

杜教筛

用于求积性函数的前缀和。

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

设 $g(n)$ 是一个数论函数，则：

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

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 $\sum_{i=1}^n (f * g)(i)$ ，并且数论分块求得 $\sum_{i=2}^n g(i)S\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$ ，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 `unordered_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：

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

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 $\text{id} = \varphi * 1$ ，根据杜教筛的理论，有：

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

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

```

3 void Euler(int n){
4     notP[0] = notP[1] = 1;
5     mu[1] = 1;
6     for(int i = 1; i <= n; i++){
7         if(notP[i] == 0){
8             pList[++pID] = i;
9             mu[i] = -1;
10        }
11        for(int j = 1; j <= pID; j++){
12            if(1ll * i * pList[j] > n) break;
13            notP[i * pList[j]] = 1;
14            if(i % pList[j] == 0){
15                mu[i * pList[j]] = 0;
16                break;
17            }
18            else mu[i * pList[j]] = -mu[i];
19        }
20    }
21 }
22
23 unordered_map<LL, LL> muS;
24 LL preS[N];
25 LL apiadu_mu(LL n){
26     if(n <= 5000000) return preS[n];
27     if(muS.find(n) != muS.end()) return muS[n];
28     LL res = 1;
29     for(LL l = 2, r; l <= n; l = r + 1){
30         r = n / (n / l);
31         res -= apiadu_mu(n / l) * (r - l + 1);
32     }
33     return muS[n] = res;
34 }
35 LL apiadu_phi(LL n){
36     LL res = 0;
37     for(LL l = 1, r; l <= n; l = r + 1){
38         r = n / (n / l);
39         res += (__int128)(n/l)*(n/l+1)/2 * (apiadu_mu(r) - apiadu_mu(l-1));
40     }
41     return res;
42 }
43
44 int main(){
45     Euler(5000000);
46     for(int i = 1; i <= 5000000; i++){
47         preS[i] = preS[i-1] + mu[i];
48         int T; for(scanf("%d", &T); T; T--){
49             LL n;
50             scanf("%lld", &n);
51             printf("%lld %lld\n", apiadu_phi(n), apiadu_mu(n));
52         }
53     }
54     return 0;
55 }

```

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。