莫比乌斯反演笔记

数学 组合数学 莫比乌斯反演

$d(ij)$是ij的因数
$\sum_{i=1}^{n}\sum_{j=1}^{m}d(ij)=\sum_{i=1}^{n}\sum_{j=1}^{m}\lfloor\frac{n}{i}\rfloor\lfloor\frac{m}{j}\rfloor[gcd(i,j)==1]$
$d(nm)=\sum_{i|n}\sum_{j|m}1[gcd(i,j==1)]$

$\sum_{n=1}^{N}\sum_{i|n}=\sum_{i=1}^{N}\lfloor\frac{N}{i}\rfloor$ 两者表示的都是从1到N的因数的个数，可以线性筛$O(n)$求解

导入

通俗的讲解：http://www.cnblogs.com/chenyang920/p/4811995.html
简单的证明：http://blog.csdn.net/outer_form/article/details/50588307

莫比乌斯反演的实质是容斥原理。
(1)
$G(n)=\sum_{d|n}F(d)=>$
$F(n)=\sum_{d|n}\mu(\frac{n}{d})G(d)$
$F(n)=\sum_{d|n}\mu(d)G(\frac{n}{d})$
(2)
$G(n)=\sum_{n|d}F(d)=>$
$F(n)=\sum_{n|d}\mu(\frac{d}{n})G(d)$

莫比乌斯函数$\mu(x)$的含义可以这样理解：$u(x)$ 是进行容斥时，$G(1)$的系数
学会在求和式中改变循环变量的顺序，以及适当的换元

莫比乌斯函数性质

设$n=p_1^{k_1}\cdot p_2^{k_2}\dots p_m^{k_m}$

$$\mu(n)=\begin{cases}1 & n=1\\ (-1)^m & \prod_{i=1}^{m}k_i=1\\ 0 & otherwise(k_i>1)\end{cases}$$

性质一：积性函数
故可以在线性筛出
性质二
$\sum_{d|n}\mu(d)=[n==1]$
只有$n=1$时上面式子等号左边才为1，否则为0.

简单应用

hdoj 1695
题目传送门：http://acm.hdu.edu.cn/showproblem.php?pid=1695

题目大意是求$i\in[1,b]$和$j\in[1,d]$中$gcd(i,j)=k$的对数，其中$gcd(i,j)$和$gcd(j,i)$看作是一对。

先做一个小trick，将问题转换成$i \in [1, b/k]$和$j \in [1, d/k]$中满足$gcd(i, j) = 1$的对数

这道题用分解质因数+容斥也可以解决，但是相比而言用莫比乌斯反演耗时显著减少。

设$f(n)$为符合条件最大公因数为n的对数
设$F(n)$为符合条件公因数包含n的对数

显然，有$F(n) = \sum_{n|d} f(d)$，而且$F(n)$是易求的，$F(n) = \lfloor \frac{b}{n} \rfloor \times \lfloor \frac{d}{n} \rfloor$，且$f(1)$即为所求。

由莫比乌斯反演的第二种形式，得
$f(n) = \sum_{n|d} \mu(\frac{d}{n}) F(d)$
即$f(1) = \sum_{i=1} \mu(i) F(i)$
由于这样计算会把(i, j)和(j, i)算成两对，因此再去一下重即可

```
#include <bits/stdc++.h>
using namespace std;
const int MAX = 1e6 + 10;
int cnt, prime[MAX], vis[MAX], mu[MAX];

void init()
{
    memset(vis, 0, sizeof vis);
    mu[1] = 1;
    cnt = 0;
    for (int i = 2; i < MAX; i ++) {
        if (!vis[i]) {
            prime[cnt ++] = i;
            mu[i] = -1;
        }
        for (int j = 0; j < cnt; j ++) {
            if (prime[j] * i >= MAX) break;
            vis[i * prime[j]] = 1;
            if (i % prime[j]) mu[i * prime[j]] = - mu[i];
            else {
                mu[i * prime[j]] = 0;
                break;
            }
        }
    }
}

int main()
{
    init();
    int t, tc = 1;
    scanf("%d", &t);
    while (t --) {
        int b, d, k;
        scanf("%*d%d%*d%d%d", &b, &d, &k);
        if (k == 0) {
            printf("Case %d: %d\n", tc ++, 0);
            continue;
        }
        b /= k;
        d /= k;
        if (b < d) swap(b, d);
        long long ans = 0, t = 0;
        for (int i = 1; i <= b; i ++) {
            ans += 1LL * mu[i] * (b / i) * (d / i);
        }
        for (int i = 1; i <= d; i ++) {
            t += 1LL * mu[i] * (d / i) * (d / i);
        }
        ans -= t / 2;
        printf("Case %d: %lld\n", tc ++, ans);
    }
    return 0;
}
```

