Back to simple example of a source

\( (p_1, p_2, p_3, p_4) = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}) \)

\[ \text{Source} \quad \text{Encoder} \]

\( p[A] = \frac{1}{2} \quad p[B] = \frac{1}{4} \quad p[C] = \frac{1}{8} \quad p[D] = \frac{1}{8} \)

Assumptions:
1. One must be able to recover the source sequence from the binary sequence.
2. One knows the "start" of the binary sequence at the receiver.
3. One would like to minimize the average number of binary digits per source letter.

Examples of codes:

1.

\[
\begin{align*}
A & \rightarrow 00 \\
B & \rightarrow 01 \\
ABAC & \rightarrow 00010010 \rightarrow ABAC \\
C & \rightarrow 10 \\
D & \rightarrow 11
\end{align*}
\]

\( \text{AV} = 2 \)

2.

\[
\begin{align*}
A & \rightarrow 0 \\
B & \rightarrow 1 \\
C & \rightarrow 00 \\
D & \rightarrow 10
\end{align*}
\]

\( \text{AV} = \frac{1}{4} \)

Violates Assumption 2.

Can't use this code!!!
\[\begin{align*}
\theta(\frac{1}{4})A &\rightarrow 0 \\
\theta(\frac{1}{4})B &\rightarrow 10 \\
\theta(\frac{1}{8})C &\rightarrow 100 \\
\theta(\frac{1}{8})D &\rightarrow 111 \end{align*}\]

**Decoding Rule**

- Follow binary sequence until you reach a code word.
- Continue.

\[I = \frac{1}{2}x_1 + \frac{1}{4}x_2 + \frac{1}{8}x_3 + \frac{1}{8}x_3 = 1\frac{3}{4}\]

No code that satisfies assumptions can have smaller average (1 or binary digits per info symbol). We will find out why this is true later.

**Tree Description of Code**
**Some Definitions**

**Block Code** - Each source symbol is represented by some sequence of code symbols called code words.

**Non-Singular Code** - Code words are distinct.

**U.D. (Uniquely Decodable) Code** - Every concatenation of m code words is distinct for every finite m.

**Instantaneous Code** - A U.D. code where we can decode each code word without seeing subsequent code words.

**Examples of Block Codes (Binary Case)**

<table>
<thead>
<tr>
<th>Source Symbols</th>
<th>Code 1</th>
<th>Code 2</th>
<th>Code 3</th>
<th>Code 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>01</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>C</td>
<td>00</td>
<td>1</td>
<td>110</td>
<td>011</td>
</tr>
<tr>
<td>D</td>
<td>01</td>
<td>11</td>
<td>111</td>
<td>111</td>
</tr>
</tbody>
</table>


Instant, Instant, Instant, Not Instant

A necessary and sufficient condition for a code to be instantaneous is that no code word be a prefix of any other code word.
Coding several source symbols at a time

<table>
<thead>
<tr>
<th>Source Symbols</th>
<th>Prob.</th>
<th>U.D. Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>.5</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>.35</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>.25</td>
<td>11</td>
</tr>
</tbody>
</table>

\[ \bar{L} = 1.5 \text{ (Binary digits / source symbol)} \]

<table>
<thead>
<tr>
<th>2 Symbols</th>
<th>Prob.</th>
<th>U.D. Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>.25</td>
<td>00</td>
</tr>
<tr>
<td>AB</td>
<td>.175</td>
<td>01</td>
</tr>
<tr>
<td>AC</td>
<td>.075</td>
<td>0001</td>
</tr>
<tr>
<td>BA</td>
<td>.175</td>
<td>000</td>
</tr>
<tr>
<td>BB</td>
<td>.125</td>
<td>101</td>
</tr>
<tr>
<td>BC</td>
<td>.0525</td>
<td>1001</td>
</tr>
<tr>
<td>CA</td>
<td>.075</td>
<td>011</td>
</tr>
<tr>
<td>CB</td>
<td>.0525</td>
<td>10000</td>
</tr>
<tr>
<td>CC</td>
<td>.0225</td>
<td>10001</td>
</tr>
</tbody>
</table>

\[ \bar{L}_2 = 2.9275 \text{ (Binary digits / 2 source symbols)} \]

\[ \frac{\bar{L}_2}{2} = 1.46375 \text{ (Binary digits / source symbol)} \]

