1. Вступ 2

2. Постановка задачі 3

3. Початкові дані 4

4. Розв’язання по розділам 5

4.1 Найпростіші коди 5

4.1.1. Двійково-десятковий код 5

4.1.2 Код Грея 7

4.3 Статистичне кодування 8

4.3.1 Код Хаффмана 9

4.3.2 Код Шеннона-Фано 11

4.4 Коди, що виявляють помилки 13

4.4.1 Код із перевіркою на парність 13

4.4.2 Код із перевіркою на непарність 14

4.4.3 Інверсний код 15

4.4.4 Кореляційний код 16

4.4.5 Код Бергера 17

4.4.6 Код на одне сполучення 18

4.4.7 Код із кількістю одиниць у комбінації, кратною трьом 19

4.5 Коди, що виправляють помилки 20

4.5.1 Код Варшамова в матричному представленні 20

4.5.2 Код Хеммінга 22

4.5.3 Розширений код Хеммінга 24

4.5.4 Ітеративний код 25

4.5.5 Коди-супутники 27

4.5.6 Циклічний код з 28

4.5.7 Коди Боуза-Чоудхурі-Хоквінгема 30

4.5.8 Рекурентний код 32

4.6 Канальні коди 33

4.6.1 Дуобінарний код 33

4.6.2 Квазітрійковий код 33

4.6.3 Код Манчестер ІІ 33

4.6.4 Код 4В3Т 34

4.5 Штрихові коди 35

5. Висновки 36

Додатки 37

Додаток А. Лістинг програми 37

Додаток Б. Таблиця ймовірностей 40

Додаток В. Кодове дерево для коду Хаффмана 41

Додаток Г. Список використаної літератури 42

1. Вступ

В даній роботі будуть наведені основні відомості про найбільш уживані коди, що застосовуються у різних областях техніки. Опис кожного коду супроводжується одним або кількома прикладами.

Особливу увагу приділено БЧХ-кодам — написана програма для кодування комбінації будь-якої довжини з можливістю виправлення будь-якої кількості помилок саме цим кодом.

1. Постановка задачі
2. Вхідні дані: ПІБ студента (ПІБ – три перші великі літери українського алфавіту) – переводимо комбінацію літер в 24 біта ASCII отримуємо ПІБ в цифровому коді.
3. Найпростіші коди. Кодувати і декодувати ПІБ наступними кодами:
   1. Двійково-десяткові(два різних коди по вазі). Розрахувати збитковість.
   2. Код Грея. Показати зменшення ваги помилки.
4. Статистичне кодування.
   1. Для тексту, складеного з повного запису прізвища, ім’я та по-батькові студента та його батьків, порахувати ймовірність появи літер. Підрахувати кількість інформації в символах, ентропію джерела.
   2. Закодований в п 1 алфавіт кодами:
      * Шеннона-Фано
      * Хаффмана. Побудувати кодове дерево.
   3. Довести розрахунками оптимальність кодів.
   4. Закодувати в нових алфавітах ПІБ
5. Коди, які знаходять помилки з підрахунком збитковості. Показати процедуру кодування і декодування ПІБ наступними кодами:
   1. З перевіркою на парність
   2. З перевіркою на непарність
   3. Інверсний код
   4. Кореляційний код
   5. Код Бергера
   6. Код на одну комбінацію
   7. Код з кількістю одиниць в кодових комбінаціях, кратним трьом.
6. Коди, які виправляють помилки: закодувати, внести помилку і виправити її при декодування.
   1. Код Варшамова в матричному вигляді
   2. Код Хеммінга (дві перших букви ПІБ)
   3. Розширений код Хеммінга
   4. Ітеративний код
   5. Коди-супутники (для першої букви ПІБ і dm=-1.2)
   6. Циклічний код з dm=3
   7. БЧХ
   8. Рекурентний
7. Канальні коди
   1. Дуобінарний
   2. Квазітрійковий
   3. Манчестер ІІ
   4. 4В3Т
8. Закодувати в графічному вигляді штрих-код EAN-8 десяткового числа, який містить послідовно число, місяць і рік народження.
9. Розробити програму роботи кодера і декодера для БЧХ коду на будь-якій алгоритмічній мові програмування.
10. Початкові дані

Три перші великі літери прізвища, імені та по-батькові в українському алфавіті

ГДЄ

Переводимо комбінацію ГДЄ в 24 біта ASCII . Одержана комбінація:

Текст для кодів Хаффмана і Шеннона-Фано

ГЛИНСЬКИЙДАНИЛОЄВГЕНОВИЧ

ГЛИНСЬКААЛСІАЄВГЕНІВНА

ГЛИНСЬКИЙЄВГЕНМИКОЛАЙОВИЧ

ГЛИНСЬКАТЕТЯНАІВАНІВНА

1. Розв’язання по розділам

4.1 Найпростіші коди

4.1.1. Двійково-десятковий код

У двійково-десятковому коді кожний розряд десяткового числа записується у вигляді чотиризначного двійкового числа, що дає змогу сформувати комбінацій. Оскільки у десятковій системі числення використовується цифр, комбінацій є надмірними.

Ваговим називається такий двійково-десятковий код, в якому вага кожного розряду для всіх комбінацій залишається сталою. Побудова такого ДДК виконується за правилом:

* Вага найменшого розряду дорівнює
* Вага другого за мінімальний значенням розряду може дорівнювати або
* Вага, що відповідає останнім двом розрядам та коду підбирається так, щоб сума їх була більшою або дорівнювала () або 7 ().

Користуючись таким правилом, можна дістати 17 варіантів ДДК з додатною вагою розрядів.

Кожна десяткова цифра може бути задана однією з комбінацій чотирьох двійкових елементів : , де — заданий ДДК.

Всі ДДК, крім коду , не мають однозначності у запису цифр (одна і та ж цифра може бути представлена кількома різними тетрадами). Всі ДДК мають надмірність (крім робочих комбінацій є 6 надлишкових), яка може застосовуватись для виявлення помилок. Проте мінімальна кодова відстань цього коду дорівнює , і тому код не може виявляти навіть усі однократні помилки.

Розрахуємо надмірність ДДК. Кількість дозволених комбінацій: , повна кількість кодових комбінацій: . Тоді надмірність ДДК:

*Вхідна послідовність*: =

**Кодування**

Закодувати число кодом 3-4-1-1:

Закодувати число кодом 4-2-2-1:

**Декодування**

Початкова послідовність : 00011011000111111100010010000111. Код 3-4-1-1

Отже, закодована послідовність 15197436.

Початкова послідовність :00011001000111111101100000111100. Код 4-2-2-1

Отже, закодована послідовність 15197436.

Результат свідчить про те, що для різних кодових комбінацій ми отримуємо різні коди, які при декодуванні утворюють однакову послідовність десяткових цифр.