利用莫比乌斯反演求欧拉函数

组合数学 P122

上一题中我们用到的小trick其实非常有用。
设$S_n^k=\sum_{i=1}^n[gcd(i,n)=k]$，由欧拉函数的定义可知，$\varphi(n)=S_n^1$

有一个很显然的事实，若$gcd(a,b)=k$，那么$gcd(a/k,b/k)=1$，
也就是说$S_n^k=S_{n/k}^1=\varphi(\frac{n}{k})$(其中$k|n$，否则$S_n^k=0$而$S_{n/k}^1\neq 0$)

令$\varphi(n)$为反演中的$f(n)$，
$F(n)=\sum_{d|n}\varphi(d)$
$F(n)$是什么呢？

容易得$\sum_{k=1}^n S_n^k=n$，而由上面的论述可知

$$S_n^k=\begin{cases}0 & k\nmid d\\ S_{n/k}^1 & k\mid d\end{cases}$$

所以
$\sum_{k=1}^n S_n^k=\sum_{d|n}S_{n/d}^1=\sum_{d|n}\varphi(\frac{n}{d})=n$
由上可得
$\sum_{d|n}\varphi(d)=\sum_{d|n}\varphi(\frac{n}{d})=n$
反演得
$\varphi(n)=\sum_{d|n}\mu(\frac{n}{d})d=\sum_{d|n}\mu(d)\frac{n}{d}$
由莫比乌斯函数的性质可以知道，$\mu(d)\neq 0\iff d=1$或d是互不相同素数的积
$\varphi(n)=n-(n/p_1+n/p_2+\ldots)+(n/(p_1p_2)+n/(p_1p_3))-\ldots+(-1)^r\times n/(p_1p_2\ldots p_r)$
即
$\varphi(n)=n\Pi_{i=1}^r(1-\frac{1}{p_i})$

再看hdoj 6134
题目传送门：http://acm.hdu.edu.cn/showproblem.php?pid=6134

就是求$ans(n)=\sum_{i=1}^n\sum_{j=1}^i\lceil\frac{i}{j}\rceil[gcd(i,j)=1]$
看似和上面的例子没有什么关系，但是反演的时候却用到了相同的思想。

由于$gcd(i,j)=1$，所以$\lceil\frac{i}{j}\rceil=i/j+1\ \ (j\neq 1)$
故$ans(n)=\sum_{i=1}^n(\sum_{j=1}^i\lfloor\frac{i}{j}\rfloor[gcd(i,j)=1]+\varphi(i)-1)$
欧拉函数比较好求，如何求其他部分？

仔细观察式子，你有没有发现S_n^k的影子？
设$f(n)=\sum_{i=1}^n\lfloor\frac{n}{i}\rfloor[gcd(i,n)=1]$
这样是不是清晰一些，式子中加数的个数就是S_n^1，这个式子相当于给S_n^k“加了权”，而且这个“权”相当有特点：
设

