ISSN 2225-9309 (Print)
2026. № 1 (35). С. 76–79
Молодежный Вестник УГАТУ
http://journal.uust.ru

УДК 519.115.4
doi 10.54708/22259309_2026_13576

РЕАЛИЗАЦИЯ ПОЛИНОМА ЖЕГАЛКИНА МЕТОДОМ ТРЕУГОЛЬНИКА ПАСКАЛЯ НА PYTHON

А. М. ХАММАТОВ[1]

[1]khammatov-2022@mail.ru

[1]ФГБОУ ВО «Уфимский университет науки и технологий» (УУНиТ)

Аннотация. В статье рассматривается метод построения полинома Жегалкина с использованием треугольника Паскаля. Представлена программная реализация алгоритма на Python для автоматизации вычислений. Исследованы теоретические основы и практическое применение метода в криптографии и теории кодирования. Проведены анализ вычислительной сложности алгоритма и сравнение с альтернативными подходами, а также сравнение предложенного метода с другими: матричным методом и методом неопределенных коэффициентов. Показано, что метод треугольника Паскаля обладает оптимальным балансом между простотой реализации и вычислительной эффективностью для функций с небольшим числом переменных.

Ключевые слова: полином Жегалкина; треугольник Паскаля; булевы функции; криптографический анализ; Python; вычислительная сложность.

ВВЕДЕНИЕ

Полином Жегалкина представляет булеву функцию $f(x_1,...,x_n)$ в виде:

$$f(x_1,...,x_n) = a_0 \oplus a_1x_1 \oplus ... \oplus a_nx_n \oplus a_{12}x_1x_2 \oplus ... \oplus a_{1...n}x_1...x_n,$$

где:

$a_i \in \{0,1\}$ – коэффициенты полинома;

$\oplus$ – операция сложения по модулю 2;

произведения переменных – конъюнкции.

Основные свойства:

1. Единственность представления для каждой булевой функции.
2. Степень полинома характеризует нелинейность функции.
3. Линейные функции имеют полиномы степени не выше 1.

Модифицированный треугольник Паскаля для построения полинома Жегалкина:

1. Исходная строка – вектор значений булевой функции.
2. Каждая следующая строка вычисляется как XOR соседних элементов предыдущей.
3. Коэффициенты полинома берутся из последней строки треугольника.
4. Пространственная сложность: $O(2^n)$, где n – количество входных наборов.

АЛГОРИТМ ПРОГРАММЫ

Алгоритм построения полинома на языке Python 3.9 представлен ниже.

Рассмотрим более детально весь код. Инициализация переменных и ввод данных изображены на рис. 1.

```
from itertools import product

triangle = []
table = []
stroka = 0
step = 0
f = ""
dop = ""
itog = ""
count = int(input("Введите количество x-ов:\n"))
function = input("Введите значение функции:\n")
```

Рис. 1. Ввод данных

На рис. 2 показано, как складывать по модулю 2 соседних элемента и выводить новую строчку, затем получая треугольник Паскаля.

```
triangle = [int(char) for char in function]
if triangle[0] == 1:
    table.append(stroka)
print("\nТреугольник Паскаля:\n")
print(*triangle)
for k in range(len(triangle) - 1):
    stroka += 1
    new_triangle = []
    for i in range(len(triangle) - 1):
        if triangle[i] + triangle[i + 1] == 1:
            new_triangle.append(1)
        else:
            new_triangle.append(0)
    triangle = new_triangle
    if len(triangle) > 0 and triangle[0] == 1:
        table.append(stroka)
    print(*triangle)
```

Рис. 2. Треугольник Паскаля

На рис. 3 изображено, как сделать таблицу истинности.

```
print("\nТаблица истинности:\n")
for y in range(count):
    dop += "x" + str(y + 1)

combinations = list(product([0, 1], repeat=count))

for s in range(count):
    itog += "x" + str(s + 1) + " | "
itog += "f |\n"
for e in range(2**count):
    for j in range(count):
        itog += f"{combinations[e][j]}  | "
    itog += f"{function[e]} |"
    itog += "\n"
print(itog)
for z in range(2**count):
    if step in table:
        if 1 not in combinations[z]:
                f += "1 |+| "
        for n in range(count):
            if combinations[z][n] == 1:
                f += "x" + str(n + 1) + ""
            if n == (count - 1) and (step < max(table)) and (1  in combinations[z]):
                f += " |+| "
    step += 1
print("Полином Жигалкина:\n")
```

