Pourchet 定理的实践：将一元非负多项式分解为五个平方和

Victor Magron
CNRS LAAS & 图卢兹数学研究所 图卢兹，法国
victor.magron@laas.fr

Przemysław Koprowski 西里西亚大学 数学研究所 卡托维兹，波兰
przemyslaw.koprowski@us.edu.pl

Tristan Vaccon 利摩日大学；CNRS，XLIM UMR 7252 法国利摩日 trista
n.vaccon@unilim.fr

arXiv:2302.02202v1 [cs.SC] 4 Feb 2023

摘要

Pourchet 于 1971 年证明，每个具有有理系数的一元非负多项式都可以表示为五个或更少平方的和。尽管如此，目前尚无已知算法能够构造这样的分解。本文唯一的目的，是介绍一组算法，将给定的非负多项式分解为六个平方和（在某个尚未证实的猜想成立时，或允许使用权重时，则为五个多项式的平方和）。此外，我们证明，二进制复杂度可以用计算机代数和算法数论中的经典运算多项式地表示。

关键词

单变量有理系数非负多项式，有理系数多项式，平方和分解，实代数几何，范数方程

ACM 引用格式：
Victor Magron、Przemysław Koprowski 和 TristanVaccon。2023年。Pourchet 定理的应用：将一元非负多项式分解为五个平方和。载于 *Proceedings of* 。ACM，纽约，NY，美国，10页。
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 引言

令 $\mathbb{Q}[x]$ 和 $\mathbb{R}[x]$ 分别表示系数为有理数和实数的单变量多项式集合。给定一个非负多项式 $f \in \mathbb{Q}[x]$，我们考虑将 f 分解为同样属于 $\mathbb{Q}[x]$ 的多项式的平方和（SOS）的一个问题，并且可能带有正有理权重。

这一问题不仅在实代数几何领域具有理论意义，也具有实际价值，例如，计算平方和李雅普诺夫证书以确保控制系统 [37], 的稳定性；为在计算机程序中求值的超越函数构造多项式近似[7, 22], ；以及借助证明助理对多项式不等式进行形式化验证[1, 15, 21]。由于此类证明助理的计算能力有限，通常的做法是依赖外部工具来提供比特大小适中的平方和证书，从而使后续验证不会过于耗时。因此，我们尤其希望设计出能够输出比特大小合理的平方和证书的算法，并且其位复杂度可能关于输入数据呈多项式级。

允许出于个人或课堂用途免费制作本作品全部或部分内容的数字版或纸质副本，前提是这些副本不用于或分发以谋取利润或商业利益，并且副本首页须附有本声明及完整引文。对于本作品中由 ACM 以外的其他方拥有版权的内容，必须尊重其版权。注明出处后，允许摘录。若要以其他方式复制、重新发布、发布到服务器或分发至列表，则须事先获得明确许可和/或支付费用。请通过 permissions@acm.org 申请许可。，，© 2023年 计算机协会。ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

相关工作。 众所周知，每个 $\mathbb{R}[x]$ 中的非负单变量多项式都可以分解为两个平方和。早期研究主要致力于寻找所需平方数最少的有理数分解。Landau 在 [19] 证明，每个 $\mathbb{Q}[x]$中的非负多项式都可以分解为至多八个多项式平方之和，且这些多项式属于 $\mathbb{Q}[x]$。Pourchet 在 [34], 改进了这一结果，证明只需五个或更少的平方。Pourchet 定理的证明高度依赖局部-整体原理，尤其是 Hasse-Minkowski 定理；乍看之下，要从中提取出一种能够输出所需平方和分解的构造性算法并非易事。关于 Pourchet 定理的英文介绍，请参阅 [36, 第 17] 章。随后，Schweighofer 在 [39] 推导出一种算法，可以为系数属于 $\mathbb{R}$任意子域的多项式生成平方和分解。该算法输出的平方项数不超过输入多项式的次数。这个递归算法在每一步递归中都会对正多项式进行实根隔离和二次近似，因此位复杂度呈指数级 [25]。在 [7, 第 5.2] 节中推导出的另一种算法，则依赖于对扰动后的正多项式的复根进行近似。与 Schweighofer 的算法相比，该算法要求输入为有理数，所得分解中可能多出三个平方项，但其位复杂度为多项式级 [25]。

或者，也可以通过检验一个半定规划（SDP）的可行性来获得近似的有理数平方和分解，即求解一个涉及线性等式和线性矩阵不等式的凸问题；参见 Parrilo[30] 和 Lasserre [20] 的开创性研究。浮点半定规划求解器的现代发展确保了这项任务在实践中能够高效完成，但要获得精确分解，必须进行后处理步骤，可采用舍入-投影方案 [31], 或扰动补偿技术 [24]。也可以直接计算此类半定规划的精确代数解 [16], , 但相关方案的可扩展性较为有限。请注意，所有这类基于半定规划的框架通常都可用于证明多变量情形下平方和分解的存在性。

这类基于舍入-投影或扰动补偿技术的方案，已被扩展用于设计和分析生成三角多项式正性证书的算法 [26],

非负回路和 [29],以及算术-几何指数和 [28]。最近，人们还研究了一些其他推广；其中包括与另一个单变量多项式共享公共实根的单变量多项式情形 [17]„ 以及梯度理想为零维根式理想的多变量多项式情形 [27]。 [25] 中提出的两种用于生成单变量有理 SOS 分解的算法，已在 RealCertify Maple 库中实现 [23]。

贡献。 据我们所知，目前还没有构造性算法能够实现 Pourchet 定理，将 $\mathbb{Q}[x]$ 中的给定单变量非负多项式分解为 $\mathbb{Q}[x]$中的多项式的五个平方和。我们还研究了位复杂度方面的权衡：一类分解使用固定数量的平方（即五个），另一类则是 [25] 中分析的现有方案，其所用平方的数量取决于输入的次数。本文的核心贡献是提出并分析一种算法（算法 8），将 $f \in \mathbb{Q}[x]$ 中的单变量非负多项式分解为多项式的**六个** 平方，且这些多项式同样具有有理系数。允许使用正有理权重时，该算法可用于得到一个系数为有理数的加权五**个** 平方和。这个主要算法依赖于其他一些过程，这些过程与输入分解中所涉及的平方数量有关。

- 首先，我们在第 2 节中关注 f为两个平方和的情形，并设计算法 1 来实现相应的分解。该算法的主要步骤是求解一个涉及 f首项系数的范数方程；
- 随后，我们在第 3 节中处理 f为四个平方和的情形，并设计算法 5 来实现相应的分解。该算法基于将正有理数分解为有理数的四个平方和、欧拉恒等式以及另一个范数方程；
- 第 4 节重点讨论如何约化到四平方情形 . 该约化过程在算法 6 中通过考察 f常数项和首项系数的 2-adic 赋值，并使用一种类似于 [7, § 5.2] 中所用方法的扰动论证来完成；
- 最后但同样重要的是，设 f为一个次数为 d、系数最大比特长度为 τ的非负一元有理多项式。我们证明，算法 8 可将 f分解为六个平方的和。该计算的预期位复杂度可用以下各项的多项式表示：（1）整数因式分解；（2） $\mathbb{Q}[x]$ 中的因式分解；（3）数域单位群的计算；（4）数域基本单位系统的大小；这些运算均应用于大小关于 d、 τ以及 f及其倒数的最小值呈多项式增长的参数。我们使用“预期”一词，是因为第 3 节中包含一些**拉斯维加斯**子算法。对于具有相同参数的数域基本单位系统，输出的比特长度也将关于其大小呈多项式增长。

计算机代数和算法数论中普通运算的复杂度。我们将以计算机代数和算法数论中经典运算的二进制复杂度来表示算法的复杂度分析。为此，我们必须考虑作为输入出现的任意有理数的高度。高度的概念定义如下。

Definition 1. *For $p, q \in \mathbb{Z}$, $q \neq 0$, we define*

$$\mathrm{height}(p/q) = \max(\lg|p|, \lg|q|),$$

其中 $\lg(\cdot) = \log(\cdot)/\log(2)$。

我们还将使用以下定义来估算复杂度。

定义 2. 对于整数 $k > 0$ 和映射 ϕ： $\mathbb{N}^k \to \mathbb{R}$，若存在一个多项式 $P \in \mathbb{Q}[x_1, \ldots, x_k]$，使得 $\phi(x_1, \ldots, x_k) = O(P(x_1, \ldots, x_k))$，则记作 $\phi(x_1, \ldots, x_k) = \mathrm{poly}(x_1, \ldots, x_k)$。

