Introduction

Versions
• 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.

Error correction
• QR codes include error correction:
  – L, which allows the code to be read even if 7% of it is unreadable.
  – M, which provides 15% error correction,
  – Q, which provides 25%,
  – H, which provides 30%.
Data modes

• There are four data modes that a QR code can encode:
  – numeric, (1 2 3 4 5…)
  – alphanumeric (.. 8 9 a b c d e …)
  – binary (0 1)
  – Japanese (kanji)

• The capacity depends on the version, error correction level, and type of encoded data

Step one : Generate a binary string

HELLO WORLD in a Version 1 QR code, with level Q error correction.

• Step 1: Encode the Mode Indicator
  – The mode indicator is a four-bit string that represents the data mode
  – HELLO WORLD, which is an alphanumeric string, our indicator is 0010

• Step 2: Encode the length of the data
  – HELLO WORLD = 11 characters (11 is binary 1011)
  – We need to encode it using a 9 bits.
  – Padding with zeros: 000001011.

• result: 0010 000001011
Generate a binary string 2

• Step 3: Encode the data
  – To encode our alphanumeric data, break up the string into pairs of characters: HE, LL, O, WO, RL, D.
  – For each pair of characters, take the ASCII value of the first character
    • Multiply it by 45.
    • Add that number to the ASCII value of the second character.
    • Convert the result into an 11-bit binary string.

<table>
<thead>
<tr>
<th>HE</th>
<th>LL</th>
<th>O</th>
<th>WO</th>
<th>RL</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>779</td>
<td>966</td>
<td>1116</td>
<td>1464</td>
<td>1236</td>
<td>11</td>
</tr>
</tbody>
</table>

Generate a binary string 3

• Step 4: Terminate the bits.
  – Make sure that it is the correct length. DATA + Error correction CWs
  – QR code version 1 with level Q error correction. For this, we must generate 104 data bits
  – If bit string is shorter than 104, then add up to four 0s to the end.
  – Since our string is 59 bits long, we add four 0s to the end.
  – Our binary string so far: 0010 00000101 | 01100001011 011110000110 1001011100 1011011100 10011010100 001101 0000

• Step 5: Delimit the string into 8-bit words.
  – Break up the string into groups of 8 bits.
  – If the last group is not 8-bits long, we pad it on the right with 0s.
  – Our binary string so far: 00100000 01011011 00001011 00110000 11010001 01110010 11011100 01001101 01000011 01000000
Generate a binary string

- Step 6: Add words at the end if the string is too short
  - If our bit string is not long enough yet, there are two special bit strings, 11101100 and 00010001, that the QR code specification requires us to put at the end of our string so far, alternating between the two until we have the required number of eight bit words (also referred to as data blocks).
  - generate 104 data bits, for a total of thirteen 8-bit words.
  - Our string so far has only 10 data blocks, so we need to add three more blocks.

Our final binary string:

- 00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 01000000 11101100 00010001 11101100

Error correction 1

QR codes use Reed-Solomon error correction.

- Step 1: Find out how many error correction code words you need to generate.
  - Version 1 with error correction level Q. This combination requires 13 blocks of data, and 13 error correction code words

- Step 2: Create your message polynomial
  - Our 13 data blocks: 00100000 01011011 00010111 01111000 11010001 01110010 11011100 01001101 01000011 01000000 11101100 00010001 11101100
  - Convert each 8-bit word from binary to decimal:
    - 32, 91, 11, 120, 209, 114, 220, 77, 67, 64, 236, 17, 236
  - Message polynomial:
    - 32x^25 + 91x^24 + 11x^23 + 120x^22 + 209x^21 + 114x^20 + 220x^19 + 77x^18 + 67x^17 + 64x^16 + 236x^15 + 17x^14 + 236x^13
  - The exponent of the first term is:
    - (number of data blocks) + (number of error correction code words) - 1
    - In our case, this is 13 + 13 - 1 = 25. So, the first term of our polynomial is 32x^25.
Error correction 2

- Step 3: Create your generator polynomial.
  - QR codes use a Galois field that has 256 elements, the numbers that we will be dealing with will always be at most 255 and at least 0.
  - The generator polynomial is always of the form \((x - \alpha)(x - \alpha^2) \ldots (x - \alpha^t)\), where \(t\) is equal to the number of required error correction code words minus 1. We need 13 error correction code words, so \(t\) in this case is 12.
  - Result of these mathematical operation: \(168x^{12} + 72x^{11} + 22x^{10} + 82x^9 + 217x^8 + 54x^7 + 156x^6 + 0x^5 + 46x^4 + 15x^3 + 180x^2 + 122x^1 + 16x^0\)
  - Coefficient after the data code words: \(32\ 91\ 11\ 120\ 209\ 114\ 220\ 77\ 67\ 64\ 236\ 17\ 236\ 168\ 72\ 22\ 82\ 217\ 54\ 156\ 0\ 46\ 15\ 180\ 122\ 16\)
  - Convert to binary.