4.1.2 Код Грея

Код Грея належить до рефлексивних кодів, які мають такі властивості:

* Кодова відстань між сусідніми кодовими комбінаціями дорівнює одиниці, тобто вони різняться цифрою тільки в одному розряді.
* Зміна елементів в кожному розряді при переході від комбінації до комбінації відбувається в два рази рідше, ніж у простому коді, завдяки чому значно спрощується кодер. При додаванні за модулем двох сусідніх кодових комбінацій кількість одиниць у одержаній сумі дорівнює кількості розрядів мінус .

**Кодування**

*Початкова послідовність*: . Кодування здійснюється шляхом додавання за модулем без переносів початкової комбінації, зсунутої відносно першої на один розряд праворуч:

Тоді закодована послідовність:

**Декодування**

Цифру старшого розряду залишаємо без змін: Наступні цифри одержуємо за правилом . Тоді декодована послідовність:

, що співпадає з початковою.

Зменшення ваги помилки

Початкова послідовність: . Внесемо помилку в 4-ий розряд: . Закодуємо кодом Грея початкову послідовність:

І внесемо помилку в той самий розряд:

Декодуємо цю послідовність і розрахуємо її вагу:

Видно, що в даному випадку зменшення ваги помилки не відбулося.

4.3 Статистичне кодування

Оптимальні коди — це коди, в яких досягнуто максимально можливого інформаційного навантаження на один символ джерела повідомлень. Оскільки характеристикою джерела є його ентропія, то в оптимальних кодах ступінь ефективності оцінюють ентропією джерела. Звичайно природні джерела мають інформаційну надмірність, оскільки для представлення заданого об’єму (кількості) інформації в них використовується більша кількість символів, ніж можна очкувати, виходячи з властивостей джерела.

Для розрахунку було обрано текст, складений з прізвищ, імен та по-батькові моїх батьків. Припускалось, що повідомленнями є лише літери алфавіту (тобто не враховувались розділові знаки). Великі та малі літери вважались однаковими. Статистична імовірність появи кожної літери розраховувалась по формулі

Де — кількість даної літери у тексті, — кількість різних літер.

Результати приведені у таблиці

|  |  |  |  |
| --- | --- | --- | --- |
| Символ а | Р(а) | Символ а | Р(а) |
| Н | 0.12903225806451613 | Е | 0.043010752688172046 |
| И | 0.10752688172043011 | І | 0.043010752688172046 |
| А | 0.10752688172043011 | Й | 0.03225806451612903 |
| В | 0.08602150537634409 | Є | 0.03225806451612903 |
| Д | 0.07526881720430108 | Ч | 0.021505376344086023 |
| Г | 0.07526881720430108 | Т | 0.021505376344086023 |
| С | 0.053763440860215055 | Я | 0.010752688172043012 |
| К | 0.053763440860215055 | М | 0.010752688172043012 |
| Ь | 0.043010752688172046 | Д | 0.010752688172043012 |
| О | 0.043010752688172046 |  |  |

4.3.1 Код Хаффмана

Код Хафмана є нерівномірним оптимальним кодом. Нерівномірність означає, що кодові комбінації містять неоднакову кількість цифр — більш імовірним повідомленням відповідають короткі комбінації, менш імовірним — довгі. Оптимальність означає, що символи і у повідомленнях є приблизно рівномірними.

Алгоритм побудови коду Хафмана

Всі повідомлення джерела, що підлягають кодуванню, записуються у стовпець в порядку спадання ймовірностей. Потім два найменш імовірних повідомлення об’єднуються в одне, якому приписується сумарна імовірність. Далі формується перша допоміжна група елементів, яка відрізняється від початкової тим, що об’єднаний елемент займає у стовпці місце, що відповідає його імовірності. Процес об’єднання пари найменш імовірних елементів і побудови наступної допоміжної групи повторюється до одержання єдиного елемента з імовірністю .

За таблицею допоміжних груп шукають кодові комбінації, що відповідають символам алфавіту джерела. Для цього будують кодове дерево. Відмічають вершину кодового дерева і приписують їй імовірність . З вершини проводять дві гілки, що закінчуються вузлами. Вузлам приписують імовірності, які при додаванні дають . Горизонтальній гілці (з меншою імовірністю) приписують символ «», а вертикальній (з більшою імовірністю) — символ «». Якщо імовірність при деякому вузлі є імовірністю деякого повідомлення, то вузлу приписують символ цього повідомлення і переходять до аналізу наступного вузла. Якщо імовірність при деякому вузлі утворена додаванням двох менших ймовірностей, то з нього знову проводять дві гілки, що закінчують вузлами. Вузлам приписують символи: вертикальній — «», горизонтальній — «». Процес побудови дерева продовжується , доки всі символи не будуть приписані вузлам. Для знаходження кодових комбінацій треба пройти по гілках шлях від кореневого вузла (вузла з імовірністю) до вузла, що відповідає заданому повідомленню, послідовно записуючи зліва направо символи та , які відповідають гілкам, що ввійшли у шлях.

**Кодування**

Виконувалось кодування джерела, заданого ансамблем повідомлень з попереднього розділу. Побудова таблиці допоміжних груп та кодового дерева здійснювалась за допомогою програми. Ці таблиці наведені у додатку А і Б.

Одержані кодові комбінації коду Хаффмана зведені у таблицю.

|  |  |  |  |
| --- | --- | --- | --- |
| И | 000 | Ч | 01101 |
| А | 001 | Ь | 10000 |
| Н | 111 | Е | 10001 |
| К | 0100 | О | 11000 |
| С | 0101 | Й | 11001 |
| І | 0111 | Є | 11011 |
| В | 1001 | Я | 110101 |
| Г | 1010 | М | 1101000 |
| Л | 1011 | Д | 1101001 |
| Т | 01100 |  |  |

Оцінимо оптимальність цього коду, прийнявши мірою оптимальності ентропію. Значення ентропії первинного джерела:

Значення ентропії вторинного джерела:

Ентропія Н вторинного джерела перевищує ентропію первинного джерела. Це свідчить про більшу оптимальність побудованого коду порівняно з початковим, тобто більшу кількість інформації на один символ нового алфавіту.

Закодуємо комбінацію ГДЄ одержаним кодом:

4.3.2 Код Шеннона-Фано

Алгоритм побудови коду Шеннона-Фано для двійкового вторинного алфавіту

1. Повідомлення та їх імовірності записуються в таблицю в порядку спадання ймовірностей
2. Таблиця ділиться на дві частини так, щоб суми ймовірностей обох частин були приблизно рівними. Якщо виходить два варіанти розбиття, для яких суми ймовірностей відрізняють на однакову величину, то різне для них повідомлення вноситься до верхньої підгрупи.
3. В верхню під таблицю у якості старшого біта вноситься , в нижню — .
4. Ділення під таблиць згідно пункту 2 проводиться рекурсивно до отримання кінцевих кодових комбінацій, тобто доки кожна під таблиця не буде містити лише одне повідомлення.

