Day 15 - Huffman Codes and Random Cryptosystem

Recap - Decryption matrix for rectangular transposition

The following matrix was obtained from the applet for breaking rectangular transposition, using the ciphertext in the first problem of HW4:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>17</td>
<td>12</td>
<td>24</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>18</td>
<td>20</td>
<td>13</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>18</td>
<td>21</td>
<td>19</td>
<td>22</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>15</td>
<td>21</td>
<td>18</td>
<td>21</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>17</td>
<td>21</td>
<td>16</td>
<td>21</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>17</td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>24</td>
<td>13</td>
<td>23</td>
<td>20</td>
<td>17</td>
</tr>
</tbody>
</table>

Rule: If the big entry is on row $i$, col $j$, then $j$ follows i in decryption perm.

Find the decrypting and encrypting permutations:

\[(a_1, a_2, a_3, a_4, a_5, a_6, a_7) = \text{decryption perm.}\]

One row w/o big entry gives the last entry in perm.

One col w/o big entry gives the first perm entry.

\[(a_1, a_2, a_3, a_4, a_5, a_6, a_7) = \text{decryption perm.}\]

Row 6 has no big entry \(\Rightarrow a_7 = 6\).

To find $a_6$, look at Col 6 \(\Rightarrow\) for the big entry.

On 6' th col, big entry is at row 3 (row 3, col 6).

\[\Rightarrow a_6 = 3\]

To find $a_5$ \(\Rightarrow\) look for big entry on col 3

\[\Rightarrow a_5 = 4\]

Decryption perm: \((4, 5, 2, 7, 4, 3, 6)\)

Encrypting perm is the inverse of decryption perm.

\[
\text{decryption perm} \quad (1,5,2,7,4,3,6) = \begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
1 & 3 & 6 & 5 & 2 & 7 & 4
\end{pmatrix} (4)
\]

To get the inverse, simply reverse the arrows w/ (*).

\[
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
1 & 3 & 6 & 5 & 2 & 7 & 4
\end{pmatrix}
\]

\[
(1, 3, 6, 5, 2, 7, 4) = \text{encrypting perm.}
\]
Example: Suppose a certain file contains only the letter with the following frequencies:

<table>
<thead>
<tr>
<th>Letter</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Construct the comma code that enables you to compress the file so that you can store it using the least number of bits.

1. Sort frequencies in increasing order.
2. Group the smallest 2 entry, and replace w/ their sum.
3. Re-sort, repeat...

A → 0010 : need 4 bits

G → 10 : only 2 bits

D → 110

More common letter 11 less binary bits needed to encode.
In general, if \( P(n) = \frac{n_m}{N} \), then the code length for \( n_m \) is 
\[
\left\lceil \log_b \left( \frac{n_m}{\sum n_m} \right) \right\rceil
\]

<table>
<thead>
<tr>
<th>Letter</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>000</td>
<td>001</td>
<td>000</td>
<td>110</td>
<td>111</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>Bits</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

File length after encryption is 
\[
\sum P(n) \log_b \left( \frac{1}{P(n)} \right) = 0.64
\]

Average number of bits per letter is 
\[
\text{average length} = \frac{0.64}{24} = 0.0266666667
\]

Compare to the entropy of the file 
\[
\sum P(n) \log_b \left( \frac{1}{P(n)} \right) = \frac{1}{24} \log_b \left( \frac{1}{1/24} \right) + \frac{1}{24} \log_b \left( \frac{1}{1/24} \right) + \frac{1}{24} \log_b \left( \frac{1}{1/24} \right) + \frac{1}{24} \log_b \left( \frac{1}{1/24} \right) + \frac{1}{24} \log_b \left( \frac{1}{1/24} \right) + \frac{1}{24} \log_b \left( \frac{1}{1/24} \right) + \frac{1}{24} \log_b \left( \frac{1}{1/24} \right) + \frac{1}{24} \log_b \left( \frac{1}{1/24} \right) = 2.6265
\]
Theorem 1. Suppose the letter counts in the plaintext are \( n_1, n_2, \ldots, n_k \) and let \( N = n_1 + \ldots + n_k \). Then the best possible code length (in terms of bits per letter) is

\[
N - \sum_{i=1}^{k} n_i \log_2 \left( \frac{1}{p_i} \right)
\]

where \( p_i = n_i/N \) for all \( 1 \leq i \leq k \).

