Московский государственный университет
имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики
Кафедра математической кибернетики

Курсовая работа студента 318 группы
Жумабай Мусахана Нуржанулы

Тема курсовой работы:
«О полиномиальных функциях одной переменной в восьмизначной логике»

Научный руководитель:
д.ф.-м.н., профессор
Селезнева С. Н.

Москва, 2023

Содержание

1 Введение

В наше время функций k-значных логик применяются, например в криптографии. В частности, теория о полиномиальности k-значных функций используется для улучшение алгоритма гомоморфного шифрования [1].

Функция k-значной логики называется полиномиальной функцией, если ее можно представить полиномом по модулю k. В общем случае для произвольного k установлено, что при простом k все функции k-значной логики представимы полиномами по модулю k. При составном k, полиномы образуют собственный замкнутый подкласс $Pol \subseteq P_k$, не являющийся предполным в P_k [2]. Другими словами при составном k найдутся функции которые невозможно представить полиномомом по модулю k. Более того разные полиномы могут представлять одну функцию, что влечет с собой не однозначность. Для одноместных функций предложены канонические виды полиномов при $k = p^m$, где p — простое число, $m \geqslant 1$ [3–6]. При помощи этих видов посчитано количество одноместных полиномиальных функций в P_k [3–6].

Для функций многих переменных при составных k, предложен канонический вид полиномиальных функций [7]. Этот канонический вид полиномов обобщает [3–5] на функций многих переменных. В настоящей работе, опираясь на [6], получаем канонические полиномы одноместных функций из P_8, что однозначно определяет все полиномиальные функций одной переменой в восьмизначной логике.

2 Основные определения

Пусть $k \geqslant 2$, $E_k = \{0, 1, ...,\text{k}-1\}$. Функция $f : E_k \to E_k$, называется одноместной функцией k-значной логики. Множество всех одноместных k-значных функций обозначим P_k.

Мономом называется: $\underbrace{x \cdot \ldots \cdot x}_{s}$, если $s > 0$, или константа 1, если $s = 0$. Полиномом называется: $\sum\limits_{i=1}^{l} c_i x^{s_i}$, где $c_i \in E_k$, x^{s_i}—различные мономы, $i = 0, 1, \ldots, l$, $l \geqslant 1$, либо константа 0. Каждый полином будем рассматривать как полином по модулю k, приводя коэффициенты по модулю k, а также рассматривая операций сложения и умножения по модулю k. Если в каком-то полиноме все коэффициенты равны нулю, то это пустой полином, который представляет функцию k-значной логики, тождественно равную нулю. Константу 0 будем называть пустым полиномом. Функция k-значной логики называется полиномиальной, если ее можно представить полиномом по модулю k. Множество всех полиномиальных функций k-значной логики обозначим Pol_k. Равенство $P_k = Pol_k$ верно тогда и только тогда, когда k простое число [2].

3 Постановка задачи

В данной курсовой работе предполагается найти все полиномиальные функций одной переменной в восьмизначнойной логике.

1. Изучить канонический вид одноместных полиномиальных функций восьмизначной логики из [6].

2. Написать программу, которая перебирает все канонические полиномы одноместных полиномиальных функций восьмизначной логики и по каждому полиному строит вектор значений функции, которую он задает.

3. Построить таблицы вывода программы из п. 2, введя классы эквивалентности функций, если требуется.

4 Основная часть

4.1 Вид канонических полиномов

Общие определения из [6]:

Определение 1. $I(k,n)$ обозначает множество всех полиномов по модулю k, зависящих от n переменных, которые представляют функцию k-значной логики, тождественно равную нулю.

Определение 2. Пусть p - простое число, $m \geqslant 1$. Определим понятие описывающие ограничения степеней полиномов по модулю p^m, представляющих одноместные полиномиальные функции p^m-значной логики. $c_{p,m}(x^s)$ для степени переменной x^s, $s \geqslant 0$, как наибольшее число t из чисел $0, 1, \ldots, m-1, m$, для которого существует полином

$$g(x) = x^s + c_{s-1}x^{s-1} + \ldots + c_1 x \in I(p^t, 1),$$

с коэффициентами $c_1, \ldots, c_{s-1} \in E_{p^m}$.