现在我们介绍将要使用的基本运算的复杂度。

定义 3. 我们将 IntFact(H) 定义为分解高度为 $\leq H$. 的整数 n 的二进制代价

使用连分数或通用数域筛法等经典算法，IntFact(H) 关于 H 是（次）指数级的。

定义 4. 我们将 PolyFact(d, H) 定义为在 $\mathbb{Q}[x]$ 中分解一个次数为 d、系数高度为 $\leq H$. 的多项式的二进制代价

使用经典算法（例如，参见 [4] 第 21 节），PolyFact(d, H) 关于 d 和 H 是多项式复杂度。

定义 5. 我们将UnitGroupComp(L, f) 定义为计算由不可约多项式 f 定义的数域 $L = \mathbb{Q}[x]/(f)$ 的单位群表示的二进制代价。接着，我们将 UnitGroupComp(d, H) 定义为计算某个数域单位群表示的二进制代价上界；该数域 L由商 $\mathbb{Q}[x]/(f)$ 定义，其不可约多项式 f 的次数为 d，系数高度为 $\leq H$。

这项运算是算法数论中的一项核心而精细的任务。最先进的算法在特定假设或启发式方法下，声称其复杂度为次指数级（参见 [2, 13, 14] [3] ）。有文献提出了一种多项式时间量子算法。

定义 6. 我们将 SizeFundUnits(L, f) 定义为：在UnitGroupComp(L, f) 二进制复杂度下，计算由不可约多项式 f 定义的数域 $L = \mathbb{Q}[x]/(f)$ 的单位群的一组基本单位系统时，使用 $\mathbb{Q}[x]$ 中的多项式表示该系统所需的二进制位数。SizeFundUnits(d, H) 定义为 SizeFundUnits(L, f)的二进制位数上界，其中不可约多项式 f 的次数为 d，系数高度为 H，且 $L = \mathbb{Q}[x]/(f)$。这些是在UnitGroupComp(d, H)二进制复杂度下计算的。

虽然存在基本单位的特殊紧凑表示（见 [41, 42]），但界定 SizeFundUnits(L, f) 或 SizeFundUnits(d, H) 同样是一个微妙的问题，它取决于 L 的判别式和调节子。采用 $\mathbb{Q}[x]$ 中的多项式表示时，我们只能假设 SizeFundUnits(d, H) 关于 d 和 H 呈指数增长。

注 7：由于所有这些复杂度都是（亚）指数级的，或者是高次数的多项式级的，因此我们可以安全地假设 PolyFact(d,H)、UnitGroupComp(d,H)、SizeFundUnits(d,H) 关于 d 和 h 呈超线性增长，例如，对于任意正的 d, d', H，有 PolyFact(d,H) +PolyFact(d',H) ≤PolyFact$(d+d',H)$。

2 求解两个平方和

以下事实众所周知。它是 [36, 定理 17.4] 的一个特例。为完整起见，我们在此给出这一事实，因为它提供了一种明确的算法方法，用于判定给定的多项式能否表示为两个平方和。

观察 8. 无平方因子多项式 f 是两个平方和，当且仅当以下两个条件同时成立：

(1) $\mathrm{lc}(f)$ 是两个平方和 (在 $\mathbb{Q}$)，

(2) -1 在 $\mathbb{Q}[x]/(p)$ 中是一个平方，对 f 的每个不可约因子 p 都成立。

现在我们可以给出一个算法（见算法 1），将给定的多项式分解为两个平方和。

命题 9. 设 $f \in \mathbb{Q}[x]$ 是一个两个平方和形式的多项式。那么算法 *1* 输出多项式 a、$b \in \mathbb{Q}[x]$，使得 $a^2+b^2=f$。

此外，若 f 的次数为 d，且其系数高度上界为 H，则某个 H_1 =多项式（d、H），输出的高度为 O（H_1），二进制运算次数的上界为 IntFact（H）+dPolyFact（$2d$、H_1）。

算法 1 计算多项式的一个分解，使其表示为两个平方和

输入： 一个先验已知为 $\mathbb{Q}[x]$中的两个平方和。

输出： 多项式 a，$b \in \mathbb{Q}[x]$ 使得 $a^2+b^2=f$。

1：构造二次域扩张 $\mathbb{Q}(i)/\mathbb{Q}$。

2：求解范数方程

$$\mathrm{lc}(f) = N_{\mathbb{Q}(i)/\mathbb{Q}}(x)$$

并将一个解记为 $a+bi \in \mathbb{Q}(i)$.

3：将 f 分解为首一不可约多项式的乘积

$$f = \mathrm{lc}(f)\cdot p_1^{e_1}\cdots p_k^{e_k}.$$

4: **对于**每个因子 p_j，若对应的指数 e_j为奇数，**则执行**

5: 将 p_j $\mathbb{Q}(i)[x]$ $p_j = g_j\ h_j$ $g_j, h_j \in$ 在 ( ) 上分解为乘积 ·，其中 ( ) 。

6: Set

$$a_j := \frac{1}{2}\cdot(g_j+h_j), \qquad b_j := \frac{1}{2i}\cdot(g_j-h_j).$$

7: 更新 a 和 b，令：

$$a := aa_j + bb_j \quad \text{and} \quad b := ab_j - ba_j.$$

8：Ψ 日期 a 和 b 设置g

$$a := a\cdot\prod_{j\le k} p_j^{2\lfloor e_j/2\rfloor} \quad \text{and} \quad b := b\cdot\prod_{j\le k} p_j^{2\lfloor e_j/2\rfloor}.$$

9: **返回** a ,b.

算法 1 的正确性证明。设 $\xi = a+bi \in \mathbb{Q}(i)$ 为算法第 2 行构造的元素。那么

$$a^2+b^2 = N_{\mathbb{Q}(i)/\mathbb{Q}}(\xi) = \mathrm{lc}(f)$$

是 f的首项系数表示为两个平方和的一种分解。根据前面的观察，这样的分解存在。因此，范数方程有解。

现在假设 p_j是 f的一个奇数重数首一不可约因子。前面的观察表明，-1是 $\mathbb{Q}[x]/(p_j)$中的一个平方。换言之，与 p_j 对应的 $\mathbb{Q}(x)$的素点在 $\mathbb{Q}(i,x)$中分裂。事实上，由于 -1是 $\mathbb{Q}[x]/(p_j)$中的一个平方，我们有$-1 = s^2+qp_j$，其中某个 q, $s \in \mathbb{Q}[x]$，或者等价地，$qp_j = s^2+1 = (s-i)(s+i)$。$s+i$在 $\mathbb{Q}(i)[x]$ 上的不可约因子分解不包含任何实因子，只有复因子，因为 $s+i$没有实根。因此，p_j在同一环中的分解只包含非实复因子（由 $s+i$产生）及其共轭。这意味着 p_j在$\mathbb{Q}(i)[x]$中可分解为两个共轭不可约多项式 $p_j = g_j\cdot h_j$的乘积，其中

$$g_j = a_j + b_j i \qquad \text{and} \qquad h_j = a_j - b_j i$$

对于某些多项式 a_j, $b_j \in \mathbb{Q}[x]$。现在显然，$p_j = a_j^2+b_j^2$和

$$a_j = \frac{1}{2}\cdot(g_j+h_j), \qquad b_j = \frac{1}{2i}(g_j-h_j).$$

暂且假设 f无平方因子，即所有指数 e_j 都等于 1。结合以下关于两个平方和乘积的著名公式，运用简单的归纳法

$$(A^2+B^2)\cdot(a^2+b^2) = (Aa+Bb)^2 + (Ab-Ba)^2$$

可以证明，执行完第 4 行的循环后，多项式 a和 b满足条件 $a^2+b^2=f$。

最后，令 f为任意多项式。我们可以将其写成乘积 $f = gh^2$，其中

$$g := \mathrm{lc}(f)\cdot\prod_{\substack{j\le k\\ 2\nmid e_j}} p_j \qquad \text{and} \qquad h := \prod_{j\le k} p_j^{\lfloor e_j/2\rfloor}.$$

那么 g无平方因子，并根据证明的前一部分可分解为两个平方和 $g = a^2+b^2$。由此可得 $f = (ah)^2+(bh)2$，从而证明了该算法的正确性。 □

