Алгоритм Берлекэмпа

Гудошников Роман

ФКТИ, группа 6372

Назначение алгоритма

- **Алгоритм Берлекэмпа** предназначен для факторизации унитарного многочлена, заданного над полем Галуа, где:
- **Унитарный многочлен** – м-н, коэффициент при старшей степени которого равен 1
- **Факторизация многочлена** - разложение м-на на неприводимые множители
- **Поле Галуа** (или **конечное поле**) – поле, состоящее из конечного числа элементов, наиболее известным примером которого являются классы вычетов по модулю числа. Теоретически, поле может состоять из любых элементов, но так как брать в качестве коэффициентов для многочлена треугольники или квадраты не очень практично, все сводится к целым числам. В дальнейшем обозначается как GF(q).
- Данный алгоритм работает над полями, **порядок** (модуль числа класса вычетов или просто кол-во элементов) которых является **простым числом** q или его степенью $p = q^m$, для таких полей $(a + b)^q = a^q + b^q$

Краткая справка

Алгоритм был разработан Элвином Берлекэмпом в 1967 году. Это был первый алгоритм факторизации многочленов над конечным полем, поэтому, несмотря на свою высокую сложность, он был основным способом решения проблемы факторизации вплоть до появления алгоритма Кантора-Цассенхауза в 1981 г. Алгоритм используется для решения многих задач в алгебре и теории чисел, например, для разложения простого рационального числа в поле алгебраических чисел. Для этого алгоритма также существуют различные усовершенствования, увеличивающие его эффективность на полях большого порядка и для многочленов высокой степени.

Шаги алгоритма

- Алгоритм Берлекэмпа можно условно разбить на 5 шагов:
- **1) Находим НОД(f(x), f'(x)),** которое будет иметь следующее свойство:
  Если $f(x) = p_1^{e1}(x)p_2^{e2}(x)\dots p_n^{en}(x)$,
  то $НОД(f(x), f'(x)) = p_1^{v1}(x)p_2^{v2}(x)\dots p_n^{vn}(x)$ (далее просто НОД),
  где $v_i = e_i-1$ при e_i не кратном q, $v_i = e_i$ при кратном
- Это позволяет «отделить» от многочлена все неприводимые множители степенью больше 1 и позволяет продолжить работу с многочленом меньшей степени. Таким образом, если НОД = 1, то многочлен свободен от квадратов, т.е. степень всех его неприводимых сомножителей равна 1, и дальнейший алгоритм работает с f(x). В случае, если НОД != 1, алгоритм продолжается для f(x)/НОД и начинается с первого шага для $f_1(x)$ = НОД.
- Соответственно, если НОД($f_1(x)$, $f_1'(x)$) != 1, алгоритм продолжается для $f_1(x)$/НОД и начинается с **1)** для $f_2(x) = f_1'(x)$, пока НОД не станет равен 1.
- В конце результаты действия для каждого из параллельных алгоритмов перемножаются, давая исходную степень для каждого множителя.

- Случай, когда $f'(x) = 0$ означает, что $f(x)$ можно представить как $g_0(x^p)$ или же как $g^p(x)$, то есть задачу можно свести к разложению многочлена меньшей степени.
- Коэффициенты многочлена $g(x)$ можно найти по правилу:
- $f(x) = a_n x^{en*p} + a_{n-1} x^{n-1*p} + \ldots + a_1$
- $g(x) = a_n x^{en} + a_{n-1} x^{n-1} + \ldots + a_1$
- Например:
- $f(x) = x^6 + 2x^3 + 2$, $q = 3$
- $g(x) = x^2 + 2x + 2$
- $f(x) = g^3(x)$
- Рассмотрим все шаги алгоритма на примере многочлена $f(x) = x^5 + x^3 + x^2 + x$, заданного над полем $GF(2)$. В данном случае $f'(x) = x^4 + x^2 + 1$, а их НОД = 1, т.е. многочлен свободен от квадратов, алгоритм продолжается для $f(x)$.

