PROGRAMLAMA PLATFORMLARI
FİNAL PROJESİ

TAKIM

191216002 - Furkan Kaya
191216008 - Ali Murat Tava
191216018 - Tolgahan Şişman
191216022 - Kaan İnce

DANIŞMAN

Ahmad Hasan Abed AL KHAS

28 Mayıs 2021

KOD BÖLÜMLERİ

Değişken ve Sabitler:

Döküman üzerinde verilen sabitleri ve kullandığımız değişkenleri dosyanın en üst bölümünde tanımladık.

```
function Main()
    p1 = [1,0,0,1,1,0,1,0,1,1,1];
    p2 = [1,1,0,1,0,1,1,1,1,0,0];
    p3 = [0,1,1,0,1,0,1,1,1,1,0];
    p4 = [0,0,1,1,0,1,0,1,1,1,1];

    A = [p1;p2;p3;p4];
    AT = transpose(A);
    G = [eye(11),AT];
    H = [A, eye(4)];
    frame = [0,1,1,1,1,1,1,1,1,0];
    sendrom = H * transpose(eye(15));
    addedZeroLen = 0;
```

Fonksiyonlar:

Dosyanın bir sonraki bölümünde kullanılan fonksiyonları yerleştirdik. Sağdaki görselde görebilirsiniz.

```
function itemP = ReturnItemProbMap(M, len)...

function itemCount = ReturnItemCountMap(1)...

function flattenArr = ReturnArrFlat(arr)...

function withBurstError = ReturnBrokenArray(arr)...

function indexNumber = GetIndexNumber(arr, item)...

function frameDeleted = DeleteBeforeFrame(arr, frame)...

function fixedArr = FixErrorInArr(arr, errors)...


function [finalMessage, dict] = Transmitter(message)...

function recived = Receiver(message, dict)...

function addedParasite = AddParasite(message)...
```

```
%% USAGE 0=text, 1=image
isImg = 1;

if (isImg == 1)
    msg = imread('istinye_universitesi_MYO.jpg');
    [rowS, colS, zdim] = size(msg);
    input = reshape(msg, 1, rowS * colS * zdim);
else
    msg = importdata("Mesaj.txt");
    input = cell2mat(msg);
end

[final, dict] = Transmitter(input);
finalWparasite = AddParasite(final);
recivedMessage = Receiver(finalWparasite, dict);

if (isImg == 1)
    img2 = reshape(recivedMessage, rowS, colS, zdim);
    img2 = uint8(img2);
    image(img2)
else
    cell2mat(recivedMessage)
end
```

Kullanım:

Projenin kullanımını 3 ayrı fonksiyona indirgedik. Veri iletimini simüle ettik. Transmitter fonksiyonu ile veriyi çıkartıyoruz. AddParasite fonksiyonu ile parazit gelmesini simüle ediyoruz ve receiver fonksiyonu ile parazitleri temizleyip veriyi ekranda gösteriyoruz. Projemizi test etmek için isImage Değişkenini 0 veya 1 ataması gerçekleştirebilirsiniz.

FONKSİYONLAR

ReturnItemCountMap():

Bu fonksiyonda verideki her bir sembolün sayısını hesaplayıp itemCount sözlüğü ile geri döndürüyoruz.

```
function itemCount = ReturnItemCountMap(l, firstElement)
    % This function count every symbol in given list.
    if (isa(firstElement, 'char'))
        itemCount = containers.Map('KeyType','char', 'ValueType', 'double');
    else
        itemCount = containers.Map('KeyType','uint32', 'ValueType', 'double');
    end
    for m = l
        if (isKey(itemCount, m))
            itemCount(m) = itemCount(m) + 1;
            continue;
        end
        itemCount(m) = 1;
    end
end
```

ReturItemProbMap():

Bu fonksiyonda her bir sembolün olasılığını hesaplıyoruz.

```
function itemP = ReturnItemProbMap(M, len, firstElement)
    % This function return probability dict based on symbol count.
    if (isa(firstElement, 'char'))
        itemP = containers.Map('KeyType','char', 'ValueType','double');
    else
        itemP = containers.Map('KeyType','uint32', 'ValueType','double');
    end
    for m = keys(M)
        thekey = m{1};
        count = double(M(thekey));
        itemP(thekey) = double(count/len);
    end
end
```

ReturnArrFlat():

