ВМК МГУ

Задание 8. Коды БЧХ

Практикум 317 группы, весна 2015

Начало выполнения задания: 7 мая 2015
Срок сдачи: **20 мая 2015 (среда), 23:59.**
Среда для выполнения задания – PYTHON.

Содержание

Необходимая теория

Задача помехоустойчивого кодирования

Рассмотрим задачу передачи потока битовой информации по каналу с шумом с возможностью автоматического исправления ошибок, допущенных при передаче. При *блоковом* кодировании входящий поток информации разбивается на блоки фиксированной длины k. Обозначим один такой блок через $\boldsymbol{u} \in \{0,1\}^k$. Предполагается, что во входном потоке данных, вообще говоря, нет избыточности. Поэтому для реализации схемы, способной исправлять ошибки, необходимо закодировать блок $\boldsymbol{u}$ в некоторое кодовое слово большей длины путем добавления избыточности в передаваемые данные. Обозначим кодовое слово через $\boldsymbol{v} \in \{0,1\}^n$, $n > k$. Для кодирования всевозможных блоков $\boldsymbol{u}$ необходимо использовать 2^k кодовых слов длины n. Определим минимальное расстояние кода d как минимальное хэммингово расстояние для всех различных пар кодовых слов. Назовём множество 2^k кодовых слов длины n с минимальным расстоянием d (n,k,d)-*блоковым кодом*, а величину $r = k/n$ – *скоростью кода*. При передаче по каналу с шумом кодовое слово $\boldsymbol{v}$ превращается в принятое слово $\boldsymbol{w} \in \{0,1\}^n$, которое, вообще говоря, отличается от $\boldsymbol{v}$. Далее алгоритм декодирования пытается восстановить переданное слово $\boldsymbol{v}$ путем поиска среди всевозможных кодовых слов ближайшего к $\boldsymbol{w}$. Обозначим результат работы алгоритма декодирования через $\hat{\boldsymbol{v}}$. На последнем этапе декодированное слово $\hat{\boldsymbol{v}}$ переводится в декодированное слово исходного сообщения $\hat{\boldsymbol{u}}$. Очевидно, что (n,k,d)-блоковый код способен гарантированно обнаруживать до $d-1$ ошибки и исправлять до $\lfloor (d-1)/2 \rfloor$ ошибок.

Кодирование с помощью (n,k,d)-линейного циклического блокового кода

Множество $\{0,1\}^n$ с операциями суммы и произведения по модулю 2 образует линейное пространство над конечным полем $\mathbb{F}_2$. (n,k,d)-блоковый код называется *линейным*, если множество его кодовых слов образует линейное подпространство размерности k общего линейного пространства $\{0,1\}^n$. Таким образом, для линейного кода произвольная линейная комбинация кодовых слов является кодовым словом. Минимальное кодовое расстояние d для линейного кода определяется как минимальный хэммингов вес (количество ненулевых бит) среди ненулевых кодовых слов. (n,k,d)-линейный блоковый код называется *циклическим*, если любой циклический сдвиг кодового слова является кодовым словом. Поставим в соответствие произвольному вектору

$\boldsymbol{v} = [v_{n-1}, v_{n-2}, \dots, v_1, v_0] \in \{0,1\}^n$ полином вида $v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \cdots + v_1 x + v_0$. Тогда можно показать, что для (n,k,d)-линейного циклического блокового кода найдется полином $g(x)$ степени $m = n-k$ такой, что

- Все кодовые слова $v(x)$ могут быть представлены как $g(x)u(x)$, где $u(x)$ – некоторый полином степени, не превышающей $k-1$;
- Полином $g(x)$ является делителем полинома $x^n - 1$.

Такой полином $g(x)$ называется *порождающим полиномом циклического кода.* Любой полином, являющийся делителем x^n-1, является порождающим для некоторого циклического кода.

