# Elements of Information Theory

# Lecture 4
# *Data Compression (Source Coding)*


***Instructor: Yichen Wang***

***Ph.D./Professor***


**School of Information and Communications Engineering**
**Division of Electronics and Information**
**Xi'an Jiaotong University**

# Outlines

- **AEP for Data Compression (Source Coding)**
- **Fixed-Length Coding**
- **Variable-Length Coding**
- **Optimal Codes**
- **Huffman Code**

# AEP for Data Compression

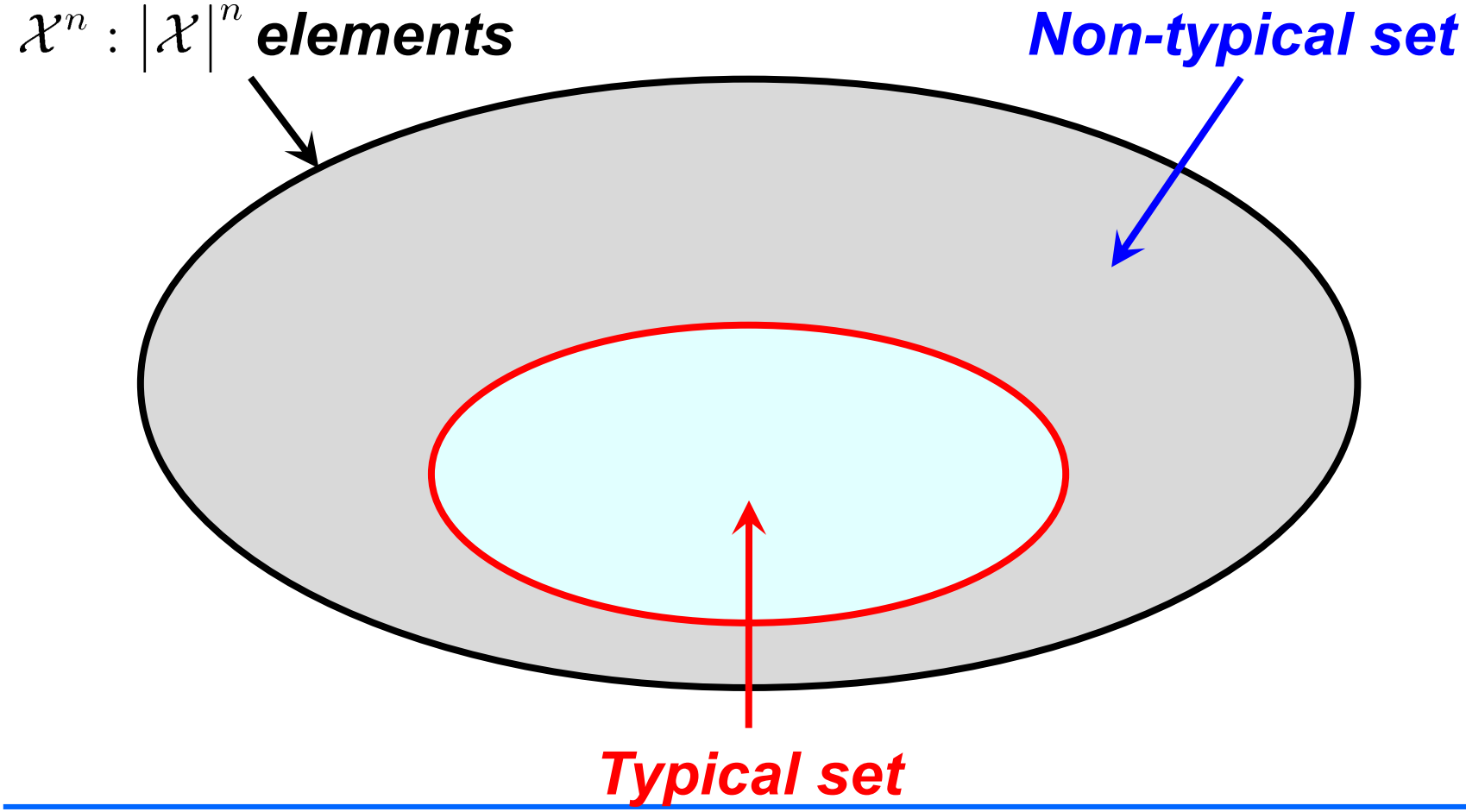
***Let $X_1, X_2, \ldots, X_n$ be independent, identically distributed (i.i.d.) random variables with probability mass function p(x).***

***How many bits are enough for describing the sequence of random variables $X_1, X_2, \ldots, X_n$?***

***Typical Sequence and Typical Set***

# AEP for Data Compression

***Divide all sequences in $\mathcal{X}^n$ into two sets***




$A_\epsilon^{(n)} : \leq 2^{n(H+\epsilon)}$ ***elements***

# AEP for Data Compression

***Indexing sequences in typical set***

- ***No more than n(H+ε)+1 bits (Why having 1 extra bit?)***
- ***Prefix all these sequences by a 0***
- ***The required total length is no more than n(H+ε)+2 bits***

***Indexing sequences in atypical set***

- ***No more than*** $n\log|\mathcal{X}|+1$ ***bits (Why?)***
- ***Prefixg all these sequences by a 1***
- ***The required total length is no more than*** $n\log|\mathcal{X}|+2$ ***bits***

# AEP for Data Compression

- Use $x^n$ to denote a sequence $x_1, x_2, \ldots, x_n$
- Let $l(x^n)$ be the length of the codeword corresponding to $x^n$
- The expected length of the codeword is

$$
\begin{aligned}
\mathbb{E}\{l(x^n)\} &= \sum_{x^n} p(x^n) l(x^n) \\
&= \sum_{x^n \in A_\epsilon^{(n)}} p(x^n) l(x^n) + \sum_{x^n \in A_\epsilon^{(n)c}} p(x^n) l(x^n) \\
&\leq \sum_{x^n \in A_\epsilon^{(n)}} p(x^n) \Big[ n(H+\epsilon) + 2 \Big] + \sum_{x^n \in A_\epsilon^{(n)c}} p(x^n) \Big[ n\log|\mathcal{X}| + 2 \Big] \\
&= \Pr\Big\{ A_\epsilon^{(n)} \Big\} \Big[ n(H+\epsilon) + 2 \Big] + \Pr\Big\{ A_\epsilon^{(n)c} \Big\} \Big[ n\log|\mathcal{X}| + 2 \Big] \\
&\leq n(H+\epsilon) + 2 + \epsilon \Big[ n\log|\mathcal{X}| + 2 \Big] \\
&= n \left( H + \epsilon + \epsilon\log|\mathcal{X}| + \frac{2}{n} + \frac{2\epsilon}{n} \right)
\end{aligned}
$$

*Can be arbitrary small*

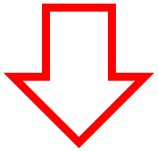
# AEP for Data Compression

***Theorem***

***Let $X^n$ be i.i.d. ~ p(x). Let $\varepsilon > 0$. Then there exists a code that maps sequences $x^n$ of length n into binary strings such that the mapping is one-to-one (and therefore invertible) and***

$$\mathbb{E}\left\{\frac{1}{n}l\left(x^n\right)\right\} \leq H(X) + \epsilon$$

***for n sufficiently large.***



***We can represent sequences $X^n$ using nH(X) bits on average.***

# Outlines

- AEP for Data Compression (Source Coding)
- Fixed-Length Coding
- Variable-Length Coding
- Optimal Codes
- Huffman Code

# Fixed-Length Coding

**Consider a sequence of *L* consecutive letters from the discrete source**

$$\boldsymbol{u}^L = \left(u_1, u_2, \cdots, u_L\right),\ u_l \in \left\{a_1, a_2, \cdots, a_K\right\},\ \forall l$$

**Encode such sequences into *fixed-length* code words. Moreover, we assume that the *code alphabet contains D symbols* and the *length of each code word is N*.**

