Problem 5. «JPEG Encoding»

In order to decrease the readability of the exchanged messages, Alice and Bob decided to encode their messages using JPEG image compression. They write (or draw) their message in a graphics software, save it as a JPEG file and then encrypt the resulting file using some encryption algorithm.

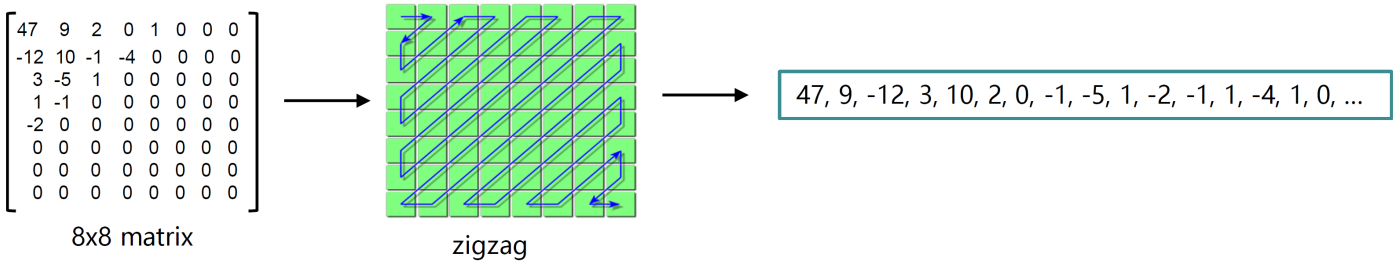
Let us describe the details of the JPEG encoding. The matrix of pixels is first divided into 8×8 matrices, and then the matrices of the type presented below are obtained from them using discrete cosine transform (DCT) and quantization. An interesting characteristic of these matrices is that most of the non-zero data is concentrated in the upper left corner of the matrix, and most of the data in the lower right corner is 0. After that, the matrix is encoded using 0's and 1's.

One example of the matrix encoding is the following algorithm:

1. First, the zigzag rule is used to convert the 8×8 matrix into a one-dimensional vector;
2. Then the Exp-Golomb code is used to encode each number in the vector. Each number (aside from 0, which is encoded as just one bit 0) is encoded by three parts:
   - *length*: a sequence of 1's corresponding to the length of the binary representation of the number, followed by 0 to mark the end of the length sequence;
   - *sign*: a bit representing the sign of the number: 0 for negative, 1 for positive number;
   - *residual*: the binary representation of the number, with the leading 1 omitted.

   For example, the number 47 is encoded as the sequence $\underbrace{1111110}_{length}\ \underbrace{1}_{sign}\ \underbrace{01111}_{residual}$;
3. All encoded sequences are then concatenated and a 6-bit sequence is added to the front. These 6 bits represent the number of non-zero elements in the encoded sequence.

An example. Let us consider how the algorithm works.



We can see that after Exp-Golomb coding, the 8×8 DCT quantized matrix above can be binarized using 91 bits (see below). Note that using the inverse process of the encoding method, we can get the original 8×8 matrix from these 91 bits.

$$\underbrace{001110}_{\#\text{ of non-zero elements}}\ \underbrace{1111110101111}_{47}\ \underbrace{111101001}_{9}\ \underbrace{111100100}_{-12}\ \underbrace{11011}_{3}\ \underbrace{111101010}_{10}\ \underbrace{11010}_{2}\ \underbrace{0}_{0}\ \underbrace{100}_{-1}\ \underbrace{1110001}_{-5}\ \underbrace{101}_{1}\ \underbrace{11000}_{-2}\ \underbrace{100}_{-1}\ \underbrace{101}_{1}\ \underbrace{1110000}_{-4}\ \underbrace{101}_{1}$$

Problem for a special prize! Your task is to design an encoding algorithm providing as short as possible output strings for the given 100 000 matrices (here is a file with matrices, and non-zero elements of each matrix are concentrated in the upper left corner). The less the sum of the lengths of the strings, the more scores you get for this problem. The encoding process must be reversible, that is, the original matrix can be obtained from the bit string using inverse coding.