

# CT111 Introduction to Communication Systems

## Lecture 12: Source Coding and Data Compression

Yash M. Vasavada

Associate Professor, DA-IICT, Gandhinagar

5th March 2018

# Overview of Today's Talk

- 1 Block Diagrams
- 2 Compression of Digital Data
- 3 Information Theoretical Aspects
- 4 Discrete Memoryless Source
- 5 Proof of Source Coding Theorem
- 6 Practical Data Compression Schemes
  - Huffman Coding
  - Lempel-Ziv Coding

# Overview of Today's Talk

- 1 Block Diagrams
- 2 Compression of Digital Data
- 3 Information Theoretical Aspects
- 4 Discrete Memoryless Source
- 5 Proof of Source Coding Theorem
- 6 Practical Data Compression Schemes
  - Huffman Coding
  - Lempel-Ziv Coding

# Overview of Today's Talk

- 1 Block Diagrams
- 2 Compression of Digital Data
- 3 Information Theoretical Aspects
- 4 Discrete Memoryless Source
- 5 Proof of Source Coding Theorem
- 6 Practical Data Compression Schemes
  - Huffman Coding
  - Lempel-Ziv Coding

# Overview of Today's Talk

- 1 Block Diagrams
- 2 Compression of Digital Data
- 3 Information Theoretical Aspects
- 4 Discrete Memoryless Source
- 5 Proof of Source Coding Theorem
- 6 Practical Data Compression Schemes
  - Huffman Coding
  - Lempel-Ziv Coding

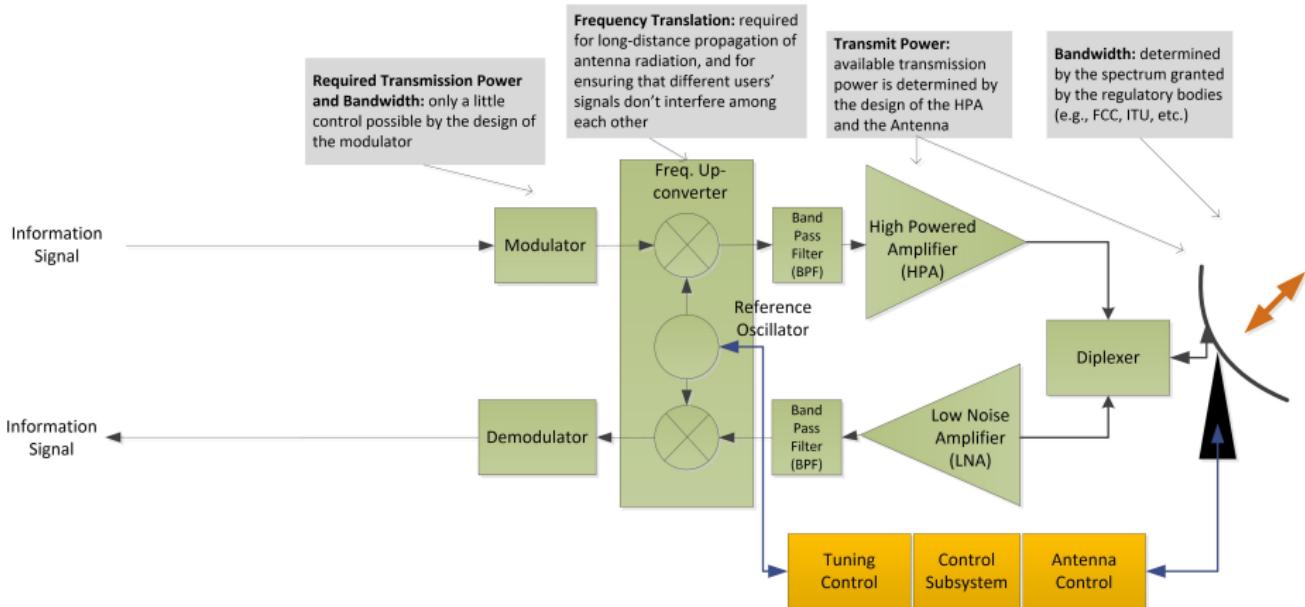
# Overview of Today's Talk

- 1 Block Diagrams
- 2 Compression of Digital Data
- 3 Information Theoretical Aspects
- 4 Discrete Memoryless Source
- 5 Proof of Source Coding Theorem
- 6 Practical Data Compression Schemes
  - Huffman Coding
  - Lempel-Ziv Coding

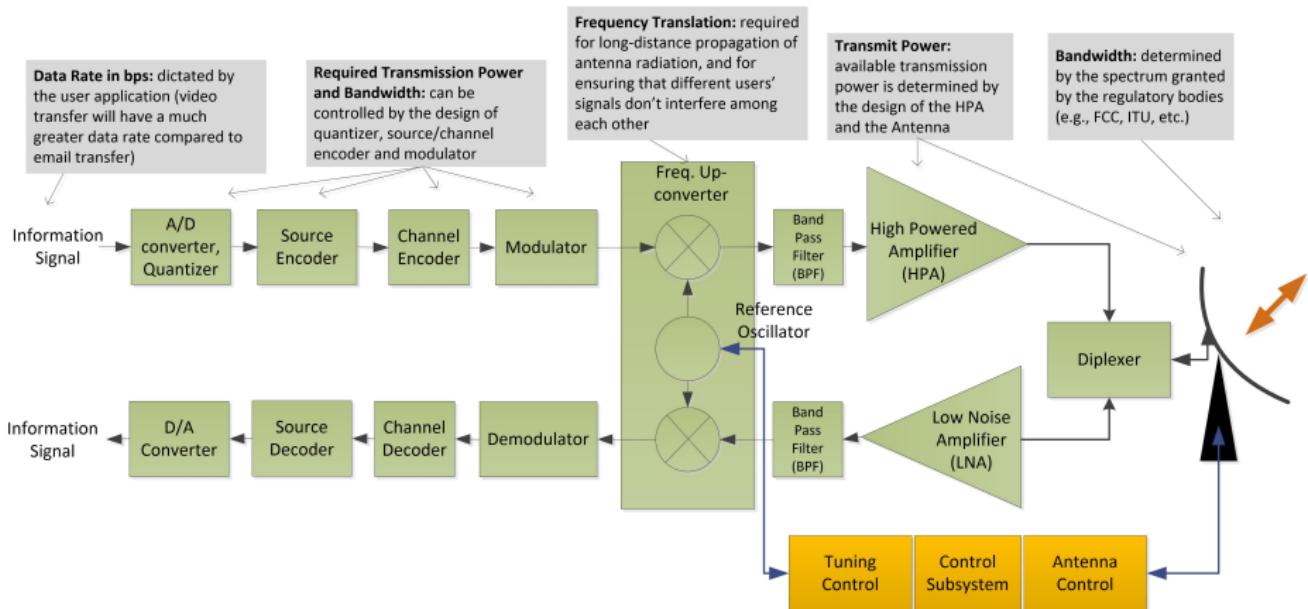
# Overview of Today's Talk

- 1 Block Diagrams
- 2 Compression of Digital Data
- 3 Information Theoretical Aspects
- 4 Discrete Memoryless Source
- 5 Proof of Source Coding Theorem
- 6 Practical Data Compression Schemes
  - Huffman Coding
  - Lempel-Ziv Coding

# Block Diagram of an Analog Communication Transceiver



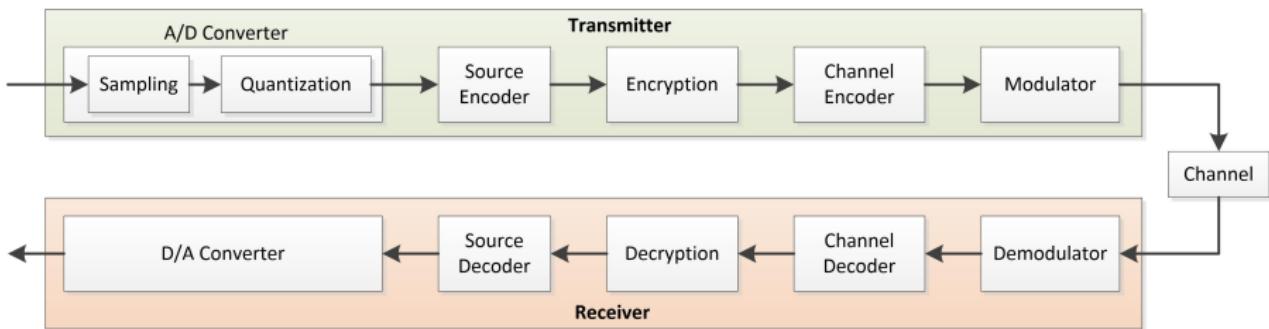
# Block Diagram of a *Digital* Communication Transceiver



# Digital Communication Transceiver

## Block Diagram

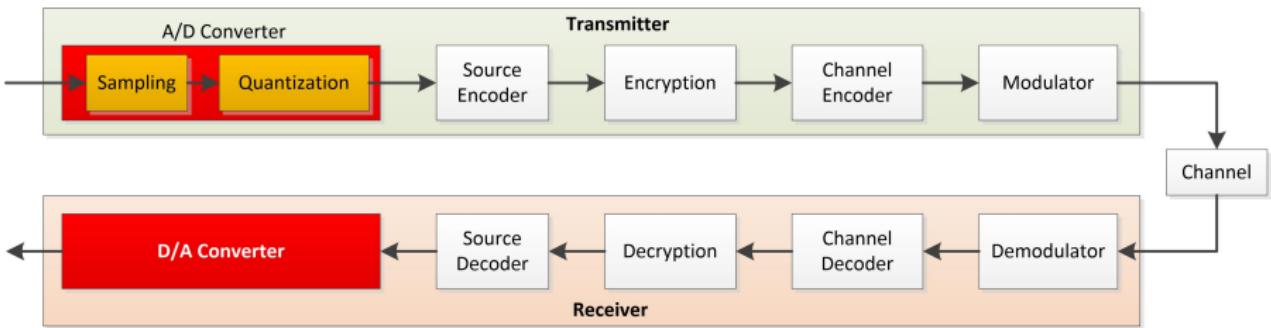
- A simplified model of a digital communication transceiver



