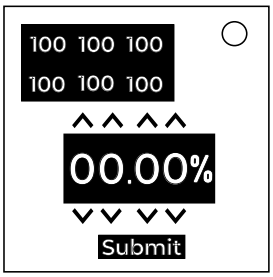
On the Subject of Huffman Coding



When every bit counts... literally.

- The module displays the probabilities (in percent) of six symbols in a discrete, memoryless system, with random values ranging from 1 to 100 inclusive.
- The module also has a percentage display, where each digit can be incremented or decremented, and a "Submit" button.
- To disarm the module, submit the coding efficiency (in percent), rounded to two decimal places, at the right time, when Huffman coding is applied to these symbols.

Note: For the purposes of calculation in **Steps 1-2,** treat the probabilities as numbers themselves and not as percentages.
For the purposes of calculation **in the succeeding steps,** treat the probabilities as percentages. Converting them to decimals would simplify calculations.

Step 1: Edgework-Based Adjustments to Initial Probabilities

Use the reading order to determine the position of the probability of a symbol, starting the count with the first and ending at the sixth.

- Take the serial number and convert all letters to their numeric position in the alphabet (A = 1, B = 2, etc.).
- Then, add the value of the (first/second/.../sixth) character of the serial number, in reading order, to the probability of the (first/second/.../sixth) symbol.

Afterwards, apply the following rules in succession. Use the modified probabilities from the previous step as the new values for the next step.

- If there are more letters than digits in the serial number, multiply the probabilities of the odd-positioned symbols by 5.
  - Otherwise, multiply the probabilities of the even-positioned symbols by 5.
- If there are more ports than batteries, add 10 to all even-positioned probabilities.
  - Otherwise, add 10 to all odd-positioned probabilities.
- If the serial number contains at least one letter in the word "HUFFMAN", add the digital root of the highest probability to all probabilities.
  - Otherwise, add the digital root of the current lowest probability to all probabilities.

Step 2: Normalize the Current Set of Probabilities

- After all applicable rules have been applied, take the sum of all current probabilities, divide each probability by this sum, and multiply each by 100, rounding each to the nearest whole number.
  - A decimal of exactly .50 rounds to the next whole number.
- After this step, the sum of the probabilities should equal 100.
- If not, add or subtract the necessary value to the probability of the sixth symbol for the sum of the probabilities to equal 100.
- Let a_n be the nth symbol in reading order.

Step 3: Calculate the Entropy (H)

- Taking the probabilities obtained in Step 2 as percentages (or their decimal equivalents), multiply each probability by the log-base-2 of its reciprocal.
  - **Do not round these values.**
- Add all of these values and round the sum to three decimal places.
- The result of this operation is the entropy, H, in bits per symbol.

Step 4: Huffman Coding

Huffman Coding Algorithm:

A. Sort all probabilities in descending order, arranging them in a vertical column.
B. Add the two lowest probabilities, keeping this value in mind. Visually indicate which two probabilities were combined with an upper branch and a lower branch.
C. Leaving the unaffected probabilities as is, include the new probability obtained from Step B, then repeat Step A. The newly created column should now have one less probability than the column to its left.
D. Draw an arrow from the symbol drawn in Step B to the current position of the summed probability in the newly created column.
E. Repeat Steps B to D until only two probabilities remain.
F. For every sum of probabilities throughout the procedure, including the two probabilities in the rightmost column, assign the upper branch to have a binary 0 and the lower branch to have a binary 1.
G. Working backwards by starting from the column to the left of the rightmost column, for every probability:

   a. If the current probability is summed to form a probability on the column to its right, **PREPEND** the binary value from that summed probability to the beginning of its current code.
   b. Otherwise, copy the binary value assigned to the same probability from the column to its right.

- The binary codes associated with each symbol will be used in the succeeding step.
- See an example of how to use the algorithm in Appendix HUFFMAN.

Step 5: Calculate the Average Codeword Length (L)

- Using the probabilities and binary codes obtained in the leftmost column of the Huffman Coding algorithm in Step 4, multiply the probability of each symbol by the number of bits its binary code contains.
  - **Do not round these values.**
- Add all of these values and round the sum to three decimal places.
- The result of this operation is the average codeword length, L, in bits.

