Exp. No:6

19/9/24

HAMMING CODE

Aim:

To write a program to implement error detection and correction using Hamming code.

Sender program:

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

Receiver program:

- Receiver program should read the input from channel file.
- Apply hamming code on the binary data to check for erros.
- If there is an error, display the position of the error.
- Else remove the redundant bits and convert the binary data to ascii and display the output.

Code :

```
# sender.py

def text_to_binary(text):
    return ''.join(format(ord(i), '08b') for i in text)

def calc_parity_positions(m):
    r = 0
    while 2**r < m+r+1 :
        r += 1
    return r

def insert_parity_bits(data, r):
    n = len(data)
    result = ['0'] *(n +r)
    j=0
    for i in range (1, len(result) +1);
        if (i & (i-1) ==0):
            continue
        result [i-1] = data[j]
        j+=1
    return ''.join(result)

def set_parity_bits(data, r):
    n = len(data)
    result = list(data)
    for i in range(d):
        idx = 2**i -1
        parity = 0
        for j in range (1, n+1):
            if j & (2**i) != 0:
                parity ^= int (result[j-1])
        result[idx] = str(parity)
    return ''.join(result)

def hamming_encode(text):

    binary_data = text_to_binary(text)
    m = len(binary_data)
    r = calc_parity_positions(m)
    data_with_parity = insert_parity_bits(binary_data, r)
    encoded_data = set_parity_bits(data_with_parity, r)
    return encoded_data

def save_to_channel(encoded_data):
    with open("channel.txt", "w") as f:
        f.write(encoded_data)

text = input("Enter text to send: ")
encoded_data = hamming_encode(text)
save_to_channel(encoded_data)
print(f"Encoded data saved to 'channel.txt':
        {encoded_data}")


# receiver.py

def calc_parity_positions(m):

    r = 0
    while 2**r < m+r+1:
        r += 1
    return r
```

```
def read_from_channel():
    with open("channel.txt", "r") as f:
        return f.read()

def detect_and_correct_error(data, r):
    n = len(data)
    result = list(data)
    error_pos = 0

    for i in range(r):
        idx = 2**i - 1
        parity = 0
        for j in range(1, n+1):
            if j & (2**i) != 0:
                parity ^= int(result[j-1])
        error_pos += parity * (2**i)

    if error_pos > 0:
        print(f"Error detected at position: {error_pos}")
        result[error_pos-1] = '1' if result[error_pos - 1] == '0'
                                                        else '0'
        print(f"Corrected data: {''.join(result)}")
    else:
        print("No errors detected.")
    return ''.join(result)
```

```
def remove_parity_bits (data, r):
    n = len (data)
    result = [ ]
    for i in range (1, n+1):
        if i & (i-1) == 0:
            continue
        result.append (data[i-1])
    return ''.join (result)

def binary_to_text(binary):
    text = ''.join ([chr(int (binary[i:i+8], 2)) for i
           in range (0, len(binary), 8)])
    return text

def hamming_decode ():
    encoded_data = read_from_channel ( )
    m = len (encoded_data)
    r = calc_parity_positions (m - len ([1 for i in range(m)
                                 if i & (i+1) == 0]))
    corrected_data = detect_and_correct_error (encoded_data, r)
    data_without_parity = remove_parity_bits (corrected_data, r)
    decoded_text = binary_to_text (data_without_parity)
    return decoded_text


decoded_text = hamming_decode( )
print (f"Decoded text : {decoded_text}")
```

Input :

Enter text to send : 1011

Output :

```
Encoded data saved to 'channel.txt' : 01000111000100 101
                                      0000001100100111
                                      0001

No errors detected .
Decoded text : 1011
```

19/11

9/10

Result :

Thus the program to implement error detection and correction using hamming code has been implemented successfully .