Chapter 5: Data Compression

Hauffman Code 1

Based on frequencies of characters.

Frequencies for the Letters A through E

<table>
<thead>
<tr>
<th>LETTER</th>
<th>FREQUENCY (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>25</td>
</tr>
<tr>
<td>B</td>
<td>15</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>20</td>
</tr>
<tr>
<td>E</td>
<td>30</td>
</tr>
</tbody>
</table>
Hauffman Code 2

(a) Initial tree

Weights: A .25, B .15, C .10, D .20, E .30

(b) After 1 merge

Weights: A .25, B .25, C .20, D .30

Weights: A .25, B .45, C .30

(c) After 2 merges

(d) After 3 merges

Weights: .55, .45

(e) After 4 merges

Hauffman Code 3

Huffman Code for the Letters A through E

<table>
<thead>
<tr>
<th>LETTER</th>
<th>CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>01</td>
</tr>
<tr>
<td>B</td>
<td>110</td>
</tr>
<tr>
<td>C</td>
<td>111</td>
</tr>
<tr>
<td>D</td>
<td>10</td>
</tr>
<tr>
<td>E</td>
<td>00</td>
</tr>
</tbody>
</table>
Hauffman-Coded Message Interpretation

Bit stream transmission

0 1 1 0 0 0 1 1 1 0 1 1 0 1 1 0 1 1 1

First character sent

A  B  E  C  A  D  B  C

Last character sent

Arithmetic Compression 1

Assigning Ranges to Letters Based on Frequency

<table>
<thead>
<tr>
<th>LETTER</th>
<th>FREQUENCY (%)</th>
<th>SUBINTERVAL [p, q]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>25</td>
<td>[0, 0.25]</td>
</tr>
<tr>
<td>B</td>
<td>15</td>
<td>[0.25, 0.40]</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>[0.4, 0.5]</td>
</tr>
<tr>
<td>D</td>
<td>20</td>
<td>[0.5, 0.7]</td>
</tr>
<tr>
<td>E</td>
<td>30</td>
<td>[0.7, 1.0]</td>
</tr>
</tbody>
</table>
Arithmetic Compression

Intervals in Arithmetic Coding

Step 1: C

Step 2: A

Step 3: B

Step 4: A

depends on $p = 0.25$ and $q = 0.5$

$x + w \times p = 0.3 + 0.6 \times 0.25 = 0.45$

$x + w \times q = 0.3 + 0.6 \times 0.5 = 0.6$

$x (0.3)$

$w = 0.9 - 0.3 = 0.6$

$y (0.9)$
Arithmetic Compression 4

Table 5.5  Steps in the Arithmetic Encoding Process

<table>
<thead>
<tr>
<th>Step</th>
<th>String</th>
<th>Next Character</th>
<th>Current Interval [p, q]</th>
<th>Interval Width (r = y − x)</th>
<th>Calculation for New x</th>
<th>Calculation for New y</th>
<th>Calculation for New y (y = x + w × p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>—</td>
<td>C</td>
<td>(0.1)</td>
<td>0.0 + 1 × 0.4 = 0.4</td>
<td>0 + 1 × 0.5 = 0.5</td>
<td>0.1 × 0.4 = 0.04</td>
<td>0.25 × 0.1 = 0.025</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>A</td>
<td>(0.4, 0.6)</td>
<td>0.1 × 0.1 × 0 = 0.4</td>
<td>0.4 + 0.1 × 0.25 = 0.425</td>
<td>0.4 + 0.025 × 0.4 = 0.44</td>
<td>0.40625 × 0.1 = 0.040625</td>
</tr>
<tr>
<td>3</td>
<td>CA</td>
<td>B</td>
<td>(0.25, 0.40)</td>
<td>0.025 × 0.4 + 0.025 × 0.25</td>
<td>0.40625 × 0.25 = 0.1025</td>
<td>0.40625 × 0.025 = 0.01025</td>
<td>0.40625 × 0.009375 × 0.25 = 0.0040375</td>
</tr>
<tr>
<td>4</td>
<td>CAB</td>
<td>A</td>
<td>(0.00625, 0.41)</td>
<td>0.00375 × 0.4 = 0.0015</td>
<td>0.40625 × 0.009375 × 0.25 = 0.0040375</td>
<td>0.40625 × 0.009375 × 0.25 = 0.0040375</td>
<td>0.40625 × 0.009375 × 0.5 = 0.0040375</td>
</tr>
<tr>
<td>5</td>
<td>CABA</td>
<td>C</td>
<td>(0.40625, 0.6071875)</td>
<td>0.0000075 × 0.4 = 0.00003</td>
<td>0.40625 × 0.009375 × 0.25 = 0.0040375</td>
<td>0.40625 × 0.009375 × 0.25 = 0.0040375</td>
<td>0.40625 × 0.009375 × 0.5 = 0.0040375</td>
</tr>
</tbody>
</table>