В [6] получено следующая теорема.

Теорема 1. Если $k = p^m$, где p — простое число, $m \geqslant 1$, то каждая функция $f(x_1, ..., x_n) \in Pol_k$ однозначно задается полиномом по модулю k вида

$$f(x_1, ..., x_n) = \sum_{j=1}^{l} c_j x^{s_j},$$

$c_{p,m}(x^{s_j}) \leqslant m-1$, $c_j \in E_k$, $c_j < p^{m - c_{p,m}(x^{s_j})}, j = 1, \ldots, l$.

Как следствие из условий теоремы 1, получены канонические полиномы в P_8. Они выглядят следующим образом:
$ax^3 + bx^2 + cx + d$, где a, $b \in \{0,1,2,3\}$, c, $d \in \{0,1,2,3,4,5,6,7\}$.

Утверждение 1. Справедливо равенство

$$|Pol_8^{(1)}| = 1024.$$

Доказательство. Для функций $ax^3 + bx^2 + cx + d$, где a, $b \in \{0,1,2,3\}$, c, $d \in \{0,1,2,3,4,5,6,7\}$, перебирая коэффициенты a, b, c, d существует $4^2 \cdot 8^2 = 2^{10} = 1024$ различных комбинаций, что определяет количество функции в $Pol_8^{(1)}$.

4.2 Реализация программы.

Основная идея программы: перебор коэффициентов для полинома. Для каждого набора коэффициентов для полинома программа вычисляет его вектор значений. На выходе программа строит таблицы вектор значений функций. Так как полиномы являются каноническими они отличны друг от друга.

Таблицы в работе были построены, заданных следующими полиномами: Так как в программе полиномы хранятся как вектор коэффициентов, будем их рассматривать в виде $[a, b, c, d]$. Коэффициенты a, b, c, d ограничены: $a, b \in \{0, 1, 2, 3\}, c, d \in \{0, 1, 2, 3, 4, 5, 6, 7\}$.

1. $f_0(x) = x^3 + bx^2 + cx + d$; ($[1, b, c, d]$ - всего 256 функции)
2. $f_1(x) = 2x^3 + bx^2 + cx + d$; ($[2, b, c, d]$ - всего 256 функции)
3. $f_2(x) = x^2 + cx + d$; ($[0, 1, c, d]$ - всего 64 функции)
4. $f_3(x) = 2x^2 + cx + d$; ($[0, 2, c, d]$ - всего 64 функции)
5. $f_4(x) = x + d$; ($[0, 0, 1, d]$ - всего 8 функции)
6. $f_5(x) = 2x + d$; ($[0, 0, 2, d]$ - всего 8 функции)
7. $f_6(x) = 4x + d$; ($[0, 0, 4, d]$ - всего 8 функции)
8. $f_7(x) = d$. ($[0, 0, 0, d]$ - всего 8 функции)

Еще есть функции $3f_0$, $3f_2$, $3f_4$, $5f_4$, $7f_4$, $3f_6$, их нет в таблице, так как они выражаются через функций из таблицы.

4.3 Вывод программы в виде таблиц значений

1. Таблицы значений для функций вида [1, b, c, d]

1.1. Таблица для f = [1, 0, c, d]:

x	1,0,0,0	1,0,1,0	1,0,2,0	1,0,3,0	1,0,4,0	1,0,5,0	1,0,6,0	1,0,7,0
0	0	0	0	0	0	0	0	0
1	1	2	3	4	5	6	7	0
2	0	2	4	6	0	2	4	6
3	3	6	1	4	7	2	5	0
4	0	4	0	4	0	4	0	4
5	5	2	7	4	1	6	3	0
6	0	6	4	2	0	6	4	2
7	7	6	5	4	3	2	1	0

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

1.2. Таблица для f = [1, 1, c, d]:

x	1,1,0,0	1,1,1,0	1,1,2,0	1,1,3,0	1,1,4,0	1,1,5,0	1,1,6,0	1,1,7,0
0	0	0	0	0	0	0	0	0
1	2	3	4	5	6	7	0	1
2	4	6	0	2	4	6	0	2
3	4	7	2	5	0	3	6	1
4	0	4	0	4	0	4	0	4
5	6	3	0	5	2	7	4	1
6	4	2	0	6	4	2	0	6
7	0	7	6	5	4	3	2	1

