Math-Net.Ru

Общероссийский математический портал

А. Э. Маевский, Алгоритм вычисления корней многочленов с коэффициентами из кольца многочленов над произвольной областью целостности, *Матем. заметки*, 2009, том 85, выпуск 1, 73–88

DOI: <https://doi.org/10.4213/mzm4034>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 3.81.55.215

26 апреля 2023 г., 15:18:37

УДК 512.622

Алгоритм вычисления корней многочленов с коэффициентами из кольца многочленов над произвольной областью целостности

А. Э. Маевский

Построен детерминированный алгоритм вычисления корней многочленов одной переменной с коэффициентами из кольца многочленов нескольких переменных над произвольной областью целостности. Получена оценка арифметической сложности алгоритма в наихудшем случае.

Библиография: 11 названий.

1. Постановка задачи. Пусть $\mathbb{N}$ – множество натуральных чисел, $\mathbb{Z}_{\geq 0} = \mathbb{N} \cup \{0\}$, $\mathbb{Z}_{\geq 0}^n = \mathbb{Z}_{\geq 0} \times \cdots \times \mathbb{Z}_{\geq 0}$ (n раз), $\mathbb{D}$ – область целостности (коммутативное ассоциативное кольцо с единицей без делителей нуля), $\mathbb{D}[x_1, \dots, x_n]$ – кольцо многочленов от n переменных ($n > 0$) с коэффициентами из $\mathbb{D}$, $\mathbb{D}[x_1, \dots, x_n][T]$ – кольцо многочленов от переменной T с коэффициентами из $\mathbb{D}[x_1, \dots, x_n]$, отождествляемое с кольцом $\mathbb{D}[x_1, \dots, x_n, T]$. Для любого

$$f = \sum_{\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{Z}_{\geq 0}^n} f_{\alpha} x_1^{\alpha_1} \cdots x_n^{\alpha_n} \quad (\in \mathbb{D}[x_1, \dots, x_n])$$

рассмотрим множество

$$\Gamma_f \stackrel{\text{def}}{=} \{\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{Z}_{\geq 0}^n \mid f_{\alpha} \neq 0\}.$$

Под *полной степенью* многочлена f будем понимать число

$$\deg(f) \stackrel{\text{def}}{=} \max_{(\alpha_1, \dots, \alpha_n) \in \Gamma_f} \{\alpha_1 + \cdots + \alpha_n\},$$

равное максимальной из степеней мономов, входящих в f . Под T -*степенью* многочлена $Q(T)$ ($\in \mathbb{D}[x_1, \dots, x_n][T]$) будем понимать число $\deg_T(Q(T))$, равное максимальному показателю степени переменной T , с которым она входит в $Q(T)$. Многочлен f ($\in \mathbb{D}[x_1, \dots, x_n]$) будем называть T -*корнем* многочлена $Q(T)$, если $Q(f) \equiv 0$.

Для заданного многочлена $Q(T)$ ($\in \mathbb{D}[x_1, \dots, x_n][T]$) и заданного целого числа d (> 0) рассмотрим множество

$$\Omega_{Q,n}(d) = \{f \in \mathbb{D}[x_1, \dots, x_n] \mid \deg(f) \leq d, Q(f) \equiv 0\} \quad (1.1)$$

всех T -корней $Q(T)$ полной степени не выше d . Отметим, что множество (1.1) находится во взаимно однозначном соответствии с множеством делителей $Q(T)$ вида $(T - f)$, где $f \in \mathbb{D}[x_1, \dots, x_n]$ и $\deg(f) \leq d$. Как следствие, мощность множества $\Omega_{Q,n}(d)$ не превышает T -степени $Q(T)$. Целью данной работы является построение алгоритма, вычисляющего для произвольных многочлена $Q(T)$ ($\in \mathbb{D}[x_1, \dots, x_n][T]$) и целого числа d (> 0) множество $\Omega_{Q,n}(d)$.

В случае $n = 1$ и $\mathbb{D} = \mathbb{F}_q$ (конечное поле из q элементов) рассматриваемая задача решена в работе Рота и Руkenштейн [1]. Они описали рекурсивный алгоритм, по-ко-эффективно вычисляющий элементы множества $\Omega_{Q,1}(d)$, и привели краткую схему доказательства его корректности. В настоящей статье построен алгоритм вычисления множества $\Omega_{Q,n}(d)$ для произвольного n (> 0) и тех областей $\mathbb{D}$, для которых существует алгоритм поиска всех T -корней произвольного многочлена из $\mathbb{D}[T]$ (если $\mathbb{D} = \mathbb{F}_q$ или $\mathbb{D} = \mathbb{Z}$, такие алгоритмы существуют и приведены, например, в [2; гл. 3.4, 3.5]). В случае $n = 1$ построенный алгоритм на уровне идей совпадает с алгоритмом Рота и Руkenштейн, поэтому может рассматриваться как распространение алгоритма Рота–Руkenштейн на случай колец многочленов многих переменных над $\mathbb{D}$.

Алгоритм вычисления множества $\Omega_{Q,n}(d)$ в случае $\mathbb{D} = \mathbb{F}_q$ может быть применен в теории помехоустойчивого кодирования, а именно, в задачах списочного декодирования линейных кодов [3]. Отметим, что Рот и Руkenштейн построили свой алгоритм для реализации части списочного декодера Судана кодов Риды–Соломона. В работе [4] списочный декодер Судана перенесен на специальный класс алгебро-геометрических кодов на проективных кривых, при этом возникла задача вычисления всех T -корней степени не выше d (параметр, совпадающий с размерностью кода) некоторого многочлена $Q(T)$ либо из кольца $\mathbb{F}_q[x_1, x_2][T]$, либо из кольца $\mathbb{F}_q[x_1, x_2, x_3][T]$. Построенный в настоящей работе алгоритм может быть применен к решению этой задачи. Представляется, что в случае $n > 3$ он может быть полезен также при построении списочного декодера алгебро-геометрических кодов на проективных гиперповерхностях.

Следует отметить, что если область $\mathbb{D}$ задана, и существует алгоритм факторизации многочленов нескольких переменных над $\mathbb{D}$ (в случае конечных и алгебраических полей см., например, обзор [5]), то этот алгоритм можно применить для вычисления множества $\Omega_{Q,n}(d)$. Однако если многочлен $Q(T)$ имеет большое количество делителей, не являющихся T -корнями, то часть вычислительного времени алгоритм факторизации потратит на поиск этих делителей, что в результате может привести к нецелесообразности его применения для поиска T -корней степени не выше d .

Статья организована следующим образом. В п. 2 вводятся специальные последовательности многочленов, зависящие от $Q(T)$ и некоторого его T -корня f , рассматривается ряд полезных свойств этих последовательностей. В п. 3 построен алгоритм FindRoots нахождения множества $\Omega_{Q,n}(d)$ для произвольного ненулевого многочлена $Q(T)$ и приведено обоснование его корректности. Пункт 4 посвящен оценке арифметической сложности построенного алгоритма в наихудшем случае. Отметим, что многие леммы и теоремы, доказанные в работе, могут быть полезны при построении программной реализации алгоритма.

