INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Project: Source and Channel Coding

Instructions:

• The sub-parts of each question are created in a graded manner so as to aid you in writing the code.

• Please use Python (whatever extension of Python is applicable) or MATLAB.

• Your code should be clean and should carry helpful comments at every stage (please include comments about any special functions you use as it is useful for the reader).

• Please add helpful English statements when the code runs to enable the user to understand what (s)he is giving as input and how it should be given.

• Both code and video should be submitted. All members of the team should have their equal video/audio contribution in the video. Only audio is not allowed. Zoom can be used for recording as per instructions similar to Video Summaries. Special slides are not necessary but can be prepared for aiding your video presentation. Other technical details regarding the video contents are in the question.

Information and Communication: IIIT Hyderabad

Instructor: Dr.Prasad Krishnan

Problem I : Huffman Encoding (15 marks)

1. **Generate Source Files**: Create three text source files of the approximately same size (5-10 KB or 1-2 pages of text) having the following properties:

(a) File 1: Contains only your name (or any one word) multiple times

(b) File 2: Repeat a paragraph of around 30 words multiple times

(c) File 3: Normal 1-2 pages of paragraphs (english text) with no repetitions

We repeat that the files need to be of same size roughly. This will help you to compare the compression in the three scenarios.

2. **Count Frequencies**: Examine the source file's contents and count the number of occurrences of each character. *Note that all characters must be included in the frequency table, including spaces, and any punctuation.* This will give you a probability distribution.

3. **Construct Huffman code**: Construct a Huffman code for each of the probability distributions obtained from the frequency tables. Output the symbols along with corresponding codewords of the Huffman code. (For each symbol, the codeword should be displayed). 4. **Encode data**: Using Huffman Algorithm encode the source file's contents into another file. Display the size of the encoded file.

5. **Decode Data**: From the encoded file, decode the original source file by extracting the symbols from the codewords. Check the decoded output with the original file (via a code snippet) and ensure there is no error in decoding.

6. **Recording your algorithm and observations via video:**
The video for this file should contain the flow of your code, example executions, description of your outputs, and the inferences you derived by comparing the lengths of the compressed sequence, time taken for compression/decoding, etc. Also, note if you tried anything else apart from the problem statement, if some part of the problem could not be executed by you (what were the difficulties, what you tried to overcome them?). Any other observations also you are free to record. The whole video can be between 10-15 minutes.

Problem II : Error probability of Random codes over a BSC Channel (15 marks)

Perform the steps below for the following parameters:
(a) n = 15, k = 10, p = 0.015
(b) n = 15, k = 10, p = 0.1
(c) n = 15, k = 10, p = 0.45
(d) n = 20, k = 10, p = 0.015
(e) n = 20, k = 10, p = 0.1
(e) n = 20, k = 10, p = 0.45

1. **Create Random Code:** Pick 2^k random vectors from the binary vector space $\{0,1\}^n$ where n is the code length and the rate is $\frac{k}{n}$. This is your code $\mathcal{C}$.

2. **Decoding:** Define a counter E which calculates number of decoding errors. Initially $E = 0$. Repeat following steps (in bullets) for N iterations (N should be at least 500, best to use something of the order of 1000-2000. It depends upon the execution time on your machine. Higher N gives better approximation).

- Pick a random codeword $\underline{c}$ from the code $\mathcal{C}$. This is your transmitted codeword
- *Simulating BSC channel:* Obtain the received vector $\underline{y}$ by flipping each position of $\underline{c}$ with probability p.

- Run the Minimum Distance Decoding algorithm on $\underline{y}$ to obtain the estimate $\hat{\underline{c}}$, (This will be the most time consuming step).
- Maintain a indicator value $\mathbb{I}(\hat{\underline{c}} \neq \underline{c})$ for each iteration. This is 1 if decoder made an error, and 0 otherwise.
- Compute $E = E + \mathbb{I}(\hat{\underline{c}} \neq \underline{c})$
- **Probability of error:** Record approximate average probability of error as $P_E(n, k, p, \mathcal{C}) \triangleq \frac{E}{N}$.

3. Repeat the steps 1, 2 for five times for each value of k, n, p as defined in the problem. You will get one $P_E(n, k, p, \mathcal{C})$ value for each code $\mathcal{C}$.

4. **Show outputs:** For each (k, n, p) values, output the $P_E(n, k, p, \mathcal{C})$ values you get.

5. For each (n, k, p) value-tuple, define

$$P_E(n, k, p) \triangleq \min_{\mathcal{C}} P_E(n, k, p, \mathcal{C}).$$

This corresponds to the best code among the ones you obtained for specific (n, k, p). Plot the values of $P_E(n, k, p)$ for different (n, k, p) as given in the problem.

6. **Recording algorithm and observations on video:** For this problem you will record the flow of your algorithm, example executions, outputs, inferences you gained from the plots, how it corresponds to the theory you learnt in the class (compare with BSC capacity for each value of p and connect the performance of the codes with this). Also record if you have done anything more than what the question asked, like run the algorithm for other values of n, k, p; implementing the encoding process, etc.