算法 1 的复杂度分析。设 f的次数为 d，且其系数的高度不超过 H。为求解第 2 行中的范数方程，只需：(1) 分解 $\mathrm{lc}(f)$；(2) 针对每个素因子求解范数方程；(3) 利用婆罗摩笈多–斐波那契恒等式合并解，即 $(\alpha^2+\beta^2)(\gamma^2+\delta^2) = (\alpha\gamma-\beta\delta)^2+(\alpha\delta+\beta\gamma)^2 = (\alpha\gamma+\beta\delta)^2+(\alpha\delta-\beta\gamma)^2$。第一项的复杂度为 IntFact$(H)$。根据 [43, p. 128] 和 [38],，第二项关于 H是多项式时间的。重组关于 H也是多项式时间的，且输出解的高度不超过 H。

第 3 行在 PolyFact(n,H) 中完成。对于某个关于 H 和 d 的 H_1 多项式，根据 [8, § 3.6.2]，第 5 行在 PolyFact$(2d,H_1)$ 中完成。该步骤之后，仅剩算术运算。

总而言之，对于某个关于 H 和 d 的 H_1 多项式，算法 1 的总复杂度为 IntFact(H) +dPolyFact$(2d,H_1)$。□

备注 10。为求解算法 1 第 2 行中的范数方程（以及下文算法 3 第 2 行中的范数方程），也可以采用 [9–12, 40] 中介绍的方法。

3 求解四个平方和

本节旨在设计并分析一种过程：在假设分解存在的前提下，将 $\mathbb{Q}[x]$ 中的非负元素分解为 $\mathbb{Q}[x]$中的四个平方和。下文的算法 3 和算法 4 将实现这一目标，并且需要将 -1 分解为数域 $K := \mathbb{Q}[x]/(f)$ 中的两个平方和。这基本归结为求解范数方程 $-1 = N_{L/K}x$，其中 $L = K(\sqrt{-1})$。为此，可以采用 [9–12, 40] 中介绍的任意一种方法。尽管如此，为了能够对算法 3 进行复杂度分析，我们在算法 2 中给出 Simon 求解范数方程的简化版本（参见 [40]），该版本足以满足我们的目的。

回顾一下，若域 K中的 -1 是平方和，则称域 K非实。将 -1 表示为平方和所需的最少项数称为 K的层数，记作 $s(K)$。

算法 2 在数域 $K := \mathbb{Q}[x]/(f)$ 上将 -1 分解为两个平方和。输入：层数为 $s(K) \le 2$的数域 K。输出： a、$b \in K$，使得 $a^2 + b^2 = -1$。1：若存在 $c \in K$ 使得 $c^2 = -1$，则 2：返回 $a = c$、$b = 0$。3：构造二次域扩张 $L := K(\sqrt{-1})$。4：分别构造 K和 L的单位群 U_K和 U_L。5：构造商群 $G_K := U_K/U_K^2$和 G_L：$= U_L\ U_L^2$ $\mathscr{K} = \kappa_1 \dots \kappa_k \subset U_K$ $\mathscr{L} = \lambda_1 \dots \lambda_l \subset /$。令：$\{,,\}$和：$\{,,\}$ U_L为一组元素，分别构成 G_K和 G_L的基，并将它们视为 $\mathbb{F}_2$-向量空间。6：令 $V = (v_1, \dots, v_k)$ 为陪集 $-1 \cdot U_K^2$ 相对于 $\mathscr{K}$ 的坐标。7：对于每个 $i \le l$，记 $(m_{i1}, \dots, m_{ik})$ 为 $N_{L/K}(\lambda_i) \cdot U_K^2$ 相对于 $\mathscr{K}$的坐标。8：求解 $\mathbb{F}_2$ 个线性方程组成的方程组 $M^T \cdot X = V$，其中 $M = (m_{ij})$。将一个解记为 $(\varepsilon_1, \dots, \varepsilon_l)$。9：令 $\lambda := \lambda_1^{\varepsilon_1} \cdots \lambda_l^{\varepsilon_l}$，并令 $c \in K$满足 $c^2 = -N_{L/K}(\lambda)$。10：将 λ/c写成 $\lambda/c = a + b\sqrt{-1}$。11：返回 a、b。

命题 11。 设 $K = \mathbb{Q}[x]/(f)$是层数为 $s(K) \le 2$且由其生成多项式 f指定的数域。于是，算法 *2* 输出 $a, b \in K$，使得 $a^2 + b^2 = -1$。

此外，若 f的次数为 d，且其系数高度的上界为 H，则 a, b 的二进制长度关于 SizeFundUnits$(2d, 2Hd + 8d(1 + \lg(d)))$ 呈多项式关系，二进制运算次数关于 UnitGroupComp$(2d, 2Hd + 8d(1 + \lg(d)))$ 和 SizeFundUnits$(2d, 2Hd + 8d(1 + \lg(d)))$呈多项式关系。

算法 2 的正确性证明。如果算法在第 1 行终止，那么其输出的正确性显而易见。因此，不失一般性，我们可以假设 $-1 \notin K^{\times 2}$。特别地，这意味着 $L = K(\sqrt{-1})$ 是 K 的真二次扩张。根据假设，-1 是 K 中两个平方和，记作 $-1 = a^2 + b^2$，其中 $a,\ b \in K^\times$。于是 $-1 = N_{L/K}(\mu)$，其中 $\mu = a + b\sqrt{-1}$。注意到 μ 必须是 L 中的单位，因为

$$N_{L/\mathbb{Q}}(\mu) = (N_{K/\mathbb{Q}} \circ N_{L/K})(\mu) = N_{K/\mathbb{Q}}(-1) = (-1)^{(K:\mathbb{Q})}.$$

由此可知，我们可以将元素 μ写成

$$\mu = \lambda_1^{\varepsilon_1} \cdots \lambda_l^{\varepsilon_l} \cdot \nu^2,$$

其中，$\nu \in U_L$和 $\varepsilon_1, \dots, \varepsilon_l \in \{0, 1\}$ 是陪集 $\mu\ U_L^2$ $\mathscr{L} = \lambda_1 \dots \lambda_l$ $G_l = \cdot$相对于 $/U_L\ U_L^2$的基 $\{,,\}$的坐标。计算上述等式两边的范数，我们得到

$$-1 = N_{L/K}(\mu) = N_{L/K}(\lambda_1)^{\varepsilon_1} \cdots N_{L/K}(\lambda_l)^{\varepsilon_l} \cdot N_{L/K}(\nu)^2.$$

现在，令 m_{ij}和 v_j满足算法第 (6–7) 步中的条件。在商群 $G_K = U_K/U_K^2$中，我们有

$$\left(\kappa_1^{v_1} \cdots \kappa_k^{v_k}\right) \cdot U_K^2 = \prod_{i \le l} \prod_{j \le k} \kappa_j^{m_{ij}\varepsilon_i} \cdot U_K^2.$$

$\kappa_1, \dots, \kappa_k$的陪集构成 G_K的一组基。特别地，它们线性无关。因此，对每个 $j \le k$，我们都有

$$v_j = m_{1j}\varepsilon_1 + \cdots + m_{lj}\varepsilon_l.$$

这样，我们证明了第 8 行中考虑的 $\mathbb{F}_2$-线性方程组有解。

反过来，设 $\varepsilon_1, \dots, \varepsilon_l \in \{0, 1\}$ 是方程组 $M^T \cdot X = V$的一个解，并令 $\lambda = \lambda_1^{\varepsilon_1} \cdots \lambda_l^{\varepsilon_l}$。由证明的前一部分可知，陪集 $-1 \cdot U_K^2$和 $N_{L/K}(\lambda) \cdot U_K^2$相同。因此，存在一个单位 $c \in U_K$，使得 $-c^2 = N_{L/K}(\lambda)$。因此 $-1 = N_{L/K}(\lambda/c)$，于是若 $\lambda/c = a + b\sqrt{-1}$，便得到所求的分解 $-1 = a^2 + b^2$。□