**If we wish to *provide a separate code word for each source sequence*, we must have**

$$\frac{N}{L} \geq \frac{\log K}{\log D}$$

***What does it mean?***

# Fixed-Length Coding

***Question 1:***

***If we wish to use fewer than logK/logD code letters per source symbol, what should we do?***

***------ We must relax our insistence on always being able to decode the source sequence from the code sequence.***

***Question 2:***

***If we assign the code words only to a subset of the source sequences of length L, what will happen when L is sufficient large?***

# Fixed-Length Coding

**We also consider the sequence of *L* consecutive letters from a discrete source**

$$\boldsymbol{u}^L = \left(u_1, u_2, \cdots, u_L\right),\ u_l \in \left\{a_1, a_2, \cdots, a_K\right\}$$

**For the *discrete memoryless source*, the probability of a given sequence $\boldsymbol{u}^L$ is**

$$\Pr\left\{\boldsymbol{u}^L\right\} = \prod_{l=1}^{L} P\left(u_l\right)$$

**Moreover, the *self-information* of the sequence is**

$$I\left(\boldsymbol{u}^L\right) = \sum_{l=1}^{L} I\left(u_l\right)$$

# Fixed-Length Coding

**The law of large numbers tells us that**

$$\frac{I(\boldsymbol{u}^L)}{L} \longrightarrow H(U), \text{ if } L \text{ is sufficiently large}$$

**The probability of a given *L*-length sequence**

$$\Pr\{\boldsymbol{u}^L\} \longrightarrow 2^{-LH(U)}$$

**The number of such *L*-length sequences**

$$M_T \longrightarrow 2^{LH(U)}$$

***Typical sequence and Typical set***

# Fixed-Length Coding

**By employing the *weak law of large numbers*, we have**

$$\Pr\left\{\left|\frac{I(\boldsymbol{u}^L)}{L} - H(U)\right| > \delta\right\} \leq \epsilon$$

**Let *T* be the set of sequences $\boldsymbol{u}^L$, for which**

$$\left|\frac{I(\boldsymbol{u}^L)}{L} - H(U)\right| \leq \delta,\ \boldsymbol{u}^L \in T$$

**Then, we have that the probability of event *T* satisfies**

$$\Pr\{T\} \geq 1 - \epsilon$$

**The number of sequences in set *T* satisfies the following two constraints**

$$\begin{cases} M_T \leq 2^{L[H(U)+\delta]} \\ M_T \geq [1-\epsilon]2^{L[H(U)-\delta]} \end{cases}$$

# Fixed-Length Coding

***Theorem (Fixed-Length Source Coding Theorem)***

***Let a discrete memoryless source have finite entropy H(U) and consider a coding from sequences of L source letters into sequences of N code letters from a code alphabet of size D. Only one source sequence can be assigned to each code sequence and we let $P_e$ be the probability of occurrence of a source sequence for which no code sequence has been provided.***

***Then, for any δ>0, if***

$$\frac{N}{L} \geq \frac{H(U)+\delta}{\log D}$$

***$P_e$ can be made arbitrarily small by making L sufficiently large.***

***Conversely, if***

$$\frac{N}{L} \leq \frac{H(U)-\delta}{\log D}$$

***then $P_e$ must become arbitrarily close to 1 as L is made sufficiently large.***

# Fixed-Length Coding

***Question:***

***Is fixed-length coding good enough?***

**Coding Efficiency**

$$\eta = \frac{H(U)}{\frac{N}{L}\log D}$$

***The number of bits for describing each source symbol after source coding***

***We hope that the coding efficiency can reach 1, i.e.,*** $\eta \to 1$

***For fixed-length coding, we have***

$$\eta = \frac{H(U)}{H(U)+\delta} \longrightarrow \delta = \frac{1-\eta}{\eta} H(U)$$

# Fixed-Length Coding

**The error probability $P_e$**

$$P_e \le \epsilon(L,\delta) = \frac{D\left[I(u_l)\right]}{L\delta^2}$$

***For the given variance and parameter δ, the error probability $P_e$ can be smaller than arbitrary positive number if L is sufficiently large.***

$$L \ge \frac{D\left[I(u_l)\right]}{H^2(U)} \frac{\eta^2}{(1-\eta)^2\epsilon}$$

$$D\left[I(u_l)\right] = \sum_{u_l \in \mathcal{U}} P(u_l)\left[\log P(u_l)\right]^2 - H^2(U)$$

***The required length L increases as $P_e$ decreases!***

# Fixed-Length Coding

***Example***

***The alphabet for the source is {$a_1$, $a_2$, $a_3$, $a_4$} and the corresponding probability distribution is 1/2, 1/4, 1/8, and 1/8, respectively. If we require the coding efficiency is no smaller than 95% and the error probability $P_e$ cannot larger than $10^{-6}$, please calculate the required minimum length of source letter sequence.***

$$H(U) = -\sum_{i=1}^{4} P(a_i)\log P(a_i) = 1.75 \text{ bits}$$

$$D\left[I(u)\right] = \sum_{i=1}^{4} P(a_i)\left[\log P(a_i)\right]^2 - H^2(U) = 0.6875 \text{ bits}^2$$

# Fixed-Length Coding

***Example***

***The alphabet for the source is $\{a_1, a_2, a_3, a_4\}$ and the corresponding probability distribution is 1/2, 1/4, 1/8, and 1/8, respectively. If we require the coding efficiency is no smaller than 95% and the error probability $P_e$ cannot larger than $10^{-6}$, please calculate the required minimum length of source letter sequence.***

$$L \geq \frac{D\left[I(u)\right]}{H^2(U)} \frac{\eta^2}{(1-\eta)^2 \varepsilon} = 0.81 \times 10^8$$

***Fixed-length coding: Storage and Delay***

# Outlines

- AEP for Data Compression (Source Coding)
- Fixed-Length Coding
- Variable-Length Coding
- Optimal Codes
- Huffman Code

# Variable-Length Coding

***Example***

***The alphabet for the source is $\{a_1, a_2, a_3, a_4\}$ and the corresponding probability distribution is 1/2, 1/4, 1/8, and 1/8, respectively. If we encode source symbols as follows:***

| $a_1$ | $a_2$ | $a_3$ | $a_4$ |
|---|---|---|---|
| 0 | 10 | 110 | 111 |

***Then, we have***

$$\overline{n} = \sum_{i=1}^{4} p_i n_i = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + 2 \times \frac{1}{8} \times 3 = 1.75 \text{ bits} \longrightarrow \eta = \frac{H(U)}{\overline{n}} = 1$$

# Variable-Length Coding

***Most source coding is variable-length in the reality.***

***Basic Principles for Variable-Length Coding Design***

1. ***Assign short descriptions to the most frequent outcomes of the data source;***
2. ***Assign longer descriptions to the less frequent outcomes.***

***Morse Code***

***Morse code is a reasonably efficient code for English alphabet using an alphabet of four symbols: dot, dash, letter space, and word space.***

# Variable-Length Coding

**International Morse Code**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | ·– | J | ·––– | S | ··· | 1 | ·———— |
| B | –··· | K | –·– | T | – | 2 | ··––– |
| C | –·–· | L | ·–·· | U | ··– | 3 | ···–– |
| D | –·· | M | –– | V | ···– | 4 | ····– |
| E | · | N | –· | W | ·–– | 5 | ····· |
| F | ··–· | O | ––– | X | –··– | 6 | –···· |
| G | ––· | P | ·––· | Y | –·–– | 7 | ––··· |
| H | ···· | Q | ––·– | Z | ––·· | 8 | –––·· |
| I | ·· | R | ·–· | 0 | ————— | 9 | ————· |

1. *The length of a dot is one unit;*
2. *A dash is three units;*
3. *The space between parts of the same letter is one unit;*
4. *The space between letters is three units;*
5. *The space between words is seven units;*