# Digital Communication Transceiver

## Block Diagram

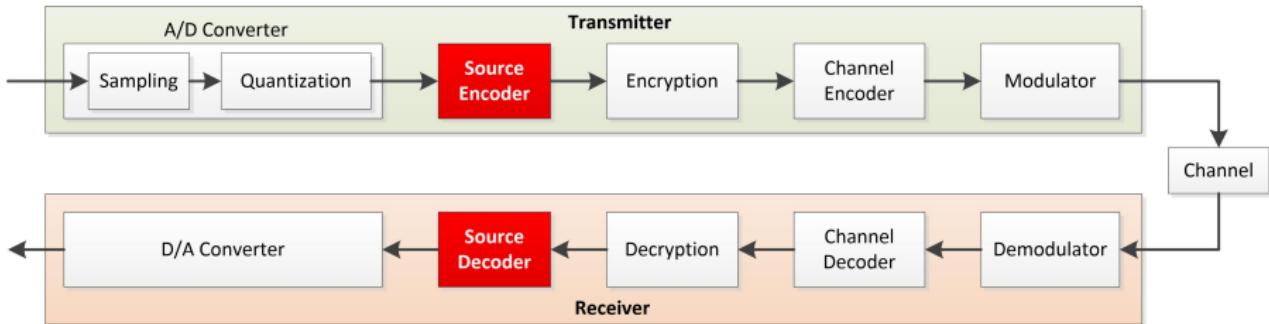
- We have studied the process of quantization of an analog information source



# Digital Communication Transceiver

## Block Diagram

- In this lecture, we will be studying in detail the concepts and practical algorithms (Huffman Coding and Lempel-Ziv Coding) for source encoding



# Source Coding

Solves the Problem of Data Compression

- Many digital data streams contain a lot of redundant information. For example, a digital image file may contain more zeros than ones:  
0,0,0,0,1,0,0,1,0,0,0,1,1
- We wish to squeeze out the redundant information to minimize the amount of data needed to be stored or transmitted
- Definition of Data Compression Problem:
  - ① What are the good algorithms that achieve the maximal data compression?
  - ② What is the maximum data compression that can be achieved if we want to recover the exact bit sequence after decompression?
- Importance of Data Compression Problem:
  - ▷ Cannot be overstated given so much data is getting uploaded/downloaded and stored in today's world of YouTube, Facebook, etc.

# Source Coding

## Data Compression Problem: Definitions

**Source Alphabet:** source is modeled as generating symbols each of which can take  $L$  symbols from a set  $\{x_1, x_2, \dots, x_L\}$ . This set is the source alphabet.

- ▷ Output of the quantizer is binary ( $L = 2$ )
- ▷ However, in general (e.g., for emails, webpages, etc. written in English),  $L$  has to be at least equal to the number of letters ( $= 26$ ) of English alphabet
- ▷ A common method for text encoding: ASCII character set ( $L = 128$ )

**Symbol Probabilities:** some of the source symbols are more likely than the others. Probability  $P(X = x_k)$  of  $k^{\text{th}}$  symbol is denoted as

$p_k$ . Since the source alphabet is the universal set,  $\sum_{k=1}^K p_k = 1$ .

**Discrete Memoryless Source (DMS):** in DMS, each source symbol occurs independently of the other symbols. We will mostly assume that the source is DMS

- With more complicated notations, the theory for DMS can be extended to correlated sources

# Source Coding

## for DMS

- A source code is a rule which maps symbols from the source alphabet to sequence of bits
- Fixed length source code maps source symbols into codewords of the same length:
  - Example:  $x_1 \Rightarrow 00, x_2 \Rightarrow 01, x_3 \Rightarrow 10, x_4 \Rightarrow 11$
- Variable length source code maps source symbols into codewords of different lengths:
  - Example:  $x_1 \Rightarrow 1, x_2 \Rightarrow 01, x_3 \Rightarrow 101, x_4 \Rightarrow 11$
- Variable length source code is uniquely decodable if no codeword is a prefix of another codeword:
  - Uniquely decodable:  $x_1 \Rightarrow 01, x_2 \Rightarrow 000, x_3 \Rightarrow 10, x_4 \Rightarrow 111$
  - Counter-example:  $x_1 \Rightarrow 0, x_2 \Rightarrow 01, x_3 \Rightarrow 10, x_4 \Rightarrow 101$

# Source Coding

for DMS

- Rate  $R$  of a source code is the average number of bits required to represent a source symbol
- Let  $n_k$  be the number of bits in the codeword which represents a source symbol  $x_k$ .

$$\rightarrow \text{Therefore, } R = \sum_{k=1}^L p_k n_k$$

$$\rightarrow \text{For fixed length codes, } R \geq \lceil \log_2 L \rceil$$

- **Goal of Source Encoding:** find a *uniquely decodable* code that minimizes  $R$
- **Strategy:** assign short codewords (small  $n_k$ ) to the most likely symbols (large  $p_k$ )

# Source Coding

for DMS

- Rate  $R$  of a source code is the average number of bits required to represent a source symbol
- Let  $n_k$  be the number of bits in the codeword which represents a source symbol  $x_k$ .

$$\rightarrow \text{Therefore, } R = \sum_{k=1}^L p_k n_k$$

$\rightarrow$  For fixed length codes,  $R \geq \lceil \log_2 L \rceil$

- **Goal of Source Encoding:** find a *uniquely decodable* code that minimizes  $R$
- **Strategy:** assign short codewords (small  $n_k$ ) to the most likely symbols (large  $p_k$ )

# Source Coding

## for DMS

- Rate  $R$  of a source code is the average number of bits required to represent a source symbol
- Let  $n_k$  be the number of bits in the codeword which represents a source symbol  $x_k$ .

$$\rightarrow \text{Therefore, } R = \sum_{k=1}^L p_k n_k$$

$$\rightarrow \text{For fixed length codes, } R \geq \lceil \log_2 L \rceil$$

- **Goal of Source Encoding:** find a *uniquely decodable* code that minimizes  $R$
- **Strategy:** assign short codewords (small  $n_k$ ) to the most likely symbols (large  $p_k$ )

# Source Coding

for DMS

- Rate  $R$  of a source code is the average number of bits required to represent a source symbol
- Let  $n_k$  be the number of bits in the codeword which represents a source symbol  $x_k$ .

$$\rightarrow \text{Therefore, } R = \sum_{k=1}^L p_k n_k$$

$$\rightarrow \text{For fixed length codes, } R \geq \lceil \log_2 L \rceil$$

- **Goal of Source Encoding:** find a *uniquely decodable* code that minimizes  $R$
- **Strategy:** assign short codewords (small  $n_k$ ) to the most likely symbols (large  $p_k$ )

# A Measure of Information Content of the Source

## Entropy

- What is the information generated when the source emits  $k^{th}$  symbol  $x_k$  with probability  $p_k$ ?
  - Information should be inversely proportional to  $p_k$
  - Information, when measured in bits, should be a logarithmic function
- Information generated is therefore  $\log_2 \left( \frac{1}{p_k} \right) = -\log_2 p_k$  bits. This is called self-information of the source.
- What is  $R$  if each symbol gets encoded using the number of bits that equals its self-information?
  - ▷ 
$$R = - \sum_{k=1}^L p_k \log_2 p_k \text{ bits}$$

# A Measure of Information Content of the Source

## Entropy

- What is the information generated when the source emits  $k^{th}$  symbol  $x_k$  with probability  $p_k$ ?
  - Information should be inversely proportional to  $p_k$
  - Information, when measured in bits, should be a logarithmic function
- Information generated is therefore  $\log_2 \left( \frac{1}{p_k} \right) = -\log_2 p_k$  bits. This is called self-information of the source.
- What is  $R$  if each symbol gets encoded using the number of bits that equals its self-information?

$$\triangleright R = - \sum_{k=1}^L p_k \log_2 p_k \text{ bits}$$

# A Measure of Information Content of the Source

## Entropy

- Entropy  $H(X)$  is a measure of uncertainty in the source output, which is modeled as a random variable  $X$
- $H(X)$  is defined as the average information generated by the source, where average is taken over the entire source alphabet
  - Information generated by  $k^{th}$  symbol with probability of  $p_k$ :  
 $n_k = -\log_2 p_k$  bits
  - Average of the information generated by the source:

$$H(X) = - \sum_{k=1}^L p_k \log_2 p_k, \quad \text{in bits}$$

- $H(X)$  is maximized if  $p_k = \frac{1}{L}$
- $H(x)$  is minimized if  $p_k = 1$  for some  $k$

# A Measure of Information Content of the Source

## Entropy

- Entropy  $H(X)$  is a measure of uncertainty in the source output, which is modeled as a random variable  $X$
- $H(X)$  is defined as the average information generated by the source, where average is taken over the entire source alphabet
  - Information generated by  $k^{th}$  symbol with probability of  $p_k$ :  
 $n_k = -\log_2 p_k$  bits
  - Average of the information generated by the source:

$$H(X) = - \sum_{k=1}^L p_k \log_2 p_k, \quad \text{in bits}$$

- $H(X)$  is maximized if  $p_k = \frac{1}{L}$
- $H(x)$  is minimized if  $p_k = 1$  for some  $k$