Step 6: Calculate the Coding Efficiency

- Divide H by L, then multiply the result by 100% to determine the coding efficiency. Round this value to two decimal places.
- Input this value into the module for submission.

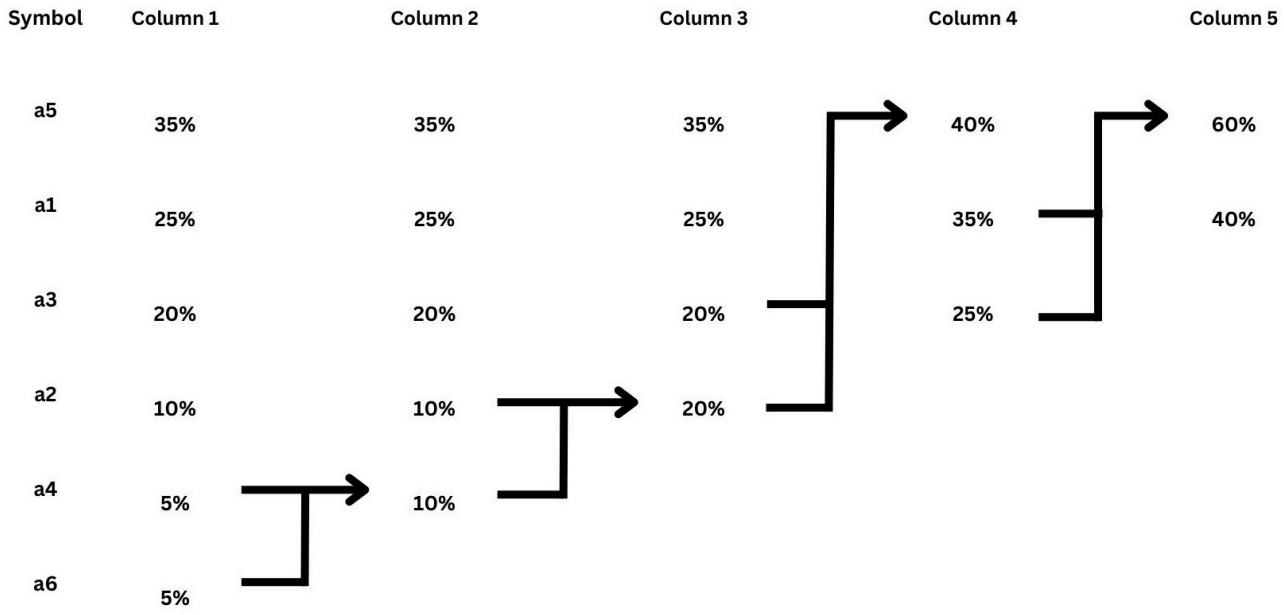
Step 7: Additional Rules

- Using the coding efficiency obtained at the end of Step 6, omit the decimal point and calculate the digital root of this value.
  - If the number of strikes is even, submit the percentage when the last digit of the timer matches the said digital root.
  - Otherwise, submit the percentage at any time.

