

Санкт-Петербургский государственный университет
Прикладная математика и информатика

Кизеев Данил Владимирович
Группа №222

«Построение кодов, исправляющих ошибки»

Курсовая работа

Научный руководитель:
Генералов Александр Иванович
Кафедра высшей алгебры

Санкт-Петербург
2019

Введение

В эпоху информационных технологий возникла необходимость передавать данные на большие расстояния. Не существует идеальных каналов связи, а значит, в передаваемых цифровых последовательностях существует вероятность появления ошибок.

Для борьбы с этими ошибками и были открыты классы кодов, контролирующих ошибки.

Задача кодирования состоит в добавлении к информационным символам дополнительных с целью корректировки помех, возникающих в процессе передачи информационной последовательности. Таким образом, исходная последовательность представляется более длинной последовательностью для корректировки помех.

Дадим определение данных цифровых последовательностей, или двоичных кодов (поскольку все последовательности построены из нулей и единиц).

Определение:

Двоичным кодом мощности M и длины n представляет собой множество из M двоичных слов длины n , называемых кодовыми словами. Обычно $M = 2^k$, где k - некоторое целое число; такой код называется двоичным (n, k) -кодом.

Первой задачей курсовой работы было построить кодер с вылавливанием ошибок для (19437, 19408) кодов Файра с порождающим многочленом

$$g(x) = (x^{19} - 1) \times p(x),$$

где $p(x)$ - примитивный многочлен степени 10.

Вторая задача заключалась в построении эффективных кодов с помощью перемежения и укорочения менее эффективных кодов из таблиц, представленных в книге Ричарда Блейхута "Теория и практика кодов, контролирующих ошибки".
Параметры кодов соответственно:

1. (1072, 1024)-код для исправления пакетов ошибок длины 16 (циклический код).
2. (1080, 1024)-код для исправления пакетов ошибок длины 20 (код Файра)

Каждая из задач была успешно выполнена.

Обе задачи были реализованы в виде компьютерных программ на языке C++.
Использованная среда разработки - CLion 2019.1

Описание выполненных задач

Задача №1

Создание кодера для кодов Файра

Для реализации кодера вылавливающего ошибки, была использована парадигма объектно-ориентированного программирования - для потенциального расширения программы в будущем.

Для облегчения чтения и уменьшения объема отчёта, код был загружен на сайт github.com и доступен по ссылке: github.com/breadfan/Crypto_proj.

Алгебраическая структура задачи

На протяжении всего доклада мы будем рассматривать и работать с кодами, к которым добавляются избыточные проверочные разряды, предназначенные для исправления потенциально возможных ошибок. Такие коды(с избыточностью) называются корректирующими кодами и широко используются в криптографии.

Для помехоустойчивых кодов наиболее лучше подходят так называемые циклические коды. Приведём строгое определение:

Определение:

Линейный код - это код, что множество их кодовых слов образует k -мерное линейное подпространство в n -мерном линейном пространстве, изоморфное пространству n -битовых векторов

Для удобства мы будем рассматривать наши кодовые последовательности в виде полиномов:

$$Q(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0,$$

где $a_i, i = \overline{0, (n-1)}$ - ноль или единица соответственно кодовой последовательности. Данное представление предоставляет нам возможность свести операции с кодами к операциям с многочленами, причём все операции здесь выполняются в конечном поле $GF(2)$, а значит, все операции здесь производятся по модулю два. Для сложения двух полиномов нам нужно просто будет сложить по модулю два коэффициенты при одинаковых степенях. Для умножения же нужно перемножить степени и сложить по модулю два коэффициенты при одинаковых степенях.

Идея построения кодов Файра основана на использовании неприводимых многочленов, то есть таких многочленов, которые нельзя представить в виде произведения многочленов меньших степеней.

Определение:

Кодом Файра называется исправляющий пакеты ошибок циклический код над конечным полем $GF(q)$ с порождающим многочленом

$$g(x) = (x^{2t-1} - 1)p(x),$$

где $p(x)$ - примитивный многочлен над $GF(q)$, степень m которого не меньше длины t исправляемого пакета и который не делит $x^{2t-1} - 1$. Длина n кода Файра равна наименьшему целому n такому, что $g(x)$ делит $x^n - 1$

Также длину можно узнать из уравнения

$$n = c(2^m - 1),$$

где $c = 2t - 1$ Число проверочных разрядов в каждом слове равно

$$r = c + m$$

По условию нам дан порождающий полином $g(x) = (x^{19} - 1) \times p(x)$, где $p(x)$ можно найти из таблицы: это многочлен

$$p(x) = x^{10} + x^3 + 1,$$

тогда

$$g(x) = (x^{19} - 1)(x^{10} + x^3 + 1) = x^{29} + x^{22} + x^{19} + x^{10} + x^3 + 1$$

и длина кода

$$n = 19437$$

Количество информационных символов:

$$k = 19408.$$

. Тогда количество проверочных символов

$$r = n - k = 19437 - 19408 = 29$$

Получим длину исправляемого пакета $c = 2t - 1 = 9$. Тогда $t = 10$.