**Кодування**

Виконувалось кодування джерела, заданого ансамблем повідомлень. Результат кодування представлений у таблиці нижче:

|  |  |  |
| --- | --- | --- |
| Н | 0.12903225806451613 | 0 0 |
| И | 0.10752688172043011 | 0 1 0 |
| А | 0.10752688172043011 | 0 1 1 |
| В | 0.08602150537634409 | 1 0 0 0 |
| Д | 0.07526881720430108 | 1 0 0 1 |
| Г | 0.07526881720430108 | 1 0 1 0 |
| С | 0.053763440860215055 | 1 0 1 1 |
| К | 0.053763440860215055 | 1 1 0 0 0 |
| Ь | 0.043010752688172046 | 1 1 0 0 1 |
| О | 0.043010752688172046 | 1 1 0 1 0 |
| Е | 0.043010752688172046 | 1 1 0 1 1 |
| І | 0.043010752688172046 | 1 1 1 0 0 |
| Й | 0.03225806451612903 | 1 1 1 0 1 |
| Є | 0.03225806451612903 | 1 1 1 1 0 0 |
| Ч | 0.021505376344086023 | 1 1 1 1 0 1 |
| Т | 0.021505376344086023 | 1 1 1 1 1 0 |
| Я | 0.010752688172043012 | 1 1 1 1 1 1 0 |
| М | 0.010752688172043012 | 1 1 1 1 1 1 1 0 |
| Д | 0.010752688172043012 | 1 1 1 1 1 1 1 1 |

Оцінимо оптимальність цього коду, прийнявши мірою оптимальності ентропію. Значення ентропії первинного джерела:

Значення ентропії вторинного джерела:

Закодуємо комбінацію ГДЄ одержаним кодом:

4.4 Коди, що виявляють помилки

Особливість кодів, які виявляють помилки, полягає в тому, що кодові комбінації, які входять до складу цих кодів, різняться кодовою відстанню, не меншою ніж .

4.4.1 Код із перевіркою на парність

Це найпоширеніший код, який застосовується для виявлення поодиноких помилок і всіх помилок непарної кратності. Код містить інформаційних та один перевірний елементи, належить до систематичних кодів.

Перевірний елемент коду визначається сумою за модулем 2 всіх інформаційних елементів: , тобто він утворюється доповненням комбінації -елементного первинного коду одним елментом таким чином, щоб кількість одиниць у новому -розрядному коді була парною. Кодова відстань .

Для виявлення помилки на приймальному боці перевіряють на парність усю прийняту кодову комбінацію, визначаючи кодовий синдром

де— прийняті на приймальному боці відповідно інформаційні та перевірний елементи. Вважається, що при помилки в комбінації немає, а при помилка є. Надмірність коду визначається виразом

**Кодування**

*Початкова послідовність*: , перевірний елемент . Тоді закодована комбінація —

Надмірність коду:

**Декодування**

*Прийнята комбінація* — , кодовий синдром для неї , отже помилка присутня.

4.4.2 Код із перевіркою на непарність

Цей код відрізняється від попереднього тим, що кожна його комбінація має непарну кількість одиниць, тобто додатковий перевірний елемент формують, виходячи з кількості одиниць у початковій кодовій комбінації: при парній кількості перевірний елемент дорівнює одиниці, а при непарній — нулю.

Для виявлення помилки в кодовій комбінації на приймальному боці її перевіряють на непарність. Код є подільним завдовжки інформаційних та один перевірний елементи; він може так само виявляти помилки та має надмірність, як і код з перевіркою на парність.

**Кодування**

*Початкова послідовність*: . Перевірний елемент . Тоді закодована комбінація

Надмірність коду:

**Декодування**

*Прийнята комбінація* — , кількість одиниць у ній непарна, отже помилок немає.

4.4.3 Інверсний код

Інверсний код (із повторенням та інверсією) є подільним лінійним кодом, який має інформаційних і стільки ж перевірних елементів. Його відмінність від попереднього коду полягає в тому, що значення перевірних елементів у ньому залежать від значення суми за модулем всіх інформаційних елементів. За умови , тобто при парній кількості одиниць у початковій кодовій комбінації, перевірні елементи просто повторюють інформаційні , а за умови , тобто при непарній кількості зазначених одиниць, перевірні елементи повторюють інформаційні в інвертованому вигляді (в оберненому коді): .

Надмірність коду визначається виразом

**Кодування**

*Початкова послідовність*: . Кількість одиниць у комбінації парна, отже перевірні елементи повторюють інформаційні, і закодована послідовність має вигляд:

**Декодування**

*Прийнята комбінація* —

Розбиваємо прийняту послідовність навпіл і підраховуємо кількість одиниць у першій половині. Оскільки вона є непарною, то порівнюємо поелементно першу і другу частини інвертуючи другу:

,

Оскільки послідовності не збігаються, отже були присутні помилки.

4.4.4 Кореляційний код

У цьому коді кожний розряд двійкового початкового коду записується у вигляді двох елементів: як , а – як . Так, початковій кодовій комбінації відповідатиме комбінація кореляційного коду. В технічній літературі такий запис дуже часто називається Манчестер-кодом.

Приймальний пристрій в кожному такті, що складається з двох сусідніх елементів кореляційного коду, має зафіксувати перехід або . У разі прийняття двох нулів або одиниць приймальний пристрій фіксує наявність помилки.

Кореляційний код дає змогу виявляти помилки будь-якої кратності, але не здатний виявити двократні дзеркальні помилки, коли сусідні елементи одного такту під впливом завад змінюються на протилежні за значенням.

Надмірність коду визначається виразом

До переваг кореляційного коду, крім відсутності постійної складової в напрузі кодового сигналу при передачі кодової комбінації по каналу зв’язку,можна віднести також можливість самосинхронізації генератора приймача, оскільки прийняття кожного біта супроводжується фронт сигналу, що приймається, в центрі біта.

**Кодування**

*Початкова послідовність*: . Закодована послідовність

**Декодування**

*Прийнята послідовність*: . Декодування здійснюється шляхом заміни 01 на 0 і 10 на 1. Тоді декодована комбінація:

.

4.4.5 Код Бергера

Код Бергера належить до несистематичних кодів. Існує декілька варіантів побудови цього коду. В найбільш простому варіанті кодування здійснюється наступним чином: в інформаційній частині коду підраховується число одиниць, і перевірні розряди є інвертованим записом цього числа у двійковій формі. Отже кількість перевірних розрядів дорівнює найменшому цілому числу, що перевищує . Коди Бергера призначені для використання в асиметричних каналах зв’язку, де можливе або тільки перетворення нулів в одиниці, або навпаки. Код Бергера може виявляти всі поодинокі та деякі багатократні помилки.

**Кодування**

*Початкова послідовність*: . Умова, накладена на кількість перевірних розрядів: , отже . Кількість одиниць дорівнює . Інвертований запис цього числа — . Закодована послідовність:

Надмірність коду

**Декодування**

*Прийнята послідовність* — , кількість одиниць у інформаційній частині — . Перевірні розряди мають вигляд , що не співпадає з перевірними розрядами, отже послідовність передана неправильно.

4.4.6 Код на одне сполучення

Код на одне сполучення — це код з незмінною кількістю одиниць і нулів у комбінаціях. Кількість комбінацій цього коду визначається виразом , де — кількість одиниць у кодовій комбінації завдовжки .

