杜教筛

用于求积性函数的前缀和。

设 $S(n)=\sum\limits_{i=1}^{n}f(i)$ 为数论函数 $f(i)$ 的前缀和。我们试图构造 $S(n)$ 关于 $S\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$ 的递归式：

设 $g(n)$ 是一个数论函数，则：

$$
\begin{aligned}
\sum_{i=1}^{n}(f*g)(i) &= \sum_{i=1}^{n}\sum_{d\mid i}f\left(\frac{i}{d}\right)g(d) \\
&= \sum_{d=1}^{n}\sum_{i=1}^{\left\lfloor\frac{n}{d}\right\rfloor}f(i)g(d) \\
&= \sum_{d=1}^{n}g(d)S\left(\left\lfloor\frac{n}{d}\right\rfloor\right)
\end{aligned}
$$

于是乎，

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

如果我们能够快速求得 $\sum\limits_{i=1}^{n}(f*g)(i)$，并且数论分块求得 $\sum\limits_{i=2}^{n}g(i)S\left(\left\lfloor\frac{n}{d}\right\rfloor\right)$，那么我们就可以得到 $S(n)$.

莫比乌斯函数前缀和

由于 $\mu*1=\epsilon$，根据杜教筛的理论，有：

$$
S_{\mu}(n)=1-\sum_{i=2}^{n}S_{\mu}\left(\left\lfloor\frac{n}{i}\right\rfloor\right)
$$

对其进行数论分块，时间复杂度为 $O\left(n^{\frac{3}{4}}\right)$.

但我们还可以继续优化，如果对于前 $n^{\frac{2}{3}}$ 项直接用线性筛预处理，只对剩余项进行杜教筛，那么复杂度降到 $O\left(n^{\frac{2}{3}}\right)$.

ATT：代码中使用 `unordered_map` 存储较大的 n 对应的 $S_{\mu}(n)$，而对于较小的 n 则直接存在一个数组里，因为 `onordered_map` 的建立常数很大。

> 杜教筛=狄利克雷卷积+整除分块+线性筛。

欧拉函数前缀和

欧拉函数前缀和可以直接从莫比乌斯函数前缀和得到：

$$
\sum_{i=1}^{n}\varphi(i)=\sum_{i=1}^{n}\sum_{d\mid i}\mu(d)\cdot\frac{i}{d}=\sum_{d=1}^{n}\sum_{i=1}^{\left\lfloor\frac{n}{d}\right\rfloor}i\cdot\mu(d)=\sum_{d=1}^{n}\mu(d)\left\lfloor\frac{n}{d}\right\rfloor
$$

数论分块结合杜教筛求莫比乌斯函数前缀和即可。

> 当然也可以直接杜教筛，由 $\mathrm{id}=\varphi*1$，根据杜教筛的理论，有：
>
> $$S_{\varphi}(n)=\frac{n(n+1)}{2}-\sum_{i=2}^{n}S_{\varphi}\left(\left\lfloor\frac{n}{i}\right\rfloor\right)$$

```
int mu[N], pList[N], pID;
bool notP[N];
```

```cpp
void Euler(int n){
    notP[0] = notP[1] = 1;
    mu[1] = 1;
    for(int i = 1; i <= n; i++){
        if(notP[i] == 0){
            pList[++pID] = i;
            mu[i] = -1;
        }
        for(int j = 1; j <= pID; j++){
            if(1ll * i * pList[j] > n)  break;
            notP[i * pList[j]] = 1;
            if(i % pList[j] == 0){
                mu[i * pList[j]] = 0;
                break;
            }
            else    mu[i * pList[j]] = -mu[i];
        }
    }
}

unordered_map<LL, LL> muS;
LL preS[N];
LL apiadu_mu(LL n){
    if(n <= 5000000)    return preS[n];
    if(muS.find(n) != muS.end())    return muS[n];
    LL res = 1;
    for(LL l = 2, r; l <= n; l = r + 1){
        r = n / (n / l);
        res -= apiadu_mu(n / l) * (r - l + 1);
    }
    return muS[n] = res;
}
LL apiadu_phi(LL n){
    LL res = 0;
    for(LL l = 1, r; l <= n; l = r + 1){
        r = n / (n / l);
        res += (__int128)(n/l)*(n/l+1)/2 * (apiadu_mu(r) - apiadu_mu(l-1));
    }
    return res;
}

int main(){
    Euler(5000000);
    for(int i = 1; i <= 5000000; i++)
        preS[i] = preS[i-1] + mu[i];
    int T; for(scanf("%d", &T); T; T--){
        LL n;
        scanf("%lld", &n);
        printf("%lld %lld\n", apiadu_phi(n), apiadu_mu(n));
    }
    return 0;
}
```

其他积性函数前缀和

只要能找到合适的 $g(n)$ 使得 $(f*g)(n)$ 的前缀和易于计算、就可以使用杜教筛。