Verdiğimiz diziyi 1xN boyutunda geri döndürmek için yazdığımız fonksiyon.

```
function flattenArr = ReturnArrFlat(arr)
    flattenArr = [];
    for k = arr
        flattenArr = [flattenArr, transpose(k(1:end))];
    end
end
```

Transmitter():

Gelen mesajın olasılık sözlüğünü ve sayı sözlüğünü çıkarttıktan sonra huffmandict ile huffman sözlüğünü oluşturuyoruz. huffmanenco ile huffman kaynak kodlaması yapıyoruz. Ardından her bir bloğun 11 bitten oluşması için uzunluğunu 11'e bölünebilir hale getiriyoruz. Mesajı G matrisi ile çarpıp düzleştiriyoruz. Ardından başına frame ekleyip son mesajı oluşturuyoruz.

```
function [finalMessage, dict] = Transmitter(message)
    itemCounts = ReturnItemCountMap(message);
    itemPs = ReturnItemProbMap(itemCounts, length(message));

    [dict, avglen] = huffmandict(keys(itemPs), cell2mat(values(itemPs)));

    huffEncoded = huffmanenco(message, dict);
    if (mod(length(huffEncoded),11) ~= 0)
        remain = 11 - mod(length(huffEncoded), 11);
        addedZeroLen = remain;
        huffEncoded = [huffEncoded, zeros(1, remain)];
    end
    reshapedArr = reshape(huffEncoded, length(huffEncoded)/11, 11);
    encodedMessage = mod(reshapedArr*G, 2);
    interLeavedMessage = ReturnArrFlat(encodedMessage);
    finalMessage = [frame, interLeavedMessage];
end
```

ReturnBrokenArray():

8. maddede belirtilen 235-249 arasındaki bitleri ters çevirmek için kullandığımız fonksiyon.

```
function withBurstError = ReturnBrokenArray(arr)
    for i=235:249
        arr(i) = mod(arr(i)+1, 2);
    end
    withBurstError = arr;
end
```

AddParasite():

İletişim kanalında ortaya çıkabilecek bazı hataları simüle ettiğimiz fonksiyondur. Başı random int değerler ekleyip daha sonrasında 8. maddeyi de gerçekleştirerek diziyi geri döndürüyoruz.

```
function addedParasite = AddParasite(message)
    randomInt = randi([1 100]);
    randomBinary = de2bi(randomInt);
    finalWrandom = [randomBinary, message];
    addedParasite = ReturnBrokenArray(finalWrandom);
end
```

DeleteBeforeFrame():

Verilen dizi içerisindeki frame'i bulup frame ve öncesini silerek geri döndürüyoruz.

```
function frameDeleted = DeleteBeforeFrame(arr, frame)
    index = 1;
    frameLen = length(frame);
    while(1)
        if(index>=length(arr))
            break;
        end
        if(arr(index:index+frameLen-1)==frame)
            index = index+frameLen;
            break;
        end
        index=index+1;
    end
    frameDeleted = arr(index:end);
end
```

GetIndexNumber():

Dizinin içindeki item'ın index numarasını döndüren fonksiyon.

```
function indexNumber = GetIndexNumber(arr, item)
    indexNumber = 0;
    for a=arr
        indexNumber=indexNumber+1;
        if(a==item)
            break;
        end
    end
end
```

FixErrorInArr():

Dizinin içinde bulunan hataları bulup, düzelten fonksiyon.

```
function fixedArr = FixErrorInArr(arr, errors)
   counter = 1;
   for k=errors
       if(ismember(1,k))
           erroredBit = GetIndexNumber(sendrom, k);
           erroredRow = counter;
           arr(erroredRow, erroredBit) = mod(arr(erroredRow, erroredBit)+1,2);
       end
       counter=counter+1;
   end
   fixedArr=arr;
end
```

Receiver():

Gelen mesajın frame'ini ve hatalarını temizleyerek paradi sütunlarını silip huffmandeco kullanarak kod çözme gerçekleştirdik ve orjinal veriyi geri döndürdük.

```
function recived = Receiver(message, dict)
    frameDeleted = DeleteBeforeFrame(message, frame);
    reshaped2 = reshape(frameDeleted, length(frameDeleted)/15, 15);
    errors = mod(H*transpose(reshaped2),2);
    fixed = FixErrorInArr(reshaped2, errors);

    %delete paradi columns
    fixed(:,[12, 13, 14, 15]) = [];
    fixed = ReturnArrFlat(fixed);
    fixed = fixed(1:end-addedZeroLen);

    recived = huffmandeco(fixed, dict);
end
```