- Целью следующих шагов является нахождения **разлагающих многочленов,** позволяющих разложить исходный мн-н на не менее чем два множителя.
- **f-разлагающим** называется такой многочлен h(x), что $1 <= deg(h(x)) < deg(f(x))$, и $h(x)^q \equiv h(x) \pmod{f(x)}$ (q – это по-прежнему порядок поля)
- **2) Построим матрицу B следующим способом:**
- Для каждого ряда матрицы номера $i = 0…n-1$ (а n – это все еще deg(f(x)) найдем многочлен, равный $x^{i*q} \bmod f(x)$ и запишем его коэффициенты в обратном порядке, как строку этой матрицы. Продолжая пример:
- $f(x) = x^5+x^3+x^2+x$, $q = 2$, $n = 5$ => матрица будет иметь размерность 5x5.
- $x^{0*2} \equiv 1 \bmod f(x) = 1 \quad \Leftrightarrow \quad (1, 0, 0, 0, 0)$
- $x^{1*2} \equiv x^2 \bmod f(x) = x^2 \quad \Leftrightarrow \quad (0, 0, 1, 0, 0)$
- $x^{2*2} \equiv x^4 \bmod f(x) = x^4 \quad \Leftrightarrow \quad (0, 0, 0, 0, 1)$
- $x^{3*2} \equiv x^6 \bmod f(x) = x^4+x^3+x^2 \quad \Leftrightarrow \quad (0, 0, 1, 1, 1)$
- $x^{4*2} \equiv x^8 \bmod f(x) = x \quad \Leftrightarrow \quad (0, 1, 0, 0, 0)$

$1 \quad x \quad x^2 \quad x^3 \quad x^4$

Получившаяся матрица B

- Эта матрица важна, потому что многочлен $h(x) = a_0+a_1x+…+a_{n-1}x^{n-1}$ будет являться f-разлагающим лишь при условии $(a_0,a_1,…,a_{n-1})\, B = (a_0,a_1,…,a_{n-1})$, иначе говоря, если вектор коэффициентов этого многочлена будет являться собственным вектором матрицы B при собственном числе 1.

- **3) Проведем с матрицей B необходимые преобразования**, а именно:
- Вычтем из нее единичную матрицу E, а затем транспонируем. ($B_1 = (B\text{-}E)^{\text{т}}$)

- $$B - E = \begin{bmatrix} 1&0&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&0&1 \\ 0&0&1&1&1 \\ 0&1&0&0&0 \end{bmatrix} - \begin{bmatrix} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0 \\ 0&1&1&0&0 \\ 0&0&1&0&1 \\ 0&0&1&0&1 \\ 0&1&0&0&1 \end{bmatrix}$$

- $$(B\text{-}E)^{\text{т}} = \begin{bmatrix} 0&0&0&0&0 \\ 0&1&0&0&1 \\ 0&1&1&1&0 \\ 0&0&0&0&0 \\ 0&0&1&1&1 \end{bmatrix}$$

Получившаяся матрица B_1

- Обычно эти действия не выделяют, но так как они имеют мало общего со следующим шагом, а также ради лучшей детализации, я решил отделить их.

- **4) Найдем базис пространства решений системы линейных уравнений**

$$B_1 \begin{pmatrix} x_0 \\ \dots \\ x_{n-1} \end{pmatrix} = 0,$$

- Обычное действие из курса линейной алгебры, приведем матрицу к более удобному виду, обозначим ряды матрицы как вектора, а их столбцы как соответствующие координаты и выразим зависимые переменные.
- $$\begin{bmatrix} 0&0&0&0&0 \\ 0&1&0&0&1 \\ 0&1&1&1&0 \\ 0&0&0&0&0 \\ 0&0&1&1&1 \end{bmatrix} = \begin{array}{ccccc} \multicolumn{5}{c}{\begin{bmatrix} 0&1&0&0&1 \\ 0&0&1&1&1 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{bmatrix}} \\ x_1 & x_2 & x_3 & x_4 & x_5 \end{array} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x_1 \\ -x_5 \\ -x_4-x_5 \\ x_4 \\ x_5 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + x_5 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$
- Полученный базис: $e_1 = (1, 0, 0, 0, 0)$ – данный вектор всегда будет первым в найденном базисе, если он является единственным, то многочлен неразложим, и алгоритм заканчивается.
  $e_2 = (0, 0, 1, 1, 0)$, $e_3 = (0, 1, 1, 0, 1)$
- Векторы, соответствующие разлагающим многочленам:
- $h_1(x) = x^3 + x^2$, $h_2(x) = x^4 + x^2 + x$, $r = 3$

- Перед следующим шагам стоит упомянуть главное свойство разлагающего многочлена, которое и позволяет с его помощью разложить исходный мн-н на несколько множителей:

$$f(x) = \prod_{c \in GF(q)} \text{НОД}(f(x),\, h(x) - c),$$

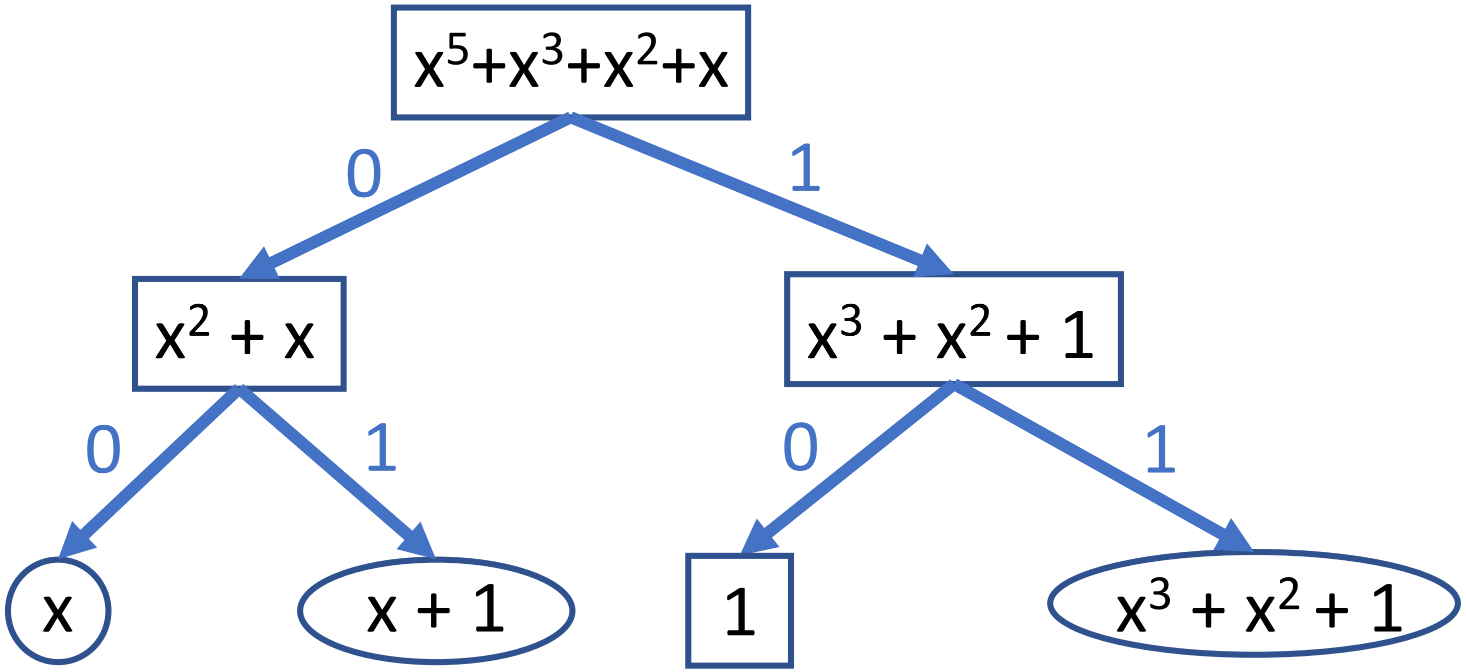
- То есть, многочлен f равен произведению всех НОД этого многочлена и разницы его разлагающего многочлена с каждым из элементов поля.
- Это означает, что для каждого из разлагающих многочленов придется провести перебор всех элементов поля, что существенно сказывается на эффективности алгоритма.
- Сложность алгоритма составляет $O(n^3 + qkn)$, то есть он является эффективным лишь при сравнительно небольшом порядке поля.

- **5) По найденным разлагающим многочленам разложим многочлен f(x) на r (количество векторов в базисе) множителей:**
- Для этого найдем $t_c(x)$ = НОД(f(x), $h_1(x)$ - c) перебором c, ($\forall$ c $\in$ GF(q))
- Если мы получили r множителей $t_c(x)$!= 1, то у нас базисе было всего 2 вектора, и разложение найдено.
  В противном случае продолжаем искать t_{ic} = НОД(u(x), $h_2(x)$ - c), где u(x) = $t_c(x)$, ($\forall$ $t_c(x)$!= 1), t_{iic}= НОД($t_{ii}(x)$, $h_3(x)$ - c) и так далее, для каждого c и h_i, пока количество найденных t(x)!=1 не станет равным r. Если НОД(u(x), $h_i(x)$ - c) = u(x), то множитель u(x) является неприводимым, и можно переходить к следующему.
  При достижении r множителей алгоритм заканчивается, а найденное разложение является полным.

- Продолжая пример:
- $f(x) = x^5+x^3+x^2+x$, $h_1(x) = x^3 + x^2$, $h_2(x) = x^4 + x^2 + x$, $r = 3$
- НОД($f(x)$, $h_1(x) - 0$) = $x^2 + x$, НОД($f(x)$, $h_1(x) - 1$) = $x^3 + x^2 + 1$, найдено 2 множителя при $r = 3$, поэтому:
- НОД($x^2 + x$, $h_2(x) – 0$) = x, НОД($x^2 + x$, $h_2(x) – 1$) = $x + 1$, найдено 3 множителя, что равно r, поэтому продолжать алгоритм нет смысла, но чтобы убедиться:
- НОД($x^3 + x^2 + 1$, $h_2(x) – 0$) = 1, НОД($x^3 + x^2 + 1$, $h_2(x) – 0$) = $x^3 +x^2+ 1$, $\Leftrightarrow$ $x^3 + x^2 + 1$ – действительно неразложим.
- Итак, $f(x) = x^5 + x^3 + x^2 + x = x * (x + 1) * (x^3 +x^2+ 1)$

