Niyati Savant
C31-TE
2103156

Experiment-02

31/7/23

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 ditectee and correct the errors that occurs when data is moved from sender to receiver. It was developed by R.W Hamming for error correction. Redundant bits (Extra binary bits) are generated and added to the information-carring bits of data transfer to ensure that no bits were lost.

The formula to find number of redundant bits is - $2^r \geq m + r + 1$

where m: data bit
r: redundant bit

The value of these parity/redundant bits is such that an even or odd parity is maintained. In case of even parity, if the count of number of 1's is odd, parity bit is 1 else 0. For odd parity, if the count g 1's is odd, parity bit is 0 else 1.

The redundant bits are placed at positions that are numbered corresponding to the power of 2. i.e 1, 2, 4, 8 ..... and so the data bits take up positions 3, 5, 6, 7 . . .

TO assign the bit value for a redundant bit we do the following:

- Parity bit P1 checks data bits ~~on~~ which have 1 in the LSB ie (001, 011, 101, 111 .. etc)
Thus P1 checks bits 1, 3, 5, 7, 9,11 etc.
- Parity bit P2 checks bits which have 1 in the second least significant position (010, 011, 110, 111 e
Thus P2 checks bits 2,3,6,7 etc
- Parity bit P4 checks bits with 1 in the ~~of~~ binary at third least significat place ie (100, 101, 110, 111) ie ~~bit~~ location 4,5,6,7 ets.

When the data is send, these parity bits are rechecked and if they voilate their parity their value is set to 1 else set to 0. These parity bits are then checked in reverse order to find the error bit which is inverted to get the correct code.

Some of its features are

- It can detect and correct single bit errors but only detect double bit errors.
- It is a relatively simple and efficient technique which makes it ideal for low-power and low-bandwidth communication networks
- Has wide varity of applications including telecommunications, computer networks & data storage systems

iyati Savant
C31 - TE
2103156

31|7|23

Example - Consider the data "1011" to be shared

As there are 4 data bits, we need 3 redundant bits since

$2^3 \geqslant 3 + 4 + 1$ (as $8 \geqslant 8$)

Thus parity bits are P1, P2 and P4
The values of these parity bits are -

P1 - Check positions 1, 3, 5, 7 ie 1 at 3 positions. For Even parity P1 is set to 1

P2 - Check positions 2, 3, 6, 7 ie 1 at 2 positions and to maintain parity P2 is set to 0

P4 - Check positions 4, 5, 6, 7 ie 1 at 2 positions and P4 is 0

Thus, hamming code (has m+r ie 4+3 = 7 bits):

D7	D6	D5	P4	D3	P2	P1
1	0	1	0	1	0	1

Suppose, due to some error, the data bit D7 was change and the data ~~sent that was~~ received was "0010101".

At the receiver end, the parity bits are checked again.

For P1, ~~there are~~ we check bits 1,3,5,7, as there are 3 1's, the bit value is set to 1.

For P2 we check bits 2,3,6,7 and there ~~are~~ is One 1 thus bit value is set to 1.

For P4, we check 4,5,6,7 there is only one 1 and bit value is 1.

Thus there is error in bit 111 ie bit 7.
Therefore corrected code is 1010101.

A

2/1/81

Program: Hamming Code For Error Detection And Correction

Code:

```
#include <stdio.h>

#include <math.h>

int total_bits,hamming[20],received_data[20];


int pow_of_two(int x)

{

   for(int i=0;i<10;i++)

   {

      if(pow(2,i)==x)

         return 1;

   }

   return 0;

}


int parity_bit(int a)

{

   int i=a,count_no=a;

   int b=0;

   int bit[20];

   int arr_len = total_bits-a+1,count_one=0; //9-2+1=8

   for(b=0;b<arr_len ;b++,i++)

      bit[b]=i;

   b=0;

   while(b<arr_len)

   {

      if(count_no!=0)

      {

         if(hamming[bit[b]]==1)
```

```
            {
                count_one++;
            }
            count_no--;
            b++;
        }
        else
        {
            count_no=a;
            b+=a;
        }
    }
    if(count_one%2==0)
        return 0;
    else
        return 1;
}

int bit_checker(int a)
{
    int i=a,count_no=a;
    int b=0;
    int bit[20];
    int arr_len = total_bits-a+1,count_one=0; //9-2+1=8
    for(b=0;b<arr_len ;b++,i++)
        bit[b]=i;
    b=0;
    while(b<arr_len)
    {
        if(count_no!=0)
        {
            if(received_data[bit[b]]==1)
            {
                count_one++;
            }
            count_no--;
            b++;
        }
        else
        {
            count_no=a;
            b+=a;
        }
    }
    if(count_one%2==0)
        return 0;
    else
        return 1;
}

int main()
{
    int p,i,input_data[15],n,k,j,x,error[5],error_bit;
    for(i=0;i<5;i++)
        error[i]=0;
    printf("Niyati’s program for Hamming Code \n");
    printf("Enter the number of data bits: ");
    scanf("%d",&n);

    printf("Enter data bits- \n");
    for(i=1;i<=n;i++)
        scanf("%d",&input_data[i]);
```

```
/*No of Parity Bits */
for(i=1;i<10;i++)
{
   if(pow(2,i)>=(n+i+1))
   {
      p=i;
      break;
   }
}
printf("\nNo. of parity bits = %d \n",p);

// Finding Hamming Code
k=1;
total_bits = n+p;
for(i=1;i<=total_bits;i++)
{
   hamming[i]=111;
}
printf("Hamming Code is --\n");
for(i=total_bits;i>=1;i--)
{
   //Find Parity bits
   if (pow_of_two(i))
   {
      printf(" P%d",i);
      hamming[i]=parity_bit(i);
   }
   else
   {
        hamming[i]=input_data[k];
        printf(" D%d",i);
        k++;
    }
}
printf("\n");
for(i=total_bits;i>=1;i--)
    printf("  %d",hamming[i]);

printf("\nEnter received code:");
for(i=total_bits;i>=1;i--)
    scanf("%d",&received_data[i]);



printf("\nError bit in binary is: ");
k=5;
for(i=total_bits;i>=1;i--)
{
    if (pow_of_two(i))
    {
        error[k]=bit_checker(i);
        k--;
    }
}
for(i=5;i>k;i--)
    printf("%d",error[i]);
    //k=2

error_bit=0;
for(i=k+1,j=0;i<=5;i++,j++)
{
        error_bit +=error[i]*pow(2,j);
    }
    printf(" i.e in binary bit D%d ",error_bit);


    if (received_data[error_bit]==0)
        received_data[error_bit]=1;
    else
        received_data[error_bit]=0;
    printf("\nThus the corrected code is : ");
    for(i=total_bits;i>=1;i--)
        printf("%d",received_data[i]);


    return 0;
}
```

Output:

```
Niyati's program for Hamming Code
Enter the number of data bits: 4
Enter data bits-
1 0 1 1
No. of parity bits = 3
Hamming Code is --
 D7 D6 D5 P4 D3 P2 P1
  1   0   1   0   1   0   1
Enter received code:0 0 1 0 1 0 1
Error bit in binary is: 111 i.e in binary bit D7
Thus the corrected code is : 1010101
```