Такий код утворюється з двійкового простого коду відбором комбінацій, що мають однакову кількість одиниць . Приймальний пристрій, підраховуючи кількість одиниць у прийнятій кодовій комбінації, виявляє помилки, якщо кількість перших відрізнятиметься від m.

Код зі сталою вагою має мінімальну кодову відстань . Він виявляє всі помилки непарної кратності, а також усі помилки парної кратності, що призводять до порушення умови . Надмірність коду визначається виразом .

Порівняно з кодом із простим повторенням цей код при меншій його надмірності дає змогу виявляти помилки тієї самої кратності.

**Кодування**

Потрібно закодувати кодом на одне сполучення комбінацію ГДЄ (три символи). Для цього необхідно вибрати такий код зі сталою вагою, у якому кількість кодових комбінацій не менша, ніж 32 (за кількістю букв алфавіту). Такій умові задовольняє код з , оскільки для нього кількість кодових комбінацій дорівнює . Тоді можна поставити у відповідність:

.

Закодована комбінація набуває вигляду

**Декодування**

*Прийнята послідовність*: . Для кожного боку з символів виконується умова , тому робимо припущення, що послідовність прийнято вірно.

4.4.7 Код із кількістю одиниць у комбінації, кратною трьом

Цей код можна утворити або додаванням до кожної комбінації початкового коду перевірних елементів, або зменшенням кількості дозволених комбінацій початкового коду з накладанням додаткової умови: кількість одиниць у кожній комбінації має бути кратною трьом.

У першому випадку до початкової кодової комбінації додаються два перевірних розряди, які мають такі значення, що сума одиниць у кодовій комбінації стає кратною трьом. Так, комбінація початкового коду, закодована кодом з кількістю одиниць кратною трьом, матиме вигляд .

У другому випадку з усіх комбінацій початкового коду вибирають тільки ті, які мають вагу та . Решту комбінацій використовувати не можна.

Код дає змогу виявити всі поодинокі помилки та деякі помилки більшої кратності, що призводять до порушення умови або , де — кількість одиниць у кодовій комбінації.

Надмірність коду з доповненням до необхідної кількості одиниць визначається виразом , а коду, що утворюється відбором із загальної кількості комбінацій х відповідною кількістю одиниць ( або ), — виразом

**Кодування**

Початкова послідовність: . Кількість одиниць у послідовності , отже її необхідно доповнити розрядами . Закодована послідовність: . Надмірність коду: .

**Декодування**

Перевіримо кількість одиниць у прийнятій комбінації на кратність трьом. Оскільки 16 не ділиться на 3, то вважаємо, що послідовність прийнята невірно.

4.5 Коди, що виправляють помилки

4.5.1 Код Варшамова в матричному представленні

Код Варшамова належить до систематичних кодів. Це означає, що сума за модулем двох дозволених кодових комбінацій теж є дозволеною кодовою комбінацією. Серед інших властивостей слід зазначити такі:

Основою систематичного коду є твірна матриця . Це таблиця, що містить стовпців та рядків, елементи матриці є символами алфавіту.

Матриця може бути зведена до канонічного трапецієвидного вигляду: **,** де — одинична матриця, —матриця коефіцієнтів. Отже, не потрібно одразу при побудові коду знаходити дозволених кодових комбінацій. Достатньо знайти векторів довжиною , які є рядками матриці .

|  |  |
| --- | --- |
| Перевірна матриця систематичного коду — таблиця, що містить символи з алфавіту коду (множина ). Її елементи підбираються так, що рядки є ортогональними рядкам **.**  представляється у вигляді двох під матриць:  **.**  З побудови матриці випливає, що її добуток на будь-яку транспоновану дозволену кодову комбінацію дає вектор-стовпець нулів:  (фундаментальне рівняння систематичного коду).  **Кодування**  *Початкова послідовність*: . Потрібно побудувати код, здатний виправляти одну помилку при передачі повідомлень з інформаційних символів. |  |

Тоді мінімальна кодова відстань . З умови знаходимо . Сформуємо матрицю (на попередній сторінці).

За цією матрицею можна побудувати матрицю:

Запишемо комбінацію, яку потрібно закодувати, з перевірними символами в загальному вигляді:

Згідно з фундаментальним рівнянням систематичного коду . Тому, помноживши кожний рядок на і прирівнявши добутки до нуля, отримаємо систему рівнянь відносно :

Тоді закодована комбінація має вигляд:

**Декодування**

*Нехай прийнято послідовність* . Припускається, що в ній може бути одна помилка. Знайдемо кодовий синдром:

.

Оскільки синдром містить ненульові компоненти, то мала місце похибка при передачі повідомлення. Цей синдром співпадає з -тим стовпцем матриці . Тому потрібно інвертувати -й символ в прийнятій комбінації. Таким чином, отримано виправлену комбінацію:

4.5.2 Код Хеммінга

Код Хеммінга належить до систематичних кодів з кодовою відстанню , що виправляють всі поодинокі помилки або виявляють подвійні

При передачі кодового вектора може бути спотворений будь-який елемент, кількість таких ситуацій . До цього слід додати ще одну ситуацію, коли помилка не виникає. Таки чином, загальна кількість комбінацій перевірних елементів має перевищувати кількість можливих помилкових ситуацій в коді з урахуванням відсутності помилок для правильного розрізнення їх і визначення місця помилки:

Характерна особливість перевірної матриці коду з полягає в тому, зо її стовпці є різними ненульовими комбінаціями завдовжки . Таким чином, якщо взяти комбінації - елементного двійкового коду й відкинути нульову комбінацію, можна досить легко дістати перевірну матрицю, записавши всі кодові комбінації послідовно в стовпці матриці .

Хеммінг запропонував розташувати стовпці перевірної матриці та, щоб номер -го стовпця матриці і номер розряду кодової комбінації відповідали двійковому поданню числа . Тоді синдром, знайдений з перевірних рівнянь, буде двійковим поданням номера розряду кодової комбінації, в якій виникла помилка. Для цього перевірні розряди мають знаходитись не в кінці кодової комбінації, а на номерах позицій, які подаються степенем двійки, тому що кожний з них входить тільки до одного з перевірних рівнянь. В останньому випадку перевірні розряди розміщуються між інформаційними.

**Кодування**

*Потрібно закодувати послідовність* з інформаційних символів. З умови знаходимо кількість перевірних символів . Складемо перевірну матрицю :

Запишемо комбінацію, яку потрібно закодувати, з перевірними символами в загальному вигляді:

Згідно з фундаментальним рівнянням систематичного коду . Тому Помноживши кожний рядок на і прирівнявши добутки до нуля, отримаємо систему рівнянь відносно :

Тоді закодована комбінація має вигляд:

**Декодування**

*Нехай прийнято послідовність* . Припускається, що в ній може бути одна помилки. Знайдемо кодовий синдром:

Оскільки синдром містить ненульові компоненти, то мала місце похибка при передачі повідомлення. Враховуючи, що синдром є двійковим записом номеру позиції з помилкою і , то виправлення помилки полягає у інвертування елемента на першій позиції:

4.5.3 Розширений код Хеммінга

Розширений код Хеммінга можна утворити з коду Хеммінга, в якого кодова відстань . Для цього кодову комбінацію останнього просто доповнюють додатковим перевірним елементом , який знаходять за допомогою перевірки кодової комбінації на парність. При цьому перевірний елемент має дорівнювати одиниці, якщо кількість одиниць в закодованій комбінації непарна, й нулю, якщо ця кількість парна.