# A Measure of Information Content of the Source

## Entropy

- Entropy  $H(X)$  is a measure of uncertainty in the source output, which is modeled as a random variable  $X$
- $H(X)$  is defined as the average information generated by the source, where average is taken over the entire source alphabet
  - Information generated by  $k^{th}$  symbol with probability of  $p_k$ :  
 $n_k = -\log_2 p_k$  bits
  - Average of the information generated by the source:

$$H(X) = - \sum_{k=1}^L p_k \log_2 p_k, \quad \text{in bits}$$

- $H(X)$  is maximized if  $p_k = \frac{1}{L}$
- $H(x)$  is minimized if  $p_k = 1$  for some  $k$

# A Measure of Information Content of the Source

## Entropy

- Entropy  $H(X)$  is a measure of uncertainty in the source output, which is modeled as a random variable  $X$
- $H(X)$  is defined as the average information generated by the source, where average is taken over the entire source alphabet
  - Information generated by  $k^{th}$  symbol with probability of  $p_k$ :  
 $n_k = -\log_2 p_k$  bits
  - Average of the information generated by the source:

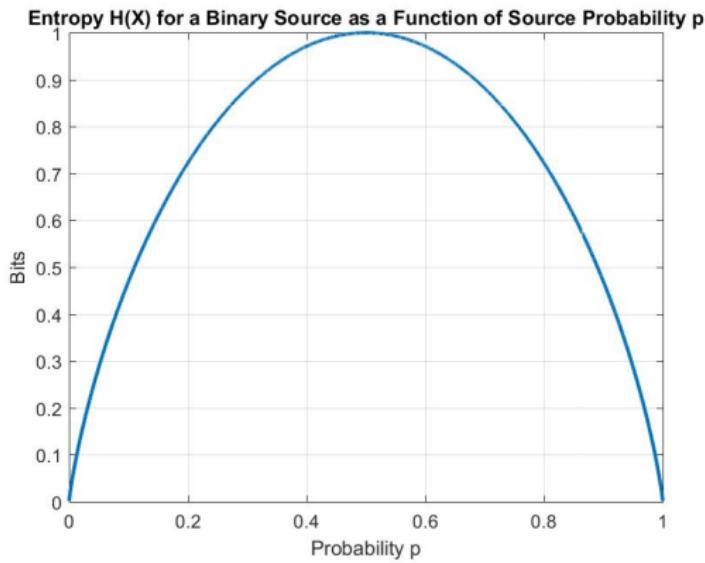
$$H(X) = - \sum_{k=1}^L p_k \log_2 p_k, \quad \text{in bits}$$

- $H(X)$  is maximized if  $p_k = \frac{1}{L}$
- $H(x)$  is minimized if  $p_k = 1$  for some  $k$

# A Measure of Information Content of the Source

## Entropy for Binary RV

- For binary RV  $X$  for which  $p_X(x) = \begin{cases} 1-p, & x=0, \\ p, & x=1 \end{cases}$ , entropy  $H(X)$  is given as  $H(X) = -p \log_2 p - (1-p) \log_2 (1-p)$



# Source Coding Theorem

- *A fundamental theorem of the Information Theory [Shannon 1948 paper]:*
  - For a uniquely decodeable source code that provides lossless compression (i.e., perfect reconstruction of the source symbols at the source decoder),  $R \geq H(X)$ .
    - ▷ Rate can be made arbitrarily close to, but not less than,  $H(X)$
  - This is the Source Coding (or Data Compression) Theorem
- How to prove this theorem?

# Source Coding Theorem

- A fundamental theorem of the Information Theory [Shannon 1948 paper]:
  - For a uniquely decodeable source code that provides lossless compression (i.e., perfect reconstruction of the source symbols at the source decoder),  $R \geq H(X)$ .
    - ▷ Rate can be made arbitrarily close to, but not less than,  $H(X)$
  - This is the Source Coding (or Data Compression) Theorem
- How to prove this theorem?

# Source Coding Theorem

- A fundamental theorem of the Information Theory [Shannon 1948 paper]:
  - For a uniquely decodeable source code that provides lossless compression (i.e., perfect reconstruction of the source symbols at the source decoder),  $R \geq H(X)$ .
    - ▷ Rate can be made arbitrarily close to, but not less than,  $H(X)$
  - This is the Source Coding (or Data Compression) Theorem
- How to prove this theorem?

# A Question

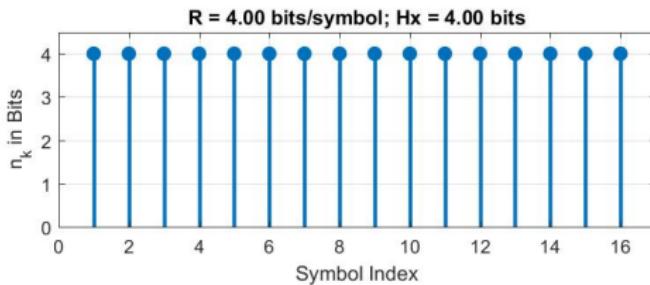
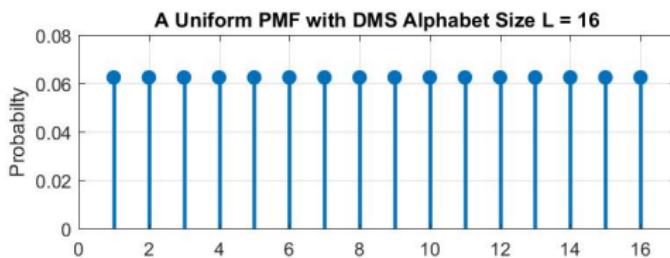
- Why is the length  $n_k$  of codeword in bits a *logarithmic* function of the probability  $p_k$  of  $k^{\text{th}}$  symbol?
- Answer given by examples on next two slides

# A Question

- Why is the length  $n_k$  of codeword in bits a *logarithmic* function of the probability  $p_k$  of  $k^{\text{th}}$  symbol?
- Answer given by examples on next two slides

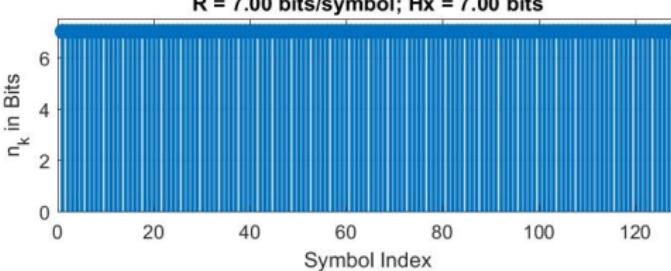
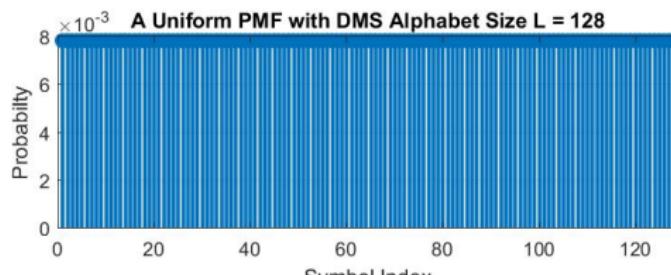
# A Uniform PMF

- When the DMS alphabet size is  $L = 2^n$ , and its PMF is uniform, the probability of each letter is  $p = 1/L = 2^{-n}$ , and each letter is encoded using a fixed-length code of length  $-\log_2(p) = n$  bits. Shown here when  $n = 4$  bits.



# A Uniform PMF

- Example of previous slide, shown when  $n = 7$  bits. Probability  $p$  reduces, and hence the information content –  $\log_2 p$  increases to 7 bits



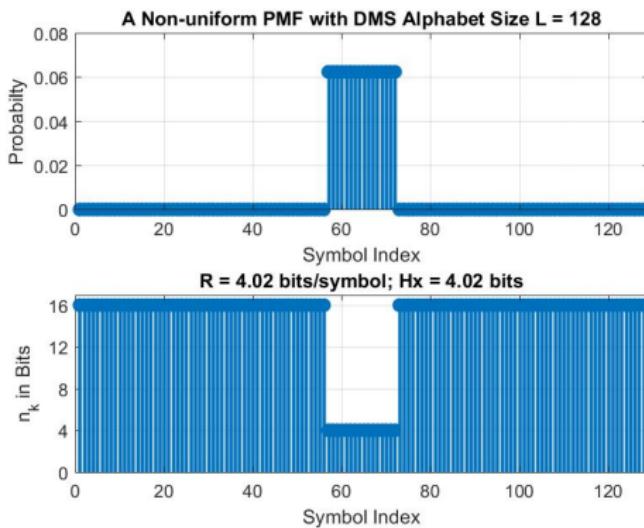
## A Uniform PMF

- Thus, when the DMS has uniform PMF, and has a total of  $L = 2^n$  symbols, we require  $n$  bits to represent each symbol
- *Cannot* use less than  $n$  bits and still represent all  $L$  symbols

# A Non-uniform PMF

Data Compression Possible

- Suppose  $L = 128$ , however, only  $M = 16$  symbols actually occur realistically. Remaining 112 symbols have vanishing probability of occurrence. In this case of non-uniform PMF, one needs  $n = 4$  bits instead of 7 bits (using than 4 bits will not be enough)



# A Claim

- An arbitrary DMS with non-uniform probabilities can be converted to a case in which most of its output sequences have vanishing probability
- A hint:
  - ▷ Unless the PMF is uniform distribution (all  $L$  symbols have equal probability of  $1/L$ ), *encoding a sequence of symbols* is better compared to encoding one symbol at a time

# A Claim

- An arbitrary DMS with non-uniform probabilities can be converted to a case in which most of its output sequences have vanishing probability
- A hint:
  - ▷ Unless the PMF is uniform distribution (all  $L$  symbols have equal probability of  $1/L$ ), *encoding a sequence of symbols* is better compared to encoding one symbol at a time

## A Few More Questions

- When  $L = 2$ , the DMS already generates bits. What is the meaning of  $R$  when its interpretation will change from bits/symbol to bits/bit?
- Let us take that, with  $L = 2$ , the probability of 0 is  $p = 0.1$  and probability of 1 is  $1 - p = 0.9$ . The self-information of individual symbols (bits in this case) is  $-\log_2 p_k = [3.3219, \quad 0.1520]$  bits. How to represent bit 0 using 3.3219 bits and bit 1 using 0.152 bits (since bits cannot be divided)?