Fact: If \( p_i \)'s and \( q_k \)'s are probability distributions

\[
P_1 + P_2 + \ldots + P_k = \sum_{i=1}^{k} p_i = \sum_{i=1}^{k} q_i = 1
\]

then

\[
\sum_{i=1}^{k} p_i \log_2 \left( \frac{1}{P_i} \right) \leq \sum_{i=1}^{k} q_i \log_2 \left( \frac{1}{Q_i} \right)
\]
due to the fact that \( \log_2() \) is a concave gen.

Take any binary tree with leaf heights \( h_1, h_2, \ldots, h_k \).

File length after encryption

\[
= N \sum_{i=1}^{k} \frac{n_i}{N} \log_2 \left( 2^{h_i} \right)
= N \sum_{i=1}^{k} \frac{n_i}{N} \log_2 \left( \frac{1}{q_i} \right) \geq N \sum_{i=1}^{k} p_i \log_2 \left( \frac{1}{p_i} \right) = (\text{entropy of } p_i) \cdot N
\]

where \( q_i = \frac{1}{2^{h_i}} \)

So

\[
\text{File length after encryption} \geq \text{entropy of } p_i / N
\]
Theorem 2. The Huffman tree constructed from the probabilities \( p_1, p_2, \ldots, p_k \) yields an expected code length that is within 2 bit of the entropy

\[
H = \sum_{i=1}^{k} p_i \log_2 \left( \frac{1}{p_i} \right)
\]

Expected code length = \[ \sum_{i=1}^{k} p_i \cdot \frac{1}{p_i} \leq \sum_{i=1}^{k} p_i \cdot \log_2 \left( \frac{1}{p_i} \right) \leq \sum_{i=1}^{k} p_i \cdot \left( 1 + \log_2 \left( \frac{1}{p_i} \right) \right) \]

So, \[
H \leq \text{expected code length} \leq \sum_{i=1}^{k} p_i + \sum_{i=1}^{k} p_i \cdot \log_2 \left( \frac{1}{p_i} \right)
\]
Random Crypto-systems

The set up of cryptography:
- Message/plaintext space $M = \{m_1, m_2, \ldots, m_n\}$
- Key space $K = \{k_1, k_2, \ldots, k_n\}$
- Ciphertext space $C = \{c_1, c_2, \ldots, c_n\} = C$
- The encrypting function corresponding to the key $k$: $c = E_k(m)$
- The decrypting function corresponding to the key $k$: $m = D_k(c)$

In a random crypto-system:
- $M$, $m$ is the chosen message
- $K$, $k$ is the chosen key
- $C$, $c$ is the resulting ciphertext

**Remarks:**
- We choose the key $k$ independently of the message $m$.
- $C = E_k(M)$ as the ciphertext $C$ is a random variable which depends on $M$ and $k$.
- $M = D_k(C)$.
- $H(K|C)$ is the remaining uncertainty about the key after we intercept the ciphertext.
- $H(K|C) = 0$ if the ciphertext determines the key: very bad!

uncertainty about $K|C$

is the same as

the uncertainty about $K,M$
Choose a key by spinning the wheel. Example.

- Thanos → universe
- Destroy ½
- Chiller x watch
- Sun set

: M

Octopus

Salmon → Tuna → Oyster

Ciphertext

Good: if key = k2
Bad: if key = k3
Theorem 3. For 

\[ H(K|C) = H(K) + H(M) - H(C) \]
Theorem 4. For known plaintext attack:

\[ H(K, C, M) = H(K) - H(C|M) \]
Definition 1. A cryptosystem is said to attain perfect secrecy if the cipher text gives no information about the plaintext. That is, $M, C$ are random variables, namely,

$$P(M = m_i | C = c_j) = P(M = m_i) \cdot P(C = c_j)$$

for all $m_i \in M$ and $c_j \in C$. 