2. Специальные многочлены и некоторые свойства T -корней. Рассмотрим произвольный многочлен $Q(T)$ из $\mathbb{D}[x_1, \dots, x_n][T]$. Пусть r – такое неотрицательное целое число, что x_n^r делит $Q(T)$, но x_n^{r+1} не делит $Q(T)$. Положим

$$Q^{(x)}(T) \stackrel{\text{def}}{=} \frac{Q(T)}{x_n^r}.$$

Пусть $I_n = (x_n)$ – простой главный идеал в $\mathbb{D}[x_1, \dots, x_n]$, порожденный x_n , σ_n – канонический эпиморфизм кольца $\mathbb{D}[x_1, \dots, x_n]$ на факторкольцо $\mathbb{D}[x_1, \dots, x_n]/I_n$, сопоставляющий многочлену f класс эквивалентности $f + I_n$, а τ_n – естественный изоморфизм $\mathbb{D}[x_1, \dots, x_n]/I_n$ и $\mathbb{D}[x_1, \dots, x_{n-1}]$. Через ψ_n обозначим композицию

$$\psi_n = \tau_n \circ \sigma_n : \mathbb{D}[x_1, \dots, x_n] \rightarrow \mathbb{D}[x_1, \dots, x_{n-1}].$$

Аналогичным образом, пусть $J_n = (x_n)$ – простой главный идеал в $\mathbb{D}[x_1, \dots, x_n][T]$, σ_n^T – канонический эпиморфизм $\mathbb{D}[x_1, \dots, x_n][T]$ на $\mathbb{D}[x_1, \dots, x_n][T]/J_n$, τ_n^T – естественный изоморфизм колец $\mathbb{D}[x_1, \dots, x_n][T]/J_n$ и $\mathbb{D}[x_1, \dots, x_{n-1}][T]$,

$$\psi_n^T = \tau_n^T \circ \sigma_n^T : \mathbb{D}[x_1, \dots, x_n][T] \rightarrow \mathbb{D}[x_1, \dots, x_{n-1}][T].$$

Отметим, что $\psi_n(f)$ (или $\psi_n^T(Q(T))$) можно рассматривать как подстановку в многочлен f (в многочлен $Q(T)$) нулевого элемента $\mathbb{D}$ вместо переменной x_n . Кроме того, поскольку для любого f из $\mathbb{D}[x_1, \dots, x_n]$ и любого неотрицательного m верно соотношение

$$\psi_n^T(fT^m) = \psi_n(f)T^m,$$

для любого многочлена $Q(T) = \sum_{m=0}^{\deg_T(Q(T))} q_m T^m$, $q_m \in \mathbb{D}[x_1, \dots, x_n]$, имеем

$$\psi_n^T(Q(T)) = \sum_{m=0}^{\deg_T(Q(T))} \psi_n(q_m) T^m. \quad (2.1)$$

Пусть f – произвольный многочлен полной степени d из кольца $\mathbb{D}[x_1, \dots, x_n]$. Для всех целых $i \in [0, d]$ рекуррентно определим многочлены f_i ($\in \mathbb{D}[x_1, \dots, x_n]$) и $Q_i(T)$, $Q_i^{(x)}(T)$ из $\mathbb{D}[x_1, \dots, x_n][T]$ следующим образом. Положим $f_0 = f$, $Q_0(T) = Q_0^{(x)}(T) = Q^{(x)}(T)$, и для всех $i \in [1, d]$

$$f_i = \frac{(f_{i-1} - \psi_n(f_{i-1}))}{x_n}, \quad (2.2)$$

$$Q_i(T) = Q_{i-1}^{(x)}(x_n T + \psi_n(f_{i-1})), \quad (2.3)$$

$$Q_i^{(x)}(T) = \frac{Q_i(T)}{x_n^{r_i}}, \quad (2.4)$$

где r_i (≥ 0) – такое целое число, что $x_n^{r_i}$ делит $Q_i(T)$, а $x_n^{r_i+1}$ не делит $Q_i(T)$.

В следующих трех леммах и теореме 4 сформулируем и докажем некоторые свойства многочленов f_i , $Q_i(T)$ и $Q_i^{(x)}(T)$, которые понадобятся нам в дальнейшем.

ЛЕММА 1. Для всех $i \in [0, d]$ многочлен f_i имеет полную степень не выше $d - i$.

ДОКАЗАТЕЛЬСТВО. При $i = 0$ многочлен $f_0 = f$ согласно выбору имеет степень не выше d . Предположим, что многочлен f_j для некоторого $j \in [0, d - 1]$ имеет полную степень не выше $d - j$. Запишем f_j как $f_j = f_j^{(1)} + f_j^{(2)}$, где $f_j^{(1)}$ состоит из мономов f_j , не содержащих переменную x_n , а $f_j^{(2)}$ состоит из мономов f_j , содержащих переменную x_n . Нетрудно видеть, что $\deg(f_j^{(1)}) \leq d - j$, $\deg(f_j^{(2)}) \leq d - j$ и $\psi_n(f_j) = f_j^{(1)}$. Поэтому $f_{j+1} = (f_j - \psi_n(f_j))/x_n = f_j^{(2)}/x_n$ и $\deg(f_{j+1}) \leq d - j - 1$.

ЛЕММА 2. Пусть f – произвольный многочлен из $\mathbb{D}[x_1, \dots, x_n]$, $d = \deg(f)$. Имеет место равенство

$$f = \sum_{i=0}^d x_n^i \psi_n(f_i). \quad (2.5)$$

ДОКАЗАТЕЛЬСТВО. Воспользуемся методом индукции по полной степени многочлена f . При $d = 0$ многочлен f является константой, и утверждение леммы выполнено, так как в этом случае $\psi_n(f_0) = f$. Пусть утверждение леммы выполняется для всех многочленов f , полная степень которых не превышает s . Рассмотрим многочлен f полной степени $s + 1$ вместе с последовательностью многочленов f_i , вычисленных по формуле (2.2). Согласно лемме 1 многочлен f_1 имеет степень не выше s , поэтому по индуктивному предположению $f_1 = \sum_{i=0}^s x_n^i \psi_n(f'_i)$, где f'_i – последовательность многочленов, вычисленных по формуле (2.2) для многочлена f_1 . Легко видеть, что

$$f'_i = f_{i+1}, \quad f = \psi_n(f) + x_n f_1.$$

Следовательно,

$$f = \psi_n(f) + \sum_{i=0}^s x_n^{i+1} \psi_n(f_{i+1}) = \sum_{i=0}^{s+1} x_n^i \psi_n(f_i).$$

ЛЕММА 3. Рассмотрим произвольное целое число $i \in [1, d]$. Многочлен $(T - f_i)$ делит многочлен $Q_i^{(x)}(T)$ в том и только в том случае, когда многочлен $(T - f_{i-1})$ делит многочлен $Q_{i-1}^{(x)}(T)$.

ДОКАЗАТЕЛЬСТВО. Пусть $(T - f_i)$ делит многочлен $Q_i^{(x)}(T) = Q_i(T)/x_n^{r_i}$. Тогда $(T - f_i)$ делит также $Q_i(T) = Q_{i-1}^{(x)}(x_n T + \psi_n(f_{i-1}))$. Следовательно,

$$Q_{i-1}^{(x)}(x_n T + \psi_n(f_{i-1})) = (T - f_i)U(T)$$

для некоторого $U(T) \in \mathbb{D}[x_1, \dots, x_n][T]$. В последнем равенстве сделаем замену переменных $T \mapsto (T - \psi_n(f_{i-1}))/x_n$ и получим

$$Q_{i-1}^{(x)}(T) = \left(\frac{(T - \psi_n(f_{i-1}))}{x_n} - f_i \right) U \left(\frac{(T - \psi_n(f_{i-1}))}{x_n} \right).$$

Умножим обе части полученного равенства на x_n^L , где L – достаточно большое натуральное число, и с учетом (2.2) получим

$$x_n^L Q_{i-1}^{(x)}(T) = (T - f_{i-1})V(T),$$

$V(T) \in \mathbb{D}[x_1, \dots, x_n][T]$. Таким образом, $(T - f_{i-1})$ делит $x_n^L Q_{i-1}^{(x)}(T)$, и, так как x_n^L и $(T - f_{i-1})$ взаимно просты, то $(T - f_{i-1})$ делит $Q_{i-1}^{(x)}(T)$.