Кодирование называется *систематическим*, если все биты исходного сообщения $\boldsymbol{u}$ копируются в некоторые биты кодового слова $\boldsymbol{v}$. При систематическом кодировании обратный процесс преобразования из декодированного кодового слова $\hat{\boldsymbol{v}}$ в декодированное слово сообщения $\hat{\boldsymbol{u}}$ становится тривиальным. Для циклического кода, задаваемого порождающим полиномом $g(x)$, процесс систематического кодирования может быть реализован как

$$v(x) = x^m u(x) + \mod(x^m u(x), g(x)).$$

Здесь через $\mod(f(x), g(x))$ обозначена операция взятия остатка от деления многочлена $f(x)$ на многочлен $g(x)$.

Коды БЧХ: кодирование

Полином $m_\alpha(x) \in \mathbb{F}_2[x]$ называется *минимальным полиномом* для элемента $\alpha \in \mathbb{F}_2^q$, если он является неприводимым полиномом минимальной степени, для которого α является корнем. В частности, минимальный полином для примитивного элемента α называется *примитивным полиномом.* Можно показать, что корнями минимального полинома $m_\alpha(x)$ являются

$$\{\alpha, \alpha^2, \alpha^4, \dots, \alpha^{2^s}\}.$$

Данный набор элементов из поля $\mathbb{F}_2^q$ называется *циклотомическим классом смежности* для элемента α. Количество элементов в смежном классе либо равно q, либо является делителем q. Циклотомические классы, порождённые различными элементами поля, либо совпадают, либо не пересекаются. Можно показать, что полином

$$\prod_{i=0}^{s}(x + \alpha^{2^i}) = x^{s+1} + \lambda_s x^s + \cdots + \lambda_1 x + \lambda_0$$

имеет коэффициенты из $\mathbb{F}_2$ и является минимальным полиномом для α, а также для всех элементов поля, входящих вместе с α в один циклотомический класс. Отсюда выводится метод построения минимального полинома для заданного элемента поля α:

1. Построить циклотомический класс, порожденный элементом α;
2. Найти коэффициенты полинома $m_\alpha(x)$ путем перемножения многочленов $x + \alpha^{2^i}$ для всех $i = 0, \dots, s$.

Пусть $n = 2^q - 1$, $t \le \lfloor (n-1)/2 \rfloor$. Тогда *кодом БЧХ* называется (n,k)-линейный циклический код, в котором порождающий многочлен $g(x)$ определяется как минимальный многочлен для элементов $\alpha, \alpha^2, \alpha^3, \dots, \alpha^{2t}$ из поля $\mathbb{F}_2^q$, где α – произвольный примитивный элемент поля $\mathbb{F}_2^q$. Набор элементов $\alpha, \alpha^2, \dots, \alpha^{2t}$ называется *нулями БЧХ-кода.* Можно показать, что минимальное кодовое расстояние кода БЧХ d не меньше, чем величина $2t+1$. В результате БЧХ-коды по построению способны исправлять не менее t ошибок.

Коды БЧХ: декодирование

Поставим в соответствие позициям принятого слова $\boldsymbol{w} = [w_{n-1}, \dots, w_0]$ элементы $\alpha^{n-1}, \dots, \alpha^0$. При передаче по шумовому каналу кодовое слово $v(x)$ переходит в слово $w(x) = v(x) + e(x)$, где $e(x) = x^{j_1} + \cdots + x^{j_\nu}$ – полином ошибок, а $j_1, \dots, j_\nu$ – позиции, в которых произошли ошибки. Назовем *синдромами* принятого сообщения $w(x)$ значения полинома $w(x)$ в нулях БЧХ-кода, т.е. $s_i = w(\alpha^i),\ i = 1, \dots, 2t$. Если $w(x)$ является кодовым словом, то все синдромы $s_i = 0$. Рассмотрим *полином локаторов ошибок*

$$\Lambda(z) = \prod_{i=1}^{\nu}(1 + \alpha^{j_i} z) = \Lambda_\nu z^\nu + \cdots + \Lambda_1 z + 1.$$