$F(n)=\sum_{d|n}f(d)$

$=\sum_{d|n}\sum_{i=1}^{d}\lfloor\frac{d}{i}\rfloor[gcd(i,d)=1]$

设 $kd=n$

原式 $=\sum_{d|n}\sum_{i=1}^{d}\lfloor\frac{kd}{ki}\rfloor[gcd(ki,kd)=k]$

$=\sum_{d|n}\sum_{i=1}^{n}\lfloor\frac{n}{i}\rfloor[gcd(i,n)=d]$

是不是很熟悉？就是把k乘上去，把$S^1_{n/d}$变成S^k_n，恰好我们的“权”并不改变

$$S^k_n=\begin{cases}0 & k\nmid d\\ S^1_{n/k} & k\mid d\end{cases}$$

我们有 $\sum_{k|n}S^k_n=n$

所以原式 $=\sum_{i=1}^{n}\lfloor\frac{n}{i}\rfloor$

反演：

$f(n)=\sum_{d|n}\mu(\frac{n}{d})F(d)$

代入：

$ans(n)=\sum_{i=1}^{n}(f(i)+\varphi(i)-1)$

或：

$f(n)$

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

$=\sum_{i=1}^{n}\lfloor\frac{n}{i}\rfloor\sum_{d|gcd(i,n)}\mu(d)$

$=\sum_{d|n}$

```
#include <bits/stdc++.h>
using namespace std;
const int MAX = 1e6 + 10;
const int mod = 1e9 + 7;
int vis[MAX], mu[MAX], fac[MAX], prime[MAX], phi[MAX], d[MAX], cnt, ans[MAX];

void init()
{
    mu[1] = 1;
    phi[1] = 1;
    fac[1] = 1;
    for (int i = 2; i < MAX; i ++) {
        if (!vis[i]) {
            prime[cnt ++] = i;
            phi[i] = i - 1;
            mu[i] = -1;
            fac[i] = 2;
            d[i] = 1;
        }
        for (int j = 0; j < cnt; j ++) {
            if (1LL * i * prime[j] >= MAX) break;
            vis[i * prime[j]] = 1;
            if (i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j];
                mu[i * prime[j]] = 0;
                fac[i * prime[j]] = fac[i] / (d[i] + 1) * (d[i] + 2);
                d[i * prime[j]] = d[i] + 1;
                break;
            } else {
                phi[i * prime[j]] = phi[i] * (prime[j] - 1);
                mu[i * prime[j]] = -mu[i];
                fac[i * prime[j]] = fac[i] * 2;
                d[i * prime[j]] = 1;
            }
        }
    }
}
```

```
void table()
{
    for (int i = 1; i < MAX; i ++) {
        fac[i] = (fac[i - 1] + fac[i]) % mod;
    }
    for (int i = 1; i < MAX; i ++) {
        for (int j = i; j < MAX; j += i) {
            if (mu[j / i] == -1) {
                ans[j] = ((ans[j] - fac[i]) % mod + mod) % mod;
            } else if (mu[j / i] == 1) {
                ans[j] = (ans[j] + fac[i]) % mod;
            }
        }
    }
    for (int i = 1; i < MAX; i ++) {
        ans[i] = (ans[i] - 1 + phi[i]) % mod;
        ans[i] = (ans[i] + ans[i - 1]) % mod;
    }
}
int main()
{
    init();
    table();
    int n;
    while (~scanf("%d", &n)) {
        printf("%d\n", ans[n]);
    }
    return 0;
}
```

还是套路

再看一题：
hdoj 5321
题目传送门：http://acm.hdu.edu.cn/showproblem.php?pid=5321

题目大意是给你一个多重集合，有两个很闲的人分别定义了多重集合的beautiful value

第一个人这样定义：
先定义排列的beautiful value：是所有连续区间gcd的和。而多重集合的beautiful value则是集合元素全排列的所有排列beautiful value的和（做全排列时相同元素视为不同）

第二个人这样定义：
从集合中任选k个元素，这k个元素的beautiful value是$k * gcd$（$a_1, a_2, ..., a_k$），集合的beautiful value是所有可能选择的k个元素的beautiful value（选择时集合中相同数字仍然看作不同）。

先看第一个
暴力肯定不行，看起来比较靠谱的是看看多少个区间的gcd恰好为g.
假如说恰好有k个数字的$gcd = g$, 那么这k个数字对答案的贡献为$g * k! * (n-k)! * (n-k+1)$,怎么讲呢？$k$个数字的全排列，$n-k$个数字的全排列，还要考虑$n-k$个数字所产生的$n-k+1$个间隔，最后乘$g$.

