浅谈一种互质数对与最大公约数的维护算法

南京外国语学校 杜冠成

Overview

摘要

- 本文首创了一种互质数对与最大公约数的维护算法，并将后者命名为 Dynamic-gcd algorithm，同时结合例题对这两个算法的应用加以阐述，以体现其广泛的应用前景。

定义与约定

- $Prime$ 为所有质数构成的集合。
- $P(x)$ 表示 x 的最大质因子。

引入

- 给定 n ，你需要对初始所有元素均为 1 的序列 $a_1, a_2, \cdots, a_n$ 进行若干次修改，使得 $\forall i \in [1, n]$，存在一个时刻满足 $\forall a_j = [\gcd(i, j) = 1]$。

维护方式

- 注意到，i 与 j 不互质，当且仅当 $\exists p \in Prime, p|i, p|j$。

维护方式

- 注意到，i 与 j 不互质，当且仅当 $\exists p \in Prime, p|i, p|j$。
- 考虑列出 n 以内的所有质数，并采用暴力搜索的方式，对于每个质数 p，考虑 p 是否能整除 i，并将其作为两个分支继续搜索下去。

维护方式

- 注意到，i 与 j 不互质，当且仅当 $\exists p \in Prime, p|i, p|j$。
- 考虑列出 n 以内的所有质数，并采用暴力搜索的方式，对于每个质数 p，考虑 p 是否能整除 i，并将其作为两个分支继续搜索下去。
- 具体来说，我们在搜索的过程中，维护 x_0, i_0 及序列 a，表示目前考虑到了第 x_0 小的质数，且钦定需要整除 i 的所有质数乘积为 i_0。初始 $x_0 = i_0 = 1$ 且 a 中所有元素均为 1。令 p_0 为第 x_0 小的质数，在 $i_0 p_0 \le n$ 的前提下，若钦定 p_0 整除 i，则将 $a_{p_0}, a_{2p_0}, \cdots$ 赋值为 0，并向分支 $(x_0', i_0') = (x_0 + 1, i_0 p_0)$ 继续搜索；然后，从 (x_0', i_0') 回溯后，撤销对 $a_{p_0}, a_{2p_0}, \cdots$ 的修改，并向分支 $(x_0 + 1, i_0)$ 搜索。

正确性证明

- 可以发现，若在搜索过程中，所有钦定整除 i 的质数构成的集合恰好与某个 i 的质因子集合相同，则该时刻满足条件。换言之，若对于每个 i，求出 i 的所有质因子的乘积 t_i，则当 $i_0 = t_i$ 时，该时刻恰好满足关于 i 的要求。

正确性证明

- 可以发现，若在搜索过程中，所有钦定整除 i 的质数构成的集合恰好与某个 i 的质因子集合相同，则该时刻满足条件。换言之，若对于每个 i，求出 i 的所有质因子的乘积 t_i，则当 $i_0 = t_i$ 时，该时刻恰好满足关于 i 的要求。
- 注意到这个时刻总存在，因此前述算法的正确性得到了保障。

减枝及时间复杂度证明

- 注意到，当 $i_0p_0 > n$ 时，此后总有 $i_0p_0 > n$，那么不会再额外钦定任意质数整除 i。此时直接返回即可。

减枝及时间复杂度证明

- 注意到，当 $i_0p_0 > n$ 时，此后总有 $i_0p_0 > n$，那么不会再额外钦定任意质数整除 i。此时直接返回即可。
- 考虑分析算法的时间复杂度。首先，dfs 本身的时间复杂度是 $\Theta(n)$ 的，复杂度的瓶颈在于枚举 p_0 倍数的部分。

减枝及时间复杂度证明

- 注意到，当 $i_0 p_0 > n$ 时，此后总有 $i_0 p_0 > n$，那么不会再额外钦定任意质数整除 i。此时直接返回即可。
- 考虑分析算法的时间复杂度。首先，dfs 本身的时间复杂度是 $\Theta(n)$ 的，复杂度的瓶颈在于枚举 p_0 倍数的部分。
- 注意到 $i_0 p_0$ 的最大质因子为 p_0，而 dfs 过程中所有 $i_0 p_0$ 两两不同，因此时间复杂度为 $O\left(\sum_{i=1}^{n} \frac{n}{P(i)}\right)$。