Пусть теперь $(T - f_{i-1})$ делит многочлен $Q_{i-1}^{(x)}(T)$, т.е.

$$Q_{i-1}^{(x)}(T) = (T - f_{i-1})U(T)$$

для некоторого многочлена $U(T)$ ($\in \mathbb{D}[x_1, \dots, x_n][T]$). Используя это соотношение из (2.3) получим

$$Q_i(T) = (x_n T + \psi_n(f_{i-1}) - f_{i-1})U(x_n T + \psi_n(f_{i-1})) = x_n(T - f_i)V(T),$$

где $V(T) = U(x_n T + \psi_n(f_{i-1}))$. Следовательно, $(T - f_i)$ делит $Q_i(T)$. Но $Q_i(T) = x_n^{r_i} Q_i^{(x)}(T)$, поэтому $(T - f_i)$ делит и $Q_i^{(x)}(T)$.

ТЕОРЕМА 4. *Следующие утверждения эквивалентны:*

- (i) $(T - f)$ делит $Q(T)$;
- (ii) $\exists i \in [1, d]: (T - f_i)$ делит $Q_i^{(x)}(T)$;
- (iii) $\forall i \in [1, d]: (T - f_i)$ делит $Q_i^{(x)}(T)$;
- (iv) T делит $Q_{d+1}(T)$, где $Q_{d+1}(T) = Q_d^{(x)}(x_n T + \psi_n(f_d))$.

ДОКАЗАТЕЛЬСТВО. Импликация (i) $\Rightarrow$ (ii). Отметим, что утверждение (i) эквивалентно следующему: $(T - f_0)$ делит $Q_0(T)$. Тогда согласно лемме 3 $(T - f_1)$ делит $Q_1(T)$; следовательно, i в утверждении (ii) можно положить равным 1.

Импликация (ii) $\Rightarrow$ (iii) следует из леммы 3.

Импликация (iii) $\Rightarrow$ (iv). Пусть имеет место (iii). Тогда $Q_d^{(x)}(T) = (T - f_d)U(T)$, и $Q_{d+1}(T) = (x_n T + \psi_n(f_d) - f_d)U(x_n T + \psi_n(f_d))$. Но согласно лемме 1 $\deg(f_d) = 0$, поэтому $\psi_n(f_d) = f_d$, и $Q_{d+1}(T) = (x_n T)U(x_n T + \psi_n(f_d))$. Следовательно, T делит $Q_{d+1}(T)$.

Импликация (iv) $\Rightarrow$ (i). Пусть T делит $Q_{d+1}(T) = Q_d^{(x)}(x_n T + \psi_n(f_d))$. Используя те же рассуждения, что и при доказательстве первой части леммы 3, и учитывая, что $\psi_n(f_d) = f_d$, получим, что $(T - f_d)$ делит $Q_d^{(x)}(T)$. Используя далее лемму 3, получим, что $(T - f)$ делит $Q(T)$.

Для всех целых $i \in [0, d]$ рекуррентно определим многочлены $M_i(T)$ ($\in \mathbb{D}[x_1, \dots, x_{n-1}][T]$) следующим образом:

$$M_i(T) = \psi_n^T(Q_i^{(x)}(T)). \quad (2.6)$$

Отметим, что если многочлен $Q_i^{(x)}(T)$ ненулевой, то таким же будет и $M_i(T)$.

ТЕОРЕМА 5. *Если $(T - f)$ делит $Q(T)$, то для всех целых $i \in [0, d]$ многочлен $(T - \psi_n(f_i))$ делит многочлен $M_i(T)$.*

ДОКАЗАТЕЛЬСТВО. Пусть $(T - f)$ делит $Q(T)$. Тогда согласно теореме 4 многочлен $(T - f_i)$ делит многочлен $Q_i^{(x)}(T)$ для всех целых $i \in [0, d]$. Так как ψ_n^T — кольцевой гомоморфизм, то $\psi_n^T(T - f_i) = (T - \psi_n(f_i))$ делит $\psi_n^T(Q_i^{(x)}(T)) = M_i(T)$.

ЗАМЕЧАНИЕ. Последнюю теорему можно переформулировать следующим образом: *если f является T -корнем полной степени d многочлена $Q(T)$, то для всех $i \in [0, d]$ многочлен $\psi_n(f_i)$ принадлежит множеству $\Omega_{M_i, (n-1)}(d-i)$ всех T -корней полной степени $d-i$ многочлена $M_i(T)$ (см. (1.1)).*

Теорема 5 доставляет нам метод нахождения T -корней $Q(T)$. Согласно (2.5) любой T -корень можно представить в виде $f = \sum_{i=0}^d x_n^i \psi_n(f_i)$, поэтому, зная все $\psi_n(f_i)$, можно восстановить f . Отметим, что множество $\Omega_{M_0, (n-1)}(d)$ зависит только от известного многочлена $Q(T)$. Предположим, что мы умеем находить $\Omega_{M_0, (n-1)}(d)$ для произвольного $Q(T)$. Тогда в соответствии с замечанием к теореме 5 для любого T -корня f многочлена $Q(T)$ многочлен $\psi_n(f_0)$ принадлежит $\Omega_{M_0, (n-1)}(d)$. Следовательно, если с помощью каждого g из $\Omega_{M_0, (n-1)}(d)$ вычислить $Q_1(T)$ согласно (2.3), полагая $\psi_n(f_0) = g$, то можно найти множества $\Omega_{M_1, (n-1)}(d-1)$ всех кандидатов на элемент $\psi_n(f_1)$. Продолжив этот процесс до тех пор, пока не будут вычислены кандидаты на $\psi_n(f_d)$, можно согласно (2.5) вычислить множество Θ кандидатов на T -корни $Q(T)$. Те многочлены из Θ , для которых T делит $Q_{d+1}(T)$, согласно построению и теореме 4 будут являться T -корнями $Q(T)$. В следующем пункте мы формализуем описанный процесс в виде рекурсивного алгоритма, включающего в себя также метод нахождения множеств $\Omega_{M_i, (n-1)}(d-i)$, и докажем, что результат его работы всегда совпадает с множеством $\Omega_{Q, n}(d)$.

3. Алгоритм вычисления множества $\Omega_{Q, n}(d)$. Рассмотрим следующий алгоритм **FindRoots** поиска всех T -корней полной степени не выше d многочлена $Q(T)$. Он использует рекурсивную процедуру **RecursiveSearch**, построенную согласно методу, изложенному в конце предыдущего пункта. Будем полагать известным алгоритм **FindRootsOne** нахождения всех T -корней произвольного многочлена $Q(T) \in \mathbb{D}[T]$ (в случае $\mathbb{D} = \mathbb{F}_q$ или $\mathbb{D} = \mathbb{Z}$ см., например, [2; гл. 3.4, 3.5]). Символом $\emptyset$ будем обозначать пустое множество.

АЛГОРИТМ FindRoots (поиск для многочлена $Q(T)$ всех T -корней из кольца $\mathbb{D}[x_1, \dots, x_n]$ полной степени не выше d):

Вход: ненулевой многочлен $Q(T) \in \mathbb{D}[x_1, \dots, x_n][T]$, числа $d \in \mathbb{Z}_{\geq 0}$, $n \in \mathbb{N}$;

Выход: множество всех T -корней полной степени не выше d многочлена $Q(T)$;

1. вычислить $\Theta_{Q, n}(d) = \text{RecursiveSearch}(Q(T), d, 0, n, 0, \emptyset)$;
2. вернуть множество $\Theta_{Q, n}(d)$ и выйти из алгоритма.

Конец алгоритма.