Рис. 3. Таблица истинности

На рис. 4 продемонстрирован вывод полинома Жегалкина для заданной функции.

```
print("Полином Жигалкина:\n")
print(f"f({dop}) = {f}\n")
```

Рис. 4. Полином Жегалкина

На рис. 5 мы видим, как работает программа на примере f=10110101 и 3 x-ов.

```
C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe
Введите количество x-ов:
3
Введите значение функции:
10110101

Треугольник Паскаля:

1 0 1 1 0 1 0 1
1 1 0 1 1 1 1
0 1 1 0 0 0
1 0 1 0 0
1 1 1 0
0 0 1
0 1
1

Таблица истинности:

x1 | x2 | x3 | f |
0  | 0  | 0  | 1 |
0  | 0  | 1  | 0 |
0  | 1  | 0  | 1 |
0  | 1  | 1  | 1 |
1  | 0  | 0  | 0 |
1  | 0  | 1  | 1 |
1  | 1  | 0  | 0 |
1  | 1  | 1  | 1 |

Полином Жигалкина:

f(x1x2x3) = 1 |+| x3 |+| x2x3 |+| x1 |+| x1x2x3

Press any key to continue . . .
```

Рис. 5. Пример работы программы

Рассмотрим сравнение различных методов, представленных в табл. 1.

Таблица 1

Сравнительный анализ различных методов

Метод	Временная сложность	Преимущества	Недостатки
Треугольник Паскаля	$O(n \cdot 2^n)$	Простота реализации	Требует полной ТИ
Матричный метод	$O(2^{3n})$	Системный подход	Высокая сложность
Метод неопределенных коэффициентов	$O(2^{3n})$	Системный подход	Экспоненциальная сложность

ЗАКЛЮЧЕНИЕ

В работе представлены эффективный метод построения полинома Жегалкина с использованием модифицированного треугольника Паскаля и его программная реализация на Python. Разработанный алгоритм обладает приемлемой вычислительной сложностью и может быть использован для анализа булевых функций в различных прикладных областях.

СПИСОК ЛИТЕРАТУРЫ

1. **Николаев, Д. В.** Методы анализа булевых функций в криптографии. – М.: Физматлит, 2022. – 256 с.
2. **Сергеев, И. В.** Современные алгоритмы дискретной математики. – СПб.: Лань, 2023. – 320 с.
3. **Применение** полиномов Жегалкина в криптографии // Кибернетика и системный анализ. – 2021. – № 4. – С. 45-53.

ОБ АВТОРАХ

ХАММАТОВ Артур Маратович, студент ПРО ИИМРТ УУНиТ.

METADATA

Title: Implementation of the Zhegalkin polynomial by the newton triangle method in Python

Author: A.M. Hammatov[1]

Affiliation:

[1] Ufa University of Science and Technology (UUST), Russia.

Email: [1]khammatov-2022@mail.ru

Language: Russian.

Source: Molodezhnyj Vestnik UGATU (scientific journal of Ufa University of Science and Technology), no. 1 (35), pp. 76-79, 2026. ISSN 2225-9309 (Print).

Abstract: The article discusses the method of constructing the Zhegalkin polynomial using Pascal's triangle. A software implementation of the algorithm in Python for automating calculations is presented. The theoretical foundations and practical application of the method in cryptography and coding theory are investigated. The computational complexity of the algorithm is analyzed and compared with alternative approaches. The proposed method is compared with others: the matrix method and the method of indeterminate coefficients. It is shown that the Pascal triangle method has an optimal balance between simplicity of implementation and computational efficiency for functions with a small number of variables.

Key words: Zhegalkin's polynomial, Pascal's triangle, Boolean functions, cryptographic analysis, Python, computational complexity.

About authors:

Hammatov Artur Maratovich, student, Dept. of Computational Mathematics and Cybernetics (UUST)