PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

Jean Lucca Uchaki

# Relatório de Introdução à Redes de Computadores: Trabalho 2

Porto Alegre

2021

**SUMÁRIO**

**1 INTRODUÇÃO.....**
**2 REDUNDÂNCIA DE BLOCO.....**
**3 CRC .....**
**4 CÓDIGO DE HAMMING .....**

## 1 Introdução

O trabalho consiste na implementação de codificadores e decodificadores para detecção e correção de erros usando as técnicas de redundância de bloco, CRC e código de Hamming. Os codificadores e decodificadores foram desenvolvidos na linguagem JAVA, para a execução dos codificadores e decoficadores não é necessario nenhum tipo de IDE basta compilar os códigos fonte e executá-los via linha de comando como no exemplo abaixo:

- javac *.java
- java Bcc_encoder pucrs
- java Bcc_decoder e1ebc6e4e7cf

## 2 Redundância de bloco

**Encoder:**

Monta um Array com os códigos binários e armazenas nas últimas posições de o bit de paridade.

```
public static String encode( String str ) {
        String res = "";
        String[] binArray = new String[str.length()+1];
        binArray[binArray.length-1] = "";
        for( int i = 0; i < str.length(); i++ ) {
            String aux = Integer.toBinaryString(str.charAt(i)-0);
            binArray[i] = aux += Util.countParity(aux);
        }
        for( int i = 0; i < 8; i++ ) {
            String aux = "";
            for( int j = 0; j < binArray.length-1; j++ ) {
                aux += binArray[j].charAt(i);
            }
            binArray[binArray.length-1] += Util.countParity(aux);
        }
        for( int j = 0; j < binArray.length; j++ ) {
            res += Util.binaryToHex(binArray[j]);
        }
        return res;
    }
```

**Decoder:**

Percorre o array verificando os bits de paridade armazenados nas últimas posições remove os bits de redundância e converte para ASCII caso não exista nenhum erro.

```java
public static String decode(String hex) {
        String res = "";
        String bin = Util.hexToBinary(hex);
        String[] binArray = new String[bin.length()/8];
        for( int i = 0; i < binArray.length; i++ ) {
            binArray[i] = "";
            for( int j=0; j<8; j++ ) {
                binArray[i] += bin.charAt(8*i + j);
            }
        }
        for( int i = 0; i < binArray.length; i++ ) {
            if ( Util.countParity( binArray[i].substring(0,7) ) == '0' &&
binArray[i].charAt(7) != '0' ) {
                return "ERRO";
            }
        }
        String bcc = binArray[binArray.length-1];
        for( int i = 0; i < 7; i++) {
            String aux = "";
            for( int j = 0; j < binArray.length-1; j++ ) {
                aux += binArray[j].charAt(i);
            }

            if ( Util.countParity( aux ) != bcc.charAt(i) ) {
                return "ERRO";
            }
        }
        for( int i = 0; i < binArray.length-1; i++ ) {
            res += Util.binToASCII(binArray[i].substring(0, 7));
        }
        return res;
    }
```

**Exemplo:**

```
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Bcc_encoder PUCRS
a0aa87a5a68e

C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Bcc_decoder a0aa87a5a68e
PUCRS

C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Bcc_decoder a0aa87a5a78e
ERRO

C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Bcc_encoder REDES
a58b888ba68b

C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Bcc_decoder a58b888ba68b
REDES

C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Bcc_decoder a58b898ba68b
ERRO

C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>
```

## 3 Crc

**Encoder:**

Para cada caractere concatena a quantidade de zeros correspondente ao tamanho do polinômio gerador, efetua a divisão pelo polinômio gerador e concatena com a string resultante.