Данный полином имеет корни α^{-j_i}. Можно показать, что коэффициенты полинома $\Lambda(z)$ удовлетворяют следующей СЛАУ:

$$\begin{bmatrix} s_1 & s_2 & \dots & s_\nu \\ s_2 & s_3 & \dots & s_{\nu+1} \\ \dots & \dots & \dots & \dots \\ s_\nu & s_{\nu+1} & \dots & s_{2\nu-1} \end{bmatrix} \begin{bmatrix} \Lambda_\nu \\ \Lambda_{\nu-1} \\ \dots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} s_{\nu+1} \\ s_{\nu+2} \\ \dots \\ s_{2\nu} \end{bmatrix}. \tag{1}$$

Отсюда получаем следующую общую схему декодирования БЧХ-кода:

1. Для принятого слова $w(x)$ вычислить синдромы $s_i = w(\alpha^i),\ i = 1, \dots, 2t$. Если все $s_i = 0$, то вернуть $w(x)$ в качестве ответа;

2. Найти количество допущенных ошибок ν и коэффициенты полинома локаторов ошибок путем решения СЛАУ (1);

3. Найти все корни полинома $\Lambda(z)$ путем полного перебора, по найденным корням вычислить номера позиций $j_1, \dots, j_\nu$, в которых произошли ошибки;

4. Исправить ошибки в позициях $j_1, \dots, j_\nu$ путем инвертирования соответствующих битов в $w(x)$.

Различные алгоритмы декодирования БЧХ-кодов по-разному решают задачу на шаге 2 общего алгоритма декодирования. Рассмотрим две схемы декодирования.

Декодер PGZ (Peterson–Gorenstein–Zierler)

Данный декодер предполагает непосредственное решение СЛАУ (1). Основная трудность здесь – это определить количество фактически допущенных при передаче ошибок ν. В декодере PGZ происходит перебор по всем значениям ν, начиная с t. При текущем ν делается попытка решить СЛАУ (1). Если матрица СЛАУ является невырожденной, то текущее ν признается количеством допущенных ошибок, а коэффициенты полинома локаторов ошибок находятся из решения СЛАУ. Если матрица СЛАУ является вырожденной, то $\Lambda_\nu = 0$, величина ν уменьшается на единицу, и процесс повторяется. Если СЛАУ решить не удается ни на одной итерации, то выдается отказ от декодирования. Также отказ от декодирования выдаётся в случае, если после исправления синдромы $\hat{v}(x)$ не равны нулю (кодовое слово не найдено).

Декодер Euclid

Рассмотрим *синдромный полином* вида $S(z) = s_{2t}z^{2t} + s_{2t-1}z^{2t-1} + \dots + s_1 z + 1$, где s_i – вычисленные ранее синдромы. Тогда можно показать, что $S(z)$ и $\Lambda(z)$ удовлетворяют следующему уравнению:

$$z^{2t+1}A(z) + S(z)\Lambda(z) = r(z).$$

Здесь $r(z)$ – некоторый многочлен из $\mathbb{F}_2^q[x]$, степень которого не превышает t. Решение данного уравнения $A(z), \Lambda(z), r(z)$ для заданных многочленов z^{2t+1} и $S(z)$ может быть найдено с помощью расширенного алгоритма Евклида. Здесь итерации алгоритма Евклида проводятся до тех пор, пока степень текущего остатка $r(z)$ не станет меньше или равна t. Степень найденного $\Lambda(z)$ равна количеству фактически допущенных при передаче ошибок ν. Если количество корней у $\Lambda(z)$ не совпадает с ν, то выдаётся отказ от декодирования.

Формулировка задания

В задании выдаётся список всех примитивных многочленов степени q над полем $\mathbb{F}_2$ для всех $q = 2, \dots, 16$. В этом списке каждый многочлен представлен десятичным числом, двоичная запись которого соответствует коэффициентам полинома над $\mathbb{F}_2$, начиная со старшей степени.

