Experiment No-06

Experiment Name: Check Optimality of Huffman Code

Theory:

To check the optimality of a Huffman code, you can use the Kraft inequality, which states that the sum of the code word lengths (in bits) raised to the power of -1 must be less than or equal to 1. If a code satisfies the Kraft inequality, it is considered optimal.

Another way to check the optimality of Huffman code is to verify that the code is prefix-free, meaning no code word is a prefix of any other code word. A prefix-free code is always optimal.

A third way to check the optimality of Huffman code is by comparing the average length of the code to the entropy of the source. In all cases, if the code is not optimal, you can use the standard algorithm for constructing a Huffman code to generate a new, optimal code.

In this source code, I am using Kraft inequality check for proving optimality of Huffman code.

Source Code

```
clc;
clear all;
close all;
X = {'a', 'b', 'c', 'd', 'e'};  % Symbol set
disp('Symbols:')
disp(X)
frequency = [25 25 20 15 15];  % Corresponding frequencies
disp('Corresponding frequency of symbols:')
disp(frequency)
if length(frequency) ~= length(X)
    disp('Equal length is expected')
else
    probability_X = frequency / sum(frequency);  % Probability of symbols
    disp('Corresponding probability of symbols:')
    disp(probability_X)
   % Create a dictionary for Huffman codes
    [dictionary, avg_len] = huffmandict(X, probability_X);
    disp('Huffman codes:')
    disp(dictionary)
    % Calculate inequity (efficiency)
    inequity = 0;
    for i = 1:length(X)
        disp(['Symbol: ', dictionary{i, 1}, ', Huffman Code: ', dictionary{i, 2}])
        Li = length(dictionary{i, 2});
        inequity = inequity + (probability_X(i) * Li);  % Expected length of code
    end
```

```
    disp('Inequity (Expected Length):')
    disp(inequity)

    % Check if the code is optimal
    if inequity <= log2(length(X))
        disp('This is an optimal code.')
    else
        disp('Not optimal.')
    end
end
```

Output:

Symbols: 'a', 'b', 'c', 'd', 'e'

Corresponding frequency of symbols: 25 25 20 15 15

Corresponding Probability of symbols: 0.2500 0.2500 0.2000 0.1500 0.1500

Huffman codes:

A

1 0

B

0 1

C

1 1

D

0 1 0

E

0 0 0

Inequity: 0.4

Thus, this is an optimal code.