1 杜教筛

一种求积性函数的前缀和的方法

1.1 一些前置知识

狄利克雷乘积 Dirichlet product

对两个数论函数 f, g，定义狄利克雷乘积为 $(f*g)(n)=\sum\limits_{d|n} f(d)g(\frac{n}{d})$

该运算满足

交换律 $f*g=g*f,$

结合律 $(f*g)*h=f*(g*h),$

分配律 $(f+g)*h=f*h+g*h.$

常见数论函数

$I(n)=1$ $\quad \epsilon(n)=[n=1]$ $\quad id(n)=n$ $\quad \varphi(n)$ 欧拉函数 $\quad \mu(n)$ 莫比乌斯反函数

1.2 方法推导

需计算 $F(n)=\sum\limits_{i=1}^{n} f(n)$

其中 f 是积性函数。

若有积性函数 h, g，满足 $h=f*g$，

$$\sum_{i=1}^{n} h(i)=\sum_{i=1}^{n}(f*g)(i)=\sum_{i=1}^{n}\sum_{d|i} f(\frac{i}{d})g(d)$$

于是改为枚举 d，和 d 的 k 倍，用 $d\cdot k$ 替换右式中的 i，

$$\sum_{i=1}^{n} h(i)=\sum_{d=1}^{n}\sum_{k=1}^{\lfloor\frac{n}{d}\rfloor} f(\frac{kd}{d})g(d)=\sum_{d=1}^{n} g(d)\sum_{k=1}^{\lfloor\frac{n}{d}\rfloor} f(k)=\sum_{d=1}^{n} g(d)F(\lfloor\frac{n}{d}\rfloor)$$

提出 $d=1$ 项，

$$\sum_{i=1}^{n} h(i)=g(1)F(n)+\sum_{d=2}^{n} g(d)F(\lfloor\frac{n}{d}\rfloor)$$

$$F(n)g(1)=\sum_{i=1}^{n} h(i)-\sum_{d=2}^{n} g(d)F(\lfloor\frac{n}{d}\rfloor) \tag{1}$$

我们发现，(1)式的第三项中含有向下取整，因此可以考虑是使用整除分块和递归。提前预处理 h, g 在小范围的值（可以线性筛）作为递归边界。

至此，问题转化为，寻找合适的 h, g 函数使得可以快速得应用(1)式进行运算。

1.3 实例

求函数 μ 的前缀和

函数 μ 满足 $\sum\limits_{d|n}\mu(d)=[n=1]$。考虑函数 I 和函数 ϵ，易证 $\epsilon=\mu*I$。带入(1)式

$$F(n)=1-\sum_{d=2}^{n} F(\lfloor\frac{n}{d}\rfloor)$$

求函数 φ 的前缀和

φ 的性质：$\sum_{d|n}\varphi(d)=n$。考虑函数 I 和函数 $id(n)$，

$$F(n)=\sum_{i=1}^{n}i-\sum_{d=2}^{n}F(\lfloor\frac{n}{d}\rfloor)=\frac{n(n+1)}{2}-\sum_{d=2}^{n}F(\lfloor\frac{n}{d}\rfloor)$$

洛谷模板代码 (C++)

```cpp
#include <cstdio>
#include <unordered_map>
using namespace std;
using ll = long long;
const int N = 5000000;
int prime[N], cnt = 0;
ll phi[N + 10], mu[N + 10];
void Euler() {
        phi[1] = 1; mu[1] = 1;
        for (int i = 2; i <= N; ++i) {
                if (!phi[i]) {
                        prime[cnt++] = i; phi[i] = i - 1; mu[i] = -1;
                }
                for (int j = 0; j != cnt && prime[j] * i <= N; ++j) {
                        if (i % prime[j]) {
                                phi[prime[j] * i] = phi[i] * phi[prime[j]];
                                mu[prime[j] * i] = mu[i] * mu[prime[j]];
                        }
                        else {
                                phi[prime[j] * i] = phi[i] * prime[j];
                                mu[prime[j] * i] = 0;
                        }
                }
        }
        for (int i = 1; i <= N; ++i) {
                phi[i] += phi[i - 1]; mu[i] += mu[i - 1];
        }
}
unordered_map<int, ll> fp, fm;
ll getPhiSum(int n) {
        if (n <= N) return phi[n];
        if (fp[n]) return fp[n];
        ll res = (long long)n * (n + 1) / 2;
        for (int l = 2, r = 0; l <= n; l = r + 1) {
                r = n / (n / l);
                res -= (r - l + 1) * getPhiSum(n / l);
        }
        return fp[n] = res;
}
ll getMuSum(int n) {
        if (n <= N) return mu[n];
        if (fm[n]) return fm[n];
        ll res = 1;
        for (int l = 2, r = 0; l <= n; l = r + 1) {
                r = n / (n / l);
                res -= (r - l + 1) * getMuSum(n / l);
        }
        return fm[n] = res;
}
int main() {
        Euler();
        int T = 0, n = 0;
        scanf("%d", &T);
        for (int t = 1; t <= T; ++t) {
                scanf("%d", &n);
                printf("%lld %lld\n", getPhiSum(n), getMuSum(n));
        }
        return 0;
}
```