***Short sequences represent frequent letters (e.g., E); long sequences represent infrequent letters (e.g., Q).***

# Variable-Length Coding

***For a given source, we can design many source coding schemes.***

| *Source Letters* | $P(a_k)$ | *Code 1* | *Code 2* | *Code3* | *Code 4* |
|---|---|---|---|---|---|
| $a_1$ | *0.5* | *0* | *0* | *0* | *0* |
| $a_2$ | *0.25* | *0* | *1* | *01* | *10* |
| $a_3$ | *0.125* | *1* | *00* | *011* | *110* |
| $a_4$ | *0.125* | *10* | *11* | *0111* | *1110* |

***Question:***

***Which coding scheme is the best?***

# Variable-Length Coding

***Definition (Source Code)***

***A source code C for a random variable X is a mapping from $\mathcal{X}$, the range of X, to $\mathcal{D}^*$, the set of finite-length strings of symbols from a D-ary alphabet.***

***Let C(x) denote the codeword corresponding to x.***

***Let l(x) denote the length of C(x).***

***For example***

***C(red) = 00 and C(blue) = 11 are source codes for $\mathcal{X} = \{red, blue\}$ with alphabet $\mathcal{D} = \{0, 1\}$.***

# Variable-Length Coding

***Definition***

***The expected length L(C) of a source code C(x) for a random variable X with probability mass function p(x) is given by***

$$L(C) = \sum_{x \in \mathcal{X}} p(x) l(x)$$

***where l(x) is the length of the codeword associated with x.***

***The expected length is a critically important metric to judge the source coding scheme!!!***

# Variable-Length Coding

***Definition***

***A code is said to be nonsingular if every element of the range of X maps into a different string in $\mathcal{D}^\star$; that is,***

$$x \neq x' \Longrightarrow C(x) \neq C(x')$$

***Sending sequences of symbols X***

***Definition***

***The extension $C^*$ of a code C is the mapping from finite-length strings of $\mathcal{X}$ to finite-length of $\mathcal{D}$, defined by***

$$C(x_1 x_2 \cdots x_n) = C(x_1)C(x_2)\cdots C(x_n)$$

***where $C(x_1)C(x_2)\cdots C(x_n)$ indicates concatenation of the corresponding codewords.***

# Variable-Length Coding

***Definition***

***A code is called uniquely decodable if its extension is nonsingular.***

1. ***Any encoded strings comes from an unique sequence of source symbols;***
2. ***For any sequence of source symbol, the sequence can be reconstructed unambiguously from the encoded bit sequence.***

# Variable-Length Coding

***Challenge for Uniquely Decodable Code Design***

***Decode the codewords from the encoded sequence as fast as possible ---- reducing decoding delay.***

***Definition***

***A code is called a prefix code or an instantaneous code if no codeword is a prefix of any other codeword.***

⇩

***The decoder can decode each codeword of a prefix code immediately on the arrival of the last bit in that word.***

# Variable-Length Coding

***Example***

| *Source Letters* | $P(a_k)$ | *Code 1* | *Code 2* | *Code3* | *Code 4* |
|---|---|---|---|---|---|
| $a_1$ | *0.5* | *0* | *0* | *0* | *0* |
| $a_2$ | *0.25* | *0* | *1* | *01* | *10* |
| $a_3$ | *0.125* | *1* | *00* | *011* | *110* |
| $a_4$ | *0.125* | *10* | *11* | *0111* | *1110* |

***Code 1: Singular***

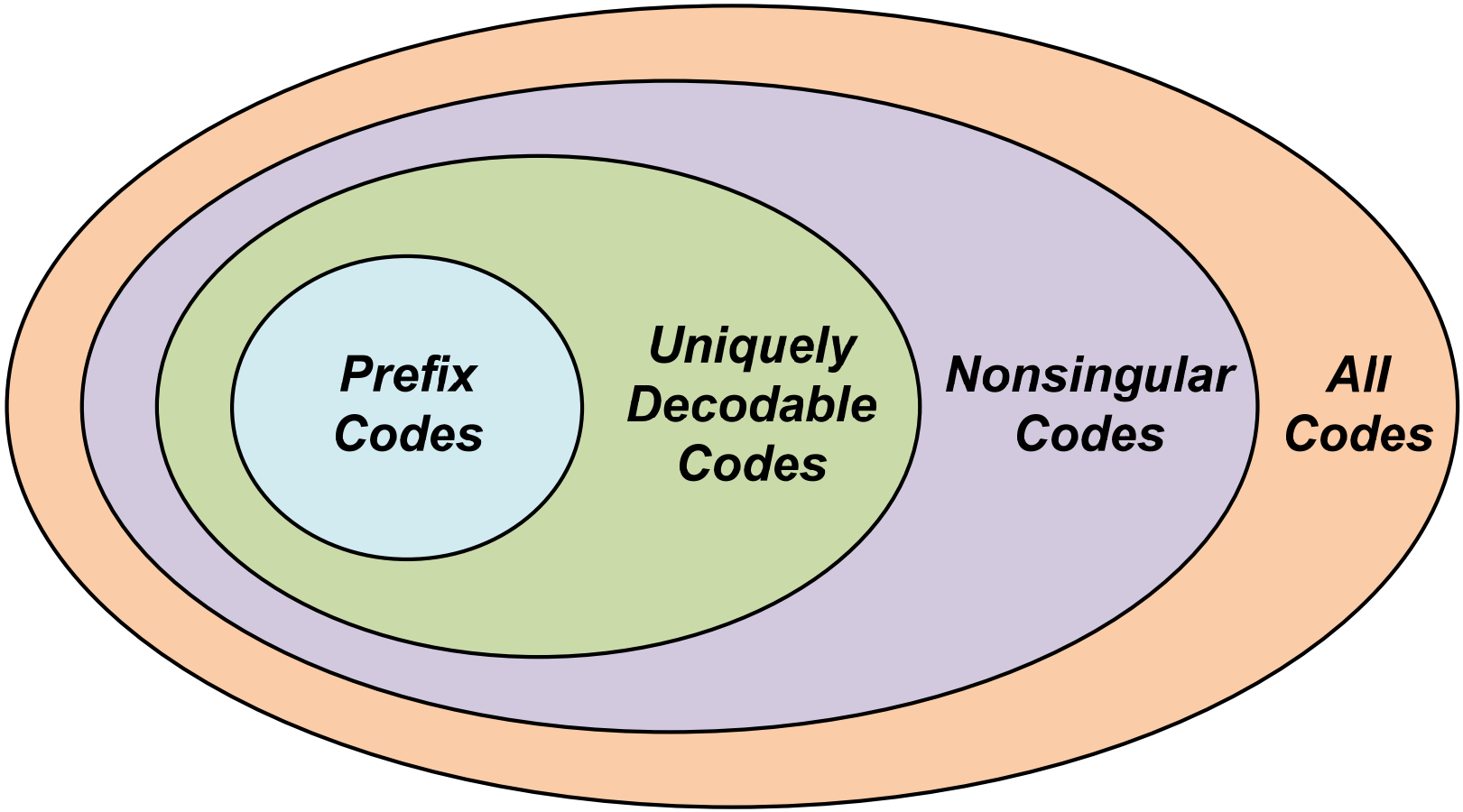
***Code 2: Nonsingular, but not uniquely decodable***

***Code 3: Uniquely decodable, but not instantaneous***

***Code 4: Instantaneous, also prefix codes***

# Variable-Length Coding

***Relationships among Prefix Code, Uniquely decodable codes, and Other Codes***



# Variable-Length Coding

## *Code Tree (Binary Case)*

- *A corresponding binary code tree can be defined which grows from a root on the left to leaves on the right representing codewords.*
- *Each branch is labeled 0 or 1 and each string corresponding to the branch labels from the root to that node.*
- *The tree is extended just enough to include each codeword.*