算法 2 的复杂度分析。假设 $K = \mathbb{Q}[x]/(f)$，其中 f的次数为 d，且为首一不可约多项式，系数的高度为 $\le H$。首先估算构造 L 的开销。令 $g = x^2 + 1$，则对于某个满足 $|c| \le d^2$ 的 $c \in \mathbb{Z}$，$u(x) := \mathrm{Res}_t(f(t)\ g(x - ct))$ 是定义 L.的不可约多项式。事实上，只需保证 c不是 f的根之差与 g的根之商，且这类商严格少于 d^2 个。由于 f和 g是首一的，$u(x) = f((x-i)/c)f((x+i)/c)$。于是，$u$ 的系数高度可以用 $2Hd + 8d(1 + \lg(d))$ 作为上界。这意味着，第 4 行计算 U_K和 U_L的开销为 UnitGroupComp$(2d, 2Hd + 8d(1 + \lg(d)))$。从基本单位系统构造商群是直接的。第 8 行的线性方程组可在关于 d和 SizeFundUnits$(2d, 2Hd + 8d(1 + \lg(d)))$ 的多项式时间内求解。a, b的输出位大小关于 SizeFundUnits$(2d, 2Hd + 8d(1 + \lg(d)))$ 呈多项式级。□

命题 12. 设 $f \in \mathbb{Q}[x]$ 是一个次数为 d、系数高度上界为 H的不可约多项式，且已知它是 $\mathbb{Q}[x]$中的 3 个或4 个平方之和。那么算法 *3* 会计算出多项式 h和 $g_1, \dots, g_4$，它们属于 $\mathbb{Q}[x]$，且满足 $\deg h \le \deg f - 2$以及 $fh = g_1^2 + \cdots + g_4^2$。

二进制复杂度关于UnitGroupComp($2d\ 2Hd+,(\lg()))$和 $8d\ 1+\ d$ $2d\ 2Hd+8d\ 1+\ d$ SizeFundUnits(,(lg()))是多项式级的。输出的二进制位数关于SizeFundUnits(2*d*, 2*Hd*+8*d*(1+lg(*d*)))是多项式级的。

算法 3 初始解：模平方和

输入： 一个不可约多项式 $f \in \mathbb{Q}[x]$，其先验已知为三个或四个平方和。

输出： 多项式 h 和 $g_1,\dots,g_4$ 属于 $\mathbb{Q}[x]$，使得 $\deg h \le \deg f - 2$ 和 $fh = g_1^2+\cdots+g_4^2$。

1：构造数域：

$$K := \mathbb{Q}[x]/(f) \quad \text{and} \quad L := K(i)。$$

2：求解范数方程

$$-1 = N_{L/K}(x)$$

并将解记为 $\xi = \overline{g}_1 + \overline{g}_2 i$，其中 $g_1, g_2 \in \mathbb{Q}[x]$ 是次数严格小于 $\deg f$ 的多项式，而 $\overline{g}_j$ 表示 g_j 在典范满同态 $\mathbb{Q}[x] \twoheadrightarrow K$ 下的像。

3：设 $g_3 := 1, g_4 := 0$，并令 $h := (g_1^2+\cdots+g_4^2)/f$。

4：**返回** h, g_1, g_2, g_3, g_4。

算法 3 的正确性证明。根据假设，f 是多项式的 4 个或更少个平方之和，记为

$$f = p_1^2 + p_2^2 + p_3^2 + p_4^2.$$

并非所有 p_j 都为零。假设 $p_1 \neq 0$。那么在 K 中，我们有

$$-1 = \left(\frac{\overline{p}_2}{\overline{p}_1}\right)^2 + \left(\frac{\overline{p}_3}{\overline{p}_1}\right)^2 + \left(\frac{\overline{p}_4}{\overline{p}_1}\right)^2.$$

这里，$\overline{p}_j$ 再次表示 p_j 在 K 中的像。由此可知，K 是一个非实域，且其层数不超过 3。但众所周知（例如参见 [18, 定理 XI.2.2]），非实域的层数总是 2 的幂。因此，-1 是 K 中的两平方和。于是，存在多项式 $g_1, g_2 \in \mathbb{Q}[x]$，使得 $\deg g_j < \deg f$ 且

$$\overline{g}_1^2 + \overline{g}_2^2 = N_{L/K}(\overline{g}_1 + \overline{g}_2 i) = -1.$$

现在显然，$g_1^2+g_2^2+1$ 可被 f 整除。将商记为 h，可得 $fh = g_1^2+g_2^2+1$。此外，我们有 $\deg g \le \deg f - 1$，因此 $\deg h$ 不可能超过 $\deg f - 2$，这正如预期。□

算法 3 的复杂度分析。根据算法 2 的分析，若 f 的次数为 d，且其系数的高度为 $\le H$，则算法 UnitGroupComp(, (lg( ))) 和 SizeFundUnits(, (lg( ))) 是多项式级的。输出的二进制位数也关于 SizeFundUnits(, (lg( ))) 是多项式级的。□

现在我们可以介绍一种方法（见算法 4），将不可约有理多项式分解为 $\mathbb{Q}[x]$ 中的三个或四个平方和，前提是这种分解存在。我们需要确定在相乘四平方和时将采用欧拉恒等式的哪一种变体。以下变体可确保后续算法输出多项式。

欧拉恒等式。

$$\begin{aligned}(A^2+B^2+C^2+D^2)(a^2+b^2+c^2+d^2) &= \\ &= (Aa+Bb+Cc+Dd)^2 + (-Ab+Ba-Cd+Dc)^2 \\ &\quad + (-Ac+Bd+Ca-Db)^2 + (-Ad-Bc+Cb+Da)^2.\end{aligned}$$

算法 4 将首一不可约多项式分解为三个或四个平方和

输入： 首一不可约多项式 $f \in \mathbb{Q}[x]$，已知其为一个三个或四个平方和。

输出： 多项式 $f_1, \dots, f_4 \in \mathbb{Q}[x]$，使得 $f_1^2+\cdots+f_4^2 = f$。

1：执行算法 3 构造多项式 h，g_1，……，g_4，使得 $fh = g_1^2+\cdots+g_4^2$，且 $\deg h \le \deg f - 2$。

2：**当** $\deg h > 0$ **do**

3：计算 $g_1,\dots,g_4$ 模 h 的余式 $r_j := (g_j$ 数 h。

4：利用欧拉恒等式，将乘积表示为

$$(g_1^2+g_2^2+g_3^2+g_4^2)(r_1^2+r_2^2+r_3^2+r_4^2)$$

四个平方和。将结果记为 $g_1', \dots, g_4'$。

5：更新 $g_1,\dots,g_4$，令 $g_j := g_j'/h$，其中 $j \in \{1,2,3,4\}$。

6：更新 h 设置 $h := (g_1^2+\cdots+g_4^2)/f$。

7：将 h 分解为有理数的四个平方和 $h = a_1^2+\cdots+a_4^2$，其中 $a_1, \dots, a_4 \in \mathbb{Q}$。

8：使用欧拉恒等式表示乘积

$$\left(g_1^2+\cdots+g_4^2\right)\cdot\left(\left(\frac{a_1}{h}\right)^2+\cdots+\left(\frac{a_4}{h}\right)^2\right)$$

为四个平方和。将结果存储在 $f_1, \dots, f_4$。

9：**返回** $f_1, \dots, f_4$。

命题 13。设 $f \in \mathbb{Q}[x]$ 是一个次数为 d、系数高度不超过 H 的不可约多项式，并且已知它是有理数域中的 3 个或 4 个平方之和。那么算法 4 会计算出有理数域中的多项式 $g_1, \dots, g_4$，使得 $f = g_1^2+\cdots+g_4^2$。

二进制复杂度关于UnitGroupComp( ) 和 SizeFundUnits(, (lg())) 呈多项式关系。输出的二进制位数关于SizeFundUnits($2d, 2Hd + 8d(1 + \lg(d))$)。

算法 4 的正确性证明。我们将主循环第 k 次迭代后变量 h、g_j、g_j' 和 r_j 的值分别记为 h_k、g_{kj}、g_{kj}' 和 r_{kj}。特别地，将第 1 行计算得到的初始值记为 h_0 和 $g_{01},\dots,g_{04}$。

根据第 6 行，对于每个 $k \ge 0$，我们都有 $fh_k = g_{k1}^2+\cdots+g_{k4}^2$。我们将证明所有 h_k 都是多项式，且它们的次数构成严格递减序列；同样，所有 g_{k1}、……、g_{k4} 也都是多项式。当然，h_0 和 g_{01}、……、g_{04} 都是多项式。让我们考察算法第 4 行中出现的两个平方和的乘积：

$$\left((g_{k+1,1}')^2+\cdots+(g_{k+1,4}')^2\right) = \left(g_1^2+g_2^2+g_3^2+g_4^2\right)\left(r_1^2+r_2^2+r_3^2+r_4^2\right).$$