ПРОЦЕДУРА RecursiveSearch (вспомогательная рекурсивная процедура):

Вход: ненулевой многочлен $Q(T) \in \mathbb{D}[x_1, \dots, x_m][T]$, числа $\delta, i \in \mathbb{Z}_{\geq 0}$ такие, что $\delta \geq i$, число $m \in \mathbb{N}$, многочлен $f \in \mathbb{D}[x_1, \dots, x_m]$, множество Θ многочленов из $\mathbb{D}[x_1, \dots, x_m]$;

Выход: множество многочленов из $\mathbb{D}[x_1, \dots, x_m]$ полной степени не выше $\delta + \deg(f)$;

1. если $Q(T)$ имеет вид $q_1 T + q_0$ для некоторых $q_1, q_0 \in \mathbb{D}[x_1, \dots, x_m]$, то
 - 1.1. если $(q_1 = 0)$ или $(q_1$ не делит $q_0)$, то вернуть пустое множество и выйти из процедуры;
 - 1.2. если полная степень многочлена $-q_0/q_1$ не превышает δ , то вернуть $\{-q_0/q_1\}$ и выйти из процедуры, иначе вернуть пустое множество и выйти из процедуры;

2. найти такое целое неотрицательное число r , что x_m^r делит $Q(T)$, но x_m^{r+1} не делит $Q(T)$;
3. вычислить $Q^{(x)}(T) = Q(T)/x_m^r$, $M(T) = \psi_m^T(Q^{(x)}(T))$;
4. если $m = 1$, то вычислить $\Phi_{m,i} = \text{FindRootsOne}(M(T))$, иначе вычислить $\Phi_{m,i} = \text{RecursiveSearch}(M(T), \delta - i, 0, m - 1, 0, \emptyset)$;
5. если $\Phi_{m,i}$ непусто, то для каждого из многочленов $h_{k,i} \in \Phi_{m,i}$ выполнить:
 - 5.1. вычислить $R_1(T) = Q^{(x)}(T + h_{k,i})$;
 - 5.2. вычислить $R_2(T) = R_1(x_m T)$;
 - 5.3. если $(i = \delta)$, то
 - 5.3.1. если T делит $R_2(T)$, то вернуть множество $\{f + x_m^\delta h_{k,i}\}$, иначе вернуть пустое множество;
 - 5.3.2. выйти из процедуры;
 - 5.4. положить $\Theta_k = \text{RecursiveSearch}(R_2(T), \delta, i + 1, m, f + x_m^i h_{k,i}, \Theta)$;
 - 5.5. положить $\Theta := \Theta \cup \Theta_k$;
6. вернуть Θ и выйти из процедуры.

Конец процедуры.

ЗАМЕЧАНИЕ. Для уменьшения вычислительной сложности алгоритма можно использовать следующие соображения.

(i) Шаги 1.1 и 1.2 процедуры **RecursiveSearch** не являются обязательными. Они предназначены для уменьшения вычислительной сложности алгоритма в случае, когда T -корень многочлена $Q(T)$, поступившего на вход процедуры, легко найти. Однако если n велико, операция деления многочленов нескольких переменных может занять довольно большое время (см. далее п. 4 и замечание к теореме 13). Поэтому при больших n от шагов 1.1 и 1.2 можно отказаться.

(ii) Многочлен $Q(T)$, поступающий на вход алгоритма **FindRoots** или **RecursiveSearch**, может иметь делители кратности 2 и выше. Если в $\mathbb{D}[x_1, \dots, x_n]$ существует алгоритм вычисления наибольшего общего делителя (например, если $\mathbb{D}$ – евклидова область или область с однозначным разложением на множители, см. [6; с. 162]), то по аналогии с некоторыми алгоритмами факторизации (см., например, [2], [5], [7]) для уменьшения вычислительной сложности к многочлену $Q(T)$ можно применить алгоритм вычисления многочлена $Q^{(f)}(T)$, свободного от квадратов (не имеющего кратных делителей).

Для обоснования корректности алгоритма **FindRoots** необходимо показать, что при любых допустимых значениях входных параметров $Q(T)$, d и n алгоритм заканчивает свою работу за конечное число шагов, а также, что множество $\Theta_{Q,n}(d)$, возвращаемое алгоритмом, совпадает с множеством $\Omega_{Q,n}(d)$. Ясно, что свойства алгоритма целиком зависят от процедуры **RecursiveSearch**, поэтому кратко обсудим ее работу.

Во время выполнения процедуры дважды происходит рекурсивный вызов – на шагах 4 и 5.4. На шаге 4 процедура вызывается для определения всех T -корней степени не выше $\delta - i$ многочлена $M(T)$ из кольца $\mathbb{D}[x_1, \dots, x_{m-1}][T]$, в то время как вызов процедуры на шаге 5.4 – переход к определению следующих компонентов T -корней многочлена $Q(T)$. В связи с этим при анализе работы процедуры удобно разделить эти вызовы, полагая, что на шаге 4 происходит обращение к отдельно выполняемому алгоритму **FindRoots**($M(T), \delta - i, m - 1$). С учетом этого разделения весь

процесс работы алгоритма удобно представить в виде ориентированного леса. Каждая вершина леса соответствует процессу выполнения процедуры `RecursiveSearch` с первого шага по шестой. Переход между вершинами осуществляется в двух случаях: при рекурсивном вызове процедуры и при окончании выполнения процедуры на данной вершине. При рекурсивном вызове процедуры на шаге 5.4 происходит переход от вершины к вершине внутри дерева, а при вызове процедуры на шаге 4 происходит переход от вершины текущего дерева к корню другого дерева. После окончания работы процедуры на какой-либо вершине осуществляется переход между вершинами в обратном порядке (возврат на “предыдущую” вершину). Таким образом, каждое дерево леса описывает последовательность рекурсивных вызовов процедуры на шаге 5.4. Дерево, соответствующее вызову процедуры из алгоритма `FindRoots`, назовем *главным*, а лес, порожденный работой алгоритма, назовем *лесом вызовов*. Отметим основные свойства любого дерева леса вызовов:

- путь в дереве (последовательность вызовов на шаге 5.4) обрывается либо сразу после корня на шаге 1, либо когда множество $\Phi_{m,i}$, вычисленное на шаге 4, пусто, либо когда $i = \delta$;
- глубина каждой вершины (длина пути из корня дерева в эту вершину) равна значению переменной i , корень всегда имеет глубину, равную нулю;
- поскольку для любого дерева $\delta \leq d$, то все пути имеют длину, не превышающую d , а все деревья – высоту, не превышающую d .

В следующей лемме будет показано, что количество вершин в деревьях вызовов всегда конечно, лес состоит из конечного числа деревьев, и, как следствие, алгоритм `FindRoots` заканчивает свою работу за конечное число шагов.

ЛЕММА 6. *Процедура `RecursiveSearch`, вызванная с параметрами $(Q(T), d_0, i_0, n, f, \Theta)$, где $Q(T)$ – некоторый ненулевой многочлен из $\mathbb{D}[x_1, \dots, x_n][T]$, d_0, i_0 – такие неотрицательные числа, что $d_0 \geq i_0$, $n \in \mathbb{N}$, f – некоторый многочлен из $\mathbb{D}[x_1, \dots, x_n]$, Θ – подмножество $\mathbb{D}[x_1, \dots, x_n]$, полностью заканчивает свою работу за конечное число шагов.*

ДОКАЗАТЕЛЬСТВО. Используем индукцию по n . При $n = 1$ и любом допустимом значении i_0 всегда будет выполняться первая ветвь условия на шаге 4. Поэтому лес вызовов будет содержать только главное дерево и, так как $Q(T) \neq 0$ (а следовательно, и $M(T) \neq 0$, $R_2(T) \neq 0$), множество $\Phi_{1,i}$ будет всегда состоять из конечного числа элементов (возможно, нулевого). Следовательно, на каждой вершине на шаге 5.4 при всех значениях $i \in [i_0, d_0]$ будет происходить конечное число рекурсивных вызовов процедуры, причем при каждом вызове значение i увеличивается, а значение $\delta = d_0$ не меняется. Таким образом, главное дерево содержит конечное число путей, имеющих длину, не превышающую $d_0 - i_0$. Необходимо отметить, что из конечности путей в дереве следует тот факт, что процедура добавляет конечное число многочленов к входному множеству Θ . Поэтому при $n = 1$ процедура `RecursiveSearch` при любых допустимых значениях входных параметров всегда заканчивает свою работу за конечное число шагов.