## A Few More Questions

- When  $L = 2$ , the DMS already generates bits. What is the meaning of  $R$  when its interpretation will change from bits/symbol to bits/bit?
- Let us take that, with  $L = 2$ , the probability of 0 is  $p = 0.1$  and probability of 1 is  $1 - p = 0.9$ . The self-information of individual symbols (bits in this case) is  $-\log_2 p_k = [3.3219, \quad 0.1520]$  bits. How to represent bit 0 using 3.3219 bits and bit 1 using 0.152 bits (since bits cannot be divided)?

## A Few More Questions

- When  $L = 2$ , the DMS already generates bits. What is the meaning of  $R$  when its interpretation will change from bits/symbol to bits/bit?
- Let us take that, with  $L = 2$ , the probability of 0 is  $p = 0.1$  and probability of 1 is  $1 - p = 0.9$ . The self-information of individual symbols (bits in this case) is  $-\log_2 p_k = [3.3219, \quad 0.1520]$  bits. How to represent bit 0 using 3.3219 bits and bit 1 using 0.152 bits (since bits cannot be divided)?

# A Claim

- An arbitrary DMS with non-uniform probabilities can be converted to a case in which most of its output sequences have vanishing probability
- We will show this using an example binary DMS ( $L = 2$ ) with probability of 1 equal to  $p$  and probability of 0 equal to  $1 - p$
- We will take a sequence of  $M$  consecutive bits that are generated by this DMS
  - In this example,  $p$  is set to 0.1, whereas  $M$  is varied
- The example shown next generalizes to any arbitrary DMS with  $L > 2$

# A Sequence of $M$ DMS symbols

$M = 2, p = 0.1$

- We first take a sequence of  $M = 2$  consecutive bits coming out of this DMS
  - We will see  $2^M = 4$  possible sequences
  - These can be divided into  $M + 1 = 3$  different, non-overlapping, subsets of binary sequences
    - ① Subset 1 is all-zero sequence (no ones)
      - Has one sequence [0, 0], which occurs with a probability of  $(1 - p)^2 = 0.9^2 = 0.81$
    - ② Subset 2 is a set of all sequences with exactly one 1
      - Has  $\binom{M=2}{1} = 2$  sequences ([0, 1] and [1, 0]), each of which occurs with probability of  $0.9 \times 0.1 = 0.09$ . Therefore, total probability of this subset is  $2 \times 0.09 = 0.18$ .
    - ③ Finally, subset 3 is a set of all-ones sequences
      - Has one sequence [1, 1], which occurs with a probability of  $(1 - p)^2 = 0.1^2 = 0.01$

# A Sequence of $M$ DMS symbols

$M = 2, p = 0.1$

- We first take a sequence of  $M = 2$  consecutive bits coming out of this DMS
- We will see  $2^M = 4$  possible sequences
  - These can be divided into  $M + 1 = 3$  different, non-overlapping, subsets of binary sequences
    - ① Subset 1 is all-zero sequence (no ones)
      - Has one sequence [0, 0], which occurs with a probability of  $(1 - p)^2 = 0.9^2 = 0.81$
    - ② Subset 2 is a set of all sequences with exactly one 1
      - Has  $\binom{M=2}{1} = 2$  sequences ([0, 1] and [1, 0]), each of which occurs with probability of  $0.9 \times 0.1 = 0.09$ . Therefore, total probability of this subset is  $2 \times 0.09 = 0.18$ .
    - ③ Finally, subset 3 is a set of all-ones sequences
      - Has one sequence [1, 1], which occurs with a probability of  $(1 - p)^2 = 0.1^2 = 0.01$

# A Sequence of $M$ DMS symbols

$M = 2, p = 0.1$

- We first take a sequence of  $M = 2$  consecutive bits coming out of this DMS
- We will see  $2^M = 4$  possible sequences
- These can be divided into  $M + 1 = 3$  different, non-overlapping, subsets of binary sequences
  - Subset 1 is all-zero sequence (no ones)
    - Has one sequence [0, 0], which occurs with a probability of  $(1 - p)^2 = 0.9^2 = 0.81$
  - Subset 2 is a set of all sequences with exactly one 1
    - Has  $\binom{M=2}{1} = 2$  sequences ([0, 1] and [1, 0]), each of which occurs with probability of  $0.9 \times 0.1 = 0.09$ . Therefore, total probability of this subset is  $2 \times 0.09 = 0.18$ .
  - Finally, subset 3 is a set of all-ones sequences
    - Has one sequence [1, 1], which occurs with a probability of  $(1 - p)^2 = 0.1^2 = 0.01$

# A Sequence of $M$ DMS symbols

$M = 2, p = 0.1$

- We first take a sequence of  $M = 2$  consecutive bits coming out of this DMS
- We will see  $2^M = 4$  possible sequences
- These can be divided into  $M + 1 = 3$  different, non-overlapping, subsets of binary sequences
  - ① Subset 1 is all-zero sequence (no ones)
    - Has one sequence [0, 0], which occurs with a probability of  $(1 - p)^2 = 0.9^2 = 0.81$
  - ② Subset 2 is a set of all sequences with exactly one 1
    - Has  $\binom{M=2}{1} = 2$  sequences ([0, 1] and [1, 0]), each of which occurs with probability of  $0.9 \times 0.1 = 0.09$ . Therefore, total probability of this subset is  $2 \times 0.09 = 0.18$ .
  - ③ Finally, subset 3 is a set of all-ones sequences
    - Has one sequence [1, 1], which occurs with a probability of  $(1 - p)^2 = 0.1^2 = 0.01$

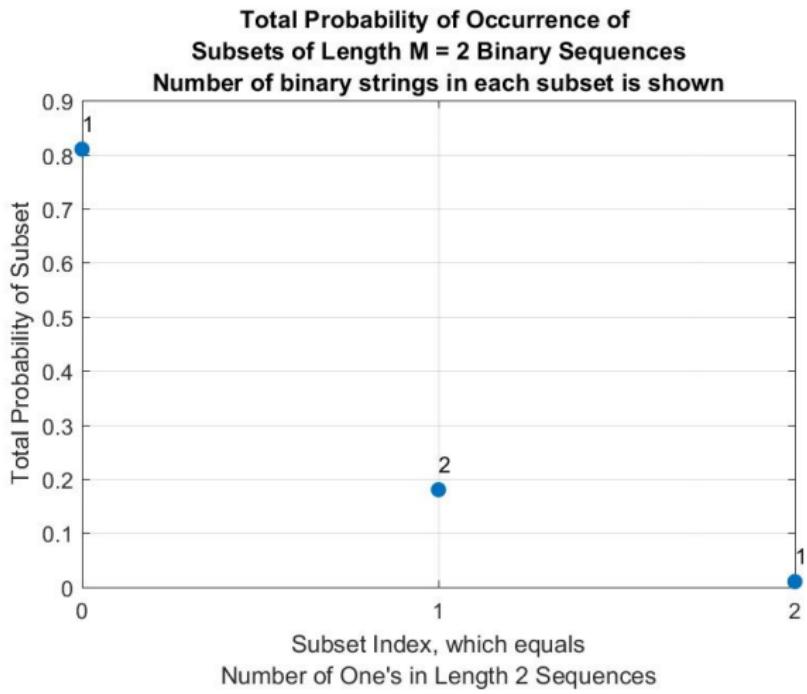
# A Sequence of $M$ DMS symbols

$M = 2, p = 0.1$

- We first take a sequence of  $M = 2$  consecutive bits coming out of this DMS
- We will see  $2^M = 4$  possible sequences
- These can be divided into  $M + 1 = 3$  different, non-overlapping, subsets of binary sequences
  - ① Subset 1 is all-zero sequence (no ones)
    - Has one sequence [0, 0], which occurs with a probability of  $(1 - p)^2 = 0.9^2 = 0.81$
  - ② Subset 2 is a set of all sequences with exactly one 1
    - Has  $\binom{M=2}{1} = 2$  sequences ([0, 1] and [1, 0]), each of which occurs with probability of  $0.9 \times 0.1 = 0.09$ . Therefore, total probability of this subset is  $2 \times 0.09 = 0.18$ .
  - ③ Finally, subset 3 is a set of all-ones sequences
    - Has one sequence [1, 1], which occurs with a probability of  $(1 - p)^2 = 0.1^2 = 0.01$