这里，对于每个 $j \le 4$，都有 $r_{kj} = g_{kj} - q_{kj}h_k$，其中某个多项式 $q_{kj} \in \mathbb{Q}[x]$ 满足 $\deg r_{kj} < \deg h_k$。欧拉恒等式给出

$$\begin{aligned} g'_{k+1,j} &= g_{k1}r_{k1} + g_{k2}r_{k2} + g_{k3}r_{k3} + g_{k4}r_{k4} \\ &= g_{k1}\cdot(g_{k1} - q_{k1}h_k) + g_{k2}\cdot(g_{k2} - q_{k2}h_k) \\ &\quad + g_{k3}\cdot(g_{k3} - q_{k3}h_k) + g_{k3}\cdot(g_{k3} - q_{k3}h_k) \\ &= \left(g_{k1}^2 + g_{k2}^2 + g_{k3}^2 + g_{k4}^2\right) \\ &\quad - \left(g_{k1}q_{k1} + g_{k2}q_{k2} + g_{k3}q_{k3} + g_{k4}q_{k4}\right)\cdot h_k \\ &= \left(f - g_{k1}q_{k1} - g_{k2}q_{k2} - g_{k3}q_{k3} - g_{k4}q_{k4}\right)\cdot h_k. \end{aligned}$$

这表明 $g_{k+1,1}$ 能被 h_k 整除，因此 $g_{k+1,1} = g_{k+1,1}/h_k$ 确实是一个多项式。同理

$$\begin{aligned} g'_{k+1,2} &= -g_{k1}r_{k2} + g_{k2}r_{k1} - g_{k3}r_{k4} + g_{k4}r_{k3} \\ &= -g_{k1}\cdot(g_{k2} - q_{k2}h_k) + g_{k2}\cdot(g_{k1} - q_{k1}h_k) \\ &\quad - g_{k3}\cdot(g_{k4} - q_{k4}h_k) + g_{k4}\cdot(g_{k3} - q_{k3}h_k) \\ &= (g_{k1}q_{k2} - g_{k2}q_{k1} + g_{k3}q_{k4} - g_{k4}q_{k3})\cdot h_k. \end{aligned}$$

因此，$g_{k+1,2} = g_{k+1,2}/h_k$ 也是一个多项式。类似的论证同样适用于 $g_{k+1,3}$ 和 $g_{k+1,4}$。

现在，假设我们已证明，对于某个 $k \ge 0$，h_k 是一个多项式。我们有

$$fh_k = g_{k1}^2 + \cdots + g_{k4}^2 \tag{1}$$

以及 $r_{kj} = (g_{kj} \bmod h_k)$。由此可得

$$r_{k1}^2 + \cdots + r_{k4}^2 \equiv 0 \pmod{h_k}.$$

因此，存在某个 h_{k}，使得

$$h_k h'_k = r_{k1}^2 + \cdots + r_{k4}^2. \tag{2}$$

现在，对所有 $j \le 4$，$\deg r_{kj} < \deg h_k$，因此 $\deg h_{k} \le \deg h_k - 2$。将 (1) 与 (2) 合并，得到

$$f\cdot h_k^2\cdot h'_k = (g_{k1}^2+\cdots+g_{k4}^2)(r_{k1}^2+\cdots+t_{k4}^2) = (g'_{k+1,1})^2+\cdots+(g'_{k+1,k})^2.$$

由此得到 $fh_{k} = g_{k+1,1}^2 + \ldots + g_{k+1,4}^2$。因此

$$h'_k = \frac{g_{k+1,1}^2 + \cdots + g_{k+1,4}^2}{f} = h_{k+1}.$$

这表明 h_{k+1} 是一个次数为 $\le \deg h_k - 2$ 的多项式，从而证明了我们的结论。因此，经过有限步后，h_k 的次数最终会降为零，算法也就会终止。

现在，令 h_k 满足 $\deg h_k = 0$。我们知道

$$fh_k = g_{k1}^2 + \cdots + g_{k4}^2.$$

根据假设，f 本身是一个平方和。因此，h_k 必须是一个正有理数，故可表示为四个（或更少个）平方之和。令 $h_k = a_1^2 + \cdots + a_4^2$ 表示某些有理数 $a_1, \ldots, a_4 \in \mathbb{Q}$。由此可知，算法第 8 行中的乘积是一个等于 f 的四个平方和。这证明了所给算法的正确性。□

算法 4 的复杂度分析。根据算法 3 的分析，若多项式 f 的次数为 d，且其系数高度的上界为 H，则第 1 行的二进制复杂度是关于 UnitGroupComp($2d,\ 2Hd + 8d(1+\lg(d))$) 和 SizeFundUnits($2d,\ 2Hd+8d(1+\lg(d))$) 的多项式。输出的二进制位数是关于 SizeFundUnits($2d,\ 2Hd+8d(1+\lg(d))$) 的多项式。**当** 循环执行次数少于 d，因此涉及的算术运算次数关于 d 是多项式量。

根据 [33, 35]，第 7 行的预期（拉斯维加斯）二进制复杂度是关于 h 的高度的多项式。总而言之，该算法的预期二进制复杂度是关于 UnitGroupComp($2d,\ 2Hd+8d(1+\lg(d))$) 和 SizeFundUnits($2d,\ 2Hd+8d(1+\lg(d))$) 的多项式，且输出的二进制位数仍是关于 SizeFundUnits($2d\ 2Hd+,\ (\lg())$) 的多项式。□

将任意多项式分解为四个平方和（前提是这种分解存在）现在已十分直接。为完整起见，下面给出一个明确的算法。

算法 5 分解为三个或四个平方和

输入： 一个多项式 $f \in \mathbb{Q}[x]$，先验已知它是三个或四个平方和。

输出： 多项式 $f_1, \ldots\ldots, f_4 \in \mathbb{Q}[x]$ 使得 $f_1^2 + \cdots + f_4^2 = f$。

1: 利用无平方因子分解，构造 $g,\ h \in \mathbb{Q}[x]$ 使得使得 $f = \mathrm{lc}(f)\cdot g\cdot h^2$，其中 g 是首一无平方因子的。
2: 将 $\mathrm{lc}(f)$ 分解为有理数的四个平方和，数，并用所得结果初始化 $f_1, \ldots\ldots, f_4$。
3: 将 g 因子分解为首一不可约多项式的乘积：$g = p_1 \cdots p_k$。
4: **对于**每个 p_j **执行**
5: 执行算法 4，将 p_j 表示为四个平方数之和
$$p_j = g_1^2 + \cdots + g_4^2.$$
6: 使用欧拉恒等式，将乘积 $(f_1^2+\cdots+f_4^2)(g_1^2+\cdots+g_4^2)$ 表示为四个平方数之和。将结果再次存入 $f_1、\ldots、f_4$。
7: **返回** $f_1、\ldots、f_4$。

命题 14。设 $f \in \mathbb{Q}[x]$ 是一个次数为 d、系数高度不超过 H 的多项式，且已知它是3个或4个平方之和。那么，算法 5 会计算出多项式 $g_1, \ldots, g_4 \in \mathbb{Q}[x]$，使得 $f = g_1^2 + \cdots + g_4^2$。

预期的二进制复杂度是关于 UnitGroupComp($8d\ 1+\ \ d\ \ 2d\ 2Hd+8d\ 1+\ \ d\ \ 2d\ 2Hd+$, (lg())) 和 SizeFundUnits(,(lg()))的多项式。输出的二进制位数关于SizeFundUnits($2d\ 2Hd+$,(lg()))呈多项式增长。

鉴于算法 4，算法 5 的正确性显而易见。因此，我们仅讨论复杂度分析。

算法 5 的复杂度分析。该算法依赖于多项式因式分解、对若干因子应用算法 4，以及利用欧拉恒等式进行重组。

得益于我们的超线性假设，我们可以将例如各因子的 UnitGroupComp 的代价总和上界为 UnitGroupComp($2d, 2Hd + 8d(1+\lg(d))$)。总而言之，该算法的期望二进制复杂度关于 PolyFact(d, H), UnitGroupComp($2d, 2Hd+8d(1+\lg(d))$) 和 SizeFundUnits($2d, 2Hd+8d(1+\lg(d))$) 呈多项式增长，并且输出的二进制位数仍关于 SizeFundUnits($2d, 2Hd + 8d(1+\lg(d))$) 呈多项式增长。□