Пусть теперь $n > 1$, и утверждение леммы справедливо для всех значений входного параметра m , меньших n . При $m = n$ и при любом допустимом значении i_0 на шаге 4 всегда будет выполняться вторая ветвь условия, причем согласно индуктивному предположению вызываемая на шаге 4 процедура завершит свою работу

за конечное число шагов, а множество $\Phi_{n,i}$ будет всегда состоять из конечного числа элементов (возможно, нулевого). Повторяя теперь все предыдущие рассуждения, получим, что главное дерево рекурсивных вызовов имеет конечное количество путей и конечную высоту. Следовательно, утверждение леммы верно и при $m = n$.

Покажем теперь, что если алгоритм **FindRoots** вызван с параметрами $(Q(T), d, n)$, то по завершении своей работы он вернет все элементы множества $\Omega_{Q,n}(d)$. Рассмотрим вспомогательные утверждения.

ЛЕММА 7. Пусть $Q(T) (\in \mathbb{D}[x_1][T])$ – некоторый ненулевой многочлен, d – неотрицательное целое число, $\Theta = \text{RecursiveSearch}(Q(T), d, 0, 1, 0, \emptyset)$. Если Θ непусто, то любой многочлен f из Θ имеет полную степень не выше d .

ДОКАЗАТЕЛЬСТВО. По условию леммы $n = m = 1$, поэтому при любом $i \in [0, d]$ на шаге 3 процедуры **RecursiveSearch** многочлен $M(T)$ всегда принадлежит кольцу $\mathbb{D}[T]$, а на шаге 4 всегда выполняется первая ветвь условия, и множество $\Phi_{1,i}$, если оно непусто, состоит из многочленов полной степени 0 (элементов области $\mathbb{D}$). Если Θ непусто, то, как нетрудно видеть, для любого значения $i \in [0, d]$ найдется такая последовательность рекурсивных вызовов процедуры **RecursiveSearch** на шаге 5.4, что множество $\Phi_{1,i}$ непусто, а любой многочлен g из Θ имеет вид $g = \sum_{i=0}^d x_1^i h_{k,i}$, где $h_{k,i} \in \Phi_{1,i}$, и возвращается процедурой при $i = d$. Таким образом, любой многочлен из Θ имеет степень не выше d .

ЛЕММА 8. Для всех $d (\geq 0)$ и $n (> 0)$ любой многочлен f из $\Theta = \text{RecursiveSearch}(Q(T), d, 0, n, 0, \emptyset)$ имеет полную степень не выше d .

ДОКАЗАТЕЛЬСТВО. Воспользуемся методом индукции по количеству переменных n . При $n = 1$ утверждение верно согласно лемме 7. Пусть утверждение верно для всех $d \geq 0$ и всех $m \leq n - 1$. Тогда при $m = n$ для всех $i \in [0, d]$ любой многочлен $h_{k,i}$, вычисляемый на шаге 4 процедуры **RecursiveSearch**, согласно индуктивному предположению имеет полную степень не выше $d - i$. Для доказательства леммы достаточно показать, что при всех $i \in [0, d]$ многочлен $f + x_m^i h_{k,i}$, подаваемый на шаге 5.4 на вход процедуры, имеет полную степень не выше d . Действительно, при $i = 0$ многочлен f – нулевой, а моном $x_m^0 h_{k,0}$ имеет полную степень не выше d . Пусть теперь при некотором $i \in [1, d]$ многочлен f имеет полную степень не выше d . Тогда многочлен $f + x_m^i h_{k,i}$ также будет иметь полную степень не выше d . Следовательно, при $i = d$ многочлен $f + x_m^d h_{k,d}$ будет также иметь полную степень не выше d .

ТЕОРЕМА 9. Алгоритм **FindRoots**, запущенный с начальными параметрами $(Q(T), d, n)$, возвращает все T -корни полной степени не выше d многочлена $Q(T)$.

ДОКАЗАТЕЛЬСТВО. Пусть $\Theta_{Q,n}(d) (\subset \mathbb{D}[x_1, \dots, x_n])$ – множество всех многочленов, возвращаемых алгоритмом **FindRoots** $(Q(T), d, n)$. Покажем, что $\Theta_{Q,n}(d) = \Omega_{Q,n}(d)$. Предположим, что $\Theta_{Q,n}(d)$ непусто. Пусть $f \in \Theta_{Q,n}(d)$. Согласно лемме 8 полная степень f не превышает d . Многочлен f построен таким образом, что при $i = d$ многочлен T делит многочлен $R_2(T) = Q^{(x)}(x_n T + h_{k,d})$ (условие на шаге 5.3.1). Кроме того, легко проверить, что $h_{k,d} = \psi_n(f_d)$, где f_d – многочлен, рекурсивно вычисленный по формуле (2.2). Поэтому согласно теореме 4 f является T -корнем

$Q(T)$ и $\Theta_{Q,n}(d) \subset \Omega_{Q,n}(d)$. Включение $\Omega_{Q,n}(d) \subset \Theta_{Q,n}(d)$ вытекает из теоремы 5 и того факта, что процедура **RecursiveSearch** при каждом значении i осуществляет полный перебор делителей вида $(T - \psi_n(f))$ многочлена $\psi_n^T(Q_i^{(x)}(T))$.

4. Анализ сложности алгоритма FindRoots. Рассмотрим некоторый многочлен $Q(T)$ из $\mathbb{D}[x_1, \dots, x_n][T]$ T -степени b . Пусть алгоритм **FindRoots** вызван с параметрами $(Q(T), d, n)$. Согласно лемме 6 алгоритм закончит свою работу за конечное число шагов. Для оценки временной сложности алгоритма вычислим верхние границы количества рекурсивных вызовов процедуры **RecursiveSearch** и общего количества вершин деревьев леса, а также максимальную из полных степеней коэффициентов многочленов, возникающих во время вычислений.

4.1. Рассмотрим несколько вспомогательных лемм.

ЛЕММА 10. Пусть $n > 1$, $Q(T) \in \mathbb{D}[x_1, \dots, x_n][T]$ – такой ненулевой многочлен T -степени b , что x_n не делит $Q(T)$, $h \in \mathbb{D}[x_1, \dots, x_{n-1}]$ – T -корень полной степени не выше d многочлена $\psi_n^T(Q(T))$ кратности $\gamma (> 0)$. Положим

$$\begin{aligned} R_1(T) &= Q(T+h) = \sum_{k=0}^b s_k T^k, \\ R_2(T) &= R_1(x_n T) = \sum_{k=0}^b s_k x_n^k T^k, \\ R_2^{(x)}(T) &= \frac{R_2(T)}{x_n^r}, \end{aligned}$$

где $s_k \in \mathbb{D}[x_1, \dots, x_n]$, r – такое наибольшее неотрицательное целое число, что x_n^r делит $R_2(T)$, а x_n^{r+1} не делит $R_2(T)$. Тогда T -степень многочлена $M(T) = \psi_n^T(R_2^{(x)}(T))$ не превосходит γ .