Notes

1. It is more efficient to build a code for 2 source symbols.

2. The codes given above are Huffman codes. The procedure for making Huffman codes will be described next.
Binary Huffman Codes

1. Order probabilities - Highest to Lowest
2. Add two lowest probabilities
3. Reorder probabilities
4. Break ties in any way you want

Example:

1. \( \frac{1}{3}, \frac{1}{3}, \frac{1}{2}, \frac{1}{4}, \frac{1}{16}, \frac{1}{16} \) → \( \frac{1}{4}, \frac{1}{2}, \frac{1}{3}, \frac{1}{16}, \frac{1}{16} \)

2. \( \frac{1}{3} \)
   \( \frac{1}{25} \)
   \( \frac{1}{2} \)
   \( \frac{1}{15} \)
   \( \frac{1}{1} \)

3,4. Get either

\( \frac{1}{3} \) → \( \frac{1}{25} \)
\( \frac{1}{2} \) → \( \frac{1}{15} \)
\( \frac{1}{1} \) → \( \frac{1}{25} \)

5. Assign 0 to top branch and 1 to bottom branch (or vice versa)

6. Continue until we have only one probability equal to 1.

7. Average length = \( \sum \) of probabilities of combined nodes (i.e., the circled ones)
EXAMPLE CONTINUED

In this case the two ways of breaking the tie led to two different codes with the same set of code lengths. This is not always the case—sometimes we get different codes with different code lengths. (See next example.)
1. **Binary Huffman Code** will have the shortest average length as compared with any v.d. code for that set of probabilities. (No v.d. will have a shorter average length).

2. The Huffman code is not unique. Breaking ties in different ways can result in very different codes. The average length, however, will be the same for all of these codes.

**Example**

![Huffman Code Diagram]

\( \bar{L} = 2.5 \)

\( \bar{L} = 2.5 \)
ON OPTIMALITY OF BINARY HUFFMAN CODES

THE PROOF THAT A BINARY HUFFMAN CODE IS OPTIMAL — THAT IS, HAS THE SHORTEST AVERAGE CODE WORD LENGTH AS COMPARED WITH ANY U.D. CODE FOR THAT SAME SET OF PROBABILITIES — IS OMMITTED.

However it is based on the fact that in the process of constructing the Huffman code for that set of probabilities other codes are formed for other sets of probabilities, all of which are optimal.

Example: \((p_1, p_2, p_3, p_4) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})\)

```
  0 .5
  1 .25
  10 .125
  110 .125
  111 .125
```