减枝及时间复杂度证明

- 注意到，当 $i_0p_0 > n$ 时，此后总有 $i_0p_0 > n$，那么不会再额外钦定任意质数整除 i。此时直接返回即可。
- 考虑分析算法的时间复杂度。首先，dfs 本身的时间复杂度是 $\Theta(n)$ 的，复杂度的瓶颈在于枚举 p_0 倍数的部分。
- 注意到 i_0p_0 的最大质因子为 p_0，而 dfs 过程中所有 i_0p_0 两两不同，因此时间复杂度为 $O\left(\sum_{i=1}^{n} \frac{n}{P(i)}\right)$。
- Erdos, Ivic, Pomerance 的论文中给出

$$\sum_{i=1}^{n} \frac{1}{P(i)} = n\delta(n)\left(1 + O\left(\frac{\log\log n}{\log n}\right)^{\frac{1}{2}}\right)$$

减枝及时间复杂度证明

- 注意到，当 $i_0 p_0 > n$ 时，此后总有 $i_0 p_0 > n$，那么不会再额外钦定任意质数整除 i。此时直接返回即可。
- 考虑分析算法的时间复杂度。首先，dfs 本身的时间复杂度是 $\Theta(n)$ 的，复杂度的瓶颈在于枚举 p_0 倍数的部分。
- 注意到 $i_0 p_0$ 的最大质因子为 p_0，而 dfs 过程中所有 $i_0 p_0$ 两两不同，因此时间复杂度为 $O\left(\sum_{i=1}^{n} \frac{n}{P(i)}\right)$。
- Erdos, Ivic, Pomerance 的论文中给出 [1]

$$\sum_{i=1}^{n} \frac{1}{P(i)} = n\delta(n)\left(1 + O\left(\frac{\log\log n}{\log n}\right)^{\frac{1}{2}}\right)$$

- 这个算法的实际运行效率相当优秀，在 $n \le 10^6$ 时，其时间复杂度可以看做 $\Theta(n\sqrt{n})$，常数较小。

[1] [1] P. Erdos, A. Ivic and C. Pomerance, On sums involving reciprocals of the largest prime factor of an integer. Glasnik Matematicki. 21(41) 1986, 283-300.

算法伪代码

Algorithm 1 互质数对的维护

p_0 := the x_0-th smallest prime
if $i_0 p_0 > n$ **then**
  for each $t(i) == i_0$ **do**
    claim requirements of i have been satisfied
  end for
else
  dfs($x_0 + 1$, i_0)
  for each j which is a multiple of p_0 **do**
    a_j := 1
  end for
  dfs($x_0 + 1$, $i_0 p_0$)
  for each j which is a multiple of p_0 **do**
    revoke operations on a_j to recover
  end for
end if

引入

- 给定 n ，你需要对初始所有元素均为 1 的序列 $a_1, a_2, \cdots, a_n$ 进行若干次修改，使得 $\forall i \in [1, n]$，存在一个时刻满足 $\forall a_j = \gcd(i, j)$。

维护方式

- 注意到，$\gcd(i,j)$ 为所有满足 $p^\alpha|i$，$p^\alpha|j(p \in Prime, \alpha \in \mathrm{N}+)$ 的 p 的乘积。

维护方式

- 注意到，$\gcd(i,j)$ 为所有满足 $p^\alpha|i$，$p^\alpha|j(p \in Prime, \alpha \in \mathrm{N}+)$ 的 p 的乘积。
- 类似互质数对的维护，我们列出 n 以内的所有质数，并暴力搜索。对于每个质数 p，枚举 i 质因数分解后 p 的最高幂次 β，将其作为多个分支继续搜索下去。在此过程中，维护 x_0, i_0 及序列 a，表示考虑到了第 x_0 小的质数，而 i_0 表示所有 p^β 的乘积。

维护方式

- 注意到，$\gcd(i,j)$ 为所有满足 $p^\alpha|i$，$p^\alpha|j(p\in Prime,\alpha\in \mathrm{N}+)$ 的 p 的乘积。
- 类似互质数对的维护，我们列出 n 以内的所有质数，并暴力搜索。对于每个质数 p，枚举 i 质因数分解后 p 的最高幂次 β，将其作为多个分支继续搜索下去。在此过程中，维护 x_0, i_0 及序列 a，表示考虑到了第 x_0 小的质数，而 i_0 表示所有 p^β 的乘积。
- 在枚举 β 的过程中，每当 β 增大 1 后，若 $i_0p^\beta\le n$，则将所有满足 $p^\beta|j$ 的 a_j 乘 p，向分支 (x_0+1, i_0p^β) 继续搜索。在 (x_0, i_0) 的所有分支处理完毕后，对于所有满足 $p|j$ 的 a_j，撤销对 a_j 的所有乘 p 操作。

算法证明

- 根据算术基本定理，$\forall i \in [1, n]$，都存在某个时刻满足 $i_0 = i$，算法的正确性得到了保障。

算法证明

- 根据算术基本定理，$\forall i \in [1, n]$，都存在某个时刻满足 $i_0 = i$，算法的正确性得到了保障。
- 2.2 节提出的减枝与时间复杂度证明同样适用于此处。

