**Двоичное кодирование целых чисел и коды Грея**

# Введение

Современные цифровые системы оперируют дискретными величинами, и двоичное кодирование лежит в основе хранения и обработки информации в компьютерах и электронике. Однако при практическом опыте работы с измерительным оборудованием и средствами передачи данных выясняется, что привычное двоичное положение разрядов может приводить к ошибкам при одновременном переключении нескольких линий. В качестве альтернативы в 1930-х годах был предложен код, названный в честь Фрэнка Грея, в котором последовательные значения отличаются ровно одним разрядом. Данный доклад подробно рассматривает: принципы двоичного кодирования целых чисел; устройство и алгоритмы формирования кодов Грея; сравнение двух подходов; реальные области применения Gray-кода; примеры и иллюстрации.

# Двоичное кодирование целых чисел

Позиционная система счисления с основанием 2 использует два символа: 0 и 1. Чтобы перевести число из **десятичной системы** (основание 10) в **двоичную** (основание 2), нужно:

1. Последовательно делить число на 2.
2. Записывать остатки от деления.
3. Читать остатки **снизу вверх**.

def to\_binary(n):

if n == 0:

return '0'

b = ''

while n:

b = str(n % 2) + b

n //= 2

return b

n = int(input())

print(to\_binary(n))

# Коды Грея

Код Грея — последовательность двоичных кодов, где каждое последующее значение отличается от предыдущего только одним битом. Построение осуществляется рекурсивно или через операцию XOR. Преимущество кода Грея — минимизация ошибок при переходах между значениями, особенно в аппаратных средствах.

1 СПОСОБ

Коды Грея можно генерировать рекурсивно с использованием метода отражения:

1. Для 1 бита код Грея: 0, 1.
2. Для n битов:
   * Берется код Грея для n-1 битов.
   * Отражается (записывается в обратном порядке).
   * К оригинальному списку добавляется префикс 0, к отраженному — префикс 1.
   * Объединяются оба списка.

**Пример для n=3:**

* 2-битный код: 00, 01, 11, 10
* Отражение: 10, 11, 01, 00
* Префиксы: 000, 001, 011, 010 (оригинал), 110, 111, 101, 100 (отраженный)
* Итог: 000, 001, 011, 010, 110, 111, 101, 100

## Граф переходов для 3-битного кода Грея

000 ↔ 001 ↔ 011 ↔ 010 ↔ 110 ↔ 111 ↔ 101 ↔ 100 ↔ 000

Таблица кодов Грея для n=3:

|  |  |  |
| --- | --- | --- |
| **Число** | **Двоичный код** | **Код Грея** |
| 0 | 000 | 000 |
| 1 | 001 | 001 |
| 2 | 010 | 011 |
| 3 | 011 | 010 |
| 4 | 100 | 110 |
| 5 | 101 | 111 |
| 6 | 110 | 101 |
| 7 | 111 | 100 |

# def gray\_codes(n):

# codes = []

# total = 2 \*\* n

# for i in range(total):

# gray = i ^ (i >> 1)

# code\_str = format(gray, f'0{n}b')

# codes.append(code\_str)

# return codes

# if \_\_name\_\_ == "\_\_main\_\_":

# n = 3

# result = gray\_codes(n)

# print(result) # ['000', '001', '011', '010', '110', '111', '101', '100']

# 2 СПОСОБ

# def grey\_code(n):

# a = [0 for i in range(n + 1)]

# j = 0

# while True:

# yield a[n - 1::-1]

# a[n] = 1 - a[n]

# if a[n] == 1:

# j = 0

# else:

# for i in range(n):

# if a[i] == 1:

# j = i + 1

# break

# if j >= n:

# return

# a[j] = 1 - a[j]

# for code in grey\_code(3):

# print(code)

# Сравнение двоичного кода и кода Грея

Код Грея минимизирует количество изменяющихся битов при переходе между последовательными числами, что критично в задачах точного считывания состояния. Двоичное кодирование проще в арифметике, но менее надежно в условиях физических ошибок.

**Где используются коды Грея перед двоичным кодированием**

Коды Грея обладают значительными преимуществами в приложениях, где важна надежность:

1. **Минимизация ошибок при переходах**  
   В стандартном двоичном кодировании переход между числами может изменять несколько битов (например, от 3 (011) к 4 (100) меняются все три бита), что может вызвать ошибки из-за несинхронности. Коды Грея изменяют только один бит, снижая риск (Код Грея).
2. **Комбинаторные задачи**  
   Коды Грея используются для генерации всех подмножеств или перестановок, где каждое следующее состояние отличается минимально. Например, они помогают решать задачу о рюкзаке или генерировать комбинации.
3. **Применение в датчиках положения**  
   В ротационных и линейных энкодерах коды Грея обеспечивают точность, так как изменение положения меняет только один бит, минимизируя ошибки измерения.
4. **Коррекция ошибок в цифровых коммуникациях**  
   В системах связи, таких как квадратурная модуляция, коды Грея упрощают исправление ошибок, так как соседние точки созвездия отличаются одним битом.
5. **Снижение энергопотребления**  
   Меньшее число переключений битов может уменьшить энергопотребление в цифровых системах, что важно для мобильных устройств.
6. **Дополнительные применения**  
   Коды Грея используются в генетических алгоритмах, играх и для адресации в дисковых массивах.

**Пример использования кода Грея в декодирование сигнала ротационного энкодера**

**def gray\_to\_binary(gray: int) -> int:**

**binary = gray**

**while gray > 0:**

**gray >>= 1**

**binary ^= gray**

**return binary**

**class RotaryEncoder:**

**def \_\_init\_\_(self, resolution\_bits: int):**

**self.max\_count = 1 << resolution\_bits**

**def code\_to\_angle(self, gray\_code: int) -> float:**

**position = gray\_to\_binary(gray\_code)**

**return position / self.max\_count \* 360.0**

**encoder = RotaryEncoder(resolution\_bits=8)**

**raw\_gray = 0b11001010 # пример прочитанного кода**

**angle = encoder.code\_to\_angle(raw\_gray)**

**print(f"Угол: {angle:.2f}°")**

**Заключение**

Коды Грея — это мощный инструмент в цифровой электронике и информатике, обеспечивающий надежность в системах, где важна минимизация ошибок при переходах. Их свойство однобитного различия делает их незаменимыми в датчиках, системах связи и других областях.