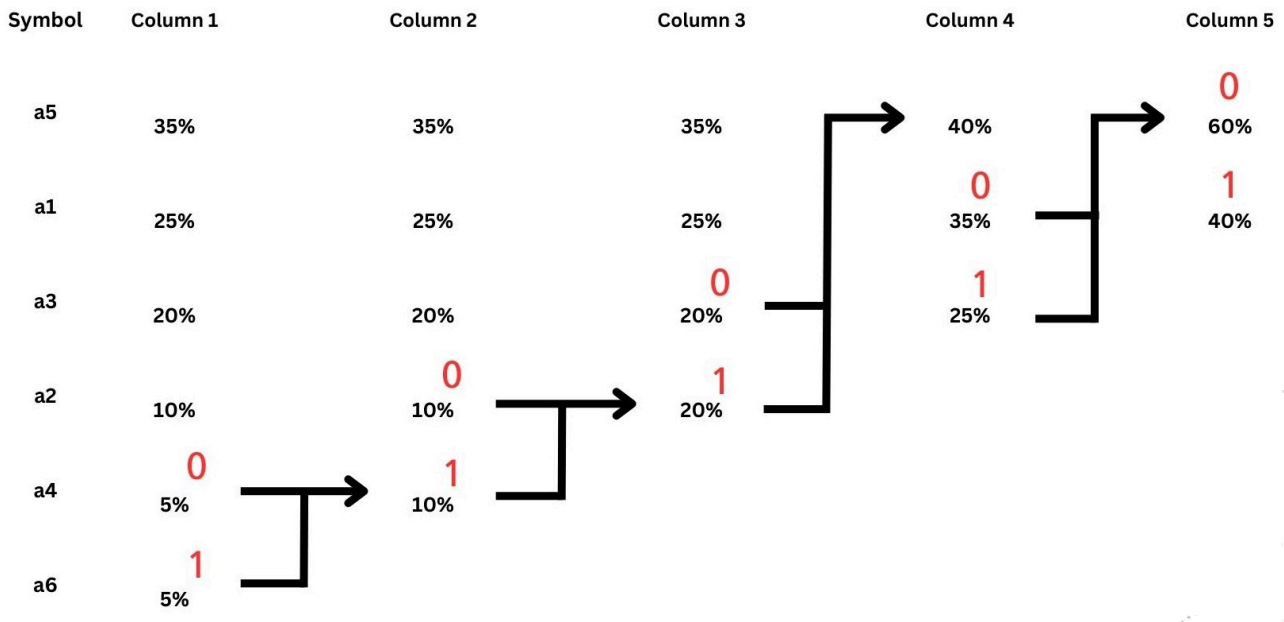
Appendix HUFFMAN

Example:
a_1: 25%, a_2: 10%, a_3: 20%, a_4: 5%, a_5: 35%, a_6: 5%.

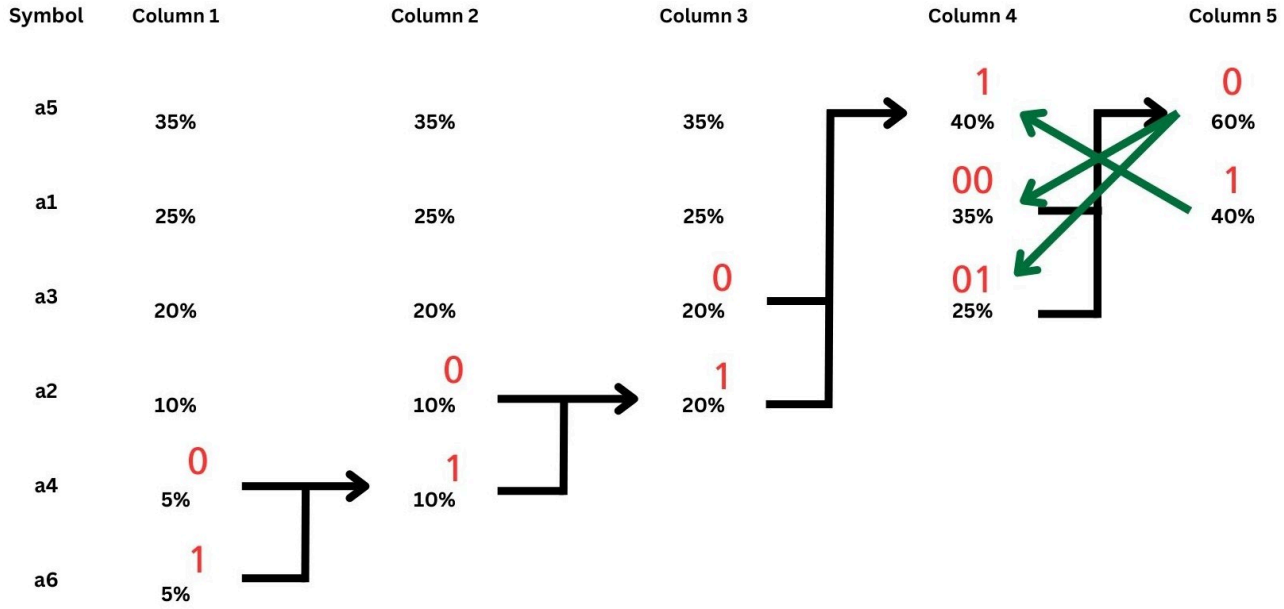
After Step E:



