QR Code Tutorial: Introduction

A QR code is a special type of barcode that can encode information like numbers, letters, and Kanji characters. This tutorial is written for programmers who want to learn how to encode a QR code. The encoding process is complicated, particularly during the step where you generate error correction code words. The tutorial will attempt to explain the entire process in simple terms. **This tutorial assumes that you have at least some basic programming knowledge.**

History and Information

The QR code (http://en.wikipedia.org/wiki/QR_Code) format was created in 1994 by Japanese company Denso-Wave (http://www.denso-wave.com/), which is a subsidiary of Toyota that manufactures auto components. The standard is defined in ISO/IEC 18004:2006 (http://www.iso.org/iso/catalogue_detail?csnumber=43655). The use of QR codes is license-free.

The smallest QR codes are 21x21 pixels, and the largest are 177x177. The sizes are called **versions**. The 21x21 pixel size is version 1, 25x25 is version 2, and so on. The 177x177 size is version 40.

In addition, QR codes include error correction: when you encode the QR code, you also create some redundant data that will help a QR reader accurately read the code even if part of it is unreadable. There are four levels of error correction that you can choose from. The lowest is L, which allows the code to be read even if 7% of it is unreadable. After that is M, which provides 15% error correction, then Q, which provides 25%, and finally H, which provides 30%.

The capacity of a given QR code depends on the version and error correction level, as well as on the type of data that you are encoding. There are four data modes that a QR code can encode: numeric, alphanumeric, binary, or Kanji. The Denso-Wave web site's list of QR versions (http://www.denso-wave.com/qrcode/vertable1-e.html) includes information about how many data bits you can encode in each version.

General Overview of Creating a QR Code

The following pages of the tutorial will explain the QR code encoding process in detail. Here is a general overview of the process that you can read before moving on to the more detailed steps.

Step 1: Data Analysis

A QR code encodes a string of text. The QR standard has four modes for encoding text: numeric, alphanumeric, byte, and Kanji. Each mode encodes the text as a string of bits (1s and 0s), but each mode uses a different method for converting the text into bits, and each encoding method is optimized to encode the data with the shortest possible string of bits. Therefore, your first step should be to perform data analysis (data-analysis) to determine whether your text can be encoded in numeric, alphanumeric, byte, or Kanji mode, then select the most optimal mode for your text.

Step 2: Data Encoding

Now that you have selected the appropriate encoding mode for your text, the next step is to encode the text. The data encoding (data-encoding) section describes this process in detail for each encoding mode. The result of this step is a string of bits that is split up into data codewords that are each 8 bits long.

Step 3: Error Correction Coding

As explained above, QR codes use error correction. This means that after you create the string of data bits that represent your text, you must then use those bits to generate error correction codewords using a process called Reed-Solomon error correction.

QR scanners read both the data codewords and the error correction codewords. By comparing the two, the scanner can determine if it read the data correctly, and it can correct errors if it did not read the data correctly. The error correction coding (error-correction-coding) section explains the process of generating error correction codewords in detail. For more information, read Wikipedia's article on Reed-Solomon error correction (http://en.wikipedia.org/wiki/Reed–Solomon_error_correction).

Step 4: Structure Final Message

The data and error correction codewords generated in the previous steps must now be arranged in the proper order. For large QR codes, the data and error correction codewords are generated in blocks, and these blocks must be interleaved according to the QR code specification. This process is explained in the structure final message (structure-final-message) section.

Step 5: Module Placement in Matrix

After generating the data codewords and error correction codewords and arranging them in the correct order, you must place the bits in the QR code matrix. The codewords are arranged in the matrix in a specific way. During this step, you will also place the patterns that are common to all QR codes, such as the boxes on the three corners. This process is explained in detail in the module placement in matrix (module-placement-matrix) section.

Step 6: Data Masking

Certain patterns in the QR code matrix can make it difficult for QR code scanners to correctly read the code. To counteract this, the QR code specification defines eight mask patterns, each of which alters the QR code according to a particular pattern. You must determine which of these mask patterns results in the QR code with the fewest undesirable traits. This is done by evaluating each masked matrix based on four penalty rules. Your final QR code must use the mask pattern that resulted in the lowest penalty score. The masking process is explained in the data masking (data-masking) section.

Step 7: Format and Version Information

The final step is to add format and (if necessary) version information to the QR code by adding pixels in particular areas of the code that were left blank in previous steps. The format pixels identify the error correction level and mask pattern being used in this QR code. The version pixels encode the size of the QR matrix and are only used in larger QR codes. For details about this final step, read the format and version information (format-version-information) section.

Page last updated 2015-05-12T22:29:22-05:00