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

1.3. Таблица для f = [1, 2, c, d]:

x	1,2,0,0	1,2,1,0	1,2,2,0	1,2,3,0	1,2,4,0	1,2,5,0	1,2,6,0	1,2,7,0
0	0	0	0	0	0	0	0	0
1	3	4	5	6	7	0	1	2
2	0	2	4	6	0	2	4	6
3	5	0	3	6	1	4	7	2
4	0	4	0	4	0	4	0	4
5	7	4	1	6	3	0	5	2
6	0	6	4	2	0	6	4	2
7	1	0	7	6	5	4	3	2

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

1.4. Таблица для f = [1, 3, c, d]:

x	1,3,0,0	1,3,1,0	1,3,2,0	1,3,3,0	1,3,4,0	1,3,5,0	1,3,6,0	1,3,7,0
0	0	0	0	0	0	0	0	0
1	4	5	6	7	0	1	2	3
2	4	6	0	2	4	6	0	2
3	6	1	4	7	2	5	0	3
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	4	2	0	6	4	2	0	6
7	2	1	0	7	6	5	4	3

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

2. Таблицы значений для полиномов вида: [2, b, c, d]

2.1. Таблица для f = [2, 0, c, d]:

x	2,0,0,0	2,0,1,0	2,0,2,0	2,0,3,0	2,0,4,0	2,0,5,0	2,0,6,0	2,0,7,0
0	0	0	0	0	0	0	0	0
1	2	3	4	5	6	7	0	1
2	0	2	4	6	0	2	4	6
3	6	1	4	7	2	5	0	3
4	0	4	0	4	0	4	0	4
5	2	7	4	1	6	3	0	5
6	0	6	4	2	0	6	4	2
7	6	5	4	3	2	1	0	7

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

2.2. Таблица для f = [2, 1, c, d]:

x	2,1,0,0	2,1,1,0	2,1,2,0	2,1,3,0	2,1,4,0	2,1,5,0	2,1,6,0	2,1,7,0
0	0	0	0	0	0	0	0	0
1	3	4	5	6	7	0	1	2
2	4	6	0	2	4	6	0	2
3	7	2	5	0	3	6	1	4
4	0	4	0	4	0	4	0	4
5	3	0	5	2	7	4	1	6
6	4	2	0	6	4	2	0	6
7	7	6	5	4	3	2	1	0

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

2.3. Таблица для f = [2, 2, c, d]:

x	2,2,0,0	2,2,1,0	2,2,2,0	2,2,3,0	2,2,4,0	2,2,5,0	2,2,6,0	2,2,7,0
0	0	0	0	0	0	0	0	0
1	4	5	6	7	0	1	2	3
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	4	1	6	3	0	5	2	7
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

2.4. Таблица для f = [2, 3, c, d]:

x	2,3,0,0	2,3,1,0	2,3,2,0	2,3,3,0	2,3,4,0	2,3,5,0	2,3,6,0	2,3,7,0
0	0	0	0	0	0	0	0	0
1	5	6	7	0	1	2	3	4
2	4	6	0	2	4	6	0	2
3	1	4	7	2	5	0	3	6
4	0	4	0	4	0	4	0	4
5	5	2	7	4	1	6	3	0
6	4	2	0	6	4	2	0	6
7	1	0	7	6	5	4	3	2

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

3. Таблицы значений для полиномов вида: [0, 1, c, d]

x	0,1,0,0	0,1,1,0	0,1,2,0	0,1,3,0	0,1,4,0	0,1,5,0	0,1,6,0	0,1,7,0
0	0	0	0	0	0	0	0	0
1	1	2	3	4	5	6	7	0
2	4	6	0	2	4	6	0	2
3	1	4	7	2	5	0	3	6
4	0	4	0	4	0	4	0	4
5	1	6	3	0	5	2	7	4
6	4	2	0	6	4	2	0	6
7	1	0	7	6	5	4	3	2