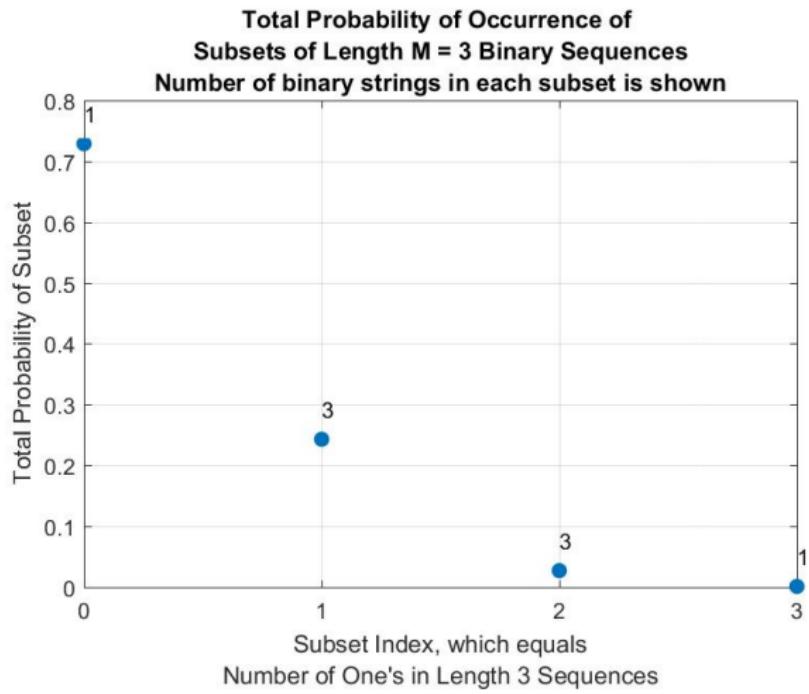
# A Sequence of $M$ DMS symbols

$M = 2, p = 0.1$



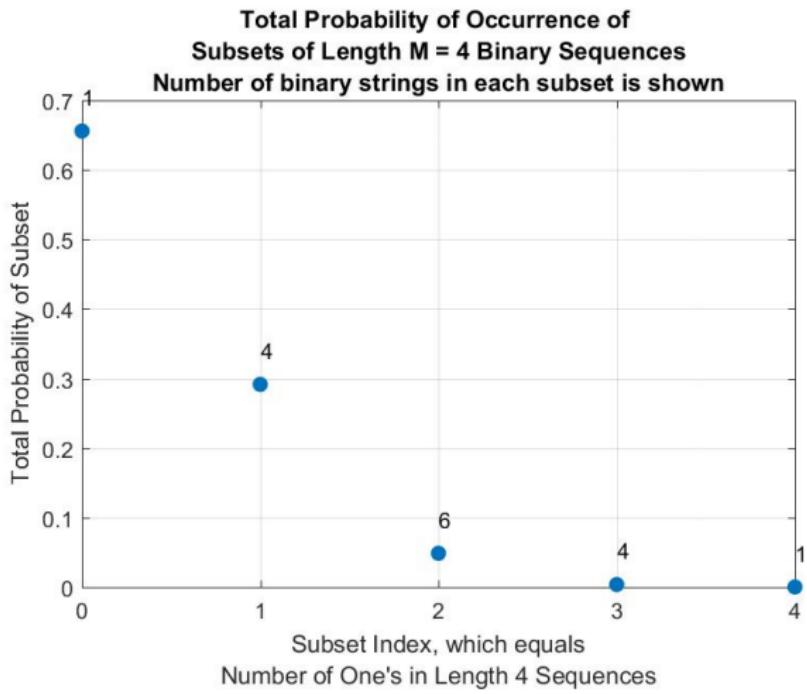
# A Sequence of $M$ DMS symbols

$M = 3, p = 0.1$



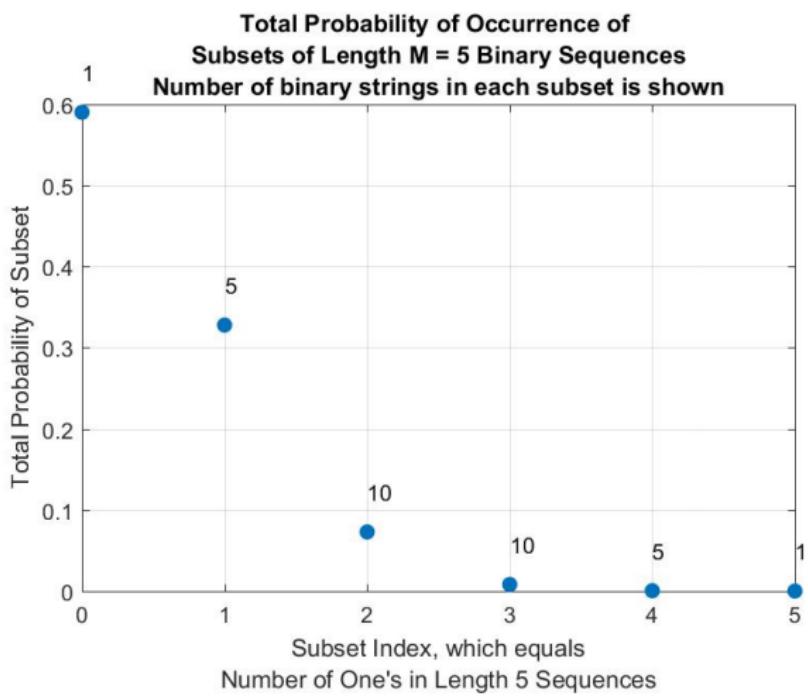
# A Sequence of $M$ DMS symbols

$M = 4, p = 0.1$



# A Sequence of $M$ DMS symbols

$M = 5, p = 0.1$



## A Few More Questions

- What do we expect to see as we keep on increasing  $M$ ?
- Again a claim: only one subset will survive!
- Which one?

## A Few More Questions

- What do we expect to see as we keep on increasing  $M$ ?
- Again a claim: only one subset will survive!
- Which one?

## A Few More Questions

- What do we expect to see as we keep on increasing  $M$ ?
- Again a claim: only one subset will survive!
- Which one?

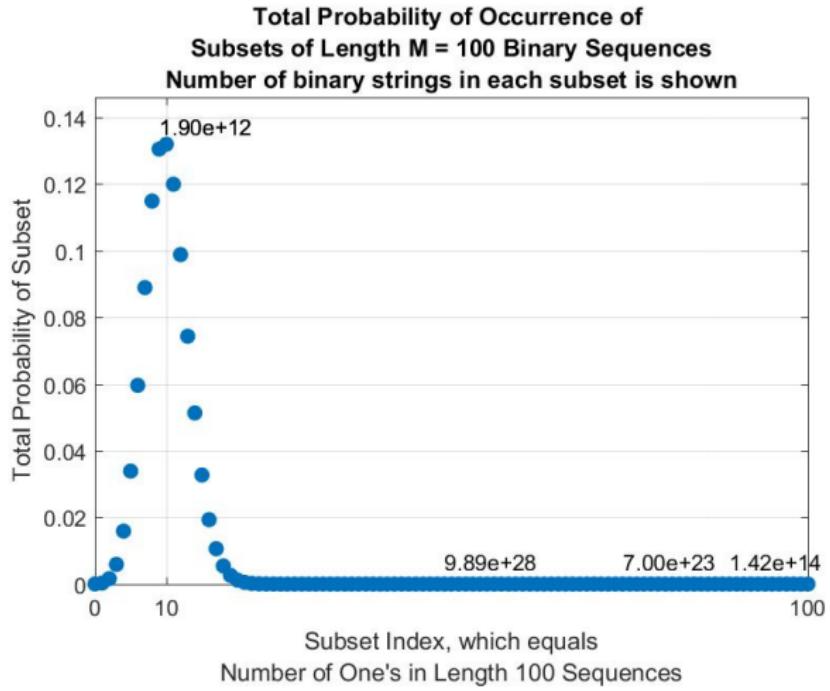
# A Note

- As we keep on increasing  $M$ , the set of binary sequences, which has total  $2^M$  members, becomes *huge*
  - $\rightarrow M = 58$ ,  $2^M$  is the age of universe in seconds
  - $\rightarrow M = 171$ ,  $2^M$  is the number of electrons in the Earth
  - $\rightarrow M = 190$ ,  $2^M$  is the number of electrons in the solar system
  - $\rightarrow M = 266$ ,  $2^M$  is the number of electrons in the universe

(REF: David MacKay book: “Information Theory, Inference, and Learning Algorithms,” available on the web for free download)

