 1;j <= tot && i * prime[j] <= N;++j)
        {
            R int k = i * prime[j];
            isprime[k] = false;
            if(i % prime[j] == 0){mu[k] = 0;break;}
            else mu[k] = -mu[i];
        }
    }
    for(R int i = 1;i <= N;++i)summu[i] = (summu[i - 1] + (mu[i] + MOD) % MOD) % MOD;
    R int ans = 0;
    for(R ll l = 1,r;l <= n;l = r + 1)
```

```
    {
        r = n / (n / l);
        ans = (ans + 1ll * (sum(k,r) - sum(k,l - 1) + MOD) % MOD * g(n / l) % MOD) %
MOD;
    }
    cout << 1ll * power(2,MOD - 2) * ((ans + sum(k,n) - 1) % MOD) % MOD << endl;
    return 0;
}
```

对于 20% 的数据：$1 \leqslant n \leqslant 10^{10}, 1 \leqslant k \leqslant 10^9$

考虑 $1 \leqslant n \leqslant 10^6$ 那部分的做法，求 φ 的前缀和可以杜教筛，然后求和用除法分块优化即可。

时间复杂度 $O(n^{\frac{2}{3}})$ 。

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstdio>
#include<cmath>
#include<map>
#include<cctype>
#include<cstring>
using namespace std;
typedef long long ll;
ll n;
ll k;
#define MAXN 10000010
#define N 10000000
#define MOD 1000000007
bool isprime[MAXN];
ll prime[MAXN],tot = 0;
ll phi[MAXN],sumphi[MAXN];
#define I inline
#define R register
I ll power(ll a,ll b)
{
    R ll res = 1;
    while(b > 0)
    {
        if(b & 1)res = 1ll * res * a % MOD;
        a = 1ll * a * a % MOD;
        b = b >> 1;
    }
    return res;
}
I ll sum(ll q,ll n)
{
    if(q == 1)return n % MOD;
    else return 1ll * (power(q,n + 1) - 1) * power(q - 1,MOD - 2) % MOD;
}
/*struct table
{
    #define MO 19260817
    int head[MO];
    ll st[4000];
    int val[4000],nxt[4000];
    int cntnum;
    I int& operator [] (ll x)
    {
        R int modx = x % MO;
        for(R int i = head[modx];i != 0;i = nxt[i])if(st[i] == x)return val[i];
        ++cntnum;
        st[cntnum] = x;val[cntnum] = 0;nxt[cntnum] = head[modx];
        head[modx] = cntnum;
```

```
        return val[cntnum];
    }
    I bool find(ll x)
    {
        R int modx = x % MO;
        for(R int i = head[modx];i != 0;i = nxt[i])if(st[i] == x)return true;
        return false;
    }
}phi_;*/
map<ll,ll> phi_;
I ll calc(ll n)
{
    if(n == 1)return 1;
    if(n <= N)return sumphi[n];
    if(phi_.find(n) != phi_.end())return phi_[n];
    R ll sum;
    if(n % 2 == 0)sum = (n / 2) % MOD * (n % MOD + 1) % MOD;
    else sum = n % MOD * ((n + 1) / 2 % MOD) % MOD;
    for(R ll l = 2,r;l <= n;l = r + 1)
    {
        r = n / (n / l);
        sum = (sum - 1ll * (r - l + 1) % MOD * calc(n / l) % MOD + MOD) % MOD;
    }
    phi_[n] = sum;
    return sum;
}
int main()
{
    scanf("%lld%d",&n,&k);
    for(R int i = 2;i <= N;++i)isprime[i] = true;
    phi[1] = sumphi[1] = 1;
    for(R int i = 2;i <= N;++i)
    {
        if(isprime[i])prime[++tot] = i,phi[i] = i - 1;
        for(R int j = 1;j <= tot && i * prime[j] <= N;++j)
        {
            R int k = i * prime[j];
            isprime[k] = false;
            if(i % prime[j] == 0){phi[k] = phi[i] * prime[j];break;}
            else phi[k] = phi[i] * phi[prime[j]];
        }
    }
    for(R int i = 1;i <= N;++i)sumphi[i] = (sumphi[i - 1] + phi[i]) % MOD;
    R ll ans = 0;
    for(R ll l = 1,r;l <= n;l = r + 1)
    {
        r = n / (n / l);
        ans = (ans + 1ll * (sum(k,r) - sum(k,l - 1) + MOD) * calc(n / l) % MOD) % MOD;
    }
    cout << ans << endl;
```

```
    return 0;
}
```