算法证明

- 根据算术基本定理，$\forall i \in [1, n]$，都存在某个时刻满足 $i_0 = i$，算法的正确性得到了保障。
- 2.2 节提出的减枝与时间复杂度证明同样适用于此处。
- 在实际运行效率方面，当 $n \leq 10^6$ 时，其时间复杂度可看做 $\Theta(n\sqrt{n})$，但常数较大。

算法伪代码

Algorithm 2 最大公约数的维护

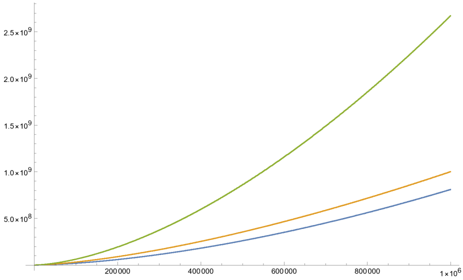
p_0 := the x_0-th smallest prime
if $i_0 p_0 > n$ **then**
    claim requirements of $i = i_0$ have been satisfied
else
    dfs($x_0 + 1, i_0$)
    for each $\beta \in \mathrm{N}+$ satisfying $i_0 p_0^{\beta} \le n$ **do**
        for each j which is a multiple of p_0^{β} **do**
            $a_j := pa_j$
        end for
        dfs($x_0 + 1,\ i_0 p_0^{\beta}$)
    end for
    for each j which is a multiple of p_0 **do**
        revoke operations on a_j to recover
    end for
end if

算法总结

- 可以看出，此算法能够通过一定次数的修改，动态维护每一对 n 以内正整数的最大公约数，因此我将其命名为 Dynamic-gcd algorithm。

算法总结

- 可以看出，此算法能够通过一定次数的修改，动态维护每一对 n 以内正整数的最大公约数，因此我将其命名为 Dynamic-gcd algorithm。
- 下图蓝线、绿线分别为互质数对维护算法、Dynamic-gcd algorithm 中 $a_j \leftarrow 1$ 和 $a_j \leftarrow pa_j$ 的执行次数关于 n 的函数图像，黄线表示 $f(x) = x\sqrt{x}$ 的函数图像。



例题

例

- 给定长度为 n 的序列 A，T 次询问，每次给定正整数 x, l, r，求 $A_{\gcd(x,l)}, A_{\gcd(x,l+1)}, \cdots, A_{\gcd(x,r)}$ 的最大子段和。
- 保证 $1 \le T \le 10^6, 1 \le n \le 5 \times 10^4, 1 \le x, l, r \le n, l \le r$。

- 本题为笔者的原创题。

解法

解

- 考虑 Dynamic-gcd algorithm，在维护 a 的同时，维护序列 b，满足 $b_i = A_{a_i}$。

解法

解

- 考虑 Dynamic-gcd algorithm，在维护 a 的同时，维护序列 b，满足 $b_i = A_{a_i}$。
- 每次对 a 的单点修改后，对 b 也进行单点修改。

解法

解

- 考虑 Dynamic-gcd algorithm，在维护 a 的同时，维护序列 b，满足 $b_i = A_{a_i}$。
- 每次对 a 的单点修改后，对 b 也进行单点修改。
- 将所有询问统一离线处理后，问题转化为：$\Theta(n\sqrt{n})$ 次单点修改，T 次查询区间最大子段和。

解法

解

- 考虑 Dynamic-gcd algorithm，在维护 a 的同时，维护序列 b，满足 $b_i = A_{a_i}$。
- 每次对 a 的单点修改后，对 b 也进行单点修改。
- 将所有询问统一离线处理后，问题转化为：$\Theta(n\sqrt{n})$ 次单点修改，T 次查询区间最大子段和。
- 使用线段树维护即可，时间复杂度 $\Theta((n\sqrt{n} + T)\log n)$。

总结

- 本文提出了互质数对的维护算法，与最大公约数的维护算法 Dynamic-gcd algorithm，通过例题可以看出，它们可以通过较少次数的修改，动态维护每一对 n 以内正整数间互质性、最大公约数的信息，在与互质、$\gcd$ 有关的数论、数据结构等题型中，均有广阔的应用前景。
- 笔者希望本文能起到抛砖引玉的作用，吸引读者对数论信息的动态维护进行深入研究。

致谢

- 感谢中国计算机学会提供学习和交流的平台。
- 感谢南京外国语学校张超老师对我的教导。
- 感谢家人对我的关心与支持。