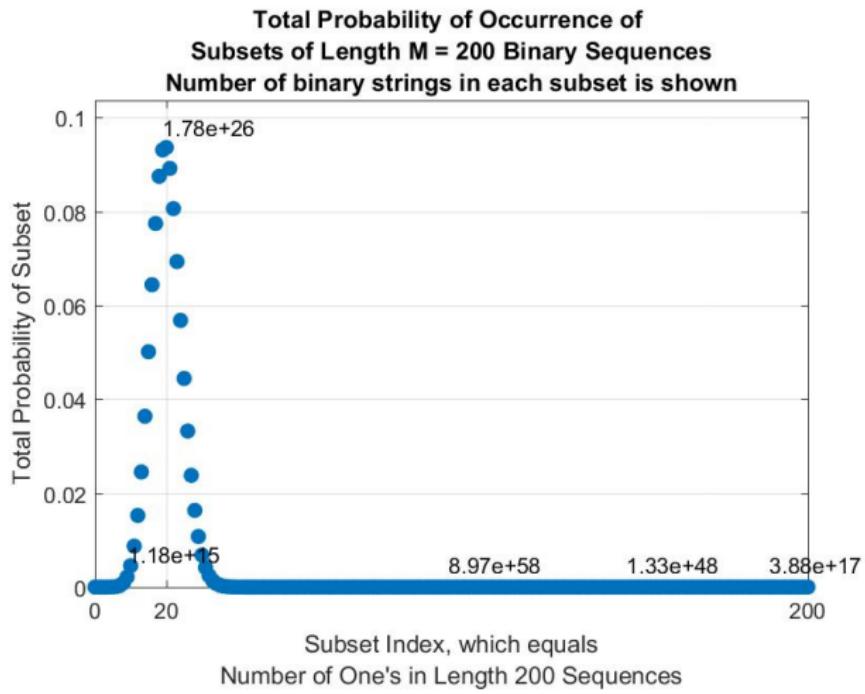
# A Sequence of $M$ DMS symbols

$M = 100, p = 0.1$



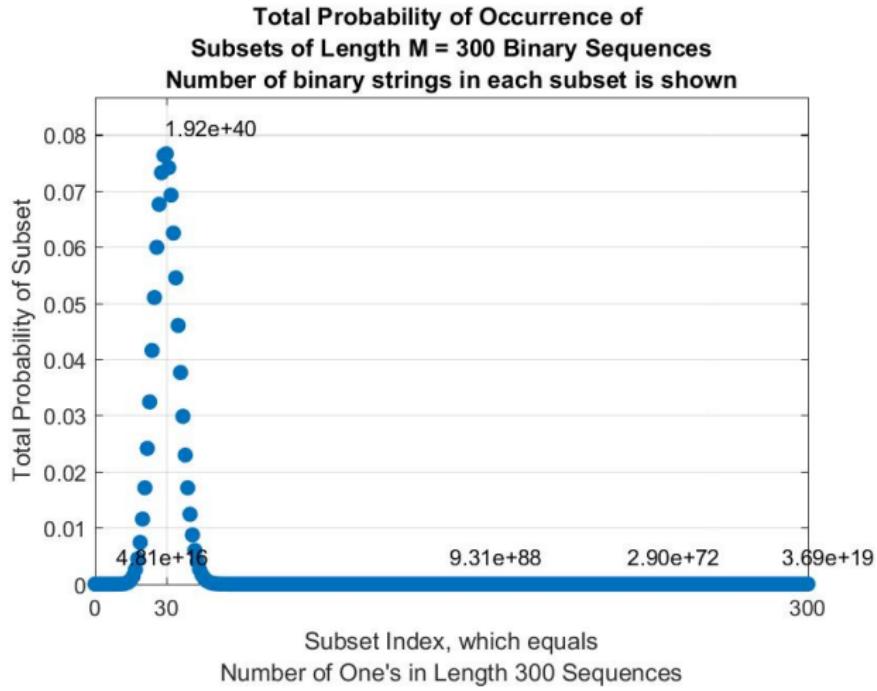
# A Sequence of $M$ DMS symbols

$M = 200, p = 0.1$



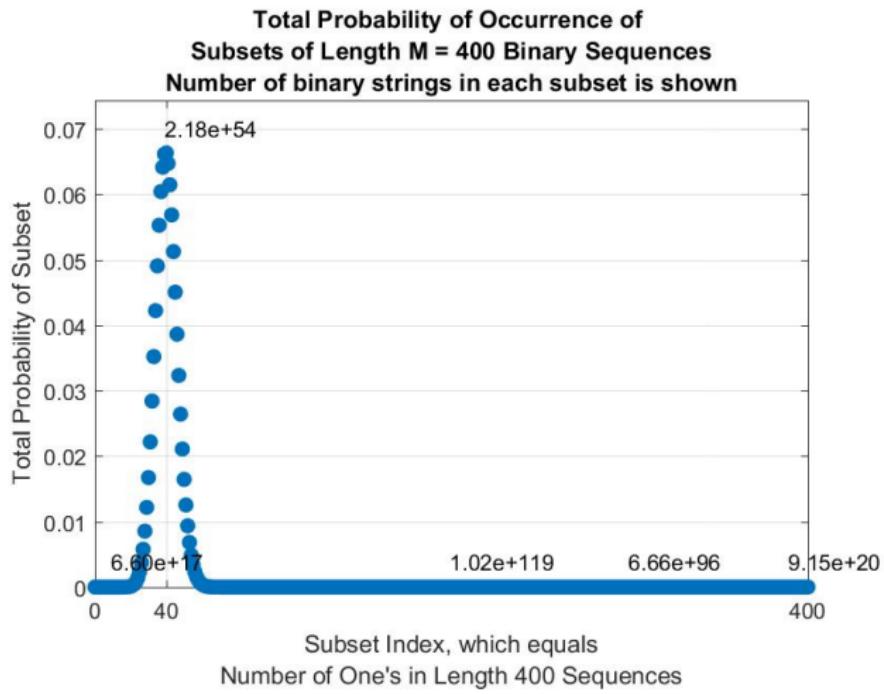
# A Sequence of $M$ DMS symbols

$M = 300, p = 0.1$



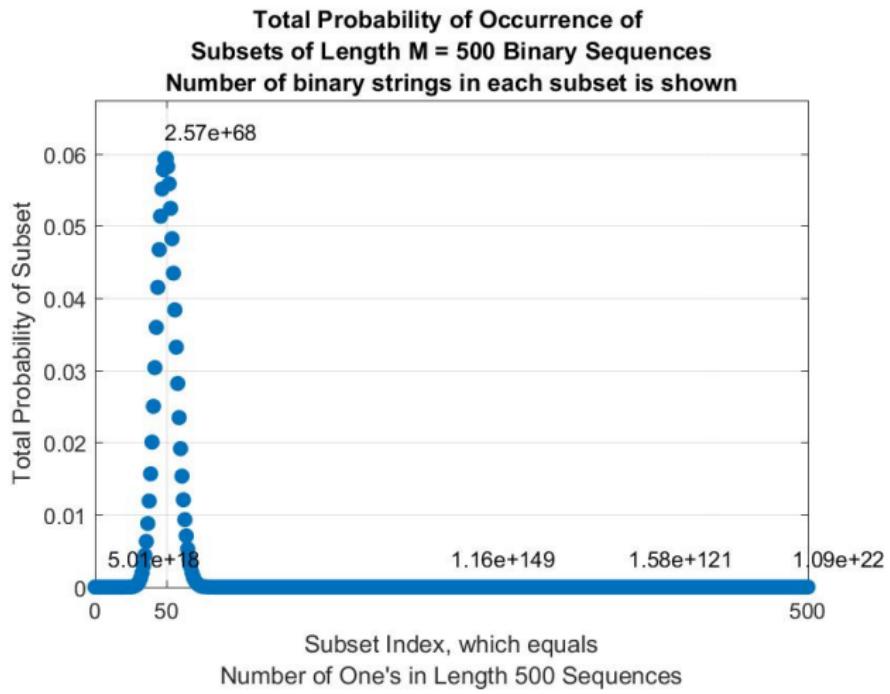
# A Sequence of $M$ DMS symbols

$M = 400, p = 0.1$



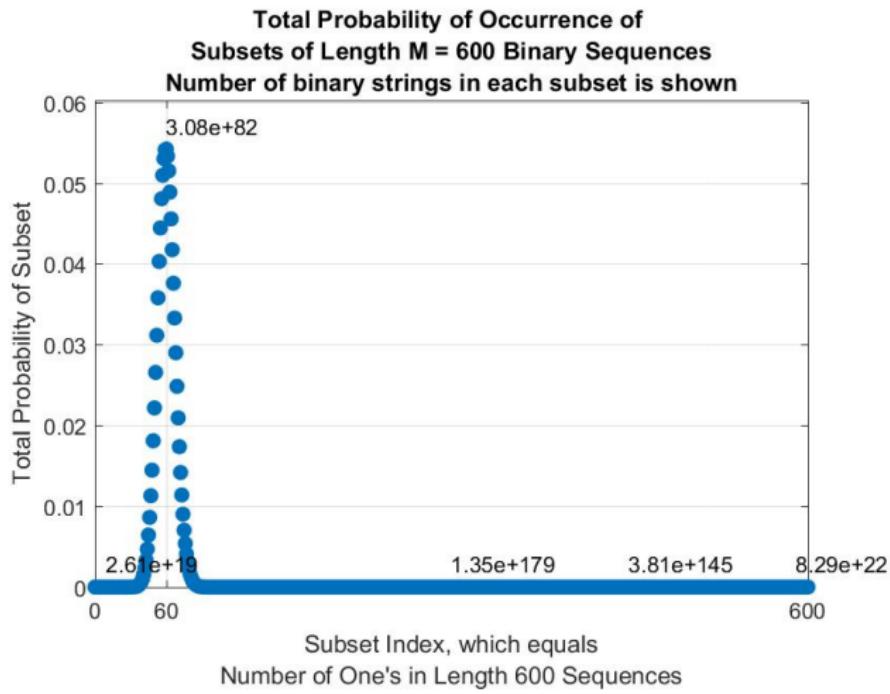
# A Sequence of $M$ DMS symbols

$M = 500, p = 0.1$



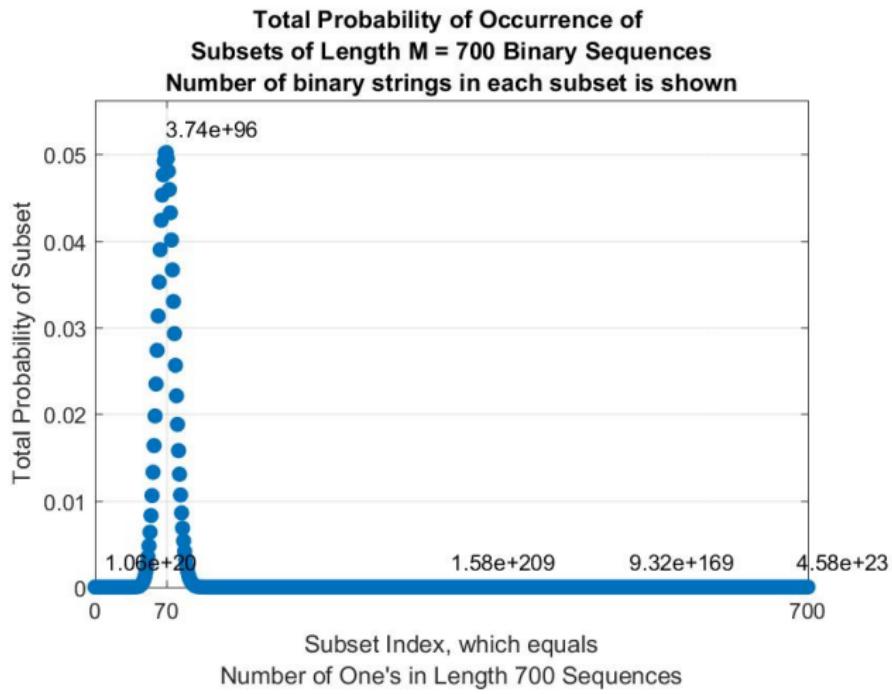
# A Sequence of $M$ DMS symbols

$M = 600, p = 0.1$



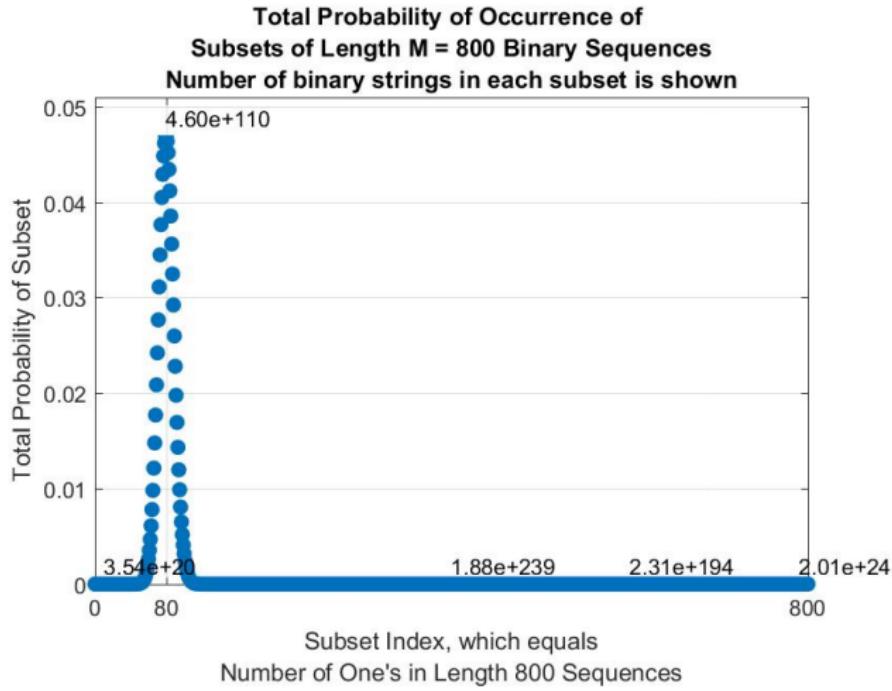
# A Sequence of $M$ DMS symbols

$M = 700, p = 0.1$



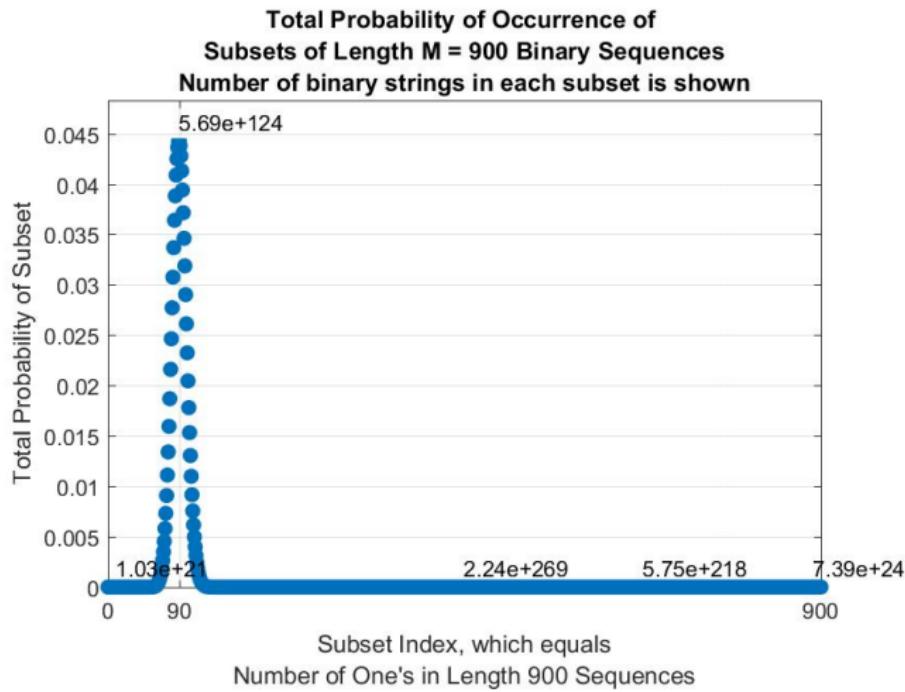
# A Sequence of $M$ DMS symbols

$M = 800, p = 0.1$



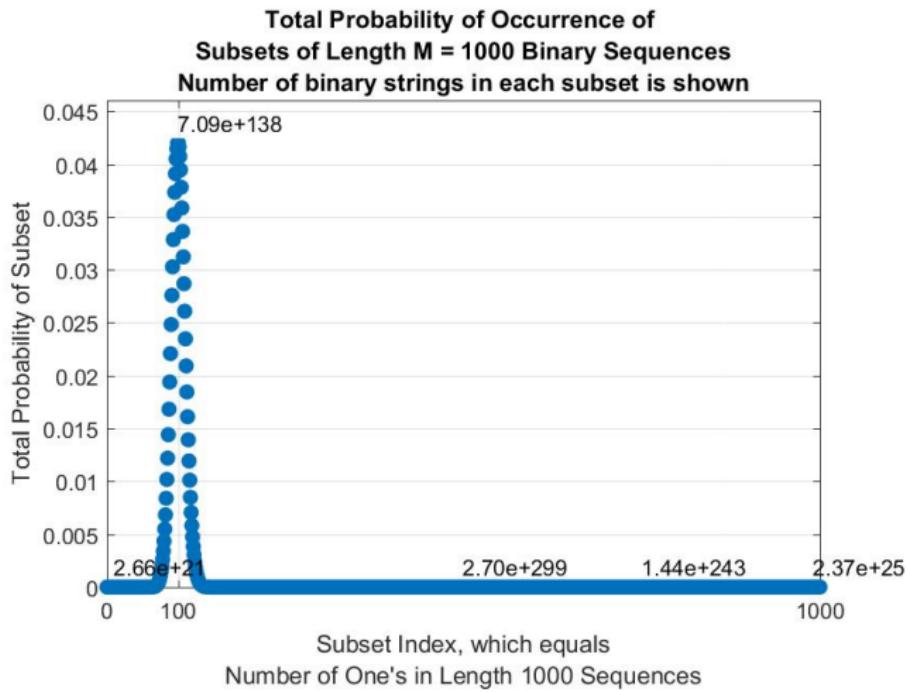
# A Sequence of $M$ DMS symbols

$M = 900, p = 0.1$



# A Sequence of $M$ DMS symbols

$M = 1000, p = 0.1$



## A Few More Questions

- What do we expect to see as we keep on increasing  $M$ ?
  - Again a claim: only one subset will survive!
  - Which one?
  - The subset formed by *all* binary sequences of length  $M$  which have  $M_1 = p \times M$  ones
  - Why?
  - Because that is *exactly* the definition of probability  $p$ :

$$p = \lim_{M \rightarrow \infty} \frac{M_1}{M}$$

## A Few More Questions

- What do we expect to see as we keep on increasing  $M$ ?
- Again a claim: only one subset will survive!
  - Which one?
  - The subset formed by *all* binary sequences of length  $M$  which have  $M_1 = p \times M$  ones
  - Why?
- Because that is *exactly* the definition of probability  $p$ :

$$p = \lim_{M \rightarrow \infty} \frac{M_1}{M}$$

## A Few More Questions

- What do we expect to see as we keep on increasing  $M$ ?
- Again a claim: only one subset will survive!
- Which one?