注 15. 为了将一个自然数分解为四个平方和（见算法 4 第 7 行和算法 5 第 2 行）

例如，可以使用 [5, 32, 33, 35]。如果需要确定性算法，可以：(1) 将条目 z 在 $\mathbb{Z}$ 上因式分解；(2) 对每个素因子 p，使用 [5, §4.3] 计算 p 的四个平方和分解，然后使用 [35, §3]；(3) 使用欧拉恒等式重新组合。此时，复杂度关于 IntFact(height(z))，以及 height(z) 为多项式级。

4 化简为四个平方

本节介绍算法 6 和 7，它们将我们要解决的主要问题——把正多项式分解为平方和——化归为把多项式分解为四个平方和的问题，而我们已在上一节处理了这一问题。给定 $f = c_0 + c_1x + \cdots + c_dx^d \in \mathbb{Q}[x]$，我们定义相应的倒数多项式 $f_* := c_d + c_{d-1}x + \cdots + c_0x^d \in \mathbb{Q}[x]$。我们首先关注 c_d 的 2-adic 赋值为奇数的情形，此时采用算法 6；随后通过算法 7 处理一般情形。

算法 6 化归为四个平方和：赋值为奇数的情形

输入： 正的无平方因子多项式 $f = c_0 + c_1x + \cdots + c_dx^d \in \mathbb{Q}[x]$。$f$的系数的 2-adic 赋值为 k_j：对于 $0 \le j \le d$，$= \mathrm{ord}_2\, c_j$。确保 k_d为奇数。假设 f不是四个平方和。

输出： 一个多项式 $h \in \mathbb{Q}[x]$，使得 $f - h^2$ 是四个（或更少个）平方的和。

1：找到一个正数 ε，使得

$$\varepsilon < \text{下确界}\{f(x) \mid x \in \mathbb{R}\}.$$

2：令 $l_1 := \lceil -1/2 \cdot \lg \varepsilon \rceil$。

3：令 $l_2 := \lceil -k_0/2 \rceil + 1$。

4：令

$$l_3 := \left\lfloor \text{最大值} \ \frac{jk_d - dk_j}{2d - 2j} \mid 0 < j < d \right\rfloor.$$

5：初始化 $l := \max\{l_1,\ l_2,\ l_3\}$。

6: **当** 最大公因数$(d, 2l + k_d) \neq 1$ **时执行**

7: $l := l + 1$。

8: **返回** $h := 2^{-l}$。

命题 16。设 f是有理数域$\mathbb{Q}[x]$上的一个正的无平方因子多项式，其次数为 d，系数的高度为 $\le H$.。假设f不是四个（或更少个）平方和。则算法 7 利用算法 6，$g_1\ g_2 \in \mathbb{Q}[x]$ $f - g_1^2 - g_2^2$ 计算出多项式，使得其为（或更少个）平方和。4

输出的二进制复杂度和比特长度是关于 d、H、lg(下确界 f) 和 lg(下确界 f_*)的多项式。

算法 6 的正确性证明。我们假设f不是四个平方和。首先注意，第 6 行的循环会终止。事实上，根据假设，k_d是奇数，因此根据狄利克雷等差数列素数定理，等差数列 $(k_d + 2l \mid l \ge l_1\ l_2\ l_3)$ max{,}中包含无穷多个素数。特别地，其中必有一个数与次数 d互素。此外，$(k_d + 2l$ 模 $d)$ 的周期为 $\le d$，因此第 6 行的循环至多执行 d 次，算法便会终止。

算法 7 归约为四个平方和：一般情形

输入： 一个正的无平方因子多项式 $f = c_0 + c_1x + \cdots + c_dx^d \in \mathbb{Q}[x]$，即先验已知它不是四个平方数的和。

输出： 多项式 g_1，$g_2 \in \mathbb{Q}[x]$ 使得 $f - g_1^2 - g_2^2$是一个和由四个（或更少）平方数组成的和。

1：将常数项和首项系数的 2-adic 赋值分别记为 f的 $k_0 := \mathrm{ord}_2\, c_0$ 和 $k_d := \mathrm{ord}_2\, c_d$，分别为——地。

2: **如果** k_d 是奇数 **则**

3: 执行算法 6。将其输出记为 h。

4: **返回** $g_1 := h$，$g_2 := 0$。

5: **否则，如果** k_0 是奇数**则**

6: 令 $f_* := c_d + c_{d-1}x + \cdots + c_0x^d$。

7: 对 f_* 执行算法 6，并将其输出记为 h。

8: **返回** $g_1 := x^d/2 \cdot h(1/x)$，$g_2 := 0$。

9: **否则**

10: 对 $2f$ 执行算法 6，并将其输出记为 h。

11: **返回** $g_1 := h/2$，$g_2 := h/2$。

令 h为算法构造的多项式。我们断言，$f - h^2$ 在 $\mathbb{Q}_2[x]$中不可约。我们有$l \ge l_2 > -k_0/2$,，因此 $f - h^2$ 的常数项的 2-adic 赋值为

$$\mathrm{ord}_2\left(c_0 - 2^{-2l}\right) = \min\{k_0, -2l\} = -2l.$$

$f - h^2$ 的所有其他系数均与 f 的相应系数相同。条件 $l \ge l_3$ 意味着，对于每个 $j \in \{1, \ldots, d-1\}$，点 $(j,\ k_j)$ 位于端点为 $(0,\ -2l)$ 和 $(d,\ k_d)$ 的线段上或其上方。因此，$f - h^2$ 的牛顿多边形仅由这条线段构成，其斜率为 $(2l + k_d)/d$。现在，d 和 $2l + k_d$ 互素，因此根据牛顿多边形的众所周知的性质，$f - h^2$ 在 $\mathbb{Q}_2[x]$ 中不可约（例如，参见 [6, 引理 3.5]）。这证明了我们的断言。

最后，多项式 f 为正且无平方因子，因此不可能有任何实根。于是它与零分离，这意味着可以找到如第 1 行所示的 $\varepsilon > 0$。现在，条件 $l \ge l_1 \ge \lceil -1/2 \cdot \lg \varepsilon \rceil$ 表明 $f - h^2 = f - 2^{-2l}$ 也为正。因此，[36, 定理 17.2] 断言，$f - h^2$ 在 $\mathbb{Q}_2[x]$ 中为正且不可约，因而是四个（或更少）平方之和。□

算法 7 的正确性证明。首先注意，f 为正，因此不可能有任何实根。特别地，它的常数项 c_0 必定非零。现在，如果 k_d 为奇数，则第 4 行的正确性可由已证明正确的算法 6 推出。接下来，假设 f 的常数项具有奇数 2-adic 赋值。第 6 行构造的倒数多项式 f_* 满足恒等式

$$f_*(x) = x^d \cdot f\left(1/x\right).$$

因此，它是正的，故为平方和；其首项系数为 c_0，其 2-adic 赋值为奇数。因此，使用算法 6，我们可以找到 $h \in \mathbb{Q}[x]$，使得 $g := f_* - h^2$ 是四个平方和。设 $g = a_1^2 + \cdots + a_4^2$。令 g_1 如第 8 行所示。我们

have

$$f - g_1^2 = x^d \cdot f_*\left(\frac{1}{x}\right) - \left(x^{\frac{d}{2}} \cdot h\left(\frac{1}{x}\right)\right)^2 = x^d \cdot g\left(\frac{1}{x}\right)$$
$$= \left(x^{\frac{d}{2}} \cdot a_1\left(\frac{1}{x}\right)\right)^2 + \cdots + \left(x^{\frac{d}{2}} \cdot a_4\left(\frac{1}{x}\right)\right)^2$$

也是四个平方和。这证明了第 8 行的正确性。最后一种情形，即 k_0 和 k_d 均为偶数时，正确性显而易见。□

算法 6 和算法 7 的复杂度分析。对于算法 6，可通过检查 $f - 2^{2^s}$ 是否有实根来计算所需的 ε，并逐步增大 s.。这可以通过笛卡尔符号法则等经典方法，或计算由 $f - 2^{2^s}$ 的系数定义的特殊汉克尔矩阵的惯性指数来完成。其复杂度关于 d、c_j 的高度以及 $\lg(\inf f)$ 呈多项式级。其余操作也同样如此。输出 h 的大小关于这些量呈多项式级。因此，显然算法 7 的复杂度完全由算法 6 的复杂度决定。□