***Binary Code Tree ⟶ D-ary Code Tree***

# Variable-Length Coding

***Relationship between Prefix Code and Code Tree***

- *Root ➔ Codeword beginning*
- *Number of branches ➔ Number of symbols (#-ary)*
- *Intermedium node ➔ Part of a codeword*
- *Leaves ➔ A codeword*
- *Full tree ➔ Fixed-length code*
- *Not full tree ➔ Variable-length code*

***The prefix condition implies that no codeword is an ancestor of any other codeword on the tree. Hence, each codeword eliminates its descendants as possible codewords.***

# Variable-Length Coding

***What is the ideal source code for us?***

- ***Prefix code (Instantaneous code)***
- ***Minimize the expected length as much as possible***

***Theorem (Kraft Inequality)***

***For any instantaneous code (prefix code) over an alphabet of size D, the codeword lengths $l_1$, $l_2$, … , $l_m$ must satisfy the inequality***

$$\sum_{i=1}^{m} D^{-l_i} \leq 1$$

***Conversely, given a set of codeword lengths that satisfy this inequality, there exists an instantaneous code with these word lengths.***

# Variable-Length Coding

***Some Discussions***

- ***Any codeword set that satisfies the prefix condition has to satisfy the Kraft inequality. (necessary condition)***
- ***Kraft inequality is a sufficient condition for the existence of codeword set with the specified set of codeword lengths.***

# Variable-Length Coding

***Theorem (Extended Kraft Inequality)***

***For any countably infinite set of codewords that form a prefix code, the codeword lengths satisfy the extended Kraft inequality,***

$$\sum_{i=1}^{\infty} D^{-l_i} \leq 1$$

***Conversely, given any $l_1$, $l_2$, … satisfying the extended Kraft inequality, we can construct a prefix code with these codeword lengths.***

# Outlines

- AEP for Data Compression (Source Coding)
- Fixed-Length Coding
- Variable-Length Coding
- Optimal Codes
- Huffman Code

# Optimal Codes

***Prefix code with the minimum expected length***

- ***The codeword length of prefix code, $l_1, l_2, \ldots, l_m$, satisfy the Kraft equality***
- ***The expected length is less than the expected length of any other prefix code***

$$\underset{l_1,l_2,\cdots,l_m}{\text{Minimize}} \quad L = \sum_{i=1}^{m} p_i l_i$$

$$\text{s.t.} \quad \sum_{i=1}^{m} D^{-l_i} \leq 1$$

$$l_1, l_2, \cdots, l_m \text{ are integers}$$

# Optimal Codes

***We first neglect the integer constraint on $l_i$***

$$\underset{l_1,l_2,\cdots,l_m}{\text{Minimize}} \quad L = \sum_{i=1}^{m} p_i l_i$$

$$\text{s.t.} \qquad \sum_{i=1}^{m} D^{-l_i} \le 1$$

***Lagrangian Multiplier***

$\lambda \ge 0$

***Construct the Lagrangian function:***

$$\mathcal{J}\left(\{l_i\}_{i=1}^{m}, \lambda\right) = \sum_{i=1}^{m} p_i l_i + \lambda\left(\sum_{i=1}^{m} D^{-l_i} - 1\right)$$

***Differentiating with respect to $l_i$, we can obtain***

$$\frac{\partial \mathcal{J}}{\partial l_i} = p_i - \lambda D^{-l_i} \log_e D$$

# Optimal Codes

***Setting the derivative to 0, we can obtain***

$$D^{-l_i} = \frac{p_i}{\lambda \log_e D}$$

***We hope the following equality can be held:***

$$\sum_{i=1}^{m} D^{-l_i} = 1$$

***Then, we derive that the optimal Lagrangian multiplier is***

$$\lambda^* = 1/\log_e D$$

***Consequently, we have***

$$l_i^* = -\log_D p_i$$

# Optimal Codes

***This noninteger choice of codeword lengths yields expected codeword length***

$$L^* = \sum_{i=1}^{m} p_i l_i^* = -\sum_{i=1}^{m} p_i \log_D p_i = H_D(X)$$

***Choose a set of codeword lengths $l_i$ "close" to the optimal set.***

***Theorem***

***The expected length L of any instantaneous D-ary code for a random variable X is greater than or equal to the entropy $H_D(X)$; that is***

$$L \geq H_D(X)$$

***with equality if and only if*** $D^{-l_i} = p_i$ .

# Optimal Codes

***We have obtained the lowerbound of the optimal expected length. How about the upperbound?***

***Choose the set of codeword length $\{l_1, l_2, \ldots, l_m\}$ such that***

$$l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil, \quad i = 1, 2, \cdots, m$$

$$\sum_{i=1}^{m} D^{-l_i} = \sum_{i=1}^{m} D^{-\left\lceil \log_D \frac{1}{P_i} \right\rceil} \leq \sum_{i=1}^{m} D^{-\log_D \frac{1}{P_i}} = 1$$

$$\downarrow \quad \log_D \frac{1}{p_i} \leq l_i < \log_D \frac{1}{p_i} + 1$$

$$H_D(X) \leq L < H_D(X) + 1$$

# Optimal Codes

***Theorem***

***Let*** $l_1^\star, l_2^\star, \cdots, l_m^\star$ ***be optimal codeword lengths for a source distribution*** $\mathbf{p}$ ***and a D-ary alphabet, and let be the associated expected length of an optimal code***

$$L^\star = \sum_{i=1}^{m} p_i l_i^\star$$

***Then, we have***

$$H_D(X) \leq L^\star < H_D(X) + 1$$

- ***There is an overhead that is at most 1 bit due to the integer constraint***
- ***We can reduce the overhead per symbol by spreading it out over many symbols.***

# Optimal Codes

***Consider n consequent symbols, we have***

$$L_n = \frac{1}{n}\sum p(x_1, x_2, \cdots, x_n) l(x_1, x_2, \cdots, x_n) = \frac{1}{n}\mathbb{E}\Big\{ l(X_1, X_2, \cdots, X_n) \Big\}$$

$$H(\underbrace{X_1, X_2, \cdots, X_n}_{\textit{i.i.d.}}) \leq \mathbb{E}\Big\{ l(X_1, X_2, \cdots, X_n) \Big\} < H(X_1, X_2, \cdots, X_n) + 1$$

$$\longrightarrow \quad H(X) \leq L_n < H(X) + \frac{1}{n}$$

***By using large block lengths we can achieve an expected code-length per symbol arbitrarily close to the entropy.***

***Theorem***

***The minimum expected codeword length per symbol satisfies***

$$\frac{H(X_1, X_2, \cdots, X_n)}{n} \leq L_n^{\star} < \frac{H(X_1, X_2, \cdots, X_n)}{n} + \frac{1}{n}$$

# Outlines

- AEP for Data Compression (Source Coding)
- Fixed-Length Coding
- Variable-Length Coding
- Optimal Codes
- Huffman Code

# Huffman Code

- *We have discussed several properties of the optimal codes*
- *The problem is, how to construct an optimal code?*
- *Huffman code is an optimal (shortest expected length) prefix code for a given distribution*
- *There are many optimal codes*
- *Huffman procedure constructs one such optimal code by a simple algorithm*

# Huffman Code

## *Procedure of Huffman Codes (Binary Case)*

| *Source X* | $x_1$ | $x_2$ | …… | $x_m$ |
|---|---|---|---|---|
| *Probability* | $p_{x1}$ | $p_{x2}$ | …… | $p_{xm}$ |

***$x_i$ (i = 1, 2, … , m) ------ Binary strings***

1. ***Rearrange the m source symbols in a descent order according to their probabilities, i.e.,*** $p_1 \geq p_2 \geq \cdots \geq p_{m-1} \geq p_m$ ***;***
2. ***Assign codeword 0 and 1 to the two source symbols with the least probabilities, and combine these two messages into a new symbol with the probability that is the sum of the least two probabilities;***