- The subset formed by *all* binary sequences of length  $M$  which have  $M_1 = p \times M$  ones
- Why?
- Because that is *exactly* the definition of probability  $p$ :

$$p = \lim_{M \rightarrow \infty} \frac{M_1}{M}$$

## A Few More Questions

- What do we expect to see as we keep on increasing  $M$ ?
- Again a claim: only one subset will survive!
- Which one?
- The subset formed by *all* binary sequences of length  $M$  which have  $M_1 = p \times M$  ones
- Why?
- Because that is *exactly* the definition of probability  $p$ :

$$p = \lim_{M \rightarrow \infty} \frac{M_1}{M}$$

## A Few More Questions

- What do we expect to see as we keep on increasing  $M$ ?
- Again a claim: only one subset will survive!
- Which one?
- The subset formed by *all* binary sequences of length  $M$  which have  $M_1 = p \times M$  ones
- Why?
- Because that is *exactly* the definition of probability  $p$ :

$$p = \lim_{M \rightarrow \infty} \frac{M_1}{M}$$

## A Few More Questions

- What do we expect to see as we keep on increasing  $M$ ?
- Again a claim: only one subset will survive!
- Which one?
- The subset formed by *all* binary sequences of length  $M$  which have  $M_1 = p \times M$  ones
- Why?
- Because that is *exactly* the definition of probability  $p$ :

$$p = \lim_{M \rightarrow \infty} \frac{M_1}{M}$$

## Coming to a Conclusion

- As  $M \rightarrow \infty$ , only one subset of  $2^M$  binary sequences survives
  - Total probability of this subset  $\rightarrow 1$
  - Total probability of all other (nonsurviving) subsets  $\rightarrow 0$
- No matter which binary sequence is picked from this subset,
  - it has  $p \times M$  ones and  $(1 - p) \times M$  zeros, and
  - therefore, the probability of each of these sequences is *identical* and equal to  $p^{pM}(1 - p)^{(1-p)M}$

