Problem Setting: Communication over a channel

- **Sender**
  - Message
  - Properties: digital or analog, contains redundancies
  - Image file, text file, real-time video, utterance

- **Channel**
  - Encoding
    - Source coding
      - Removes redundancies
    - Channel coding
      - Adds redundancies
  - Properties: digital or analog, noisy or noise-free, finite transfer rate, "channel capacity"

- **Receiver**
  - Decoding
    - Source coding
      - Decompression
    - Error correction
  - Properties: lossy vs. lossless, streaming ("progressive"), or "bulk" (seekable)

- **Channel**
  - Properties: internet (TCP/UDP), fiber optics, sound waves, storage device

Goal: Transmit message from S to R: fast + reliable
LOSSLESS COMPRESSION I: SYMBOL CODES

Problem Setting

- communicate over a noise free channel
- sender has message \( x \), wants to transmit it losslessly to receiver in as few bits as possible
- encoder:
  \[ x \mapsto C^*(x) \in \{0, 1\}^* \]
  \( \text{Kleene star} \)
  \( \text{set of all bit strings of arbitrary length} \)

- more generally:
  \[ C^*(x) \in \{0, 1\}^* \]
  \( \text{with } B \in \{2, 3, 4, \ldots \} \)  \( \text{"B-ary code"} \)
  \( \text{(commonly: } B = 2) \)

Symbol Codes

- message \( x \) is a sequence of symbols \( x_i \) from a discrete alphabet \( X \):
  \[ x = (x_1, x_2, \ldots, x_k) \equiv (x_i)_{i=1}^k \]
  \( \text{where } x_i \in X \forall i \text{ and } k \in \mathbb{N} \)
  \( \text{and } X \text{ is finite (or countably infinite)} \)
- encoder:
  \[ C^*(x) = C(x_1) \| C(x_2) \| \ldots \| C(x_k) \]
  \( \text{Concatenation} \)
  \( C \) is called the "code book"
  \( C(x) \) is called the "code word for symbol \( x \)"
  \( \text{Def: } l(x) := \text{length of } C(x) \)
  \( \text{(i.e., number of bits)} \)
Examples of Symbol Codes

1) Morse code: B=3 (dot, dash, pause)
2) UTF-8: B=2
   \( X = \{ \text{all UM CODE code points} \} \)
   \( C(x) = \text{UTF-8 representation of } x \)
   \( k(x) \in \{8, 16, 24, 32\} \) (bits)
3) "Simplified game of Monopoly":
   - throw a pair of dice several times, after each time, write down their sum as a new symbol \( x \).
   - for simplicity, let's use 3-sided dice

\[ X = \{ 2, 3, 4, 5, 6 \} \]
\[ 1+1, 1+2, 2+1 \]
\[ C(2, 6) = 10110 = C^{3}(5, 2) \]

Possible code books:
\( C^{(1)}(x) = \text{binary representation of } x \)
\( C^{(2)}(x) = \underbrace{\text{-------}}_{(x-2)} \)
\( C^{(3)}(x) = \underbrace{\text{------}}_{(x-2)} \) padded to consistent length
\( C^{(4)}(x), C^{(5)}(x): \text{see table} \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>( C^{(1)}(x) )</th>
<th>( C^{(2)}(x) )</th>
<th>( C^{(3)}(x) )</th>
<th>( C^{(4)}(x) )</th>
<th>( C^{(5)}(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>000</td>
<td>010</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>001</td>
<td>10</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>010</td>
<td>00</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>011</td>
<td>11</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>101</td>
<td>100</td>
<td>100</td>
<td>011</td>
<td>110</td>
</tr>
</tbody>
</table>

Reminder: We ultimately want to encode & decode a sequence of symbols, not just a single one (in as few bits as possible).