Розширений код Хеммінга декодують у зворотній послідовності: спочатку виконують загальну перевірку прийнятої кодової комбінації, а потім — її перевірку без елемента .

З урахуванням викладеного коди Хеммінга з використовуються, як правило для виявлення одно -, двох - та трикратних помилок.

**Кодування**

Нехай необхідно *закодувати послідовність* розширеним кодом Хеммінга. В попередньому пункті для неї було отримано код Хеммінга . Доповнимо його перевірним елементом. Оскільки кількість одиниць у комбінації коду Хеммінга непарна (), то перевірний елемент дорівнює . Тоді результат кодування має вигляд:

**Декодування**

*Нехай прийнято послідовність* . Кодовий синдром перевірки на парність (кількість одиниць непарна). Знайдемо кодовий синдром без елемента .

Отримано ненульовий синдром . Оскільки перший кодовий синдром теж був ненульовим, то в повідомленні була однократна помилка, яку виправляємо інвертуванням -го символу: .

4.5.4 Ітеративний код

Ітеративні коди характеризуються двома або більшою кількістю перевірок усередині кодової комбінації, а властивості цих кодів повністю визначаються їх параметрами.

Так, довжина кодової комбінації, кількість інформаційних параметрів та мінімальна кодова відстань визначаються виразами.

де — параметри кодів; — кратність ітерацій.

На практиці широко застосовуються двовимірні лінійні ітеративні коди з кодуванням за рядками та стовпцями з однією перевіркою на парність. Дозволяється використовувати коди з кількістю перевірних елементів розміром (з рядками та стовпцями). При цьому .

Ці коди мають мінімальну кодову відстань і дають змогу виявити помилки будь-якої кратності, за винятком деяких чотири -, шести - та восьмикратних помилок, якщо вони розміщуються в вершинах прямокутників або попарно в певному порядку. В режимі виправлення та виявлення помилок код виправляє будь-які поодинокі помилки і виявляє всі подвійні та деякі помилки більшої кратності.

Надмірність двовимірних ітеративних кодів становить:

.

Суттєвим недоліком ітеративних кодів є порівняно висока надмірність їх, яка значно перевищує надмірність циклічних кодів, здатних виявляти та виправляти ту саму кількість помилок за інших однакових умов. Однак їх використання в системах передач даних зумовлює більш просте порівняно з циклічними кодами кодування та декодування за допомогою ЕОМ.

Ітеративні коди знайшли широке застосування для виявлення та виправлення помилок, які виникають при запису, зберіганні та зчитуванні цифрової інформації на магнітних носіях.

**Кодування**

*Початкова послідовність*: . Послідовно запишемо символи у матрицю і перевіримо на парність кожен рядок та стовпець:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |

Таким чином, закодована послідовність має вигляд .

**Декодування**

*Нехай на декодер надійшла послідовність* , закодована ітеративним кодом з . Декодуємо її:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 0 | ***1*** | 0 |
| 1 | 1 | 1 | 1 | 1 | ***0*** | 1 |
| 0 | 1 | 0 | 0 | 1 | ***1*** | 1 |
| ***1*** | ***1*** | ***1*** | ***1*** | ***0*** | ***1*** | ***0*** |
| 1 | 0 | 1 | 0 | 0 | ***0*** | 0 |

З перевірних рядків та стовпців прийнятої комбінації на парність випливає, що елемент на перетині четвертого рядка та шостого стовпця переданий невірно, тому для виправлення його потрібно інвертувати. Декодована комбінація, звільнена від перевірних елементів набуває вигляду:

.

4.5.5 Коди-супутники

Для кожної дозволеної кодової комбінації записують усі вектори, відстань від яких до даної комбінації не перевищує деякого заданого значення, — коди-супутники. Якщо прийнята комбінація співпадає з одним з кодів-супутників, то її декодують як . При цьому важливо, щоб множини кодів-супутників не перетинались.

**Кодування**

Складемо таблицю кодів-супутників з для вектора

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 00000001 | 11100110 | 00100010 | 11000101 |
| 00000010 | 11100101 | 01000010 | 10100101 |
| 00000100 | 11100011 | 10000010 | 01100101 |
| 00001000 | 11101111 | 00001100 | 11101011 |
| 00010000 | 11110111 | 00010100 | 11110011 |
| 00100000 | 11000111 | 00100100 | 11000011 |
| 01000000 | 10100111 | 01000100 | 10100011 |
| 10000000 | 01100111 | 10000100 | 01100011 |
| 00000011 | 11100100 | 00011000 | 11111111 |
| 00000101 | 11100010 | 00101000 | 11001111 |
| 00001001 | 11101110 | 01001000 | 10101111 |
| 00010001 | 11110110 | 10001000 | 01101111 |
| 00100001 | 11000110 | 00110000 | 11010111 |
| 01000001 | 10100110 | 01010000 | 10110111 |
| 10000001 | 01100110 | 10010000 | 01110111 |
| 00000110 | 11100001 | 01100000 | 10000111 |
| 00001010 | 11101101 | 10100000 | 01000111 |
| 00010010 | 11110101 | 11000000 | 00100111 |

**Декодування**

*Нехай прийнято послідовність* . За таблицею кодів-супутників знаходимо відповідний вектор помилки . Додамо його до прийнятої послідовності. Одержана комбінація:

4.5.6 Циклічний код з

Циклічний коди є різновидом систематичних кодів. Подання комбінації в них виконують у вигляді поліномів формальної змінної , що дає змогу звести дії над кодовими комбінаціями до дій над поліномами.

Основна властивість циклічних кодів — циклічний зсув дозволеної комбінації також є дозволеною комбінацією. Якщо комбінація представлена поліномом, то циклічна перестановка виникає після множення полінома на .

Особливу роль у теорії циклічних кодів відіграють твірні поліноми. Як твірні поліноми можуть використовуватись всі незвідні поліноми та їх добутки, оскільки вони є дільниками двочлена . Кожна комбінація даного циклічного - коду ділиться на деякий твірний поліном степені , який є дільником двочлена .

Алгоритм кодування

1. Подати інформаційну частину з елементів у вигляді полінома степеня ;
2. Помножити на ;
3. Поділити поліном на вибраний твірний поліном , степінь якого дорівнює , і визначити остачу від ділення ;
4. Комбінація циклічного коду дорівнює .