注17。 算法 6 始终输出一个常数多项式，因此在上文第 8 行中，实际上有 $g_1 = x^{d/2} \cdot h(1/x) = x^{d/2} \cdot h$。我们使用“繁复”的记法 $h(1/x)$，以便用任何具有相同输入和输出规范的其他算法替换算法 6。

5 最终算法：六个平方和

现在，我们已经可以介绍主要算法，该算法将 $\mathbb{Q}[x]$ 中任意单变量非负多项式分解为 $\mathbb{Q}[x]$ 中多项式的六个平方和。由于先前算法均正确，因此该算法显然也是正确的。

定理 18. 若 $f \in \mathbb{Q}[x]$ 是次数为 d、系数高度为 $\leq H$ 的单变量非负多项式，则算法 *8* 会计算出 f 的一个至多包含 6 个平方的分解。存在某个 $H_1 =$ 多项式$(d, H,$ 无穷 $(f),$ 无穷$(f_*))$，使得期望二进制复杂度关于 IntFact(H)、PolyFact(d, H_1) ，UnitGroupComp$(2d, H_1)$ 和 SizeFundUnits$(2d, H_1)$ 呈多项式关系。输出规模关于 SizeFundUnits$(2d, H_1)$ 呈多项式关系。

算法 8 的复杂度分析。首先应当指出，判断 f 是否为两平方和或四个平方和（分别通过观察 8 或 [36, 定理 17.2] ）的代价至多与计算相应分解的代价相同。因此，总复杂度由分解为两平方和、四个平方和以及多项式因式分解的代价决定。

如果 f的次数为 d且系数的高度为 $\leq H$，那么存在某个 $H_1 =$ poly$(d, H, \inf(f), \inf(f_*))$，该函数由算法 1、算法 7 以及 $u \mapsto 2ud+8d(1+\lg(d))$ 映射给出，使得总预期二进制成本关于 IntFact(H)、PolyFact(d, H_1)、UnitGroupComp$(2d, H_1)$ 和 SizeFundUnits$(2d, H_1)$ 是多项式级的。输出大小关于 SizeFundUnits$(2d, H_1)$ 是多项式级的 . □

注19。 在算法 7 中，要么 $g_2 = 0$，要么 $g_1 = g_2$。这意味着对于某个 $a_5 = 1$ 或 2，算法 8 的输出具有以下形式

$$f = f_1^2 + f_2^2 + f_3^2 + f_4^2 + a_5 f_5^2.$$

算法 8 将单变量有理系数非负多项式分解为六平方和 输入：非负多项式 $f \in \mathbb{Q}[x]$。输出：多项式 $f_1, \ldots, f_6 \in \mathbb{Q}[x]$，使得 $f_1^2 + \cdots + f_6^2 = f$. 1: 如果 f 是一个平方，则 2: 返回 $f_1 := \mathrm{p}\sqrt{f}$，$f_2 := \cdots f_6 := 0$. 3: 如果 f是两平方和 {使用观察 8 进行检查}，则 4: 执行算法

1 $f_1\ f_2 \in \mathbb{Q}[x]$ $f_1^2 +$ 以得 $f_2^2 = f$ 到 ，使得 。 5: 返回 f_1, f_2 和 $f_3 := \cdots f_6 := 0$. 6: 如果 f是四个平方和 {使用 [36, 定理 17.2] 进行检查}，则 7: 执行算法 5，得到 $f_1, \ldots, f_4 \in \mathbb{Q}[x]$，使得 $f_1^2 + \cdots + f_4^2 = f$.

8 f_1 f_4 $f_5 = f_6 = 0$: 返回 , . . . , 和 : : : 计算

9 $f = g \cdot h^2$ 的无平方因子分解，其中 $g, h \in \mathbb{Q}[x]$ 且 g 无平方因子。 10: 执行算法 7，以 g 为输入，得到 $g_1, g_2 \in \mathbb{Q}[x]$，使得 $g - g_1^2 - g_2^2$ 是 $\mathbb{Q}[x]$中的四个平方和。 11: 执行算法 5，将 $g - g_1^2 - g_2^2$ 分解为 $\mathbb{Q}[x]$中的四个平方和。将输出记为 $g_3, \ldots, g_6$。 12: 返回 $f_1 := g_1 h, \ldots, f_6 := g_6 h$。

换句话说，输出是一个*5* 个平方的加权和。

注 20。 可以调整算法 1 和算法 5，以输出形如 $f = \mathrm{lc}(f) f_1^2 + \mathrm{lc}(f) f_2^2$或 $f = \mathrm{lc}(f) f_1^2 + \cdots + \mathrm{lc}(f) f_4^2$的加权平方和。利用前面的注释，随后可以调整算法 8，以输出一个*5* 个平方的加权和

$$f = a_1 f_1^2 + \cdots + a_5 f_5^2$$

其中的 a_i均为非负整数。这样做的一大好处是，调整后的算法 1 无需对 $\mathrm{lc}(f)$ 进行整数因式分解，因此也不再需要引用 IntFact。

算法 8 并非最优，因为它会产生**六个**平方，而众所周知实际上只需要**五个**。下文介绍算法 9，它是算法 6 的一个变体。

我们对 20 000 多个随机非负单变量多项式进行了实证验证，算法 9 对它们都运行良好。然而，我们仍不知道如何证明它最终会终止，除非 $\mathrm{ord}_2 c_0$ 或 $\mathrm{ord}_2 c_d$ 为奇数；在这两种情况下，它都会归结为算法 6。另一方面，该算法输出结果的正确性显而易见。因此，如果我们能证明它会停止，就可以用它将任意非负多项式分解为五个平方之和（而不是六个），从而得到所需的分解。这是本工作留下的首个后续研究方向。我们还打算将我们的算法与文献 [25] 中分析的其他算法进行比较。

参考文献

[1] Frédéric Besson. 2007. 快速反身算术策略：线性情形及其他。载于国际证明与程序类型研讨会。Springer，柏林海德堡，48–62。[2] Jean-François Biasse 和 Claus Fieker。2014. 高次数数域中的次指数类群与单位群计算。伦敦数学学会计算与数学期刊 17, A (2014), 385–403。

算法 9 化归为四个平方和

输入： 一个正的无平方因子多项式 $f = c_0+c_1x+\cdots+c_dx^d \in \mathbb{Q}[x]$。

输出： 一个多项式 $h \in \mathbb{Q}[x]$，使得 $f - h^2$ 是四个（或更少个）平方的和。

1: **如果** f 是四个平方和**那么**
2: **返回** $h := 0$。
3: 设 $f_* := c_d + c_{d-1}x + \cdots + c_0x^d$。
4: 求一个正数 ε 使得

$$\varepsilon < \text{下确界}\{f(x) \mid x \in \mathbb{R}\} \quad \text{and} \quad \varepsilon < \inf\{f_*(x) \mid x \in \mathbb{R}\}.$$

5: 初始化 $l := \lceil -\tfrac{1}{2} \cdot \lg \varepsilon \rceil$.
6: **当** True**时执行**
7: **如果** $f - 2^{-2l}$ 在 $\mathbb{Q}_2[x]$ 中不可约**则**
8: **返回** $h := 2^{-l}$.
9: **如果** $f - 2^{-2l}x^d$ 在 $\mathbb{Q}_2[x]$ 中不可约，**则**
10: **返回** $h := 2^{-l}x^d/2$。
11: $l := l + 1$.