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

4. Таблица значений для полиномов вида: [0, 2, c, d]

x	0,2,0,0	0,2,1,0	0,2,2,0	0,2,3,0	0,2,4,0	0,2,5,0	0,2,6,0	0,2,7,0
0	0	0	0	0	0	0	0	0
1	2	3	4	5	6	7	0	1
2	0	2	4	6	0	2	4	6
3	2	5	0	3	6	1	4	7
4	0	4	0	4	0	4	0	4
5	2	7	4	1	6	3	0	5
6	0	6	4	2	0	6	4	2
7	2	1	0	7	6	5	4	3

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

5-8. Таблица значений для полиномомов вида: [0, 0, c, d],
где c = {0, 1, 2, 4}

x	0,0,1,0	0,0,2,0	0,0,4,0	0,0,0,0
0	0	0	0	0
1	1	2	4	0
2	2	4	0	0
3	3	6	4	0
4	4	0	0	0
5	5	2	4	0
6	6	4	0	0
7	7	6	4	0

Еще есть функции: f + d, где d = {1, 2, 3, 4, 5, 6, 7}

5 Заключение

В ходе работы были изучены теория о числе полиномиальных функций многозначной логики по составному модулю k и о каноническом виде полиномов. Реализована программа, которая выдает в виде таблиц значений все полиномиальные функции одной переменной в восьмизначной логике.

6 Литература

Список литературы

[1] Robin G., Ilia I. On polynomial functions modulo p^e and faster bootstrapping for homomorphic encryption // advances in cryptology–EUROCRYPT 2023: 42nd annual international conference on the theory and applications of cryptographic techniques, Lyon, France, Proc. Part III. 2023. P. 257–286.

[2] Яблонский С. В. Функциональные построения в k-значной логике // Труды Матем. ин-та им. В.А. Стеклова АН СССР. 1958. Т. 51. С. 5–142.

[3] Niven I., Warren L.J̇. A generalization of Fermat's theorem // Proc. Amer. Math. Soc. 1957. V. 8. P. 306–313.

[4] Keller G., Olson F.Ṙ. Counting polynomial (mod p^n) // Duke Math. J.. 1968. V. 35, N 4. P. 835–838.

[5] Singmaster D. On polynomial functions (mod m) // J. Number Theory. 1974. V. 6, N 5. P. 345–352.

[6] Селезнева С. Н. О числе полиномиальных функций k-значной логики по составному модулю k // Дискретная математика. 2016. Т. 28, вып. 2. С. 81–91.

[7] Селезнева С. Н. Быстрый алгоритм построения для k-значных функций полиномов по модулю k при составных k // Дискретная математика. 2011. Т. 23, вып. 3. С. 3–22.