Для выполнения задания требуется:

1. Реализовать основные операции в поле $\mathbb{F}_2^q$: сложение, умножение, деление, решение СЛАУ, поиск минимального многочлена из $\mathbb{F}_2[x]$ для заданного набора корней из поля $\mathbb{F}_2^q$;

2. Реализовать основные операции для работы с многочленами из $\mathbb{F}_2^q[x]$: произведение многочленов, деление многочленов с остатком, расширенный алгоритм Евклида для пары многочленов, вычисление значения многочлена для набора элементов из $\mathbb{F}_2^q$;

3. Реализовать процедуру систематического кодирования для циклического кода, заданного своим порождающим многочленом;

4. Реализовать процедуру построения порождающего многочлена для БЧХ-кода при заданных n и t;

5. Построить графики зависимости скорости БЧХ-кода $r = k/n$ от количества исправляемых кодом ошибок t для различных значений n. Какие значения t следует выбирать на практике для заданного n?

6. Реализовать процедуру вычисления истинного минимального расстояния циклического кода d, заданного своим порождающим многочленом, путем полного перебора по всем $2^k - 1$ кодовым словам. Привести пример БЧХ-кода, для которого истинное минимальное расстояние больше, чем величина $2t + 1$;

7. Реализовать процедуру декодирования БЧХ-кода с помощью метода PGZ и на основе расширенного алгоритма Евклида. Провести сравнение двух методов декодирования по времени работы;

8. С помощью метода стат. испытаний реализовать процедуру оценки доли правильно раскодированных сообщений, доли ошибочно раскодированных сообщений и доли отказов от декодирования для БЧХ-кода. С помощью этой процедуры убедиться в том, что БЧХ-код действительно позволяет гарантированно исправить до t ошибок. Может ли БЧХ-код исправить больше, чем t ошибок? Как ведут себя характеристики кода при числе ошибок, превышающем t?

9. Составить отчет в формате PDF обо всех проведённых исследованиях.

Рекомендации по выполнению задания

- Для реализации операций умножения и деления ненулевых элементов в поле $\mathbb{F}_2^q$ удобно пользоваться представлением элементов поля как степеней некоторого примитивного элемента α: $\mathbb{F}_2^q = \{0, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2^q-2}, \alpha^{2^q-1} = 1\}$. Тогда произведение двух элементов поля α^{k_1} и α^{k_2} равно $\alpha^{k_1+k_2 \mod 2^q-1}$. Аналогично частное этих двух элементов равно $\alpha^{k_1-k_2 \mod 2^q-1}$. Для быстрого перехода от десятичного представления элементов поля к степенному и обратно удобно завести таблицу размера $(2^q - 1)\times 2$. В первой колонке этой таблицы в позиции i будет находится число $j : \ \alpha^j = i$, а во второй колонке в позиции i – значение α^i.

- При реализации алгоритмов задания рекомендуется, помимо прочего, использовать следующие проверки на корректность:
  - порождающий полином БЧХ-кода должен быть делителем многочлена $x^n - 1$ (иначе код не будет циклическим);
  - произвольное кодовое слово БЧХ-кода $v(x)$ должно делиться без остатка на порождающий многочлен кода $g(x)$, а также обращаться в ноль на нулях кода (все синдромы кодового слова равны нулю);
  - минимальный многочлен $m_\alpha(x)$ для элемента $\alpha \in \mathbb{F}_2^q$, вычисляемый как многочлен с корнями $\alpha, \alpha^2, \alpha^4, \ldots, \alpha^{2^q}$, должен иметь коэффициенты из $\mathbb{F}_2$;
  - минимальное кодовое расстояние БЧХ-кода d, найденное полным перебором, должно быть не меньше, чем величина $2t + 1$.

Оформление задания

Выполненное задание с отчётом и всеми исходными кодами необходимо прислать преподавателю. Далее следует описание прототипов реализуемых функций. При желании студента и при дополнительном согласовании с преподавателем разрешается реализовать задание с другими прототипами с использованием классов PYTHON, перегрузкой операторов, введением дополнительного базового типа в `numpy` и проч. Интересные реализации будут поощряться бонусными баллами.