QR Mask Patterns Explained

- There are eight mask patterns that can be used to change the outputted matrix.
- Each mask pattern changes the bits according to their coordinates in the QR matrix.
- The purpose of a mask pattern is to make the QR code easier for a QR scanner to read.
- Each mask pattern uses a formula to determine whether or not to change the color of the current bit.

Each mask generates a different QR code. After generate the eight different QR codes internally, calculate a penalty score according to the rules defined in the QR code standard. Then the result is the QR code that has the best score.
Generate the QR Code

- To generate a QR code matrix, start by making an empty matrix that is the correct size as specified in the version capacity table. *For a version 1 QR code, we need a 21x21 matrix.*
- There are three position detection patterns that are always placed in the top left, top right, and bottom right corners of the matrix.
- All QR codes must have a black pixel to the right of the top right pixel of the bottom left position detection pattern.
- All QR codes have timing patterns, which are lines of alternating black and white pixels that go between the position detection patterns.

Position adjustment pattern

- If a QR code of version 2 or larger, it is needed to add position adjustment patterns to the matrix. Lists the coordinates of where to put the position adjustment patterns.
Add Type Information

• The information about the error correction level and the mask pattern is encoded in strips to the sides of the position detection patterns.

<table>
<thead>
<tr>
<th>ECC Level</th>
<th>Mask Pattern</th>
<th>Type Information Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>0</td>
<td>11101111000100</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>11100111101001</td>
</tr>
<tr>
<td>L</td>
<td>2</td>
<td>11111010100101</td>
</tr>
<tr>
<td>L</td>
<td>3</td>
<td>11100100101111</td>
</tr>
<tr>
<td>L</td>
<td>4</td>
<td>11001000101100</td>
</tr>
<tr>
<td>L</td>
<td>5</td>
<td>110110010001001</td>
</tr>
<tr>
<td>L</td>
<td>6</td>
<td>1101100100000001</td>
</tr>
<tr>
<td>L</td>
<td>7</td>
<td>1101010111011010101</td>
</tr>
<tr>
<td>M</td>
<td>0</td>
<td>1101010000010010</td>
</tr>
<tr>
<td>M</td>
<td>1</td>
<td>11010101101011110101</td>
</tr>
<tr>
<td>M</td>
<td>2</td>
<td>11010110100101110001</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
<td>11010111011010110111</td>
</tr>
<tr>
<td>M</td>
<td>4</td>
<td>11010111110111010001</td>
</tr>
<tr>
<td>M</td>
<td>5</td>
<td>1101011111101001</td>
</tr>
<tr>
<td>M</td>
<td>6</td>
<td>110101111101011101</td>
</tr>
<tr>
<td>M</td>
<td>7</td>
<td>10000001011010000001</td>
</tr>
<tr>
<td>O</td>
<td>0</td>
<td>0101010101011111</td>
</tr>
<tr>
<td>O</td>
<td>1</td>
<td>010000011010000001</td>
</tr>
</tbody>
</table>

Table displays the type information bits that are required for the different error correction levels and mask patterns.

Add Version Information

• If QR code is version 7 or larger, add version information bits to the matrix. These are placed to the left of the top-right position detection pattern and above the bottom-left position detection pattern.

In our example, the QR code is smaller than version 7, so we don't need to add version information bits to the code.
**Add Data Bits**

**Upward Column**
- The data bits are added in a particular order. two-pixel column continues to go upward, skipping any pixels that are already set, until it reaches the top of the QR code.

**Add Data Bits 2**

**Downward Column**
- Once it reaches the top, a new two-pixel column starts, this time going downward.
- Be Sure to Skip the Horizontal Timing Pattern
Penalty rules

- Before generate the QR code, try each of the 8 mask patterns to find out which one gets the lowest penalty.

There are four penalty rules:
1. a penalty for each group of five or more same-colored pixels in a row.
2. a penalty for each 2x2 area of same-colored pixels.
3. a large penalty if there are patterns that look similar to the position detection patterns.
4. a penalty if more than half of the pixels are dark or light.

Penalty Rule 1

- If five or more of the same colored pixels are next to each other in a row or column. For the first five consecutive pixels, the penalty score is increased by 3.
- Each consecutive pixel after that adds 1 to the penalty.
Penalty Rule 2 AND 3

• 2nd rule: add 3 to the penalty for each 2x2 block of same-colored pixels there are.

• 2nd rule: looks for patterns of dark-light-dark-dark-light-dark that have four light pixels on either or both sides. It looks for any of the following three patterns.

Each time this pattern is found, you add 40 to the penalty score.

Penalty Rule 4

• The final penalty rule is based on the ratio of dark to light pixels. The closer the ratio is to 50% dark and 50% light, the better the penalty score will be for this step.
Chouse the best one

Thank you for your attention

- Reference: This presentation based on materials from the site http://www.thonky.com/qr-code-tutorial/