ДОКАЗАТЕЛЬСТВО. Так как h является T -корнем многочлена $\psi_n^T(Q(T))$ кратности γ , то

$$\psi_n^T(Q(T)) = (T-h)^\gamma U(T)$$

для некоторого $U(T) \in \mathbb{D}[x_1, \dots, x_{n-1}][T]$, причем $(T-h)$ не делит $U(T)$. Тогда

$$\psi_n^T(R_1(T)) = \psi_n^T(Q(T+h)) = (T)^\gamma U(T+h),$$

и T не делит $U(T+h)$. Так как T^γ делит $\psi_n^T(R_1(T))$, и $T^{\gamma+1}$ не делит $\psi_n^T(R_1(T))$, то с учетом (2.1) $\psi_n(s_k) = 0$ для всех $k \in [0, \gamma-1]$, но $\psi_n(s_\gamma) \neq 0$. Это равнозначно тому, что x_n делит многочлены s_k для всех $k \in [0, \gamma-1]$ и не делит s_γ . Следовательно, x_n делит $R_2(T)$, но $x_n^{\gamma+1}$ не делит $R_2(T)$. Таким образом, число r из условия леммы принадлежит отрезку $[1, \gamma]$. Запишем многочлен $R_2^{(x)}(T)$ как

$$R_2^{(x)}(T) = \frac{R_2(T)}{x_n^r} = \sum_{k=0}^r \frac{[s_k x_n^k T^k]}{x_n^r} + \sum_{k=r+1}^b s_k x_n^{k-r} T^k.$$

Легко видеть, что T -степень $M(T) = \psi_n^T(R_2^{(x)}(T))$ не превышает $r \in [1, \gamma]$.

СЛЕДСТВИЕ 1. Если на вход процедуры **RecursiveSearch** поступает такой многочлен $Q(T)$, что все T -корни $M(T) = \psi_n^T(Q^{(x)}(T))$ имеют кратность 1, то все многочлены $R_2(T)$, получаемые на шаге 5.2 и передаваемые как параметр при рекурсивном вызове на шаге 5.4, будут обладать тем свойством, что T -степень $\psi_n^T(R_2^{(x)}(T))$ не превысит 1 и, следовательно, их T -корни просто найти.

СЛЕДСТВИЕ 2. T -степень любого многочлена, возникающего в процессе выполнения процедуры **RecursiveSearch**, не превышает b .

ЛЕММА 11. Пусть $Q(T) (\in \mathbb{D}[x_1, \dots, x_n][T])$ – ненулевой многочлен T -степени b . Рассмотрим лес вызовов алгоритма **FindRoots**, запущенного с параметрами $(Q(T), d, n)$. Верны следующие утверждения:

- (i) ширина любого дерева леса не превышает b ;
- (ii) максимальное количество вершин любого дерева не превышает $bd + 1$;
- (iii) общее количество деревьев леса не превышает $O(b^{n-1}d^{n-1})$;
- (iv) суммарное количество вершин деревьев леса не превышает $O(bd^n)$.

ДОКАЗАТЕЛЬСТВО. Рассмотрим главное дерево вызовов. Для каждого $i \in [0, d]$ через Π_i обозначим множество вершин глубины i , а через A_i – мощность Π_i . Вершины из множества Π_i занумеруем числами отрезка $[0, A_i - 1]$. Для каждого $j \in [0, A_i - 1]$ через $\gamma_{i,j}$ обозначим T -степень многочлена $M(T)$, вычисляемого на j -й вершине глубины i . Если Π_i пусто, положим $\gamma_{i,j} = 0$. Пусть

$$\omega_i = \sum_{j=0}^{A_i-1} \gamma_{i,j}.$$

Отметим, что $A_0 = 1$, и существует единственный многочлен $M(T) = \psi_n^T(Q^{(x)}(T))$ T -степени не выше b , поэтому $\omega_0 \leq b$. Согласно лемме 10 $\omega_i \leq \omega_{i-1}$ для каждого $i \in [1, d]$. Следовательно, $\omega_i \leq b$, $A_i \leq b$, ширина главного дерева не превышает b , а количество его вершин с учетом корня не превышает $1 + \sum_{i=0}^d \omega_i \leq 1 + bd$.

Далее, любое дерево леса вызовов можно рассматривать как главное дерево вызовов алгоритма **FindRoots**, запущенного с параметрами $(M(T), \delta - i, m - 1)$ (см. шаг 4). Нетрудно видеть, что во время работы алгоритма всегда выполняются неравенства $\delta - i \leq d$ и $\deg_T(M(T)) \leq b$, поэтому оценки (i) и (ii) верны для любого дерева леса. Отметим, что в некоторых случаях оценка (ii) может оказаться довольно грубой.

Для доказательства утверждений (iii) и (iv) индуктивно оценим величины $T_n(b, d)$ – максимальное количество деревьев леса вызовов, и $D_n(b, d)$ – суммарное количество вершин деревьев леса вызовов. При $n = 1$ лес состоит только из главного дерева, поэтому $T_1(b, d) = 1$ и $D_1(b, d) \leq bd + 1 \leq bd + O(b)$. Предположим, что утверждения (iii) и (iv) верны для всех $n \in [1, m]$. Рассмотрим случай $n = m + 1$. Главное дерево вызовов по-прежнему содержит не более $bd + 1$ вершин, на которых может произойти вызов алгоритма **FindRoots** для многочлена $M(T) \in \mathbb{D}[x_1, \dots, x_m][T]$. Поэтому

$$T_{m+1}(b, d) \leq T_m(b, d) + b \sum_{i=1}^d T_m(b, d - i) + 1 = \sum_{i=0}^m \binom{m+1}{i+1} \binom{d}{i} b^i = O(b^m d^m).$$

Кроме вершин главного дерева лес вызовов содержит не более

$$\sum_{i=0}^d \sum_{j=0}^{A_i-1} D_m(\gamma_{i,j}, d-i)$$

вершин остальных деревьев. Так как функция D_m согласно индуктивному предположению линейна относительно первого аргумента, то

$$\begin{aligned} D_{m+1}(b, d) &= bd + 1 + \sum_{i=0}^d D_m(\omega_i, d-i) \leq bd + 1 + \sum_{i=0}^d D_m(b, d-i) \\ &\leq b \sum_{i=1}^m \binom{d-1+i}{i} + \sum_{i=0}^{m-1} (m-i) \binom{d-1+i}{i} = O(bd^{m+1}). \end{aligned}$$

ЗАМЕЧАНИЕ. Если суммарное количество вершин деревьев леса оценить как количество деревьев, умноженное на максимальное количество вершин в дереве, то получим более грубую оценку $O(b^n d^n)$.

ЛЕММА 12. Пусть $Q(T) = \sum_{k=0}^b q_k T^k \in \mathbb{D}[x_1, \dots, x_n][T]$ – такой ненулевой многочлен T -степени b , что полная степень любого коэффициента q_k не превосходит s . Тогда полная степень коэффициентов многочленов, вычисляемых в ходе работы алгоритма **FindRoots**, запущенного с параметрами $(Q(T), d, n)$, не превышает $s + b \binom{d+2}{2}$.

ДОКАЗАТЕЛЬСТВО. Пусть $S_n(s, b, d)$ – максимальная полная степень коэффициентов многочленов, вычисляемых в ходе работы алгоритма **FindRoots**, запущенного с параметрами $(Q(T), d, n)$. Рассмотрим вершину глубины $i \in [0, d]$ главного дерева вызовов и работу процедуры **RecursiveSearch** на этой вершине. Пусть на вход процедуры поступает многочлен $Q_i(T) = \sum_{j=0}^b q_j^{(i)} T^j$, причем $\deg q_j^{(i)} \leq s_i$. Оценим сверху величину s_i . Ясно, что $s_0 = s$. Пусть

$$\begin{aligned} R_1(T) &= Q_i^{(x)}(T + h_{k,i}) = \sum_{j=0}^b q_j^{(i)} \frac{(T + h_{k,i})^j}{x_n^r} = \sum_{j=0}^b r_{k,j}^{(i)} T^j, \\ R_2(T) &= R_1(x_n T) = \sum_{j=0}^b r_{k,j}^{(i)} x_n^j T^j \end{aligned}$$

– многочлены, вычисляемые на шагах 5.1 и 5.2. Тогда

$$s_{i+1} = \max\{\deg(x_n^j r_{k,j}^{(i)})\} \leq s_i + b \deg(h_{k,i}) \leq s_i + b(d-i).$$