- Code for \((.5, .5)\)
  - .5 0
  - .5 1

- Code for \((.5, .25, .25)\)
  - .5 0
  - .25 10
  - .25 11
```
**Binary Source** \{A, B\} \((p_1, p_2) = (0.9, 0.1)\)

We now construct a series of Huffman codes, by encoding \(N\) source symbols at a time for \(N = 1, 2, 3, 4\).

**N = 1**

\[\begin{align*}
A & \rightarrow 0 \\
B & \rightarrow 1 \quad L_1 = 1
\end{align*}\]

**Code Book**

\[A \leftrightarrow 0 \quad B \leftrightarrow 1\]

**N = 2**

\((AA, AB, BA, BB) \quad (p_1, p_2, p_3, p_4) = (0.81, 0.09, 0.09, 0.01)\)

\[\begin{align*}
\text{Code Book} \\
0 & \rightarrow AA \quad 0.81 \\
11 & \rightarrow AB \quad 0.09 \\
100 & \rightarrow BA \quad 0.09 \\
101 & \rightarrow BB \quad 0.01
\end{align*}\]

\[\begin{align*}
\bar{L}_2 &= 1.29 \\
\bar{L}_2/2 &= 0.645
\end{align*}\]
\[ d=3 \{ AAA, AAB, ABA, BAA, ABB, BAA, BBA, BBB \} \]

\[ (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) \]

\[ (0.729, 0.02, 0.03, 0.02, 0.09, 0.09, 0.09, 0.01) \]

\[ L_3 = 1.27 + 0.16 + 0.10 + 0.02 + 0.01 + 0.01 = 1.59 \]

\[ \overline{L_{3/2}} = 0.533 \]
Huffman Code

4 - 4(c)  \( n = 4 \)

\( P(c) = 0.9 \)
\( P(o) = 0.1 \)

\[ \frac{L}{n} = 0.49 \]
Another Binary Coding Technique for U.D. Codes

Shannon-Fano Codes — This technique is not necessarily optimal.

1) Order probabilities in decreasing order
2) Partition into 2 sets that are as close to equally probable as possible.
3) Continue using step 2 over and over

Same example

a) \( m = 1 \)

\[
\begin{array}{c|c}
0.9 & 0 \\
0.1 & 1 \\
\end{array}
\]

b) \( m = 2 \)

\[
\begin{array}{c|c}
0.8 & 0 \\
0.2 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
0.09 & 1 & 0 \\
0.91 & 1 & 1 \\
\end{array}
\]

\[
L_2 = 1 \times 0.81 + 2 \times 0.2 \times 0.09 + 3 \times (0.09 + 0.1) = 1.29 \quad \frac{L_2}{2} = 0.645
\]

c) \( m = 3 \)

\[
\begin{array}{c|c|c}
0.729 & 0 \\
0.081 & 1 & 0 \\
0.081 & 1 & 1 \\
0.009 & 1 & 1 \end{array}
\]

\[
\begin{array}{c|c|c|c}
0.009 & 1 & 1 & 0 \\
0.001 & 1 & 1 & 1 \\
\end{array}
\]

\[
L_3 = 1 \times 0.729 + 3 \times (0.081 + 0.081 + 0.009) + 5 \times (0.009 + 0.009 + 0.009 + 0.001) = 1.729 + 0.729 + 0.140 = 2.598 \quad \frac{L_3}{3} = 0.866
\]

Same as Huffman in all of these cases

But it is not the same for \( m = 4 \)!! It is worse
$M = 4$  
Shannon - Fano
Binary Huffman Codes

(1)

\[ L = 1 \times 5 + 2 \times 2 + 3 \times 15 + 3 \times 15 \]
\[ = 5 + 4 + 45 + 45 = 94 \]

(2)

\[ L = 1 \times 4 + 2 \times 3 + 3 \times 1 + 5 \times (0.05 + 0.05 + 0.05 + 0.05) \]
\[ = 4 + 6 + 3 + 1 = 2.3 \]
\[ \bar{L} = 1 \times 0.4 + 2 \times 0.3 + 4 \times 0.1 + 5 \times (0.05 + 0.05) + 4 \times (0.05 + 0.05) \]
\[ = 0.4 + 0.6 + 0.4 + 0.5 + 0.4 = 2.3 \]

Has different length distribution but same average length
Binary Shannon-Fano Codes

1. 0.5
    0
    0.2
    1 0
    0.15
    1 1 0
    0.15
    1 1 1

2. 0.4
    0
    0.3
    1 0
    0.1
    1 1 0 0
    0.05
    1 1 0 1
    0.05
    1 1 1 0
    0.05
    1 1 1 1

\[ \bar{Z} = 1 \times 0.4 + 2 \times 0.3 + 4 \times 0.1 + 4 \times 0.05 + 4 \times 0.05 + 5 \times 0.05 + 5 \times 0.05 \]

= 0.4 + 0.6 + 0.4 + 0.2 + 0.2 + 0.25 + 0.25

\[ \bar{Z} = 2.3 \] (Same as Huffman code)