# Huffman Code

***Procedure of Huffman Codes (Binary Case)***

| *Source X* | $x_1$ | $x_2$ | …… | $x_m$ |
|---|---|---|---|---|
| *Probability* | $p_{x1}$ | $p_{x2}$ | …… | $p_{xm}$ |

$x_i$ *(i = 1, 2, … , m) ------ Binary strings*

3. *Rearrange the new source symbols in descent order according to their new probabilities;*

4. *Take the procedure iteratively until all messages are combined.*

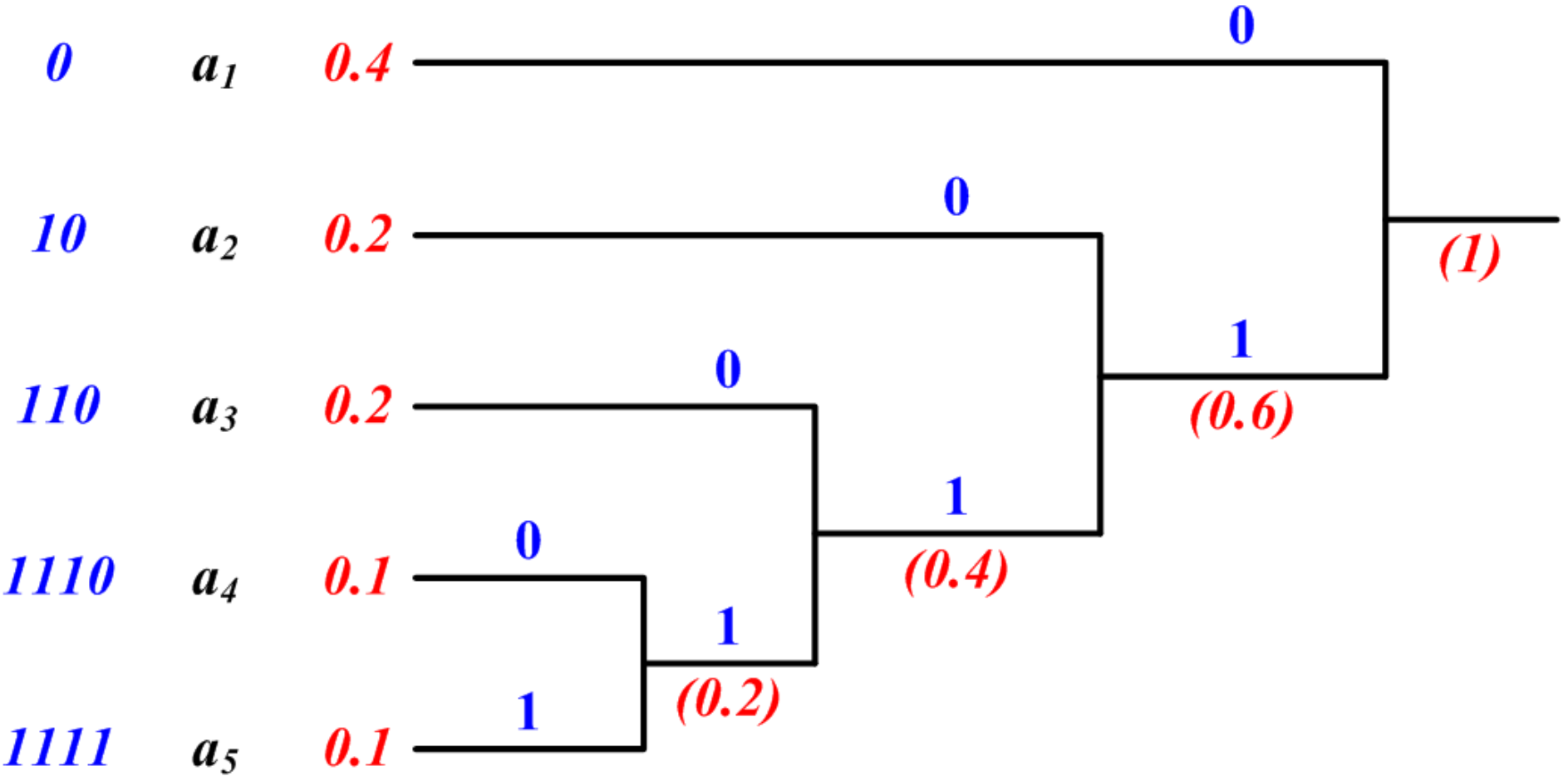
# Huffman Code

***Example***

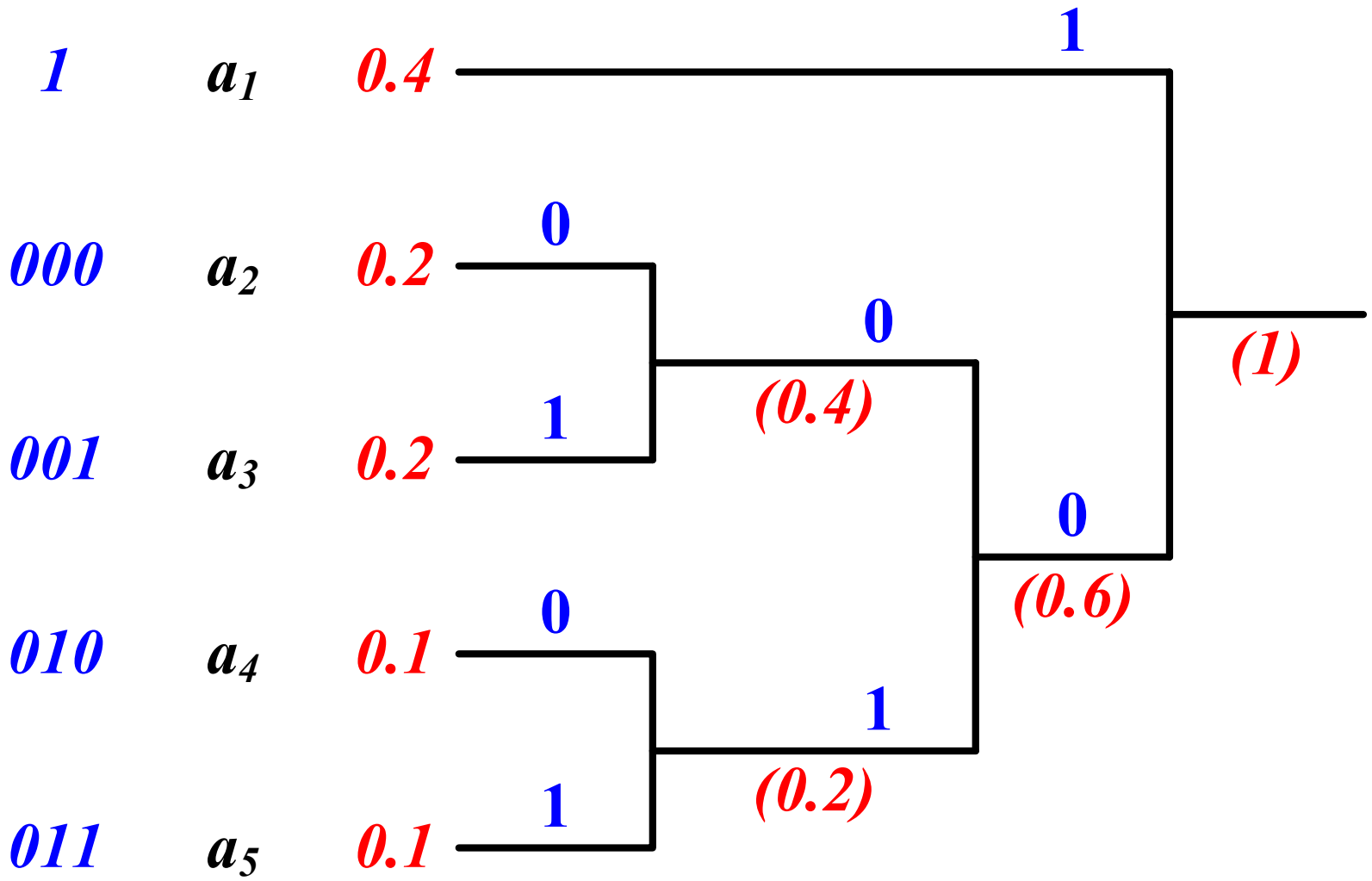
***Please find the Huffman code of the following source, and the coding efficiency.***

| *Source X* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| *Probability p(X)* | *0.4* | *0.2* | *0.2* | *0.1* | *0.1* |