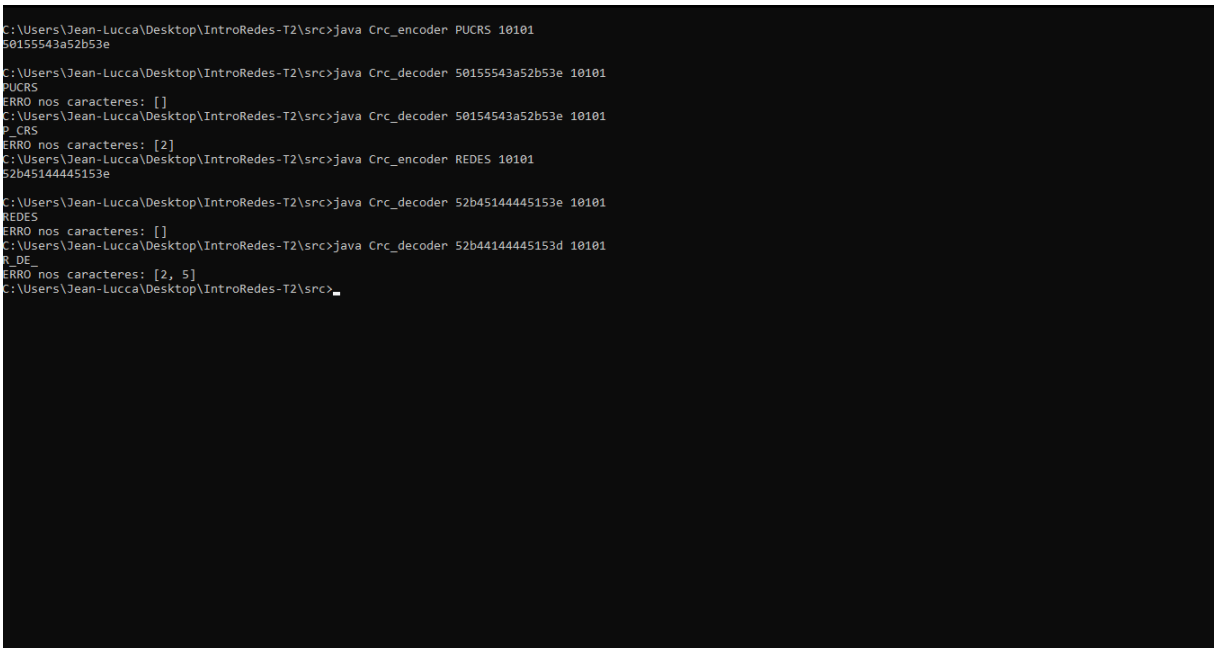
```
public static String encode(String str, String pol) {
        String res = "";
        for( int i =0; i < str.length(); i++ ) {
            String aux = Integer.toBinaryString(str.charAt(i)-0);
            res += Util.binaryToHex(aux) +
                   Util.binaryToHex(Util.divide(aux+Util.appendNZeros(pol.
length()-1), pol));
        }
        return res;
    }
```

**Decoder:**

Para cada 3 caracteres hexadecimais executa a divisão pelo polinômio gerador, caso a divisão resulte em zero concatena o resultado senão descarta o caractere.

```
public static void decode(String str, String pol) {
        String res = "";
        LinkedList<Integer> err = new LinkedList<Integer>();
        int count = 1;
        for( int i = 0; i < str.length()/3; i++ ) {
            String aux = "";
            for( int j=0; j<3; j++ ) {
                aux +=  str.charAt(3*i + j);
            }
            aux = Util.hexToBinary(aux);
            int check = Util.binToDec(Util.divide(aux, pol));
            if( check == 0 ) {
                res += Util.binToASCII(aux.substring(0 , aux.length() -
pol.length()+1));
            } else {
                err.add(count);
                res += "_";
            }
            count++;
        }
        System.out.println(res);
        System.out.print("ERRO nos caracteres: ");
        System.out.print(err);
    }
```

## Exemplo:


C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Crc_encoder PUCRS 10101
50155543a52b53e
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Crc_decoder 50155543a52b53e 10101
PUCRS
ERRO nos caracteres: []
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Crc_decoder 50154543a52b53e 10101
P_CRS
ERRO nos caracteres: [2]
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Crc_encoder REDES 10101
52b45144445153e
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Crc_decoder 52b45144445153e 10101
REDES
ERRO nos caracteres: []
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Crc_decoder 52b44144445153d 10101
R_DE_
ERRO nos caracteres: [2, 5]
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>

## 4 Código de Hamming

**Encoder:**

Inverte a string adiciona os bits de Hamming (representados como ‘x’) procura as posições com bit 1 calcula a paridade dessas posições substitui os x’s e concatena na string resultante.

```
public static String encode(String str) {
        String res = "";
        String[] binArray = new String[str.length()+1];
        binArray[binArray.length-1] = "";
        for( int i = 0; i < str.length(); i++ ) {
            String aux = Integer.toBinaryString(str.charAt(i)-0);
            binArray[i] = "0" + aux;
            res += run(hamming(Util.reverse(binArray[i])));
        }
        System.out.println(res);
        return res;
    }
public static String run(String str) {
        int count = 1;
        ArrayList<String> binArray = new ArrayList<String>();
        for( int i =0; i < str.length(); i++ ) {
            if( str.charAt(i) == '1' ) {
                binArray.add( String.format("%4s",
Integer.toBinaryString(count)).replace(' ', '0') );
            }
            count++;
        }
        binArray.add("");
        for( int i = 0; i < 4; i++ ) {
            String aux = "";
            for( int j = 0; j < binArray.size()-1; j++ ) {
                aux += binArray.get(j).charAt(i);
            }
            binArray.set(binArray.size()-1, binArray.get(binArray.size()-
1)+Util.countParity(aux));
        }
        String reversedParity = Util.reverse(binArray.get(binArray.size()-
1));
        str = removeX(str, reversedParity);
        return Util.binaryToHex(Util.reverse(str));
    }
```