Arithmetic Compression 5

<table>
<thead>
<tr>
<th>Step</th>
<th>N</th>
<th>Interval [p, q]</th>
<th>Width</th>
<th>Character</th>
<th>N − p</th>
<th>Divide by Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4067</td>
<td>[0.4, 0.5]</td>
<td>0.10</td>
<td>C</td>
<td>0.0067</td>
<td>0.0067</td>
</tr>
<tr>
<td>2</td>
<td>0.067</td>
<td>[0.0, 0.25]</td>
<td>0.25</td>
<td>A</td>
<td>0.067</td>
<td>0.268</td>
</tr>
<tr>
<td>3</td>
<td>0.268</td>
<td>[0.25, 0.40]</td>
<td>0.15</td>
<td>B</td>
<td>0.018</td>
<td>0.12</td>
</tr>
<tr>
<td>4</td>
<td>0.12</td>
<td>[0.0, 0.25]</td>
<td>0.25</td>
<td>A</td>
<td>0.12</td>
<td>0.48</td>
</tr>
<tr>
<td>5</td>
<td>0.48</td>
<td>[0.4, 0.5]</td>
<td>0.10</td>
<td>C</td>
<td>0.08</td>
<td>0.8</td>
</tr>
</tbody>
</table>
Run-Length Encoding

(a) Stream prior to compression

<table>
<thead>
<tr>
<th>Bit stream</th>
<th>0 . . 010 . . 0110 . . 010 . . 0110 . . 0</th>
<th>90 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of 0s in run</td>
<td>14 9 20 30 11</td>
<td>no 0s no 0s</td>
</tr>
</tbody>
</table>

run lengths (binary) | 1110 1001 0000 1111 0101 1111 1111 0000 0000 1011 | 40 bits |
run lengths (decimal) | 14 9 0 15 5 15 15 0 0 11 |

(b) Run-length-encoded stream

Some Facsimile Compression Codes

<table>
<thead>
<tr>
<th>.</th>
<th>Number of Pixels in Run</th>
<th>Code: White Pixel Run</th>
<th>Code: Black Pixel Run</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminating Codes</td>
<td>0</td>
<td>00110101</td>
<td>0000110111</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>000111</td>
<td>010</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0111</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1000</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>00111</td>
<td>000100</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0001000</td>
<td>00001101000</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>00000011</td>
<td>00001101000</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>00101001</td>
<td>00001101000</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>01010011</td>
<td>00001010010</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>01001011</td>
<td>00000101100</td>
</tr>
<tr>
<td>Makeup Codes</td>
<td>64</td>
<td>11011</td>
<td>000001111</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>10010</td>
<td>000011001000</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>0110111</td>
<td>000010110111</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>0110101</td>
<td>000001101100</td>
</tr>
<tr>
<td></td>
<td>768</td>
<td>011001101</td>
<td>00000101100</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>0110101</td>
<td>0000011110100</td>
</tr>
<tr>
<td></td>
<td>1280</td>
<td>011010101</td>
<td>000001010010</td>
</tr>
<tr>
<td></td>
<td>1536</td>
<td>010011001</td>
<td>000001211010</td>
</tr>
</tbody>
</table>