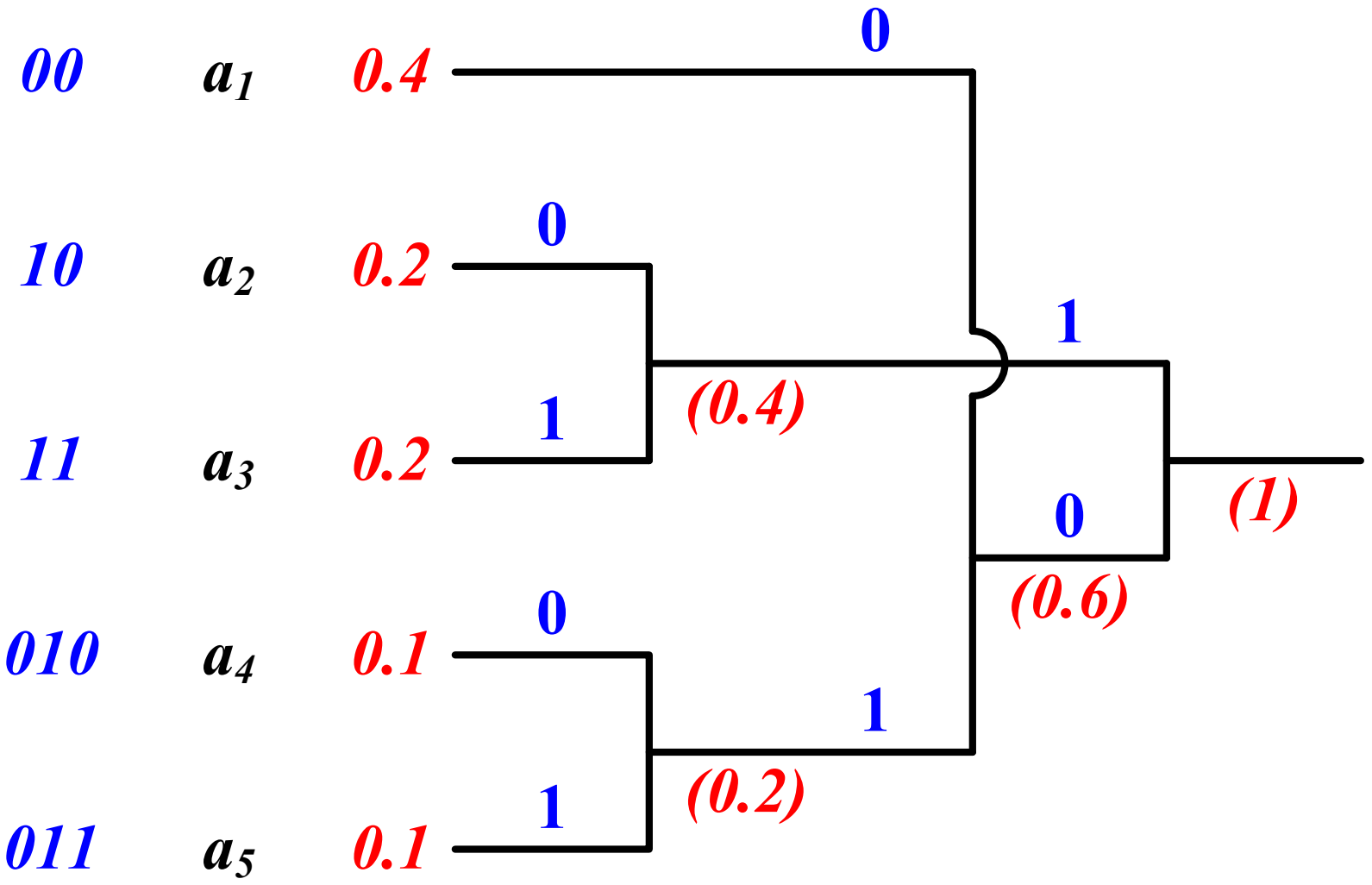
# Huffman Code



# Huffman Code



# Huffman Code



# Huffman Code

***The average length***

$$\overline{L}_1 = 0.4 \times 1 + 0.2 \times 2 + 0.2 \times 3 + 0.1 \times 4 + 0.1 \times 4 = 2.2 \text{ bits}$$

$$\overline{L}_2 = 0.4 \times 1 + 0.2 \times 3 + 0.2 \times 3 + 0.1 \times 3 + 0.1 \times 3 = 2.2 \text{ bits}$$

$$\overline{L}_3 = 0.4 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.1 \times 3 + 0.1 \times 3 = 2.2 \text{ bits}$$

***The entropy of the source***

$$H(X) = -\sum_{i=1}^{5} p(x_i)\log p(x_i) = 2.122 \text{ bits}$$

***Coding efficiency***

$$\eta = \frac{H(X)}{L} = \frac{2.122}{2.2} = 96.4\%$$

# Huffman Code

***Question:***

***The above three obtained Huffman codes have the same average codeword length. However, which one is better?***

$$\sigma^2 = \mathbb{E}\left\{\left(l_i - \overline{L}\right)^2\right\} = \sum_{i=1}^{m} p_i \left(l_i - \overline{L}\right)^2$$

$$\sigma_1 = 0.4 \times 1.2^2 + 0.2 \times 0.2^2 + 0.2 \times 0.8^2 + 0.1 \times 1.8^2 \times 2 = 1.36$$

$$\sigma_2 = 0.4 \times 1.2^2 + 0.2 \times 0.8^2 \times 2 + 0.1 \times 0.8^2 \times 2 = 0.96$$

$$\sigma_3 = 0.4 \times 0.2^2 + 0.2 \times 0.2^2 \times 2 + 0.1 \times 0.8^2 \times 2 = 0.16$$

***The third one is the best!!!***

# Huffman Code

- *There are many optimal source codes. Huffman procedure constructs one such optimal code.*
- *Can we find some properties that at least one optimal code satisfies?*
- *We restrict our attention to prefix codes (instantaneous codes)*
- *Without loss of generality, we order the probability masses as follows:*

$$p_1 \geq p_2 \geq \cdots \geq p_m$$

# Huffman Code

***Lemma (Binary Case)***

***For any distribution, there exists an optimal instantaneous code (with minimum expected length) that satisfies the following properties:***

1. ***The lengths are ordered inversely with the probabilities (i.e., if $p_j > p_k$, then $l_j \leq l_k$).***
2. ***The two longest codewords have the same length.***
3. ***Two of the longest codewords differ only in the last bit and correspond to the two least likely symbols.***

# Huffman Code

***Proof for Property 1***

***Consider an optimal code $C_m$ in which $p_j > p_k$.***

***Consider another code $C'_m$ where the codewords j and k of $C_m$ interchanged.***

$$
\begin{aligned}
L(C'_m) - L(C_m) &= \sum p_i l'_i - \sum p_i l_i \\
&= p_j l_k + p_k l_j - p_j l_j - p_k l_k \\
&= (p_j - p_k)(l_k - l_j) \geq 0
\end{aligned}
$$

