Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Кафедра интеллектуально-информационных технологий

Лабораторная работа №2

По дисциплине «Современные методы защиты компьютерных систем»

Выполнил:

студент 4 курса

группы ИИ-22

Копанчук Е. Р.

Проверил:

Хацкевич А. С.

Брест-2024

**Ход работы:**

**Задание:** закрепить теоретические знания по использованию итеративных кодов для повышения надежности передачи и хранения в памяти компьютера двоичных данных, разработать приложение для кодирования/декодирования двоичной информации итеративным кодом с различной относительной избыточностью кодовых слов.

**Код программы:**

class MatrixCode {

private k: number;

private k1: number;

private k2: number;

private z: number;

private parity: number[];

private isPpBits: boolean;

private size: number;

private pSize: number;

constructor(k: number, k1: number, k2: number, z: number = 1, parity: number[] = [2, 3], isPpBits: boolean = true) {

this.k = k;

this.k1 = k1

this.k2 = k2

this.z = z

this.parity = parity

this.isPpBits = isPpBits

this.size = Math.round(this.k / this.z)

this.pSize = 0

if (this.parity.includes(2)) this.pSize += this.k1

if (this.parity.includes(3)) this.pSize += this.k2

if (this.parity.includes(4)) this.pSize += this.k1 + this.k2 - 1

if (this.parity.includes(5)) this.pSize += this.k1 + this.k2 - 1

}

private \_splitPBits(pBits: string) {

const splitted = []

for (let i = 0; i < this.z; i++) {

const zSplit = pBits.slice(i \* this.pSize, (i + 1) \* this.pSize)

const k1Split = zSplit.slice(0, this.k1).split("")

const k2Split = zSplit.slice(this.k1, this.k1 + this.k2).split("")

const k3Split = zSplit.slice(this.k1 + this.k2, (this.k1 + this.k2) \* 2 - 1).split("")

const k4Split = zSplit.slice((this.k1 + this.k2) \* 2 - 1, (this.k1 + this.k2) \* 4 - 2).split("")

splitted.push([k1Split, k2Split, k3Split, k4Split])

}

return splitted

}

private \_getErrPos(pBits: string, pBitsTest: string) {

const splittedPBits = this.\_splitPBits(pBits)

const splittedPBitsTest = this.\_splitPBits(pBitsTest)

let k = 0

while (splittedPBits[k].map(p => p.join("")).join("") === splittedPBitsTest[k].map(p => p.join("")).join("")) k++;

let i = 0

while (splittedPBits[k][0][i] === splittedPBitsTest[k][0][i]) i++;

let j = 0

while (splittedPBits[k][1][j] === splittedPBitsTest[k][1][j]) j++;

return k \* this.size + i \* this.k2 + j

}

private \_getMatrix3d(message: string) {

const matrix3d = []

for (let k = 0; k < this.z; k++) {

const matrix2d = []

for (let i = 0; i < this.k1; i++) {

const row = []

for (let j = 0; j < this.k2; j++) {

row.push(message[k \* this.size + i \* this.k2 + j])

}

matrix2d.push(row)

}

matrix3d.push(matrix2d)

}

return matrix3d

}

private \_getParityBits(matrix2d: number[][]) {

const h = []

if (this.parity.includes(2)) {

for (let i = 0; i < matrix2d.length; i++) {

let bit = 0

for (let j = 0; j < matrix2d[i].length; j++) {

bit ^= matrix2d[i][j]

}

h.push(bit)

}

}

const v = []

if (this.parity.includes(3)) {

for (let j = 0; j < matrix2d[0].length; j++) {

let bit = 0

for (let i = 0; i < matrix2d.length; i++) {

bit ^= matrix2d[i][j]

}

v.push(bit)

}

}

const d = []

if (this.parity.includes(4)) {

for (let k = 0; k < matrix2d.length; k++) {

let [bit, i, j] = [0, 0, k]

while (i < matrix2d.length && j >= 0) {

bit ^= matrix2d[i][j]

i++

j--

}

d.push(bit)