Осталось проверить достаточность проверочных символов для исправления пакетов длиной $t = 10$: количество проверочных символов должно быть не менее, чем $3t - 1$. При $t = 10$ и $r = 29$ это условие выполняется.

Приступим к разработке *кодера*.

Введём в обозначения операцию

$$R_{b(x)}[a(x)]$$

для многочленов $a(x)$ и $b(x)$ - она будет означать остаток от деления $a(x)$ на $b(x)$
Чтобы закодировать слово, нам нужно представить его в виде

$$c(x) = x^{n-k}i(x) + t(x),$$

где $i(x)$ - исходная информационная последовательность, а $t(x)$ выбирается так, чтобы выполнялось условие

$$R_{g(x)}[c(x)] = 0$$

. Получим, что

$$R_{g(x)}[x^{n-k}i(x)] + R_{g(x)}[t(x)] = 0$$

и

$$t(x) = -R_{g(x)}[x^{n-k}i(x)].$$

$t(x)$ будем называть *синдромом* многочлена. Наконец, нам нужно написать сложить по модулю два $t(x)$ и $x^{n-k}i(x)$ Таким образом, для *кодирования* исходной информационной последовательности нам нужно приписать справа r нулей, разделить на порождающий полином, узнав таким образом остаток, и приписать этот остаток на места добавленных нулей.

Техническая структура задачи

Весь код доступен по ссылке: github.com/breadfan/Crypto_proj

Программа была написана языке программирования C++ с использованием парадигмы объектно-ориентированного программирования, а именно: был создан класс "Polynomial" для расширения возможностей программы в будущем - создания большего множества объектов

Задача №2:

Программа была написана на языке C++.

Для реализации эффективных кодов был использован ранее написанный класс "Polynomial".

Были задействованы перемежение и укорочение циклических кодов.

Алгебраическая структура задачи

1. Построение первого кода с параметрами (1072, 1024) для исправления пакетов ошибок длины 16. Построить данный код предлагалось, выбрав за основу один из кодов, приведённых в таблице 5.1 книги Р.Блейхута. Подбором было выяснено, что подходящим является код с параметрами (511, 499), длиной исправляемого пакета, равной 4 и порождающим многочленом

$$x^{12} + x^8 + x^5 + x^3 + 1.$$

Теперь определим понятия *укороченных* циклических кодов и *перемежения* их же.

Любой циклический код можно **укоротить**, т.е. от (n, k) кода перейти к $(n - b, k - b)$ кода просто выбрасыванием b информационных позиций в каждом кодовом слове. Будем полагать, что выбрасываются b старших разрядов кода и $b < k$. Сами выбрасываемые символы положим равными нулю и передавать не будем. На декодере это никак не скажется - он будет работать по полной кодовой последовательности, в начале своей работы полностью её восстановив.

Важно заметить, что укороченный циклический код уже не является циклическим.

Чтобы из (n, k) -кода получить (jn, jk) -код, нужно выбрать из исходного кода j произвольных слов и укрупнить кодовые слова, чередуя их символы. Так, получим, если исходный код исправлял пакет ошибок длиной t , то укрупнённый код будет исправлять пакеты ошибок длиной jt . Чтобы получить код с помощью перемежения, нужно заменить порождающий многочлен $g(x)$ исходного кода на $g(x^j)$ и пересчитать многочлен.

В нашем случае код был сокращен на $b = 243$ до кода (268, 256), а далее перемежением модифицирован до (1072, 1024)-кода

2. Построение второго кода с параметрами (1080, 1024) для исправления пакетов ошибок длины 16. Не найдя подходящего кода в таблице 5.1, я обратился к следующей таблице - 5.2 и нашёл там идеально подходящий код. Этим кодом

являлся (279, 265)-код, исправляющий пакеты ошибок длиной 5. Он был укорочен на $b = 9$ до (270, 259)-кода, а далее перемежением модифицирован до (1080, 1024)-кода.

Техническая структура задачи

Для удобства работы с большими кодовыми последовательностями, в качестве контейнера для кодов используется псевдоконтейнер `std::string`. Сами коды задаются с помощью метода `random()` из библиотеки `<iostream>`, также доступен ввод с клавиатуры.