1. Модуль `gf.py` с реализацией основных операций в конечном поле $\mathbb{F}_2^q$ и операций над многочленами из $\mathbb{F}_2^q[x]$:

   (a) `gen_pow_matrix(primpoly)`
   Описание параметров:

- `primpoly` – примитивный многочлен, десятичное число, двоичная запись которого соответствует коэффициентам полинома над $\mathbb{F}_2$, начиная со старшей степени.

Функция возвращает матрицу соответствия между десятичным представлением и степенным представлением ненулевых элементов поля по стандартному примитивному элементу α, `numpy.array`-матрица размера $2^q - 1 \times 2$, в которой в первой колонке в позиции i стоит степень $j : \alpha^j = i$, а во второй колонке в позиции i стоит значение α^i, $i = 1, \ldots, 2^q - 1$.

(b) `add(X, Y)`
Описание параметров:

- `X, Y` – две матрицы одинакового размера из элементов поля $\mathbb{F}_2^q$, `numpy.array`-матрицы, каждый элемент в матрицах представляет собой десятичное число, двоичная запись которого соответствует коэффициентам полинома над полем $\mathbb{F}_2$, первый разряд соответствует старшей степени полинома;

Функция возвращает `numpy.array`-матрицу размера `X`, являющуюся поэлементным суммированием матриц `X` и `Y`.

(c) `sum(X, axis=0)`
Описание параметров:

- `X` – матрица из элементов поля $\mathbb{F}_2^q$, `numpy.array`-матрица, каждый элемент в матрице представляет собой десятичное число, двоичная запись которого соответствует коэффициентам полинома над полем $\mathbb{F}_2$, первый разряд соответствует старшей степени полинома;

Функция возвращает результат суммирования матрицы `X` по размерности, определяемой параметром `axis`.

(d) `prod(X, Y, pm)`, `divide(X, Y, pm)`
Описание параметров:

- `X, Y` – две матрицы одинакового размера из элементов поля $\mathbb{F}_2^q$, `numpy.array`-матрицы, каждый элемент в матрицах представляет собой десятичное число, двоичная запись которого соответствует коэффициентам полинома над полем $\mathbb{F}_2$, первый разряд соответствует старшей степени полинома;
- `pm` – матрица соответствия между десятичным и степенным представлением в поле $\mathbb{F}_2^q$;

Функции возвращают `numpy.array`-матрицу размера `X`, являющуюся соответственно поэлементным произведением или делением матриц `X` и `Y`.

(e) `linsolve(A, b, pm)`
Описание параметров:

- `A` – квадратная матрица из элементов поля $\mathbb{F}_2^q$;
- `b` – вектор из элементов поля $\mathbb{F}_2^q$;
- `pm` – матрица соответствия между десятичным и степенным представлением в поле $\mathbb{F}_2^q$;

Функция возвращает решение СЛАУ в случае невырожденности `A` и `numpy.nan` иначе.

(f) `minpoly(x, pm)`
Описание параметров:

- `x` – вектор из элементов поля $\mathbb{F}_2^q$;
- `pm` – матрица соответствия между десятичным и степенным представлением в поле $\mathbb{F}_2^q$;

Функция осуществляет поиск минимального полинома в $\mathbb{F}_2[x]$ для набора корней, задаваемых `x`. Функция возвращает кортеж из переменных:

- найденный минимальный полином, `numpy.array`-вектор с бинарными числами;
- все корни минимального полинома (набор корней `x`, а также все смежные с ним), `numpy.array`-вектор из элементов поля $\mathbb{F}_2^q$.

(g) `polyval(p, x, pm)`
Описание параметров:

- `p` – полином из $\mathbb{F}_2^q[x]$, `numpy.array`-вектор коэффициентов, начиная со старшей степени;
- `x` – вектор из элементов поля $\mathbb{F}_2^q$;
- `pm` – матрица соответствия между десятичным и степенным представлением в поле $\mathbb{F}_2^q$;

Функция возвращает значения полинома `p` для набора элементов `x`.