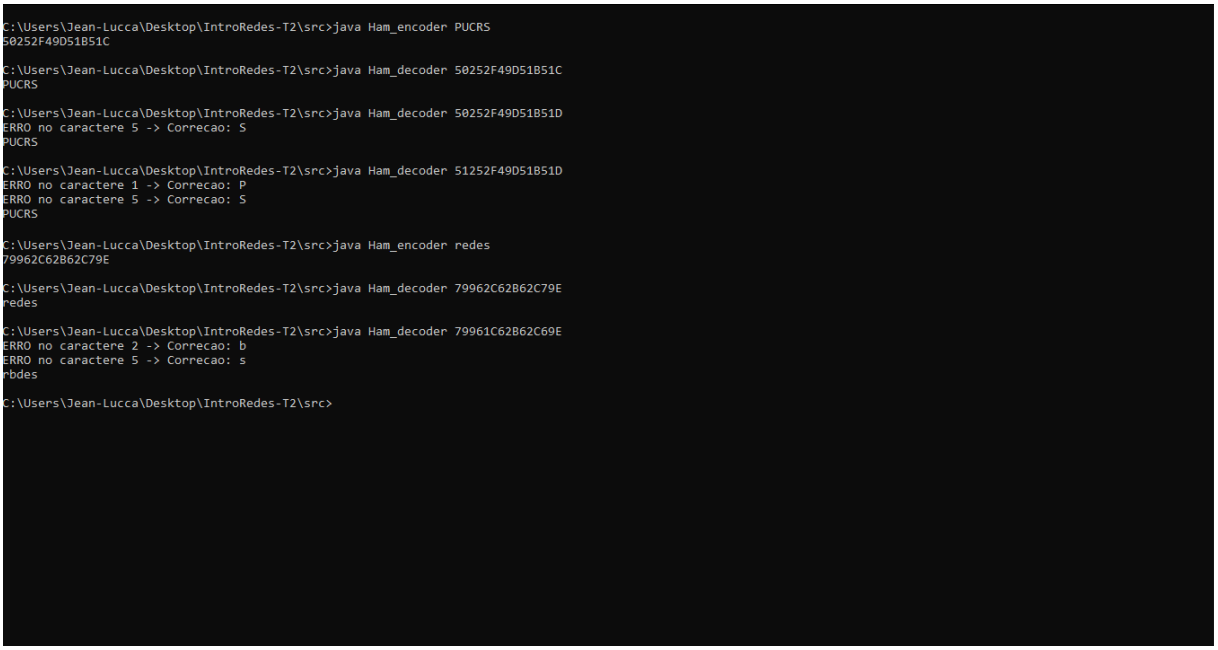
**Decoder:**

Procura as posições com bit 1 calcula a paridade caso o resultado seja 0 remove os bit's de hamming e concatena com a string resultante.

Caso encontre algum erro inverte o bit na posição do resultado do calculo de paridade.

```
public static String decode(String hex) {
        String res = "";
        for( int i = 0; i < hex.length()/3; i++ ) {
            String aux = "";
            for( int j=0; j<3; j++ ) {
                aux += hex.charAt(3*i + j);
            }
            res +=
Util.binToASCII(run(Util.reverse(Util.hexToBinary(aux)),i));
        }
        return res;
    }
public static String run(String str, int charIndex) {
        int count = 1;
        ArrayList<String> binArray = new ArrayList<String>();
        for( int i =0; i < str.length(); i++ ) {
            if( str.charAt(i) == '1' ) {
                binArray.add( String.format("%4s",
Integer.toBinaryString(count)).replace(' ', '0') );
            }
            count++;
        }
        binArray.add("");
        for( int i = 0; i < 4; i++ ) {
            String aux = "";
            for( int j = 0; j < binArray.size()-1; j++ ) {
                aux += binArray.get(j).charAt(i);
            }
            binArray.set(binArray.size()-1, binArray.get(binArray.size()-
1)+Util.countParity(aux));
        }
        int check = Util.binToDec(binArray.get(binArray.size()-1));
        if( check == 0 ) {
            return Util.reverse(removeHamming(str));
        } else {
            String res = Util.reverse(removeHamming(replaceXor(str, check-
1, (char) ((int) str.charAt(check-1) ^ 1))));
            System.out.println( "ERRO no caractere " +(charIndex+1)+" ->
Correcao: "+ Util.binToASCII(res));
            return res;
        }
    }
```

## Exemplo:


C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Ham_encoder PUCRS
50252F49D51B51C
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Ham_decoder 50252F49D51B51C
PUCRS
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Ham_decoder 50252F49D51B51D
ERRO no caractere 5 -> Correcao: S
PUCRS
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Ham_decoder 51252F49D51B51D
ERRO no caractere 1 -> Correcao: P
ERRO no caractere 5 -> Correcao: S
PUCRS
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Ham_encoder redes
79962C62B62C79E
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Ham_decoder 79962C62B62C79E
redes
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>java Ham_decoder 79961C62B62C69E
ERRO no caractere 2 -> Correcao: b
ERRO no caractere 5 -> Correcao: s
rbdes
C:\Users\Jean-Lucca\Desktop\IntroRedes-T2\src>