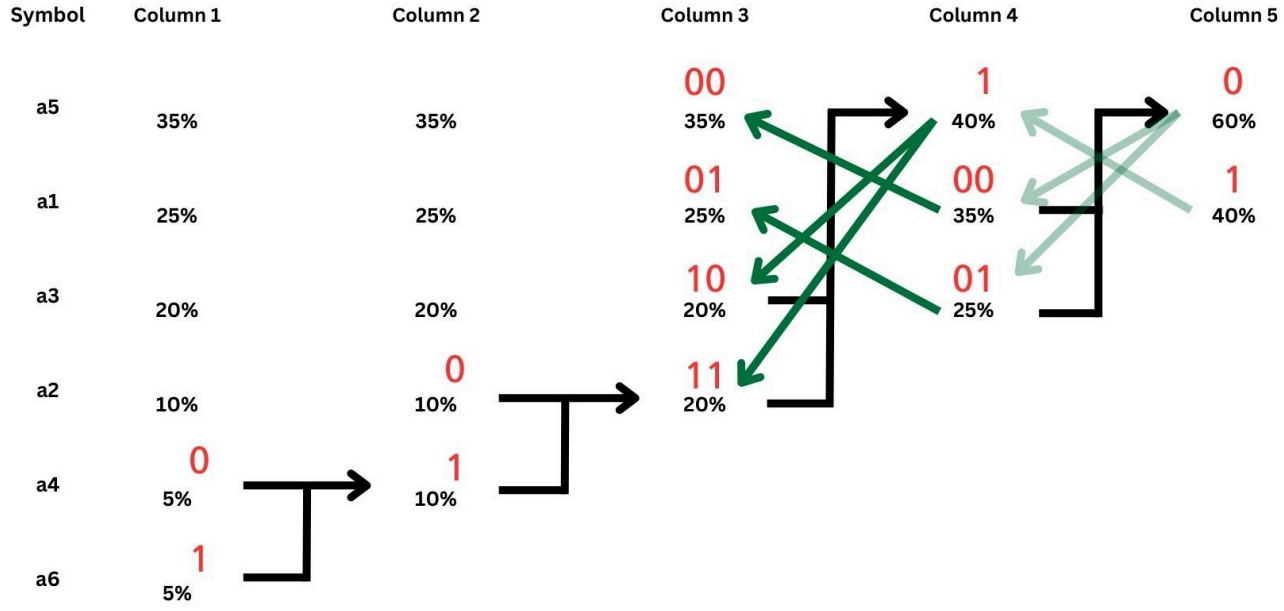
After Step F:



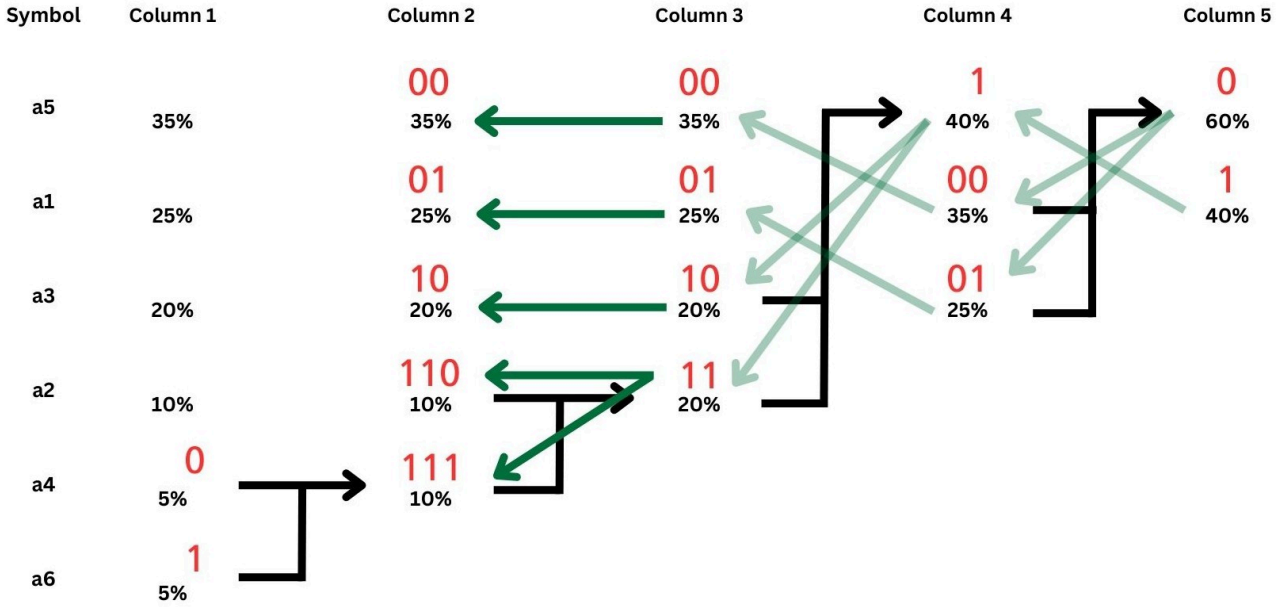
Applying Step G to Column 4:



