**DIGITAL IMAGE PROCESSING**

Homewprk #3

**Huffman Method**

We are asked for implementing Huffman theory to create a binary tree of the source symbols, using the probabilities.

Firstly, we try using string character ‘Adhi’ as input and the probability distribution values as follows [10 19 30 40 50]. Then,

The result :

num\_bits =

2

Character Probability:

A -->65b

d -->100b

h -->104b

i -->105b

total =

374

sorted\_tree =

'ihdA' 'ih' 'dA' 'i' 'h' 'd' 'A'

sorted\_tree\_prob =

374 209 165 105 104 100 65

codeBook =

'ihdA' ''

'ih' '1'

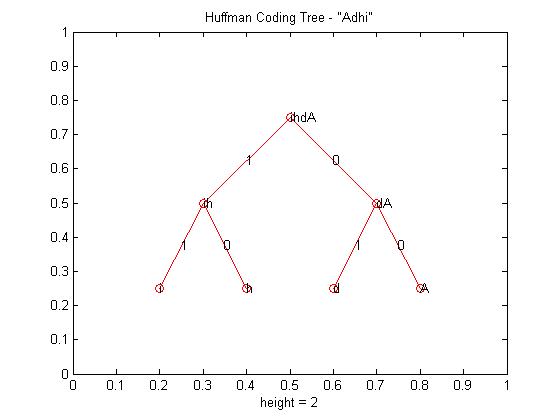
'dA' '0'

'i' '11'

'h' '10'

'd' '01'

'A' '00'



**Figure 1**. Huffman Tree Code of ‘Adhi’.

We also try using longer character such ‘Prabancoro Adhi’, and the result are;

num\_bits =

4

Character Probability:

P -->80

r -->114

a -->97

b -->98

a -->97

n -->110

c -->99

o -->111

r -->114

o -->111

-->32

A -->65

d -->100

h -->104

i -->105

total =

1437

sorted\_tree =

Columns 1 through 7

'oonihdcbA aaPrr' 'oonihdcb' 'A aaPrr' 'ooni' 'hdcb' 'A aaP' 'rr'

Columns 8 through 19

'oo' 'ni' 'hd' 'cb' 'A a' 'aP' 'r' 'r' 'o' 'o' 'n' 'i'

Columns 20 through 29

'h' 'd' 'c' 'b' 'A ' 'a' 'a' 'P' 'A' ' '

sorted\_tree\_prob =

Columns 1 through 7

1437 838 599 437 401 371 228

Columns 8 through 14

222 215 204 197 194 177 114

Columns 15 through 21

114 111 111 110 105 104 100

Columns 22 through 28

99 98 97 97 97 80 65

Column 29

32

codeBook =

'oonihdcbA aaPrr' ''

'oonihdcb' '1'

'A aaPrr' '0'

'ooni' '11'

'hdcb' '10'

'A aaP' '01'

'rr' '00'

'oo' '111'

'ni' '110'

'hd' '101'

'cb' '100'

'A a' '011'

'aP' '010'

'r' '001'

'r' '0011'

'o' '1111'

'o' '11111'

'n' '1101'

'i' '1100'

'h' '1011'

'd' '1010'

'c' '1001'

'b' '1000'

'A ' '0111'

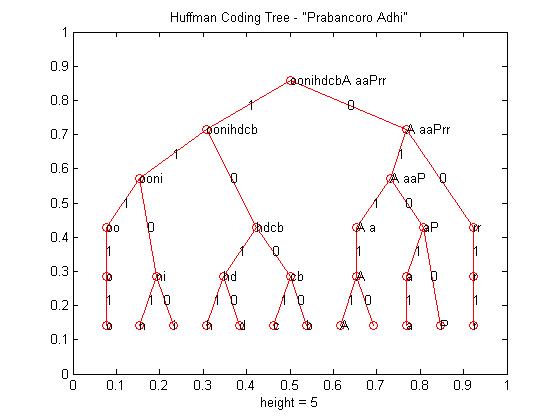
'a' '0101'

'a' '01011'