Relative Encoding

<table>
<thead>
<tr>
<th>First frame</th>
<th>Second frame</th>
<th>Third frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 7 6 2 8 6 6 3 5 6</td>
<td>6 5 7 5 6 3 2 4 7</td>
<td>5 7 6 2 8 6 6 3 5 6</td>
</tr>
<tr>
<td>6 5 7 5 6 3 2 4 7</td>
<td>6 5 7 6 5 6 3 2 3 7</td>
<td>6 5 8 6 5 6 3 3 3 7</td>
</tr>
<tr>
<td>8 4 6 8 5 6 4 8 8 5</td>
<td>8 4 6 8 5 6 4 8 8 5</td>
<td>8 4 6 8 5 6 4 8 8 5</td>
</tr>
<tr>
<td>5 1 2 9 8 6 5 5 6 6</td>
<td>5 1 3 9 8 6 5 7 6</td>
<td>5 1 3 9 7 6 5 5 6 6</td>
</tr>
<tr>
<td>5 5 2 9 6 8 9 5 1</td>
<td>5 5 2 9 6 8 9 5 1</td>
<td>5 5 2 9 6 8 9 5 1</td>
</tr>
</tbody>
</table>

Transmitted frame contains the encoded differences between the first and second frames.

Transmitted frame contains the encoded differences between the second and third frames.

---

**Lempel-Ziv Compression 1**

```c
void compress(FILE *file) {
    // Compression algorithm
}

void decompress(FILE *file) {
    // Decompression algorithm
}
```
Lempel-Ziv Compression

<table>
<thead>
<tr>
<th>LOOP PASS</th>
<th>BUFFER</th>
<th>C</th>
<th>WHAT IS SENT</th>
<th>WHAT IS STORED IN TABLE</th>
<th>NEW BUFFER VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
<td>0 (code for A)</td>
<td>AB (code = 3)</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>A</td>
<td>1 (code for B)</td>
<td>BA (code = 4)</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>B</td>
<td>—</td>
<td>—</td>
<td>AB</td>
</tr>
<tr>
<td>4</td>
<td>AB</td>
<td>A</td>
<td>3 (code for AB)</td>
<td>ABA (code = 5)</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>B</td>
<td>—</td>
<td>—</td>
<td>AB</td>
</tr>
<tr>
<td>6</td>
<td>AB</td>
<td>C</td>
<td>3 (code for AB)</td>
<td>ABC (code = 6)</td>
<td>C</td>
</tr>
<tr>
<td>7</td>
<td>C</td>
<td>B</td>
<td>2 (code for C)</td>
<td>CB (code = 7)</td>
<td>B</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>A</td>
<td>—</td>
<td>—</td>
<td>BA</td>
</tr>
<tr>
<td>9</td>
<td>BA</td>
<td>B</td>
<td>4 (code for BA)</td>
<td>BAB (code = 8)</td>
<td>B</td>
</tr>
<tr>
<td>10</td>
<td>B</td>
<td>A</td>
<td>—</td>
<td>—</td>
<td>BA</td>
</tr>
<tr>
<td>11</td>
<td>BA</td>
<td>B</td>
<td>—</td>
<td>—</td>
<td>BAB</td>
</tr>
<tr>
<td>12</td>
<td>BAB</td>
<td>A</td>
<td>8 (code for BAB)</td>
<td>BABA (code = 9)</td>
<td>A</td>
</tr>
<tr>
<td>13</td>
<td>A</td>
<td>B</td>
<td>—</td>
<td>—</td>
<td>AB</td>
</tr>
<tr>
<td>14</td>
<td>AB</td>
<td>C</td>
<td>—</td>
<td>—</td>
<td>ABC</td>
</tr>
<tr>
<td>15</td>
<td>ABC</td>
<td>B</td>
<td>6 (code for ABC)</td>
<td>ABCB (code = 10)</td>
<td>B</td>
</tr>
<tr>
<td>16</td>
<td>B</td>
<td>A</td>
<td>—</td>
<td>—</td>
<td>BA</td>
</tr>
<tr>
<td>17</td>
<td>BA</td>
<td>B</td>
<td>—</td>
<td>—</td>
<td>BAB</td>
</tr>
<tr>
<td>18</td>
<td>BAB</td>
<td>A</td>
<td>—</td>
<td>—</td>
<td>BABA</td>
</tr>
<tr>
<td>19</td>
<td>BABA</td>
<td>B</td>
<td>9 (code for BABA)</td>
<td>BABAB (code = 11)</td>
<td>B</td>
</tr>
</tbody>
</table>