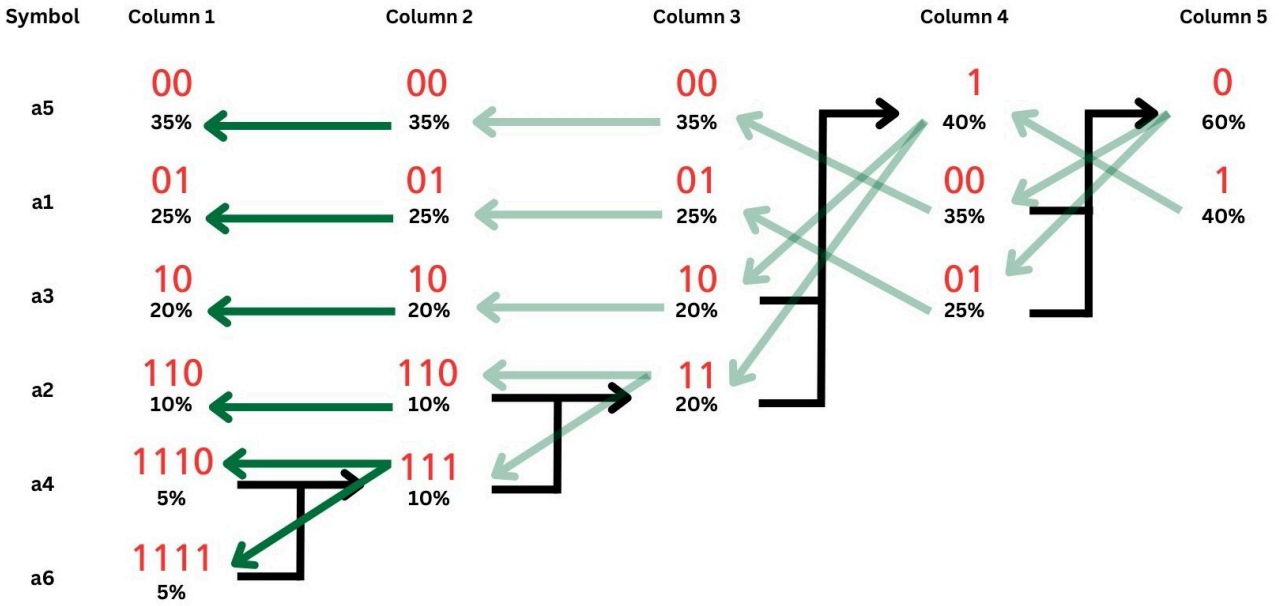
Applying Step G to Column 3:



Applying Step G to Column 2:



Finally, applying Step G to Column 1:



Thus: a_1: 01, a_2: 110, a_3: 10, a_4: 1110, a_5: 00, a_6: 1111

To check your answer by principle, the symbols with higher probabilities are assigned shorter codes, while the symbols with lower probabilities are assigned longer codes.

Another way to check your Huffman tree is that the sum of probabilities in each column should still be 100%.