'P' '0100'

'A' '01111'

' ' '01110'



**Figure 2**. Huffman Tree Code of ‘Prabancoro Adhi’.

List Program:

%The original source code is written by Jason

%This code has been edited by me

clc;

clear all;

close all;

% Define the character string

my\_str = 'Prabancoro Adhi';

auto\_prob = 1;

if (auto\_prob == 1)

% Automatically calculate the probability distribution

% Get ASCII version of each character

% Each ASCII value represents the probability of finding the character

prob\_dist = double(my\_str);

else

% Manually define the probability distribution

prob\_dist = [10 19 30 40 50];

end

num\_bits = ceil(log2(length(prob\_dist)))

disp('Character Probability:');

for i = 1:length(prob\_dist)

display(strcat(my\_str(i),' --> ',num2str(prob\_dist(i))));

end

total = sum(prob\_dist)

for i = 1:length(my\_str)

sorted\_str{i} = my\_str(i);

end

% Save initial set of symbols and probabilities for later use

init\_str = sorted\_str;

init\_prob = prob\_dist;

sorted\_prob = prob\_dist;

rear = 1;

while (length(sorted\_prob) > 1)

% Sort probs

[sorted\_prob,indeces] = sort(sorted\_prob,'ascend');

% Sort string based on indeces

sorted\_str = sorted\_str(indeces);

% Create new symbol

new\_node = strcat(sorted\_str(2),sorted\_str(1));

new\_prob = sum(sorted\_prob(1:2));

% Dequeue used symbols from "old" queue

sorted\_str = sorted\_str(3:length(sorted\_str));

sorted\_prob = sorted\_prob(3:length(sorted\_prob));

% Add new symbol back to "old" queue

sorted\_str = [sorted\_str, new\_node];

sorted\_prob = [sorted\_prob, new\_prob];

% Add new symbol to "new" queue

newq\_str(rear) = new\_node;

newq\_prob(rear) = new\_prob;

rear = rear + 1;

end

tree = [newq\_str,init\_str];

tree\_prob = [newq\_prob, init\_prob];

% Sort all tree elements

[sorted\_tree\_prob,indeces] = sort(tree\_prob,'descend');

sorted\_tree = tree(indeces);

parent(1) = 0;

num\_children = 2;

for i = 2:length(sorted\_tree)

% Extract my symbol

me = sorted\_tree{i};

% Find my parent's symbol (search until shortest match is found)

count = 1;

parent\_maybe = sorted\_tree{i-count};

diff = strfind(parent\_maybe,me);

while (isempty(diff))

count = count + 1;

parent\_maybe = sorted\_tree{i-count};

diff = strfind(parent\_maybe,me);

end

parent(i) = i - count;

end

treeplot(parent);

title(strcat('Huffman Coding Tree - "',my\_str,'"'));

display(sorted\_tree)

display(sorted\_tree\_prob)

[xs,ys,h,s] = treelayout(parent);

text(xs,ys,sorted\_tree);

for i = 2:length(sorted\_tree)

% Get my coordinate

my\_x = xs(i);

my\_y = ys(i);

% Get parent coordinate

parent\_x = xs(parent(i));

parent\_y = ys(parent(i));

% Calculate weight coordinate (midpoint)

mid\_x = (my\_x + parent\_x)/2;

mid\_y = (my\_y + parent\_y)/2;

% Calculate weight (positive slope = 1, negative = 0)

slope = (parent\_y - my\_y)/(parent\_x - my\_x);

if (slope > 0)

weight(i) = 1;

else

weight(i) = 0;

end

text(mid\_x,mid\_y,num2str(weight(i)));

end

for i = 1:length(sorted\_tree)

% Initialize code

code{i} = '';

% Loop until root is found

index = i;

p = parent(index);

while(p ~= 0)

% Turn weight into code symbol

w = num2str(weight(index));

% Concatenate code symbol

code{i} = strcat(w,code{i});

% Continue towards root

index = parent(index);

p = parent(index);

end

end

codeBook = [sorted\_tree', code']