Алгоритм виявлення помилок ґрунтується на тому, що за відсутності помилок закодована комбінація ділиться на твірний поліном без остачі. В цьому випадку декодування полягає у відкиданні контрольних символів. Якщо кодову комбінацію прийнято зі спотворенням, то декодування здійснюється таким чином:

Обчислити остачу(синдром) . Якщо , то комбінацію прийнято без помилок. Наявність остачі свідчить про те, що комбінацію було спотворено;

Знайти вагу остачі . Якщо ( — коректувальна здатність коду), то до прийнятої комбінації додають за модулем остачу і одержують виправлену комбінацію;

Якщо , то здійснюють циклічний зсув на один символ ліворуч і отриману комбінацію ділять на твірний многочлен. Якщо після цього , то до зсунутої комбінації додають остачу і після додавання зсувають на один символ праворуч. В результаті одержують виправлену комбінацію;

Якщо після циклічного зсуву , то здійснюють додаткові циклічні зсуви ліворуч з діленням на на кожному кроці. При виконанні умови виконують дії, вказані у попередньому пункті, причому зсувів праворуч потрібно буде виконати стільки ж, скільки було виконано зсувів ліворуч.

**Кодування**

*Потрібно закодувати послідовність* інформаційних символів. З умови знаходимо кількість перевірних символів . Виберемо твірний поліном , у двійковому записі . Виконаємо ділення на (використаємо для цього програму, написану спеціально для циклічних кодів):

>>> Result = divPoly(multiNPoly(Qx, r),Px)

>>> sf(Result[1])

‘101’

Таким чином, знайдено . Тоді закодована комбінація:

.

**Декодування**

*Нехай прийнято послідовність*. Здійснимо декодування за допомогою програми:

>>> sf(BCH\_Decode(fs("11100111111001001011110000101"), fs("100101"),2))

'111001111110010010111100'

Оскільки процедура декодування автоматично відкинула перевірні біти, то ця комбінація — результат декодування з виправленням помилки.

4.5.7 Коди Боуза-Чоудхурі-Хоквінгема

Ці коди є різновидом циклічних кодів з кодовою відстанню . Вони дають змогу виявляти та виправляти будь-яку кількість помилок. При кодуванні задаються кількістю помилок, яку слід виправити, або мінімальною кодовою відстанню та загальною кількістю перевірних елементів у кодовій комбінації. Кількість інформаційних і перевірних елементів визначають при побудові коду Боуза-Чоудхурі-Хоквінгема (БЧХ). Розглянемо деякі правила цієї побудови.

Довжину комбінації кодів БЧХ можна визначити так: , або , де — ціле число; — непарне число, при діленні на яке стає цілим непарним числом. Таким чином, довжина може мати тільки непарну кількість елементів.

Кількість перевірних елементів коду визначається виразом

А кількість інформаційних елементів — виразом

Твірний поліном коду БЧХ є найменшим спільним кратним (НСК) мінімальних поліномів , де — порядок полінома . Отже, кількість мінімальних поліномів визначається кількістю помилок , які виправляються кодом: .

Найбільше значення степеня мінімального полінома є найменшим цілим числом, при якому ділиться на або без остачі. Звідси випливає, що .

Кількість помилок, які можуть виправляти коди БЧХ, не обмежена, але зі збільшенням кратності помилки значно зростає складність пристроїв декодування, що призводить до зменшення швидкості передачі інформації.

**Кодування**

*Вхідна послідовність* . Запустимо програму для автоматичного кодування та визначення твірного поліному:

comb = fs("111001111110010011111100")

errors = 7

result = BCH\_Encode(comb, errors)

print("Encoding results:")

print("-- result:",sf(result[0]), " len=",len(result[0]))

print("-- poly :",sf(result[1]),"\n")

Отримаємо:

Encoding results:

-- result: 111001111110010011111100000000010000000010101111011100110011010 len = 63

-- poly : 1111011010011010110000100000100100100001

Це і буде закодована комбінація з можливістю виправлення 7 помилок і довжиною інформаційної частини 24 біти. Надмірність коду

**Декодування**

*Нехай прийнято комбінацію*

Запустимо програму-декодер:

print(sf(trimZero(BCH\_Decode(fs(“111001111110010000000000000000000000000010101111011100110011010”),fs(“1111011010011010110000100000100100100001”),7))))

Отримаємо

Decoding result:

-- Maxdist: 39

-- Got : 111001111110010011111100

-- Origin : 111001111110010011111100

Ok

4.5.8 Рекурентний код

Рекурентними (неперервними) називаються коди, що подаються неперервною послідовністю елементів без поділу на окремі комбінації. Від блокових рекурентні коди відрізняється тим, що дають змогу кодувати інформаційну послідовність неперервно, не поділяючи її на блоки фіксованої дожини з інформаційними та перевірними елементами. В них при передачі кожний перевірний елемент формується додаванням за модулем двох інформаційних елементів, відстань між якими дорівнює кроку додавання .

Кількість перевірних елементів, сформованих за час , дорівнює кількості інформаційних елементів, які надійшли за той самий час. Ці елементи передаються через один . На приймальному боці вони розділяються та реєструються незалежно.

**Кодування**

*Початкова послідовність*: . Кодування здійснюється шляхом додавання за модулем 2 двох інформаційних елементів, відстань між якими дорівнює кроку додавання .

Тоді послідовність перевірних елементів:

.

**Декодування**

Нехай прийнято неперервну послідовність, яку було розділено на інформаційну та перевірну частини:

Сформуємо за прийнятою інформаційною частиною перевірну:

.

Порівняння та показує, що . Звідси випливає, що елемент переданий зі спотворенням. Після інвертування цього елемента та відкидання перевірної частини отримуємо:

4.6 Канальні коди

Канальні коди використовуються в цифрових системах передачі для вторинного кодування повідомлень при їх передачі по лініям(каналам) зв’язку.

4.6.1 Дуобінарний код

В дуобінарному коді двійкової інформаційної послідовності передається паузою, а - імпульсами додатної та від’ємної полярності, зі зміною полярності у кожному наступному імпульсі порівняно з попереднім. Таке кодування дає можливість звузити спектр інформаційної послідовності, що передається.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

4.6.2 Квазітрійковий код

При передачі елементів інформаційної послідовності квазітрійковим кодом використовуються прямокутні імпульси меншої довжини порівняно з дуобін.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | | 1 | | 1 | | 0 | 0 | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | | 0 | 0 | 1 | | 0 | 0 | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | | 0 | 0 |
|  |  |  | |  |  |  |  |  | |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  | |  |  |  | |  |  |
|  | |  |  |  | |  |  |  |  |  | |  |  |  | |  |  |  | |  |  |  |  |  |  |  | |  |  |  | |  |  |  | |  |  |  |  |

4.6.3 Код Манчестер ІІ

У цьому коді кожний розряд двійкового початкового коду записується у вигляді двох елементів: — як , а — як . Так, початковій кодовій комбінації відповідатиме комбінація кореляційного коду.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | | 1 | | 1 | | 0 | | 0 | | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | | 0 | | 0 | | 1 | | 0 | | 0 | | 1 | | 1 | | 1 | | 1 | | 1 | | 1 | | 0 | | 0 | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

4.6.4 Код 4В3Т

В коді 4В3Т чотирьом бітовим елементам інформаційної послідовності ставляться в відповідність елементи тріскового коду, в якому передається паузою, — імпульсом від’ємної полярності, а — імпульсом додатної полярності. Це дає можливість зменшити довжину закодованої інформаційної послідовності і час, що витрачається на її передачу. Існує три варіанти такого коду — R1, R2 та R3.