[3] Jean-François Biasse 和 Fang Song。2016. 用于计算类群并解决任意次数数域中主理想问题的高效量子算法。载于第二十七届 *ACM-SIAM* 离散算法年度研讨会论文集。SIAM，阿灵顿，弗吉尼亚州，美国，893–902。[4] Alin Bostan、Frédéric Chyzak、Marc Giusti、Romain Lebreton、Grégoire Lecerf、Bruno Salvy 和 Éric Schost。2017. 符号计算中的高效算法。Frédéric Chyzak（自费出版），帕莱索。https://hal.archives-ouvertes.fr/AECF/ 686页。由CreateSpace印制。另有电子版。[5] Richard T. Bumby。1996. 四平方和。载于数论（纽约，*1991–1995*）。Springer，纽约，1–8。[6] David G. Cantor 和 Daniel M. Gordon。2000. 在 p-进域上分解多项式。载于算法数论（莱顿，*2000*）。《计算机科学讲义》，第1838卷。Springer，柏林，185–208。https://doi.org/10.1007/10722028_10。[7] Sylvain Chevillard、John Harrison、Mioara Joldeş 和 Ch Lauter。2011. 高效且精确地计算逼近误差的上界。《理论计算机科学》412, 16 (2011), 1523–1543。[8] Henri Cohen。1993. 计算代数数论教程。第8卷。施普林格出版社，柏林。[9] Henri Cohen。2000. 计算数论进阶主题。《数学研究生教材》，第193卷。施普林格出版社，纽约。xvi+578 页。https://doi.org/10.1007/978-1-4419-8489-0 https://doi.org/10.1007/978-1-4419-8489-0。[10] C. Fieker、A. Jurk 和 M. Pohst。1997. 关于代数数域中的相对范数方程求解。《数学计算》66, 217 (1997), 399–410。https://doi.org/10.1090/S0025-5718-97-00761-8 https://doi.org/10.1090/S0025-5718-97-00761-8。[11] U. Fincke 和 M. Pohst。1983. 确定给定范数的代数整数的一种方法。载于计算机代数（伦敦，*1983*）。《计算机科学讲义》，第162卷。Springer，柏林，194–202。https://doi.org/10.1007/3-540-12868-9_103 https://doi.org/10.1007/3-540-12868-9_103。[12] Dennis A. Garbanati。1980. 求一个范数为给定有理数的代数数的算法。纯粹与应用数学杂志316 (1980), 1–13。https://doi.org/10.1515/crll.1980.316.1 https://doi.org/10.1515/crll.1980.316.1。[13] Alexandre Gélin。2017. 数域中的类群计算及其在密码学中的应用。博士论文。皮埃尔和玛丽·居里大学（巴黎第六大学）。[14]Alexandre Gélin 和 Antoine Joux。2016. 化简数域定义多项式：在类群计算中的应用。伦敦数学学会计算与数学期刊 19, A (2016), 315–331。https://doi.org/10.1112/S1461157016000255[15] Thomas Hales、Mark Adams、Gertrud Bauer、Tat Dat Dang、John Harrison、Hoang Le Truong、CezaryKaliszyk、Victor Magron、Sean McLaughlin、Tat Thang Nguyen 等。2017. 开普勒猜想的形式化证明。载于《数学论坛》，*Pi*，第5卷。剑桥大学出版社，英国，1–29。[16] Didier Henrion、Simone Naldi 和 Mohab Safey El Din。2019. SPECTRA——用于以精确算术求解线性矩阵不等式的 Maple 程序库。优化方法与软件34, 1 (2019), 62–78。[17] Teresa Krick、Bernard Mourrain 和 Agnes Szanto。2021. 单变量有理平方和。arXiv:2112.00490 [18] Tsit Yuen Lam. 2005.《域上的二次型导论》。数学研究生教材丛书，第67卷。美国数学会，罗德岛州普罗维登斯。xxii+550 页。[19] Edmund Landau. 1906.《关于定函数的平方表示》。《数学年刊》62, 2 (1906), 272–285。[20] Jean B Lasserre. 2001.《多项式全局优化与矩问题》。SIAM 优化学报11, 3 (2001), 796–817。[21] Victor Magron、Xavier Allamigeon、Stéphane Gaubert 和 Benjamin Werner. 2015.《非线性优化的形式化证明》。《形式化推理学报》8, 1 (2015), 1–24。https://doi.org/10.6092/1972-5787-4319(1月)，–。[22] 2017. https://doi.org/。/issn。-/ Victor Magron、George Constantinides 和 Alastair Donaldson. .《利用半定规划认证舍入误差界》。《ACM 数学软件汇刊》(TOMS) 43, 4 (2017), 1–31。[23] Victor Magron 和 Mohab Safey El Din. 2018.《RealCertify：用于非负性认证的 Maple 软件包》。《ACM 计算机代数通讯》52, 2 (2018), 34–37。[24] Victor Magron 和 Mohab Safey El Din. 2021.《关于 Reznick、Hilbert-Artin 和 Putinar 表示的精确形式》。《符号计算学报》107 (2021), 221–250。[25] Victor Magron、Mohab Safey El Din 和 Markus Schweighofer. 2019.《非负单变量多项式加权平方和分解算法》。《符号计算学报》93 (2019), 200–220。[26] Victor Magron、Mohab Safey El Din、Markus Schweighofer 和 Trung Hieu Vu. 2022.《高斯系数三角一元多项式的精确 SOHS 分解》。载于《2022年国际符号与代数计算研讨会论文集》。ACM，法国利摩日，325–332。[27] Victor Magron、Mohab Safey El Din 和 Trung-Hieu Vu. 2022.《具有有理系数的多项式在其梯度理想上的平方和分解》。arXiv:2107.11825，已接受发表于 SIAM 优化学报。[28] Victor Magron、Henning Seidler 和 Timo De Wolff. 2019.《通过非负回路和算术-几何平均指数进行精确优化》。载于《2019年国际符号与代数计算研讨会论文集》。ACM，中国北京，291–298。[29] Victor Magron 和 Jie Wang. 2023.《通过二阶锥规划进行 SONC 优化和精确非负性认证》。《符号计算学报》115 (2023), 346–370。[30] Pablo A Parrilo. 2000.《鲁棒性与优化中的结构化半定规划及半代数几何方法》。美国加州理工学院。[31] Helfried Peyrl 和 Pablo A Parrilo. 2008.《计算有理系数平方和分解》。《理论计算机科学》409, 2 (2008), 269–281。[32] Paul Pollack 和 Peter Schorn. 2019.《狄利克雷三平方定理的证明：算法视角》。《数学计算》88, 316 (2019), 1007–1019。https://doi.org/10.1090/mcom/3349 https://doi.org/10.1090/mcom/3349。[33] Paul Pollack 和 Enrique Treviño. 2018.《求出拉格朗日定理中的四个平方数》。《整数》18A (2018), 论文编号 A15, 16。[34] Yves Pourchet. 1971.《代数数域上单变量多项式的平方和表示》。《算术学报》 19, 1 1971 89 104 ( ), –。[35] Michael O. Rabin 和 Jeffery O. Shallit. 1986.《数论中的随机化算法》。《纯粹与应用数学通讯》39, S, 增刊 (1986), S239–S256。https://doi.org/10.1002/cpa.3160390713《数学科学前沿：1985》（纽约，1985）。[36] A. R. Rajwade. 1993.《平方》。伦敦数学会讲义系列，第171卷。剑桥大学出版社，英国。xii+286 页。https://doi.org/10.1017/9780511566028。/CBO Anders Rantzer 和 Pablo A Parrilo. .《非线性系统稳定化中的凸性》。载于《第39届 IEEE 决策与控制会议论文集》（第00CH37187号），第3卷。IEEE，澳大利亚悉尼，2942–2945。[38] René Schoof. 1985.《有限域上的椭圆曲线与模 p 平方根的计算》。《数学计算》44, 170 (1985), 483–494。http://www.jstor.org/stable/2007968 [39] 1999 Markus Schweighofer. .《关于非负性与正性定理的算法证明》。硕士论文，帕绍大学136 (1999), 1–105。[40] Denis Simon. 2002.《利用 S-单位求解相对数域中的范数方程》。《数学计算》71, 239 (2002), 1287–1305。https://doi.org/10.1090/S0025-5718-02-01309-1 https://doi.org/10.1090/S0025-5718-02-01309-1. [41] Christoph Thiel. 1994.《在广义黎曼猜想成立的假设下，验证类数属于 NP ∩co-NP》。载于《算法数论》，Leonard M. Adleman 和 Ming-Deh Huang（编）。Springer Berlin Heidelberg，柏林，海德堡，234–247。[42] Christoph Thiel. 1995.《利用代数整数紧凑表示的短证明》。《复杂性学报》11, 3 (1995), 310–329。https://doi.org/10.1006/jcom.1995.1014

[43]Stan Wagon. 1990.《编者专栏：欧几里得算法再次出击》。《美国数学月刊》 97, 2 (1990), 125–129。http://www.jstor.org/stable/2323912