Lempel-Ziv Compression 3

Table 5.9 Table Produced by Compression Algorithm

<table>
<thead>
<tr>
<th>String</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>AB</th>
<th>BA</th>
<th>ABA</th>
<th>ABC</th>
<th>CB</th>
<th>BAB</th>
<th>BABA</th>
<th>ABCB</th>
<th>BABAB</th>
<th>BABC</th>
<th>CBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
</tbody>
</table>

Input string: ABAABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABAB

Transmitted code: ABAABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABABCBABAB

8
Lempel-Ziv Compression 4

<table>
<thead>
<tr>
<th>LOOP PASS</th>
<th>PRIOR (STRING)</th>
<th>CURRENT (STRING)</th>
<th>IS CURRENT CODE IN TABLE?</th>
<th>c</th>
<th>tempstring/Code Pair</th>
<th>WHAT IS PRINTED (current or tempstring)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0(A)</td>
<td>1(B)</td>
<td>Yes</td>
<td>B</td>
<td>AB/3</td>
<td>B (current)</td>
</tr>
<tr>
<td>2</td>
<td>1(B)</td>
<td>3(AB)</td>
<td>Yes</td>
<td>A</td>
<td>BA/4</td>
<td>AB (current)</td>
</tr>
<tr>
<td>3</td>
<td>3(AB)</td>
<td>3(AB)</td>
<td>Yes</td>
<td>A</td>
<td>ABA/5</td>
<td>AB (current)</td>
</tr>
<tr>
<td>4</td>
<td>3(AB)</td>
<td>2(C)</td>
<td>Yes</td>
<td>C</td>
<td>ABC/6</td>
<td>C (current)</td>
</tr>
<tr>
<td>5</td>
<td>2(C)</td>
<td>4(BA)</td>
<td>Yes</td>
<td>B</td>
<td>CB/7</td>
<td>BA (current)</td>
</tr>
<tr>
<td>6</td>
<td>4(BA)</td>
<td>8</td>
<td>No</td>
<td>B</td>
<td>BAB/8</td>
<td>BAB (current)</td>
</tr>
<tr>
<td>7</td>
<td>8(BAB)</td>
<td>6(ABC)</td>
<td>Yes</td>
<td>A</td>
<td>BABA/9</td>
<td>ABC (current)</td>
</tr>
<tr>
<td>8</td>
<td>6(ABC)</td>
<td>9(BABA)</td>
<td>Yes</td>
<td>B</td>
<td>ABCB/10</td>
<td>BAB (current)</td>
</tr>
<tr>
<td>9</td>
<td>9(BABA)</td>
<td>8(BAB)</td>
<td>Yes</td>
<td>B</td>
<td>BABAB/11</td>
<td>BAB (current)</td>
</tr>
<tr>
<td>10</td>
<td>8(BAB)</td>
<td>7(CB)</td>
<td>Yes</td>
<td>C</td>
<td>BABC/12</td>
<td>CB (current)</td>
</tr>
<tr>
<td>11</td>
<td>7(CB)</td>
<td>0(A)</td>
<td>Yes</td>
<td>A</td>
<td>CBA/13</td>
<td>A (current)</td>
</tr>
</tbody>
</table>

