НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Лабораторная работа №3

По дисциплине «Криптографические методы защиты информации»

«Преобразования Уолша и Фурье. Статистическая структура булевой функции»

Группа: А-14м-24

Студент: Гороховский И. А.

Преподаватель: Фролов А. Б.

Москва 2025

Задание 3.1

Напишите на Python программу построения преобразования Уолша индуктивным методом и представьте листинги ее применения к функциям от 6 переменных.

Решение

Для задания была выбрана следующая функция от 6 переменных:

Таблица истинности:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | X4 | X5 | X6 | f |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | X4 | X5 | X6 | f |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | X4 | X5 | X6 | f |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Используя куб, создаются сигнальный вектор и характеристическая последовательность функции:

*# Генерация куба истинности для N переменных*def generate\_cube(N):  
 return [bin(i)[2:].zfill(N) for i in range(2 \*\* N)]  
  
  
*# Булева функция для 6 переменных*def boolean\_function(cube):  
 return [(int(it[0]) ^ (int(it[1]) \* int(it[2])) ^ (int(it[3]) \* int(it[4]) \* int(it[5]))) for it in cube]  
  
  
*# Функция для преобразования сигнального вектора в характеристическую последовательность*def sign\_to\_char(f):  
 return [(-1) \*\* bit for bit in f]  
  
  
N = 6 *# Число переменных*cube = generate\_cube(N) *# Генерация куба истинности для 6 переменных*f = boolean\_function(cube) *# Вычисление сигнального вектора*F = sign\_to\_char(f) *# Вычисление характеристической последовательности*

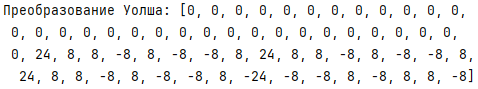
Преобразование Уолша вычисляется индуктивно с использованием метода деления на подзадачи. Исходная функция сначала представляется как последовательность чисел, а затем для каждого уровня преобразования применяется деление и сложение по определённым индексам. Это позволяет получить новое представление функции, использующее только операции сложения и вычитания.

*# Преобразование Уолша*def walsh\_transform(a, N):  
 w = a[:] *# Копируем список для преобразования* for n in range(1, N + 1): *# Проходим по всем уровням преобразования* for i in range(0, 2 \*\* (N - n)):  
 for j in range(2 \*\* (n - 1)):  
 t = 2 \*\* n \* i + j  
 tt = t + 2 \*\* (n - 1)  
 if tt < len(w): *# Проверяем, чтобы индексы не выходили за пределы* d = w[t]  
 w[t] = d + w[tt]  
 w[tt] = d - w[tt]  
 return w

w = walsh\_transform(F, N)

print("Преобразование Уолша:", w)

Результат:



Задание 3.2

Покажите, что преобразование Фурье также можно строить индуктивным методом и напишите программу, представьте листинги ее применения к функциям от 6 переменных.

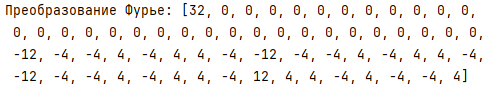
Решение

Преобразование Фурье использует тот же индуктивный метод, что и преобразование Уолша, так как они математически эквивалентны. В данном коде преобразование Фурье выполняется с использованием функции walsh\_transform, что демонстрирует их аналогию. Единственная разница – на вход Фурье подаётся сигнальный вектор, а не характеристическая последовательность.

*# Преобразование Фурье (использует преобразование Уолша)*def fourier\_transform(f, N):  
 return walsh\_transform(f, N)

fft\_result = fourier\_transform(f, N)  
print("Преобразование Фурье:", fft\_result)

Результат:



Задание 3.3

Написать или применить программу вычисления статистической структуры булевой функции и привести листинг для функции от 6 переменных.

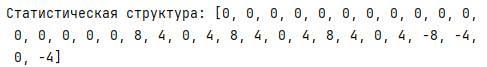
Решение

Статистическая структура вычисляется с использованием метода рекурсивного обновления для всех элементов функции. Для каждого уровня применяются операции сложения и вычитания для соответствующих элементов массива, что позволяет оценить структуру функции с точки зрения её статистических характеристик.

*# Вычисление статистической структуры функции*def statistical\_structure(a, N):  
 d = [(a[2 \* j] + a[2 \* j + 1]) // 2 for j in range(2 \*\* (N - 1))]  
 for n in range(2, N + 1):  
 for i in range(0, 2 \*\* (N - n)):  
 for j in range(2 \*\* (n - 1)):  
 t = 2 \*\* n \* i + j  
 tt = t + 2 \*\* (n - 1)  
 *# Проверяем, чтобы индексы не выходили за пределы* if t < len(d) and tt < len(d):  
 d[t], d[tt] = d[t] + d[tt], d[t] - d[tt]  
 return d

d = statistical\_structure(F, N) *# Статистическая структура*print("Статистическая структура:", d)

Результат:



Задание 3.4

Проиллюстрируйте метрическую трактовку преобразования Уолша.

Решение

Метрическая трактовка преобразования Уолша иллюстрируется через вычисление расстояний Хемминга между функцией и её линейными или аффинными образами. Эти расстояния показывают, насколько сильно различаются функции, представленные в различных формах. Это даёт представление о том, как преобразование Уолша меняет структуру функции в метрическом пространстве. В коде для этого написана функция hamming\_distance, которая на вход получает сигнальный вектор и куб, а на выходе даёт расстояния до линейной и аффинной функции.

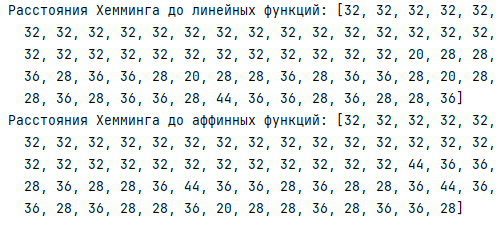
*# Вычисление расстояний Хемминга до линейных и аффинных функций*def hamming\_distance(f, cube):  
 hdl, hda = [], []  
 for it in cube:  
 l = linear\_function(cube, it)  
 *# Расстояние до линейной функции* wl = sum((l[i] + f[i]) % 2 for i in range(len(f)))  
 *# Расстояние до аффинной функции* wa = sum((l[i] + f[i] + 1) % 2 for i in range(len(f)))   
 hdl.append(wl)  
 hda.append(wa)  
 return hdl, hda

hd = hamming\_distance(f, cube) # Расстояния Хемминга

print("Расстояния Хемминга до линейных функций:", hd[0])

print("Расстояния Хемминга до аффинных функций:", hd[1])

Результат:



Полная версия кода программы доступна на Github:

<https://github.com/Zaicol/CryptoMPEI/blob/master/task_3.py>