怎么找这k个数字呢？很显然，只有若干个g的倍数求gcd之后才有可能是g，但是，这样求出的答案也有可能是g的整数倍。想到了吗？没想到可以回头去看看第一个题。

设$f(n)$是gcd等于n的区间个数，$F(n) = \sum_{n|d} f(n)$
（由于反演的时候无法考虑g, 不妨先把g忽略，反演求出$f(n)$后再考虑g）
设$cnt[i]$为集合中i和i的倍数的个数，则
$F(n) = \sum_{i=1}^{cnt[n]} A_{cnt[n]}^{i} (n-i+1)!$
反演得，
$f(n) = \sum_{n|d} \mu(\frac{d}{n}) F(d)$
那么，
$ans1 = \sum_{i=1} i f(i)$

第二个人怎么算呢？
和第一个相同的思路，考虑选出的k个数的gcd是g的倍数，这样其对答案的贡献是：
$\sum_{i=1}^{cnt[g]} iC_{cnt[g]}^{i} = cnt[g] * 2^{cnt[g]-1}$
（将组合数写成阶乘形式就可得出）

$$F(n) = cnt[n] * 2^{cnt[n]-1}$$
$$f(n) = \sum_{n|d} \mu(\frac{d}{n})F(d)$$

```
#include <bits\stdc++.h>
using namespace std;
const long long mod = 258280327LL;
const int MAX = 1e5 + 10;
int n, maxx, vis[MAX], mu[MAX], prime[MAX], pcnt;
long long f[2][MAX], fac[MAX], inv[MAX], cnt[MAX];

long long p(long long base, long long r)
{
    long long ret = 1;
    while (r) {
        if (r & 1) {
            ret = (ret * base) % mod;
        }
        base = (base * base) % mod;
        r >>= 1;
    }
    return ret;
}

void init()
{
    fac[0] = fac[1] = 1;
    inv[0] = inv[1] = 1;
    for (int i = 2; i < MAX; i ++) {
        fac[i] = (1LL * fac[i - 1] * i) % mod;
        inv[i] = p(fac[i], mod - 2);
    }
    mu[1] = 1;
    for (int i = 2; i < MAX; i ++) {
        if (!vis[i]) {
            prime[pcnt ++] = i;
            mu[i] = -1;
        }
        for (int j = 0; j < pcnt; j ++) {
            if (1LL * prime[j] * i >= MAX) {
                break;
            }
            vis[i * prime[j]] = 1;
            if (i % prime[j] == 0) {
                mu[i * prime[j]] = 0;
                break;
            } else {
                mu[i * prime[j]] = - mu[i];
            }
        }
    }
}
inline long long A(int n, int m)
{
    return (fac[n] * inv[n - m]) % mod;
}
void solve()
{
    memset(f, 0, sizeof f);
    for (int i = 1; i <= maxx; i ++) {
        if (cnt[i]) {
            for (int j = 1; j <= cnt[i]; j ++) {
                f[0][i] = (f[0][i] + A(cnt[i], j) * fac[n - j + 1]) % mod;
            }
            f[1][i] = (f[1][i] + cnt[i] * p(2, cnt[i] - 1)) % mod;
        }
    }
}
int main()
```

```
{
    init();
    while (~scanf("%d", &n)) {
        memset(cnt, 0, sizeof cnt);
        maxx = -1;
        int t;
        for (int i = 1; i <= n; i ++) {
            scanf("%d", &t);
            maxx = max(t, maxx);
            ++ cnt[t];
        }
        for (int i = 1; i <= maxx; i ++) {
            for (int j = 2 * i; j <= maxx; j += i) {
                cnt[i] = (cnt[i] + cnt[j]) % mod;
            }
        }
        solve();
        long long ans0 = 0, ans1 = 0;
        for (int i = 1; i <= maxx; i ++) {
            long long t0 = 0, t1 = 0;
            for (int j = i; j <= maxx; j += i) {
                t0 = (t0 + mu[j / i] * f[0][j]) % mod;
                t1 = (t1 + mu[j / i] * f[1][j]) % mod;
            }
            ans0 = (ans0 + 1LL * t0 * i) % mod;
            ans1 = (ans1 + 1LL * t1 * i) % mod;
        }
        if (ans0 > ans1) {
            printf("Mr. Zstu %lld\n", ans0);
        } else if (ans0 < ans1) {
            printf("Mr. Hdu %lld\n", ans1);
        } else {
            printf("Equal %lld\n", ans0);
        }
    }
}
```