***Consequently, property 1 holds.***

# Huffman Code

- *The lemma tells us that if $p_1 \geq p_2 \geq \ldots \geq p_m$, there exists an optimal code with $l_1 \leq l_2 \leq \ldots \leq l_{m-1} = l_m$, and codewords $C(x_{m-1})$ and $C(x_m)$ that differ only in the last bit.*

- *We call codes that satisfy the properties in the lemma as canonical codes.*

- *By trimming, rearranging, and swapping, we can convert any prefix code to the canonical form.*

# Huffman Code

## *Optimality of Huffman Code*

| *Source X* | $x_1$ | $x_2$ | …… | $x_m$ |
|---|---|---|---|---|
| *Probability* | $p_1$ | $p_2$ | …… | $p_m$ |
| *Codeword-length* | $l_1$ | $l_2$ | …… | $l_m$ |

$$\mathbf{p} = (p_1, p_2, \cdots, p_{m-1}, p_m)$$

**Huffman reduction** ↓ ↑ **Extension**

$$\mathbf{p}' = (p_1, p_2, \cdots, p_{m-2}, p_{m-1} + p_m)$$

# Huffman Code

$$p'_i = \begin{cases} p_i, & i \le m-2 \\ p_{m-1} + p_m, & i = m-1 \end{cases} \qquad l_i = \begin{cases} l'_i, & i \le m-2 \\ l'_{m-1} + 1, & i = m-1, m \end{cases}$$

$$\overline{L}(\mathbf{p}) = \overline{L}(\mathbf{p}') + p'_{m-1}$$

- ***If a prefix code is optimal for X, then the corresponding prefix code for its Huffman reduction is also optimal.***
- ***If a prefix code is optimal for X, then the corresponding prefix code for its extension is also optimal.***

# Huffman Code

Optimal for $(x_1, x_2, \ldots, x_{m-1})$

$$\overline{L}(\mathbf{p}) = \overline{L}^*(\mathbf{p}') + p_{m-1} + p_m$$

$$\overline{L}(\mathbf{p}') = \overline{L}^*(\mathbf{p}) - p_{m-1} - p_m$$

Optimal for $(x_1, x_2, \ldots, x_m)$

$$\left[\overline{L}(\mathbf{p}') - \overline{L}^*(\mathbf{p}')\right] + \left[\overline{L}(\mathbf{p}) - \overline{L}^*(\mathbf{p})\right] = 0$$

***Theorem (Arbitrary D-ary coding)***

***Huffman coding is optimal; that is, if $C^*$ is a Huffman code and $C'$ is any other uniquely decodable code, we have***

$$\overline{L}(C^*) \leq \overline{L}(C')$$

# Huffman Code

***Question:***

***How to design Huffman code for any arbitrary D-ary source coding?***

***For D-ary source coding, define a complete code tree as a finite tree in which each intermediate node has D nodes of the next higher order stemming from it.***

***Lemma***

***The number of terminal nodes in a complete code tree with alphabet size D must be of the form D+m(D-1) for some integer m.***

# Huffman Code

***Binary case:***

***As D=2, the number of terminal nodes in the complete code tree can be written as m+2***

***If the number of source symbols is larger than 1, the source symbols can be represented by a complete code tree.***

***The above conclusion does not hold if D>2.***

# Huffman Code

***D-ary case (D>2):***

***For an optimal code, all the unused terminal nodes must have the same length as the longest code word.***

***All unused terminal nodes can be arranged to differ in only the last digit.***

***An optimal code tree must have, at most, D-2 unused terminal nodes.***

# Huffman Code

***D-ary case (D>2):***

***Denote the number of unused terminals by B. The source has K symbols. Then, we have***

$$B + K = D + m(D-1)$$

$$K - 2 = m(D-1) + (D-2-B)$$

***The remainder upon dividing K-2 by D-1*** → $R_{D-1}(K-2) = D-2-B$

$$D - B = 2 + R_{D-1}(K-2)$$

***The first step is to group together the least probable*** $D - B = 2 + R_{D-1}(K-2)$ ***nodes.***

# Huffman Code

***Example***

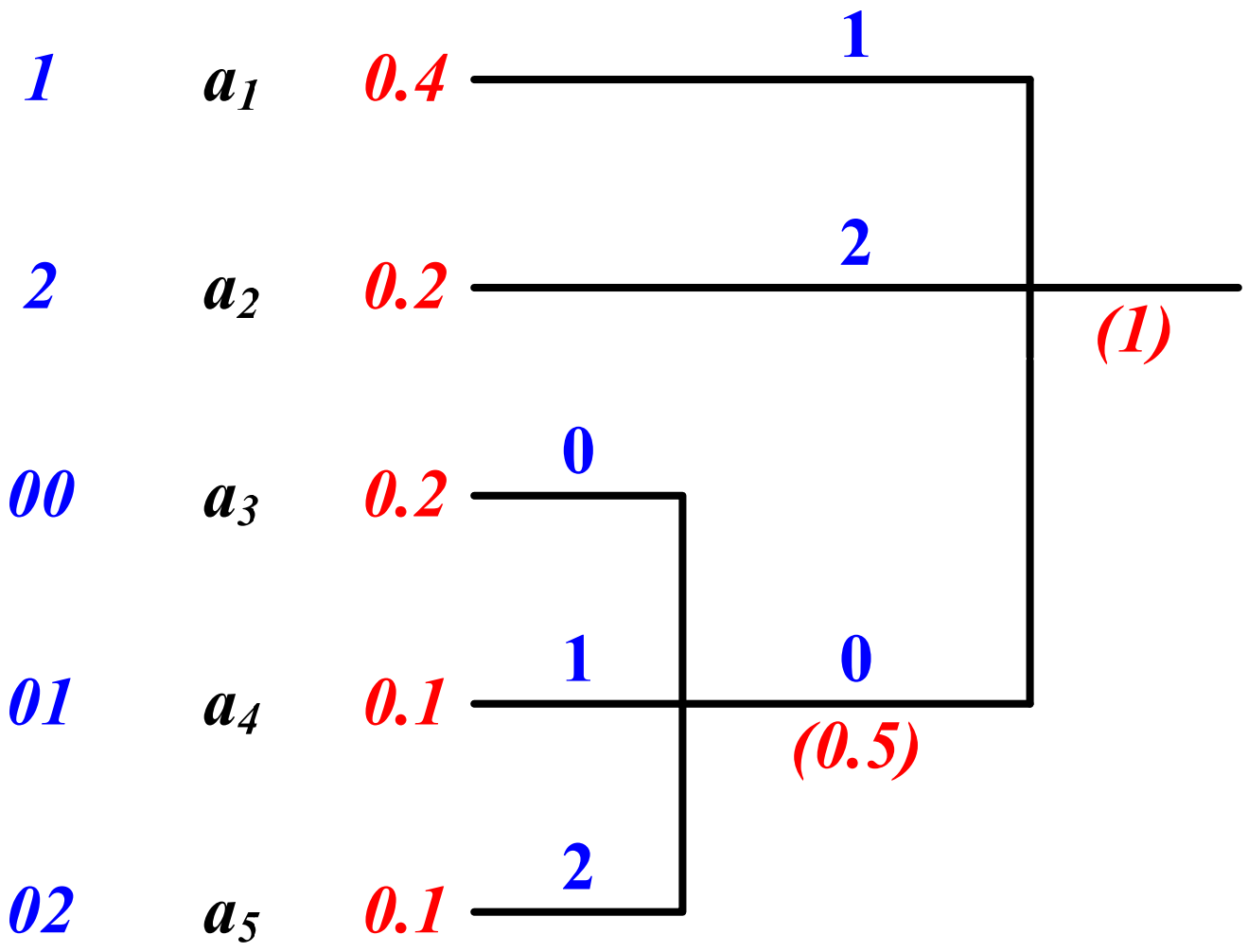
***Please find the Huffman code of the following source when D=3***

| *Source X* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| *Probability p(X)* | *0.4* | *0.2* | *0.2* | *0.1* | *0.1* |

