QUESTION 6

Code

```
% Huffman Block Coding for IID Binary RV's

% CASES: p = 0.35, p' = 0.05

% BLOCK LENGTH = M = 1,2,3

% GENERATE RANDOM SEQUENCES OF VARIABLE n

% ENCODE, CALCULATE AVERAGE CODE LENGTH PER BINARY SYMBOL

% AVERAGE OVER 20-40 REALIZATIONS OF BINARY SEQUENCES

% COMPARE TO ENTROPY BOUND

% DISCUSS RESULTS

% We assume we are performing Huffman Binary Coding


M_vec = [1 2 3];

p_vec = [0.35 0.05];                    % probability of 1, as such, in both

n_vec = [6 12 18 24 30 36 42 48 54 60];

% What will be n values?

% M=1, n=6,12,18,24,30,36,42,48,54,60
% M=2, n=6,12,18,24,30,36,42,48,54,60
% M=3, n=6,12,18,24,30,36,42,48,54,60

% Once a value of M and p selected, we know n range, and for each n value,
% the trial will be performed 30 times (between 20 to 40, 30 chosen)
```

```
% Huffman Coding has been done, a priori, by hand, and will be utilized
% directly here using input to code dictionary

% Coding from scratch for Huffman, in MATLAB, could've been done, however,
% it seemed time consuming despite being simple. Thus Coding done on paper,
% and here we will use those dictionaries directly.


%------------------------------------------------------------------------

% p = 0.35

% M = 1

% input                   huffman code
% 0 ----------------------    0
% 1 ----------------------    1

% M = 2

% input                   huffman code
% 00 --------------------     0
% 01 --------------------     10
% 10 --------------------     110
% 11 --------------------     111

% M = 3

% input                   huffman code
% 000 --------------------    00
```

```
% 000 --------------------    010
% 010 --------------------    011
% 010 --------------------    100
% 101 --------------------    110
% 101 --------------------    111
% 111 --------------------    1010
% 111 --------------------    1011


%------------------------------------------------------------------------

% p' = 0.05

% M = 1

% input                     huffman code
% 0 -----------------------   0
% 1 -----------------------   1

% M = 2

% input                     huffman code
% 00 --------------------     0
% 01 --------------------     10
% 10 --------------------     110
% 11 --------------------     111

% M = 3

% input                     huffman code
% 000 --------------------    0
% 000 --------------------    100
% 010 --------------------    101
```

```
% 010 --------------------   110
% 101 --------------------   11100
% 101 --------------------   11101
% 111 --------------------   11110
% 111 --------------------   11111

%------------------------------------------------------------------------

% Encoding and obtaining the required quantities

p1_1 = [0 1;
        1 1];

p1_2 = [0 1;
        1 2;
        2 3;
        3 3];

p1_3 = [0 2;
        1 3;
        2 3;
        3 3;
        4 3;
        5 3;
        6 4;
        7 4];



p2_1 = p1_1;

p2_2 = p1_2;

```

```
p2_3 = [0 1;
        1 3;
        2 3;
        3 3;
        4 5;
        5 5;
        6 5;
        7 5];


dict_cell = cell(2,3);

dict_cell{1,1} = p1_1;
dict_cell{1,2} = p1_2;
dict_cell{1,3} = p1_3;

dict_cell{2,1} = p2_1;
dict_cell{2,2} = p2_2;
dict_cell{2,3} = p2_3;


for i = 1:2
    p = p_vec(1,i);
    H = (-p*log(p)-(1-p)*log(1-p))/log(2);
    for j = 1:3
        M = j;
        vec_len = zeros(1,10);
        for k = 1:10
            n = 6*k;
            len_ovrl = 0;
            for r = 1:30
                seq = rand(1,n);
                seq = (seq <= p);
```

```
                seq = (seq <= p);
                pieces = n/M;
                tot_len = 0;
                for t = 1:pieces
                    strt = 1+(t-1)*M;
                    ed = strt + M - 1;
                    foc = seq(1,strt:ed);
                    dec = bi2de(foc,'left-msb');
                    loc_dict = dict_cell{i,j};
                    code_len = loc_dict(dec+1,2);
                    tot_len = tot_len + code_len;
                end
                len_ovrl = len_ovrl + tot_len;
            end
            avrg_len = len_ovrl/30;
            avrg_len_sym = avrg_len/n;
            vec_len(1,k) = avrg_len_sym;
        end
        disp('The p value is')
        disp(p)
        disp('The M value is')
        disp(M)
        disp('The entropy rate or entropy currently for the p value is')
        disp(H)
        disp('For n = 6,12,..,60 we obtain the length per symbol vector as follo
        disp(vec_len)
    end
end
```