}

for (let k = 1; k < matrix2d[0].length; k++) {

let [bit, i, j] = [0, k, matrix2d.length - 1]

while (i < matrix2d.length && j >= 0) {

bit ^= matrix2d[i][j]

i++

j--

}

d.push(bit)

}

}

const dr = []

if (this.parity.includes(5)) {

for (let k = 0; k < matrix2d.length; k++) {

let [bit, i, j] = [0, 0, k]

while (i < matrix2d.length && j < matrix2d[0].length) {

bit ^= matrix2d[i][j]

i++

j++

}

dr.push(bit)

}

for (let k = 1; k < matrix2d[0].length; k++) {

let [bit, i, j] = [0, k, 0]

while (i < matrix2d.length && j < matrix2d[0].length) {

bit ^= matrix2d[i][j]

i++

j++

}

dr.push(bit)

}

}

if (this.isPpBits) {

let ppBit = 0

for (let i = 0; i < h.length; i++) {

ppBit ^= h[i]

}

for (let i = 0; i < v.length; i++) {

ppBit ^= v[i]

}

return [...h, ...v, ...d, ...dr, ppBit]

}

return [...h, ...v, ...d, ...dr]

}

encode(message: string) {

const matrix3d = this.\_getMatrix3d(message)

const pBits = []

for (let i = 0; i < this.z; i++) {

pBits.push(this.\_getParityBits(matrix3d[i]))

}

const encoded = message + pBits.map(p => p.join("")).join("")

return encoded

}

decode(encoded: string) {

const pBits = encoded.slice(this.k, encoded.length)

const message = encoded.slice(0, this.k)

const encodeTest = this.encode(message)

if (encoded === encodeTest) {

return { errPos: null, message }

} else {

const pBitsTest = encodeTest.slice(this.k, encodeTest.length)

const errPos = this.\_getErrPos(pBits, pBitsTest)

const messageCorrected = message.slice(0, errPos) + (message[errPos] !== "0" ? "0" : "1") + message.slice(errPos + 1, message.length)

return { errPos, message: messageCorrected }

}

}

}

**Пример:**

const message = "11010010111001111001101010011000"

const errPos = 30

let coder = new MatrixCode(32, 4, 8, 1, [2, 3], false)

let encoded = coder.encode(message)

let encodedWithError = encoded.slice(0, errPos) + (encoded[errPos] !== "0" ? "0" : "1") + encoded.slice(errPos + 1, encoded.length)

console.log(message)

console.log(encoded)

console.log(encodedWithError)

console.log(coder.decode(encodedWithError))

coder = new MatrixCode(32, 2, 16, 1, [2, 3], false)

encoded = coder.encode(message)

encodedWithError = encoded.slice(0, errPos) + (encoded[errPos] !== "0" ? "0" : "1") + encoded.slice(errPos + 1, encoded.length)

console.log(message)

console.log(encoded)

console.log(encodedWithError)

console.log(coder.decode(encodedWithError))

coder = new MatrixCode(32, 8, 2, 2, [2, 3, 4, 5], false)

encoded = coder.encode(message)

encodedWithError = encoded.slice(0, errPos) + (encoded[errPos] !== "0" ? "0" : "1") + encoded.slice(errPos + 1, encoded.length)

console.log(message)

console.log(encoded)

console.log(encodedWithError)

console.log(coder.decode(encodedWithError))

coder = new MatrixCode(32, 4, 4, 2, [2, 3, 4, 5], false)

encoded = coder.encode(message)

encodedWithError = encoded.slice(0, errPos) + (encoded[errPos] !== "0" ? "0" : "1") + encoded.slice(errPos + 1, encoded.length)

console.log(message)

console.log(encoded)

console.log(encodedWithError)

console.log(coder.decode(encodedWithError))