$$D - B = 2 + R_{D-1}(K-2) = 2 + R_2(3) = 3$$

***The first step is to group together $x_3$, $x_4$, and $x_5$.***

# Huffman Code



# Huffman Code

***Example***

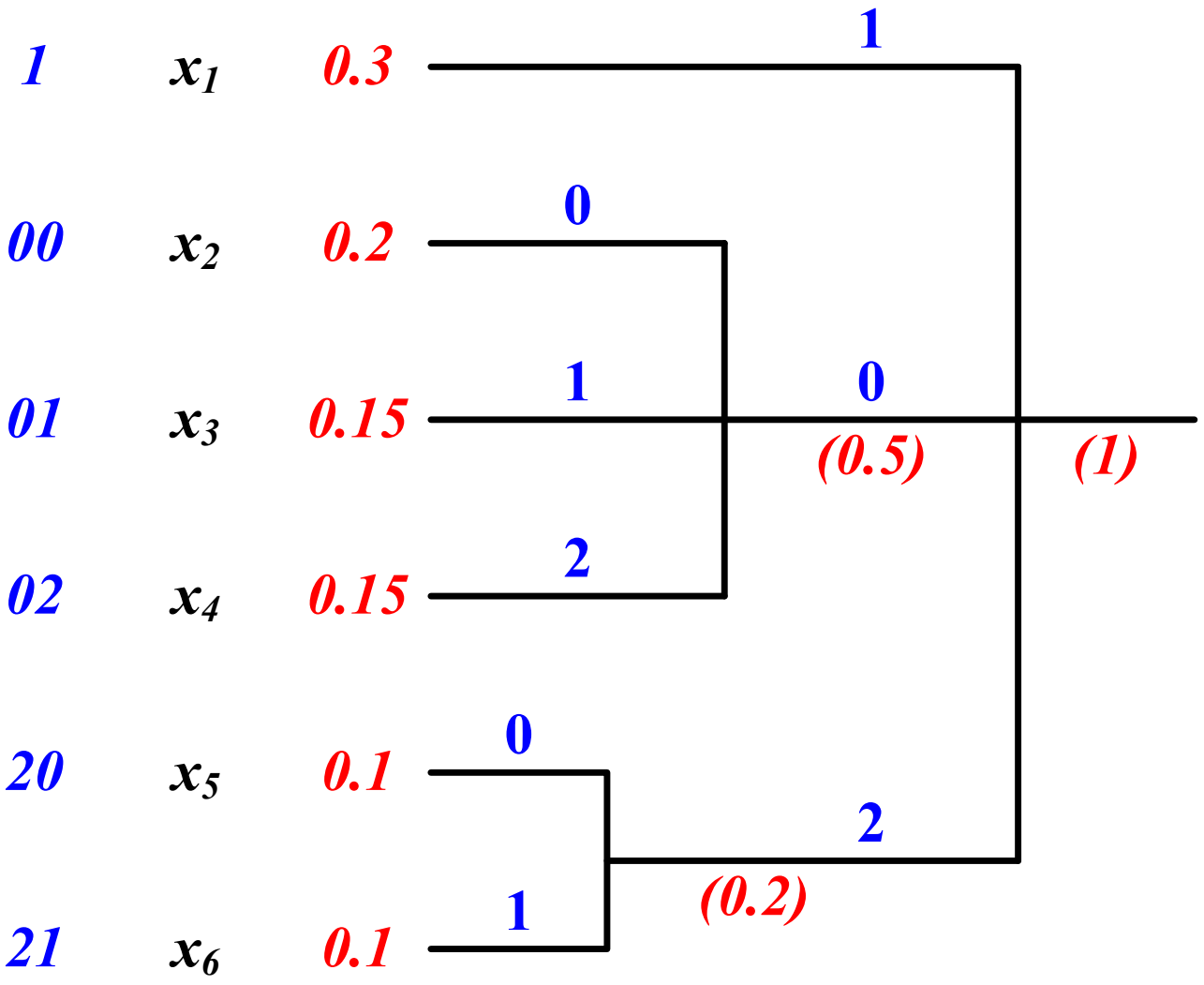
***Please find the Huffman code of the following source when D=3***

| *Source X* | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ |
|---|---|---|---|---|---|---|
| *Probability p(X)* | *0.3* | *0.2* | *0.15* | *0.15* | *0.1* | *0.1* |

$$D - B = 2 + R_{D-1}(K-2) = 2 + R_2(4) = 2$$

***The first step is to group together $x_5$ and $x_6$.***

# Huffman Code



# Review

## *1. AEP for Data Compression (Source Coding)*

***Basic idea:***

***Divide all sequences into two sets: Typical set and Atypical set***

***Theorem***

***Let $X^n$ be i.i.d. ~ p(x). Let $\varepsilon > 0$. Then there exists a code that maps sequences $x^n$ of length x into binary strings such that the mapping is one-to-one (and therefore invertible) and***

$$\mathbb{E}\left\{\frac{1}{n}l\left(x^n\right)\right\} \leq H(X) + \epsilon$$

***for n sufficiently large.***

# Review

## 2. Fixed-Length Source Coding

***Theorem (Fixed-Length Source Coding Theorem)***

***Let a discrete memoryless source have finite entropy H(U) and consider a coding from sequences of L source letters into sequences of N code letters from a code alphabet of size D. Only one source sequence can be assigned to each code sequence and we let $P_e$ be the probability of occurrence of a source sequence for which no code sequence has been provided.***

***Then, for any δ>0, if***

$$\frac{N}{L} \geq \frac{H(U)+\delta}{\log D}$$

***$P_e$ can be made arbitrarily small by making L sufficiently large.***

***Conversely, if***

$$\frac{N}{L} \leq \frac{H(U)-\delta}{\log D}$$

***then $P_e$ must become arbitrarily close to 1 as L is made sufficiently large.***

# Review

## *2. Fixed-Length Source Coding*

***Coding Efficiency***

$$\eta = \frac{H(U)}{\frac{N}{L}\log D}$$

→ ***The number of bits for describing each source symbol after source coding***

***How to determine the required codeword length L under the given source distribution, the targeted coding efficiency η, and the error probability $P_e$?***

# Review

## *3. Variable-Length Source Coding*

***Basic Principles:***

***Assign short descriptions to the most frequent outcomes of the data source and longer descriptions to the less frequent outcomes***

***Some Definitions:***

***Expected Length, Nonsingular, Extension, Uniquely Decodable, Prefix Code/Instantaneous Code***

***Theorem (Kraft Inequality)***

***For any instantaneous code (prefix code) over an alphabet of size D, the codeword lengths $l_1, l_2, \ldots, l_m$ must satisfy the inequality***

$$\sum_{i=1}^{m} D^{-l_i} \leq 1$$

***Conversely, given a set of codeword lengths that satisfy this inequality, there exists an instantaneous code with these word lengths.***

# Review

## *4. Optimal Source Coding*

***Theorem***

***Let*** $l_1^\star, l_2^\star, \cdots, l_m^\star$ ***be optimal codeword lengths for a source distribution*** **p** ***and a D-ary alphabet, and let*** $L^\star$ ***be the associated expected length of an optimal code***

$$L^\star = \sum_{i=1}^{m} p_i l_i^\star$$

***Then, we have***

$$H_D(X) \leq L^\star < H_D(X) + 1$$

***Theorem***

***The minimum expected codeword length per symbol satisfies***

$$\frac{H(X_1, X_2, \cdots, X_n)}{n} \leq L_n^\star < \frac{H(X_1, X_2, \cdots, X_n)}{n} + \frac{1}{n}$$

# Review

## *5. Huffman Code*

### *(1) Binary Huffman Code*

### *(2) Optimality of Huffman Code*

- *Huffman Reduction*
- *Extension*

### *(3) Arbitrary D-ary Huffman Code*

*How many symbols should be combined in the first step?*