7 Приложение

```
#include <iostream>
#include <vector>

class Polynomial_Functions {
        private:
        std::vector<std::vector<uint32_t>> Functions;
        std::vector<std::string> Polynoms;
        std::vector<uint32_t> coeff_1 = {0, 1, 2, 3};
        std::vector<uint32_t> coeff_2 = {0, 1, 2, 3, 4, 5, 6, 7};
        public:
        void get_all_functions();
        void get_vector(std::vector<uint32_t> coef, std::vector<uint32_t> &result);
        void make_string(std::vector <uint32_t> vector);
        void print();
};

void Polynomial_Functions::get_all_functions() {
        std::vector <uint32_t> polynom(4);
        std::vector <uint32_t> result(8);
        uint32_t j,k,m;

        for(j=0; j<4; ++j) {
                for(k=0; k<8; ++k) {
                        polynom[0] = 1;
                        polynom[1] = coeff_1[j];
                        polynom[2] = coeff_2[k];
                        polynom[3] = 0;
                        get_vector(polynom, result);
                        make_string(polynom);
                        Functions.push_back(result);
                }
        }

        for(j=0; j<4; ++j) {
                for(k=0; k<8; ++k) {
                        polynom[0] = 2;
                        polynom[1] = coeff_1[j];
                        polynom[2] = coeff_2[k];
                        polynom[3] = 0;
                        get_vector(polynom, result);
                        make_string(polynom);
                        Functions.push_back(result);
```

```
        }
}

for(k=0; k<8; ++k) {
        polynom[0] = 0;
        polynom[1] = 1;
        polynom[2] = coeff_2[k];
        polynom[3] = 0;
        get_vector(polynom, result);
        make_string(polynom);
        Functions.push_back(result);
}

for(k=0; k<8; ++k) {
        polynom[0] = 0;
        polynom[1] = 2;
        polynom[2] = coeff_2[k];
        polynom[3] = 0;
        get_vector(polynom, result);
        make_string(polynom);
        Functions.push_back(result);
}

polynom[0] = 0;
polynom[1] = 0;
polynom[2] = 1;
polynom[3] = 0;
get_vector(polynom, result);
make_string(polynom);
Functions.push_back(result);

polynom[0] = 0;
polynom[1] = 0;
polynom[2] = 2;
polynom[3] = 0;
get_vector(polynom, result);
make_string(polynom);
Functions.push_back(result);

polynom[0] = 0;
polynom[1] = 0;
polynom[2] = 4;
polynom[3] = 0;
get_vector(polynom, result);
```

```
        make_string(polynom);
        Functions.push_back(result);

        polynom[0] = 0;
        polynom[1] = 0;
        polynom[2] = 0;
        polynom[3] = 0;
        get_vector(polynom, result);
        make_string(polynom);
        Functions.push_back(result);
}

void Polynomial_Functions::get_vector(std::vector<uint32_t> coef, std::vector<uint32_t>
&result) {
        for(int i = 0; i < 8; ++i) {
                result[i] = (coef[0]*i*i*i + coef[1]*i*i + coef[2]*i + coef[3]) % 8;
        }
}

void Polynomial_Functions::make_string(std::vector<uint32_t> vector) {
        std::string str;
        for(int i=0;i<4;++i) {
                if(i==3) {
                        str += std::to_string(vector[i]);}
                else
                str += std::to_string(vector[i]) + ',';
        }
        Polynoms.push_back(str);
}

void Polynomial_Functions::print() {
        uint32_t number = 8;
        std::cout<<"|x|";
        for(int m=0;m<8;++m) {
                std::cout << Polynoms[m]<< "|";
        }
        std::cout<<std::endl;
        for(int k=0;k<11;++k) {
                for(int i=0;i<8;++i) {
                        std::cout << "|" << i << "|___";
                        for(int j=0;j<8;j++) {
                                if(j+8*k>=84) {
                                        continue;}
                                else {
```

```
                                        std::cout << Functions[j+8*k][i] <<"   |   ";}
                                if(((i+1) % 8 == 0) && ((j+1) % 8 ==0 && k!=10)) {
                                        std::cout<<"\n\n|x|";
                                        for(int m=0;m<8;++m) {
                                                if(number>=84) {
                                                        continue;}
                                                else {
                                                        std::cout<< Polynoms[number]<< "|";
                                                        number++;}
                                        }
                                }
                        }
                        std::cout << std::endl;
                }
        }
}

int main() {
        Polynomial_Functions Test;
        Test.get_all_functions();
        Test.print();
        return 0;
}
```

Пояснение к программе:

class Polynomial: имеет четыре приватных поля. Functions - для хранение вектор значений, Polinoms - для хранения полиномомов в виде вектора коэффициентов. Векторы целых чисел coeff_1 и coeff_2 нужны для перебора коэффциентов. Есть еще четыре публичных метода.

Основной метод "get_all_functions". В этом методе осуществляется перебор коэффициетов. Восемь отдельных циклов, каждая соответсвует классу функций из таблицы в разделе "основной части". В цикле происходит переобор коэффициентов, для мономов x^3 и x^2 коэффициент из {0, 1, 2, 3}, для мономов x^1 и x^0 из {0, 1, 2, 3, 4, 5, 6, 7}. Для каждого набора коэффициентов программа вычисляет вектор значение для соответсвующей функций, для этого используется метод "get_vector". Полиномы храним в виде вектора коэффициентов, есть метод "make_string" для удобного хранения в виде строки в поле $Polinoms$. Метод print() выводит все функции в виде таблицы значений на консоль.