1. Huffman Coding

Give a sequence of integers as shown below

10, 12, 14, 13, 15, 17, 14, 16, 14, 16, 18, 20, 21, 23, 24, 23

Encode the sequence using DPCM and use a Huffman code on the difference samples. Determine how many bits you can save to represent the entire sequence.

Sol

From a given sequence the difference between the maximum and minimum of values is (24-10) = 14, so we need 4 bits to represent each number and 4*16 = 64 to represent the entire sequence.

We then encode the sequence using DPCM. The result is as shown below:

10,+2,+2,-1,+2,+2,-3,+2,-2,+2,+2,+2,+1,+2,+1,-1

Assume that the first number is encoded separately from the difference numbers. The difference between the maximum and minimum values is (2-(-3)) = 5 (or say there are 6 possible values). Hence, we need 4 bits to represent the first number and 3 bits to represent the reminder numbers. This requires totally 4+3*15 = 49 bits to encode the entire sequence.

We now further encode the resulted sequence with a Huffman coding technique. We first need to determine the frequency of occurrence of each value. We can see that -3, -2, -1, +1, +2 and +10 occur 1, 1, 2, 2, 9, 1 times respectively. We then construct the Huffman tree for it, as illustrated below:
We can thus encode the numbers -1, -3, -2, +10, +1 and +2 as 000, 0010, 0011, 010, 011 and 1 respectively. We now construct the table to calculate how many bits we need to represent the entire sequence.

<table>
<thead>
<tr>
<th>Number</th>
<th>frequency</th>
<th>representation</th>
<th># bits per each</th>
<th>total # bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>2</td>
<td>000</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>-3</td>
<td>1</td>
<td>0010</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>-2</td>
<td>1</td>
<td>0011</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>+10</td>
<td>1</td>
<td>010</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>+1</td>
<td>2</td>
<td>011</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>+2</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>

In conclusion, we need totally 6+4+4+3+6+9 = 32 bits using Hoffman code.

2. Harmonic Factor in Display

2.1 How does the television solve the problem of a harmonic factor? If you have to fix the fresh rate to be 29.9 frames per second, how would you do it?

The harmonic factor occurs when we display frames at 30 Hz refresh rate. This can be avoided if we display frames slightly slower or faster (29.97 Hz by the standard).

In this problem, we need to slow down the fresh rate to be 29.9. Relatively speaking, we display only 299 frames in lieu of 300 frames per 10 seconds. We can fix this by dropping one frame number every ten second. The following illustrates how we fix it.

Let consider the sequence of frame numbers that can be observed from the window of 10 seconds. (Note that the numbers with red color are frames outside the window)

Original: |0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29| 0 1 2 … 29

We remove the first frame number and shift the rest to the left.

Fixed: |1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0 1 2 ... 29 0| 1 2 … 29

Because the television displays frames in a manner with respect to synchronization information (frame number), the first frame in the sequence (with frame number 1) will be delayed for 1/30 second. Therefore, only frames (with black frame numbers) will be display in 10 seconds, thus yielding 29.9 refresh rate.