YIQ vs. RGB Representation

\[
Y = 0.30R + 0.59G + 0.11B \\
I = 0.60R - 0.28G - 0.32B \\
Q = 0.21R - 0.52G + 0.31B
\]
JPEG Phases

Image Division into 8x8 Blocks
Discrete Cosine Transform

\[ T[i][j] = 0.25C(i)C(j) \sum_{x=0}^{7} \sum_{y=0}^{7} P[x][y] \cos \left( \frac{(2x+1)i\pi}{16} \right) \cos \left( \frac{(2y+1)j\pi}{16} \right), \]

for \( i = 0, 1, 2, \ldots, 7 \) and \( j = 0, 1, 2, \ldots, 7 \) and where

\[ C(i) = \begin{cases} 1/\sqrt{2} & \text{if } i = 0 \\ 1 & \text{otherwise} \end{cases} \]

Discrete Cosine Transform: Examples

<table>
<thead>
<tr>
<th>( P ) array</th>
<th>( T ) array (values rounded to the nearest integer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 30 40 50 60 70 80 90</td>
<td>720 −182 0 −19 0 −6 0 −1</td>
</tr>
<tr>
<td>30 40 50 60 70 80 90 100</td>
<td>−182 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>40 50 60 70 80 90 100 110</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>50 60 70 80 90 100 110 120</td>
<td>−19 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>60 70 80 90 100 110 120 130</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>70 80 90 100 110 120 130 140</td>
<td>−6 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>80 90 100 110 120 130 140 150</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>90 100 110 120 130 140 150 160</td>
<td>−1 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

(a) \( P \) array representing small changes in the image

<table>
<thead>
<tr>
<th>( P ) array</th>
<th>( T ) array (values rounded to the nearest integer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 150 50 100 100 150 200 120</td>
<td>835 15 −17 59 5 −56 69 −38</td>
</tr>
<tr>
<td>200 10 110 20 200 120 30 120</td>
<td>46 −60 −36 11 14 −60 −71 110</td>
</tr>
<tr>
<td>10 200 130 30 200 20 150 50</td>
<td>−32 −9 130 105 −37 81 −17 24</td>
</tr>
<tr>
<td>100 10 90 190 120 200 10 100</td>
<td>59 −13 27 −12 30 28 −27 −48</td>
</tr>
<tr>
<td>10 200 200 120 90 190 20 200</td>
<td>50 −71 −24 −56 −40 −36 67 −189</td>
</tr>
<tr>
<td>150 120 20 200 150 70 10 100</td>
<td>−23 −18 4 54 −66 152 −61 35</td>
</tr>
<tr>
<td>200 30 150 10 10 120 190 10</td>
<td>2 13 −37 −53 15 −80 −185 −62</td>
</tr>
<tr>
<td>120 120 50 100 10 190 10 120</td>
<td>32 −14 52 −93 −210 −48 −76 80</td>
</tr>
</tbody>
</table>

(b) \( P \) array representing large changes in the image
Inverse Discrete Cosine Transform

\[ P[x][y] = 0.25 \sum_{i=0}^{7} \sum_{j=0}^{7} C(i)C(j)T[i][j] \cos \left( \frac{(2x + 1)i\pi}{16} \right) \cos \left( \frac{(2y + 1)j\pi}{16} \right) \]

Quantization 1

\[
T = \begin{bmatrix}
152 & 0 & -48 & 0 & -8 & 0 & -7 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-48 & -0 & 38 & 0 & -3 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-8 & 0 & -3 & 0 & 13 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-7 & 0 & 2 & 0 & -1 & 0 & 7 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

(5.3)

