

基于值域预处理的快速 GCD

算法简介

$O(V)$ 预处理 $O(1)$ 查询 gcd

算法实现

首先考虑将每个数 n 分解成 a, b, c 满足 $a \leq b \leq c$ 且若 $c > \sqrt{n}$ 则 c 一定是质数。

利用线性筛处理上述过程，线性筛会枚举 n 的最小素因子 p 假设 $\frac{n}{p}$ 的分解为 a', b', c' 则 n 的分解为 $a'p, b', c'$

下面进行正确性的证明：

若 $p \leq n^{\frac{1}{4}}$ 由于 $a' \leq (\frac{n}{p})^{\frac{1}{3}}$ 于是 $a'p \leq \sqrt{n}$ 成立。

若 $p > n^{\frac{1}{4}}$ 由于 p 是 n 的最小素因子，所以 n 的其他因子不超过两个，所以 $a'=1$ 即 $a'p$ 是素数，也成立。

接下来考虑计算 $\text{gcd}(x, y)$ 设 $y = abc$ 于是可以先计算 x, a 的公因数，再令 x 除以 a 依次处理 b, c 的贡献。

于是只需要考虑 $\text{gcd}(x, a) = \text{gcd}(x \bmod a, a)$ 如何计算，根据分解规则知 a 要么为素数要么不超过 $\sqrt{V}$

若 a 是素数只需要考虑是否整除即可，否则可以 $O(V)$ 预处理出任意两个不超过 $\sqrt{V}$ 的数的 gcd 然后 $O(1)$ 查询。

算法模板

[洛谷p5435](#)

```
const int MAXV=1e6+5,MAXB=1e3+5;
namespace GCD{
    int prime[MAXV],pcnt,f[MAXV][3],_gcd[MAXB][MAXB];
    bool vis[MAXV];
    void init(){
        f[1][0]=f[1][1]=f[1][2]=1;
        _for(i,2,MAXV){
            if(!vis[i]){
                prime[pcnt++]=i;
                f[i][0]=f[i][1]=1;
                f[i][2]=i;
            }
        }
    }
}
```

```

        for(int j=0;j<pcnt&&prime[j]<MAXV;j++){
            int k=i*prime[j];
            vis[k]=true;
            f[k][0]=f[i][0]*prime[j];
            f[k][1]=f[i][1];
            f[k][2]=f[i][2];
            if(f[k][0]>f[k][1])swap(f[k][0],f[k][1]);
            if(f[k][1]>f[k][2])swap(f[k][1],f[k][2]);
            if(i%prime[j]==0)break;
        }
    }
    _for(i,1,MAXB)_gcd[i][0]=_gcd[0][i]=i;
    _for(i,1,MAXB){
        _rep(j,1,i)
        _gcd[i][j]=_gcd[j][i]=_gcd[j][i%j];
    }
}

int gcd(int a,int b){
    int ans=1;
    _for(i,0,3){
        int g;
        if(f[a][i]>=MAXB)
            g=(b%f[a][i])?1:f[a][i];
        else
            g=_gcd[f[a][i]][b%f[a][i]];
        ans*=g;
        b/=g;
    }
    return ans;
}

}

const int MAXN=5e3+5,mod=998244353;
int a[MAXN],b[MAXN];
int main(){
    GCD::init();
    int n=read_int();
    _rep(i,1,n)
    a[i]=read_int();
    _rep(i,1,n)
    b[i]=read_int();
    _rep(i,1,n){
        int ans=0;
        for(int j=n;j;j-- )
            ans=1LL*(ans+GCD::gcd(a[i],b[j]))*i%mod;
        enter(ans);
    }
    return 0;
}

```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E5%9F%BA%E4%BA%8E%E5%80%BC%E5%9F%9F%E9%A2%84%E5%A4%84%E7%90%86%E7%9A%84%E5%BF%AB%E9%80%9F_gcd

Last update: 2021/09/04 21:41