R1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | - | 0 | - | + | 0 | + | - | 0 | 0 | + | + | + | + | - | + | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

R2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | - | 0 | - | + | + | + | - | 0 | - | 0 | 0 | + | - | - | + | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

R3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | - | 0 | - | - | + | + | - | 0 | - | 0 | 0 | + | - | - | 0 | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

4.5 Штрихові коди

Найпоширенішим способом запису штрихових кодів є запис у вигляді штрихів та пробілів різної ширини.

Штрихові коди призначені для кодування цифр ( і п’яти додаткових символів (СТАРТ, СТОП, і розділові знаки)ю

Код двоспрямований і може мати кодове слово завдовжки , , знаків. Однак існують два основних різновиди цього коду: -13 і -8, де цифрою позначено довжину коду (кількість знаків у кодовому слові).

Остання цифра є контрольною, вона формується за спеціальним алгоритмом:

1. Крок 1. Знаходять суму цифр, розташованих на непарних позиціях кодового слова (перегляд виконують справа наліво), і множать здобутий результат на .
2. Крок 2. Знаходять суму цифр, розташованих на парних позиціях кодового слова.
3. Крок 3. Додають результати, здобуті у попередніх кроках.
4. Крок 4. Контрольна цифра дорівнює найменшому числу, що не перевищує , яке, якщо його додати до результату, здобутому на попередньому кроці, дає число, кратне .

**Кодування**

Закодуємо послідовність кодом -8. Знайдемо перевірну цифру.

Тоді перевірна цифра дорівнює . Закодована комбінація:

Графічне представлення коду:



1. Висновки

Під час виконання роботи було досліджено деякі коди, їх характеристики, обмеження, властивості та області застосування. Процеси кодування та декодування розглянуто на прикладах.

Було написано програму-кодер та декодер БЧХ кодів з можливістю виправлення будь-якої кількості помилок у комбінації будь-якої довжини. Також було написано програму, яка будує дерево для коду Хаффмана.

Додатки

Додаток А. Лістинг програми

"""

Bose-Chaudhuri-Hocquenghem code encoding-decoding

Автор: Глинський Данило

Використання:

Всі поліноми задаються списком одининць або нулів. Приклад:

[1,0,0,1,1] - поліном 11001, тобто х^4 + x^3 + 1

string sf(Poly) - перетворює поліном в рядок (приклад вище)

string fs(st) - перетворює рядок в поліном (обернена операція до sf() )

Poly trimZero(poly) - відсікає ведучі(лишні) нулі

(Encoded, Polynomial) BCH\_Encode(comb, s) - закодовує поліном(вхідну комбінацію)

з заданою кількістю помилок s. Результат поміщується у змінну

Encoded, твірний поліном, використаний для кодування, поміщується у

Polynomial

Encoded BCH\_Encode\_Manual(comb,poly) - загальний для циклічних кодів

кодер. Кодує комбінацію comb по модулю полінома poly і повертає

вихідну(закодовану) комбінацію

Decoded BCH\_Decode(comb, poly, s) - декодує комбінацію comb за допомогою заданого

полінома poly і заданої максимальної кількості помилок s

У разі помилки (довжина пакету помилок більша граничної або помилок більше

ніж s) видає [-1]

"""

import inspect, re

def plusOne(c):

i = 0

while not i == -1:

if i == len(c):

c += [0]

c[i] = c[i] ^ 1

if c[i] == 0:

i += 1

else:

i = -1

return c

def addPoly(p1, p2):

summ = []

for i in range(0,max(len(p1),len(p2))):

summ += [0]

try:

summ[i] ^= p1[i]

except:

pass

try:

summ[i] ^= p2[i]

except:

pass

return summ

def trimZero(poly):

for i in range(len(poly)-1,0,-1):

if poly[i] == 0:

poly.pop()

else:

break;

return poly

def multiNPoly(p, n):

return [0]\*n + p

def multiPoly(p1, p2):

res = []

for i in range(0,len(p1)):

res = addPoly(res, multiNPoly(p2, i)\*p1[i])

return res

def divPoly(p1,p2):

mod = trimZero(p1[:])

p2 = trimZero(p2)

res = [0]\*(len(p1)-len(p2)+1)

while True:

if len(mod)<len(p2):

return (res, mod)

res[len(mod)-len(p2)] = 1

mod = trimZero(addPoly(mod, multiNPoly(p2,len(mod)-len(p2))))

def isPrime(poly, plist):

for i in plist:

if len(poly)<len(i):

return False

if divPoly(poly, i)[1] == [0]:

return False

return True

def getPrimes(startPoly, N, existList):

p = startPoly

i = 0

while len(p) < N:

if isPrime(p, existList):

existList += [p[:]]

i += 1

p = plusOne(p)

return existList

def sf(poly):

return "".join([str(i) for i in poly])[::-1]

def fs(st):

return [int(i) for i in st[::-1]]

def galuaElem(poly, N):

return divPoly(multiNPoly([1],N % (2\*\*(len(poly)-1)-1)),poly)[1]

def computePoly(compPoly, basePoly, power):

res = [0]

for i in range(0,len(compPoly)):

if compPoly[i] == 1:

res = addPoly(res, galuaElem(basePoly, power\*i))[:]

return trimZero(res)

def printComp(compPoly, basePoly, power):

print(sf(computePoly(fs(compPoly),fs(basePoly),power)))

def GetPrimesPow(N):

return [i for i in getPrimes([0,1],N+2,[[0,1]]) if len(i) == N + 1]

def getPolyPowM(p, M):

if M == 1:

l = GetPrimesPow(p)

for i in l:

if sum(i) == 3:

return i

else:

first = getPolyPowM(p, 1)

poly = [1]

mc = 0

i = 0

polis = []

while True:

newAdd = trimZero(addPoly([0,1],galuaElem(first, M\*2\*\*i)))

if newAdd in polis:

break

polis += [newAdd[:]]

mc += 1

poly = multiPoly(poly, newAdd)

i += 1

if len(divPoly(poly,first)[1])-1 < mc:

return addPoly(divPoly(poly,first)[1],first)

else:

return divPoly(poly,first)[1]

def NSK(plist):

l = []

res = [1]

for i in plist:

if i not in l:

l += [i[:]]

res = multiPoly(res, i)

return res

def shiftPoly(poly, ns):

return poly[-ns:]+poly[:-ns]

def BCH\_Encode\_Manual(comb, poly):

res = addPoly(multiNPoly(comb, len(poly)-1),divPoly(multiNPoly(comb, len(poly)-1),poly)[1])

for h in range(2,50):

if len(res) <= 2\*\*h-1:

c = int((2\*\*h-1-len(res)))

return res + [0 for i in range(0,c)]

def BCH\_Encode(comb, s):

h = 0

for h in range(3,50):

if 2\*\*h - len(NSK([getPolyPowM(h,i) for i in range(1,2\*s,2)])) >= len(comb):

break

poly = NSK([getPolyPowM(h,i) for i in range(1,2\*s,2)])

return (BCH\_Encode\_Manual(comb,poly),poly)

def BCH\_Decode(comb, poly, s):

for i in range(0,len(comb)):

rest = divPoly(shiftPoly(comb,i),poly)[1]

if sum(rest) <= s:

return trimZero(shiftPoly(addPoly(shiftPoly(comb,i),rest),-i)[len(poly)-1:])

return([-1])

def main():

