Pratham Rehka
CS1
150

Experiment 02

Aim: Implementation of Hamming Code for Error detection and correction.

Theory:

Hamming Code is a set of error-correction codes that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver. It is a technique developed by R.W. Hamming for error correction.

Redundant bits — Redundant bits are extra bits (binary) that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The number of bits to be added can be calculated using the formula.

$$2^p \geq n + p + 1$$

E.g: For data bits — 1011001,

$n = 7$

$\therefore 2^4 \geq 7 + 4 + 1$

$\therefore$ Redundant bits = 4.

Redundant bits are placed at positions that correspondly power of 2 → 1, 2, 4, 8.

Total no. of bits ⇒ 11. (7+4)

11	10	9	8	7	6	5	4	3	2	1
1	0	1	R_8	1	0	0	R_4	1	R_2	R_1

For R_1 → 1, 3, 5, 7, 9, 11.

Since total no. of 1's in all the bit positions correspondly to R_1 is an even number.
∴ $R_1 = 0$.

Similarly, R_2 → 2, 3, 6, 7, 10, 11
No. of 1s is odd ∴ $R_2 = 1$

R_4 → check bits 4, 5, 6, 7.
∴ No. of 1's is odd.
∴ $R_4 = 1$.

R_8 → Check bits 8, 9, 10, 11
∴ No. of bits is even.
∴ $R_8 = 0$.

∴ Data transmitted:
1 01 01 00 1 1 10

Error detection & correction:

Suppose the 6th bit is changed from 0 to 1 during transmission.

∴ 10101101110.

$R_1 \rightarrow$ 1, 3, 5, 7, 9, 11 (Even 1's)
∴ $R_1 \rightarrow 0$.

$R_2 \rightarrow$ 2, 3, 6, 7, 10, 11 (Odd 1's)
∴ $R_2 \rightarrow 1$

$R_4 \rightarrow$ 4, 5, 6, 7 (Odd 1's)
∴ $R_4 \rightarrow 1$

$R_8 \rightarrow$ 8, 9, 10, 11 (Even 1's)
∴ $R_8 \rightarrow 0$

∴ These bits give binary number 0110.
This represents 6.
Thus, 6th bit has error.

To correct it, 6th bit is changed from 1 to 0.

Ranodk
22/8/23

150

A

Experiment 02 - Hamming Code

Code:

```
import java.util.*;

public class HammingCode {

    public static int binToDec(String str) {

        int bin = 0;

        for (int i = 0; i < str.length(); i++) {

            if (str.charAt(i) == '1')

                bin += Math.pow(2, i);

        }

        return bin;

    }



    public static int detectError(int[] received, int r) {

        StringBuilder errorWord = new StringBuilder();

        for (int i = 0; i < r; i++) {

            int count = 0;

            for (int j = 1; j < received.length; j++) {

                if (((j >> i) & 1) == 1) {

                    if (received[j] == 1)

                        count++;

                }
```

```
            }
            if (count % 2 == 0)
                errorWord.append('0');
            else
                errorWord.append('1');
        }
        System.out.println("Error Word: " + errorWord);
        int errorPosition = binToDec(errorWord.toString());
        return errorPosition;
    }

    public static int[] generateCode(int[] data, int n, int r) {
        int arr[] = new int[n + r + 1];
        int j = 1;
        for (int i = 1; i < arr.length; i++) {
            if ((Math.ceil(Math.log(i) / Math.log(2)) == Math.floor(Math.log(i) /
Math.log(2))))
                arr[i] = 0;
            else {
                arr[i] = data[j];
                j++;
            }
        }
```

```
        for (int i = 0; i < r; i++) {
            int x = (int) Math.pow(2, i);
            for (j = 1; j < arr.length; j++) {
                if (((j >> i) & 1) == 1) {
                    if (x != j)
                        arr[x] = arr[x] ^ arr[j];
                }
            }
        }
        return arr;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of data bits:");
        int n = sc.nextInt();
        System.out.println("Enter data bits:");
        int data[] = new int[n + 1];
        for (int i = 1; i <= n; i++)
            data[i] = sc.nextInt();
        int r = 0;
        while ((Math.pow(2, r)) < n + r + 1)
            r++;
```

```
        int[] encoded = generateCode(data, n, r);

        System.out.println("Generated Code:");

        for (int i = 1; i <= n + r; i++)

            System.out.print(encoded[i] + " ");

        System.out.println("\nEnter received code for error detection and
correction:");

        int received[] = new int[n + r + 1];

        for (int i = 1; i <= n + r; i++)

            received[i] = sc.nextInt();

        int errorPosition = detectError(received, r);

        if (errorPosition != -1) {

            System.out.println("Error detected at position: " + errorPosition);

            received[errorPosition] = received[errorPosition] ^ 1;

            System.out.println("Corrected code:");

            for (int i = 1; i <= n + r; i++)

                System.out.print(received[i] + " ");

        } else

            System.out.println("No error detected.");

    }

}
```

Output:

```
PS C:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Sem-5\CN> cd
"c:\Users\rohra\OneDrive\Desktop\Pratham's Stuff\Sem-5\CN\" ; if ($?) { javac
HammingCode.java } ; if ($?) { java HammingCode }
Enter number of data bits:
4
```

```
Enter data bits:
1 1 0 1
Generated Code:
1 0 1 0 1 0 1
Enter received code for error detection and correction:
1 1 1 0 1 0 1
Error Word: 010
Error detected at position: 2
Corrected code:
1 0 1 0 1 0 1
```