

万能欧几里得算法

算法简介

一种 $O(\log n)$ 计算形如 $f(n) = \sum_{i=1}^n \lfloor \frac{ai+b}{c} \rfloor$ 的函数的算法。

算法原理

不妨考虑待计算函数就是 $f(n) = \sum_{i=1}^n \lfloor \frac{ai+b}{c} \rfloor$ 同时假设 $b \leq c$ 考虑按如下过程计算 $f(n)$

考虑在 $(0, n]$ 范围内逐渐增加 x 当 $y = \frac{ax+b}{c}$ 恰好等于某个整数时 $y \rightarrow y+1$ 称为 U 操作。

当 x 恰好等于某个整数时有 $\text{ans} \rightarrow \text{ans} + y$ 称为 R 操作。如果遇到整点，先进行 U 操作再进行 R 操作。

于是计算 $f(n)$ 等价于处理由 $y = \frac{ax+b}{c} (x \in (0, n])$ 产生的操作序列 S

先考虑如何得到这个操作序列，设 $g(a, b, c, n, U, R)$ 表示由 $y = \frac{ax+b}{c} (x \in (0, n])$ 产生的操作序列 S

定义字符串幂操作 $S^k = S^{k-1}S$

如果 $a \geq c$ 显然 R 操作前面至少有 $\lfloor \frac{ac}{c} \rfloor$ 个 U 操作。将这 $\lfloor \frac{ac}{c} \rfloor$ 个 U 操作和一个 R 操作视为一个整体，于是有

$$g(a, b, c, n, U, R) = g(a \bmod c, b, c, n, U, U^{\lfloor \frac{ac}{c} \rfloor} R)$$

如果 $b \geq c$ 显然有

$$g(a, b, c, n, U, R) = U^{\lfloor \frac{bc}{c} \rfloor} g(a, b \bmod c, c, n, U, R)$$

最后，如果 $c > \max(a, b)$ 则考虑构造 $y = \frac{ax+b}{c}$ 按 $y = x$ 做对称变换得到的直线 $y = \frac{cx-b}{a}$

不难发现 $y = \frac{cx-b}{a}$ 产生的序列和 $y = \frac{ax+b}{c}$ 产生的序列大致是互补的，即 U/R 互补。

但要处理遇到整点的情况，此时两条直线都是先 U 再 R 不满足互补关系，考虑将 $y = \frac{cx-b}{a}$ 整体向下偏移得到 $y = \frac{cx-b-1}{a}$

这样原先在 $y = \frac{cx-b}{a}$ 先 U 再 R 的整点等价于 $y = \frac{cx-b-1}{a}$ 先 R 在 U 于是整点也满足了互补性质。

但偏移也导致 $y = \frac{cx-b-1}{a}$ 的边界发生了变化，考虑暴力处理边界，设 $m = \frac{an+b}{c}$ 于是有

$$g(a, b, c, n, U, R) = R^{\lfloor \frac{c-b-1}{a} \rfloor} U g(c, (c-b-1) \bmod a, a, m-1, R, U) R^{n - \lfloor \frac{c-b-1}{a} \rfloor}$$

$\lfloor \frac{cm-b-1}{a} \rfloor$

边界条件为 $m=0$ 此时说明只有 R 操作，于是 $g(a,b,c,n,U,R)=R^n$

假如不考虑字符串拼接等操作的复杂度，上述式子可以 $O(\log n)$ 计算。

接下来考虑如何通过得到的字符串即操作序列来计算答案。事实上可以在拼接字符串时维护答案。

设 $h(S)$ 表示字符串 S 对应的答案，两个串分别为

S_1, S_2 $dx = \text{cntR}(S_1), dy = \text{cntU}(S_1), n = \text{cntR}(S_2)$

$$h(S_1S_2) = h(S_1) + \sum_{i=1}^n (\lfloor \frac{ai+b}{c} \rfloor + dy) = h(S_1) + \sum_{i=1}^n (\lfloor \frac{ai+b}{c} \rfloor + n \times dy) = h(S_1) + h(S_2) + n \times dy$$

至于 S^k 可以利用快速幂计算。整体操作复杂度为 $T(a,c) = T(c \bmod a, a) + O(\log \frac{ca}{a})$

$T(a,c)$ 产生的 $O(\log c) - O(\log a)$ 可以和随后 $T(c \bmod a, a)$ 产生的 $O(\log a) - O(\log (c \bmod a))$ 部分抵消。

最终 $T(a,c) = O(\log c) - O(\log \gcd(a,c)) \sim O(\log c)$

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

Permanent link: https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E4%B8%87%E8%83%BD%E6%AC%A7%E5%87%A0%E9%87%8C%E5%BE%97%E7%AE%97%E6%B3%95&rev=1629382956

Last update: 2021/08/19 22:22