If we divided each value by 10 and rounded to the nearest integer, we would have

\[
Q = \begin{bmatrix}
15 & 0 & -5 & 0 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-5 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

(5.4)
Quantization 2

\[ T = \begin{bmatrix}
150 & 0 & -50 & 0 & -10 & 0 & -10 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-50 & 0 & 40 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-10 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-10 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix} \]

Uneven Quantization

\[ Q[i][j] = \text{Round}(T[i][j]/U[i][j]) \quad \text{for } i = 0, 1, 2, \ldots, 7 \text{ and } j = 0, 1, 2, \ldots \]
Uneven Quantization Example

\[
U = \begin{bmatrix}
1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \\
3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\
5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\
7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\
9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\
11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\
13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\
15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \\
\end{bmatrix}
\]

then the quantization would yield

\[
Q = \begin{bmatrix}
152 & 0 & -10 & 0 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-10 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

Reversing Uneven Quantization

\[
T = \begin{bmatrix}
152 & 0 & -50 & 0 & -9 & 0 & -13 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-50 & 0 & 36 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-9 & 0 & 0 & 0 & 17 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
<table>
<thead>
<tr>
<th>152</th>
<th>0</th>
<th>-10</th>
<th>0</th>
<th>-1</th>
<th>0</th>
<th>-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-10</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
MPEG Frame Sequence

Array Reduction 16x16 to 8x8
MPEG Layer 3 Audio Compression 1

- Layer 1 produces about a 4-to-1 compression ratio, and sound can be produced at a bit rate of 192 Kbps for each channel.
- Layer 2 produces about an 8-to-1 compression ratio and is designed for bit rates of 128 Kbps per channel.
- Layer 3 (MP3) compresses at about a 12-to-1 ratio and is suitable for bit rates of about 64 Kbps per channel.
MPEG Layer 3 Audio Compression 2

<table>
<thead>
<tr>
<th>Compression Technique</th>
<th>Type of Redundancy Exploited</th>
<th>How It Compresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huffman code</td>
<td>Certain characters appear more frequently than others.</td>
<td>Uses short bit patterns for more frequently used letters and longer ones for less frequently used letters.</td>
</tr>
<tr>
<td>Run-length encoding</td>
<td>Data contain long strings of the same character or bit.</td>
<td>Replaces a long run of a particular bit or character with its run length.</td>
</tr>
<tr>
<td>Facsimile compression</td>
<td>Looks for both long strings of the same bit and the frequency with which specific strings appear.</td>
<td>Divides a line of pixels into alternating runs of white and black pixels. Each run is encoded using a modified Huffman algorithm.</td>
</tr>
<tr>
<td>Relative encoding</td>
<td>Two consecutive pieces of data differ by very little.</td>
<td>Encodes small differences between consecutive frames instead of actual frames.</td>
</tr>
<tr>
<td>Lempel-Ziv encoding</td>
<td>Certain character strings appear more frequently than others.</td>
<td>Replaces repeated occurrences of strings with generated codes.</td>
</tr>
<tr>
<td>JPEG</td>
<td>Small subsets of pictures often contain little detail.</td>
<td>Compresses still images by applying discrete cosine transforms to 8 x 8 blocks of pixels, quantizing the results, and encoding the quantized frequency coefficients.</td>
</tr>
<tr>
<td>MPEG</td>
<td>Consecutive frames often contain nearly identical scenes.</td>
<td>Uses methods similar to JPEG compression but also takes advantage of redundancy between successive frames to use interframe compression by calculating differences between successive frames and using motion prediction techniques.</td>
</tr>
<tr>
<td>MP3 (MPEG layer 3 audio compression)</td>
<td>Signal components that are masked by other, more powerful signals or are outside the range of the human auditory system.</td>
<td>Uses complex psychoacoustic models and filter banks to determine which parts of an audio signal will be inaudible and seeks to remove those parts.</td>
</tr>
</tbody>
</table>