Results

>> HW8_6

The p value is

0.350000000000000

The M value is

1

The entropy rate or entropy currently for the p value is

0.934068055375491

For n = 6,12,..,60 we obtain the length per symbol vector as follows (for each n, this is the averaged value of 30 repeats)

1 1 1 1 1 1 1 1 1 1

The p value is

0.350000000000000

The M value is

2

The entropy rate or entropy currently for the p value is

0.934068055375491

For n = 6,12,..,60 we obtain the length per symbol vector as follows (for each n, this is the averaged value of 30 repeats)

Columns 1 through 9

0.983333333333333   0.969444444444444   0.974074074074074   0.976388888888889
1.002222222222222   0.975925925925926   0.980952380952381   0.977083333333333
0.951234567901235

Column 10

0.941666666666667

The p value is

0.350000000000000

The M value is

3

The entropy rate or entropy currently for the p value is

0.934068055375491

For n = 6,12,..,60 we obtain the length per symbol vector as follows (for each n, this is the averaged value of 30 repeats)

Columns 1 through 9

0.994444444444444   0.922222222222222   0.951851851851852   0.930555555555555
0.944444444444444   0.953703703703704   0.954761904761905   0.950694444444444
0.950000000000000

Column 10

0.953333333333333

The p value is

0.050000000000000

The M value is

1

The entropy rate or entropy currently for the p value is

0.286396957115956

For n = 6,12,..,60 we obtain the length per symbol vector as follows (for each n, this is the averaged value of 30 repeats)

1   1   1   1   1   1   1   1   1   1

The p value is

0.050000000000000

The M value is

2

The entropy rate or entropy currently for the p value is

0.286396957115956

For n = 6,12,..,60 we obtain the length per symbol vector as follows (for each n, this is the averaged value of 30 repeats)

Columns 1 through 9

0.583333333333333   0.577777777777778   0.570370370370370   0.579166666666667
0.562222222222222   0.581481481481481   0.576984126984127   0.565972222222222
0.576543209876543

Column 10

0.570000000000000

The p value is

0.050000000000000

The M value is

3

The entropy rate or entropy currently for the p value is

0.286396957115956

For n = 6,12,..,60 we obtain the length per symbol vector as follows (for each n, this is the averaged value of 30 repeats)

Columns 1 through 9

0.422222222222222   0.427777777777778   0.500000000000000   0.494444444444444
0.477777777777778   0.457407407407407   0.466666666666667   0.456944444444444
0.451851851851852

Column 10

0.466666666666667

>>

The results have been pasted for Result Tab in MATLAB due to the length.

A few notes

p=0.35 shows a pretty good characteristic of moving towards the entropy rate as n increases

p=0.05 for the value of n we selected has the average length of codeword per symbol much farther from the entropy rate. The value of n, if we would have increased, would have shown better traits for p=0.05.

M=1 is a useless case since Huffman binary encoding is just a copy paste of the binary input.

Thus, we did, to a certain extent verify that as the value of n increases the Huffman code generates a code that has a tendency of moving towards the entropy rate bounds on average length of code per symbol.

Increase in M makes the average codeword length per symbol move closer and closer to the entropy rate. Thus, as M increases overall code optimality increases.

This again shows that to achieve true optimality via Huffman codes, the block length should be as large as possible. However, doing so is computationally expensive, which is why coding strategies such as arithmetic coding are so widely used to encode sequences of random variables.]

These are the results and the comments.