#Simple comb error check

#Вхідна комбінація

comb = fs("111001111110010011111100")

errors = 7

result = BCH\_Encode(comb, errors)

comba = fs("111001111110010011111100000000010000000010101111011100110011010")

comberr1 = fs("111001111110010000000000000000000000000010101111011100110011010")

poly = fs("1111011010011010110000100000100100100001")

print("Encoding results:")

print("-- result:",sf(result[0]), " len =",len(result[0]))

print("-- poly :",sf(result[1]),"\n")

print("Decoding result:")

print("-- Maxdist:",len(result[0])-(len(result[0])-len(result[1])+1))

print("-- Got :",sf(trimZero(BCH\_Decode(comberr1,result[1],errors)+[0])))

print("-- Origin :",sf(trimZero(comb)))

#Multi errors (check power of BCH)

#Перевірка можливості БЧХ-кодів виправляти помилки. Методом повного

#перебору вказуємо усі можливі помилки, вносимо і декодуємо

"""

comb = fs("10111011")

errors = 0

Encoded = fs("0000000010111011111100101110010")

Poly = fs("1000111110101111")

print(len(Encoded)-(len(Encoded)-len(Poly)+1))

print("#Encoding results:")

print("-- result:", sf(Encoded))

print("-- poly :", sf(Poly), "\n")

print("#Decoding result:")

#print("-- Got :",sf(trimZero(BCH\_Decode(Encoded,Poly,errors))))

print("-- Origin:",sf(trimZero(comb)))

if errors >= 1:

print("#Testing one error...")

for i in range(0,len(Encoded)):

errorcomb = Encoded[:]

errorcomb[i] = 1 - errorcomb[i]

if not trimZero(BCH\_Decode(errorcomb,Poly,errors)) == trimZero(comb):

print("Error found :", sf(errorcomb)," ",i)

if errors >= 2:

print("#Testing two errors...")

for i in range(0,len(Encoded)):

for j in range(i+1, len(Encoded)):

errorcomb = Encoded[:]

errorcomb[i] = 1 - errorcomb[i]

errorcomb[j] = 1 - errorcomb[j]

if not trimZero(BCH\_Decode(errorcomb,Poly,errors)) == trimZero(comb):

print("Error found :", sf(errorcomb)," ",i,j)

if errors >= 3:

print("#Testing three errors...")

for i in range(0,len(Encoded)):

for j in range(i+1, len(Encoded)):

for k in range(j+1, len(Encoded)):

errorcomb = Encoded[:]

errorcomb[i] = 1 - errorcomb[i]

errorcomb[j] = 1 - errorcomb[j]

errorcomb[k] = 1 - errorcomb[k]

if not trimZero(BCH\_Decode(errorcomb,Poly,errors)) == trimZero(comb):

print("Error found :", sf(errorcomb)," ",i,j,k)

"""

print(" Ok")

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

main()

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 0.12903 | 0.12903 | 0.12903 | 0.12903 | 0.12903 | 0.12903 | 0.12903 | 0.12903 | 0.12903 | 0.13978 | 0.15053 | 0.17204 |
| 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.12903 | 0.13978 | 0.15053 |
| 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.10752 | 0.12903 | 0.13978 |
| 0.08602 | 0.08602 | 0.08602 | 0.08602 | 0.08602 | 0.08602 | 0.08602 | 0.08602 | 0.10752 | 0.10752 | 0.10752 | 0.12903 |
| 0.07526 | 0.07526 | 0.07526 | 0.07526 | 0.07526 | 0.07526 | 0.08602 | 0.08602 | 0.08602 | 0.10752 | 0.10752 | 0.10752 |
| 0.07526 | 0.07526 | 0.07526 | 0.07526 | 0.07526 | 0.07526 | 0.07526 | 0.08602 | 0.08602 | 0.08602 | 0.10752 | 0.10752 |
| 0.05376 | 0.05376 | 0.05376 | 0.05376 | 0.06451 | 0.07526 | 0.07526 | 0.07526 | 0.08602 | 0.08602 | 0.08602 | 0.10752 |
| 0.05376 | 0.05376 | 0.05376 | 0.05376 | 0.05376 | 0.06451 | 0.07526 | 0.07526 | 0.07526 | 0.08602 | 0.08602 | 0.08602 |
| 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.05376 | 0.05376 | 0.06451 | 0.07526 | 0.07526 | 0.07526 | 0.08602 |  |
| 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.05376 | 0.05376 | 0.06451 | 0.07526 | 0.07526 |  |  |
| 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.05376 | 0.05376 | 0.06451 |  |  |  |
| 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.04301 | 0.05376 |  |  |  |  |
| 0.03225 | 0.03225 | 0.03225 | 0.04301 | 0.04301 | 0.04301 | 0.04301 |  |  |  |  |  |
| 0.03225 | 0.03225 | 0.03225 | 0.03225 | 0.04301 | 0.04301 |  |  |  |  |  |  |
| 0.02150 | 0.02150 | 0.03225 | 0.03225 | 0.03225 |  |  |  |  |  |  |  |
| 0.02150 | 0.02150 | 0.02150 | 0.03225 |  |  |  |  |  |  |  |  |
| 0.01075 | 0.02150 | 0.02150 |  |  |  |  |  |  |  |  |  |
| 0.01075 | 0.01075 |  |  |  |  |  |  |  |  |  |  |
| 0.01075 |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|  |  |  |  |  | 0.19354 | 0.21505 | 0.26881 | 0.32258 | 0.40860 | 0.59139 | 1.0 |
|  |  |  |  |  | 0.17204 | 0.19354 | 0.21505 | 0.26881 | 0.32258 | 0.40861 |  |
|  |  |  |  |  | 0.15053 | 0.17204 | 0.19354 | 0.21505 | 0.26881 |  |  |
|  |  |  |  |  | 0.13978 | 0.15053 | 0.17204 | 0.19354 |  |  |  |
|  |  |  |  |  | 0.12903 | 0.13978 | 0.15053 |  |  |  |  |
|  |  |  |  |  | 0.10752 | 0.12903 |  |  |  |  |  |
|  |  |  |  |  | 0.10752 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Додаток Б. Таблиця ймовірностей

Додаток В. Кодове дерево для коду Хаффмана

Додаток Г. Список використаної літератури

1. Жураковський, Полторак. Теорія інформації та кодування
2. Пітерсон У. Коди, які виправляють помилки. 1976 рік.