НОД$(u_1(x), h_1(x) - c)$

НОД$(u_2(x), h_2(x) - c)$



$R = 3$

Анализ реализации

- Как видно, «шаги» алгоритма сами по себе являются весьма сложными действиями и тоже требуют алгоритмизации.
- Поэтому, необходимо реализовать арифметику многочленов над конечным полем, некоторые дополнительные функции для них (НОД и производные), функции составления таблицы B по многочлену, ее обработки, нахождения базиса решений системы уравнений по таблице и перебор разлагающих многочленов.
- Для этого были созданы два класса и соответствующие шагам алгоритма функции.

Описание класса многочлена над GF

- class gPol
- {
- public:
- gPol(const vector<int>& coefs); //Инициализация многочлена как массива коэффициентов
- int getDeg()const; //Нахождение степени многочлена
- gPol operator+(const gPol& obj)const;
- gPol operator-(const gPol& obj)const;
- gPol operator*(const gPol& obj)const;
- gPol operator/(const gPol& obj)const;
- gPol operator%(const gPol& obj)const;
- bool operator==(const gPol& obj)const;
- bool operator!=(const gPol& obj)const;

//Арифметические и логические действия
//с многочленами

- bool isZero()const; //Является ли многочлен нулевым

- gPol differentiate()const; //Производная многочлена
- gPol gcd(const gPol& obj)const; //НОД многочленов
- gPol checkSquares()const; //Проверка на квадраты
- vector<int> getCoefficients(); //Возвращает коэффициенты многочлена
- pair<gPol, gPol> divmod(const gPol& obj)const; //”Двойное деление” Возвращает и //результат деления и остаток от него
- private:
- unsigned short order; //Порядок поля
- vector<int> coefficients; //Коэффициенты многочлена
- void normalize(); //Приведения к нормальному виду //(избавление от лишних нулей)
- };
- ostream& operator<<(ostream& out, gPol p); //Вывод многочлена

Описание класса матрицы

- class Matrix
- {
- public:
- static Matrix unit(const int size); //Единичная матрица нужного размера
- void set(const int row, const int column, int value); //Установить значение в ячейку матрицы
- char get(const int row, const int column)const; //Получить значение из ячейки матрицы
- int getSize()const; //Размер матрицы
- Matrix operator+(const Matrix& obj)const; //Арифметика матриц
- Matrix operator-(const Matrix& obj)const;
- void swapRows(const int r1, const int r2); //Поменять ряды местами
- void addRow(const int to, const int from); //Сложить ряды
- Matrix transpose()const; //Транспонирование матрицы
- private:
- int size; //Размерность матрицы
- vector<int> entries; //Значения матрицы
- };
- ostream& operator<<(ostream& out, Matrix matr); //Вывод матрицы

Описание основных функций

```
vector<gPol> factorize(const gPol& p);
```

//Основная функция факторизации, проверяет разложимость, объединяет шаги **№1-5**

```
Matrix berlekampBmatrix(const gPol& p);
```

//Составление матрицы B, шаг **№2**

```
vector<vector<int>> matrixNullSpace(const Matrix& B);
```

//Нахождение пространства решений матрицы B, шаги **№3-4**

```
vector<gPol> berlekamp(const gPol& p);
```

//Общий алгоритм Берлекэмпа, объединяет шаги **№2-5**

Пример работы программы

C:\Users\user\Desktop\Алгоритм Берлекэмпа.exe

```
Возьмем многочлен над полем GF(2): x^5 + x^3 + x^2 + x

Матрица B:
1, 0, 0, 0, 0
0, 0, 1, 0, 0
0, 0, 0, 0, 1
0, 0, 1, 1, 1
0, 1, 0, 0, 0

После преобразований:
0, 0, 0, 0, 0
0, 1, 0, 0, 1
0, 1, 1, 1, 0
0, 0, 0, 0, 0
0, 0, 1, 1, 1

Базис пространства решений:
1, 0, 0, 0, 0,
0, 0, 1, 1, 0,
0, 1, 1, 0, 1,

Факторизированный многочлен:
(x^1)(x^1 + 1)(x^3 + x^2 + 1)

Умножив x на x+1 и на x^3+x^2+1 получим: x^5 + x^3 + x^2 + x,
то есть исходный многочлен => разложение верно.

Для продолжения нажмите любую клавишу . . .
```

Средства разработки

- Программа написана на языке программирования C++ с использованием среды разработки Microsoft Visual Studio 14