# Conclusions

- As  $M \rightarrow \infty$ , only one subset of binary sequence survives
- Let the probability of occurrence of this subset be denoted as  $p_{typ}$ . As  $M \rightarrow \infty$ ,  $p_{typ} \rightarrow 1$ .
- Each of binary sequences picked from the surviving subset has
  - $p \times M$  ones and  $(1 - p) \times M$  zeros, and
  - probability of occurrence which is equal to  $p_i = p^{pM}(1 - p)^{(1-p)M}$
- Suppose the size of the surviving subset is  $K$  (i.e., it has  $K$  binary sequences)
  - $p_{typ} = K \times p_i = K \times p^{pM}(1 - p)^{(1-p)M} \rightarrow 1$
  - Therefore,  $K = p^{-pM}(1 - p)^{-(1-p)M}$

# Conclusions

- The size of surviving subset, which we will now call *typical set*, is  
 $K = p^{-pM}(1 - p)^{-(1-p)M}$
- Each of its member sequences has equal probability of  
 $p_i = p^{pM}(1 - p)^{(1-p)M}$
- Therefore, we can use *fixed-length* coding to represent these  $K$  sequences
- This fixed-length code will require *exactly*

$$\begin{aligned}\log_2 K &= \log_2 \left\{ p^{-pM}(1 - p)^{-(1-p)M} \right\} \\ &= M \times (-p \log_2 p - (1 - p) \log_2 (1 - p)) \\ &= M \times H(X)\end{aligned}$$

bits. Less than  $M \times H(X)$  bits will not be sufficient. More than  $M \times H(X)$  bits are too many.

- Since  $H(X) \approx 0.5$  when  $p = 0.1$ , we will need only  $\approx 0.5 \times M$  bits instead of  $M$  bits to represent this source output
- This proves the Source Coding (Data Compression) Theorem

# Practical Source Coding Algorithms

- There are two widely used source coding methods which make  $R$  approach the lower bound of  $H(X)$  promised by data compression theorem:
  - ① Huffman Coding: requires the knowledge of probabilities of DMS
  - ② Lempel-Ziv Coding: operates in a *blind* manner (i.e., without requiring the knowledge of the DMS' PMF)

# Source Coding Theorem

## Huffman Coding Algorithm

Huffman coding is based upon the idea of allocating short codewords to highly probable symbols. The algorithm for Huffman coding belongs to a class of algorithms based upon the Greedy Approach.

- ① List all symbols in descending order of probability
- ② Group the two lowest probability symbols into a new combined symbol
- ③ Repeat steps 1 and 2 until only a single combined symbol remains
- ④ Assign a 1 to each upper branch and 0 to each lower branch of the resulting binary tree
- ⑤ Codeword for each symbol is the sequence of bits from the root of the tree to that symbol

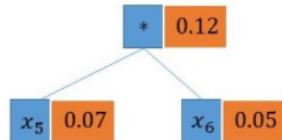
## Huffman Coding

## Source Coding Theorem

## Huffman Coding Algorithm: An Example

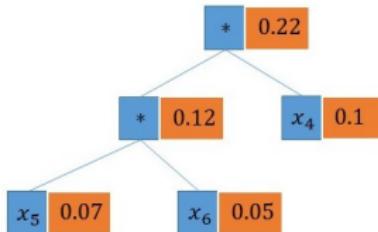
1

Symbol	Probability
$x_1$	0.45
$x_2$	0.2
$x_3$	0.13
$x_4$	0.1
$x_5$	0.07
$x_6$	0.05



2

Symbol	Probability
$x_1$	0.45
$x_2$	0.2
$x_3$	0.13
*	0.12
$x_4$	0.1



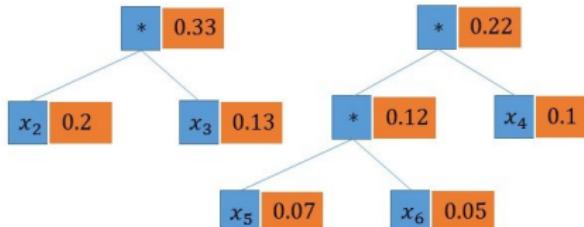
## Huffman Coding

## Source Coding Theorem

## Huffman Coding Algorithm: An Example

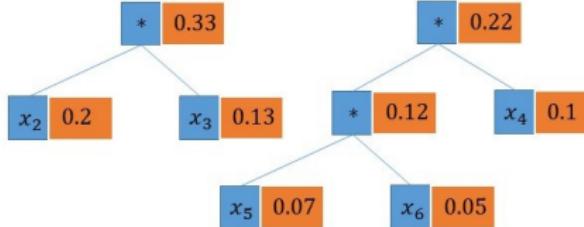
3

Symbol	Probability
$x_1$	0.45
*	0.22
$x_2$	0.2
$x_3$	0.13



4

Symbol	Probability
$x_1$	0.45
*	0.33
*	0.22



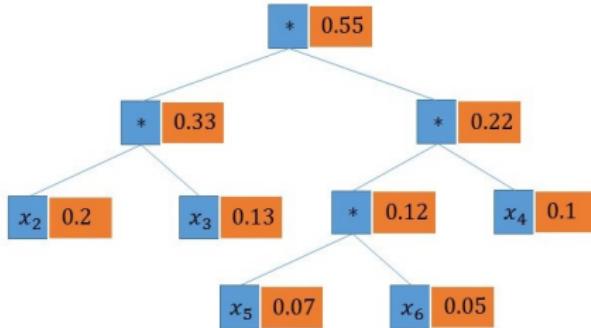
## Huffman Coding

## Source Coding Theorem

## Huffman Coding Algorithm: An Example

5

Symbol	Probability
*	0.55
$x_1$	0.45



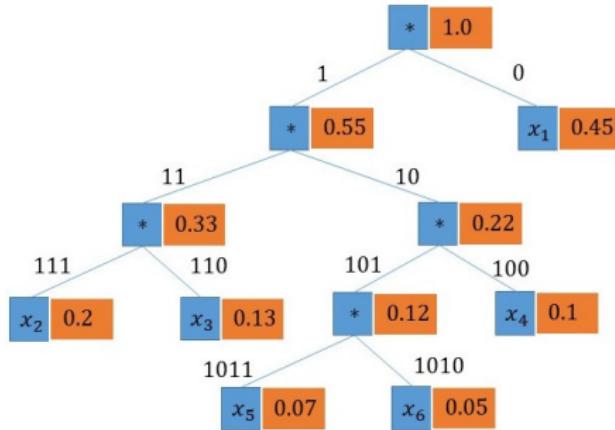
## Huffman Coding

## Source Coding Theorem

## Huffman Coding Algorithm: An Example

6

Symbol	Probability	Codeword
$x_1$	0.45	0
$x_2$	0.2	111
$x_3$	0.13	110
$x_4$	0.1	100
$x_5$	0.07	1011
$x_6$	0.05	1010



# Source Coding Theorem

Huffman Coding Algorithm: Numerical Analysis

Symbol	Probability $p_k$	Codeword	Length $L_k$	$p_k L_k$	$-\log_2 p_k$	$-p_k \log_2 p_k$
$x_1$	0.45	0	1	0.45	1.15	0.518
$x_2$	0.2	111	3	0.6	2.32	0.464
$x_3$	0.13	110	3	0.39	2.94	0.382
$x_4$	0.1	100	3	0.3	3.32	0.332
$x_5$	0.07	1011	4	0.28	3.83	0.268
$x_6$	0.05	1010	4	0.2	4.32	0.216
				2.2		2.18

# Source Coding Theorem

## Huffman Coding Algorithm: Summary

- How do we get even closer to Entropy?
  - Use Huffman Algorithm on pairs of symbols, or
  - Even better, on triplets of symbols, or quadruplets or,
  - In general,  $m$ -tuple of symbols.
- A limitation of Huffman Algorithm is that it requires the source symbol probabilities to be known. These are not known exactly but they can be estimated from a sufficiently large sample of source output. Mismatch between the estimated and exact symbol probabilities contributes to a degraded performance of Huffman's Algorithm.
- Another limitation of Huffman Code is that the symbol probabilities are assumed to be fixed. If these probabilities change with time, Huffman Code may provide suboptimal data compression.

Lempel-Ziv Coding

# Lempel-Ziv Coding

## Encoding Algorithm

- In Lempel-Ziv coding, the source sequence is sequentially parsed into strings that have not appeared so far.
  - Continually parse the symbol sequence and insert a comma on detecting the shortest sequence that has not been seen earlier
- Example sequence: 1011010100010...
- Sequential parsing method in Lempel-Ziv encoding:
  - 1,011010100010...
  - 1,0,11010100010...
  - 1,0,11,010100010...
  - 1,0,11,01,0100010...
  - 1,0,11,01,010,0010...
  - 1,0,11,01,010,00,10...
  - 1,0,11,01,010,00,10,...

## Lempel-Ziv Coding

# Lempel-Ziv Coding

## Encoding Algorithm

- Prefix of each “phrase” (phrase is the symbol string between two commas), i.e., the string consisting of all but the last symbol of each phrase, must have occurred earlier.
- Encode this phrase by giving the location of the prefix and the value of the last bit.
- Let  $C(n)$  be the number of phrases in the parsing of the input sequence with  $n$  symbols.
- Thus,  $\log C(n)$  bits are needed to describe the location of the prefix to the phrase and one bit to describe the last bit.
- In the above example, the code sequence is:  
 $(000, 1), (000, 0), (001, 1), (010, 1), (100, 0), (010, 0), (001, 0)$ 
  - First number of each pair gives the index of the prefix and the second number gives the last bit of the phrase.