(h) `polyprod(p1, p2, pm)`
Описание параметров:

- `p1`, `p2` – полиномы из $\mathbb{F}_2^q[x]$, `numpy.array`-вектор коэффициентов, начиная со старшей степени;
- `pm` – матрица соответствия между десятичным и степенным представлением в поле $\mathbb{F}_2^q$;

Функция возвращает результат произведения двух полиномов в виде `numpy.array`-вектора коэффициентов, начиная со старшей степени.

(i) `polydivmod(p1, p2, pm)`
Описание параметров:

- `p1`, `p2` – полиномы из $\mathbb{F}_2^q[x]$, `numpy.array`-вектор коэффициентов, начиная со старшей степени;
- `pm` – матрица соответствия между десятичным и степенным представлением в поле $\mathbb{F}_2^q$;

Функция осуществляет деление с остатком многочлена `p1` на многочлен `p2`. Функция возвращает кортеж из переменных:

- частное, `numpy-array`-вектор коэффициентов, начиная со старшей степени;
- остаток от деления, `numpy-array`-вектор коэффициентов, начиная со старшей степени.

(j) `euclid(p1, p2, pm, max_deg=0)`
Описание параметров:

- `p1`, `p2` – полиномы из $\mathbb{F}_2^q[x]$, `numpy.array`-вектор коэффициентов, начиная со старшей степени;
- `pm` – матрица соответствия между десятичным и степенным представлением в поле $\mathbb{F}_2^q$;
- `max_deg` – максимально допустимая степень остатка, число, если равно нулю, то алгоритм Евклида работает до конца;

Функция реализует расширенный алгоритм Евклида для пары многочленов `p1` и `p2`. Функция возвращает кортеж из переменных:

- остаток, `numpy-array`-вектор коэффициентов, начиная со старшей степени;
- коэффициент при `p1`, `numpy-array`-вектор коэффициентов, начиная со старшей степени;
- коэффициент при `p2`, `numpy-array`-вектор коэффициентов, начиная со старшей степени.

2. Модуль `bch.py` с реализацией операций кодирования/декодирования БЧХ кодов:

(a) `coding(U, g)`
Описание параметров:

- `U` – набор исходных сообщений для кодирования, `numpy.array`-матрица, бинарная матрица размера <число_сообщений>$\times k$;
- `g` – порождающий многочлен кода, `numpy.array`-вектор длины $m+1$;

Функция осуществляет систематическое кодирование циклического кода и возвращает `numpy.array`-матрицу с закодированными сообщениями размера <число_сообщений>$\times(k+m)$.

(b) `dist(g, n)`
Описание параметров:

- `g` – порождающий многочлен кода, `numpy.array`-вектор;
- `n` – длина кода, число;

Функция возвращает кодовое расстояние (число), найденное полным перебором.

(c) `genpoly(n, t)`
Описание параметров:

- `n` – длина кода, число;
- `t` – исправляемое число ошибок, число;

Функция строит порождающий многочлен БЧХ-кода по заданным параметрам. Функция возвращает кортеж из переменных:

- порождающий многочлен кода, `numpy.array`-вектор коэффициентов, начиная со старшей степени;
- нули кода, `numpy.array`-вектор десятичных чисел, соответствующих элементам из $\mathbb{F}_2^q$;

- матрица соответствия между десятичным и степенным представлением в поле $\mathbb{F}_2^q$.

(d) `decoding(W, R, pm, method=’euclid’)`
Описание параметров:

- `W` – набор принятых сообщений, `numpy.array`-матрица размера $<$число_сообщений$>\times n$;
- `R` – нули кода, `numpy.array`-вектор;
- `pm` – матрица соответствия между десятичным и степенным представлением в поле $\mathbb{F}_2^q$;
- `method` – алгоритм декодирования, ’euclid’ или ’pgz’;

Функция осуществляет декодирование БЧХ кода и возвращает `numpy.array`-матрицу с декодированными сообщениями размера $<$число_сообщений$>\times n$. В случае отказа от декодирования соответствующая строка матрицы состоит из `numpy.nan`.