Следовательно,

$$s_{i+1} \leq s + b \sum_{l=0}^i (d-l+1) = s + b(i+1) \left(d+1 - \frac{i}{2} \right) \leq s + b \binom{d+2}{2}.$$

С помощью индукции по n оценим теперь величину $S_n(s, b, d)$. При $n = 1$ лес вызовов состоит только из главного дерева. Так как $S_1(s, b, d) = \max_i \{s_i\}$, то $S_1(s, b, d) \leq s + b \binom{d+2}{2}$. Пусть утверждение леммы верно для всех $n \in [1, m]$. Рассмот-

рим случай $n = m + 1$. Максимальная полная степень коэффициентов многочленов, вычисляемых на вершинах главного дерева, не превышает $s + b \binom{d+2}{2}$. На каждой вершине глубины i главного дерева вызовов на шаге 4 происходит обращение к алгоритму **FindRoots** с параметрами $(M(T), d - i, n - 1)$. Согласно индуктивному предположению во время работы этого алгоритма максимальная полная степень коэффициентов не превысит $S_{n-1}(s_i, b, d - i)$. Но

$$\begin{aligned} S_{n-1}(s_i, b, d - i) &\leq s_i + \frac{b(d - i + 1)(d - i + 2)}{2} \\ &\leq s + b \sum_{l=0}^{i-1} (d - l + 1) + \frac{b(d - i + 1)(d - i + 2)}{2} \\ &= s + \frac{b(d + 1)(d + 2)}{2} = s + b \binom{d + 2}{2}, \end{aligned}$$

что доказывает утверждение леммы.

4.2. Выберем модель вычислений, способ записи многочленов нескольких переменных, а также оценим сложность основных операций с многочленами, встречающихся в алгоритме. Поскольку тип области $\mathbb{D}$ и представление ее элементов нам априори неизвестны, мы не можем оценить битовую сложность арифметических операций в $\mathbb{D}$. Поэтому будем полагать, что базовые операции в $\mathbb{D}$ (сложение, вычитание, умножение, деление, присваивание и сравнение элементов) имеют временную сложность $O(1)$, и для хранения одного элемента области необходимо $O(1)$ ячеек памяти. Таким образом, оцениваемая нами сложность является арифметической. Отметим, что если $\mathbb{D}$ и ее машинное представление известны, то по арифметической сложности алгоритма сравнительно нетрудно оценить его битовую сложность.

Для произвольного натурального m многочлены кольца $\mathbb{D}[x_1, \dots, x_m]$ будем представлять в рекурсивной форме. А именно, в случае $m = 1$ все многочлены будем записывать как обычно, а если $m > 1$, то многочлен f полной степени s запишем как $f = \sum_{i=0}^s f_i x_m^i$, где все $f_i \in \mathbb{D}[x_1, \dots, x_{m-1}]$ и, в свою очередь, имеют рекурсивную форму записи. Например, многочлен

$$x_1 x_2 + x_1 x_3^2 + x_3^2 + x_1^2 x_3 + x_1 x_2 x_3 + 2x_2 x_3 + 4x_2$$

в рекурсивном представлении имеет вид

$$x_3^2(x_1 + 1) + x_3(x_2(x_1 + 2) + x_1^2) + x_2(x_1 + 4).$$

Отметим, что такая запись многочленов весьма удобна для вычисления $\psi_m(f)$.

Пусть f, g – многочлены из $\mathbb{D}[x_1, \dots, x_m]$ полных степеней s и t соответственно, $s \geq t$. Сравнительно легко написать рекурсивные алгоритмы сложения и умножения, сохраняющие вид записи многочленов. Например, для сложения можно использовать обычный алгоритм покоэффициентного сложения с той разницей, что если коэффициенты по-прежнему являются многочленами (содержат хотя бы одну переменную в ненулевой степени), то для их сложения рекурсивно применяется этот же алгоритм. Его арифметическая сложность оценивается как $O(t^m)$ сложений элементов $\mathbb{D}$. Для умножения многочленов f и g можно использовать рекурсивный алгоритм умножения “в столбик”, имеющий арифметическую сложность $O(s^m t^m)$

операций сложения и умножения элементов $\mathbb{D}$. Однако если $s \approx t$, то для экономии количества операций целесообразнее использовать рекурсивную версию алгоритма Карацубы или, в зависимости от свойств $\mathbb{D}$, алгоритма Тоома–Кука или Шенхаге–Штрассена (см., например, [8; гл. 4.3.3]). Известно, что арифметическая сложность алгоритма Карацубы умножения многочленов f, g степени s из $R[x]$, где R – коммутативное кольцо, равна $O(s^{\log_2 3})$ операций R . Если применить этот алгоритм к многочленам из кольца $R[x_m]$, где $R = \mathbb{D}[x_1, \dots, x_{m-1}]$, причем в R для умножения многочленов тоже использовать алгоритм Карацубы, то общую сложность получившегося рекурсивного алгоритма можно оценить как $O(s^{m \log_2 3})$ операций $\mathbb{D}$. Отметим, что сложность алгоритмов умножения многочленов, использующих быстрое преобразование Фурье, оценивается как $O(ms^m \log_2 s)$ (см. [9; с. 265]).

Кроме сложения и умножения в алгоритме **FindRoots** также встречаются операции деления многочленов (шаг 1.2) и разложения по степеням $(T + h)$ (шаг 5.1). Для деления многочленов $f, g \in \mathbb{D}[x_1, \dots, x_m]$ полных степеней s и t соответственно, представленных в рекурсивной форме, удобно использовать алгоритм пробного деления (см., например, [9; с. 266]). Он представляет собой рекурсивную версию алгоритма деления “уголком” за одним исключением: если в какой-то момент работы рекурсивного алгоритма произошло деление многочленов с ненулевым остатком, то выдается сообщение о невозможности деления исходных многочленов нацело, и происходит завершение работы алгоритма. Сложность пробного алгоритма деления можно грубо оценить сверху как $O(s^m t^m)$ операций $\mathbb{D}$. Для разложения многочлена $Q(T) \in R[T]$ T -степени b над кольцом R по степеням $(T + h)$, где $h \in R$, можно воспользоваться схемой Горнера, требующей $b(b + 1)/2 = O(b^2)$ умножений и столько же сложений в R . Если $R = \mathbb{D}[x_1, \dots, x_m]$, полная степень коэффициентов $Q(T)$ не превышает s , а $\deg(h) = d$, то арифметическую сложность разложения $Q(T)$ с помощью алгоритма Горнера можно оценить как $O(b^2 d^m (s + bd)^m)$, если использовать алгоритм умножения многочленов “в столбик”, или $O(b^2 (s + bd)^{m \log_2 3})$, если использовать метод Карацубы. Отметим, что в работе Шоу и Трауба [10] предложен алгоритм, вычисляющий разложение $Q(T)$ по степеням $(T + h)$ за $2b - 1$ умножений, $b - 1$ делений и $b(b + 1)/2$ сложений в R . Тем не менее, несмотря на кажущееся преимущество перед схемой Горнера, алгоритм Шоу–Трауба из-за промежуточных вычислений уже в случае $m > 1$ требует значительно большего количества операций $\mathbb{D}$ и для алгоритма **FindRoots** не подходит.

Арифметическую сложность алгоритма **FindRootsOne** нахождения корней многочлена $Q(T) \in \mathbb{D}[T]$ степени s будем полагать равной $F(s)$. В случае $\mathbb{D} = \mathbb{F}_q$ в работе [11] приведен алгоритм **FindRootsOne**, имеющий оценку сложности $O(s^{1.815} \log q)$.

