

Ex No: 6

DATE: 28.08.25 HAMMING CODE FOR ERRORS

Aim:

Write a program to implement error detection and correction using hamming code concept. Make a test to input data stream and verify error correction feature.

Error correction at data link layer:

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 transmitted from the sender to the receiver. It is a technique developed by R.W Hamming for error correction.

Create sender program:

1. Input to select file should be a text of any length. Program should convert the text to binary.
2. Apply hamming code concept on the binary data and add redundant bits to it.
3. Save the output in a file called channel.

Create a receiver program:

1. Receiver program should read the input from channel.
2. Apply hamming code on binary data & check errors.
3. If there is an error, display the position of error.
4. Else remove the redundant bits and convert the binary data to ascii and display the output.

Hamming code for sender:

def char to binary:

```
return format(ord(ch), '08b')
```

```

def hamming_encode(data):
    d1, d2, d3, d4 = [int(bit) for bit in data]
    p1 = d1 ^ d2 ^ d3
    p2 = d2 ^ d3 ^ d4
    p4 = d2 ^ d3 ^ d4
    return f'{p1}{p2}{d1}{p1}{p2}{d2}{p3}{d3}{p4}' if "p1p2d1p1p2d2p3d3p4" else None

tent = input("Enter tent: ")
with open("channel.txt", "w") as f:
    for ch in tent:
        bin_ch = char_to_binary(ch)
        for i in range(0, 8, 1):
            code = hamming_encode(bin_ch[i:i+7])
            f.write(code)
print("Data written to channel.txt with Hamming code")

```

Hamming code for receiver:

```

def hamming_decode(code):
    b = [0] + [int(bit) for bit in code]
    p1 = b[1] ^ b[3] ^ b[5] ^ b[7]
    p2 = b[2] ^ b[3] ^ b[6] ^ b[7]
    p4 = b[4] ^ b[5] ^ b[6] ^ b[7]
    error_pos = p1 * 1 + p2 * 2 + p4 * 4
    if error_pos != 0:
        print(f"Error detected at position {error_pos}.\n"
              f"correcting...")
        b[error_pos] ^= 1
    d1, d2, d3, d4 = b[3], b[5], b[6], b[7]
    return f'{d1}{d2}{d3}{d4}'

binary_result = ""
with open("channel.txt", "r") as f:
    code = f.read()
    for i in range(0, len(code), 7):
        binary_result += hamming_decode(code[i:i+7])
    tent = ""
    for i in range(0, len(binary_result), 8):
        byte = binary_result[i:i+8]
        tent += chr(int(byte, 2))
    print("Received tent after error correction", tent)

```

Input :

Enter 4 bit data: 1011 [ref: (410) bin] : 1011
Sender side: 0010011
Receiver side: 0010011

Output: Temperature of the object is 30.000000000000003 °C

Original data bits extracted: 1011 "break moments" (top) + 1011
"break moments" (bottom) + 1011 "break moments" (right) + 1011
"break moments" (left) + 1011 "break moments" (middle) + 1011
"break moments" (far left) + 1011 "break moments" (far right)
These bits will be used to calculate (6,6) Reed-Solomon code and
(6,8,0) parity bits. The (6,6) RS code will be used to correct up to 3 errors.
The (6,8,0) parity bits will be used to obtain punctured RS(6,6).
The punctured RS(6,6) will be used to correct up to 3 errors.
The RS(6,6) punctured RS(6,6) will be sent to the channel.
Input to encoder will be (6,6) above punctured RS(6,6).
Program takes [e6] as fid ref [(4,4)+oi] + [o] + o
First forming code word [E6]dA[E6]dA[E6]dA[E6]dA
add redundant bits to 12 [E]dA[E]dA[E]dA[E]dA
Thus the output is a [E6]dA[E6]dA[E6]dA[E6]dA

$150 \text{ kg} + 50 \text{ kg} + 100 \text{ kg} = 200 \text{ kg}$ max.

• Log-vertebrates to be kept after
removal of all other animals.

After learning code ("probabilistic" data)

The above is 90 miles, probably.

(F) d, (G) d, (H) d, (I) d, (J) d, (K) d, (L) d, (M) d, (N) d, (O) d

~~and now ("R", "P", "C", "H", "L", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z")~~

15. (libation) o' 2002-11-11

(Tran.) ~~the~~ ~~is~~ ~~primed~~ + ~~the~~ ~~one~~

2. All the above-mentioned species have a well-defined period of activity.

Result: Sender and receiver program for hamming code

script was executed and got the output:

concept was shown (not ~~difficult~~ but bad)

Scanned with CamScanner