ТЕОРЕМА 13. Пусть $Q(T) = \sum_{k=0}^b q_k T^k \in \mathbb{D}[x_1, \dots, x_n][T]$ – такой ненулевой многочлен T -степени b , что полная степень его коэффициентов q_k не превосходит s . Тогда арифметическая сложность алгоритма **FindRoots**, запущенного с параметрами $(Q(T), d, n)$, в наихудшем случае равна либо

$$O(bd^n(F(b) + (s + bd^2)^{2n} + b^2 d^{n-1}(s + bd^2)^n)),$$

если используется метод умножения многочленов “в столбик”, либо

$$O(bd^n(F(b) + (s + bd^2)^{2n} + b^2(s + bd^2)^{(n-1) \log_2 3 + 1})),$$

если используется метод Карацубы.

ДОКАЗАТЕЛЬСТВО. Оценим арифметическую сложность одной вершины леса вызовов. Для выполнения шагов 2, 3 и 5.2 процедуры **RecursiveSearch** необходимо изменить коэффициенты q_k многочлена $Q(T)$. Согласно лемме 12 полная степень всех q_k в течение работы алгоритма не превышает $s + b(d+1)(d+2)/2$. Поскольку T -степень $Q(T)$ не превышает b , с учетом рекурсивного представления многочлена $Q(T)$ сложность шагов 2, 3, 5.2 можно оценить как $O(b(s + bd^2))$.

Шаг 5.1 удобно выполнять с помощью схемы Горнера. Так как полная степень многочленов $h_{k,i}$ ($\in \mathbb{D}[x_1, \dots, x_{n-1}]$) не превышает d , а коэффициенты многочлена $Q^{(x)}(T)$ принадлежат $\mathbb{D}[x_1, \dots, x_n]$, сложность шага 5.1 можно оценить либо как $O(b^2 d^{n-1}(s + bd^2)^n)$ операций $\mathbb{D}$, если использовать алгоритм умножения “в столбик”, либо как $O(b^2(s + bd^2)(s + bd^2)^{(n-1)\log_2 3})$, если использовать метод Карацубы.

На шаге 1.1 осуществляется пробное деление многочленов из $\mathbb{D}[x_1, \dots, x_n]$ полных степеней не выше $s + b(d+1)(d+2)/2$. Следовательно, сложность этого шага можно оценить (довольно грубо) как $O((s + bd^2)^{2n})$. Оценку полной степени частного можно произвести на шаге 1.1, поэтому сложность шага 1.2 не превышает сложности шага 1.1.

Так как лес вызовов алгоритма содержит $O(bd^n)$ вершин (лемма 11) и алгоритм **FindRootsOne** вызывается $O(bd^n)$ раз (пропорционально количеству коэффициентов всех T -корней), результирующую сложность можно оценить либо как

$$O(bd^n(F(b) + (s + bd^2)^{2n} + b^2 d^{n-1}(s + bd^2)^n)),$$

либо как

$$O(bd^n(F(b) + (s + bd^2)^{2n} + b^2(s + bd^2)^{(n-1)\log_2 3+1})).$$

СЛЕДСТВИЕ 3. В случае $\mathbb{D} = \mathbb{F}_q$ и при использовании метода Карацубы арифметическая сложность алгоритма равна

$$O(b^{2.815} d^n \log q + bd^n(s + bd^2)^{2n} + b^3 d^n(s + bd^2)^{(n-1)\log_2 3+1}).$$

ЗАМЕЧАНИЕ. Поскольку для любого многочлена $Q(T)$ всегда выполняется неравенство $bd \leq s$, при достаточно больших n оценку сложности алгоритма можно несколько упростить. Так, слагаемое $bd^n(s + bd^2)^{2n}$ можно заменить на $(s^2 d^3)^n$, а слагаемые $b^3 d^{2n-1}(s + bd^2)^n$ и $b^3 d^n(s + bd^2)^{(n-1)\log_2 3+1}$ на $(sd^3)^n$ и $d(sd)^{1.58n}$ соответственно. Далее, если b фиксировано и алгоритм **FindRootsOne** имеет полиномиальную сложность, главным членом в выражении, оценивающим трудоемкость алгоритма, является слагаемое $(s^2 d^3)^n$, возникающее из-за пробного деления на шаге 1.1. Отметим, что для уменьшения сложности при больших n от шага 1 можно отказаться без ущерба корректности алгоритма. Тогда асимптотическая сложность алгоритма **FindRoots** в зависимости от применяемого способа умножения многочленов может быть грубо оценена либо как $O((sd^3)^n)$, либо как $O(d(sd)^{1.58n})$. Нетрудно видеть, что при фиксированном значении n она является полиномиальной.

5. Заключение. В работе построен алгоритм поиска всех T -корней степени не выше d произвольного многочлена $Q(T) \in \mathbb{D}[x_1, \dots, x_n][T]$ T -степени b и доказана его корректность. Арифметическая сложность алгоритма может быть грубо оценена в худшем случае как

$$O(bd(F(b) + d^{n-1}(s + bd^2)^{2n} + b^2(s + bd^2)^{(n-1)\log_2 3+1}))$$

операций области $\mathbb{D}$, где $F(b)$ – сложность алгоритма вычисления корней многочлена $f \in \mathbb{D}[T]$ степени b . Если $\mathbb{D} = \mathbb{F}_q$, арифметическая сложность алгоритма может быть оценена как

$$O(b^{2.815} d \log q + bd^n(s + bd^2)^{2n} + b^3 d(s + bd^2)^{(n-1) \log_2 3 + 1}).$$

Автор выражает благодарность В. М. Деундяку за полезные обсуждения и постоянный интерес к работе, а также рецензенту за ряд весьма полезных замечаний.

СПИСОК ЦИТИРОВАННОЙ ЛИТЕРАТУРЫ

- [1] R. M. Roth, G. Ruckenstein, “Efficient decoding of Reed–Solomon codes beyond half the minimum distance”, *IEEE Trans. Inform. Theory*, **46**:1 (2000), 246–257.
- [2] H. Cohen, *A Course in Computational Algebraic Number Theory*, Grad. Texts in Math., **138**, Springer-Verlag, Berlin, 1996.
- [3] V. Guruswami, *List Decoding of Error-Correcting Codes*, Winning thesis of the 2002 ACM Doctoral Dissertation Competition, Lecture Notes in Comput. Sci., **3282**, Springer-Verlag, Berlin, 2004.
- [4] А. Э. Маевский, “Алгоритм списочного декодирования одного класса алгебро-геометрических кодов на проективных кривых”, *Интегральные и дифференциальные уравнения*, ДГТУ, Ростов-на-Дону, 2007, 1–1.
- [5] Э. Калтофен, “Разложение полиномов на множители”, *Компьютерная алгебра: символные и алгебраические вычисления*, Мир, М., 1986, 127–150.
- [6] T. W. Hungerford, *Algebra*, Grad. Texts in Math., **73**, Springer-Verlag, New York–Berlin, 1980.
- [7] L. Bernardin, “On square-free factorization of multivariate polynomials over a finite fields”, *Theoret. Comput. Sci.*, **187**:1–2 (1997), 105–116.
- [8] Д. Кнут, *Искусство программирования для ЭВМ*, Т. 2: Получисленные алгоритмы, Мир, М., 1977.
- [9] Дж. Э. Коллинз, М. Миньотт, Ф. Уинклер, “Арифметика в основных алгебраических областях”, *Компьютерная алгебра: символные и алгебраические вычисления*, Мир, М., 1986, 237–276.
- [10] M. Shaw, J. F. Traub, “On the number of multiplications for the evaluation of a polynomial and some of its derivatives”, *J. Assoc. Comput. Mach.*, **21**:1 (1974), 161–167.
- [11] E. Kaltofen, V. Shoup, “Subquadratic-time factoring of polynomials over finite fields”, *Math. Comp.*, **67**:223 (1998), 1179–1197.

А. Э. Маевский

Южный федеральный университет, г. Ростов-на-Дону
E-mail: tim_org@mail.ru

Поступило

20.08.2007